Universidad Nacional Mayor de San Marcos
Universidad del Perú. Decana de América
Facultad de Química e Ingeniería Química
Escuela Profesional de Ingeniería Química

Diseño de un sistema de control basado en el predictor de Smith modificado para procesos complejos de plantas industriales con modelos inestables, integradores y con tiempos de retardo dominante

TESIS

Para optar el Título Profesional de Ingeniero Químico

AUTOR

Paul Martin SAAVEDRA CARRASCO

ASESOR

Dr. Eder Clidio VICUÑA GALINDO

Lima, Perú

2026



Reconocimiento - No Comercial - Compartir Igual - Sin restricciones adicionales

https://creativecommons.org/licenses/by-nc-sa/4.0/

Usted puede distribuir, remezclar, retocar, y crear a partir del documento original de modo no comercial, siempre y cuando se dé crédito al autor del documento y se licencien las nuevas creaciones bajo las mismas condiciones. No se permite aplicar términos legales o medidas tecnológicas que restrinjan legalmente a otros a hacer cualquier cosa que permita esta licencia.

Referencia bibliográfica

Saavedra, P. (2026). *Diseño de un sistema de control basado en el predictor de Smith modificado para procesos complejos de plantas industriales con modelos inestables, integradores y con tiempos de retardo dominante*. [Tesis de pregrado, Universidad Nacional Mayor de San Marcos, Facultad de Química e Ingeniería Química, Escuela Profesional de Ingeniería Química]. Repositorio institucional Cybertesis UNMSM.

Metadatos complementarios

Datos de autor	
Nombres y apellidos	Paul Martin Saavedra Carrasco
Tipo de documento de identidad	DNI
Número de documento de identidad	70862719
URL de ORCID	https://orcid.org/0009-0000-1937-123X
Datos de asesor	
Nombres y apellidos	Eder Clidio Vicuña Galindo
Tipo de documento de identidad	DNI
Número de documento de identidad	09069542
URL de ORCID	https://orcid.org/0000-0003-4980-0490
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	Pedro José Romero Y Otiniano
Tipo de documento	DNI
Número de documento de identidad	15612279
Miembro del jurado 1	
Nombres y apellidos	Jorge Luis Cárdenas Ruiz
Tipo de documento	DNI
Número de documento de identidad	06627695
Datos de investigación	
Línea de investigación	Control de procesos

Grupo de investigación	No aplica
Agencia de financiamiento	Sin financiamiento
Ubicación geográfica de la investigación	Edificio: Facultad de Química e Ingeniería Química - Universidad Nacional Mayor de San Marcos País: Perú Departamento: Lima Provincia: Lima Distrito: Cercado de Lima Latitud: -12.05586 Longitud: -77.08466
Año o rango de años en que se realizó la investigación	Enero 2025 – agosto 2026
URL de disciplinas OCDE	Tablas OCDE Ingeniería química: https://purl.org/pe-repo/ocde/ford#2.04.01 Sistemas de automatización, Sistemas de control: https://purl.org/pe-repo/ocde/ford#2.02.03 Ingeniería de procesos: https://purl.org/pe-repo/ocde/ford#2.04.02

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
(Universidad del Perú, Decana de América)
FACULTAD DE QUÍMICA E INGENIERÍA QUÍMICA

ESCUELA PROFESIONAL DE INGENIERÍA QUÍMICA

ACTA DE SUSTENTACIÓN DE TESIS

Los suscritos Miembros del Jurado Evaluador de Sustentación de Tesis, nombrados por la Dirección de la Escuela Profesional de Ingeniería Química, bajo la Presidencia del **Ing. PEDRO JOSÉ ROMERO Y OTINIANO**, el **Ing. JORGE LUIS CÁRDENAS RUIZ** (Miembro Revisor) y el **Dr. EDER CLIDIO VICUÑA GALINDO** (Asesor), habiendo presentado para el efecto la Tesis titulada:**" DISEÑO DE UN SISTEMA DE CONTROL BASADO EN EL PREDICTOR DE SMITH MODIFICADO PARA PROCESOS COMPLEJOS DE PLANTAS INDUSTRIALES CON MODELOS INESTABLES, INTEGRADORES Y CON TIEMPOS DE RETARDO DOMINANTE"**, después de sustentar de manera presencial y **habiendo aprobado la Tesis** elaborada por el Bachiller en Ingeniería Química**: PAUL MARTIN SAAVEDRA CARRASCO**, para optar el Título Profesional de **INGENIERO QUÍMICO,** el Jurado Evaluador acuerda calificarlo con la **NOTA** de:

DIECINUEVE 19
(LETRAS) (NÚMEROS)

Lima, 28 de setiembre del 2026

Ing. PEDRO JOSÉ ROMERO Y OTINIANO
Presidente

Ing. JORGE LUIS CÁRDENAS RUIZ
Miembro

Dr. EDER CLIDIO VICUÑA GALINDO
Asesor

Dr. EDER CLIDIO VICUÑA GALINDO
Director de la E.P. de Ingeniería Química

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS • ESCUELA PROFESIONAL DE INGENIERÍA QUÍMICA • Facultad de Química e Ingeniería Química

Escaneado con CamScanner

SAN MARCOS
Universidad del Perú, Decana de América

CERTIFICADO DE SIMILITUD

Yo, **EDER CLIDIO VICUÑA GALINDO** en mi condición de asesor acreditado con el Dictamen N°000011-2026-EPIQ-FQIQ/UNMSM de la Tesis cuyo título es **"DISEÑO DE UN SISTEMA DE CONTROL BASADO EN EL PREDICTOR DE SMITH MODIFICADO PARA PROCESOS COMPLEJOS DE PLANTAS INDUSTRIALES CON MODELOS INESTABLES, INTEGRADORES Y CON TIEMPOS DE RETARDO DOMINANTE"** presentado por el bachiller: **PAUL MARTIN SAAVEDRA CARRASCO,** para optar el título Profesional de Ingeniero Químico.

CERTIFICO que se ha cumplido con lo establecido en la Directiva de Originalidad y de Similitud de Trabajos Académicos, de Investigación y Producción Intelectual. Según la revisión, análisis y evaluación mediante la plataforma PIDS, el documento evaluado cuenta con el 7% de similitud, nivel **PERMITIDO** para continuar con los trámites correspondientes y para su **publicación en el repositorio institucional.**

Se emite el presente certificado en cumplimiento de lo establecido en las normas vigentes, como uno de los requisitos para la obtención del título Profesional de Ingeniero Químico

Lima, 16 de setiembre del 2026

Firma del Asesor _____

Nombres y Apellidos del asesor Eder Clidio Vicuña Galindo

DNI: 09069542

La vigencia del presente certificado corresponde a un período de noventa (90) días desde la fecha de su emisión; siempre que no haya observaciones en el texto durante el proceso de revisión por el jurado informante o surja alguna contingencia.

Escaneado con CamScanner

AGRADECIMIENTOS

La culminación de esta tesis exigió constancia, disciplina y, sobre todo, un propósito que diera sentido al esfuerzo realizado. En *El hombre en busca de sentido*, libro que llegó a mí como regalo de una persona inesperada en mi vida, encontré una reflexión que, aun sin haberla conocido antes, parecía haberme acompañado desde siempre: «Quien tiene un porqué para vivir puede soportar casi cualquier cómo», frase de Friedrich Nietzsche retomada por Viktor Frankl. Esta idea adquirió especial significado en los momentos que exigieron mayores sacrificios, particularmente durante mi etapa universitaria y, finalmente, a lo largo del desarrollo de esta tesis, con la que cierro dicha etapa.

Agradezco a Dios y a mi familia por el apoyo espiritual, la fortaleza y la motivación que me brindaron para continuar y persistir durante los momentos más exigentes de este camino. Su presencia fue fundamental para sostenerme a lo largo de esta etapa. Expreso también mi agradecimiento a mi asesor, Dr. Eder Clidio Vicuña Galindo, por su orientación y acompañamiento; y a la Universidad Nacional Mayor de San Marcos y a la Facultad de Química e Ingeniería Química por la formación recibida y las herramientas que hicieron posible el desarrollo de esta investigación.

Finalmente, la investigación se desarrolló mediante un programa computacional implementado en MATLAB, organizado en scripts que permitieron automatizar la búsqueda y refinamiento de sintonías, validar soluciones y calcular los indicadores de rendimiento y robustez. Las herramientas de inteligencia artificial constituyeron un apoyo importante para la programación, depuración y documentación del código, facilitando la resolución de dificultades técnicas. Su utilización evidenció el valor de estas herramientas como instrumentos de asistencia al trabajo de ingeniería, manteniéndose bajo responsabilidad del autor la formulación metodológica, las decisiones de diseño, la validación de los cálculos y la interpretación de los resultados.

ÍNDICE GENERAL

AGRADECIMIENTOS ii

LISTA DE TABLAS viii

LISTA DE FIGURAS x

RESUMEN xiii

ABSTRACT xiv

1. INTRODUCCIÓN 1

1.1. Descripción de la situación problemática 3

1.2. Formulación del problema 4

1.2.1. Problema general 4

1.2.2. Problemas específicos 4

1.3. Objetivos de la investigación 5

1.3.1. Objetivo general 5

1.3.2. Objetivos específicos 5

1.4. Justificación de la investigación 5

1.4.1. Justificación teórica 6

1.4.2. Justificación práctica 7

1.4.3. Justificación metodológica 7

1.5. Limitaciones de la investigación 9

1.5.1. Lenguaje de codificación 9

2. MARCO TEÓRICO 10

2.1. Antecedentes de la investigación 10

2.1.1. Relevancia del PID y sus limitaciones 10

2.1.2. Evolución del Predictor de Smith 10

2.1.3. Estudios sobre modelos de plantas y su sintonización 11

2.2. Control por retroalimentación 15

2.2.1. Principios del control por retroalimentación 15

2.2.2. Respuesta en el dominio de la frecuencia 17

2.2.3. Características deseadas en un sistema de control 18

2.3. Modelos de plantas industriales con dinámicas complejas 23

2.3.1. Modelo con retardo puro 24

2.3.2. Procesos estables autorregulados 24

2.3.3. Procesos integradores 25
2.3.4. Procesos inestables 26
2.4. Algoritmo clásico PID 28
2.4.1. Acción proporcional, integral y derivativa 28
2.4.2. Estructuras del controlador PID 31
2.5. Predictor Smith 33
2.5.1. Predictor de Smith original (SP) 33
2.5.2. Predictor de Smith Modificado (MSP) 38
3. HIPÓTESIS Y VARIABLES DE LA INVESTIGACIÓN 42
3.1. Hipótesis 42
3.1.1. Hipótesis general 42
3.1.2. Hipótesis específicas 42
3.2. Variables 43
3.2.1. Variable independiente 43
3.2.2. Variable dependiente 43
4. METODOLOGÍA 45
4.1. Materiales 45
4.2. Métodos 45
4.3. Diseño metodológico 46
4.4. Diseño muestral 46
4.5. Técnicas de recolección de datos 47
4.6. Procedimiento metodológico 47
4.7. Aspectos éticos 50
5. DISEÑO DE LOS SISTEMAS DE CONTROL 51
5.1. Modelos de procesos 51
5.2. Diseño de los sistemas de control 52
5.2.1. Control PID 52
5.2.2. Control basado en el Predictor de Smith 53
5.2.3. Control basado en el Predictor de Smith Modificado 57
5.2.4. Ecuaciones generales del MSP 58
5.2.5. Procesos estables 59
5.2.6. Plantas integradoras 61

5.2.7. Plantas inestables 62

6. RESULTADOS Y ANÁLISIS DE RESULTADOS 65

6.1. Programa de evaluación de las estrategias de control 65

6.1.1. Definición del contexto de la búsqueda 65

6.1.2. Búsqueda de sintonías por nivel de robustez 66

6.1.3. Construcción de curvas rendimiento-robustez 67

6.1.4. Evaluación de robustez 67

6.1.5. Validación temporal y sensibilidad al retardo 68

6.1.6. Exportación y trazabilidad de resultados 69

6.2. Planta P_1: Tiempo de retardo puro ($\varphi = 0$) 70

6.2.1. Pesos base resultantes 70

6.2.2. Relación **J**, **MST** y parámetros de control óptimos 70

6.2.3. Interpretación estructural 75

6.2.4. Márgenes de estabilidad y robustez 76

6.2.5. Simulación del efecto de errores de modelado del tiempo de retardo en la respuesta controlada 79

6.2.6. Síntesis de resultados para P_1 80

6.3. Planta P_2: Dinámica balanceada 83

6.3.1. Pesos base resultantes 83

6.3.2. Relación **J**, **MST** y parámetros de control óptimos 83

6.3.3. Interpretación estructural 87

6.3.4. Márgenes de estabilidad y robustez 89

6.3.5. Simulación del efecto de errores de modelado del tiempo de retardo en la respuesta controlada 90

6.3.6. Síntesis de resultados para P_2 91

6.4. Planta P_3: Dinámica dominante 94

6.4.1. Pesos base resultantes 94

6.4.2. Relación **J**, **MST** y parámetros de control óptimos 94

6.4.3. Interpretación estructural 98

6.4.4. Márgenes de estabilidad y robustez 99

6.4.5. Simulación del efecto de errores de modelado del tiempo de retardo en la respuesta controlada 101

6.4.6. Síntesis de resultados para P_3 102

6.5. Planta P_4: Próximo al integrador 105

6.5.1. Pesos base resultantes 105

6.5.2. Relación **J**, **MST** y parámetros de control óptimos 105
6.5.3. Interpretación estructural 110
6.5.4. Márgenes de estabilidad y robustez 110
6.5.5. Simulación del efecto de errores de modelado del tiempo de retardo en la respuesta controlada 112
6.5.6. Síntesis de resultados para P_4 113
6.6. Planta P_5: Integrador 116
6.6.1. Pesos base resultantes 116
6.6.2. Relación **J**, **MST** y parámetros de control óptimos 116
6.6.3. Interpretación estructural 120
6.6.4. Márgenes de retardo y fase 121
6.6.5. Simulación del efecto de errores de modelado del tiempo de retardo en la respuesta controlada 123
6.6.6. Síntesis de resultados para P_5 124
6.7. Planta P_6: Dinámica balanceada 127
6.8. Planta P_7: Inestable con dinámica dominante 128
6.8.1. Pesos base resultantes 128
6.8.2. Relación **J**, **MST** y parámetros de control óptimos 128
6.8.3. Interpretación estructural 132
6.8.4. Márgenes de estabilidad y robustez 133
6.8.5. Simulación del efecto de errores de modelado del tiempo de retardo en la respuesta controlada 134
6.8.6. Síntesis de resultados para P_7 135
7. CONCLUSIONES Y RECOMENDACIONES 138
7.1. Conclusiones 138
7.2. Recomendaciones 139
8. REFERENCIAS 141
9. ANEXOS 144
9.1. Anexo 1. Matriz de consistencia 145
9.2. Anexo 2. Matriz de operacionalización de variables 146
9.3. Anexo 3. Matriz de hallazgos – Revisión de antecedentes 147
9.4. Anexo 4. Tablas de resultados 149
9.4.1. Planta 1 149

9.4.2. Planta 2 153
9.4.3. Planta 3 157
9.4.4. Planta 4 161
9.4.5. Planta 5 165
9.4.6. Planta 7 169

9.5. Anexo 5. Código MATLAB del Programa de evaluación de estrategias de control 173

LISTA DE TABLAS

TABLA 3-1. OPERACIONALIZACIÓN DE LA VARIABLE INDEPENDIENTE DEL ESTUDIO: ESTRATEGIA DE CONTROL EVALUADA 43

TABLA 3-2. OPERACIONALIZACIÓN DE LA VARIABLE DEPENDIENTE DEL ESTUDIO: DESEMPEÑO DEL SISTEMA DE CONTROL 44

TABLA 5-1. MODELOS DE PLANTA SELECCIONADOS PARA LA EVALUACIÓN DE LOS CONTROLADORES 51

TABLA 5-2. PARÁMETROS DEL FILTRO PREDICTIVO DE LA ESTRUCTURA MSP CALCULADOS PARA CADA MODELO DE PLANTA 64

TABLA 6-1. CRITERIOS PARA LA EVALUACIÓN DE LOS SISTEMAS DE CONTROL: DEFINICIÓN DEL CONTEXTO DE LA BÚSQUEDA 66

TABLA 6-2. CRITERIOS PARA LA EVALUACIÓN DE LOS SISTEMAS DE CONTROL: BÚSQUEDA DE SINTONÍAS 66

TABLA 6-3. CRITERIOS PARA LA EVALUACIÓN DE LOS SISTEMAS DE CONTROL: CONSTRUCCIÓN DE CURVAS J VS MST 67

TABLA 6-4. CRITERIOS PARA LA EVALUACIÓN DE LOS SISTEMAS DE CONTROL: EVALUACIÓN DE ROBUSTEZ 68

TABLA 6-5. CRITERIOS PARA LA EVALUACIÓN DE LOS SISTEMAS DE CONTROL: VALIDACIÓN TEMPORAL Y SENSIBILIDAD AL RETARDO 68

TABLA 6-6. CRITERIOS PARA LA EVALUACIÓN DE LOS SISTEMAS DE CONTROL: EXPORTACIÓN Y TRAZABILIDAD 69

TABLA 6-7. PESOS BASE PARA LA NORMALIZACIÓN DEL ÍNDICE J PARA P_1 70

TABLA 6-8. PUNTOS DE MÍNIMO J POR CONTROLADOR PARA P_1 73

TABLA 6-9. INDICADORES DE RENDIMIENTO DE LAS RESPUESTAS CONTROLADAS PARA P_1 ($MST = 1.6$) 74

TABLA 6-10. RESUMEN DE RESULTADOS P_1 81

TABLA 6-11. PESOS BASE PARA LA NORMALIZACIÓN DEL ÍNDICE J PARA P_2 83

TABLA 6-12. PUNTOS DE MÍNIMO J POR CONTROLADOR PARA P_2 84

TABLA 6-13. INDICADORES DE RENDIMIENTO DE LAS RESPUESTAS CONTROLADAS PARA P_2 ($MST = 1.6$) 87

TABLA 6-14. RESUMEN DE RESULTADOS P_2 93

TABLA 6-15. PESOS BASE PARA LA NORMALIZACIÓN DEL ÍNDICE J PARA P_3 94

TABLA 6-16. PUNTOS DE MÍNIMO J POR CONTROLADOR PARA P_3 96

TABLA 6-17. INDICADORES DE RENDIMIENTO DE LAS RESPUESTAS CONTROLADAS PARA P_3 ($MST = 1.6$) 97

TABLA 6-18. RESUMEN DE RESULTADOS P_3 104

TABLA 6-19. PESOS BASE PARA LA NORMALIZACIÓN DEL ÍNDICE J PARA P_4 105

TABLA 6-20. PUNTOS DE MÍNIMO J POR CONTROLADOR PARA P_4 107

TABLA 6-21. INDICADORES DE RENDIMIENTO DE LAS RESPUESTAS CONTROLADAS PARA P_4 ($MST = 1.6$) 109

TABLA 6-22. RESUMEN DE RESULTADOS P_4 115

TABLA 6-23. PESOS BASE PARA LA NORMALIZACIÓN DEL ÍNDICE J PARA P_5 116

TABLA 6-24. PUNTOS DE MÍNIMO J POR CONTROLADOR PARA P_5 118

TABLA 6-25. INDICADORES DE RENDIMIENTO DE LAS RESPUESTAS CONTROLADAS PARA P_5 ($MST = 1.6$) 119

TABLA 6-26. RESUMEN DE RESULTADOS P_5 126

TABLA 6-27. PESOS BASE PARA LA NORMALIZACIÓN DEL ÍNDICE J PARA P_7 128

TABLA 6-28. PUNTOS DE MÍNIMO J POR CONTROLADOR PARA P_7 130

TABLA 6-29. INDICADORES DE RENDIMIENTO DE LAS RESPUESTAS CONTROLADAS PARA P_7 ($MST = 1.6$) 132

TABLA 6-30. RESUMEN DE RESULTADOS P_7 136

TABLA 9-1. RESULTADOS DE LA BÚSQUEDA DE SINTONÍAS QUE MINIMIZAN J PARA CADA NIVEL OBJETIVO DE MST EN P_1 149

TABLA 9-2. RESULTADOS DE LA BÚSQUEDA DE SINTONÍAS QUE MINIMIZAN J PARA CADA NIVEL OBJETIVO DE MST EN P_2 153

TABLA 9-3. RESULTADOS DE LA BÚSQUEDA DE SINTONÍAS QUE MINIMIZAN J PARA CADA NIVEL OBJETIVO DE MST EN P_3 157

TABLA 9-4. RESULTADOS DE LA BÚSQUEDA DE SINTONÍAS QUE MINIMIZAN J PARA CADA NIVEL OBJETIVO DE MST EN P_4 161

TABLA 9-5. RESULTADOS DE LA BÚSQUEDA DE SINTONÍAS QUE MINIMIZAN J PARA CADA NIVEL OBJETIVO DE MST EN P_5 165

TABLA 9-6. RESULTADOS DE LA BÚSQUEDA DE SINTONÍAS QUE MINIMIZAN J PARA CADA NIVEL OBJETIVO DE MST EN P_7 169

LISTA DE FIGURAS

FIGURA 2-1. ESTRUCTURA GENERAL DE UN SISTEMA DE CONTROL POR RETROALIMENTACIÓN 15

FIGURA 2-2. ESTRUCTURA PARALELA ESTÁNDAR DEL CONTROLADOR PID 31

FIGURA 2-3. ESTRUCTURA EN CASCADA DEL CONTROLADOR PID 31

FIGURA 2-4. ESTRUCTURA EN CASCADA INTERACTUANTE DEL CONTROLADOR PID 32

FIGURA 2-5. ESTRUCTURA CLÁSICA DEL PREDICTOR DE SMITH (SP)..... 34

FIGURA 2-6. ESTRUCTURA DEL PREDICTOR DE SMITH MODIFICADO (MSP) CON FILTROS DE REFERENCIA Y PREDICTIVO 39

FIGURA 2-7. ESTRUCTURA EQUIVALENTE DEL PREDICTOR DE SMITH MODIFICADO (MSP)..... 40

FIGURA 4-1. PROCEDIMIENTO METODOLÓGICO PARA LA EVALUACIÓN DE ESTRATEGIAS DE CONTROL 49

FIGURA 5-1. ESTRUCTURA DEL PREDICTOR DE SMITH UTILIZADA EN EL DISEÑO DEL CONTROLADOR SP 54

FIGURA 5-2. ESTRUCTURA EQUIVALENTE DEL CONTROLADOR BASADO EN PREDICTOR DE SMITH 54

FIGURA 6-1. CURVA DE DESEMPEÑO-ROBUSTEZ J VS. MST PARA P_1..... 71

FIGURA 6-2. PARÁMETROS DE SINTONÍA ÓPTIMOS EN FUNCIÓN DE MST PARA P_1 72

FIGURA 6-3. RESPUESTAS CONTROLADAS ANTE CAMBIO DE REFERENCIA Y PERTURBACIONES DE ENTRADA Y SALIDA PARA P_1 ($MST = 1.6$)..... 74

FIGURA 6-4. MARGEN DE RETARDO EN FUNCIÓN DE MST PARA P_1 78

FIGURA 6-5. MARGEN DE FASE EN FUNCIÓN DE MST PARA P_1 78

FIGURA 6-6. RESPUESTAS CONTROLADAS ANTE UN ERROR DE MODELADO DEL TIEMPO DE RETARDO DE -30% PARA P_1 ($MST = 1.6$)..... 80

FIGURA 6-7. CURVA DE DESEMPEÑO-ROBUSTEZ J VS. MST PARA P_2..... 84

FIGURA 6-8. PARÁMETROS DE SINTONÍA ÓPTIMOS EN FUNCIÓN DE MST PARA P_2 85

FIGURA 6-9. RESPUESTAS CONTROLADAS ANTE CAMBIO DE REFERENCIA Y PERTURBACIONES DE ENTRADA Y SALIDA PARA P_2 ($MST = 1.6$)..... 86

FIGURA 6-10. MARGEN DE RETARDO EN FUNCIÓN DE MST PARA P_2 89

FIGURA 6-11. MARGEN DE FASE EN FUNCIÓN DE MST PARA P_2 90

FIGURA 6-12. RESPUESTAS CONTROLADAS ANTE UN ERROR DE MODELADO DEL TIEMPO DE RETARDO DE -30% PARA P_2 ($MST = 1.6$)..... 91

FIGURA 6-13. CURVA DE DESEMPEÑO-ROBUSTEZ J VS. MST PARA P_3..... 95

FIGURA 6-14. PARÁMETROS DE SINTONÍA ÓPTIMOS EN FUNCIÓN DE MST PARA P_3 97

FIGURA 6-15. RESPUESTAS CONTROLADAS ANTE CAMBIO DE REFERENCIA Y PERTURBACIONES DE ENTRADA Y SALIDA PARA P_3 ($MST = 1.6$)..... 98

FIGURA 6-16. MARGEN DE RETARDO EN FUNCIÓN DE MST PARA P_3 100

FIGURA 6-17. MARGEN DE FASE EN FUNCIÓN DE MST PARA P_3 101

FIGURA 6-18. RESPUESTAS CONTROLADAS ANTE UN ERROR DE MODELADO DEL TIEMPO DE RETARDO DE +30% PARA P_3 ($MST = 1.6$) 102

FIGURA 6-19. CURVA DE DESEMPEÑO-ROBUSTEZ J VS. MST PARA P_4 106

FIGURA 6-20. PARÁMETROS DE SINTONÍA ÓPTIMOS EN FUNCIÓN DE MST PARA P_4 108

FIGURA 6-21. RESPUESTAS CONTROLADAS ANTE CAMBIO DE REFERENCIA Y PERTURBACIONES DE ENTRADA Y SALIDA PARA P_4 ($MST = 1.6$) 109

FIGURA 6-22. MARGEN DE RETARDO EN FUNCIÓN DE MST PARA P_4 111

FIGURA 6-23. MARGEN DE FASE EN FUNCIÓN DE MST PARA P_4 112

FIGURA 6-24. RESPUESTAS CONTROLADAS ANTE UN ERROR DE MODELADO DEL TIEMPO DE RETARDO DE +30% PARA P_4 ($MST = 1.6$) 113

FIGURA 6-25. CURVA DE DESEMPEÑO-ROBUSTEZ J VS. MST PARA P_5 117

FIGURA 6-26. PARÁMETROS DE SINTONÍA ÓPTIMOS EN FUNCIÓN DE MST PARA P_5 118

FIGURA 6-27. RESPUESTAS CONTROLADAS ANTE CAMBIO DE REFERENCIA Y PERTURBACIONES DE ENTRADA Y SALIDA PARA P_5 ($MST = 1.6$) 120

FIGURA 6-28. MARGEN DE RETARDO EN FUNCIÓN DE MST PARA P_5 122

FIGURA 6-29. MARGEN DE FASE EN FUNCIÓN DE MST PARA P_5 123

FIGURA 6-30. RESPUESTAS CONTROLADAS ANTE UN ERROR DE MODELADO DEL TIEMPO DE RETARDO DE +30% PARA P_5 ($MST = 1.6$) 124

FIGURA 6-31. CURVA DE DESEMPEÑO-ROBUSTEZ J VS. MST PARA P_7 129

FIGURA 6-32. PARÁMETROS DE SINTONÍA ÓPTIMOS EN FUNCIÓN DE MST PARA P_7 130

FIGURA 6-33. RESPUESTAS CONTROLADAS ANTE CAMBIO DE REFERENCIA Y PERTURBACIONES DE ENTRADA Y SALIDA PARA P_7 ($MST = 1.6$) 131

FIGURA 6-34. MARGEN DE RETARDO EN FUNCIÓN DE MST PARA P_7 133

FIGURA 6-35. MARGEN DE FASE EN FUNCIÓN DE MST PARA P_7 134

FIGURA 6-36. RESPUESTAS CONTROLADAS ANTE UN ERROR DE MODELADO DEL TIEMPO DE RETARDO DE +30% PARA P_7 ($MST = 1.6$) 135

FIGURA 9-1. ANCHO DE BANDA EN FUNCIÓN DE MST PARA P_1 150

FIGURA 9-2. COMPARACIÓN ENTRE MST REAL Y MST OBJETIVO PARA P_1 151

FIGURA 9-3. ÍNDICES $IAEdi$ E $IAEdo$EN FUNCIÓN DE MST PARA P_1 151

FIGURA 9-4. RESPUESTAS CONTROLADAS ANTE UN ERROR DE MODELADO DEL TIEMPO DE RETARDO DE +30% PARA P_1 ($MST = 1.6$) 152

FIGURA 9-5. ANCHO DE BANDA EN FUNCIÓN DE MST PARA P_2 154

FIGURA 9-6. COMPARACIÓN ENTRE MST REAL Y MST OBJETIVO PARA P_2 155

FIGURA 9-7. ÍNDICES $IAEdi$ E $IAEdo$EN FUNCIÓN DE MST PARA P_2 155

FIGURA 9-8. RESPUESTAS CONTROLADAS ANTE UN ERROR DE MODELADO DEL TIEMPO DE RETARDO DE +30% PARA P_2 ($MST = 1.6$) 156

FIGURA 9-9. ANCHO DE BANDA EN FUNCIÓN DE MST PARA P_3 158

FIGURA 9-10. COMPARACIÓN ENTRE MST REAL Y MST OBJETIVO PARA P_3 159

FIGURA 9-11. ÍNDICES $IAEdi$ E $IAEdo$EN FUNCIÓN DE MST PARA P_3 159

FIGURA 9-12. RESPUESTAS CONTROLADAS ANTE UN ERROR DE MODELADO DEL TIEMPO DE RETARDO DE -30% PARA P_3 ($MST = 1.6$) 160

FIGURA 9-13. ANCHO DE BANDA EN FUNCIÓN DE MST PARA P_4 162

FIGURA 9-14. COMPARACIÓN ENTRE MST REAL Y MST OBJETIVO PARA P_4 163

FIGURA 9-15. ÍNDICES $IAEdi$ E $IAEdo$EN FUNCIÓN DE MST PARA P_4 163

FIGURA 9-16. RESPUESTAS CONTROLADAS ANTE UN ERROR DE MODELADO DEL TIEMPO DE RETARDO DE -30% PARA P_4 ($MST = 1.6$) 164

FIGURA 9-17. ANCHO DE BANDA EN FUNCIÓN DE MST PARA P_5 166

FIGURA 9-18. COMPARACIÓN ENTRE MST REAL Y MST OBJETIVO PARA P_5 167

FIGURA 9-19. ÍNDICES $IAEdi$ E $IAEdo$EN FUNCIÓN DE MST PARA P_5 167

FIGURA 9-20. RESPUESTAS CONTROLADAS ANTE UN ERROR DE MODELADO DEL TIEMPO DE RETARDO DE -30% PARA P_5 ($MST = 1.6$) 168

FIGURA 9-21. ANCHO DE BANDA EN FUNCIÓN DE MST PARA P_7 170

FIGURA 9-22. COMPARACIÓN ENTRE MST REAL Y MST OBJETIVO PARA P_7 171

FIGURA 9-23. ÍNDICES $IAEdi$ E $IAEdo$ EN FUNCIÓN DE MST PARA P_7 171

FIGURA 9-24. RESPUESTAS CONTROLADAS ANTE UN ERROR DE MODELADO DEL TIEMPO DE RETARDO DE +30% PARA P_7 ($MST = 1.6$) 172

RESUMEN

El control de procesos con retardo dominante, dinámica integradora o comportamiento inestable presenta dificultades de rendimiento y robustez. En este contexto, el Predictor de Smith Modificado (MSP) constituye una alternativa a los controladores PI/PID y al Predictor de Smith clásico (SP). Esta investigación diseña y evalúa sistemáticamente una estrategia basada en el MSP mediante modelos autorregulados, integradores e inestables con retardo, para determinar cuándo ofrece mejoras o menor conveniencia. Se desarrolló en MATLAB una metodología para buscar sintonías, validar soluciones y construir curvas rendimiento-robustez J vs. M_{ST}, complementadas con indicadores temporales, márgenes de estabilidad y sensibilidad a errores en el retardo. Los resultados muestran que el MSP no presenta una superioridad general frente al PID ni al SP. Su principal ventaja se observó en el proceso integrador P_5 y, en menor medida, en P_4, principalmente frente a perturbaciones de entrada. En las demás plantas, su conveniencia dependió de la dinámica del proceso, la incertidumbre en el retardo y el canal de perturbación dominante. Para P_6 no se obtuvo una familia consistente de soluciones comparables, mientras que en P_7 el PID alcanzó el mejor desempeño global y el MSP mejoró principalmente el rechazo de perturbaciones de entrada respecto al SP.

Palabras clave: Predictor de Smith Modificado, control PID, retardo dominante, robustez, M_{ST}.

ABSTRACT

Industrial process control with dominant time delay, integrating dynamics, or unstable behavior presents challenges in terms of performance and robustness. In this context, the Modified Smith Predictor (MSP) constitutes an alternative to PI/PID controllers and the classical Smith Predictor (SP). This research systematically designs and evaluates an MSP-based strategy using representative self-regulating, integrating, and unstable process models with time delay, in order to determine the conditions under which it offers improvements or lower suitability. A MATLAB-based methodology was developed to search for controller tunings, validate solutions, and construct performance-robustness curves J vs. M_{ST}, complemented by time-domain indicators, stability margins, and sensitivity to time-delay modeling errors. The results show that the MSP does not present a general superiority over PID or SP. Its main advantage was observed for the integrating process P_5 and, to a lesser extent, for P_4, mainly under input disturbances. In the remaining plants, its suitability depended on the process dynamics, uncertainty in the time delay, and the dominant disturbance channel. For P_6, no consistent family of comparable solutions was obtained, whereas for P_7 the PID achieved the best overall performance and the MSP mainly improved input-disturbance rejection with respect to the SP.

Keywords: Modified Smith Predictor, PID control, dominant time delay, robustness, M_{ST}.

1. INTRODUCCIÓN

El control de procesos constituye una herramienta fundamental para mantener variables de operación, tales como temperatura, presión, flujo, nivel o concentración, en valores de referencia definidos. Su adecuada implementación permite mejorar la seguridad operativa, la calidad del producto y la eficiencia de los sistemas industriales. Aunque algunas acciones de control pueden realizarse manualmente, la creciente complejidad de los procesos ha impulsado el uso de sistemas de control automático capaces de responder de manera más rápida y consistente ante perturbaciones o cambios en las condiciones de operación.

En los últimos años, la automatización industrial se ha relacionado estrechamente con los conceptos de Industria 4.0 e Industria 5.0. La Industria 4.0 se asocia con la digitalización de los procesos productivos mediante tecnologías como el Internet de las Cosas, la Inteligencia Artificial, el Big Data, los sistemas ciberfísicos y la computación en la nube (del Real Torres et al., 2022; Maxim et al., 2019; Moeuf et al., 2018; Schumacher et al., 2016). Por su parte, la Industria 5.0 complementa este enfoque incorporando criterios de sostenibilidad, resiliencia y colaboración entre humanos y sistemas inteligentes (del Real Torres et al., 2022; Mariátegui, 2024).

En el contexto peruano, la adopción de estas tendencias puede verse limitada por restricciones técnicas, económicas y de personal especializado, especialmente en empresas de menor escala. Las micro, pequeñas y medianas empresas representan una proporción importante del tejido empresarial nacional; sin embargo, suelen enfrentar mayores barreras para acceder a tecnologías avanzadas de automatización y control (Pérez Aviles et al., 2022; Mendoza, 2017; Huang et al., 2019). En ese sentido, el estudio de estrategias de control que mantengan una estructura relativamente accesible puede proyectarse como una contribución técnica para futuras iniciativas de modernización industrial.

Dentro del control industrial, los controladores PID continúan siendo ampliamente utilizados debido a su simplicidad, robustez y facilidad de implementación (Borase et al., 2021; Desborough & Miller, 2002; Díaz-Rodríguez et al., 2019; Reznik et al., 2000; Skogestad & Grimholt, 2018; Samad, 2017). No obstante, su desempeño puede deteriorarse cuando se aplican a procesos con tiempos de retardo dominante, dinámicas

integradoras o comportamientos inestables, lo que ha motivado el desarrollo de estrategias orientadas a la compensación del retardo, como el Predictor de Smith (Åström & Hägglund, 1995; Korupu & Muthukumarasamy, 2021; Smith, 1957).

El Predictor de Smith Modificado representa una alternativa de interés porque conserva la lógica predictiva del Predictor de Smith clásico e incorpora modificaciones orientadas a mejorar el comportamiento del lazo cerrado en procesos con retardo. En esta investigación se diseña y evalúa sistemáticamente una estrategia de control basada en dicho enfoque, comparándola con el controlador PID y el Predictor de Smith clásico en modelos de plantas estables autorreguladas, integradoras e inestables con retardo dominante.

La evaluación se desarrolla mediante indicadores de rendimiento y robustez en el dominio del tiempo y de la frecuencia, con el propósito de determinar en qué condiciones el Predictor de Smith Modificado ofrece ventajas, limitaciones o un desempeño equivalente frente a las estrategias comparadas.

1.1. Descripción de la situación problemática

A pesar del desarrollo de estrategias avanzadas de control, el controlador PID continúa siendo una de las alternativas más utilizadas en la industria debido a su simplicidad, robustez y facilidad de implementación. Sin embargo, su rendimiento puede resultar insuficiente cuando los procesos presentan tiempos de retardo dominante, dinámicas integradoras o comportamientos inestables. En estos casos, la acción correctiva del controlador puede verse afectada por el retardo del proceso, generando respuestas lentas, oscilatorias o con un compromiso poco favorable entre rendimiento y robustez.

Para abordar procesos con retardo se han propuesto estrategias de compensación como el Predictor de Smith, introducido originalmente para separar el efecto del tiempo de retardo de la dinámica principal del proceso (Smith, 1957). Esta estructura puede mejorar el desempeño del lazo cerrado cuando se dispone de un modelo adecuado; sin embargo, también puede presentar limitaciones frente a errores de modelado, incertidumbre en el tiempo de retardo o dinámicas más exigentes, como las asociadas a plantas integradoras e inestables.

Por otro lado, estrategias de control avanzado, como el control predictivo basado en modelos, ofrecen mayores capacidades de regulación y pueden aplicarse a sistemas SISO y MIMO, incorporando acciones de retroalimentación y anticipación (Normey-Rico & Camacho, 2007). No obstante, su implementación práctica puede requerir un esfuerzo considerable de modelamiento, configuración y programación, especialmente en comparación con algoritmos PID (Desborough & Miller, 2002). Esta situación evidencia la necesidad de estudiar alternativas intermedias que conserven una estructura relativamente accesible y, al mismo tiempo, permitan mejorar el comportamiento del sistema frente a procesos con retardo dominante.

En este contexto, surge el Predictor de Smith Modificado (MSP) como una alternativa orientada al control de procesos con tiempo de retardo y dinámicas complejas, tales como procesos estables autorregulados, integradores e inestables. Sin embargo, su conveniencia frente al controlador PID y al Predictor de Smith clásico no debe asumirse de manera general. Por ello, esta investigación determinará dicha conveniencia mediante una metodología comparativa basada en indicadores en el dominio del tiempo y de la frecuencia, distintos niveles de robustez y escenarios de error en el modelado del tiempo

de retardo, con el fin de identificar cuándo el MSP representa una mejora, una alternativa equivalente o una opción menos conveniente frente a las estrategias evaluadas.

1.2. Formulación del problema

1.2.1. Problema general

¿Cómo diseñar y evaluar sistemáticamente una estrategia de control basada en el Predictor de Smith Modificado que permita determinar su aplicabilidad en procesos industriales con retardo dominante, comportamiento integrador o dinámica inestable, en comparación con el controlador PID y el Predictor de Smith clásico, en términos de rendimiento y robustez?

1.2.2. Problemas específicos

a. ¿Cómo formular una estructura de control basada en el Predictor de Smith Modificado que permita abordar las dinámicas de procesos industriales con retardo dominante, integradores e inestables?
b. ¿Cómo establecer un procedimiento sistemático que permita realizar la búsqueda de sintonías, la evaluación y la comparación del rendimiento y la robustez de las estrategias de control consideradas?
c. ¿En qué medida el Predictor de Smith Modificado mejora el rendimiento ante perturbaciones frente al controlador PID y al Predictor de Smith clásico en procesos industriales con retardo dominante, comportamiento integrador o dinámica inestable?
d. ¿En qué medida el Predictor de Smith Modificado mejora la robustez frente al controlador PID y al Predictor de Smith clásico en procesos industriales con retardo dominante, comportamiento integrador o dinámica inestable?

1.3. Objetivos de la investigación

1.3.1. Objetivo general

Diseñar y evaluar sistemáticamente una estrategia de control basada en el Predictor de Smith Modificado que permita determinar su aplicabilidad en procesos industriales con retardo dominante, comportamiento integrador o dinámica inestable, en comparación con el controlador PID y el Predictor de Smith clásico, en términos de rendimiento y robustez

1.3.2. Objetivos específicos

a. Formular una estructura de control basada en el Predictor de Smith Modificado que permita abordar las dinámicas de procesos industriales con retardo dominante, integradores e inestables.
b. Establecer un procedimiento sistemático que permita realizar la búsqueda de sintonías, la evaluación y la comparación del rendimiento y la robustez de las estrategias de control consideradas.
c. Evaluar en qué medida el Predictor de Smith Modificado mejora el rendimiento ante perturbaciones frente al controlador PID y al Predictor de Smith clásico en procesos industriales con retardo dominante, integradores e inestables.
d. Evaluar en qué medida el Predictor de Smith Modificado mejora la robustez frente al controlador PID y al Predictor de Smith clásico en procesos industriales con retardo dominante, integradores e inestables.

1.4. Justificación de la investigación

El rezago tecnológico de diversas empresas peruanas, especialmente aquellas con recursos técnicos y económicos limitados, dificulta la adopción de estrategias de control capaces de abordar procesos con dinámicas complejas, tales como plantas inestables, integradoras y con tiempos de retardo dominante. Esta situación hace necesario estudiar alternativas de control que mantengan una estructura relativamente simple, pero que permitan evaluar mejoras en rendimiento y robustez frente a estrategias convencionales.

En este contexto, los controladores PI/PID continúan siendo ampliamente utilizados debido a su simplicidad, facilidad de implementación y aceptación industrial. Sin embargo, su desempeño puede verse limitado en procesos con retardo dominante o dinámicas difíciles. Por ello, resulta pertinente evaluar estructuras que incorporen

mecanismos de compensación del retardo sin aumentar excesivamente la complejidad del sistema de control.

El Predictor de Smith Modificado representa una alternativa de interés porque conserva la lógica predictiva del Predictor de Smith clásico e incorpora modificaciones orientadas a mejorar el comportamiento del sistema frente a procesos con retardo. Esta investigación se justifica porque compara dicha estrategia frente a controladores PI/PID y frente al Predictor de Smith clásico, con el propósito de identificar en qué condiciones constituye una alternativa conveniente y en qué casos las estrategias convencionales resultan suficientes o más adecuadas.

1.4.1. Justificación teórica

Esta investigación se sustenta en formulaciones reportadas en la literatura sobre control PI/PID, Predictor de Smith clásico y Predictor de Smith Modificado. Si bien estos enfoques han sido estudiados para procesos con retardo, muchas comparaciones se concentran en plantas específicas, familias reducidas de modelos o condiciones particulares de sintonía.

El aporte teórico de esta tesis consiste en evaluar si las modificaciones introducidas en el Predictor de Smith Modificado se traducen en ventajas efectivas frente al Predictor de Smith clásico y a los controladores PI/PID. Para ello, el análisis se aplica a modelos representativos de procesos con retardo dominante, incluyendo plantas autorreguladas, integradoras e inestables, tales como modelos FOPDT, IPDT y UFOPDT[1].

Además, el estudio considera distintos niveles de robustez, lo que permite analizar el comportamiento de cada estrategia más allá de una sintonía puntual. De esta manera, el desempeño del Predictor de Smith Modificado se interpreta en función de la dinámica de la planta, del nivel de robustez exigido y del criterio de evaluación empleado, evitando asumir una superioridad general frente a las estrategias comparadas.

[1] FOPDT: First order plus dead time, IPDT: Integrating plus dead time, UFOPDT: Unstable first order plus dead time.

1.4.2. Justificación práctica

Desde el punto de vista práctico, la investigación proporciona criterios para la selección de estrategias de control en procesos con tiempo de retardo y dinámicas complejas. Los modelos empleados corresponden a representaciones matemáticas simplificadas ampliamente utilizadas en control de procesos (Freitas Machado et al., 2019; Grimholt & Skogestad, 2012, 2013; Holene, 2013; Skogestad & Grimholt, 2018), mediante las cuales se aproximan comportamientos característicos de plantas industriales reales, tales como:

- Procesos autorregulados con retardo, frecuentes en sistemas térmicos, mezcla, transferencia de masa o concentración.
- Procesos integradores con retardo, presentes en sistemas de nivel, acumulación, almacenamiento o transporte de material.
- Procesos inestables con retardo, asociados a operaciones donde pequeñas desviaciones pueden amplificarse si no existe una acción de control adecuada.

La comparación entre controladores PI/PID, Predictor de Smith clásico y Predictor de Smith Modificado permite evaluar aspectos relevantes para una posible aplicación industrial, tales como el desempeño ante perturbaciones, la robustez del lazo cerrado, los márgenes de estabilidad, la sensibilidad frente a errores en el tiempo de retardo y la simplicidad relativa de la estructura de control.

Este enfoque ayuda a evitar decisiones basadas únicamente en prueba y error o en un único indicador de desempeño. Además, al considerar estrategias de estructura relativamente simple, los resultados pueden ser útiles en contextos donde no sea conveniente implementar sistemas de control de mayor complejidad técnica u operativa.

En consecuencia, la investigación contribuye a orientar la modernización de sistemas de control mediante criterios de comparación aplicables a procesos industriales con dinámicas exigentes, especialmente en entornos donde se requiere equilibrar desempeño, robustez y facilidad de implementación.

1.4.3. Justificación metodológica

La metodología propuesta permite evaluar las estrategias de control consideradas mediante un procedimiento estructurado que integra modelos matemáticos

representativos, búsqueda de sintonías y comparación mediante indicadores reconocidos en el análisis de sistemas de control. Esto se desarrolla a través de:

- **Análisis de una familia de modelos con retardo dominante**, incluyendo dinámicas autorreguladas, integradoras e inestables, con el fin de evaluar el comportamiento de las estrategias de control en distintas condiciones de planta.
- **Caracterización del compromiso entre rendimiento y robustez**, mediante la construcción de curvas J vs. M_{ST}, que permiten organizar las soluciones obtenidas y observar la relación entre el desempeño del controlador y el nivel de robustez alcanzado.
- **Evaluación comparativa en el dominio del tiempo y de la frecuencia**, considerando indicadores como el IAE ante perturbaciones, el valor real de M_{ST}, los márgenes de estabilidad, el ancho de banda y la respuesta frente a errores en el tiempo de retardo.
- **Implementación computacional del procedimiento en MATLAB**, lo que permite ejecutar de manera sistemática la búsqueda, evaluación y comparación de controladores, así como registrar casos donde no se obtienen soluciones válidas para determinadas condiciones.

El aporte metodológico consiste en establecer un procedimiento reproducible para evaluar estrategias de control en distintos escenarios de análisis. La metodología no se limita a una única planta ni a una sintonía particular, sino que combina diferentes modelos de proceso, estructuras de control y niveles de robustez y rendimiento deseados, aplicando en todos los casos la misma lógica de búsqueda, validación y comparación. De esta manera, los resultados obtenidos presentan una base más consistente para analizar el desempeño relativo de cada estrategia, y el procedimiento puede extenderse posteriormente a otros procesos, controladores o condiciones de estudio.

1.5. Limitaciones de la investigación

1.5.1. Lenguaje de codificación

La investigación se desarrolla computacionalmente en MATLAB, por lo que la ejecución directa del algoritmo queda limitada a dicho entorno. Esta condición puede restringir su portabilidad inmediata hacia otros lenguajes o plataformas de cómputo científico, como Python, sin una adaptación previa de las rutinas implementadas.

Asimismo, aunque MATLAB ofrece herramientas adecuadas para el análisis de sistemas de control, los procedimientos específicos de búsqueda de sintonías, filtrado de soluciones y comparación de resultados son implementados de forma propia. En consecuencia, la confiabilidad de los resultados depende de la correcta formulación del algoritmo, de la validación de los cálculos realizados y de la consistencia de los criterios utilizados para aceptar o descartar soluciones.

Por tanto, esta limitación no afecta el planteamiento metodológico de la investigación, pero sí condiciona su aplicación fuera del entorno MATLAB. Para trasladar el procedimiento a Python, sería necesario reprogramar el algoritmo, verificar la equivalencia de los cálculos y validar nuevamente los resultados obtenidos.

No obstante, el uso de herramientas de inteligencia artificial puede facilitar la adaptación del código a Python, según las necesidades del usuario o del entorno de aplicación. Estas herramientas pueden apoyar la traducción, reorganización y documentación del algoritmo; sin embargo, cualquier versión migrada debe ser revisada y validada para asegurar que conserve la lógica metodológica y la equivalencia de los cálculos originales.

2. MARCO TEÓRICO

2.1. Antecedentes de la investigación

En esta sección se presentan los principales antecedentes relacionados con las estrategias de control consideradas en la investigación. Primero, se revisa la relevancia del controlador PID en la práctica industrial y sus limitaciones frente a procesos con tiempo de retardo y dinámicas complejas. Luego, se describe la evolución del Predictor de Smith como estrategia de compensación de retardo y las modificaciones propuestas para ampliar su aplicabilidad. Finalmente, se analizan estudios previos sobre modelos de planta, criterios de sintonización y comparación entre controladores, con el fin de establecer el contexto técnico sobre el cual se desarrolla la presente investigación.

2.1.1. Relevancia del PID y sus limitaciones

El controlador PID sigue siendo el método de control dominante en la práctica, utilizado en el 90-95% de los sistemas implementados en plantas industriales (Borase et al., 2021; Desborough & Miller, 2002; Díaz-Rodríguez et al., 2019; Reznik et al., 2000; Skogestad & Grimholt, 2018). Su popularidad se debe a su simplicidad, robustez y la amplia disponibilidad de técnicas de sintonización. Sin embargo, el rendimiento del PID y otros controladores por retroalimentación estándar presenta limitaciones importantes al enfrentarse a procesos con largos tiempos de retardo. Estos sistemas, comunes en aplicaciones industriales donde se transporta masa, energía o información, desafían la capacidad del PID para mantenerse estable bajo condiciones de incertidumbre (Normey-Rico & Camacho, 2009).

2.1.2. Evolución del Predictor de Smith

En 1957, Smith introdujo el Predictor de Smith como un compensador de tiempos de retardo (DTC), diseñado para mejorar el desempeño de controladores clásicos como el PID en plantas con tiempos muertos significativos. Su estructura, basada en un modelo predictivo simple, fue ampliamente aceptada en la industria por su facilidad de comprensión y sintonización (Normey-Rico & Camacho, 2009). Sin embargo, el Predictor original tiene inconvenientes significativos, como su alta sensibilidad al error de estimación del tiempo de retardo, lo que afecta su desempeño en aplicaciones prácticas (Normey-Rico & Camacho, 2008a).

Durante las décadas posteriores, numerosos estudios han propuesto modificaciones al Predictor de Smith original para mejorar sus propiedades. Estas adaptaciones incluyen filtros de referencia y predictivos que permiten controlar plantas estables, inestables e integradoras, ampliando su aplicabilidad (Normey-Rico & Camacho, 2008a).

2.1.3. Estudios sobre modelos de plantas y su sintonización

En el 2009, Lennartson & Kristiansson evaluaron el desempeño del Predictor de Smith combinado con controladores PI y PID en un modelo de tercer orden con tiempo de retardo. Los resultados indicaron que el Predictor de Smith proporciona mejoras significativas (15-16%) en el rendimiento para retardos pequeños y medios cuando se combina con un controlador PID. Sin embargo, también se observó que estas mejoras comprometen la estabilidad en condiciones de alta ganancia, especialmente para grandes retardos.

Ese mismo año, Normey-Rico & Camacho (2008a) publicaron *Dead-time compensators: A survey*, una revisión exhaustiva sobre las principales modificaciones del Predictor de Smith documentadas en la literatura. En su estudio, analizaron diversas estructuras y estrategias de sintonización aplicadas al control de procesos estables, integradores e inestables con tiempo de retardo.

Los resultados mostraron que el Predictor de Smith original puede superar el desempeño de un controlador PID, especialmente cuando el tiempo de retardo es dominante y está bien caracterizado, logrando un equilibrio adecuado entre estabilidad y rendimiento mediante un ajuste preciso del controlador principal y el predictor. Sin embargo, también se evidenció que las modificaciones del Predictor de Smith mejoran significativamente su capacidad para rechazar perturbaciones y seguir referencias en plantas estables. Además, estas adaptaciones incorporan estrategias específicas que fortalecen la robustez y estabilidad en sistemas inestables e integradores, algo que el Predictor de Smith en su versión original no puede lograr.

El Predictor de Smith original presenta limitaciones que impiden su implementación en sistemas reales. A pesar de su estructura conceptualmente sencilla, sus problemas de estabilidad representan una desventaja significativa.

En respuesta a estas deficiencias, Normey-Rico & Camacho (2009) en *Unified approach for robust dead-time compensator design*, propusieron un procedimiento unificado de

sintonización para una versión modificada del Predictor de Smith, incorporando filtros de referencia y predictivos. Su enfoque estaba dirigido al control de procesos con tiempo de retardo en plantas estables, inestables e integradoras.

Esta estrategia permitía seleccionar los filtros adecuados y ajustar sus parámetros de control mediante el análisis de polos inestables o de respuesta lenta, con el objetivo de eliminarlos en función de las características de cada planta. Además, simplificaba la sintonización al reducir el número de parámetros a solo dos, facilitando la incorporación de polos en lazo cerrado y logrando un equilibrio eficiente entre rendimiento y estabilidad.

En 2013, Holene publicó *Performance and Robustness of Smith Predictor Control and Comparison with PID Control*, un estudio en el que se evaluaron el rendimiento y la robustez del Predictor de Smith original en comparación con los controladores PI y PID con sintonización óptima. La investigación se llevó a cabo en cinco modelos de primer orden con tiempo de retardo y nueve modelos de segundo orden con tiempo de retardo.

El rendimiento se cuantificó mediante un promedio ponderado del índice IAE, considerando variaciones en las perturbaciones de entrada y salida del sistema, mientras que la robustez se evaluó en función del pico máximo de sensibilidad (M_{ST}). Los resultados indicaron que el Predictor de Smith ofrecía solo mejoras marginales en el rendimiento en comparación con los controladores PI y PID óptimos en procesos de segundo orden. Para procesos de primer orden, los controladores PI y PID óptimos demostraron un rendimiento superior al del Predictor de Smith.

Además, se observó que, ante variaciones en el tiempo de retardo del proceso, incluso los Predictores de Smith con valores modestos de M_{ST} tendían a volverse inestables. En consecuencia, el estudio concluyó que no existen ventajas significativas en utilizar la estructura del Predictor de Smith junto con un controlador PI o PID sobre la implementación de un controlador PI o PID óptimamente sintonizado.

Holene (2013) usó métricas de rendimiento, estabilidad y robustez que reflejaban adecuadamente la compensación entre estas tres características, permitiendo una evaluación rigurosa de los controladores y concluyendo que la estructura original del Predictor de Smith no ofrecía ventajas significativas. Posteriormente, Skogestad & Grimholt (2018), en su publicación *Should we forget the Smith Predictor?*, profundizaron en esta línea de investigación y demostraron que el desempeño del Predictor de Smith

original es considerablemente inferior al del controlador PID cuando se aplica a plantas del tipo FOPDT. Para su análisis, emplearon los mismos criterios de evaluación utilizados por Holene (2013) y encontraron que, en la mayoría de los casos, un controlador PID es sustancialmente más eficiente que un controlador PI con Predictor de Smith.

Además, evidenciaron que, si bien es posible mejorar el rendimiento del Predictor de Smith, esto ocurre a expensas de un incremento en la sensibilidad del sistema. La sensibilidad fue medida a través del margen de retardo, definido como la relación entre el margen de fase y la frecuencia crítica. Los resultados indicaron que la implementación del Predictor de Smith puede inducir inestabilidad en el sistema ante pequeñas incertidumbres en la estimación del tiempo de retardo.

A pesar de que los estudios de Holene (2013) y Skogestad & Grimholt (2018) rechazan el uso del Predictor de Smith sobre controladores PI y PID bien sintonizados, sus análisis no incluyen la evaluación del Predictor de Smith en su versión modificada. Esta variante no solo mejora el rendimiento, sino que también permite el control de plantas inestables e integradoras, lo que amplía significativamente su aplicabilidad en procesos industriales más complejos.

En 2020, da Silva et al. publicaron *Controlling industrial dead-time systems: When to use a PID or an advanced controller*, donde analizaron diversas estrategias de control, incluyendo el PID, el Predictor de Smith Filtrado (FSP) (Normey-Rico & Guzmán, 2013) y el Control de Modelo Predictivo (MPC), aplicados a procesos FOPDT, IPDT y UFOPDT. El principal aporte de este estudio fue la propuesta de un método práctico para determinar si se debe emplear un controlador PID clásico o una estrategia de control avanzada en problemas típicos de control de procesos.

El método consistió en evaluar el desempeño de cada algoritmo en función de la variabilidad de la robustez en procesos SISO con tiempo de retardo. Para ello, el rendimiento se cuantificó mediante el promedio de los valores del indicador IAE, considerando tanto cambios en la referencia como el rechazo a perturbaciones. La robustez, por su parte, se midió utilizando un indicador en el dominio de la frecuencia junto con el margen de retardo.

Los resultados mostraron que el factor determinante en la elección de la estrategia de control es la calidad del modelo del proceso, es decir, el grado de error o incertidumbre existente entre el modelo nominal y la planta real. Se demostró que, en la mayoría de los

casos, los controladores PID y FSP presentan un desempeño equivalente cuando se utiliza una sintonización robusta. Sin embargo, cuando se dispone de un modelo de proceso preciso y se permite un ajuste rápido con márgenes de robustez reducidos, el FSP logra una mejora significativa en el rendimiento en comparación con el PID.

Para entornos industriales donde se requiere una mayor robustez, es decir, cuando el modelo del proceso presenta una mayor incertidumbre, se observó que un controlador PID bien sintonizado con un mecanismo anti-windup puede proporcionar resultados comparables o incluso superiores a los de estrategias de control avanzadas como el FSP o el MPC.

Finalmente, Freitas Machado et al. (2019) publicaron *¿Por qué no debemos olvidar el predictor de Smith?*, en respuesta a los cuestionamientos de Skogestad & Grimholt (2018). En su estudio, compararon una variante del Predictor de Smith denominada Compensador de Tiempo de retardo Simplificado (SDTC) con controladores PI y PID, evaluando su desempeño en términos de rendimiento y robustez.

Para garantizar una comparación rigurosa con los resultados de Skogestad & Grimholt (2018), se emplearon los mismos criterios de evaluación de rendimiento, estabilidad y robustez, así como los mismos modelos de plantas FOPDT, que incluían atraso de transporte puro, dinámica equilibrada, dinámica lenta y comportamiento cercano a un integrador. Además, se utilizó un enfoque más conservador para medir la robustez, basado en las curvas de máxima incertidumbre robusta del tiempo en función del índice M_{ST}.

Los resultados indicaron que el SDTC ofrece un rendimiento comparable o superior al de los controladores PI/PID en la mayoría de los casos. Asimismo, se demostró que la robustez frente a las incertidumbres en el tiempo de retardo puede evaluarse con mayor precisión mediante modelos de incertidumbre multiplicativa y curvas de índice de robustez.

Siguiendo esta línea de investigación, el presente trabajo propone una estructura modificada del Predictor de Smith (MSP) con el objetivo de demostrar sus ventajas sobre el controlador PID clásico en términos de rendimiento y estabilidad. A diferencia de los estudios previos Skogestad & Grimholt (2018) y Freitas Machado et al. (2019), que se centraron en plantas estables, este estudio también analizará su desempeño en modelos inestables e integradores, ampliando así su aplicabilidad en procesos industriales con dinámicas más complejas.

2.2. Control por retroalimentación

2.2.1. Principios del control por retroalimentación

El diagrama de bloques de la **Figura 2-1** se conoce como control por realimentación, donde C y P son las funciones de transferencia del controlador y del proceso, respectivamente. La entrada al controlador es la señal de error o error de control, $e = r - y_m$, donde $y_m = y + n$, r es el setpoint o punto de consigna, y es la salida del proceso o variable de control y n es la señal de ruido. G_i y G_o son las funciones de transferencia de las perturbaciones de entrada y salida, d_i y d_o, respectivamente. La salida del controlador es k y la entrada al proceso es u. Las funciones de transferencia están expresadas en el dominio de Laplace, pero la variable s ha sido omitida por conveniencia.

El diseño de los sistemas de control del presente estudio contemplará las siguientes consideraciones/exclusiones:

- No se considerarán efectos de ruido n.
- Las funciones de transferencia de la perturbación de entrada y salida, G_i y G_o, serán iguales a la unidad.

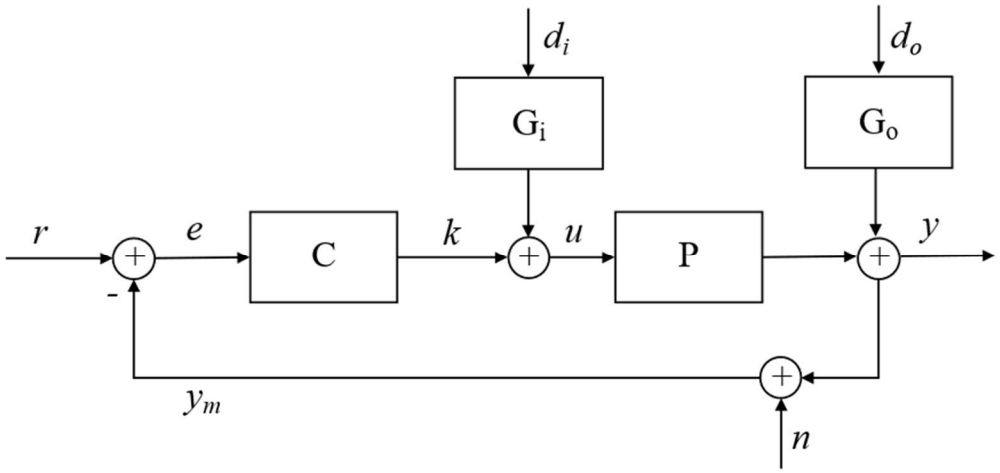

Figura 2-1. Estructura general de un sistema de control por retroalimentación

En este tipo de control, la información de salida del proceso, que contiene cambios en los valores de referencia o efectos de perturbaciones, se comunica o retroalimenta al controlador y este corrige la señal de entrada a la planta.

Según Díaz-Rodríguez et al. (2019), la complejidad en el diseño de los sistemas de control radica en dotar al controlador de las capacidades necesarias para que las señales de salida se ajusten a valores de referencia predefinidos, mantengan estos valores frente a perturbaciones desconocidas y conserven ambas propiedades pese a incertidumbres y variaciones en la dinámica de la planta. Estos requisitos corresponden a tres principios fundamentales: seguimiento de referencia, rechazo de perturbaciones y robustez del sistema (Bhattacharyya et al., 2009).

Como señalan Skogestad & Postlethwaite (2005), las principales razones para preferir un sistema de control feedback sobre otros enfoques, como el feedforward, radican en su capacidad para gestionar incertidumbres en las señales de perturbación, errores en el modelado y procesos inestables. El control eficaz de estos factores es esencial en el diseño de sistemas de control.

A partir de la **Figura 2-1**, se obtiene la siguiente expresión de la salida del sistema en función de las señales de entrada y sus respectivas funciones de transferencia,

$$y = \frac{G_o}{(1+CP)}d_o + \frac{PG_i}{(1+CP)}d_i + \frac{CP}{(1+CP)}r - \frac{CP}{(1+CP)}n \tag{2-1}$$

A partir de (2-1), se define H como la función de transferencia de lazo abierto, mientras S y T como las funciones de transferencia de sensibilidad y sensibilidad complementaria, respectivamente,

$$H = CP \tag{2-2}$$

$$S = \frac{1}{(1+CP)} \tag{2-3}$$

$$T = \frac{CP}{(1+CP)} \tag{2-4}$$

Reemplazando S y T en (2-1) y considerando $G_i = G_o = 1$,

$$y = Sd_o + SPd_i + Tr - Tn \tag{2-5}$$

Además, considerando solo los efectos de la perturbación de entrada y cambios en el setpoint, (2-5) se reduce a

$$y = SPd_i + Tr \tag{2-6}$$

2.2.2. Respuesta en el dominio de la frecuencia

La respuesta en frecuencia se obtiene al reemplazar la variable de Laplace *s* por $j\omega$ en una función de transferencia como $P(s)$ *y* se emplea para describir la respuesta de un sistema a sinusoides de frecuencia variable. Esta interpretación tiene la ventaja de estar directamente relacionada con el dominio del tiempo, y en cada frecuencia ω el número complejo $P(j\omega)$ tiene una interpretación física. Si se aplica una señal de entrada sinusoidal persistente con frecuencia ω [rad/s] y magnitud u_0, tal como

$$u(t) = u_0 \sin(\omega t + \alpha) \tag{2-7}$$

Entonces, la señal de salida es también una sinusoide persistente con la misma frecuencia, es decir

$$y(t) = y_0 \sin(\omega t + \beta) \tag{2-8}$$

Con esta respuesta, se pueden evaluar la magnitud y fase de $P(j\omega)$

$$\frac{y_0}{u_0} = |P(j\omega)|; \quad \phi = \angle P(j\omega)\ [rad]$$

Donde

$$\phi = \beta - \alpha$$

Por lo tanto, después de enviar una señal sinusoidal a través de un sistema $P(s)$, la magnitud de la señal se amplifica por un factor $|P(j\omega)|$ y la fase cambia en $\angle P(j\omega)$. Esta interpretación permite comprender en qué medida la dinámica del sistema atenúa el impacto de las señales de entrada sobre las señales de salida.

Tanto $|P(j\omega)|$, conocida como la ganancia del sistema, y $\angle P(j\omega)$ dependen de la frecuencia ω. Esta dependencia puede representarse explícitamente en diagramas de Bode (con ω como variable independiente) o algo implícitamente en un diagrama de Nyquist (diagrama de plano de fase).

Una ventaja del dominio de la frecuencia en comparación con un análisis de respuesta escalón es que considera una clase más amplia de señales (sinusoides de cualquier frecuencia). Esto facilita la caracterización de las propiedades de retroalimentación y, en particular, el comportamiento del sistema en la región de cruce: el límite entre la estabilidad y la inestabilidad.

2.2.3. Características deseadas en un sistema de control

a. Estabilidad

Así como en otras esferas, la estabilidad está referida a un estado de equilibrio, un estado estacionario que no es fácilmente perturbado por cambios en el medio. En el control de procesos, la estabilidad de un sistema de control recae en el diseño de su controlador y es observable cuando la variable que busca ser controlada no tiene un comportamiento exagerado, por ejemplo, oscilar indefinidamente con amplitudes crecientes. La estabilidad en lazo cerrado es comúnmente medida con los siguientes métodos (Skogestad & Postlethwaite, 2005):

- Evaluando los polos del sistema en lazo cerrado. Es decir, se encuentran las raíces de $1 + H(s) = 0$, donde $H(s)$ es la función de transferencia de lazo abierto representada por $C(s)P(s)$. El sistema es estable si y solo si todos los polos en lazo cerrado están en el semiplano izquierdo (LHP) y sin estar ubicados en el eje imaginario.
- Evaluando la respuesta en frecuencia de $H(j\omega)$ en el plano complejo, se cuenta el número de rodeos que hace del punto crítico -1. Por el criterio de estabilidad de Nyquist, la estabilidad en lazo cerrado se infiere al igualar el número de rodeos en sentido antihorario con el número de polos inestables en lazo abierto (polos RHP).
- Para sistemas estables en lazo abierto donde $\angle H(j\omega)$ cae con una frecuencia tal que $\angle H(j\omega)$ cruza los -180° solo una vez, se puede usar el criterio de estabilidad de Bode:

$$\frac{y_0}{u_0} = |P(j\omega)|; \quad \phi = \angle P(j\omega)\ [rad] \tag{2-9}$$

Es decir, el sistema en lazo cerrado es estable si y solo si la ganancia de lazo $|H(j\omega)|$ es menor que 1 a la frecuencia de cruce con -180°, ω_{180}. En un diagrama Bode,

$|H(j\omega_{180})|$ es la distancia vertical desde la unidad hasta el punto de corte con la frecuencia ω_{180}. Este criterio también puede ser deducido con el diagrama Nyquist.

b. Rendimiento

El rendimiento en un sistema de control se define como su capacidad para generar una respuesta que cumpla con las características deseadas. En este estudio, se consideran las siguientes propiedades:

- **Tiempo de respuesta (t_r):** también llamado tiempo de subida. Es el tiempo que tarda la salida en alcanzar el 90% de su valor final.
- **Tiempo de estabilización (t_s):** tiempo que tarda la respuesta para permanecer dentro de $\pm p$ de su valor final. Donde p comúnmente es 1, 2 o 5%, dependiendo del proceso y de las especificaciones dadas.
- **Sobrepaso máximo:** relación entre máximo pico y el valor en estacionario de la respuesta. Debería mantenerse por debajo de 1.2, es decir 20%, o menos (Åström & Hägglund, 1995; Skogestad & Postlethwaite, 2005).
- **Offset:** diferencia entre el valor final obtenido y el valor final deseado (setpoint).

El rendimiento también puede ser evaluado a través de índices y funciones de rendimiento. Grimholt & Skogestad (2012) propusieron una función que cuantifica escalarmente el rendimiento y que consiste en un promedio ponderado del índice de error absoluto integrado (IAE) para cambios escalón en las perturbaciones de entrada y salida, d_i y d_o, respectivamente. La principal ventaja de esta función es que es independiente de la ganancia del proceso, las magnitudes de cambio escalón en las perturbaciones y setpoints, y la unidad de tiempo empleada. El índice IAE integra el error de durante un período de tiempo determinado,

$$IAE = \int_0^\infty |y(t) - r(t)|dt \tag{2-10}$$

A partir de (2-10), se puede definir una función de rendimiento $J(C)$ más avanzada, que considera factores de ponderación:

$$J(C) = 0.5\left[\frac{IAE_{di}(C)}{IAE_{di}^0} + \frac{IAE_{do}(C)}{IAE_{do}^0}\right] \tag{2-11}$$

donde IAE_{di}^{0} e IAE_{do}^{0}son los factores de ponderación para controladores PI o PID de referencia. Estos se obtienen con controladores óptimos distintos tras una prueba de cambio escalón en d_i y d_o, respectivamente. Mientras que $IAE_{di}(C)$ y $IAE_{do}(C)$ se obtienen con un único controlador con determinada sintonización que compensa satisfactoriamente la estabilidad y el rendimiento.

Para calcular IAE_{di}^{0} e IAE_{do}^{0}, se empleará el máximo pico de sensibilidad (M_{ST}) como criterio de estandarización. Grimholt & Skogestad (2012) establecen un valor de $M_{ST} = 1.59$ como referencia para diseñar un controlador robusto.

c. Robustez

La robustez de un sistema de control puede ser definida como la insensibilidad a las diferencias entre el sistema real y el modelo del sistema que se utilizó para diseñar el controlador (Skogestad & Postlethwaite, 2005). Según Lødemel (Holene, 2013), la robustez de un controlador, definido desde sus márgenes de estabilidad, indica cuán cerca está un sistema de lazo cerrado de la inestabilidad.

Como se mencionó en la Sección 2.2.2, analizar los sistemas en el dominio de la frecuencia, permite caracterizar los sistemas de control hasta los límites de su estabilidad con una amplia gama de entradas (sinusoides de varias frecuencias). Algunos índices en el dominio de la frecuencia utilizados son los márgenes de ganancia (GM) y de fase (PM), el pico máximo de sensibilidad (M_{ST}) y el margen de retardo (DM).

i) Margen de ganancia

El margen de ganancia es el factor por el que $|H(j\omega)|$ puede ser aumentado antes de que el sistema en lazo cerrado se vuelva inestable. Por lo tanto, el GM puede servir como una "contingencia" ante incertidumbres en la ganancia y, generalmente, se requiere que sea mayor a 2 (Skogestad & Postlethwaite, 2005). Se define matemáticamente el GM como

$$GM = \frac{1}{|H(j\omega_{180})|} \tag{2-12}$$

ii) Margen de fase

El margen de fase (PM) indica cuánto retraso de fase puede agregarse a $H(s)$ a la frecuencia ω_c antes de que la fase a esta frecuencia alcance los -180°, que corresponde a un estado inestable. La frecuencia crítica ω_c es aquella en la que $|H(j\omega)|$ se vuelve 1 por

primera vez. El PM, de forma similar al GM, puede servir como una "contingencia" ante incertidumbres en el tiempo de retardo y, normalmente, se requiere que sea mayor a 30° (Skogestad & Postlethwaite, 2005). Se define matemáticamente el PM como

$$PM = \angle H(j\omega) + 180° \tag{2-13}$$

iii) Criterio del pico máximo

El pico máximo de sensibilidad puede ser definido como el peor caso en el que el control por retroalimentación degrada el rendimiento del sistema de control. Matemáticamente, es definido como la inversa de la menor distancia existente entre $H(j\omega)$ y -1; distancia límite con la inestabilidad. Por lo tanto, es un indicador de robustez que se buscar reducir o mantener cerca de 1 (Skogestad & Postlethwaite, 2005). Los picos máximos de sensibilidad y sensibilidad complementaria se definen matemáticamente como

$$M_S = \max\omega \, |S(j\omega)| \tag{2-14}$$

$$M_T = \max\omega \, |T(j\omega)| \tag{2-15}$$

Donde S y T son denominados funciones de sensibilidad y sensibilidad complementaria, respectivamente. Mientras M_S mide la distancia al límite de inestabilidad, M_T representa la sensibilidad a los errores de modelado de ceros y retardo de tiempo, y la amplificación de ruido en el peor de los casos (Holene, 2013).

Para las plantas estables M_S suele ser mayor que M_T, pero esto no es una regla general (Skogestad & Postlethwaite, 2005). A partir S y T, (2-3) y (2-4), se define M_{ST} como el máximo valor entre M_S y M_T.

$$M_{ST} = \max\{M_S, M_T\} \tag{2-16}$$

Además, existe una relación entre los picos máximos y los márgenes de ganancia y de fase

$$GM \geq \frac{M_S}{M_S - 1};\ PM \geq \frac{1}{M_S} \tag{2-17}$$

$$GM \geq 1 + \frac{1}{M_T};\ PM \geq \frac{1}{M_T} \tag{2-18}$$

De (2-17) y (2-18), se deduce que para altos valores de M_S y M_T se obtendrán bajos valores de GM y PM y, por lo tanto, una pobre robustez. Para garantizar los límites de robustez mencionados en las definiciones de los márgenes de ganancia y de fase, $M_S < 2$; de este modo, $GM > 2$ y $PM > 30°$ (Skogestad & Postlethwaite, 2005). También es típicamente recomendado un valor de $M_S < 1.6$ (Rivera et al., 1986), para obtener $GM > 2.67$ y $PM > 36.4°$.

iv) Margen de retardo

El margen de retardo se define como el cambio en el tiempo de retardo requerido para hacer un sistema inestable (Åström & Hägglund, 1995). Siendo ω_c la frecuencia de cruce, el margen de retardo (DM) se define matemáticamente como

$$DM = \frac{PM}{\omega_c} \tag{2-19}$$

Es pertinente señalar que el PM y ω_c deben estar en las unidades correspondientes de rad y rad/s, respectivamente. El margen de retardo se emplea para medir la robustez del sistema de control ante incertidumbres en el tiempo de retardo, algo que no se puede capturar con las medidas de robustez clásicas como el margen de ganancia y de fase (Åström & Hägglund, 1995). Además, como se verá más adelante, la robustez en términos de M_{ST} no garantizaría una en términos de DM.

Fundamentadas las características deseadas en el sistema de control, se pueden definir los siguientes términos:

- **Estabilidad nominal.** El sistema es estable con un modelo sin incertidumbres.
- **Rendimiento nominal.** El sistema cumple con las especificaciones de rendimiento con un modelo sin incertidumbres.
- **Estabilidad robusta.** El sistema es estable para todas las plantas perturbadas sobre un modelo nominal hasta el peor caso de incertidumbre de modelado.
- **Rendimiento robusto.** El sistema satisface las especificaciones de rendimiento para todas las plantas perturbadas sobre un modelo nominal hasta el peor caso de incertidumbre de modelado.

Para el presente estudio, serán el criterio de pico máximo y el margen de retardo los indicadores más representativos para la evaluación de los sistemas de control, siendo en

el primer caso ampliamente utilizado en estudios vigentes y el segundo, por influencia directa con el elemento de evaluación: tiempo de retardo.

2.3. Modelos de plantas industriales con dinámicas complejas

La dinámica de un proceso puede representarse matemáticamente mediante modelos que describen su comportamiento. Dicho modelado puede realizarse a partir de los principios de conservación de masa, energía y cantidad de movimiento, obteniéndose expresiones en forma de ecuaciones diferenciales. Posteriormente, mediante la transformada de Laplace, estas ecuaciones pueden expresarse en una forma más conveniente para el análisis dinámico y el diseño de control, como es la función de transferencia.

Las funciones de transferencia constituyen una herramienta fundamental para describir y predecir la respuesta dinámica de un proceso ante cambios en sus entradas. En particular, los modelos de orden reducido son ampliamente utilizados en control de procesos, ya que permiten representar de manera simplificada comportamientos dinámicos relevantes de diversas plantas reales y facilitan el análisis comparativo de distintas estrategias de control.

Si bien estas funciones de transferencia simples corresponden a aproximaciones de los procesos industriales, resultan útiles para capturar la dinámica dominante alrededor de un punto de operación. A partir de ello, es posible analizar la respuesta de la planta frente a cambios en la entrada y perturbaciones, así como evaluar su desempeño frente a diferentes esquemas de control. En este contexto, la relación entre cada modelo y una planta real debe entenderse como una correspondencia basada en la naturaleza de la respuesta dinámica del proceso, y no como una equivalencia estricta entre equipo y función de transferencia.

Para este estudio, se seleccionaron modelos de planta de orden reducido con tiempo de retardo, debido a que permiten abarcar distintos escenarios dinámicos de interés práctico y académico en el campo del control de procesos. Se consideraron procesos estables, procesos próximos al integrador o integradores, y procesos inestables.

2.3.1. Modelo con retardo puro

El modelo de retardo puro está representado por,

$$P_1(\mathrm{s}) = e^{-\theta s} \tag{2-20}$$

donde θ es el tiempo de retardo aparente. En este caso, el comportamiento dinámico está determinado únicamente por la presencia de tiempo de retardo, la salida reproduce la entrada sin alterar su forma ni amplitud, pero desplazada en el tiempo.

Físicamente, este modelo se asocia a situaciones donde existe un retardo entre la acción aplicada y la observación de su efecto, debido principalmente a fenómenos de transporte o medición. Aunque pocas plantas reales presentan de manera aislada un comportamiento de retardo puro, este modelo resulta útil cuando el tiempo de retardo constituye el fenómeno dominante de la respuesta observada.

Asociado con procesos reales, P_1 puede relacionarse con sistemas de transporte de material o fluidos, así como con analizadores en línea o sistemas de medición alejados del punto de actuación, donde la respuesta se observa con un retraso apreciable (Shinskey, 1967).

2.3.2. Procesos estables autorregulados

Este tipo de procesos adquiere la forma general de un modelo de primer orden más tiempo de retardo (FOPDT) representado por,

$$P(\mathrm{s}) = \frac{k_p}{\tau \mathrm{s} + 1}\, e^{-\theta s} \tag{2-21}$$

donde k_p, τ y θ son la ganancia del proceso, la constante de tiempo aparente y el tiempo de retardo aparente, respectivamente. Este es uno de los modelos más comunes en la industria y también constituye la base de la mayoría de las reglas clásicas de sintonización (Rivera et al., 1986).

Para este estudio, se usaron tres variantes, representadas por las funciones de transferencia

$$P_2(s) = \frac{e^{-s}}{s + 1} \tag{2-22}$$

$$P_3(s) = \frac{e^{-s}}{8s+1} \tag{2-23}$$

$$P_4(s) = \frac{e^{-s}}{20s+1} \tag{2-24}$$

corresponden a procesos estables autorregulados de primer orden con tiempo de retardo. En este tipo de sistemas, ante un cambio en la entrada, la salida evoluciona hasta alcanzar un nuevo estado estacionario, por lo que presentan una tendencia natural a estabilizarse.

El modelo P_2 representa una dinámica balanceada, en la que la constante de tiempo y el tiempo de retardo tienen una influencia comparable sobre la respuesta del proceso. En sistemas reales, este comportamiento se asocia a variables autorreguladas como temperatura, presión o composición, cuando la respuesta del sistema está dominada por una sola inercia principal y un retardo moderado.

Por su parte, el modelo P_3 representa un proceso también estable, pero con una constante de tiempo significativamente mayor, por lo que su respuesta es más lenta y el efecto de la dinámica de la planta es más significativo. Este tipo de dinámica puede reflejar procesos térmicos con alta inercia, líneas de transporte largas o sistemas de medición retardada, en los cuales la planta sigue siendo autorregulada, pero el retardo limita considerablemente la acción de control (Normey-Rico & Camacho, 2008b).

En el caso de P_4, la planta conserva una tendencia natural a estabilizarse ante un cambio en la entrada, ya que posee un estado estacionario finito. Sin embargo, debido a su elevada constante de tiempo, su respuesta puede ser muy lenta y, en escalas prácticas de análisis, asemejarse parcialmente a un comportamiento acumulativo, próximo al integrador. Este tipo de dinámica puede asociarse a procesos con gran inercia o elevado inventario, como sistemas térmicos de alta masa, recipientes de gran volumen o ciertas variables de proceso con autorregulación débil.

2.3.3. Procesos integradores

Este tipo de procesos adquiere la forma general de un modelo integrador más tiempo de retardo (IPDT), es empleado con éxito cuando el tiempo de retardo es muy pequeño en comparación con la constante de tiempo. Si $\tau \gg \theta$, el modelo FOPDT puede ser reducido

$$P(\mathrm{s}) = \frac{k_p}{\tau \mathrm{s} + 1}\ e^{-\theta s} \cong \frac{k_p}{\tau \mathrm{s}} e^{-\theta s} \cong \frac{k_v}{\mathrm{s}} e^{-\theta s} \tag{2-25}$$

Donde $k_v = k_p/\tau$ y es llamado ganancia de velocidad. Un proceso integrador es un caso especial de planta inestable con un polo en $\mathrm{s} = 0$, es decir, un polo en el límite de la inestabilidad. Por lo tanto, el controlador debe ser diseñado y sintonizado pertinentemente para tratarlo.

Para este estudio, se usó la siguiente variante del modelo,

$$P_5(s) = \frac{e^{-s}}{\mathrm{s}} \tag{2-26}$$

Estos procesos son conocidos por su característica no autorregulable, es decir, no alcanzan el estado estacionario durante operaciones en lazo abierto.

Un proceso integrador, en el cual la salida no converge naturalmente a un nuevo estado estacionario ante un desbalance sostenido, sino que continúa creciendo o decreciendo debido a un mecanismo físico de acumulación. Un ejemplo clásico de este comportamiento es el nivel de líquido en un tanque, donde la variable cambia continuamente mientras exista una diferencia neta entre los flujos de entrada y salida (Bequette, 2003).

En P_4, la salida aún posee una tendencia natural a estabilizarse, aunque muy lenta; en P_5, dicha tendencia no existe, ya que la dinámica está gobernada por la acumulación.

2.3.4. Procesos inestables

Un proceso inestable en lazo abierto adquiere la forma del modelo inestable de primer orden más tiempo de retardo (UFOPDT)

$$P(\mathrm{s}) = \frac{k_p}{\tau \mathrm{s} - 1} e^{-\theta s} \tag{2-27}$$

Los procesos inestables poseen polos en semiplano derecho y, en (2-27), el polo se ubicaría en $1/\tau$. Así como para el modelo IPDT, los controladores deben ser diseñados tal que eliminen, en primer lugar, los polos indeseados.

Para este estudio se emplearon dos variantes, representadas por las funciones de transferencia

$$P_6(s) = \frac{e^{-s}}{s-1} \tag{2-28}$$

$$P_7(s) = \frac{e^{-s}}{8s-1} \tag{2-29}$$

En este tipo de sistemas, pequeñas perturbaciones alrededor del punto de operación no son amortiguadas por la dinámica natural del proceso, sino amplificadas, de modo que la salida tiende a alejarse progresivamente del estado estacionario. En la literatura de control de procesos con tiempo de retardo, los sistemas inestables constituyen una clase de especial interés, ya que el retardo incrementa la dificultad de estabilización y reduce el margen disponible para la acción de control (Normey-Rico & Camacho, 2008a).

Desde el punto de vista físico, estos modelos no buscan representar de forma exacta una unidad industrial específica, sino aproximar el comportamiento dinámico dominante de procesos que, alrededor de ciertas condiciones de operación, presentan equilibrio abierto inestable. Esta aproximación es consistente con trabajos recientes donde procesos químicos o bioquímicos no lineales e inestables con tiempo de retardo son reducidos a modelos de primer orden inestable con retardo para fines de análisis y diseño de control (Camacho et al., 2023).

El modelo P_6 representa una dinámica inestable con retardo moderado. Conceptualmente, puede asociarse a procesos en los que existe una realimentación positiva interna que domina sobre los mecanismos disipativos del sistema. Un ejemplo clásico es el reactor exotérmico, donde un pequeño incremento de temperatura puede acelerar la reacción, incrementar la liberación de calor y generar un nuevo aumento de temperatura, alejando al proceso del punto de equilibrio si la remoción térmica no compensa dicho efecto. En este sentido, P_6 puede interpretarse como una representación simplificada de procesos con inestabilidad intrínseca cercana al punto de operación. (Camacho et al., 2023).

Por su parte, el modelo P_7 describe un proceso igualmente inestable, pero con una dinámica más lenta y con una interacción más marcada entre la inestabilidad y el tiempo de retardo. En este caso, además de la tendencia natural del sistema a alejarse del equilibrio, el retardo limita la capacidad del controlador para reaccionar oportunamente, ya que el efecto de la acción correctiva se observa de forma tardía. Este tipo de comportamiento puede asociarse, de manera aproximada, a reactores exotérmicos con

medición retardada, a sistemas de reacción-separación fuertemente acoplados o, en general, a procesos inestables en los que el retardo de transporte o medición agrava el problema de control (Normey-Rico & Camacho, 2008a).

La inclusión de P_6 y P_7 en este estudio permite evaluar el desempeño del controlador frente a dos escenarios particularmente exigentes: uno donde predomina la inestabilidad de la planta y otro donde, además, el tiempo de retardo adquiere un papel aún más crítico. De este modo, ambos modelos amplían el análisis hacia condiciones que, aunque más severas, siguen siendo relevantes desde el punto de vista académico y práctico en control de procesos.

2.4. Algoritmo clásico PID

En esta sección se describe el controlador PID clásico como estrategia de referencia para la comparación desarrollada en la investigación. Primero, se presentan sus acciones proporcional, integral y derivativa, destacando el efecto de cada una sobre el error, la respuesta dinámica y la estabilidad del sistema. Luego, se revisan las principales estructuras del controlador PID, incluyendo las formas paralelas, en cascada y con filtro derivativo. Finalmente, se establece la estructura seleccionada para la evaluación posterior, considerando su aplicabilidad en procesos con retardo y su papel como controlador convencional frente al Predictor de Smith clásico y al Predictor de Smith Modificado.

2.4.1. Acción proporcional, integral y derivativa

a. Acción proporcional

Con la acción proporcional en un controlador, la señal de control es proporcional al error

$$u = k_c e \tag{2-30}$$

donde k_c es la ganancia del controlador. Un controlador proporcional puro es más sofisticado que sistemas más simples como el on-off, cuyas acciones se limitan a máximos y mínimos preestablecidos, provocando un carácter inevitablemente oscilatorio. Sin embargo, el setpoint en estado estacionario es difícilmente alcanzado dejando un error de corrimiento (offset).

b. Acción integral

Un controlador proporcional puro tiene la desventaja de que la variable de proceso no consigue establecerse en el setpoint. Esto se puede corregir haciendo la acción de control proporcional a la integral del error

$$u = k_i \int_0^t e(\tau) d\tau \tag{2-31}$$

donde k_i es la ganancia integral. El objetivo principal de la acción integral es asegurarse de que la salida del proceso coincida con el setpoint en el estado estacionario. Analizando (2-31), siendo k_i constante y t creciendo sin límite, la única posibilidad de que u adquiera un valor estacionario es que el error (e) sea constante e igual a cero. De esta forma, siempre que un sistema de control posea la acción integral, la señal de error será cero en estado estacionario. El controlador PI se puede expresar matemáticamente de la siguiente forma

$$u = k_c \left(e + \frac{1}{\tau_i} \int_0^t e(\tau) d\tau \right) \tag{2-32}$$

donde τ_i es el tiempo integral. Transformando (2-32) al dominio de Laplace se obtiene la función de transferencia del controlador PI.

$$C(s) = k_c \left(\frac{\tau_i s + 1}{\tau_i s} \right) \tag{2-33}$$

c. Acción derivativa

Un refinamiento adicional consiste en dotar al controlador de una capacidad anticipativa utilizando una predicción (τ_d unidades de tiempo en el futuro) de la salida basada en una extrapolación lineal con un término de acción derivativa (Åström & Hägglund, 1995). El controlador PID se puede expresar matemáticamente como

$$u = k_c \left(e + \frac{1}{\tau_i} \int_0^t e(\tau) d\tau + \tau_d \frac{de}{dt} \right) \tag{2-34}$$

donde τ_d es el tiempo derivativo. El objetivo de tener acción derivativa en un controlador es mejorar la respuesta dinámica de la variable controlada en lazo cerrado y disminuir el

tiempo necesario para alcanzar el estado estacionario. La acción derivativa aumenta el margen de fase y, por lo tanto, la robustez del controlador (Normey-Rico & Camacho, 2007). Este aumento de robustez otorga la posibilidad de un ajuste más agresivo, lo que mejora el rendimiento y hace al controlador PID aplicable a una amplia variedad de procesos industriales. Por otro lado, la acción derivativa del controlador también agrega un cero al sistema, lo que puede aumentar su sensibilidad a altas frecuencias. Esto último puede ser corregido introduciendo un filtro a la estructura, como se explica más a profundidad en la Sección del Predictor de Smith Modificado.

La acción derivativa puede ser ajustada para contrarrestar el retraso por tiempo de retardo y permitir que la estructura restante, PI, sea sintonizada como si el proceso estuviese "libre" de este retraso. Sin embargo, esto solo es posible si no se requiere una respuesta en lazo cerrado demasiado rápida y si no se considera un proceso con un tiempo de retardo dominante. La acción derivativa predice el error que es la señal de entrada al controlador, pero no elimina el término del tiempo de retardo de la ecuación característica.

A partir de (2-1) y despejando las funciones de transferencia que relacionan la perturbación de entrada y la referencia con la salida del sistema, se obtienen

$$\frac{Y(s)}{D_i(s)} = H_d = \frac{P}{(1+CP)} \tag{2-35}$$

$$\frac{Y(s)}{R(s)} = H_r = \frac{CP}{(1+CP)} \tag{2-36}$$

De (2-35) y (2-36), se obtiene la ecuación característica en lazo cerrado para una estructura de control por retroalimentación con controlador PID clásico.

$$1 + C(s)P(s) = 1 + H(s) \tag{2-37}$$

Donde H(s) es la función de transferencia en lazo abierto. Separando P(s) en dos términos que corresponden al proceso libre de tiempo de retardo y al tiempo de retardo puro, respectivamente,

$$P(\mathrm{s}) = P_{\circ}(\mathrm{s})e^{-\theta s} \tag{2-38}$$

Reemplazando (2-38) en (2-37), se obtiene la ecuación característica en lazo cerrado para un controlador PID. Se destaca la permanencia del término del tiempo de retardo que no es posible eliminar con un controlador convencional de este tipo.

$$1 + C(\mathrm{s})P_{\circ}(\mathrm{s})e^{-\theta s} = 0 \tag{2-39}$$

2.4.2. Estructuras del controlador PID

a. Controlador PID paralelo estándar

La estructura paralela estándar, presentada en (2-40), se deriva de (2-34) transformada al dominio de Laplace. Como se muestra en la **Figura 2-2**, esta es una estructura no interactuante donde el tiempo derivativo y el integral no influyen uno sobre el otro.

$$C(s) = k_c\left(1 + \frac{1}{\tau_i s} + \tau_d s\right) \tag{2-40}$$

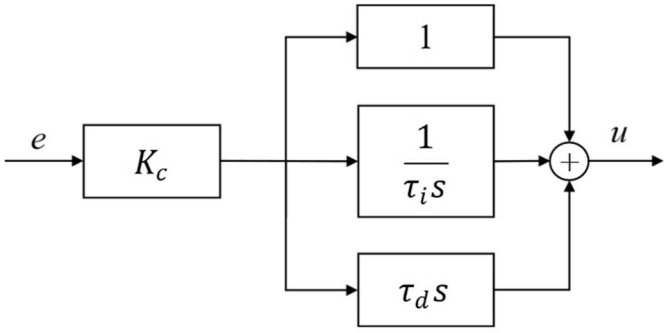

Figura 2-2. Estructura paralela estándar del controlador PID

b. Controlador PID en cascada

La estructura estándar también puede ser transformada a una en serie o en cascada como se muestra en (2-41) y la **Figura 2-3**.

$$C(s) = k_c\left(\frac{\tau_i s + 1}{\tau_i s}\right)(\tau_d s + 1) \tag{2-41}$$

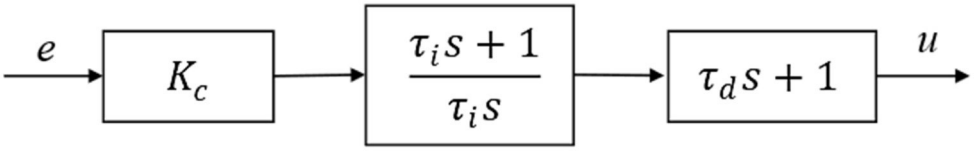

Figura 2-3. Estructura en cascada del controlador PID

Esta forma era aprovechada cuando todos los controladores eran analógicos y era conveniente tener el elemento PI y el PD en serie (Holene, 2013). En la **Figura 2-4**, se presenta una distribución alternativa "interactuante" que resalta la influencia del tiempo derivativo sobre la parte integral.

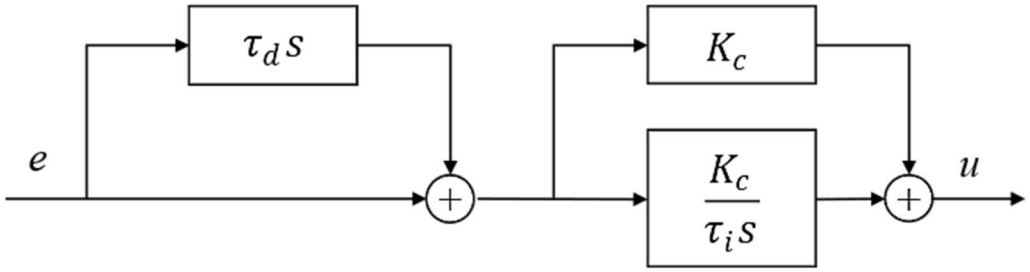

Figura 2-4. Estructura en cascada interactuante del controlador PID

Nota. Adaptado de Holene (2013).

c. Controlador PID en cascada con filtro derivativo

Las estructuras de (2-40) y (2-41) son físicamente irrealizables sin un filtro derivativo porque el grado del numerador es mayor que el del denominador (más ceros que polos), convirtiéndolas en funciones de transferencia impropias. Además, como se mencionó previamente, la acción derivativa del controlador aumenta la sensibilidad del sistema haciendo que la derivada de la variable medida se altere excesivamente cuando se experimentan fluctuaciones aleatorias de alta frecuencia del ruido de medición (Holene, 2013). La sensibilidad de alta frecuencia y la función de transferencia impropia del controlador puede ser corregidas filtrando la acción derivativa del controlador. La estructura en cascada con filtro derivativo está representada por (2-42)

$$C(s) = K_c\left(\frac{\tau_i s + 1}{\tau_i s}\right)\left(\frac{\tau_d s + 1}{\tau_f s + 1}\right) \tag{2-42}$$

donde τ_f es la constante de tiempo del filtro derivativo. A menudo, τ_f se escribe como el producto de algún factor y la constante de tiempo derivada, $\tau_f = \alpha\tau_d$, donde $\alpha \in (0, 1)$ y se denomina constante de filtro derivativo. Para controladores industriales, el valor de α normalmente varía entre 0.05 y 0.5 (Normey-Rico & Camacho, 2007). Este parámetro se puede utilizar para ajustar la atenuación del ruido y también la robustez del lazo cerrado. Grimholt & Skogestad (2013) señalan que el filtro derivativo no debería influir significativamente en las características de rendimiento y robustez del controlador pues,

de lo contrario, se estaría añadiendo un grado de libertad a la sintonización del controlador, es decir cuatro parámetros ajustables. En consecuencia, Grimholt & Skogestad (2013) recomiendan que τ_f no debería ser mayor que la tercera parte de τ_d, lo que estaría en el intervalo propuesto por Normey-Rico & Camacho (2007).

Por lo descrito anteriormente, esta última estructura PID es la seleccionada para desarrollar la evaluación de sistemas de control.

2.5. Predictor Smith

En esta sección se presenta el Predictor de Smith como una estrategia orientada a mejorar el control de procesos con tiempo de retardo. Primero, se describe la estructura clásica del Predictor de Smith, su principio de compensación del retardo y sus principales propiedades en lazo cerrado. Luego, se analizan sus alcances y limitaciones frente al seguimiento de referencia, rechazo de perturbaciones, plantas integradoras, plantas inestables e incertidumbre en el tiempo muerto. Finalmente, se introduce el Predictor de Smith Modificado como una extensión de la estructura clásica, incorporando filtros que permiten ampliar su aplicabilidad y mejorar el compromiso entre rendimiento y robustez en procesos con dinámicas complejas.

2.5.1. Predictor de Smith original (SP)

Un controlador PI sólo considera datos presentes y pasados, y un controlador PID predice el comportamiento futuro del proceso mediante extrapolación lineal. Un mecanismo predictivo puede mejorar el rendimiento del controlador, particularmente cuando el sistema tiene tiempos de retardo. Un mecanismo predictivo simple que podría tratar los procesos con grandes tiempos de retardo fue propuesto por Smith en 1957: Predictor de Smith (SP).

La **Figura 2-5** presenta la estructura básica del Predictor de Smith, donde $C(\mathrm{s})$ y $P(\mathrm{s})$ son las funciones de transferencia del controlador y del proceso. El predictor está compuesto por un modelo del proceso sin tiempo de retardo $\widetilde{P}_\circ(\mathrm{s})$ y un modelo del tiempo de retardo $e^{-\theta_\circ s}$. De este modo, se define el modelo del proceso con tiempo de retardo

$$\tilde{P} = \widetilde{P}_\circ(\mathrm{s})e^{-\theta_\circ s} \tag{2-43}$$

El modelo $\widetilde{P}_{\circ}(\mathrm{s})$, denominado modelo rápido, se utiliza para calcular una predicción de lazo abierto. El proceso y el modelo del proceso con tiempo de retardo son iguales solo en el caso nominal, es decir, cuando existe un modelado perfecto

$$P(\mathrm{s}) = \tilde{P}(\mathrm{s}) \leftrightarrow P_{\circ}(\mathrm{s}) = \widetilde{P}_{\circ}(\mathrm{s}) \wedge \theta = \theta_{\circ} \tag{2-44}$$

Para considerar los errores de modelado, la diferencia entre la salida del proceso $y(t)$ y el modelo $\tilde{\mathrm{y}}(t)$, que incluye el tiempo de retardo, se agrega a la predicción de lazo abierto $\tilde{\mathrm{y}}(t + \theta_{\circ})$, a través del error de predicción, e_p. Si no hay errores de modelado y perturbaciones, el error entre la salida del proceso real y la salida del modelo será nulo; por lo tanto, la señal de salida del predictor será la salida sin tiempo de retardo de la planta

$$\tilde{\mathrm{y}}(t) = y(t) \rightarrow y_p(t) = \tilde{\mathrm{y}}(t + \theta_{\circ}) \tag{2-45}$$

En estas condiciones, $C(\mathrm{s})$ puede sintonizarse, al menos en el caso nominal, como si la planta no tuviera tiempo de retardo.

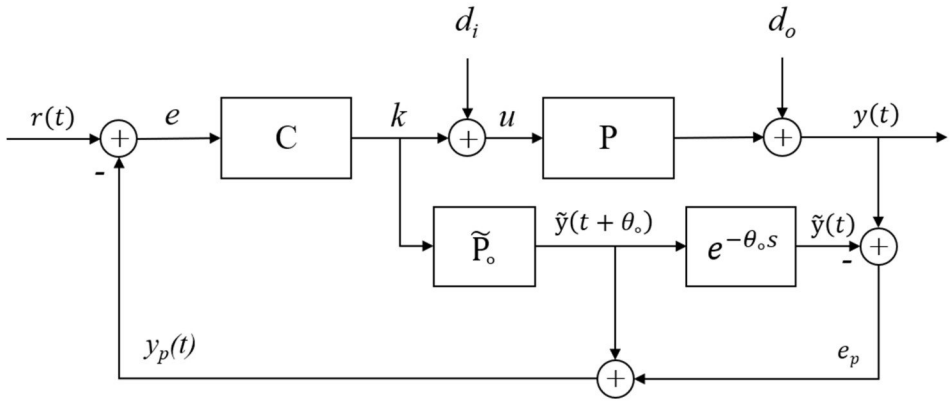

Figura 2-5. Estructura clásica del Predictor de Smith (SP)

Nota. Adaptado de Smith (1957) y Normey-Rico y Camacho (2007).

A partir de la **Figura 2-5,** se pueden obtener las siguientes ecuaciones asociadas a la respuesta controlada por cambios en la referencia, perturbación de entrada y perturbación de salida, respectivamente[2].

$$H_r^{SP}(s) = \frac{\mathrm{C(s)}\widetilde{P}_\circ(\mathrm{s})}{1 + \mathrm{C(s)}\widetilde{P}_\circ(\mathrm{s})} e^{-\theta_\circ s} \tag{2-46}$$

$$H_{di}^{SP}(s) = \widetilde{P}_\circ(\mathrm{s}) e^{-\theta_\circ s} \left[1 - \frac{\mathrm{C(s)}\widetilde{P}_\circ(\mathrm{s})}{1 + \mathrm{C(s)}\widetilde{P}_\circ(\mathrm{s})} e^{-\theta_\circ s}\right] \tag{2-47}$$

$$H_{do}^{SP}(s) = 1 - \frac{\mathrm{C(s)}\widetilde{P}_\circ(\mathrm{s})}{1 + \mathrm{C(s)}\widetilde{P}_\circ(\mathrm{s})} e^{-\theta_\circ s} \tag{2-48}$$

Alcances y limitaciones de bucle cerrado del Predictor de Smith

a. Seguimiento de la referencia

La principal ventaja del Predictor de Smith, SP, es su capacidad para mejorar el seguimiento de referencia en procesos con tiempo de retardo dominante. Esto se debe a que su estructura permite retirar el término de retardo de la ecuación característica nominal y, por lo tanto, de la dinámica que define la función de transferencia en lazo cerrado H_r^{SP}.

En condiciones ideales, es decir, cuando no existen errores de modelado ni perturbaciones, el tiempo de retardo no aparece en la ecuación característica asociada al seguimiento del punto de referencia. Por ello, el controlador puede sintonizarse como si el proceso no tuviera retardo. En este caso, la ecuación característica queda dada por

$$1 + C(\mathrm{s})\widetilde{P}_\circ(\mathrm{s}) = 0 \tag{2-49}$$

La expresión anterior corresponde al denominador de las funciones de transferencia nominales del SP. En contraste, en una estructura convencional sin predictor, como un

[2] La obtención a detalle de estas funciones de transferencia forma parte del diseño de este estudio, por lo que su desarrollo fue ubicado en el Capítulo 5: Diseño de los sistemas de control, se dejan las ecuaciones resultantes para facilitar la explicación de las propiedades del SP original.

controlador PID aplicado directamente al proceso con retardo, la ecuación característica conserva la dependencia explícita del tiempo de retardo:

$$1 + C(\mathrm{s})\widetilde{P}_{\circ}(\mathrm{s})e^{-\theta_{\circ}s} = 0 \qquad (2\text{-}50)$$

En este último caso, el retardo introduce una fase adicional negativa, lo cual reduce el margen de fase y limita la rapidez con la que puede sintonizarse el controlador sin comprometer la estabilidad (Normey-Rico & Camacho, 2007).

b. Rechazo a perturbaciones

Una de las principales limitaciones del SP clásico aparece en el rechazo de perturbaciones, especialmente cuando estas ingresan en la entrada del proceso. Aunque el SP mejora notablemente el seguimiento de referencia, su estructura original no permite modificar a la vez la dinámica de rechazo a perturbaciones, ya que los polos indeseados de la planta permanecen en la función de transferencia resultante $H_{di}^{SP}(s)$.

En esta expresión, $\tilde{P}(\mathrm{s})$ aparece como factor multiplicativo externo. Por lo tanto, los polos de la planta nominal sin retardo permanecen presentes en la respuesta ante perturbaciones. En consecuencia, aunque el controlador $C(s)$ se diseñe para obtener una respuesta rápida de referencia, los polos lentos de la planta pueden seguir dominando la respuesta de rechazo de perturbaciones.

Esto tiene una consecuencia fundamental: si los polos de $\tilde{P}(\mathrm{s})$ son más lentos que los polos definidos en lazo cerrado de H_r^{SP}, dominarán la respuesta de rechazo de perturbaciones y no será posible acelerar su dinámica.

Adicionalmente, si $\tilde{P}(\mathrm{s})$ posee polos inestables, será complicado diseñar un controlador que elimine los polos indeseados y logre una respuesta satisfactoria sin elevar la complejidad de su sintonización.

- **Plantas inestables**

El seguimiento de referencia puede llegar a estabilizarse si se selecciona un controlador $C(\mathrm{s})$ adecuado para estabilizar el modelo nominal $\widetilde{P}_{\circ}(\mathrm{s})$. Sin embargo, la transferencia de perturbación conserva la influencia de los polos inestables de la planta, debido a la aparición explícita de $\widetilde{P}_{\circ}(\mathrm{s})$ en $H_{di}^{SP}(s)$.

Por esta razón, aunque el lazo de referencia pueda presentar una respuesta aceptable, la respuesta ante perturbaciones puede volverse inestable o físicamente inaceptable. Para evitarlo sería necesario introducir cancelaciones o modificaciones estructurales adicionales, lo que incrementa la complejidad de la sintonización y motiva el uso de estructuras modificadas del Predictor de Smith.

- **Plantas integradoras**

En procesos integradores, el SP clásico presenta otra limitación importante: El SP clásico no garantiza rechazo completo de perturbaciones tipo escalón en procesos integradores. El valor estacionario de la salida queda influenciado por el tiempo de retardo del proceso, lo que limita la aplicabilidad de la estructura original del SP para este tipo de plantas (Normey-Rico & Camacho, 2007).

c. Robustez

En aplicaciones reales, el modelo nominal utilizado por el Predictor de Smith no coincide exactamente con la planta real. Por ello, el análisis de robustez debe considerar incertidumbres de modelado, entre las que destacan: dinámica no modelada, errores en la estimación de la ganancia del proceso y errores en la estimación del tiempo de retardo.

La dinámica no modelada aparece, por ejemplo, cuando un proceso de orden alto se aproxima mediante un modelo reducido. En estos casos, el controlador se diseña con base en una representación simplificada, por lo que la planta real puede presentar modos dinámicos adicionales no considerados durante la sintonización. Estos errores pueden deteriorar el desempeño, aumentando el sobrepaso, ralentizando la respuesta o reduciendo los márgenes de estabilidad.

Los errores en la ganancia del proceso también afectan la respuesta del sistema. Si la ganancia real es mayor que la ganancia nominal, el lazo puede volverse más agresivo y presentar mayor sobrepaso o menor margen de estabilidad. Si la ganancia real es menor, la respuesta puede hacerse más lenta y el rechazo de perturbaciones puede degradarse. Sin embargo, este tipo de incertidumbre suele ser manejable mediante una sintonización conservadora del controlador.

El caso más crítico para el SP corresponde a los errores en la estimación del tiempo de retardo. Esto se debe a que la estructura del predictor se basa precisamente en compensar

el retardo del modelo nominal. Si el tiempo de retardo real difiere del valor usado en el predictor, la compensación deja de ser exacta y aparece un retardo residual dentro del lazo cerrado. Dicho retardo introduce pérdida de fase, reduce los márgenes de estabilidad y puede provocar oscilaciones o incluso inestabilidad.

Por lo tanto, la robustez del SP depende fuertemente de la precisión con la que se conozca el tiempo de retardo del proceso. Mientras que errores moderados en ganancia o dinámica no modelada pueden tolerarse con una sintonización adecuada, los errores en el retardo suelen producir una degradación mucho más severa del comportamiento en lazo cerrado.

En consecuencia, el SP no debe ajustarse en aplicaciones reales considerando únicamente el modelo nominal. Es necesario evaluar explícitamente la robustez frente a incertidumbre en el tiempo de retardo y, cuando esta incertidumbre sea significativa, emplear sintonizaciones más conservadoras o estructuras modificadas que mejoren la tolerancia al desajuste del modelo.

En síntesis, el Predictor de Smith clásico presenta ventajas relevantes para el seguimiento de referencia en procesos con tiempo de retardo, debido a que permite retirar el retardo de la ecuación característica nominal. Sin embargo, su aplicabilidad se ve limitada cuando se evalúa el rechazo de perturbaciones, especialmente ante perturbaciones en la entrada del proceso, plantas con polos lentos, procesos integradores, dinámicas inestables e incertidumbre en el modelado del tiempo de retardo. Estas limitaciones afectan directamente el compromiso entre rendimiento y robustez del lazo cerrado. Por ello, esta investigación analizará en qué medida el Predictor de Smith Modificado permite abordar dichas limitaciones frente al controlador PID y al Predictor de Smith clásico, considerando indicadores en el dominio del tiempo, medidas de robustez y sensibilidad ante errores en el retardo.

2.5.2. Predictor de Smith Modificado (MSP)

Desde la propuesta original del Predictor de Smith, SP, se han desarrollado diversas modificaciones orientadas a superar sus principales limitaciones. Estas extensiones buscan mejorar el rechazo de perturbaciones medibles y no medibles, ampliar su aplicación a procesos integradores e inestables, incrementar la robustez frente a incertidumbres de modelado y facilitar la sintonización mediante estructuras de dos grados de libertad, 2DOF.

En este contexto, Normey-Rico & Camacho (2008a)[3] propusieron una estructura modificada y unificada del SP, también conocida como Predictor de Smith Filtrado o Predictor de Smith Modificado, MSP. Esta estructura incorpora filtros adicionales que permiten controlar procesos estables, integradores e inestables con tiempo de retardo, manteniendo una metodología sistemática para la selección de sus parámetros.

La estructura considerada se muestra en la **Figura 2-6**. En ella, $F_r(s)$ representa el filtro de referencia, utilizado para ajustar la respuesta ante cambios en el punto de operación, mientras que $F_p(s)$ corresponde al filtro predictivo, encargado de mejorar las propiedades de rechazo de perturbaciones y robustez del predictor.

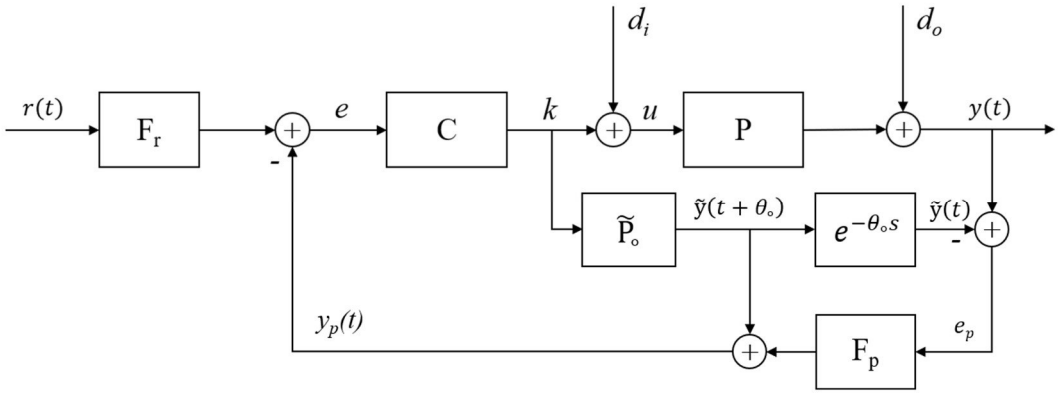

Figura 2-6. Estructura del Predictor de Smith Modificado (MSP) con filtros de referencia y predictivo

Nota. Adaptado de Normey-Rico & Camacho (2008a).

El filtro predictivo, $F_p(s)$, es una función de transferencia con ganancia estática unitaria ($F_p(0) = 1$). Esta condición asegura que el filtro no modifique la ganancia estacionaria nominal del sistema. En ausencia de perturbaciones y errores de modelado, la señal de error de predicción es nula, por lo que el filtro predictivo no altera la respuesta nominal de seguimiento de referencia.

La principal ventaja de incorporar $F_p(s)$ es que introduce grados de libertad adicionales en la dinámica de rechazo de perturbaciones. A diferencia del SP clásico, donde la

[3] A continuación, se desarrolla la explicación de los fundamentos del MSP tomando el aporte de Normey-Rico & Camacho (2008), ajustado a los objetivos de este estudio.

respuesta ante perturbaciones queda fuertemente condicionada por los polos del proceso, el MSP permite modificar el término asociado a dichos polos mediante el filtro predictivo. Esto resulta especialmente relevante en plantas con polos lentos, integradores o inestables.

La estructura de la **Figura 2-6** puede representarse mediante el esquema equivalente de la **Figura 2-7**. En esta representación, el efecto del filtro predictivo se incorpora en un controlador equivalente filtrado $\tilde{C}_{eq}(\mathrm{s})$, mientras que el filtro de referencia se agrupa en un filtro equivalente $F_{eq}(\mathrm{s})$.

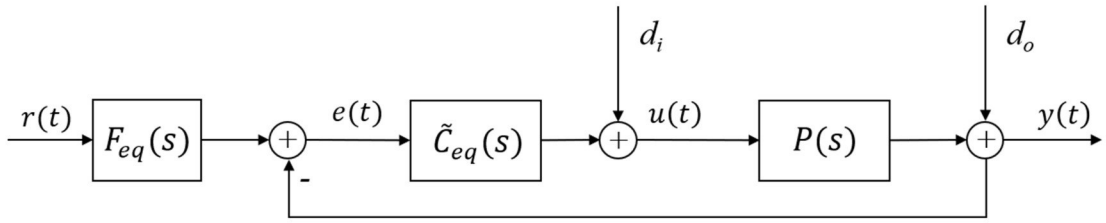

Figura 2-7. Estructura equivalente del Predictor de Smith Modificado (MSP)

Nota. Adaptado de Normey-Rico & Camacho (2008a).

Las funciones equivalentes se definen como:

$$\tilde{C}_{eq}(\mathrm{s}) = \frac{C(\mathrm{s})F_p(\mathrm{s})}{1 + C(s)\widetilde{P}_{\circ}(\mathrm{s})(1 - e^{-\theta_{\circ}s}F_p(\mathrm{s}))} \tag{2-51}$$

$$F_{eq}(\mathrm{s}) = \frac{F_r(\mathrm{s})}{F_p(\mathrm{s})} \tag{2-52}$$

Estas expresiones muestran que $F_p(\mathrm{s})$ no actúa únicamente como un filtro adicional en serie, sino que modifica la estructura equivalente del controlador. En particular, el término

$$1 - e^{-\theta_{\circ}s}F_p(\mathrm{s})$$

permite ubicar ceros que compensan polos no deseados de $\widetilde{P}_{\circ}(\mathrm{s})$ dentro de la dinámica de rechazo de perturbaciones.

Bajo condiciones nominales, es decir, cuando el modelo del proceso coincide con la planta real, la transferencia entre la referencia y la salida se mantiene como:

$$H_r^{MSP}(s) = \frac{Y(s)}{R(s)} = \frac{C(s)\widetilde{P}_\circ(s)}{1 + C(s)\widetilde{P}_\circ(s)} e^{-\theta_\circ s} \tag{2-53}$$

Por lo tanto, el MSP conserva la principal ventaja del SP clásico: el tiempo de retardo no aparece en la ecuación característica de seguimiento de referencia.

La diferencia fundamental aparece en las transferencias asociadas a perturbaciones. Para una perturbación de entrada, se obtiene:

$$H_{di}^{MSP}(s) = \widetilde{P}_\circ(s) e^{-\theta_\circ s} \left[1 - \frac{F_p(s)C(s)\widetilde{P}_\circ(s)}{1 + C(s)\widetilde{P}_\circ(s)} e^{-\theta_\circ s}\right] \tag{2-54}$$

$$H_{do}^{MSP}(s) = 1 - \frac{F_p(s)C(s)\widetilde{P}_\circ(s)}{1 + C(s)\widetilde{P}_\circ(s)} e^{-\theta_\circ s} \tag{2-55}$$

De forma equivalente,

$$H_{di}^{MSP}(s) = \widetilde{P}_\circ(s) e^{-\theta_\circ s} \left[\frac{1 + C(s)\widetilde{P}_\circ(s)(1 - e^{-\theta_\circ s}F_p(s))}{1 + C(s)\widetilde{P}_\circ(s)}\right] \tag{2-56}$$

$$H_{do}^{MSP}(s) = \frac{1 + C(s)\widetilde{P}_\circ(s)(1 - e^{-\theta_\circ s}F_p(s))}{1 + C(s)\widetilde{P}_\circ(s)} \tag{2-57}$$

Por tanto, el filtro predictivo puede diseñarse para modificar la dinámica de rechazo de perturbaciones sin alterar la respuesta nominal de referencia. Esta propiedad permite compensar polos lentos o indeseados del proceso, lo cual amplía la aplicabilidad del predictor a plantas estables, integradoras e inestables.

Además del filtro predictivo, puede incorporarse un filtro de referencia con el propósito de mejorar la respuesta ante cambios en el setpoint, principalmente limitando el sobrepaso. Una forma habitual es:

$$F_r(s) = \frac{\tau_r s + 1}{\tau_i s + 1} \tag{2-58}$$

El polo ubicado en $1/\tau_i$ permite cancelar el cero introducido por el controlador PI principal. La elección de τ_r depende del modelo del proceso y de la sintonización específica del controlador, por lo que requiere un ajuste adicional.

3. HIPÓTESIS Y VARIABLES DE LA INVESTIGACIÓN

3.1. Hipótesis

3.1.1. Hipótesis general

El Predictor de Smith Modificado permite obtener un mejor compromiso entre rendimiento y robustez que el controlador PID y el Predictor de Smith clásico en procesos con retardo dominante, especialmente en plantas integradoras e inestables, donde la compensación del retardo y la preservación de la estabilidad del lazo cerrado resultan más exigentes.

3.1.2. Hipótesis específicas

Hipótesis específica 1

La formulación del Predictor de Smith Modificado, mediante una estructura predictiva ajustada y el uso de un filtro predictivo, permite adaptar el Predictor de Smith clásico a procesos con retardo dominante, integradores e inestables, obteniendo lazos cerrados estables y evaluables bajo los criterios definidos en la investigación.

Hipótesis específica 2

El procedimiento propuesto permite evaluar estrategias de control de manera reproducible en distintos escenarios de análisis, al combinar diferentes modelos de proceso, estructuras de control y niveles de robustez y rendimiento deseados bajo una misma lógica de búsqueda, validación y comparación. Esto permite obtener una base consistente para analizar el desempeño relativo de cada estrategia y extender el procedimiento a otros procesos, controladores o condiciones de estudio.

Hipótesis específica 3

El Predictor de Smith Modificado mejora el desempeño ante perturbaciones de entrada y salida frente al controlador PID y al Predictor de Smith clásico, lo cual se refleja en una reducción del error acumulado de la respuesta en el dominio del tiempo, cuantificado mediante el índice (J), principalmente en plantas integradoras e inestables con retardo dominante.

Hipótesis específica 4

El Predictor de Smith Modificado presenta mayor robustez frente al controlador PID y al Predictor de Smith clásico, lo cual se refleja en valores adecuados de M_{ST}, mayores márgenes de retardo y una menor degradación de la respuesta dinámica ante errores en el tiempo de retardo, especialmente en plantas integradoras e inestables con retardo dominante.

3.2. Variables

3.2.1. Variable independiente

La variable independiente del estudio se define como la estrategia de control evaluada, entendida como la arquitectura de control empleada para regular la respuesta del proceso ante cambios en la referencia o perturbaciones. En esta investigación se consideran controladores PI/PID, el Predictor de Smith clásico y el Predictor de Smith Modificado, aplicados a modelos representativos de procesos con retardo dominante.

Tabla 3-1. Operacionalización de la variable independiente del estudio: Estrategia de control evaluada

Tipo	Variable	Dimensiones	Indicador
Independiente	Estrategia de control evaluada	Arquitectura de control	PI, PID, SP, MSP
		Parámetros de sintonía	Ganancia proporcional, tiempo integral, tiempo derivativo y parámetros asociados a la estructura de control
		Condición de aplicación	Tipo de planta evaluada: estable autorregulada, integradora e inestable con retardo dominante

Nota: (a) SP: Predictor de Smith original, (b) MSP: Predictor de Smith modificado.

3.2.2. Variable dependiente

La variable dependiente del estudio se define como el desempeño del sistema de control, entendido como la medición conjunta del rendimiento y la robustez del lazo cerrado. El rendimiento se asocia con la calidad de la respuesta dinámica ante cambios de referencia

o perturbaciones, mientras que la robustez se relaciona con la capacidad del sistema para conservar un comportamiento estable y aceptable frente a incertidumbres, especialmente en el tiempo de retardo.

Tabla 3-2. Operacionalización de la variable dependiente del estudio: Desempeño del sistema de control

Tipo	Variable	Dimensiones	Indicador
Dependiente	Desempeño del sistema de control	Rendimiento	Error acumulado mediante el índice (J), IAE ante perturbaciones de entrada y salida, tiempo de subida, tiempo de estabilización y sobrepaso
		Robustez	Valor real de M_{ST}, margen de retardo, margen de fase, margen de ganancia, ancho de banda y respuesta ante errores en el tiempo de retardo

4. METODOLOGÍA

En este capítulo se describe el enfoque metodológico empleado para diseñar, simular y evaluar las estrategias de control consideradas en la investigación. Primero, se presentan los materiales, herramientas computacionales y modelos matemáticos utilizados. Luego, se define el diseño metodológico, la unidad de análisis y las técnicas de recolección de datos basadas en simulaciones desarrolladas en MATLAB. Finalmente, se detalla el procedimiento seguido para formular las estructuras de control, realizar la búsqueda de sintonías, comparar el rendimiento y la robustez, y evaluar la sensibilidad de los controladores frente a errores en el tiempo de retardo.

4.1. Materiales

Para el desarrollo de la investigación se emplearon fuentes bibliográficas especializadas y herramientas computacionales orientadas al análisis de sistemas de control. Los principales materiales utilizados fueron:

- Artículos científicos, tesis y libros relacionados con control PID, Predictor de Smith, Predictor de Smith Modificado, procesos con retardo y criterios de evaluación de desempeño y robustez.
- Modelos matemáticos representativos de procesos con retardo y dinámicas complejas, ampliamente utilizados en control de procesos para aproximar comportamientos característicos de plantas industriales autorreguladas, integradoras e inestables.
- Software MATLAB para la formulación, simulación, búsqueda de sintonías y cálculo de indicadores de desempeño y robustez.

4.2. Métodos

La investigación emplea un enfoque cuantitativo basado en el análisis matemático y computacional de estrategias de control. El estudio compara controladores PI/PID, Predictor de Smith clásico y Predictor de Smith Modificado mediante indicadores comunes de rendimiento y robustez.

Los métodos considerados son:

- Revisión bibliográfica sobre estructuras de control, modelos de procesos con retardo dominante y criterios de evaluación en el dominio del tiempo y de la frecuencia.
- Formulación de las estructuras de control PI, PID, SP y MSP para los modelos de planta considerados.
- Desarrollo de un algoritmo en MATLAB para la búsqueda de sintonías, validación de estabilidad, cálculo de (M_{ST}), evaluación del índice (J) y comparación de resultados.
- Construcción de curvas (J) vs. (M_{ST}) para caracterizar el compromiso entre rendimiento y robustez.
- Evaluación de las respuestas dinámicas ante cambios en la referencia y perturbaciones de entrada y salida.
- Análisis de robustez mediante indicadores frecuenciales y pruebas frente a errores en el tiempo de retardo.

4.3. Diseño metodológico

El estudio presenta un enfoque cuantitativo, debido a que la comparación entre estrategias de control se realiza mediante indicadores numéricos de desempeño y robustez. El nivel de investigación es explicativo, ya que se busca analizar cómo la estructura de control, el tipo de planta y los niveles de robustez y rendimiento deseados influyen en el comportamiento del sistema controlado.

El diseño de investigación es cuasiexperimental, debido a que las condiciones de análisis se definen mediante modelos matemáticos y simulaciones computacionales. Este enfoque permite evaluar las estrategias de control en escenarios controlados y reproducibles, donde es posible modificar de manera sistemática la estructura de control, la dinámica de la planta, el nivel de robustez y los criterios de evaluación.

4.4. Diseño muestral

La unidad de análisis está conformada por las combinaciones entre modelos de planta, estrategias de control y niveles de robustez y rendimiento deseados. La población de estudio corresponde a estrategias de control aplicables a procesos con retardo dominante,

incluyendo controladores PI/PID y compensadores de retardo basados en el Predictor de Smith.

La muestra es no probabilística, debido a que las plantas y controladores se seleccionan según su relevancia teórica y su compatibilidad con los objetivos de la investigación. Se consideran modelos representativos de dinámicas industriales con retardo, incluyendo plantas estables autorreguladas, integradoras e inestables, así como las estructuras de control PI, PID, SP y MSP.

4.5. Técnicas de recolección de datos

La recolección de datos se realiza mediante simulaciones computacionales desarrolladas en MATLAB. A partir de cada combinación de planta, controlador y condición de evaluación, se registran los parámetros de sintonía obtenidos, la estabilidad del lazo cerrado, el valor real de (M_{ST}), el índice (J), los indicadores temporales y los indicadores frecuenciales.

Las principales técnicas utilizadas son:

- Simulación de respuestas dinámicas ante cambios en la referencia y perturbaciones de entrada y salida.
- Cálculo de índices de desempeño en el dominio del tiempo.
- Cálculo de indicadores de robustez y estabilidad en el dominio de la frecuencia.
- Registro de casos con soluciones válidas, soluciones no convenientes o ausencia de soluciones válidas.
- Organización de resultados mediante curvas (J) vs. (M_{ST}).

4.6. Procedimiento metodológico

El procedimiento metodológico se organiza de acuerdo con los objetivos específicos de la investigación.

4.6.1. Objetivo específico 1: Formulación de las estructuras de control

Se revisan las formulaciones teóricas de los controladores PI/PID, del Predictor de Smith clásico y del Predictor de Smith Modificado. A partir de ello, se definen las estructuras de control a evaluar y su forma de aplicación sobre modelos de procesos con retardo dominante.

Luego, se seleccionan los modelos representativos de planta, considerando dinámicas estables autorreguladas, integradoras e inestables. Para cada caso, se formulan las funciones de transferencia correspondientes y se establecen las relaciones necesarias para construir los lazos de control PI, PID, SP y MSP.

4.6.2. Objetivo específico 2: Desarrollo del procedimiento de búsqueda y comparación

Se implementa en MATLAB un procedimiento computacional para la búsqueda de sintonías de controladores bajo niveles de robustez y rendimiento deseados. Para cada combinación de planta y estructura de control, se evalúa la estabilidad del lazo cerrado, se calcula el valor real de (M_{ST}) y se seleccionan las soluciones válidas según los criterios establecidos.

Los resultados se organizan mediante curvas (J) vs. (M_{ST}), con el fin de caracterizar el compromiso entre rendimiento y robustez. Esta etapa permite identificar zonas de interés y seleccionar casos representativos para una evaluación más detallada en el dominio del tiempo y de la frecuencia.

4.6.3. Objetivo específico 3: Evaluación del desempeño ante cambios de referencia y perturbaciones

Para los controladores seleccionados, se simulan las respuestas del sistema ante cambios en la referencia y perturbaciones de entrada y salida. El desempeño se evalúa en el dominio del tiempo mediante el error acumulado de la respuesta, cuantificado a través del índice (J), y mediante indicadores complementarios como tiempo de subida, tiempo de estabilización, sobrepaso y características generales de la curva de respuesta.

Esta evaluación permite determinar en qué medida el Predictor de Smith Modificado mejora el desempeño frente al controlador PID y al Predictor de Smith clásico, considerando tanto la capacidad de seguimiento de referencia como el rechazo de perturbaciones, especialmente en plantas integradoras e inestables con retardo dominante.

4.6.4. Objetivo específico 4: Evaluación de la robustez

La robustez de las estrategias de control se evalúa mediante indicadores en el dominio de la frecuencia y mediante pruebas frente a errores en el tiempo de retardo. Para ello, se consideran el valor real de (M_{ST}) el margen de retardo, el margen de fase, el margen de

ganancia, el ancho de banda y la degradación de la respuesta dinámica cuando existe diferencia entre el retardo nominal y el retardo real de la planta.

Esta etapa permite comparar la capacidad de cada estrategia para mantener un comportamiento estable y aceptable ante incertidumbres en el tiempo de retardo. Finalmente, se integran los resultados de desempeño y robustez para establecer en qué condiciones el Predictor de Smith Modificado presenta ventajas, limitaciones o desempeño equivalente frente a las estrategias comparadas.

En la **Figura 4-1**, se presenta un diagrama de flujo que sintetiza la integración de los procedimientos descritos para cada uno de los objetivos específicos.

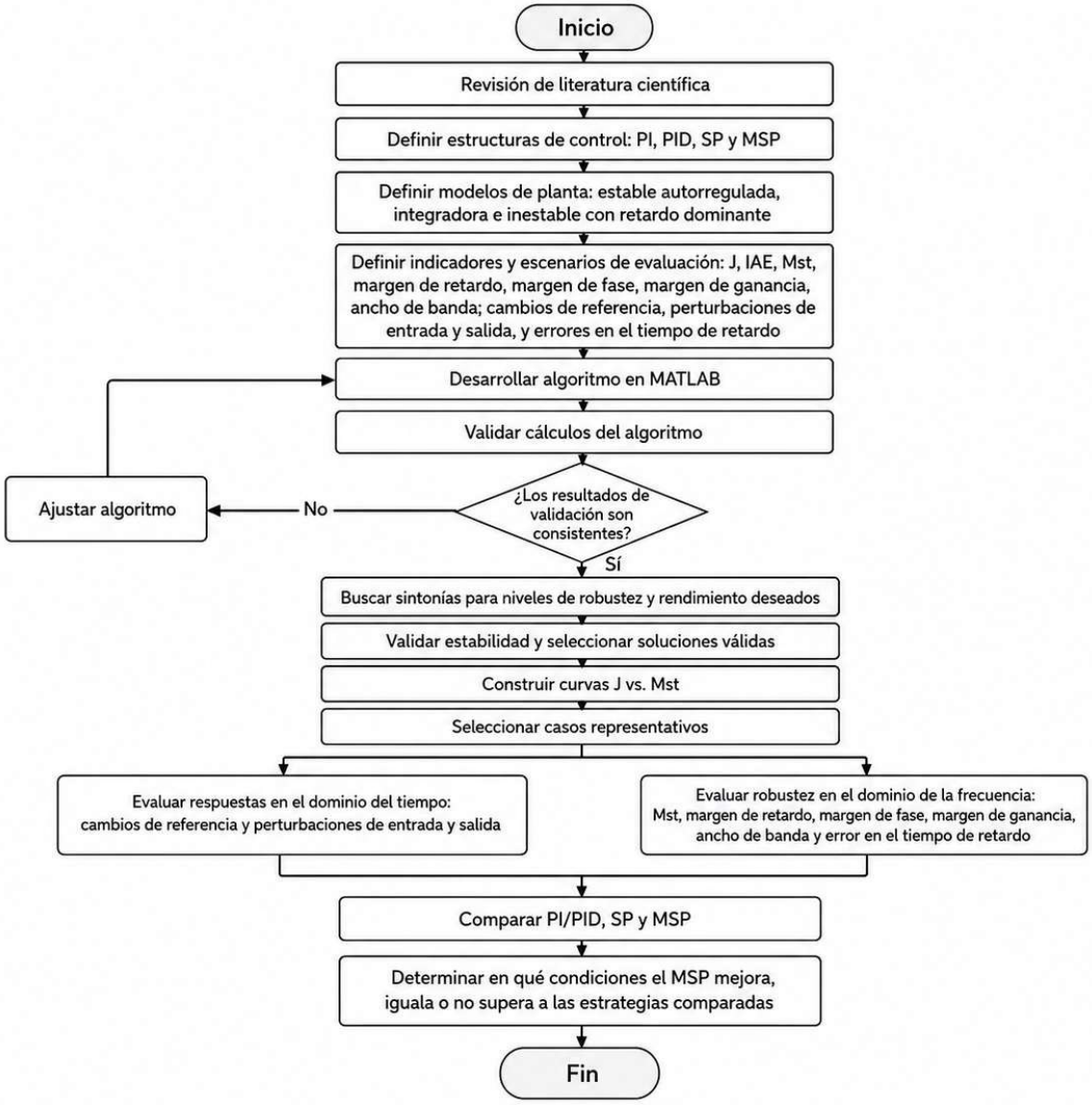

Figura 4-1. Procedimiento metodológico para la evaluación de estrategias de control

4.7. Aspectos éticos

El presente proyecto de tesis ha sido desarrollado respetando los principios de originalidad, integridad académica y propiedad intelectual. Las ideas, formulaciones, modelos, algoritmos y resultados provenientes de trabajos de terceros han sido citados y referenciados según corresponda, evitando cualquier forma de plagio o apropiación indebida de información.

Asimismo, la investigación se desarrolla mediante revisión bibliográfica, modelamiento matemático y simulación computacional, por lo que no involucra experimentación con personas, animales ni el uso de información personal o confidencial. Los modelos de planta, estructuras de control e indicadores empleados se utilizan únicamente con fines académicos y de análisis comparativo.

Finalmente, los resultados obtenidos mediante el algoritmo desarrollado en MATLAB serán reportados de manera objetiva, sin alteración ni selección indebida de datos. En caso de emplearse herramientas de apoyo computacional o inteligencia artificial para la redacción, revisión o adaptación de código, su uso será considerado como asistencia técnica, manteniendo la responsabilidad del autor sobre la verificación, interpretación y presentación final de los resultados.

5. DISEÑO DE LOS SISTEMAS DE CONTROL

En este capítulo, se establecen los **criterios de diseño** que regirán la construcción de los modelos y sistemas de control objeto de estudio.

5.1. Modelos de procesos

En los modelos presentados en la Sección 2.3, la relación entre la constante de tiempo del proceso y el tiempo de retardo constituye una medida importante de la complejidad dinámica del sistema. En este trabajo, dicha relación se define como el ratio de control:

$$\varphi = \frac{\tau}{\theta} \tag{5-1}$$

En particular, valores pequeños de φ indican que el tiempo de retardo es significativo en comparación con la dinámica propia del proceso, lo que incrementa la dificultad de control. Por el contrario, valores elevados de φ corresponden a procesos donde la constante de tiempo domina sobre el retardo, por lo que el control resulta menos exigente.

A continuación, en la **Tabla 5-1**, se presentan los modelos considerados para las pruebas posteriores de este estudio. Estos modelos fueron seleccionados a partir de referencias ampliamente utilizadas en la literatura sobre control de procesos con retardo y predictores de Smith (Freitas Machado et al., 2019; Grimholt & Skogestad, 2012, 2013; Holene, 2013; Skogestad & Grimholt, 2018).

Tabla 5-1. Modelos de planta seleccionados para la evaluación de los controladores

Orden	Tipo	Descripción	φ	Función de transferencia
P_1	Estable	Tiempo de retardo puro	0	e^{-s}
P_2		Dinámica balanceada	1	$\frac{e^{-s}}{s+1}$
P_3		Dinámica dominante	8	$\frac{e^{-s}}{8s+1}$
P_4		Próximo al integrador	20	$\frac{e^{-s}}{20s+1}$
P_5	Integrador	Integrador	∞	$\frac{e^{-s}}{s}$

Orden	Tipo	Descripción	φ	Función de transferencia
P_6	Inestable	Dinámica balanceada	1	$\frac{e^{-s}}{s-1}$
P_7		Dinámica dominante	8	$\frac{e^{-s}}{8s-1}$

5.2. Diseño de los sistemas de control

En esta sección se presentan las estructuras de control que serán utilizadas en la evaluación comparativa de la investigación. Primero, se define el controlador PID seleccionado como estrategia convencional de referencia. Luego, se desarrolla el control basado en el Predictor de Smith clásico, considerando su formulación equivalente y las funciones de transferencia asociadas a referencia y perturbaciones. Finalmente, se formula la estrategia basada en el Predictor de Smith Modificado, incorporando el filtro predictivo y las expresiones particulares requeridas para procesos estables, integradores e inestables con retardo.

5.2.1. Control PID

La estructura PID seleccionada para el desarrollo de las pruebas de este estudio es la presentada en (2-42).

$$C(s) = k_c\left(\frac{\tau_i s + 1}{\tau_i s}\right)\left(\frac{\tau_d s + 1}{\tau_f s + 1}\right) \tag{2-42}$$

La inclusión de τ_f como variable independiente dentro del algoritmo de búsqueda añadiría un grado de libertad adicional a la sintonización, pasando de tres a cuatro parámetros ajustables. Con el fin de evitar este incremento en la dimensionalidad del problema, τ_f se define en función del tiempo derivativo:

$$\tau_f = \gamma\tau_d \tag{5-2}$$

donde γ es el factor que determinará la influencia del filtro en el desempeño del controlador.

Grimholt y Skogestad (2013) señalan que el filtro derivativo no debe modificar de manera significativa las características de desempeño y robustez del controlador. Por ello,

recomiendan seleccionar τ_f suficientemente pequeño respecto a τ_d, típicamente no mayor que una tercera parte del tiempo derivativo. Si τ_f se mantuviese como parámetro sintonizable, la estructura se convertiría en un controlador FPID (PID filtrado). En este estudio se emplea:

$$\gamma = 0.1 \tag{5-3}$$

Además, para evitar problemas numéricos cuando τ_d toma valores muy pequeños, se impone un valor mínimo para el filtro:

$$\tau_f = \max(0.1\tau_d; 0.001) \tag{5-4}$$

A partir de esta estructura general se obtienen los controladores utilizados en las comparaciones. El controlador PI se obtiene anulando la acción derivativa, es decir, tomando $\tau_d = 0$. El controlador P se obtiene omitiendo tanto la acción integral como la derivativa, quedando $C(s) = k_c$.

Para evaluar el control de los modelos de la **Tabla 5-1**, se aplicarán controladores PI y PID en todos los casos. No se incluirá el controlador proporcional puro, P, como una estrategia independiente de comparación, debido a que su estructura no incorpora acción integral ni derivativa. Por ello, presenta una capacidad limitada para eliminar error estacionario, rechazar perturbaciones y modificar la dinámica transitoria del sistema. En consecuencia, su inclusión no aportaría información relevante frente a las alternativas PI y PID consideradas en este estudio.

5.2.2. Control basado en el Predictor de Smith

La formulación presentada en esta sección se desarrolla a partir de la estructura clásica del Predictor de Smith y de su tratamiento en la literatura sobre compensadores de tiempo de retardo. En particular, se toma como base la descripción del Predictor de Smith y de sus propiedades de lazo cerrado presentada por Normey-Rico y Camacho (2007), así como la revisión de compensadores de tiempo de retardo desarrollada por Normey-Rico y Camacho (2008a). En el presente trabajo, las ecuaciones se reescriben con la notación adoptada para el controlador ($C(s)$), el modelo libre de retardo ($\tilde{P}_0(s)$), el modelo con tiempo de retardo ($\tilde{P}(s)$), el tiempo de retardo nominal (θ_0) y el retardo real (L). Por tanto, las expresiones siguientes deben entenderse como una adaptación notacional de la

formulación reportada en la literatura, empleada aquí para establecer una base común de comparación con los controladores PI/PID y con el Predictor de Smith Modificado.

A partir de la **Figura 2-5** de la Sección 2.5, se desarrollan las funciones de transferencia que relacionan la salida del proceso con la referencia y las perturbaciones de entrada y salida, respectivamente.

La estructura del Predictor de Smith factoriza la planta en dos partes: $\widetilde{P}_{\circ}(\mathrm{s})$ y $e^{-\theta_{\circ}s}$. Se puede obtener una estructura alternativa del Predictor. De este modo, se define $C_{\circ}(\mathrm{s})$

$$C_{\circ}(\mathrm{s}) = \frac{\mathrm{C(s)}}{1 + \widetilde{P}_{\circ}(\mathrm{s})C(s)} \tag{5-5}$$

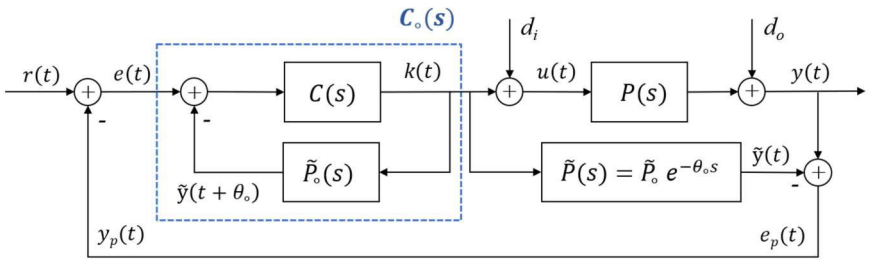

Figura 5-1. Estructura del Predictor de Smith utilizada en el diseño del controlador SP

Nota. Adaptado de Normey-Rico & Camacho (2007).

Si despejamos el término $\tilde{P}(\mathrm{s})$ de la estructura y conectamos con $C_{\circ}(\mathrm{s})$, se obtiene una función de transferencia equivalente denominada controlador Predictor Smith.

$$\tilde{C}(\mathrm{s}) = \frac{\mathrm{C(s)}}{1 + C(s)\widetilde{P}_{\circ}(\mathrm{s})(1 - e^{-\theta_{\circ}s})} \tag{5-6}$$

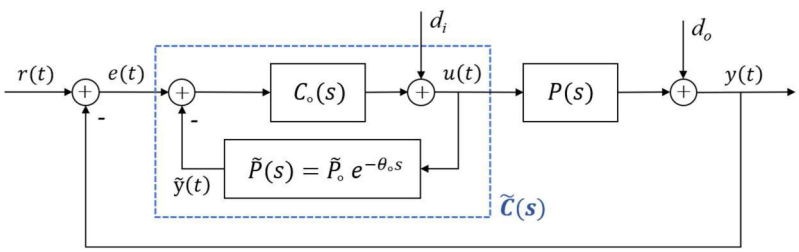

Figura 5-2. Estructura equivalente del controlador basado en Predictor de Smith

Nota 1. Adaptado de Normey-Rico & Camacho (2007).

A partir de la **Figura 5-2**, se obtiene la siguiente relación entre la salida de la planta y el valor de referencia,

$$\frac{Y(\mathrm{s})}{R(s)} = \frac{\tilde{C}(\mathrm{s})P(\mathrm{s})}{1 + \tilde{C}(\mathrm{s})P(\mathrm{s})} \tag{5-7}$$

Reemplazando (5-6) en (5-7),

$$\frac{Y(\mathrm{s})}{R(s)} = \frac{\mathrm{C}(\mathrm{s})P(\mathrm{s})}{1 + \mathrm{C}(\mathrm{s})\left[\widetilde{P_\circ}(\mathrm{s}) - \tilde{P}(\mathrm{s}) + P(\mathrm{s})\right]} \tag{5-8}$$

Si existe una identificación perfecta del proceso, entonces, $P(\mathrm{s}) = \tilde{P}(\mathrm{s})$. De este modo, se define H_r^{SP} como la función de transferencia que relaciona la salida del proceso y el setpoint en una estructura SP,

$$H_r^{SP}(s) = \frac{Y(\mathrm{s})}{R(s)} = \frac{\mathrm{C}(\mathrm{s})\tilde{P}(\mathrm{s})}{1 + \mathrm{C}(\mathrm{s})\widetilde{P_\circ}(\mathrm{s})}$$

$$H_r^{SP}(s) = \frac{\mathrm{C}(\mathrm{s})\widetilde{P_\circ}(\mathrm{s})}{1 + \mathrm{C}(\mathrm{s})\widetilde{P_\circ}(\mathrm{s})} e^{-\theta_\circ s} \tag{5-9}$$

Para el caso de una perturbación de entrada, también a partir de la **Figura 5-2**, se tiene la siguiente relación entre la salida del sistema y señal de perturbación

$$\frac{Y(\mathrm{s})}{D_i(s)} = \frac{P(\mathrm{s})}{1 + \tilde{C}(\mathrm{s})P(\mathrm{s})} \tag{5-10}$$

Reemplazando (5-6) en (5-10), se define H_{di}^{SP} como la función de transferencia que relaciona la salida del proceso y la perturbación de entrada en una estructura SP,

$$H_{di}^{SP}(s) = \frac{Y(\mathrm{s})}{D_i(s)} = \tilde{P}(\mathrm{s})\left[1 - \frac{\mathrm{C}(\mathrm{s})\tilde{P}(\mathrm{s})}{1 + \mathrm{C}(\mathrm{s})\widetilde{P_\circ}(\mathrm{s})}\right]$$

$$H_{di}^{SP}(s) = \widetilde{P_\circ}(\mathrm{s})e^{-\theta_\circ s}\left[1 - \frac{\mathrm{C}(\mathrm{s})\widetilde{P_\circ}(\mathrm{s})}{1 + \mathrm{C}(\mathrm{s})\widetilde{P_\circ}(\mathrm{s})} e^{-\theta_\circ s}\right] \tag{5-11}$$

En (5-11), el primer tiempo de retardo representaría el retraso para que el efecto de la perturbación sea observable y el segundo (dentro de corchetes), el retraso para que la acción del controlador sea observable.

De forma similar, se obtiene la función de transferencia para la perturbación de salida.

$$H_{do}^{SP}(s) = 1 - \frac{\mathrm{C(s)}\widetilde{P}_{\circ}(\mathrm{s})}{1 + \mathrm{C(s)}\widetilde{P}_{\circ}(\mathrm{s})} e^{-\theta_{\circ} s} \tag{5-12}$$

Tomando (2-3) y (2-4), para S y T, y reemplazando C por el controlador Predictor de Smith ($\tilde{C}(\mathrm{s})$) de (5-6), se obtienen expresiones ajustadas a este sistema de control que permitirán el cálculo de sus indicadores de robustez.

$$S_{SP} = 1 - \frac{\mathrm{C(s)}\widetilde{P}_{\circ}(\mathrm{s})}{1 + \mathrm{C(s)}\widetilde{P}_{\circ}(\mathrm{s})} e^{-\theta_{\circ} s} \tag{5-13}$$

$$T_{SP} = \mathrm{C(s)}\widetilde{P}_{\circ}(\mathrm{s}) e^{-\theta_{\circ} s} \left[1 - \frac{\mathrm{C(s)}\widetilde{P}_{\circ}(\mathrm{s})}{1 + \mathrm{C(s)}\widetilde{P}_{\circ}(\mathrm{s})} e^{-\theta_{\circ} s}\right] \tag{5-14}$$

Por otro lado, con el objetivo de evaluar la influencia de errores de modelado del tiempo de retardo, se define L como el tiempo de retardo del modelo del proceso, siendo este α veces el tiempo de retardo real $\theta_{\circ}$,

$$P(\mathrm{s}) = \widetilde{P}_{\circ}(\mathrm{s})\, e^{-Ls}$$

$$L = \alpha * \theta_{\circ} \tag{5-15}$$

De esta forma, se reformulan (5-9), (5-11) y (5-12),

$$H_{r}^{SP}(s) = \mathrm{C(s)}\widetilde{P}_{\circ}(\mathrm{s}) e^{-Ls} \left[\frac{1}{1 + \mathrm{C(s)}\widetilde{P}_{\circ}(\mathrm{s})(1 - e^{-\theta_{\circ} s} + e^{-Ls})}\right] \tag{5-16}$$

$$H_{di}^{SP}(s) = \left[1 + \mathrm{C(s)}\widetilde{P}_{\circ}(\mathrm{s})\left(1 - e^{-\theta_{\circ} s}\right)\right] \left[\frac{\widetilde{P}_{\circ}(\mathrm{s}) e^{-Ls}}{1 + \mathrm{C(s)}\widetilde{P}_{\circ}(\mathrm{s})(1 - e^{-\theta_{\circ} s} + e^{-Ls})}\right] \tag{5-17}$$

$$H_{do}^{SP}(s) = \left[1 + \mathrm{C(s)}\widetilde{P}_{\circ}(\mathrm{s})\left(1 - e^{-\theta_{\circ} s}\right)\right] \left[\frac{1}{1 + \mathrm{C(s)}\widetilde{P}_{\circ}(\mathrm{s})(1 - e^{-\theta_{\circ} s} + e^{-Ls})}\right] \tag{5-18}$$

Para evaluar el control de los modelos de la **Tabla 5-1**, el controlador principal del Predictor de Smith se considera de tipo PI. La excepción corresponde al modelo integrador, para el cual se emplea un controlador proporcional. Esta elección se debe a que el proceso integrador ya posee un polo en el origen; por tanto, añadir acción integral

en el controlador introduciría un segundo integrador en el lazo, reduciendo los márgenes de estabilidad y aumentando la sensibilidad frente al tiempo de retardo.

Asimismo, no se incorpora acción derivativa en el controlador principal del Predictor de Smith. La acción derivativa puede interpretarse como un mecanismo predictivo local, basado en la extrapolación de la tendencia del error. Sin embargo, la propia estructura del Predictor de Smith ya cumple una función predictiva al compensar explícitamente el efecto del tiempo de retardo mediante el modelo del proceso. Por esta razón, incluir simultáneamente acción derivativa y predicción de Smith aumentaría la complejidad de la sintonización sin aportar una mejora estructural clara para los objetivos de este estudio.

5.2.3. Control basado en el Predictor de Smith Modificado

La formulación del Predictor de Smith Modificado considerada en esta investigación se basa en las extensiones del Predictor de Smith orientadas al control de procesos estables, integradores e inestables con tiempo de retardo. Estas estructuras han sido descritas y sistematizadas en la literatura de compensadores de retardo, particularmente por Normey-Rico y Camacho (2007, 2008a), y posteriormente empleadas en reglas unificadas de sintonización para procesos con retardo por Normey-Rico y Guzmán (2013). En este trabajo, dichas expresiones se adaptan a la notación utilizada para el controlador principal ($C(s)$), el modelo libre de retardo ($\tilde{P}_0(s)$), el filtro predictivo ($F_p(\mathrm{s})$), el filtro de referencia ($F_r(\mathrm{s})$), el tiempo de retardo nominal (θ_0) y el retardo real (L). En consecuencia, las ecuaciones presentadas no constituyen una formulación original independiente, sino una adaptación de estructuras reportadas en la literatura para su aplicación al procedimiento comparativo desarrollado en esta tesis.

Las expresiones de $F_p(s)$, τ_o y β utilizadas en las secciones siguientes se adoptan de la metodología de sintonización propuesta por Normey-Rico y Guzmán (2013), y se reescriben con la notación empleada en esta investigación.

La estructura MSP incorpora un filtro predictivo $F_p(\mathrm{s})$, cuyo diseño depende del tipo de proceso considerado: estable, integrador o inestable. En procesos estables, el objetivo principal de $F_p(\mathrm{s})$ es mejorar el rechazo de perturbaciones y establecer un compromiso adecuado entre desempeño y robustez. En este caso, la estabilidad nominal del lazo cerrado puede garantizarse mediante una sintonización apropiada del controlador principal $C(s)$. En cambio, para procesos integradores e inestables, el filtro predictivo

cumple además una función estructural más crítica, ya que permite cancelar polos indeseados presentes en las funciones de transferencia asociadas al rechazo de perturbaciones.

En este estudio, no se considerará la incidencia del filtro de referencia, por lo que se asumirá ($F_r(s) = 1$). Esta decisión responde a que el objetivo principal del análisis es evaluar el comportamiento del SP y del MSP frente a perturbaciones de entrada y salida, las cuales constituyen una de las principales limitaciones de la estructura clásica del Predictor de Smith. Además, como se indicó previamente, el seguimiento de referencia representa una de las principales ventajas del SP bajo condiciones nominales; por ello, el estudio se concentra en la dinámica de regulación y rechazo de perturbaciones.

A continuación, se presentan las expresiones generales del Predictor de Smith modificado y las consideraciones particulares utilizadas para cada tipo de planta.

5.2.4. Ecuaciones generales del MSP

El proceso real se representa como:

$$P(\mathrm{s}) = \widetilde{P}_{\circ}(\mathrm{s})\, e^{-Ls}$$

donde $\widetilde{P}_{\circ}(\mathrm{s})$ corresponde a la parte del modelo libre de tiempo de retardo y L representa el retardo real del proceso. Para el caso nominal se considera:

$$L = \alpha * \theta_{\circ}$$

donde $\theta_{\circ}$ es el tiempo de retardo nominal y α permite representar posibles desajustes en el retardo.

Las funciones de transferencia del MSP quedan dadas por:

$$H_r^{MSP}(s) = \widetilde{P}_{\circ}(\mathrm{s})\, e^{-Ls}\left[\frac{\mathrm{C(s)}}{1 + \mathrm{C(s)}\widetilde{P}_{\circ}(\mathrm{s})(1 + F_p(e^{-Ls} - e^{-\theta_{\circ}s}))}\right] \tag{5-19}$$

$$H_{di}^{MSP}(s) = \widetilde{P}_{\circ}(\mathrm{s})\, e^{-Ls}\left[\frac{1 + \mathrm{C(s)}\widetilde{P}_{\circ}(\mathrm{s})\left(1 - F_p e^{-\theta_{\circ}s}\right)}{1 + \mathrm{C(s)}\widetilde{P}_{\circ}(\mathrm{s})(1 + F_p(e^{-Ls} - e^{-\theta_{\circ}s}))}\right] \tag{5-20}$$

$$H_{do}^{MSP}(s) = \left[\frac{1 + \mathrm{C(s)}\widetilde{P}_{\circ}(\mathrm{s})\left(1 - F_p e^{-\theta_{\circ}s}\right)}{1 + \mathrm{C(s)}\widetilde{P}_{\circ}(\mathrm{s})(1 + F_p(e^{-Ls} - e^{-\theta_{\circ}s}))}\right] \tag{5-21}$$

En estas expresiones, H_r^{MSP} relaciona la salida con la referencia, H_{di}^{MSP} relaciona la salida con una perturbación de entrada y H_{do}^{MSP} relaciona la salida con una perturbación de salida.

5.2.5. Procesos estables

Para una planta estable, el filtro predictivo F_p se utiliza principalmente para mejorar el rechazo de perturbaciones y ajustar el compromiso entre desempeño y robustez. Como la planta no posee polos inestables, la estabilidad nominal del lazo cerrado puede garantizarse mediante una sintonización adecuada del controlador principal $C(s)$.

El filtro se define como:

$$F_p(s) = \frac{\beta s + 1}{\tau_o s + 1} \tag{5-22}$$

El parámetro β se escoge para ubicar el cero del filtro en una posición que compense el polo dominante de la planta. Para una planta nominal de primer orden sin tiempo de retardo:

$$\widetilde{P}_\circ(\mathrm{s}) = \frac{k_p}{\tau s + 1}$$

y un controlador PI:

$$C(\mathrm{s}) = k_c\left(\frac{\tau_i s + 1}{\tau_i s}\right)$$

se impone la condición:

$$C(s)\widetilde{P}_\circ(\mathrm{s})(1 - e^{-\theta_\circ s}F_p(\mathrm{s}))=0 \tag{5-23}$$

Sustituyendo las expresiones anteriores:

$$k_c\left(\frac{\tau_i s + 1}{\tau_i s}\right)\left(\frac{k_p}{\tau s + 1}\right)\left[1 - e^{-\theta_\circ s}\left(\frac{\beta s + 1}{\tau_o s + 1}\right)\right] = 0$$

Si $k_c k_p$, $(\tau_i s + 1)$ y $(\tau_o s + 1)$ son distintos de cero, y multiplicando $\tau_i s(\tau s + 1)(\tau_o + 1)$,

$$(\tau_o s + 1) - e^{-\theta_\circ s}(\beta s + 1) = 0$$

y despejando β,

$$\beta = \left(\frac{1+\tau_o s}{e^{-\theta_\circ s}} - 1\right)\frac{1}{s}$$

Definiendo β para que el filtro tenga un cero en polo dominante de la planta, se evalúa en $s = -1/\tau$, por tanto:

$$\beta = \tau\left[1 - \left(1 - \frac{\tau_o}{\tau}\right)e^{-\frac{\theta_\circ}{\tau}}\right] \tag{5-24}$$

Con esta elección, el término $\beta s + 1$ permite compensar el polo dominante asociado a $\tau s + 1$, mientras que $\tau_o s + 1$ permite regular el compromiso entre rapidez de rechazo de perturbaciones y robustez.

De este modo, $\tilde{C}(\mathrm{s})$ se reduce a la siguiente forma

$$\tilde{C}(\mathrm{s}) = \frac{C(s)F_p(s)}{1 + \cancel{C(s)\tilde{P}_\circ(s)(1 - e^{-\theta_\circ s}F_p(s))}}$$

$$\tilde{C}(s) \cong C(s)F_p(s)$$

$$\tilde{C}(s) = k_c\left(\frac{\tau_i s + 1}{\tau_i s}\right)\left(\frac{\beta s + 1}{\tau_o s + 1}\right)$$

<u>Selección del parámetro τ_o</u>

La inclusión de F_p no debe convertir la sintonización del MSP en un problema con demasiados grados de libertad. Por ello, τ_o no se considera como una variable adicional dentro del algoritmo de búsqueda, sino que se fija mediante la regla propuesta por Normey-Rico & Guzmán (2013):

$$\tau_o = 0.42\theta_o + 0.35\tau \tag{5-25}$$

Esta elección busca proporcionar un equilibrio entre desempeño y robustez ($M_{ST} = 1.59$), manteniendo el filtro predictivo como una parte estructural del MSP y no como un parámetro libre adicional de optimización. En consecuencia, para cada planta, el ajuste principal queda determinado por los parámetros del controlador $C(s)$, principalmente k_c y τ_i.

5.2.6. Plantas integradoras

En la **Sección 2.5.1** se indicó que una de las principales limitaciones del Predictor de Smith clásico para procesos integradores es su incapacidad para eliminar adecuadamente el polo en $s = 0$ presente en la planta. Esta limitación aparece incluso cuando el controlador principal $C(s)$ posee acción integral, debido a que la estructura equivalente del SP puede comportarse como un controlador proporcional en estado estacionario.

Para superar esta dificultad, en el Predictor de Smith modificado el filtro predictivo F_p se diseña de manera que el término de (5-23**)** se anule en $s = 0$. De este modo, el filtro permite compensar el polo integrador de la planta dentro de la transferencia asociada al rechazo de perturbaciones y preservar la capacidad de regulación en estado estacionario.

Manteniendo la misma estructura utilizada para procesos estables, el filtro predictivo para una planta integradora se define como:

$$F_p(s) = \frac{\beta s + 1}{\tau_o s + 1} \tag{5-26}$$

De forma análoga al caso de los procesos estables, para procesos integradores se considera un controlador principal proporcional $C(s) = k_c$ y un modelo IPDT sin tiempo de retardo:

$$\widetilde{P}_\circ(\mathrm{s}) = \frac{k_v}{s}$$

Al sustituir los términos en (5-23**)**

$$k_c\left(\frac{k_v}{\mathrm{s}}\right)\left[1 - e^{-\theta_\circ s}\left(\frac{\beta s + 1}{\tau_o s + 1}\right)\right] = 0 \tag{5-27}$$

En esta expresión, el objetivo es que el término entre corchetes tenga un cero en $s = 0$. De esta manera, dicho cero compensa el polo integrador del modelo IPDT y evita que este domine la dinámica de rechazo de perturbaciones.

Si $k_c k_v$ y $(\tau_o s + 1)$ son distintos de cero en $s = 0$ y multiplicando por $\mathrm{s}(\tau_o s + 1)$,

$$(\tau_o s + 1) - e^{-\theta_\circ s}(\beta s + 1) = 0$$

Derivando la expresión anterior para obtener β en función de $\tau_o \; y \; \theta_\circ$

$$\frac{d}{ds}\left[(\tau_o s + 1) - e^{-\theta_\circ s}(\beta s + 1)\right] = \frac{d}{ds}0$$

$$\tau_o - \left[-\theta_\circ e^{-\theta_\circ s}(\beta s + 1) + \beta e^{-\theta_\circ s}\right] = 0$$

Evaluando la expresión anterior en $s = 0$, se define β para las plantas integradoras en lazo abierto

$$\beta = \tau_o + \theta_\circ \tag{5-28}$$

Selección del parámetro τ_o

De forma similar al caso de la planta estable, τ_o se sintoniza para establecer un equilibrio entre rendimiento y robustez. De este modo, se define τ_o mediante la regla de sintonización de Normey-Rico & Guzmán (2013).

$$\tau_o = 2.26\theta_\circ + 0.14 \tag{5-29}$$

Esta elección busca proporcionar un equilibrio entre desempeño y robustez ($M_{ST} = 1.59$), manteniendo el filtro predictivo como una parte estructural del MSP y no como un parámetro libre adicional de optimización. En consecuencia, para cada planta, el ajuste principal queda determinado por los parámetros del controlador $C(s)$, k_c para este tipo de plantas.

5.2.7. Plantas inestables

En el caso de plantas inestables, el filtro predictivo debe compensar los polos ubicados en el semiplano derecho, RHP. Para un proceso UFOPDT, la parte del modelo libre de tiempo de retardo puede expresarse como:

$$\widetilde{P_\circ}(\mathrm{s}) = \frac{k_p}{\tau s - 1}$$

De acuerdo con la estructura recomendada por Normey-Rico & Guzmán (2013), el filtro predictivo se define como:

$$F_p(s) = \frac{(\beta s + 1)}{1 + \tau_o s\left(2 + \frac{\tau_o}{\tau}\right)} \tag{5-30}$$

En este caso, el parámetro β se selecciona para que el término (5-23) se anule en $s = 1/\tau$. De este modo, dicho cero permite compensar el polo inestable de $\widetilde{P}_{\circ}(s)$ dentro de la transferencia equivalente de rechazo de perturbaciones.

Siendo $C(s)$ un controlador PI y $\widetilde{P}_{\circ}(s)$, un modelo del tipo UFOPDT sin el término del tiempo de retardo,

$$k_c\left(\frac{\tau_i s+1}{\tau_i s}\right)\left[1-e^{-\theta_{\circ} s}\left(\frac{\beta s+1}{1+\tau_o s\left(2+\frac{\tau_o}{\tau}\right)}\right)\right]=0 \tag{5-31}$$

Si k_c, $(\tau_i s+1)$ y $1+sT_o\left(2+\frac{T_o}{T}\right)$ son distintos de cero, y despejando β

$$1+\tau_o s\left(2+\frac{\tau_o}{\tau}\right)-e^{-\theta_{\circ} s}(\beta s+1)=0$$

$$\beta=\left[\frac{1+\tau_o s\left(2+\frac{\tau_o}{\tau}\right)}{e^{-\theta_{\circ} s}}-1\right]\frac{1}{s} \tag{5-32}$$

Reemplazando $s = 1/\tau$, se define β para las plantas inestables en lazo abierto,

$$\beta=\left[\frac{1+\tau_o(\frac{1}{\tau})\left(2+\frac{\tau_o}{\tau}\right)}{e^{-\theta_{\circ}(\frac{1}{\tau})}}-1\right]\frac{1}{(\frac{1}{\tau})}$$

$$\beta=\tau\left[e^{\frac{\theta_{\circ}}{\tau}}\left(1+\frac{\tau_o}{\tau}\right)^2-1\right] \tag{5-33}$$

Esta expresión define el valor de β para plantas inestables en lazo abierto.

Selección del parámetro τ_o

De forma similar a lo indicado para las plantas estables e integradoras, τ_o se sintoniza para establecer un equilibrio entre rendimiento y robustez. De este modo, se define τ_o definido mediante la regla de sintonización de Normey-Rico & Guzmán (2013),

$$\tau_o = 0.1342\tau e^{2.382\theta_{\circ}/\tau} \tag{5-34}$$

Esta elección busca proporcionar un equilibrio entre desempeño y robustez ($M_{ST} = 1.59$), manteniendo el filtro predictivo como una parte estructural del MSP y no como un

parámetro libre adicional de optimización. En consecuencia, para cada planta, el ajuste principal queda determinado por los parámetros del controlador $C(s)$, principalmente k_c y τ_i.

A continuación, se presentan los parámetros del Predictor de Smith Modificado calculados para cada una de las plantas bajo estudio. Estos valores se obtienen a partir de las expresiones desarrolladas en las secciones previas, considerando el tipo de proceso correspondiente: estable, integrador o inestable.

Tabla 5-2. Parámetros del filtro predictivo de la estructura MSP calculados para cada modelo de planta

		k_p, k_v		τ	φ	τ	β
Estable	P_1: Tiempo de retardo puro	1	1	0.005	0	0.42	0.005
	P_2: Dinámica balanceada	1	1	1	1	0.77	0.915
	P_3: Dinámica dominante	1	1	8	8	3.22	3.782
	P_4: Cerca al integrador	1	1	20	20	7.42	8.034
Integrador	P_5: Integrador	1	1	1	1	2.40	3.400
Inestable	P_6: Dinámica balanceada	1	1	1	1	1.45	15.355
	P_7: Dinámica dominante	1	1	8	8	1.45	4.638

6. RESULTADOS Y ANÁLISIS DE RESULTADOS

En este capítulo se presentan los resultados obtenidos mediante el programa de evaluación de estrategias de control desarrollado en MATLAB. El análisis se organiza por planta, considerando los modelos P1 a P7 y las estrategias PI, PID, Predictor de Smith clásico y Predictor de Smith Modificado. Para cada caso, se evalúa el compromiso entre rendimiento y robustez mediante curvas Jvs. M_{ST}, parámetros de sintonía, indicadores temporales, márgenes de estabilidad y sensibilidad frente a errores en el tiempo de retardo. Finalmente, se interpreta el comportamiento de cada estrategia con el fin de establecer en qué condiciones el MSP representa una mejora, una alternativa equivalente o una opción menos conveniente frente a las estrategias comparadas.

6.1. Programa de evaluación de las estrategias de control

El programa de evaluación propuesto se organiza como una secuencia integrada orientada a evaluar el desempeño de las estrategias PI, PID, SP y MSP. En este contexto, el desempeño se entiende como la valoración conjunta del rendimiento y la robustez. El rendimiento se cuantifica principalmente mediante el índice J, construido a partir de los errores acumulados ante perturbaciones de entrada y salida; mientras que la robustez se evalúa mediante M_{ST} y márgenes de estabilidad. De esta forma, el procedimiento no persigue únicamente la minimización del error, sino la identificación del mejor compromiso entre rapidez de respuesta, rechazo de perturbaciones y reserva de estabilidad.

En el Anexo 5 se adjunta el script en lenguaje MATLAB del programa desarrollado para la evaluación de las estrategias de control.

6.1.1. Definición del contexto de la búsqueda

En esta etapa se seleccionan las plantas y las estructuras de control incluidas en la comparación. Asimismo, se define el controlador base empleado para obtener los pesos de normalización IAE_{do}^{0} e IAE_{di}^{0}, así como el intervalo de valores objetivo de M_{ST}. Esta definición garantiza que todos los controladores sean evaluados bajo condiciones comunes de rendimiento y robustez.

Tabla 6-1. Criterios para la evaluación de los sistemas de control: Definición del contexto de la búsqueda

Parámetro	Valor adoptado	Comentario
Plantas del estudio	P1-P7	El usuario puede seleccionar una, varias o todas.
Estrategias comparadas	PI, PID, SP, MSP	En el código corresponden a PI, PID, SP y MSP.
Controlador base para pesos	PI	El usuario puede cambiarlo a PID.
Robustez base de normalización	M_{ST} base= 1.59	Se usa para construir IAE_{do}^{0} e IAE_{di}^{0}
Plantas del estudio	P1-P7	El usuario puede seleccionar una, varias o todas.

6.1.2. Búsqueda de sintonías por nivel de robustez

Para cada planta, controlador y valor objetivo de M_{ST}, el procedimiento explora el espacio de sintonización y verifica la existencia de soluciones estables y próximas al nivel de robustez solicitado. La búsqueda combina una exploración global inicial con refinamientos locales sucesivos, lo que permite localizar regiones prometedoras sin evaluar uniformemente todo el espacio de parámetros. En esta etapa no se selecciona simplemente la sintonía más rápida, sino aquella que produce el menor valor de J dentro de una condición de robustez dada.

Tabla 6-2. Criterios para la evaluación de los sistemas de control: Búsqueda de sintonías

Parámetro	Valor adoptado	Comentario
Intervalo de barrido en M_{ST}	1.0 a 2.0	El usuario puede cambiar el rango de M_{ST}
Paso del barrido principal	0.2	Definido por el usuario.
Paso de exploración global	0.2	Resolución inicial del escaneo.

Parámetro	Valor adoptado	Comentario
Secuencia de refinamiento general	[1, 0.5, 0.25, 0.10, 0.05, 0.02, 0.01]	Aplica a PI, SP y MSP.
Secuencia de refinamiento para PID	[1, 0.5, 0.25, 0.10, 0.05, 0.02]	Refinamiento específico para PID.
Máximo de iteraciones vacías consecutivas	4	Criterio de parada temprana.

6.1.3. Construcción de curvas rendimiento-robustez

A partir de las sintonías obtenidas se construyen las curvas J vs M_{ST}. Estas curvas permiten analizar la variación del rendimiento del controlador al imponer distintos niveles de robustez. Dado que esta evaluación se realiza bajo condiciones nominales de modelado, la comparación principal representa el compromiso rendimiento-robustez sin incorporar todavía desajustes entre la planta y su modelo.

Tabla 6-3. Criterios para la evaluación de los sistemas de control: Construcción de curvas J vs M_{ST}

Parámetro	Valor adoptado	Comentario
Índice de desempeño	J = 0.5*(IAE_{do}/ IAE_{do}^{0}+ IAE_{di}/ IAE_{di}^{0})	Medida principal de rendimiento.
Horizonte temporal para IAE	30 s	Usado en la evaluación nominal.
Paso temporal de simulación	0.01 s	Usado en el cálculo de IAE.
Perturbación de salida	1	Valor nominal.
Perturbación de entrada	1	Valor nominal.

6.1.4. Evaluación de robustez

Además del valor de M_{ST}, se calculan márgenes de retardo y de fase con el fin de caracterizar la tolerancia del lazo cerrado frente a incertidumbres dinámicas. Estos indicadores complementan la interpretación de las curvas J vs M_{ST}, al permitir identificar si una mejora en el rendimiento se obtiene a costa de una menor reserva de estabilidad.

Tabla 6-4. Criterios para la evaluación de los sistemas de control: Evaluación de robustez

Parámetro	Valor adoptado	Comentario
Método de evaluación de M_{ST}	Exacto	No se usa aproximación simplificada para la métrica principal.
Orden de Padé	4	Empleado en modelos con retardo.
Malla frecuencial	10^{-3} a 10^{3} rad/s, 4000 puntos	Usada en la evaluación de M_{ST}.
Indicadores complementarios	DM, PM, GM	Se usan para interpretar reserva de estabilidad.

6.1.5. Validación temporal y sensibilidad al retardo

Para un valor fijado de M_{ST}, se evalúan las respuestas temporales de las sintonías obtenidas en el barrido ante cambios de referencia y perturbaciones de entrada y salida. Esta etapa permite interpretar las diferencias observadas en las curvas J vs M_{ST} a través de indicadores como tiempo de subida, tiempo de estabilización, sobrepaso, pico absoluto, offset e IAE. Asimismo, se analiza el efecto del error en el tiempo de retardo para cuantificar la sensibilidad de cada estrategia ante desajustes en la estimación del retardo.

Tabla 6-5. Criterios para la evaluación de los sistemas de control: Validación temporal y sensibilidad al retardo

Parámetro	Valor adoptado	Comentario
Valor fijo de robustez para validación	$M_{ST} = 1.6$	Seleccionado por representar el punto de balance entre rendimiento y robustez. El usuario puede cambiarlo.
Banda para tiempo de estabilización	2%	Se usa para calcular el tiempo de estabilización

Parámetro	Valor adoptado	Comentario
Errores relativos en el tiempo de retardo[4]	-30%, -20%, -10%, 0%, +10%, +20%, +30%	Evaluación de sensibilidad.
Indicadores temporales	Tiempo de subida, tiempo de estabilización, sobrepaso, pico absoluto, offset, IAE	Se reportan para referencia y perturbaciones.

6.1.6. Exportación y trazabilidad de resultados

Finalmente, el procedimiento genera tablas, figuras, reportes y archivos de resultados. Esta organización facilita la documentación de cada corrida, la reutilización de resultados previos y la regeneración del análisis posterior sin necesidad de repetir íntegramente la búsqueda de sintonías.

Tabla 6-6. Criterios para la evaluación de los sistemas de control: Exportación y trazabilidad

Parámetro	Valor adoptado	Comentario
Exportación de figuras	Sí	Activada.
Exportación a Excel	Sí	Activada.
Exportación CSV	No	Desactivada.
Guardado del workspace	Sí	Permite reutilizar corridas previas.
Recálculo automático al cargar workspace	No	Evita alterar curvas ya guardadas.

[4] En la sección de Resultados y análisis de resultados solo se presenta el escenario más crítico identificado para cada planta, la evaluación asociada a los demás porcentajes de error evaluados puede obtenerse mediante la ejecución del programa desarrollado, cuyo script se incluye en el anexo correspondiente.

6.2. Planta P_1: Tiempo de retardo puro ($\varphi = 0$)

La planta P_1, definida como $P_1(s) = e^{-s}$, representa el caso de tiempo de retardo puro. Este caso permite evaluar de forma directa el efecto de las estructuras predictivas frente a los controladores convencionales, sin la presencia de una dinámica estable adicional en $\tilde{P}_0(s)$. Por tanto, las diferencias entre PI, PID, SP y MSP se explican principalmente por la forma en que cada estructura trata el retardo y por la sintonía resultante de k_c, τ_i y τ_d. En esta sección se analizan conjuntamente el desempeño normalizado J, la evolución de los parámetros óptimos y los márgenes clásicos de robustez, con el fin de identificar el compromiso entre desempeño, agresividad de sintonía y sensibilidad frente a errores de retardo.

6.2.1. Pesos base resultantes

Como se muestra en la **Tabla 6-7**, para P_1 se obtuvieron $IAE_{do}^0 = 1.965$ e $IAE_{di}^0 = 2.171$, calculados con el controlador PI base alrededor de $M_{ST} = 1.59$. Estos valores permiten normalizar el índice de desempeño J y comparar todos los controladores respecto a una misma referencia de desempeño.

Como fue descrito previamente, el desempeño se evalúa mediante el índice normalizado:

$$J(C) = 0.5\left[\frac{IAE_{di}(C)}{IAE_{di}^0} + \frac{IAE_{do}(C)}{IAE_{do}^0}\right]$$

por lo que valores menores de J indican mejor desempeño relativo respecto a los pesos base.

Tabla 6-7. Pesos base para la normalización del índice J para P_1

Variable	Controlador	k_c	τ_i	τ_d	M_{ST}	IAE
IAE_{do}^0	PI	0.311	0.611	0	1.590	1.965
IAE_{di}^0	PI	0.311	0.611	0	1.590	2.171

6.2.2. Relación J, M_{ST} y parámetros de control óptimos

El análisis de la **Figura 6-1** debe realizarse junto con la **Figura 6-2**, ya que los cambios en J no dependen únicamente de la región de M_{ST}, sino también de cómo la búsqueda modifica k_c, τ_i y τ_d. En general, un aumento de k_c incrementa la acción proporcional y acelera la corrección del error, mientras que una reducción de τ_i fortalece la acción

integral y mejora el rechazo de perturbaciones sostenidas. Por ello, la disminución de J en el barrido debe interpretarse como resultado conjunto de una mayor agresividad permitida por M_{ST} y de la estructura propia de cada controlador.

En la región M_{ST} de 1 a 1.1, todos los controladores presentan valores elevados de J, lo que indica una sintonía muy restringida. En esta zona, la exigencia de robustez limita la agresividad del controlador, por lo que los valores de k_c son muy bajos. El resultado es una acción de control lenta, con pobre rechazo de perturbaciones y, por tanto, valores altos de J. Aun así, desde $M_{ST} \cong 1.1$ ya se aprecia que SP empieza a diferenciarse de PI, PID y MSP, mostrando que la estructura predictiva permite mejorar el desempeño incluso en una región todavía conservadora.

Entre valores de M_{ST} de 1.1 y 1.4, las curvas de J descienden rápidamente. Esta reducción coincide con el aumento progresivo de k_c, lo cual incrementa la acción proporcional y acelera el rechazo de perturbaciones. En PI y PID, las curvas prácticamente se superponen; esto se explica porque ambos controladores seleccionan valores casi idénticos de k_c y τ_i, y porque el término derivativo del PID solo toma un valor distinto de cero en M_{ST} de 1. A partir de $M_{ST} = 1.1$, $\tau_d = 0$, por lo que el PID se comporta esencialmente como un PI. En cambio, SP logra menores valores de J en todo este intervalo, asociado a una combinación de mayor capacidad de acción proporcional y menores valores de τ_i, es decir, una acción integral más rápida. MSP también mejora respecto de PI/PID, pero no alcanza a SP.

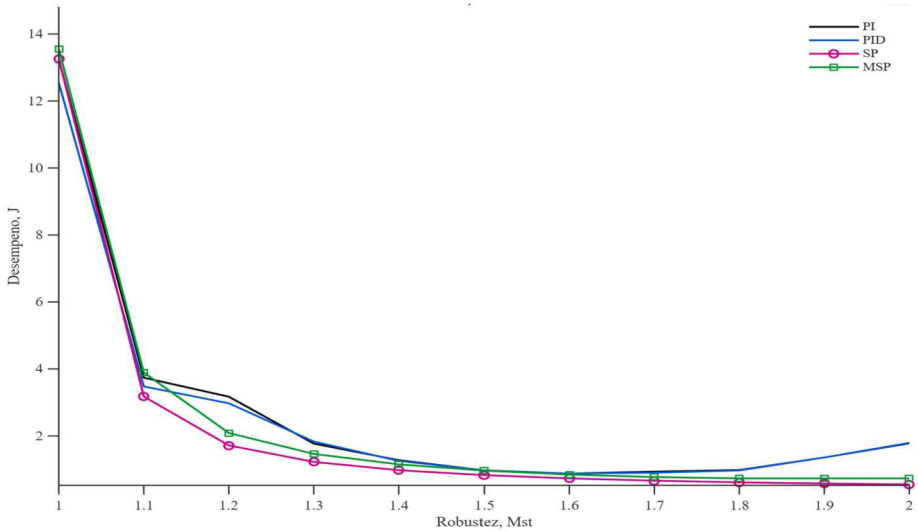

Figura 6-1. Curva de desempeño-robustez J vs. M_{ST} para P_1

En la región M_{ST} de 1.4 a 1.8, PI y PID se asientan alrededor de su zona de mejor desempeño, con valores de J cercanos a 0.88, prácticamente igual al MSP. En esta misma región, SP continúa reduciendo J, mientras que MSP se estabiliza en valores intermedios. La **Figura 6-2** muestra que MSP utiliza valores de k_c mayores que PI y PID, especialmente entre M_{ST} de 1.3 y 1.8, pero también valores de τ_i mayores que SP. Esto indica una estrategia distinta: MSP incrementa la acción proporcional, pero mantiene una acción integral menos rápida que SP. En términos prácticos, esta combinación mejora el desempeño respecto de los controladores clásicos, pero no explota el rechazo de perturbaciones con la misma eficiencia que SP.

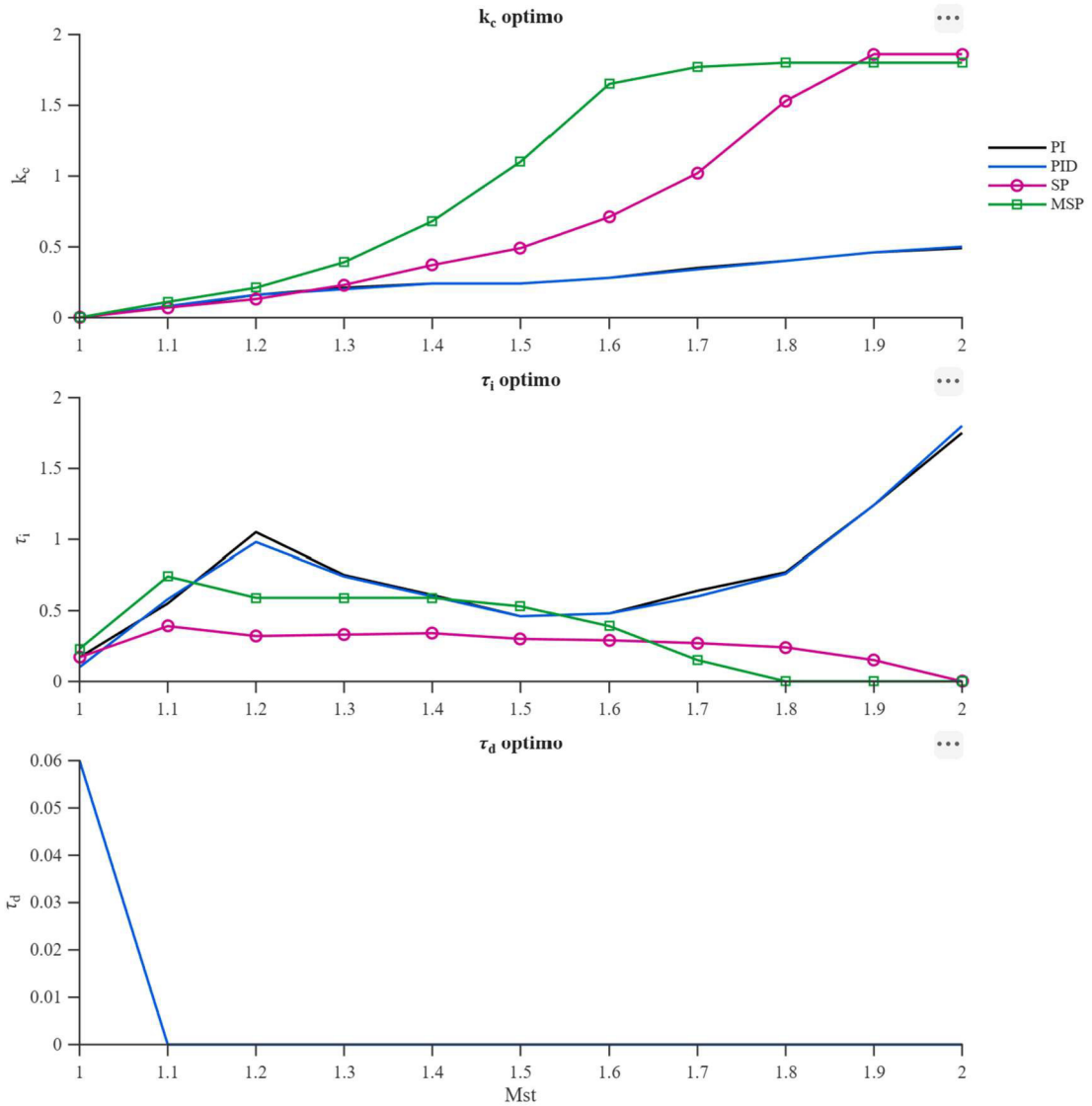

Figura 6-2. Parámetros de sintonía óptimos en función de M_{ST} para P_1

En la región M_{ST} de 1.8 a 2.0, SP alcanza el menor valor de J, como se resume en la **Tabla 6-8**. En este tramo, k_c de SP es alto y τ_i tiende a valores muy bajos, lo que implica

una sintonía más agresiva y rápida. Esta agresividad se traduce en una fuerte reducción de IAE_{do}, manteniendo además un IAE_{di} competitivo. MSP también mejora frente a PI/PID, pero no supera a SP. Por tanto, para P1, MSP no logra superar de forma confiable al predictor de Smith convencional.

La **Tabla 6-8** resume los mejores puntos de cada controlador evaluado, es decir, cuando logran el menor valor de J a un M_{ST} determinado. El detalle de todos los valores resultantes utilizados para la construcción de la gráfica J vs M_{ST} se adjunta en la sección Anexos. Es resaltante que, al no usar activamente el término derivativo del PID, este se hace un PI prácticamente en todo el horizonte M_{ST} de evaluación, mientras que el SP saca ventaja al MSP por la mejor reducción del error en la respuesta controlada de la perturbación de salida.

Tabla 6-8. Puntos de mínimo J por controlador para P_1

Planta	Tipo	Controlador	J	M_{ST}	IAE_{do}	IAE_{di}
P_1	Estable	PI	0.877	1.603	1.712	1.915
P_1	Estable	PID	0.877	1.603	1.713	1.916
P_1	Estable	SP	0.295	2.000	0.002	1.280
P_1	Estable	MSP	0.732	1.728	1.420	1.611

Para mostrar el comportamiento de estos controladores, se simuló la respuesta controlada para un valor de $M_{ST} = 1.6$, considerando un modelado ideal de la planta. Como se observa en la **Figura 6-3,** ante cambios en la referencia, el MSP presenta la respuesta más rápida entre las estrategias evaluadas, alcanzando el menor tiempo de estabilización y el menor valor de IAE, según se resume en la **Tabla 6-9**.

Ante perturbaciones de entrada y salida, el MSP muestra una respuesta más lenta que el SP, aunque menos oscilatoria. Además, alcanza el estado estable en menor tiempo que los controladores convencionales, lo que evidencia una mejora en la regulación del sistema. Este comportamiento se relaciona con los menores picos absolutos obtenidos por el MSP, los cuales contribuyen a reducir el tiempo de estabilización de la respuesta controlada.

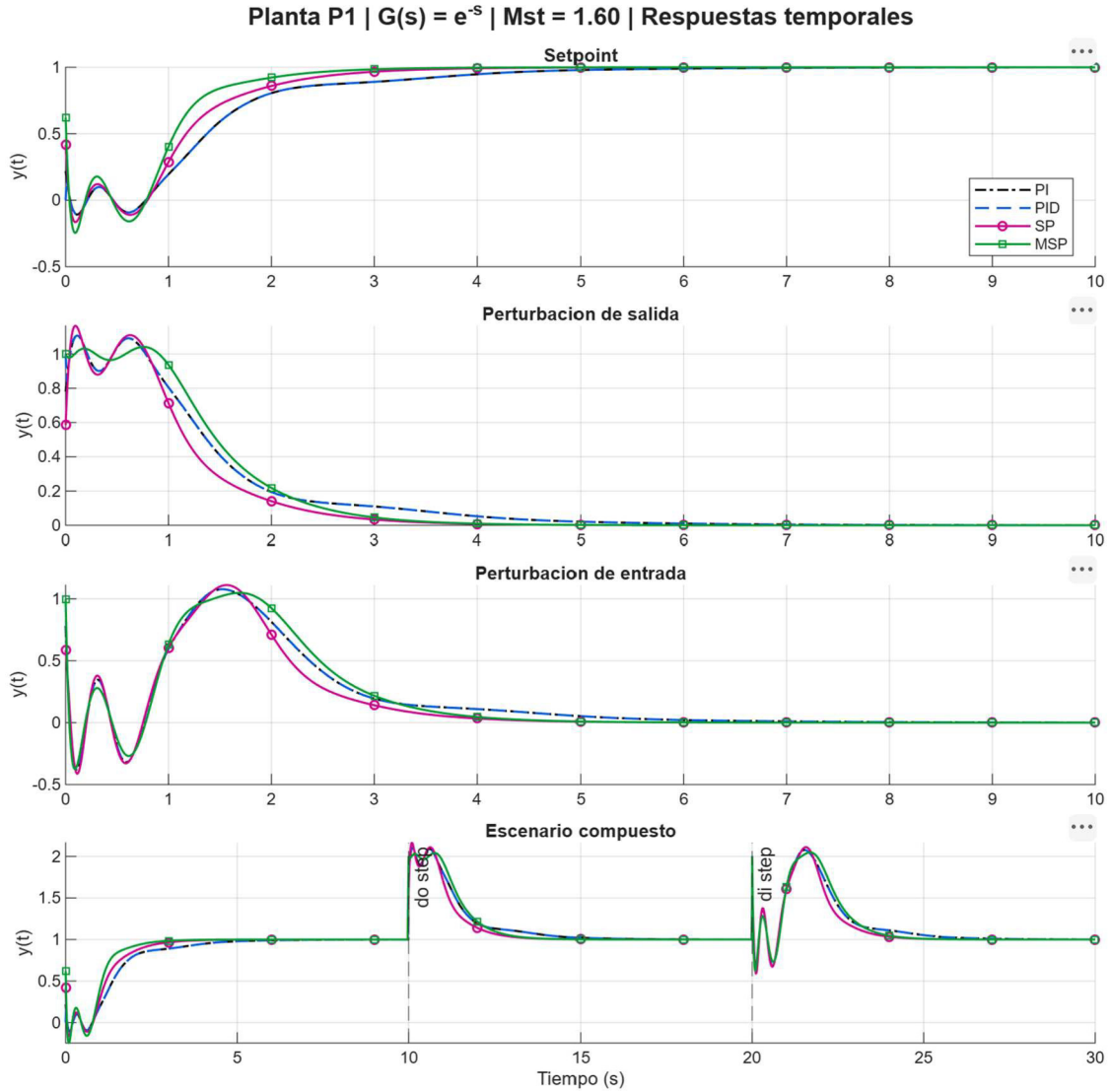

Figura 6-3. Respuestas controladas ante cambio de referencia y perturbaciones de entrada y salida para P_1 ($M_{ST} = 1.6$)

Tabla 6-9. Indicadores de rendimiento de las respuestas controladas para P_1 ($M_{ST} = 1.6$)

Controlador	Tiempo de subida	Pico absoluto	Tiempo de estabilización	Sobrepaso	Offset	IAE
Referencia						
PI	3.19	-	5.08	0.000E+00	4.597E-11	1.71
PID	3.19	-	5.07	0.000E+00	4.495E-11	1.71
SP	2.24	-	3.37	6.661E-14	1.985E-13	1.41
MSP	1.84	-	2.85	1.332E-13	2.886E-15	1.24
Perturbación de entrada						
PI	-	1.08	5.98	-	1.023E-10	1.92
PID	-	1.08	5.97	-	1.001E-10	1.92
SP	-	1.11	4.29	-	4.108E-15	1.62
MSP	-	1.05	4.50	-	2.332E-15	1.84
Perturbación de salida						

Controlador	Tiempo de subida	Pico absoluto	Tiempo de estabilización	Sobrepaso	Offset	IAE
PI	-	1.11	4.94	-	4.598E-11	1.71
PID	-	1.11	4.94	-	4.494E-11	1.71
SP	-	1.16	3.26	-	3.331E-16	1.41
MSP	-	1.04	3.51	-	1.799E-14	1.66

6.2.3. Interpretación estructural

La razón estructural de este comportamiento puede observarse comparando las funciones de transferencia. Para el lazo clásico con $P_1(s) = e^{-s}$, la transferencia de referencia puede escribirse como:

$$H_r^{PI}(s) = \frac{\mathrm{C(s)}e^{-s}}{1 + \mathrm{C(s)}e^{-s}} \tag{6-1}$$

En este caso, el retardo aparece en el denominador característico, por lo que limita directamente la agresividad admisible de la sintonía. En cambio, para el predictor de Smith ideal:

$$H_r^{SP}(s) = \frac{\mathrm{C(s)}e^{-s}}{1 + \mathrm{C(s)}} \tag{6-2}$$

El retardo permanece en el numerador, pero no en el denominador que define los polos nominales del lazo cerrado. Esto permite seleccionar sintonías más rápidas sin comprometer la estabilidad nominal de la misma forma que en el lazo clásico.

Para MSP, también bajo el modelo perfecto, el denominador nominal se reduce a:

$$1 + \mathrm{C(s)}\tilde{P}_0(s)$$

Sin embargo, en P_1 se tiene $\tilde{P}_0(s) = 1$, por lo que el filtro $F_p(s)$ no cancela polos ni compensa una dinámica propia de la planta. La diferencia aparece en la transferencia de perturbación. Para SP:

$$H_{do}^{SP}(s) = \frac{1}{1 + \mathrm{C(s)}} \tag{6-3}$$

mientras que para MSP:

$$H_{do}^{MSP}(s) = \frac{1 + \mathrm{C(s)}(1 - F_p(s)e^{-s})}{1 + \mathrm{C(s)}} \tag{6-4}$$

Esto evidencia que, bajo condiciones de modelamiento perfecto de la planta, el MSP conserva una dinámica adicional asociada al filtro $F_p(s)$ en el numerador de la función de transferencia de perturbación, sin que exista una dinámica propia de la planta que justifique una cancelación beneficiosa. Por ello, el filtro predictivo no aporta una ventaja estructural clara frente al SP en este caso; más bien, modifica la respuesta ante perturbaciones y obliga a compensar mediante mayores valores de k_c y una acción integral menos agresiva. En consecuencia, para P1 y bajo modelado ideal, el MSP supera a los controladores PI/PID, pero queda por debajo del SP en términos de desempeño nominal.

6.2.4. Márgenes de estabilidad y robustez

Los márgenes clásicos complementan el análisis de J, pero no lo reemplazan. Mientras J evalúa el desempeño integral frente a perturbaciones, PM y DM permiten observar la holgura del lazo frente a incertidumbres dinámicas. Un menor margen de fase PM significa menor separación angular respecto de la condición crítica de -180º. Esta condición representa el punto donde la realimentación negativa puede comportarse como realimentación positiva efectiva; si además la magnitud del lazo es igual o mayor que la unidad, el sistema puede perder estabilidad. Por ello, un PM bajo implica menor tolerancia a dinámicas no modeladas que agreguen atraso de fase, como polos rápidos omitidos, sensores, actuadores, filtros, discretización, aproximaciones de Padé o errores en el modelo dinámico.

El margen de retardo DM, por otro lado, mide cuánta demora adicional puede tolerar el lazo antes de consumir completamente su margen de fase. Si se agrega un retardo $\Delta\theta$, el lazo queda:

$$L_{\Delta\theta}(s) = L(s)e^{-\Delta\theta s} \tag{6-5}$$

y en frecuencia:

$$\angle L_{\Delta\theta}(j\omega) = \angle L(j\omega) - \omega\Delta\theta \tag{6-6}$$

Por tanto, el retardo adicional consume fase negativa. En la frecuencia de cruce ω_c, donde $|L(j\omega_c)| = 1$, se tiene:

$$DM = \frac{PM_{rad}}{\omega_c} \tag{6-7}$$

Esto explica por qué un controlador puede tener mayor PM y, al mismo tiempo, menor DM. El PM es una reserva angular, mientras que el DM traduce esa reserva a tiempo. Si la sintonía aumenta ω_c, el mismo retardo adicional consume más fase.

Por su lado, M_{ST} mide la amplificación máxima de sensibilidad y sensibilidad complementaria. Un M_{ST} mayor indica mayor pico de sensibilidad y, generalmente, una sintonía más agresiva; un M_{ST} menor implica una sintonía más restringida o conservadora. Su relación con PM y DM es indirecta. En particular, DM depende explícitamente de ω_c, variable que no está contenida directamente en M_{ST}.

En la **Figura 6-4**, todos los controladores parten con márgenes de retardo muy altos en $M_{ST} = 1$, pero esa zona también corresponde a los peores valores de J en la **Figura 6-1**. Es decir, una robustez excesivamente conservadora protege frente a variaciones de retardo, pero penaliza fuertemente el desempeño.

Entre M_{ST} de 1 y 1.4, el SP ya reduce J con claridad, pero lo hace a costa de una caída más rápida del margen de retardo. Esta tendencia es coherente con su sintonía: menor τ_i y mayor rapidez integral, lo que mejora el rechazo de perturbaciones, pero disminuye la holgura frente a errores adicionales de retardo. En cambio, PI y PID conservan mayores valores de DM, aunque con un J más alto. Esto muestra que los controladores convencionales son menos agresivos, pero también menos eficientes en desempeño. Por su parte, el MSP se posiciona ligeramente mejor que el SP, sin embargo, no es suficiente para alcanzar los valores de DM de los controladores convencionales.

En la región M_{ST} de 1.4 y 1.8, el contraste se vuelve más evidente. PI y PID se estabilizan alrededor de su mejor zona de J, mientras mantienen márgenes de retardo moderados. El SP continúa reduciendo J, pero su DM cae hasta valores muy bajos. Por tanto, aunque SP es el mejor en desempeño, su mejora se obtiene aceptando una menor tolerancia frente a incertidumbre de retardo. El MSP muestra un comportamiento intermedio: mejora el desempeño respecto a PI/PID, pero no alcanza al SP, y tampoco conserva un margen de retardo claramente superior.

La **Figura 6-5** complementa esta interpretación. PI y PID presentan márgenes de fase relativamente moderados, con una disminución hasta la zona media de M_{ST} y posterior recuperación hacia $M_{ST} = 2$. En cambio, SP y MSP alcanzan márgenes de fase mucho mayores. Esto indica que los controladores predictivos no pierden robustez de fase en la misma forma en que pierden margen de retardo. Dicho de otro modo, el límite principal para SP y MSP en P_1 no parece ser la fase, sino la sensibilidad frente a desajustes de retardo.

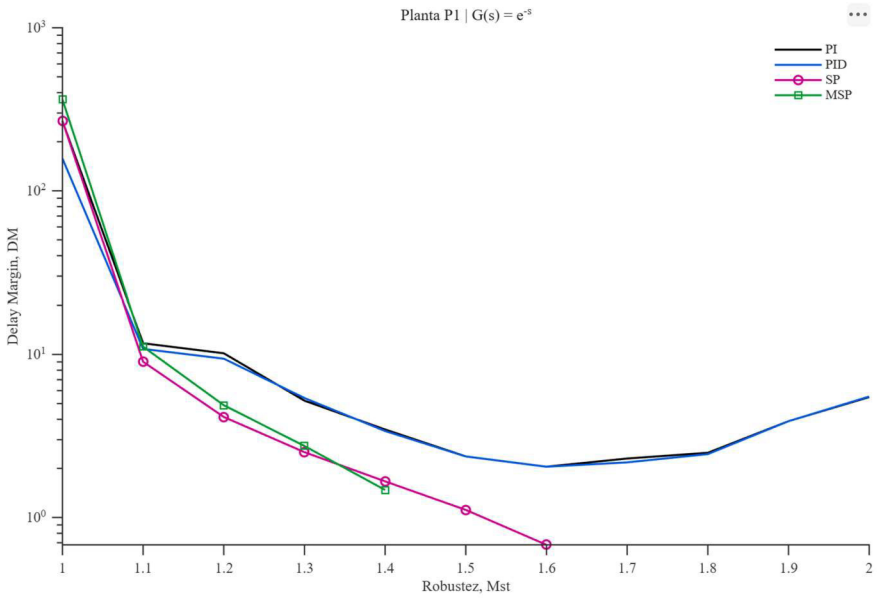

Figura 6-4. Margen de retardo en función de M_{ST} para P_1

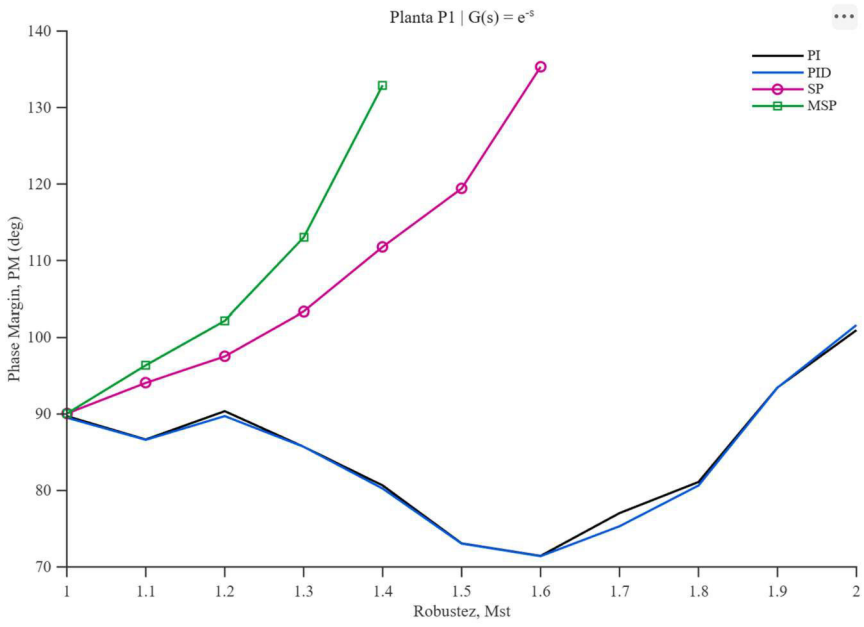

Figura 6-5. Margen de fase en función de M_{ST} para P_1

También debe señalarse que algunas curvas de margen no aparecen en todo el intervalo. Esto no implica necesariamente ausencia de una solución estable en términos de J sino que, para ciertas sintonías, el cálculo del margen no entrega un valor finito, positivo o físicamente interpretable. Este comportamiento aparece especialmente en las zonas donde los controladores predictivos adoptan sintonías más agresivas.

6.2.5. Simulación del efecto de errores de modelado del tiempo de retardo en la respuesta controlada

Para evaluar el comportamiento de los controladores frente a errores en el modelado del tiempo de retardo, se simuló la respuesta controlada para un valor de $M_{ST} = 1.6$. La **Figura 6-6** muestra las respuestas obtenidas ante un error de modelado del tiempo de retardo de -30%, correspondiente al escenario más desfavorable dentro de los casos evaluados.

Esta simulación permite observar el compromiso entre rendimiento y robustez de cada controlador. En el análisis previo de la curva J vs. M_{ST}, el SP presentó un menor valor de J que el MSP, lo que indica un mejor desempeño nominal asociado a una sintonización más agresiva. No obstante, al incorporar el efecto del error en el tiempo de retardo, se observa que el SP resulta más sensible al desajuste del modelo, presentando una respuesta más oscilatoria. Este comportamiento es coherente con su menor margen de retardo. En cambio, el MSP mantiene una respuesta relativamente estable, similar a la del PID, y presenta una menor sensibilidad al desajuste del retardo que el SP.

En este sentido, para una condición de compromiso como $M_{ST} = 1.6$, el MSP presenta ventajas de robustez frente al SP, ya que conserva una respuesta más estable ante errores en el tiempo de retardo. Sin embargo, también debe señalarse que el PID muestra una robustez considerable y alcanza desempeños cercanos a los del MSP, especialmente frente a perturbaciones, correspondiéndose con la tendencia descrita para el DM vs M_{ST} en la **Figura 6-4**. La principal ventaja del MSP se observa en el seguimiento de cambios de referencia, donde presenta una respuesta más rápida y con mejor comportamiento dinámico.

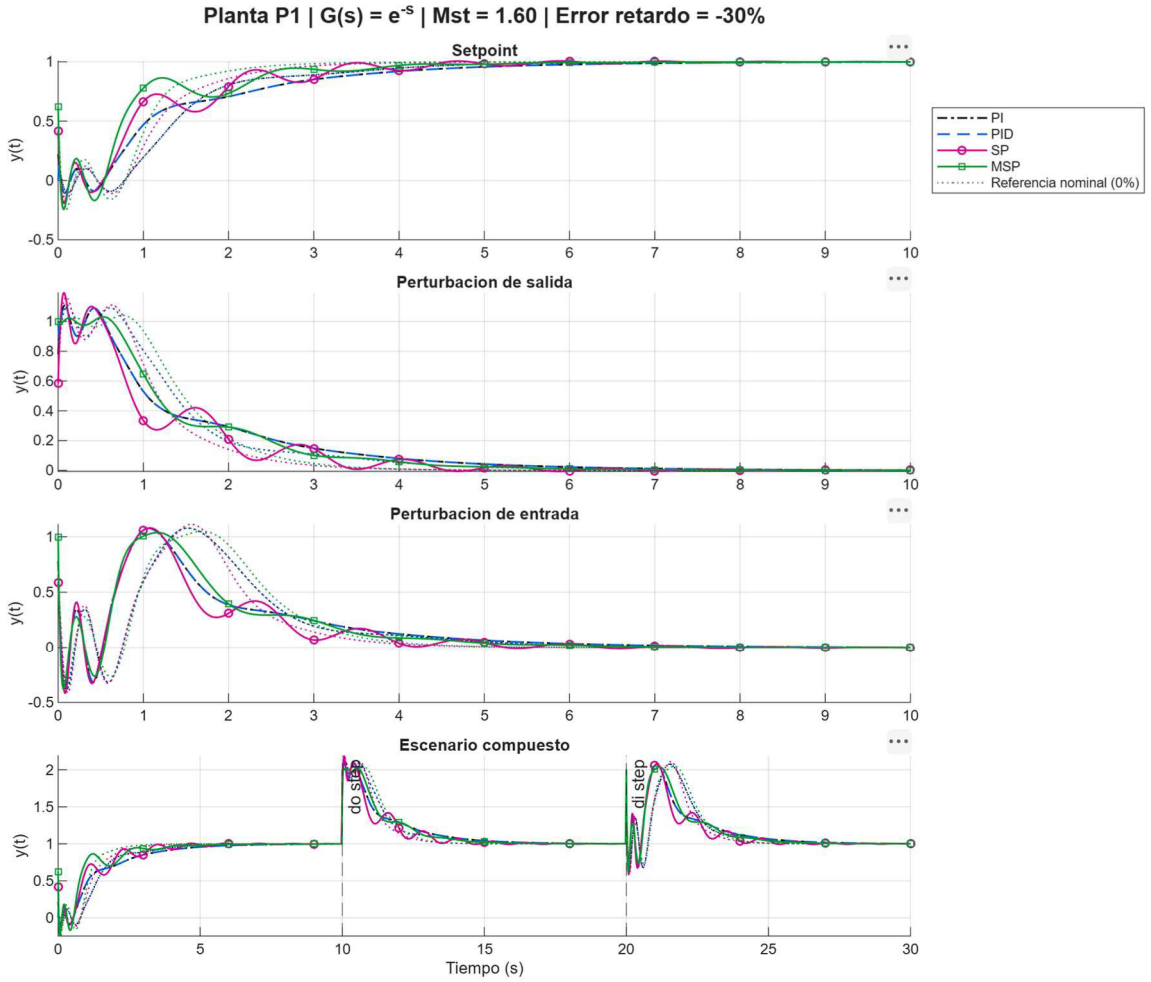

Figura 6-6. Respuestas controladas ante un error de modelado del tiempo de retardo de -30% para P_1 ($M_{ST} = 1.6$)

6.2.6. Síntesis de resultados para P_1

Bajo condiciones de modelado ideal, el SP presenta el mejor desempeño nominal, alcanzando los menores valores de J, especialmente para valores altos de M_{ST}. Esto se debe a que su estructura retira el retardo de la ecuación característica nominal, permitiendo sintonías más agresivas y una reducción importante del error acumulado. En cambio, los controladores PI y PID presentan respuestas prácticamente equivalentes, debido a que la acción derivativa del PID no participa de manera significativa en la sintonización óptima.

El MSP mejora el desempeño respecto a los controladores convencionales, pero no supera al SP en términos de J bajo modelado ideal. Esta diferencia se explica porque, en P_1, la parte libre de retardo de la planta es $\tilde{P}_0(s) = 1$, por lo que el filtro predictivo del MSP no

tiene polos de planta que compensar. En consecuencia, bajo un modelo ideal, $F_p(s)$ introduce una dinámica adicional que no aporta una ventaja estructural clara para este caso. Aun así, para $M_{ST} = 1.6$, el MSP muestra una respuesta más rápida ante cambios de referencia y una respuesta más amortiguada frente a perturbaciones, aunque con mayor error acumulado que el SP.

El análisis de robustez modifica la interpretación basada únicamente en J. La ventaja nominal del SP se obtiene con una sintonía más agresiva y con menor margen de retardo, lo que incrementa su sensibilidad frente a errores en el modelado del tiempo de retardo. Esto se evidencia en la simulación en un escenario no ideal: con error de retardo del -30%, donde el SP presenta una respuesta más oscilatoria. En contraste, el MSP conserva una respuesta más estable ante el desajuste del retardo, mostrando un comportamiento más robusto que el SP.

Por tanto, para P1, la elección del controlador depende del criterio priorizado. El SP resulta más conveniente cuando se busca minimizar Jbajo condiciones de modelado confiable, especialmente en la región de mayor M_{ST}. Para una condición intermedia, alrededor de $M_{ST} = 1.6$, el MSP ofrece el seguimiento de referencia más rápido y presenta una menor sensibilidad al error en el tiempo de retardo que el SP, aunque con un mayor error acumulado frente a perturbaciones. Por su parte, PI y PID presentan un comportamiento prácticamente equivalente y conservan mayores márgenes de retardo, por lo que resultan más favorables cuando se prioriza la tolerancia a incertidumbre en el retardo sobre el desempeño nominal.

Tabla 6-10. Resumen de resultados P_1

Bloque	Criterio	P1 – Retardo puro
Rendimiento	J nominal	SP > MSP > PI ≈ PID
	Seguimiento de referencia ($M_{ST} \approx 1.6$)	MSP > SP > PI ≈ PID
	Perturbación de salida ($M_{ST} \approx 1.6$)	SP > MSP > PI ≈ PID
	Perturbación de entrada ($M_{ST} \approx 1.6$)	SP > MSP > PI ≈ PID
Subtotal rendimiento		**SP: mejor regulación. MSP: mejor seguimiento.**
Robustez	Margen de retardo, DM	PI ≈ PID > MSP > SP
	Margen de fase, PM	SP ≈ MSP > PI ≈ PID
	Error de retardo −30% ($M_{ST} \approx 1.6$)	SP se degrada más; MSP ≈ PID más estables.
Subtotal robustez		**PI/PID: mayor DM. MSP: mayor tolerancia que SP.**

Bloque	Criterio	P1 – Retardo puro
	Lectura general	-SP: si se prioriza J y el retardo está bien modelado, especialmente con $M_{ST} \gtrsim 1.6$. -MSP: alrededor de $M_{ST} \approx 1.6$ si se prioriza seguimiento rápido y mayor tolerancia al error de retardo que SP. -PID/PI: si se prioriza margen de retardo; en P1 son prácticamente equivalentes.

6.3. Planta P_2: Dinámica balanceada

La planta P_2, definida como $P_2(s) = e^{-s}/(s+1)$, incorpora una dinámica estable de primer orden además del tiempo de retardo. A diferencia de P_1, donde $\tilde{P}_0(s) = 1$, en P_2 el modelo sin retardo contiene el polo estable $\tilde{P}_0(s) = 1/(s+1)$. Por ello, esta planta permite evaluar no solo la compensación del retardo, sino también cómo los controladores enfrentan la dinámica propia del proceso. En consecuencia, el análisis se realiza relacionando el desempeño J, la evolución de los parámetros óptimos y los márgenes de robustez.

6.3.1. Pesos base resultantes

Como se muestra en la **Tabla 6-11**, los pesos base para P_2 fueron $IAE_{do}^0 = 2.882$ e $IAE_{di}^0 = 2.900$, obtenidos con un controlador PI alrededor de $M_{ST} = 1.595$. La cercanía entre ambos pesos indica que la referencia de desempeño es equilibrada frente a perturbaciones de salida y entrada.

Tabla 6-11. Pesos base para la normalización del índice J para P_2

Variable	Controlador	k_c	τ_i	τ_d	M_{ST}	IAE
IAE_{do}^0	PI	0.701	2.021	0	1.595	2.882
IAE_{di}^0	PI	0.701	2.021	0	1.595	2.900

6.3.2. Relación J, M_{ST} y parámetros de control óptimos

En la **Figura 6-7**, se muestra la variación de J en función de los M_{ST} evaluados para la planta P_2, mientras la **Figura 6-8** que en la se presenta la variación de los parámetros de control óptimos resultantes de la búsqueda de sintonías.

En la región inicial, entre M_{ST} igual a 1 y 1.2, las curvas se ordenan de forma más estable. PI queda sistemáticamente por encima, mientras que PID, SP y MSP permanecen muy cercanos. Esta región muestra que la planta P_2 puede ser controlada de manera competitiva con PID, sin que el uso de SP o MSP genere todavía una ventaja marcada en J. De hecho, hacia $M_{ST} = 1.7$, los valores son prácticamente equivalentes: PID alcanza $J = 0.511$, SP $J = 0.509$ y MSP $J = 0.510$.

Entre M_{ST} igual a 1.2 y 1.7, SP y MSP continúan reduciendo ligeramente J, mientras que PI pierde competitividad y PID se mantiene cercano. El menor valor global lo obtiene

MSP con $J = 0.417$, seguido prácticamente por SP con $J = 0.419$ y luego PID con $J = 0.461$. Por tanto, aunque MSP presenta el mejor resultado numérico, la diferencia frente a SP es mínima y frente a PID no es suficientemente amplia como para evidenciar una ventaja dominante. Para P_2, la mejora de las estructuras predictivas existe, pero es moderada.

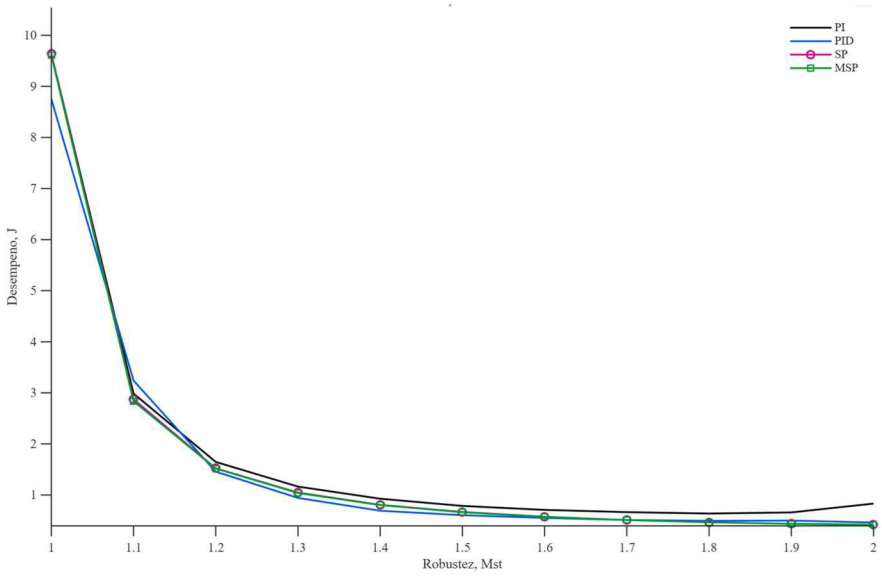

Figura 6-7. Curva de desempeño-robustez J vs. M_{ST} para P_2

En la región entre M_{ST} igual a 1.7 y 2, el orden prácticamente se mantiene. En la región alta de M_{ST}, SP y MSP presentan desempeños prácticamente equivalentes, alcanzando el MSP el menor valor global de J, aunque con una diferencia mínima respecto al SP, como se resume en la **Tabla 6-12.**

MSP obtiene $J = 0.417$, superando a PI y quedando competitivo frente a PID, cuyo mejor valor es $J = 0.461$. Esto indica que, para P_2, el filtro MSP sí aporta una mejora más clara que en P1; sin embargo, esta mejora no es suficiente para superar al predictor de Smith convencional.

Tabla 6-12. Puntos de mínimo J por controlador para P_2

Planta	Tipo	Controlador	J	M_{ST}	IAE_{do}	IAE_{di}
P_2	Estable	PI	0.637	1.804	1.955	1.727
P_2	Estable	PID	0.461	2.003	1.467	1.198

Planta	Tipo	Controlador	J	M_{ST}	IAE_{do}	IAE_{di}
P_2	Estable	SP	0.419	2.001	1.267	1.155
P_2	Estable	MSP	0.417	2.004	1.316	1.092

La **Figura 6-8** muestra que los controladores SP y MSP son menos conservadores en la sintonía, y requieren mayores valores de k_c para minimizar J, en regiones mayores de M_{ST}. Respecto a τ_i, la exigencia es similar para PID, SP y MSP, PI que toma valores relativamente mayores. Respecto a τ_d, este adquiere sus mayores valores en regiones conservadoras, luego permanece muy pequeño y cercano a 0. Esencialmente, para este tipo de procesos, los controladores diseñados con el Predictor de Smith no presentan gran ventaja respecto a los controladores tradicionales, principalmente porque estos últimos no requieren de una sintonía tan exigente para lograr desempeños similares a los primeros.

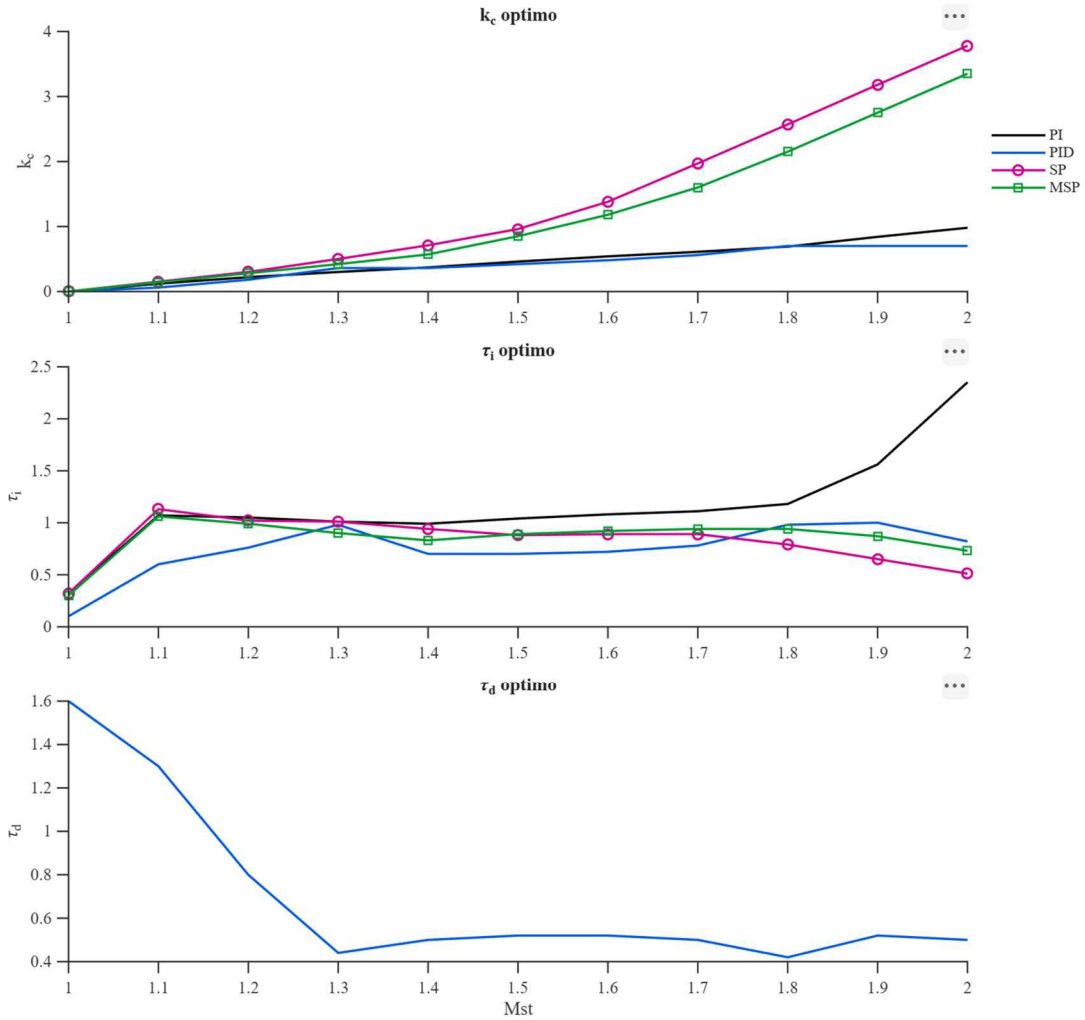

Figura 6-8. Parámetros de sintonía óptimos en función de M_{ST} para P_2

Para mostrar el comportamiento de estos controladores, se simuló la respuesta controlada para un valor de $M_{ST} = 1.6$, considerando un modelado ideal de la planta. Como se observa en la **Figura 6-9,** ante cambios en la referencia, el PID presenta una respuesta ligeramente más rápida que las otras estrategias, suficiente para alcanzar el menor IAE, según se resume en la **Tabla 6-13**.

Ante perturbaciones de entrada y salida, los resultados son consistentes con lo observado en la curva J vs M_{ST}, ya que el PID presenta un rendimiento ligeramente superior al SP y al MSP. Estos dos últimos muestran respuestas muy cercanas, prácticamente superpuestas en la representación gráfica. No obstante, la respuesta del PID presenta una oscilación inicial ligeramente mayor, aunque mantiene menores valores de error acumulado en comparación con las estrategias predictivas.

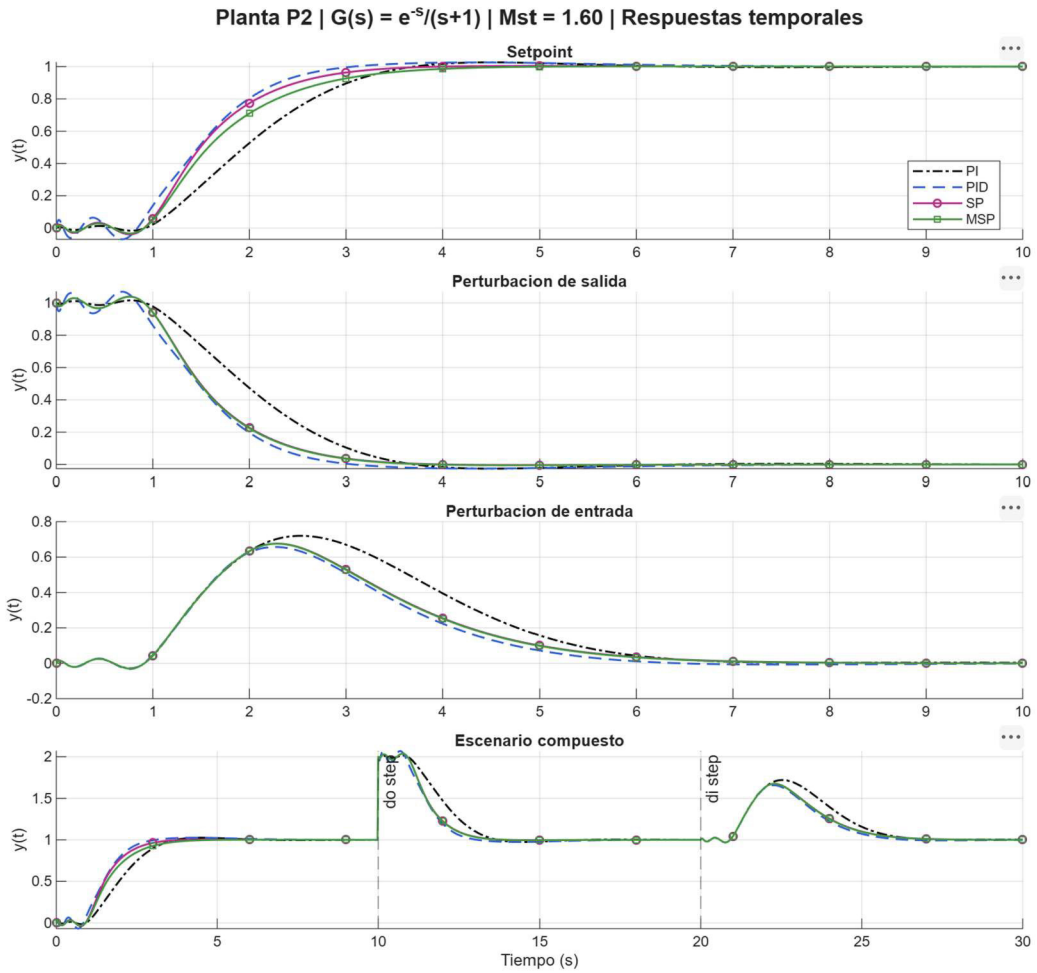

Figura 6-9. Respuestas controladas ante cambio de referencia y perturbaciones de entrada y salida para P₂ ($M_{ST} = 1.6$)

Tabla 6-13. Indicadores de rendimiento de las respuestas controladas para P_2 ($M_{ST} = 1.6$)

Controlador	Tiempo de subida	Pico absoluto	Tiempo de estabilización	Sobrepaso	Offset	IAE
Referencia						
PI	3.03	-	5.10	2.611E+00	5.691E-11	2.08
PID	2.31	-	5.21	2.568E+00	-2.044E-11	1.63
SP	2.49	-	3.27	3.561E-01	6.328E-15	1.66
MSP	2.79	-	3.82	1.084E-01	-7.927E-14	1.78
Perturbación de entrada						
PI	-	0.72	6.68	-	3.341E-10	2.02
PID	-	0.66	5.94	-	7.921E-11	1.55
SP	-	0.68	6.84	-	4.306E-13	1.66
MSP	-	0.68	6.77	-	5.014E-14	1.65
Perturbación de salida						
PI	-	1.02	5.08	-	5.692E-11	2.08
PID	-	1.07	5.08	-	2.044E-11	1.63
SP	-	1.04	3.25	-	1.021E-14	1.66
MSP	-	1.04	3.22	-	2.154E-14	1.66

6.3.3. Interpretación estructural

Para los controladores convencionales, el retardo permanece dentro del denominador de lazo cerrado. En forma general,

$$H_{do}(s) = \frac{1}{1 + C(s)\frac{e^{-s}}{s+1}} \tag{6-8}$$

$$H_{di}(s) = \frac{\frac{e^{-s}}{s+1}}{1 + C(s)\frac{e^{-s}}{s+1}} \tag{6-9}$$

Esto significa que el controlador clásico debe estabilizar directamente una dinámica que incluye tanto el polo estable $1/s + 1$ como el retardo e^{-s}. En principio, esto restringe la agresividad de la sintonía. Sin embargo, en *P_2* la dinámica sin retardo es de primer orden estable y relativamente simple, por lo que PI y, especialmente, PID pueden obtener desempeños competitivos sin requerir ganancias tan elevadas. Esto se observa en la **Figura 6-8,** donde PI y PID trabajan con valores de k_c menores que SP y MSP.

En el caso del PID, la acción derivativa sí tiene un papel funcional en *P_2*. A diferencia de *P_1*, donde τ_d prácticamente desaparecía después de la región más conservadora, aquí τ_d se mantiene distinto de cero durante el barrido. Esto indica que el PID utiliza la acción

derivativa para anticipar parcialmente la dinámica de primer orden del proceso. Por esa razón, su desempeño se acerca al de MSP y no queda tan rezagado como en P_1.

Para el Predictor de Smith, bajo modelo perfecto, el retardo nominal deja de aparecer en el denominador principal de la dinámica controlada, para el caso de la perturbación de salida:

$$H_{do}^{SP}(s) = \frac{1}{1 + C(s)\frac{1}{s+1}} \tag{6-10}$$

Esta estructura permite que SP use una sintonía más agresiva, con mayores valores de k_c y menores valores de τ_i. Esto explica que SP alcance el menor J, especialmente en la región media y alta de M_{ST}. Sin embargo, esta mejora no es gratuita: al aumentar la rapidez del lazo también aumenta la sensibilidad frente a errores en el retardo. Por eso, SP presenta menores valores de DM que PI y PID, como se describe en la sección posterior.

Para MSP, bajo modelo perfecto, las perturbaciones quedan modificadas por el filtro $F_p(s)$. Para la perturbación de salida,

$$H_{do}^{MSP}(s) = \frac{\frac{1}{s+1}(1 + F_p(s)e^{-s})}{1 + C(s)\frac{1}{s+1}} \tag{6-11}$$

El filtro $F_p(s)$ introduce una corrección adicional en el numerador de las funciones de perturbación. En P_2, como existe una dinámica estable $1/s + 1$, el filtro tiene mayor sentido que en P_1; sin embargo, no elimina una dinámica problemática ni un polo indeseado. Por ello, su efecto práctico es moderado: mejora respecto al PI clásico, pero no genera una reducción de J claramente superior a la del PID ni alcanza el desempeño del SP.

Como se describe en la sección posterior, SP logra el menor J, pero lo hace con una sintonía más exigente y menor DM. MSP queda en una posición intermedia: usa mayores valores de k_c que PI/PID, pero menores que SP, y mantiene una acción integral menos agresiva. PID, por su parte, conserva un desempeño muy competitivo gracias al uso efectivo de τ_d, con una sintonía menos exigente y mayor margen de retardo.

6.3.4. Márgenes de estabilidad y robustez

Los márgenes DM y PM se interpretan como indicadores complementarios: DM mide tolerancia frente a retardo adicional, mientras que PM representa holgura frente a atraso de fase no modelado. En la **Figura 6-10**, se observa que el margen de retardo disminuye conforme aumenta M_{ST}, lo cual es coherente con sintonías más agresivas. SP y MSP presentan menores valores de DM en la zona media y alta, confirmando que su mejora de desempeño se obtiene aceptando menor tolerancia frente a errores de tiempo de retardo.

En regiones previas al $M_{ST} = 1.6$, el DM de los controladores es similar y presentan una tendencia estable. Para regiones superiores, la diferencia es más holgada, PI y PID conservan mayores valores de DM, mientras que SP y MSP aceptan menor tolerancia frente a errores de tiempo de retardo. Esto confirma que parte de la mejora de desempeño de los predictores se obtiene a costa de una menor reserva frente a desajustes en el retardo.

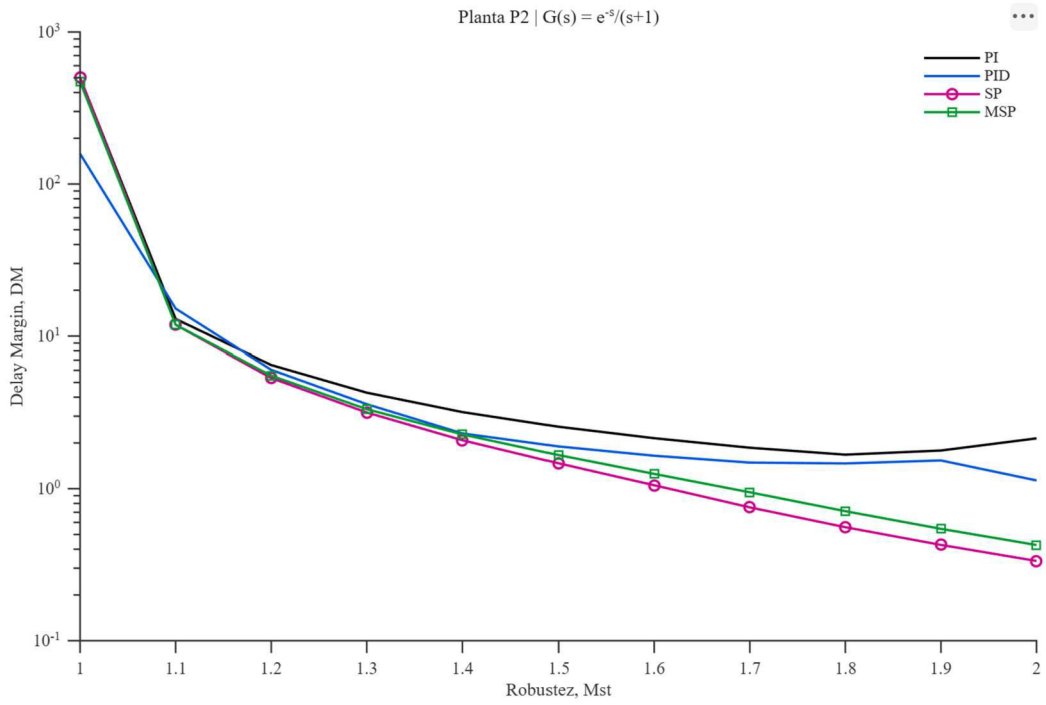

Figura 6-10. Margen de retardo en función de M_{ST} para P_2

La **Figura 6-11** muestra que SP y MSP mantienen márgenes de fase más altos que PI y PID en gran parte del barrido. Esto indica que la limitación principal de los predictores no está en una pérdida general de margen de fase, sino en su sensibilidad específica al retardo. Así, SP puede tener mejor J y buen PM, pero menor DM, porque la sintonía más

rápida aumenta la frecuencia de cruce y hace que un error de retardo consuma fase con mayor rapidez.

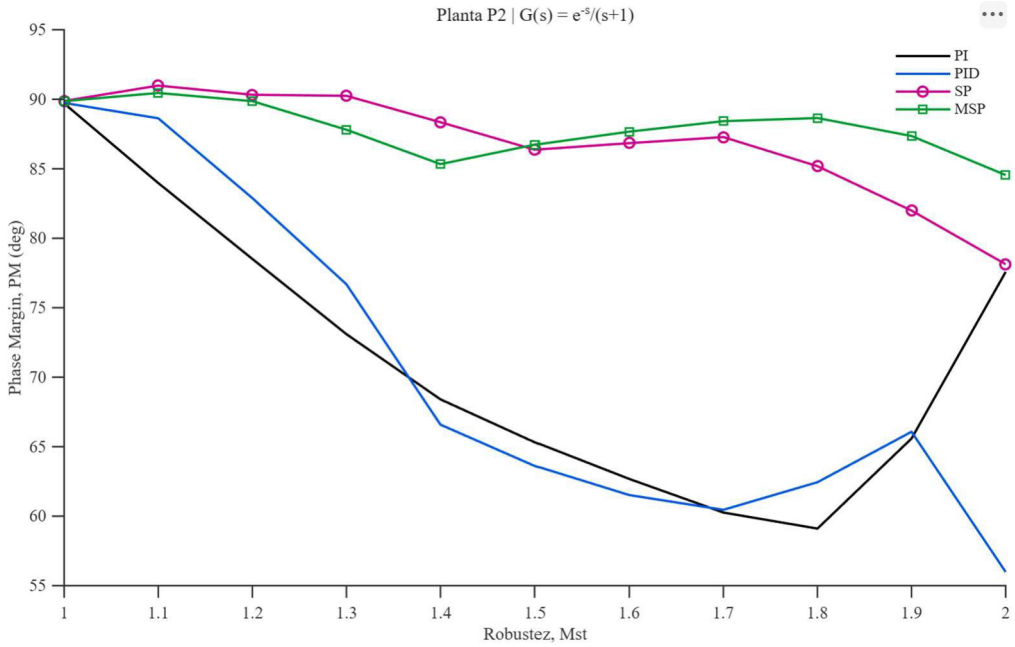

Figura 6-11. Margen de fase en función de M_{ST} para P_2

6.3.5. Simulación del efecto de errores de modelado del tiempo de retardo en la respuesta controlada

Para evaluar el comportamiento de los controladores frente a errores en el modelado del tiempo de retardo, se simuló la respuesta controlada para un valor de $M_{ST} = 1.6$. La **Figura 6-12** muestra las respuestas obtenidas ante un error de modelado del tiempo de retardo de -30%, correspondiente al escenario más desfavorable dentro de los casos evaluados.

En términos generales, los controladores no presentan una degradación significativa de la respuesta ante este nivel de error, lo cual es consistente con la tendencia observada en la curva DM vs. M_{ST}. Sin embargo, el SP es la estrategia que muestra una mayor sensibilidad relativa al desajuste del retardo, evidenciada por curvaturas más prolongadas al inicio de la respuesta controlada. A pesar de ello, los desempeños del SP, MSP y PID se mantienen bastante próximos para $M_{ST} = 1.6$, siendo el PID ligeramente superior en este punto de evaluación.

Esta cercanía entre controladores debe interpretarse dentro de la región de robustez analizada. Para valores menores de M_{ST}, las diferencias de desempeño tenderían a

incrementarse progresivamente, como se observa en la curva J vs. M_{ST}. Entre el SP y el MSP, la principal diferencia se aprecia en la respuesta ante cambios en la referencia, aunque esta no es demasiado marcada. En el caso de las perturbaciones de entrada y salida, ambas respuestas permanecen prácticamente superpuestas, lo que indica un comportamiento dinámico muy similar bajo el escenario de error considerado.

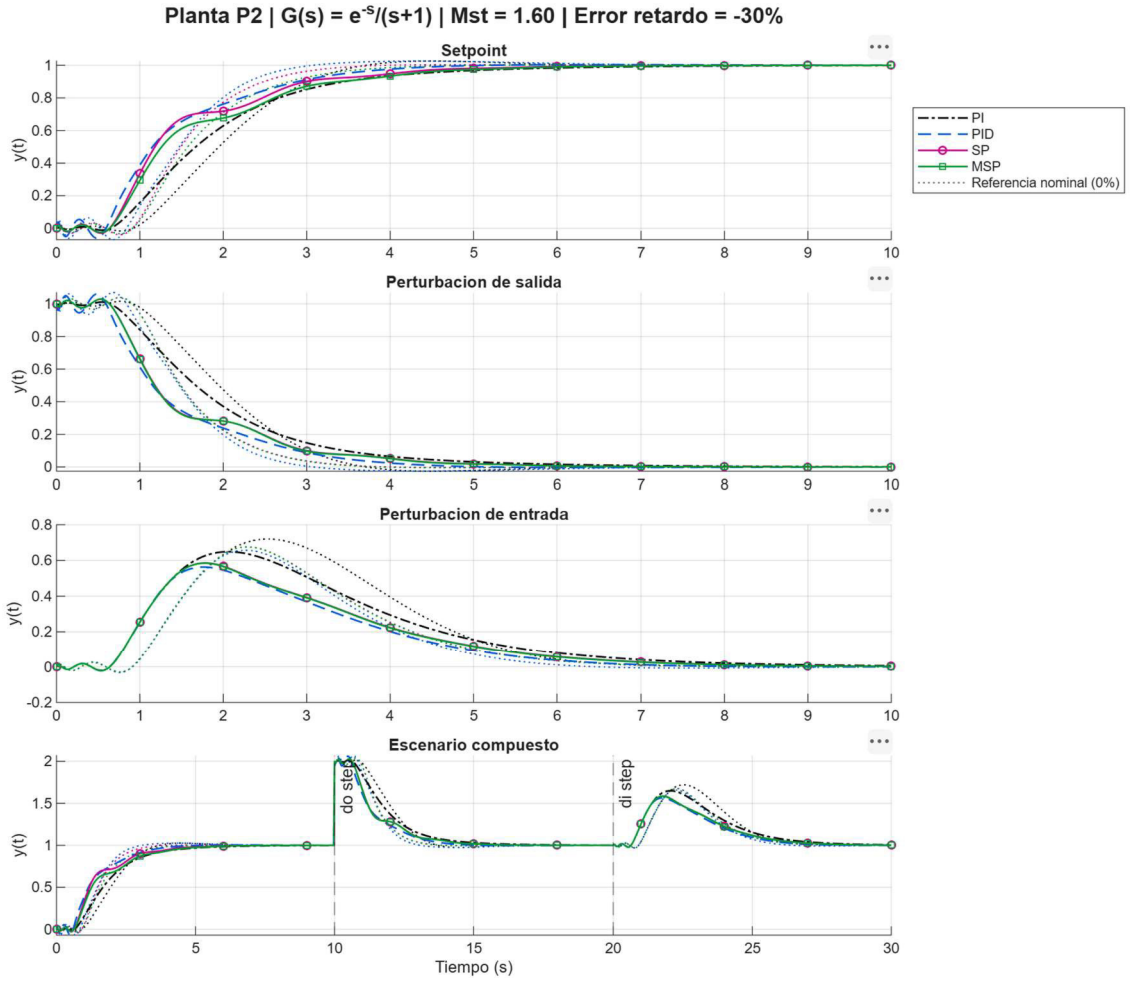

Figura 6-12. Respuestas controladas ante un error de modelado del tiempo de retardo de -30% para P_2 ($M_{ST} = 1.6$)

6.3.6. Síntesis de resultados para P_2

En síntesis, para la planta P_2, correspondiente a un proceso estable de primer orden con dinámica balanceada, las diferencias entre controladores son menos marcadas que en P_1. La presencia del polo estable $\tilde{P}_0(s) = 1/(s+1)$ hace que el problema de control no dependa únicamente de la compensación del retardo, sino también de la capacidad de cada estrategia para manejar la dinámica propia del proceso. En este caso, los

controladores PID, SP y MSP presentan desempeños cercanos, mientras que el PI queda como la alternativa menos competitiva.

Bajo condiciones de modelado ideal, el MSP alcanza el menor valor de J, aunque con una diferencia muy reducida respecto al SP y moderada respecto al PID. Por ello, su ventaja numérica no debe interpretarse como una superioridad dominante. El SP mantiene un desempeño prácticamente equivalente y, en algunas regiones de M_{ST}, incluso resulta ligeramente más favorable. El PID, por su parte, conserva un rendimiento muy competitivo debido a que la acción derivativa participa de forma efectiva en la compensación de la dinámica de primer orden, sin requerir una sintonización tan agresiva como las estructuras predictivas.

Desde el punto de vista estructural, el filtro predictivo $F_p(s)$ del MSP tiene mayor sentido en P_2 que en P_1, porque la planta ya posee una dinámica estable distinta del retardo puro. Sin embargo, esta dinámica no representa un polo problemático ni una condición particularmente exigente que justifique una ventaja clara del MSP. En consecuencia, el filtro aporta una mejora moderada, pero no suficiente para diferenciar de manera contundente al MSP frente al SP o al PID.

El análisis de robustez confirma esta lectura. SP y MSP logran mejoras de desempeño aceptando menores márgenes de retardo, mientras que el PID mantiene una mayor tolerancia frente a errores en el tiempo de retardo. En la simulación con error de modelado del retardo, los tres controladores presentan respuestas cercanas, sin una degradación significativa. El SP muestra una sensibilidad relativa ligeramente mayor, mientras que MSP y PID conservan respuestas más estables.

Por tanto, para P_2, el MSP se ubica como una alternativa competitiva, pero no claramente dominante. Su desempeño es ligeramente superior en términos de J, pero la diferencia frente al SP es mínima y frente al PID no es suficiente para justificar una ventaja concluyente bajo todas las condiciones evaluadas. En este tipo de planta balanceada, el PID sigue siendo una opción robusta y eficiente, mientras que el MSP solo resulta más atractivo si se busca mantener una estructura predictiva con buen desempeño y una respuesta algo más tolerante al desajuste del retardo que el SP.

Tabla 6-14. Resumen de resultados P_2

Bloque	Criterio	P2 – Dinámica balanceada
Rendimiento	J nominal	MSP ≈ SP > PID > PI
	Seguimiento de referencia (MST ≈ 1.6)	PID > SP > MSP > PI
	Perturbación de salida (M_{ST} ≈ 1.6)	PID > SP ≈ MSP > PI
	Perturbación de entrada (M_{ST} ≈ 1.6)	PID > MSP ≈ SP > PI
Subtotal rendimiento		**M_{ST} ≈ 1.6: PID domina.** **M_{ST} ≈ 1.7–2.0: MSP ≈ SP alcanzan los menores J.**
Robustez	Margen de retardo, DM	PI > PID > MSP > SP
	Margen de fase, PM	SP ≈ MSP > PI ≈ PID
	Error de retardo −30% (M_{ST} ≈ 1.6)	PID ≈ MSP > SP; PI con respuesta más lenta.
Subtotal robustez		**PID: mejor combinación de DM y desempeño.** **MSP: mayor tolerancia que SP.** **PI: mayor DM, pero menor rendimiento.**
Lectura general		- PID: opción más equilibrada alrededor de M_{ST} ≈ 1.6; mejor rendimiento temporal y DM alto. - MSP: más conveniente en M_{ST} ≈ 1.7–2.0 si se prioriza minimizar J; ventaja mínima sobre SP y mayor tolerancia al error de retardo. - SP: prácticamente equivalente al MSP en J, pero con menor DM y mayor sensibilidad al error de retardo. - PI: mayor DM, pero peor rendimiento; conviene solo si se prioriza fuertemente la tolerancia al retardo.

6.4. Planta P_3: Dinámica dominante

La planta P_3, definido como $P_3(s) = e^{-s}/(8s+1)$, representa un proceso estable de primer orden con una constante de tiempo mayor que la de P_2. Por tanto, aunque conserva el mismo retardo nominal, la dinámica dominante del proceso es más lenta. Esta diferencia modifica de forma importante la comparación entre controladores: la compensación del retardo ya no es el único factor determinante, sino también la capacidad de cada estructura para manejar la dinámica de primer orden asociada a $1/(8s+1)$.

A diferencia de P_1, donde el SP mostraba una ventaja muy marcada, y de P_2, donde los resultados eran más parejos, en P_3 aparece un resultado relevante: el PID alcanza el mejor desempeño global. Esto indica que, para procesos estables con dinámica dominante lenta, la acción derivativa puede ser más útil que la compensación predictiva del retardo, especialmente cuando el criterio de desempeño J pondera simultáneamente perturbaciones de salida y entrada.

6.4.1. Pesos base resultantes

Como se muestra en la **Tabla 6-15**, los pesos base para P_3 se obtuvieron alrededor de $M_{ST} = 1.595$, con $IAE_{do}^0 = 2.165$ e $IAE_{di}^0 = 1.127$. A diferencia de P_2, el peso asociado a la perturbación de entrada es considerablemente menor que el de perturbación de salida. Por ello, un aumento en IAE_{di} tiene mayor efecto relativo sobre el índice J, aspecto importante para interpretar por qué el mejor controlador global no coincide necesariamente con el que minimiza IAE_{do}.

Tabla 6-15. Pesos base para la normalización del índice J para P_3

Variable	Controlador	k_c	τ_i	τ_d	M_{ST}	IAE
IAE_{do}^0	PI	4,021	8,001	0	1,595	2,165
IAE_{di}^0	PI	3,361	3,701	0	1,595	1,127

Este punto es clave para interpretar los resultados. Un controlador puede presentar una mejora clara frente a perturbaciones de salida, pero no necesariamente obtener el menor J si su desempeño frente a perturbaciones de entrada no mejora en la misma proporción.

6.4.2. Relación J, M_{ST} y parámetros de control óptimos

En la región de M_{ST} bajo, aproximadamente entre 1.0 y 1.2, todos los controladores presentan valores altos de J, como se observa en la **Figura 6-13**. En esta zona, el PID ya

muestra una ventaja frente al PI y frente a las estructuras predictivas. Esto se confirma en la **Figura 6-14**, donde el término derivativo τ_d del PID toma valores relevantes, sobre todo en estas regiones conservadoras.

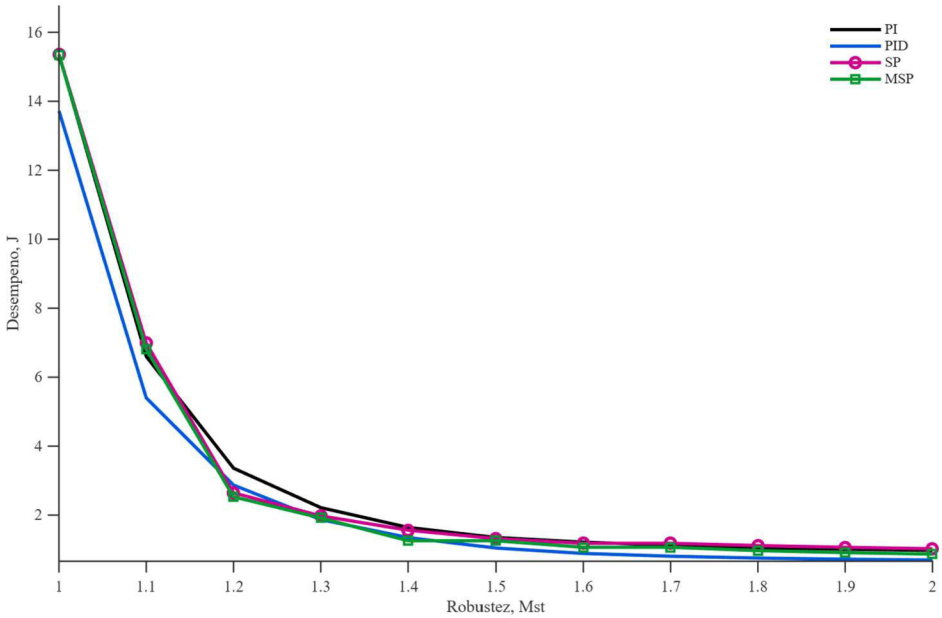

Figura 6-13. Curva de desempeño-robustez J vs. M_{ST} para P_3

En la región intermedia, aproximadamente entre M_{ST} igual a 1.2 y 1.6, el PID mantiene el mejor desempeño global. Aunque SP reduce con mayor fuerza el error asociado a perturbaciones de salida, su desempeño ante perturbaciones de entrada no mejora en la misma proporción. En general, SP logra una mayor reducción de IAE_{do} que PID, pero PID logra un IAE_{di} mucho menor. Debido a la normalización de J, esta mejora en IAE_{di} favorece al PID. Por su parte, MSP logra un rendimiento similar al PID, también causado por la mayor reducción del error asociado a la perturbación de entrada.

En las regiones próximas a $M_{ST} = 1.6$, las posiciones relativas comienzan a establecerse: PID logra la mayor reducción de J, mientras que SP se mantiene en una posición intermedia, seguido de MSP.

En la región alta, entre M_{ST} igual a 1.6 y 2.0, el orden de desempeño se consolida. Según la **Tabla 6-16**, el mejor desempeño global corresponde al PID, con $J = 0.699$ en $M_{ST} = 2.009$. SP alcanza $J = 0.795$, MSP $J = 0.871$ y PI $J = 0.977$. Aunque SP obtiene el menor IAE_{do}, con $IAE_{do} = 1.335$, el PID alcanza el menor IAE_{di}, con $IAE_{di} = 0.458$, lo que lo convierte en el mejor controlador según el índice compuesto J.

Tabla 6-16. Puntos de mínimo J por controlador para P_3

Planta	Tipo	Controlador	J	M_{ST}	IAE_{do}	IAE_{di}
P_3	Estable	PI	0,977	2,005	2,838	0,726
P_3	Estable	PID	0,699	2,009	2,147	0,458
P_3	Estable	SP	0,795	1,999	1,335	1,097
P_3	Estable	MSP	0,871	2,003	2,627	0,595

En la **Figura 6-14**, la evolución de los parámetros confirma esta interpretación. Los controladores SP y MSP requieren mayores valores de k_c en la región alta, mientras que sus valores de τ_i disminuyen de forma más marcada. Esto representa sintonías más rápidas y exigentes. En cambio, el PID no necesita incrementar k_c en la misma proporción, porque la acción derivativa aporta un mecanismo adicional para anticipar la respuesta del proceso. Así, en P_3, el buen desempeño del PID no proviene de una sintonía más agresiva que la de SP, sino de un uso más equilibrado de los tres términos del controlador.

La comparación entre SP y MSP muestra que el SP es superior en desempeño global para P_3. SP reduce de forma más marcada el error ante perturbación de salida, mientras que MSP mejora el IAE_{di} respecto al SP, pero no lo suficiente para compensar su mayor IAE_{do}. En consecuencia, el filtro $F_p(s)$ del MSP no produce una ventaja global en esta planta estable lenta; su efecto aparece más favorable sobre la perturbación de entrada, pero penaliza el rechazo de perturbaciones de salida.

Para mostrar el comportamiento de estos controladores, se simuló la respuesta controlada para un valor de $M_{ST} = 1.6$, considerando un modelado ideal de la planta. En esta planta, los tres escenarios evaluados evidencian el efecto de una dinámica más lenta, lo que se refleja en respuestas con mayores tiempos de estabilización respecto a los casos anteriores. Como se resume en la **Tabla 6-17**, el PID obtiene los menores valores de IAE tanto ante cambios en la referencia como frente a perturbaciones de entrada y salida. Esto está principalmente asociado al término derivativo independiente que posee el PID, que puede ser regulado según la necesidad, en este caso para mitigar la dinámica lenta de la planta. Esto se observa en la **Figura 6-14**, donde este término adquiere mayor protagonismo.

Este comportamiento es consistente con la curva J vs. M_{ST}, donde para valores medios y altos de M_{ST}, el PID se establece como la alternativa que ofrece el mejor compromiso entre rendimiento y robustez bajo condiciones de modelado ideal. En consecuencia, para P_3, las estructuras predictivas no presentan una ventaja clara frente al controlador PID, debido a que este logra una reducción más efectiva del error acumulado en los escenarios evaluados.

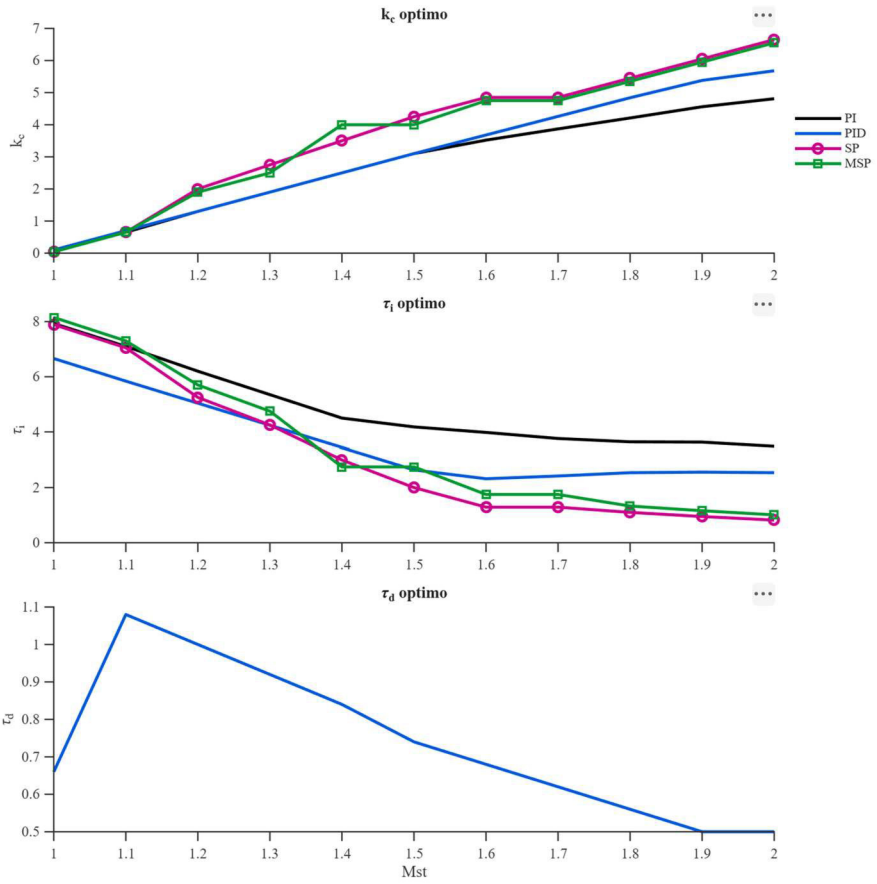

Figura 6-14. Parámetros de sintonía óptimos en función de M_{ST} para P_3

Tabla 6-17. Indicadores de rendimiento de las respuestas controladas para P_3 ($M_{ST} = 1.6$)

Controlador	Tiempo de subida	Pico absoluto	Tiempo de estabilización	Sobrepaso	Offset	IAE
Referencia						
PI	3.08	-	12.04	1.991E+01	1.008E-06	3.09
PID	2.14	-	10.14	2.017E+01	5.635E-05	2.46
SP	2.73	-	11.76	2.307E+01	-2.039E-06	2.74
MSP	3.00	-	9.43	1.836E+01	1.225E-05	2.77
Perturbación de entrada						
PI	-	0.21	13.31	-	3.863E-07	1.13
PID	-	0.16	15.34	-	3.555E-05	0.73

Controlador	Tiempo de subida	Pico absoluto	Tiempo de estabilización	Sobrepaso	Offset	IAE
SP	-	0.19	29.34	-	3.544E-03	1.24
MSP	-	0.19	12.63	-	1.923E-05	0.84
Perturbación de salida						
PI	-	1.01	12.01	-	1.008E-06	3.09
PID	-	1.08	10.07	-	5.635E-05	2.46
SP	-	1.02	11.73	-	2.039E-06	2.74
MSP	-	1.02	9.62	-	4.675E-06	3.00

Nota. Para SP se consideraron los valores asociados a 1.7, la búsqueda no encontró valores más cercanos a 1.6, ver el detalle en Anexos.

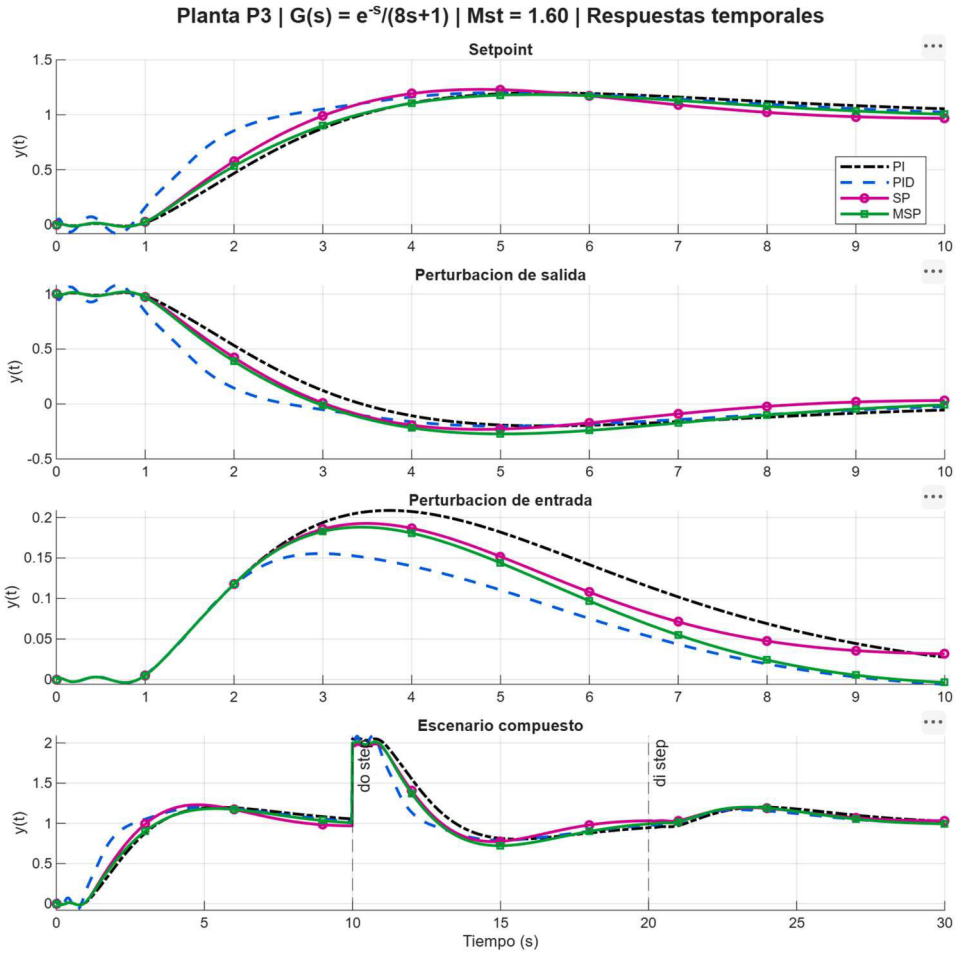

Figura 6-15. Respuestas controladas ante cambio de referencia y perturbaciones de entrada y salida para P_3 ($M_{ST} = 1.6$)

6.4.3. Interpretación estructural

Para P_3, la dinámica del proceso ya no está dominada únicamente por el retardo, sino también por un polo estable lento. Esta diferencia cambia la lectura respecto a P_1 y P_2,

porque la acción de control debe compensar simultáneamente el tiempo de retardo y la inercia propia del proceso.

Bajo modelo nominal perfecto, para SP se tiene:

$$H_{do}^{SP}(s) = \frac{1}{1 + C(s)\widetilde{P}_0(s)} \tag{6-12}$$

Donde $\tilde{P}_0(s) = 1/(8s + 1)$. Por ello, SP reduce de forma importante el IAE_{do}, ya que la perturbación de salida queda regulada por una sensibilidad construida sobre el modelo sin retardo. Mientras que para MSP, el término adicional del numerador evita que el rechazo de perturbación de salida dependa se simplifique a $1/1 + C(s)\tilde{P}_0(s)$.

$$H_{do}^{MSP}(s) = \frac{1 + C(s)\widetilde{P}_0(s)(1 + F_p(s)e^{-s})}{1 + C(s)\widetilde{P}_0(s)} \tag{6-13}$$

Para la perturbación de entrada ocurre algo distinto, la perturbación se introduce antes del proceso; por tanto, su efecto sobre la salida conserva explícitamente el factor de la planta. Como P_3 posee una dinámica lenta importante, el filtro $F_{p(s)}$ de MSP tiene más oportunidad de modificar el término residual asociado al retardo y a la dinámica del modelo. Esto permite que MSP reduzca el IAE_{di} frente a SP. No obstante, esa mejora en IAE_{di} no compensa la pérdida en IAE_{do}, por lo que MSP no supera a SP ni al PID en el índice global J.

$$H_{di}^{SP}(s) = \widetilde{P}_0(s)e^{-s}\frac{1 + C(s)\widetilde{P}_0(s)(1 + e^{-s})}{1 + C(s)\widetilde{P}_0(s)} \tag{6-14}$$

$$H_{di}^{MSP}(s) = \widetilde{P}_0(s)e^{-s}\frac{1 + C(s)\widetilde{P}_0(s)(1 + F_p(s)e^{-s})}{1 + C(s)\widetilde{P}_0(s)} \tag{6-15}$$

6.4.4. Márgenes de estabilidad y robustez

Las **Figura 6-16** y **Figura 6-17** permiten complementar el análisis desde la robustez clásica. En la **Figura 6-16**, entre M_{ST} igual a 1 y 1.5, todos los controladores presentan una caída marcada del DM. Esto indica que el aumento de agresividad en la sintonía reduce rápidamente la tolerancia frente a retardos adicionales o errores en el tiempo de retardo.

En la segunda región, desde M_{ST} igual a 1.5 hasta 2, las curvas se separan con mayor claridad. En este intervalo, SP alcanza los menores valores de DM, lo que muestra que su mejora en J se consigue a costa de una menor tolerancia frente a desajustes de retardo. En cambio, MSP conserva un margen de retardo comparable al PID y superior al SP. Esto es importante porque matiza la interpretación del filtro: aunque MSP no supera a SP en desempeño global J, sí introduce una sintonía menos agresiva frente al retardo, manteniendo una reserva temporal más parecida a la de un controlador convencional bien ajustado.

Además, a diferencia de P_1 y P_2, en P_3 el PID resulta especialmente competitivo porque la acción derivativa sí actúa sobre una dinámica lenta real. Esto se observa en que τ_d permanece distinto de cero durante todo el barrido. En consecuencia, el PID no depende únicamente de aumentar k_c, sino que utiliza la acción derivativa para anticipar parcialmente la respuesta de la planta y mejorar el compromiso entre IAE_{do}, IAE_{di} y robustez.

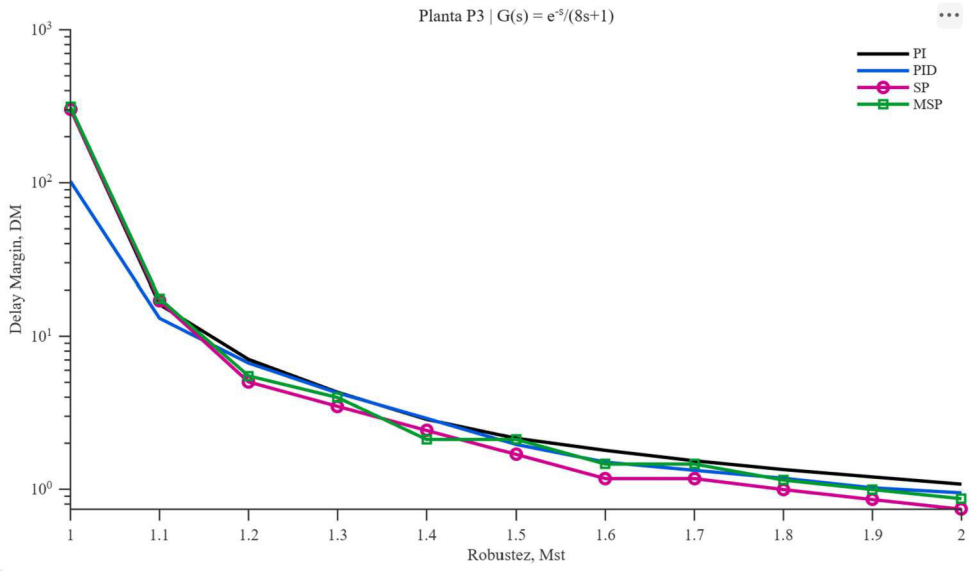

Figura 6-16. Margen de retardo en función de M_{ST} para P_3

La **Figura 6-17** muestra que SP y MSP mantienen mayores márgenes de fase que PI y PID en buena parte del barrido. Esto indica que las estructuras predictivas conservan holgura frente a atraso de fase general. Sin embargo, como se explicó para las plantas anteriores, un mayor PM no implica necesariamente mayor tolerancia a retardo adicional. El margen de retardo depende también de la frecuencia de cruce; por ello, una sintonía

más rápida puede conservar un PM aceptable y, al mismo tiempo, presentar menor DM.

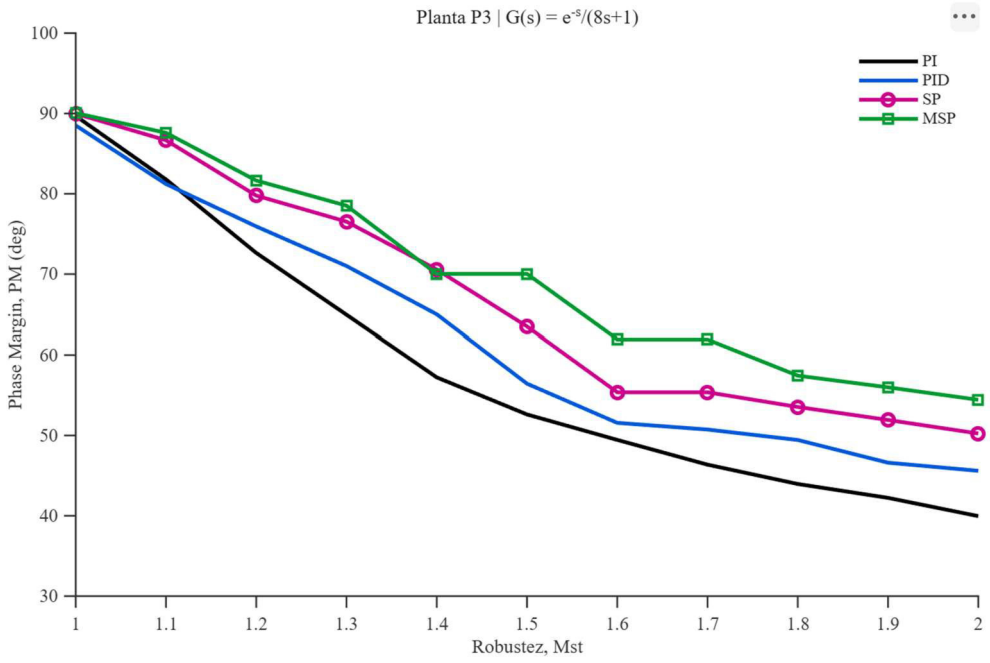

Figura 6-17. Margen de fase en función de M_{ST} para P_3

En P_3, esta diferencia es especialmente relevante. SP y MSP tienen márgenes de fase más altos, pero no alcanzan el menor J. El PID, aunque opera con menores márgenes de fase en la región alta, obtiene el mejor desempeño global porque reduce simultáneamente IAE_{do} e IAE_{di}, especialmente este último. Por tanto, el resultado de P_3 confirma que los márgenes de estabilidad y el índice J deben interpretarse de forma complementaria, no como criterios equivalentes.

6.4.5. Simulación del efecto de errores de modelado del tiempo de retardo en la respuesta controlada

Para evaluar el comportamiento de los controladores frente a errores en el modelado del tiempo de retardo, se simuló la respuesta controlada para un valor de $M_{ST} = 1.6$. La **Figura 6-18**, muestra las respuestas obtenidas ante un error de modelado del tiempo de retardo de +30%, correspondiente al escenario más desfavorable dentro de los casos evaluados.

En términos generales, los controladores no presentan una degradación significativa de la respuesta ante este nivel de error, solo una intensificación en los puntos máximos y mínimos, lo cual es consistente con la tendencia observada en la curva DM vs. M_{ST}. Sin embargo, el SP es la estrategia que muestra una mayor sensibilidad relativa al desajuste

del retardo, evidenciada por curvaturas más prolongadas al inicio de la respuesta controlada. A pesar de ello, los desempeños del SP, MSP y PID se mantienen bastante próximos para $M_{ST} = 1.6$, siendo el PID ligeramente superior en este punto de evaluación.

Esta cercanía entre controladores debe interpretarse dentro de la región de robustez analizada. Para valores mayores de M_{ST}, las diferencias de desempeño tenderían a incrementarse progresivamente, favoreciendo al PID, como se observa en la curva J vs. M_{ST}.

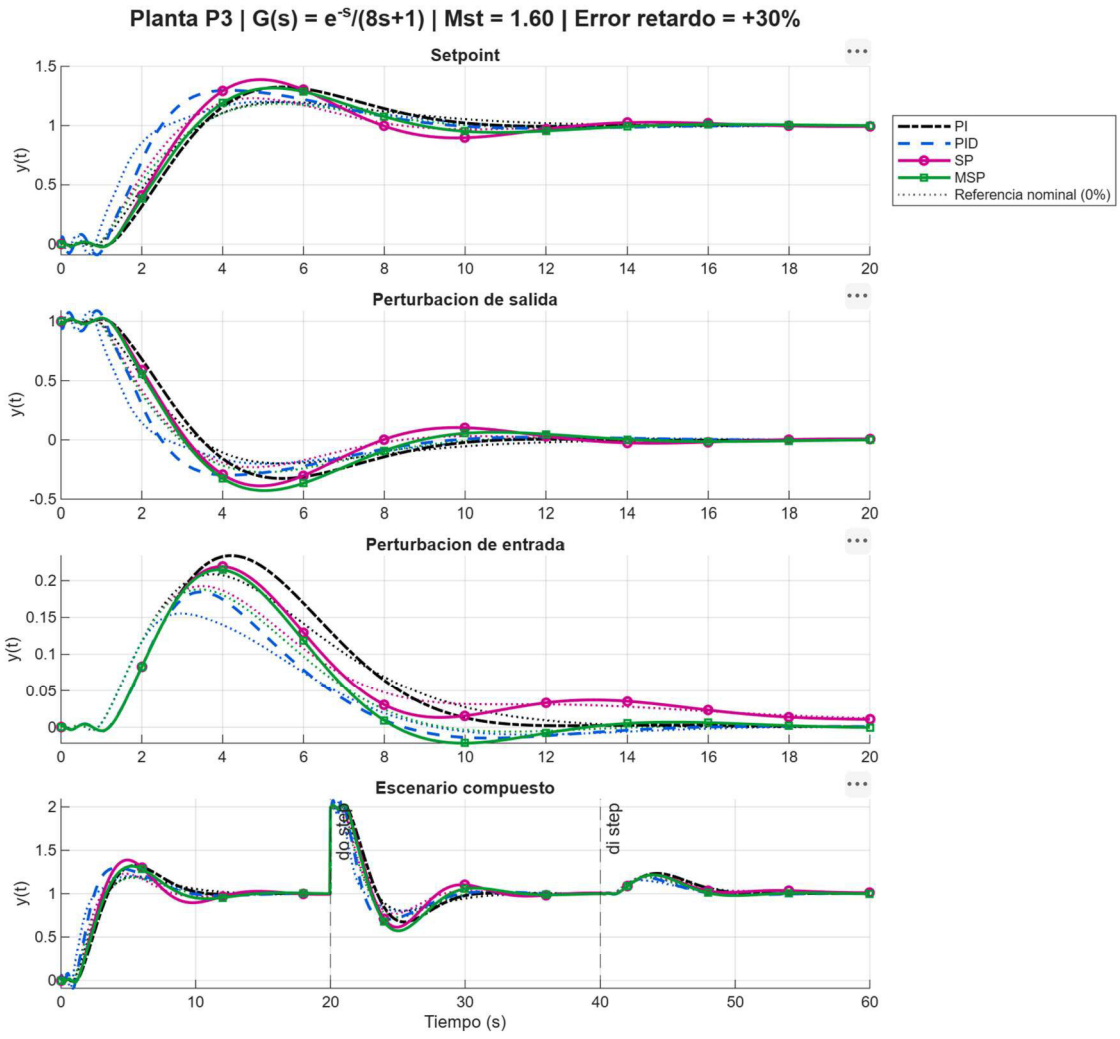

Figura 6-18. Respuestas controladas ante un error de modelado del tiempo de retardo de +30% para P_3 ($M_{ST} = 1.6$)

6.4.6. Síntesis de resultados para P_3

En síntesis, para la planta P_3, correspondiente a un proceso estable de primer orden con retardo y dinámica lenta, el controlador PID presenta el mejor desempeño global entre las

estrategias evaluadas. A diferencia de P_1, donde la compensación directa del retardo favorecía claramente al SP, en P_3 la dinámica dominante del proceso también está determinada por el polo lento de la planta. En este contexto, la acción derivativa del PID adquiere mayor relevancia, ya que permite anticipar parcialmente la respuesta del sistema sin requerir incrementos tan agresivos de k_c. Por ello, el PID logra una mejor reducción conjunta de IAE_{do} e IAE_{di}, obteniendo el menor valor de J en las regiones medias y altas de M_{ST}.

El SP mantiene un desempeño competitivo, especialmente en el rechazo de perturbaciones de salida, donde logra reducir de forma importante el error acumulado. Sin embargo, su desempeño frente a perturbaciones de entrada es menos favorable, lo que penaliza su valor global de J. Por su parte, el MSP mejora el comportamiento frente a perturbaciones de entrada respecto al SP, pero esta mejora no compensa la pérdida observada en la perturbación de salida. En consecuencia, para P_3, el filtro predictivo $F_p(s)$ no genera una ventaja global suficiente frente al SP ni frente al PID bajo condiciones de modelado ideal.

El análisis de robustez permite matizar esta conclusión. Aunque el MSP no alcanza el mejor desempeño nominal, conserva un margen de retardo superior al del SP y más cercano al del PID, lo que indica una mayor tolerancia relativa frente a errores en el tiempo de retardo. Esta diferencia también se refleja en la simulación con error de modelado del retardo, donde el SP muestra una mayor sensibilidad relativa, mientras que el MSP presenta una menor sensibilidad relativa al desajuste del retardo. Sin embargo, la degradación general de los controladores no es significativa en el escenario evaluado, y el PID conserva un comportamiento ligeramente superior.

Por tanto, para P_3, el PID se posiciona como la alternativa más conveniente, al ofrecer el mejor compromiso entre rendimiento y robustez. El SP resulta atractivo cuando se prioriza el rechazo de perturbaciones de salida, pero pierde competitividad por su mayor sensibilidad al retardo y su peor desempeño ante perturbaciones de entrada. El MSP mejora la robustez relativa del SP y ofrece una respuesta más equilibrada frente al desajuste del tiempo de retardo, pero no logra superar al PID ni al SP en desempeño nominal global. En consecuencia, para procesos estables con dinámica lenta como P_3, la estructura predictiva modificada no constituye la mejor alternativa; su aporte principal es

mejorar la tolerancia relativa al error de retardo frente al SP, pero el PID sigue siendo la estrategia dominante en términos de desempeño global.

Tabla 6-18. Resumen de resultados P_3

Bloque	Criterio	P3 – Dinámica dominante
Rendimiento	J nominal	PID > SP > MSP > PI
	Seguimiento de referencia ($M_{ST} \approx 1.6$)	PID > SP > MSP > PI
	Perturbación de salida ($M_{ST} \approx 1.6$)	PID > SP > MSP > PI
	Perturbación de entrada ($M_{ST} \approx 1.6$)	PID > MSP > PI > SP
Subtotal rendimiento		**PID domina globalmente.** **SP destaca en perturbación de salida;** **MSP mejora a SP en perturbación de entrada.**
Robustez	Margen de retardo, DM	PI > MSP > PID > SP
	Margen de fase, PM	MSP > SP > PID > PI
	Error de retardo +30% ($M_{ST} \approx 1.6$)	PID > MSP ≈ SP > PI
Subtotal robustez		**MSP presenta mejores márgenes que SP;** **PID mantiene robustez suficiente con mejor rendimiento global.**
Lectura general		- PID: alternativa preferente para P_3; desde regiones medias-altas de M_{ST} obtiene el menor J y alrededor de $M_{ST} \approx 1.6$ domina las respuestas temporales. - MSP: preferible a SP cuando se prioriza perturbación de entrada o tolerancia al error de retardo, pero no supera al PID globalmente. - SP: útil si se prioriza específicamente perturbación de salida, a costa de menor DM y peor respuesta ante perturbación de entrada. - PI: presenta el mayor DM, pero el peor rendimiento global.

6.5. Planta P_4: Próximo al integrador

La planta P_4 corresponde al proceso estable $P_4(s) = e^{-s}/(20s + 1)$, caracterizado por una dinámica de primer orden considerablemente más lenta que las plantas P_2 y P_3, por lo que la respuesta del proceso está fuertemente condicionada por la inercia del término $(20s + 1)$. Esto hace que la mejora del desempeño requiera acciones de control más exigentes, especialmente cuando se busca operar en regiones altas de M_{ST}.

6.5.1. Pesos base resultantes

Como se muestra en la **Tabla 6-19**, los pesos base para P_4 se obtuvieron alrededor de $M_{ST} = 1.595$, con $IAE_{do}^{0} = 2.165$ e $IAE_{di}^{0} = 0.595$. El valor reducido de IAE_{di}^{0} hace que el índice J sea particularmente sensible al desempeño ante perturbaciones de entrada. Por esta razón, pequeñas diferencias en IAE_{di} pueden tener un efecto importante en la comparación global de controladores.

Tabla 6-19. Pesos base para la normalización del índice J para P_4

Variable	Controlador	k_c	τ_i	τ_d	M_{ST}	IAE
IAE_{do}^{0}	PI	10.05	19.991	0	1.595	2.165
IAE_{di}^{0}	PI	8.161	4.721	0	1.595	0.595

6.5.2. Relación J, M_{ST} y parámetros de control óptimos

Como se observa en la **Figura 6-19**, todos los controladores reducen el índice de desempeño al aumentar M_{ST}, pero no lo hacen con la misma regularidad. Para esta planta, conviene dividir el análisis en cuatro regiones.

En la región conservadora inicial, entre M_{ST} igual a 1.0 y 1.3, todos los controladores presentan una disminución rápida de J. Esta caída inicial indica que pasar de una sintonía muy conservadora a una moderadamente más exigente tiene un impacto importante en el desempeño. En esta zona, PID muestra una ventaja inicial sobre PI, SP y MSP, principalmente por el uso de la acción derivativa. Esto confirma que, para una planta lenta, la acción derivativa puede anticipar parte de la dinámica del proceso y mejorar el compromiso inicial entre desempeño y robustez.

Entre M_{ST} igual a 1.3 y 1.4, las curvas entran en una región de transición. PI continúa mejorando, aunque con menor pendiente, mientras que PID reduce J de forma más

gradual. SP y MSP comienzan a acercarse a los mejores valores, pero todavía no presentan una superioridad clara. Como se muestra en la **Figura 6-20**, en esta región, los parámetros muestran incrementos progresivos de k_c y disminuciones graduales de τ_i, sin saltos abruptos. Esto sugiere que, hasta aproximadamente $M_{ST} = 1.4$, la mejora de desempeño se obtiene por un ajuste continuo de la agresividad de la sintonía.

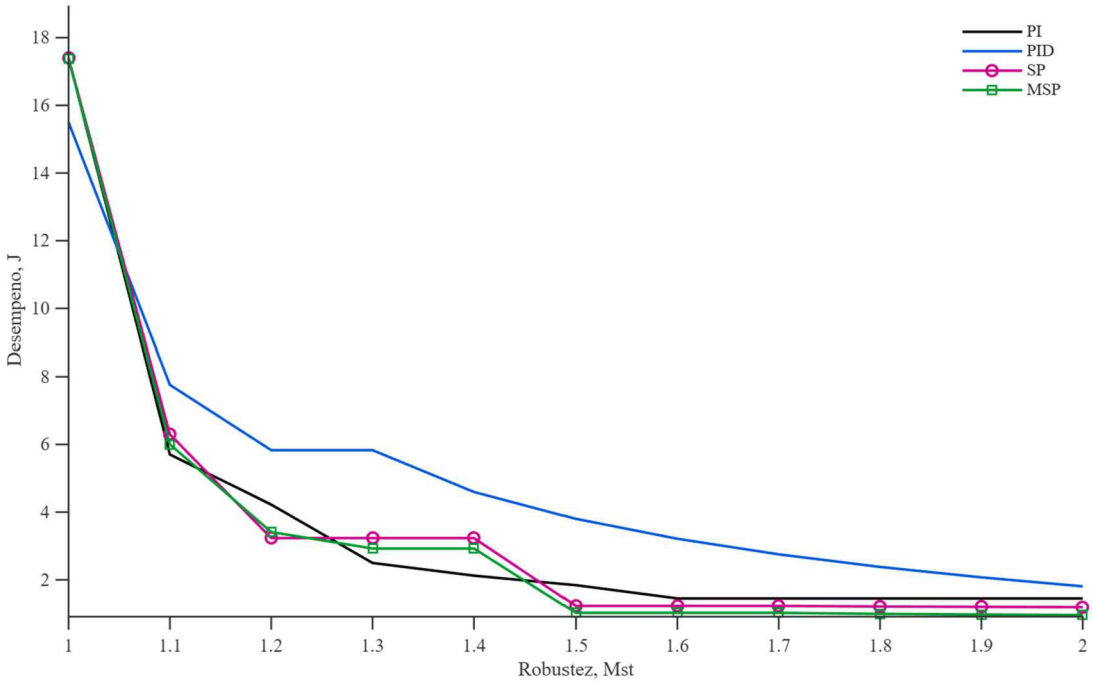

Figura 6-19. Curva de desempeño-robustez J vs. M_{ST} para P_4

El cambio más importante aparece entre M_{ST} igual a 1.4 y 1.5. En ese intervalo, SP y MSP presentan un cambio brusco de régimen: k_c aumenta hasta valores cercanos al límite superior de búsqueda, mientras que τ_i, cae abruptamente hacia valores pequeños. Este cambio coincide con la caída de J hacia valores cercanos a la unidad. Por tanto, la mejora de SP y MSP en P_4 no debe interpretarse como una ventaja gradual, sino como el resultado de una sintonía mucho más agresiva que aparece a partir de esa región.

Finalmente, entre M_{ST} igual a 1.5 y 2, SP y MSP se estabilizan como las estrategias con menor J. Según la **Tabla 6-20**, el menor valor de J corresponde a MSP, con $J = 0.959$ en $M_{ST} = 2.002$, seguido muy de cerca por SP, con $J = 0.962$ en $M_{ST} = 2.003$. La diferencia entre ambos es mínima, por lo que no debe interpretarse como una superioridad marcada del MSP, sino como un empate práctico en términos de desempeño global. SP obtiene menor IAE_{do}, mientras que MSP obtiene menor IAE_{di}. Esto confirma que las dos estrategias predictivas no mejoran el desempeño por el mismo mecanismo: SP favorece

el rechazo de perturbaciones de salida, mientras que MSP redistribuye el desempeño hacia el canal de perturbación de entrada. Es importante mencionar que la ventaja en la reducción de J respecto a los controladores tradicionales sucede a expensas de una pérdida en el margen de ganancia, como se analiza en la sección posterior.

Tabla 6-20. Puntos de mínimo J por controlador para P_4

Planta	Tipo	Controlador	J	M_{ST}	IAE_{do}	IAE_{di}
P_4	Estable	PI	1.449	1.638	3.090	0.876
P_4	Estable	PID	1.807	1.995	3.153	1.284
P_4	Estable	SP	0.962	2.003	1.232	0.806
P_4	Estable	MSP	0.959	2.002	2.583	0.432

Particularmente, para P_4, el PI resulta mejor que el PID porque el criterio J favorece la reducción integral del error, y en una planta lenta esa reducción depende principalmente de la acción proporcional-integral. En este caso, el PI alcanza una sintonía más directa para reducir el error acumulado, con $k_c = 10$ y $\tau_i = 9$, mientras que el PID obtiene su mejor punto con una ganancia de $k_c = 6.3$, $\tau_i = 8.2$ y $\tau_d = 1.7$. La presencia de τ_d no significa que la acción derivativa mejore el valor de J; más bien, indica que el buscador encontró una combinación en la que la derivada ayuda a conservar estabilidad y margen de fase, permitiendo una respuesta más robusta, aunque menos eficiente en términos de error acumulado.

La comparación entre SP y MSP revela un intercambio entre las dos perturbaciones. SP presenta el menor IAE_{do}, con $IAE_{do} = 1.232$, mientras que MSP logra el menor IAE_{di}, con $IAE_{di} = 0.432$. Dado que el peso base de IAE_{di} es reducido, la mejora del MSP en perturbación de entrada compensa su mayor IAE_{do}, produciendo un J apenas menor que el de SP. Esto sugiere que, para P4, el MSP aporta una ventaja principalmente asociada al rechazo de perturbaciones de entrada.

La **Figura 6-20** muestra un aspecto importante de la sintonía: tanto SP como MSP alcanzan valores de k_c cercanos al límite superior de búsqueda, aproximadamente $k_c = 19.991$, en la región alta de M_{ST}. Además, sus tiempos integrales disminuyen de forma marcada. Por tanto, aunque los predictores ofrecen el mejor desempeño, estos resultados

deben interpretarse considerando que las soluciones se encuentran cerca de la frontera del dominio explorado.

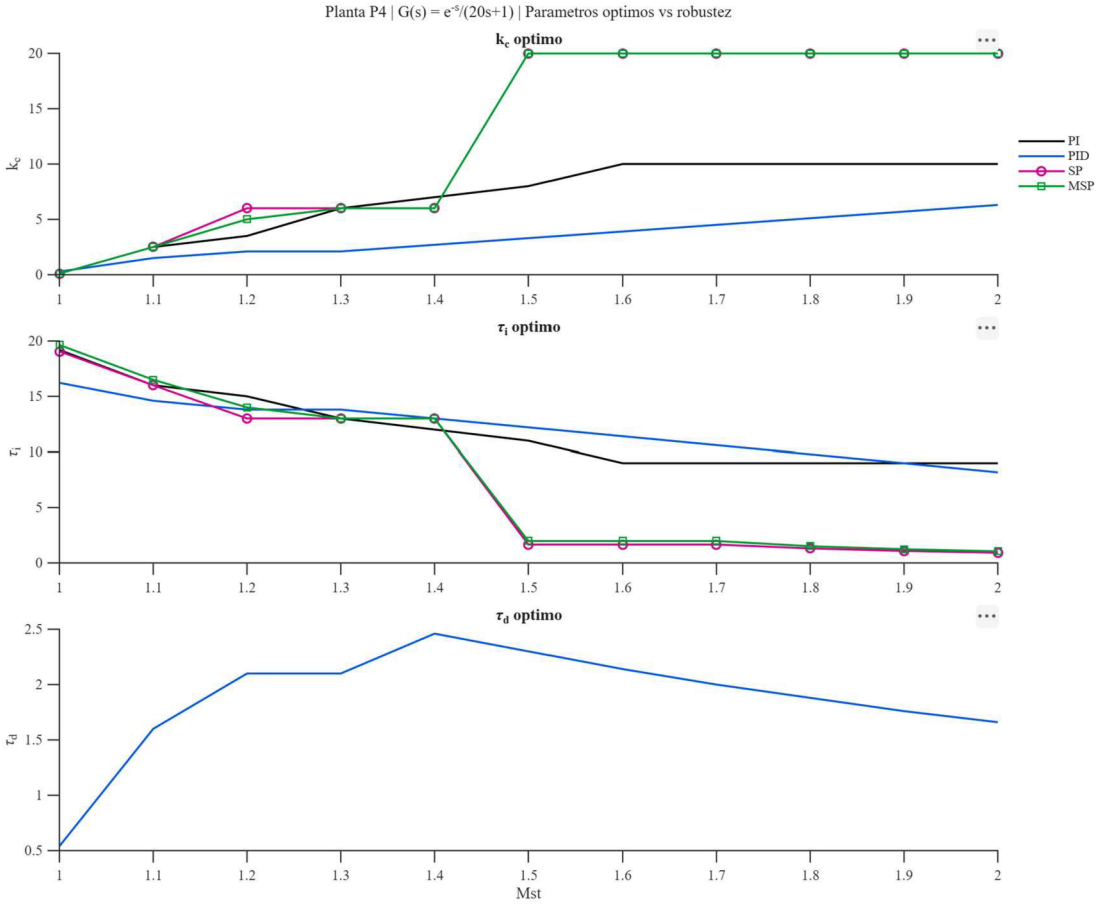

Figura 6-20. Parámetros de sintonía óptimos en función de M_{ST} para P_4

Para mostrar el comportamiento de estos controladores, se simuló la respuesta controlada para un valor de $M_{ST} = 1.6$, considerando un modelado ideal de la planta. Como se observa en la **Figura 6-21 y** la **Tabla 6-21**, el controlador PID no logra responder adecuadamente frente a una dinámica próxima al integrador, por lo que no ofrece una respuesta aceptable en los escenarios evaluados.

Por otro lado, el MSP presenta el mejor comportamiento global respecto al SP, especialmente ante perturbaciones de entrada, donde alcanza una respuesta más favorable en términos de error acumulado y tiempo de estabilización. Sin embargo, el SP mantiene un desempeño competitivo, particularmente frente a cambios en la referencia y perturbaciones de salida. A diferencia de lo observado en otras plantas, el controlador PI también muestra un comportamiento relativamente aceptable: aunque su respuesta es más lenta, logra un desempeño más cercano al de las estrategias predictivas y claramente

superior al PID. En consecuencia, para este tipo de dinámica próxima al integrador, las estrategias predictivas muestran una ventaja clara bajo condiciones de modelado ideal, siendo el MSP la alternativa con mejor desempeño global bajo condiciones de modelado ideal.

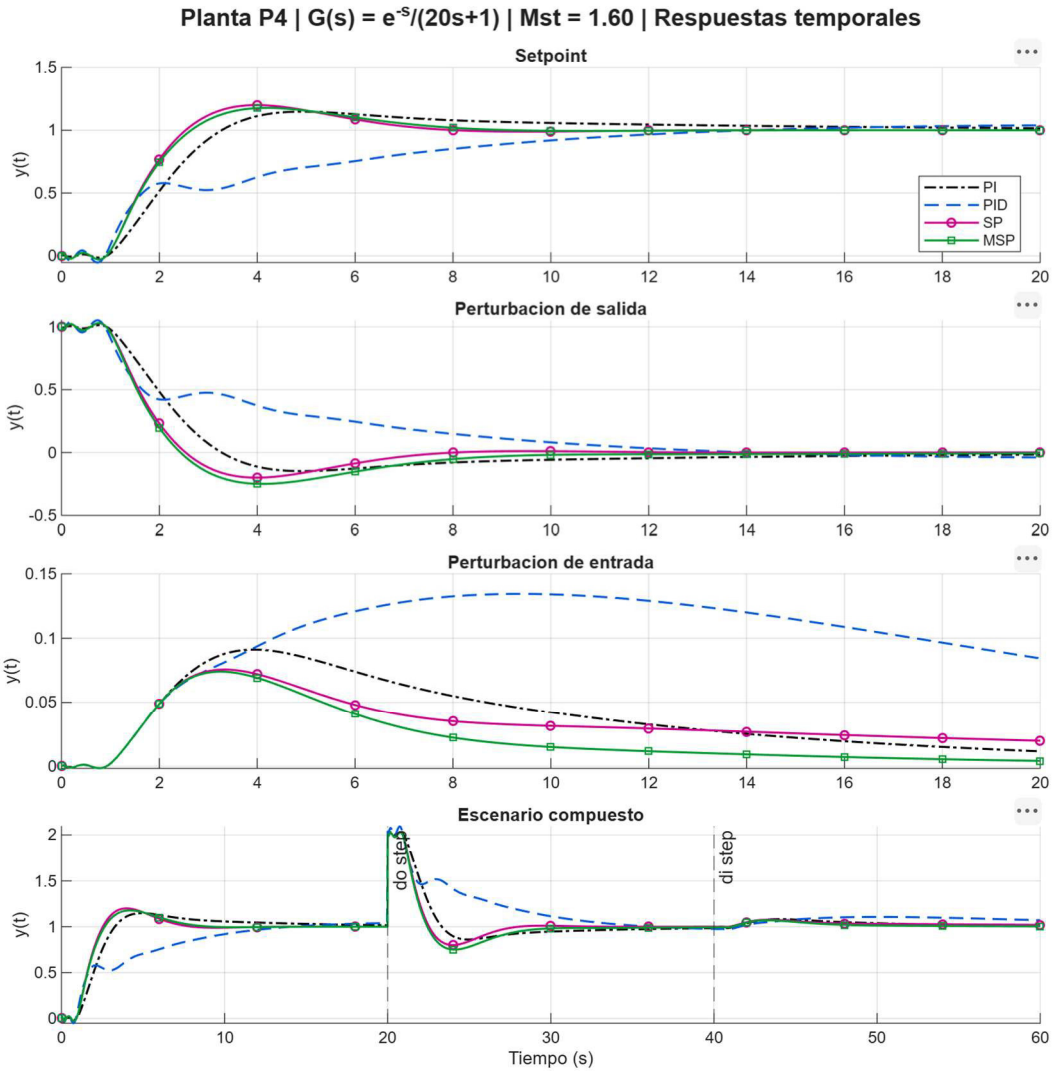

Figura 6-21. Respuestas controladas ante cambio de referencia y perturbaciones de entrada y salida para P_4 ($M_{ST} = 1.6$)

Tabla 6-21. Indicadores de rendimiento de las respuestas controladas para P_4 ($M_{ST} = 1.6$)

Controlador	Tiempo de subida	Pico absoluto	Tiempo de estabilización	Sobrepaso	Offset	IAE
Referencia						
PI	2.92	-	16.76	1.429E+01	-4.439E-03	3.09
PID	10.39	-	15.20	8.732E-01	-3.294E-02	4.37
SP	2.28	-	7.29	1.999E+01	-5.622E-09	2.25
MSP	2.35	-	7.99	1.776E+01	-1.742E-07	2.26

Controlador	Tiempo de subida	Pico absoluto	Tiempo de estabilización	Sobrepaso	Offset	IAE
Perturbación de entrada						
PI	-	0.09	*	-	3.196E-03	0.88
PID	-	0.13	*	-	3.560E-02	2.62
SP	-	0.08	*	-	1.203E-02	0.84
MSP	-	0.07	27.71	-	1.087E-03	0.48
Perturbación de salida						
PI	-	1.01	18.22	-	4.439E-03	3.09
PID	-	1.05	*	-	3.294E-02	4.37
SP	-	1.03	7.27	-	5.622E-09	2.25
MSP	-	1.03	9.74	-	1.610E-03	2.69

(*) Nota: Valores fueron omitidos por el programa al no encontrar el punto de estabilización en el tiempo de simulación.

6.5.3. Interpretación estructural

La interpretación es similar a lo ocurrido en la P_3. La respuesta de P_4 confirma el efecto de la dinámica lenta sobre las estrategias predictivas. En una planta con denominador $(20s + 1)$, la perturbación de entrada queda fuertemente condicionada por la dinámica del proceso. Por eso, el filtro de MSP tiene mayor posibilidad de mejorar la respuesta controlada a la perturbación de entrada. Esto explica que MSP alcance el menor IAE_{di}.

Sin embargo, el mismo filtro no produce una mejora equivalente en la perturbación de salida. SP mantiene una acción más directa sobre la sensibilidad del modelo sin retardo, mientras que MSP introduce el efecto del filtro en el término residual. Por ello, SP obtiene menor IAE_{do}, mientras que MSP sacrifica parte de ese rechazo para mejorar la respuesta a la perturbación de entrada.

También debe destacarse que la ventaja de SP/MSP aparece asociada a una sintonía muy agresiva. El salto de k_c y la reducción brusca de τ_i indican que las estrategias predictivas explotan con fuerza la compensación del retardo. Esto mejora el desempeño nominal, pero también puede aumentar la sensibilidad frente a errores de modelado, especialmente en el tiempo de retardo o en la constante de tiempo dominante.

6.5.4. Márgenes de estabilidad y robustez

Siguiendo lo mostrado en la **Figura 6-22**, entre M_{ST} igual a 1 y 1.3, todos los controladores presentan una caída inicial importante del DM. Esto es esperable, porque al aumentar la agresividad de la sintonía se incrementa la frecuencia de cruce y se reduce la tolerancia frente a retardos adicionales.

Entre M_{ST} igual a 1.3 y 1.6, PID se separa como el controlador con mayor margen de retardo. PI conserva valores intermedios, mientras que SP y MSP empiezan a perder DM de forma más marcada. En esta zona, aunque las estrategias predictivas comienzan a mejorar J, todavía no justifican completamente la pérdida de margen frente a PID.

A partir de $M_{ST} = 1.7$, la diferencia es mucho más clara. SP y MSP caen a valores de DM cercanos a la unidad o inferiores, mientras que PID conserva el mayor margen de retardo. Esto significa que la mejora de J obtenida por SP/MSP se logra reduciendo la tolerancia frente a errores en el tiempo de retardo. En otras palabras, son mejores en desempeño nominal, pero menos robustos frente a desajustes de retardo.

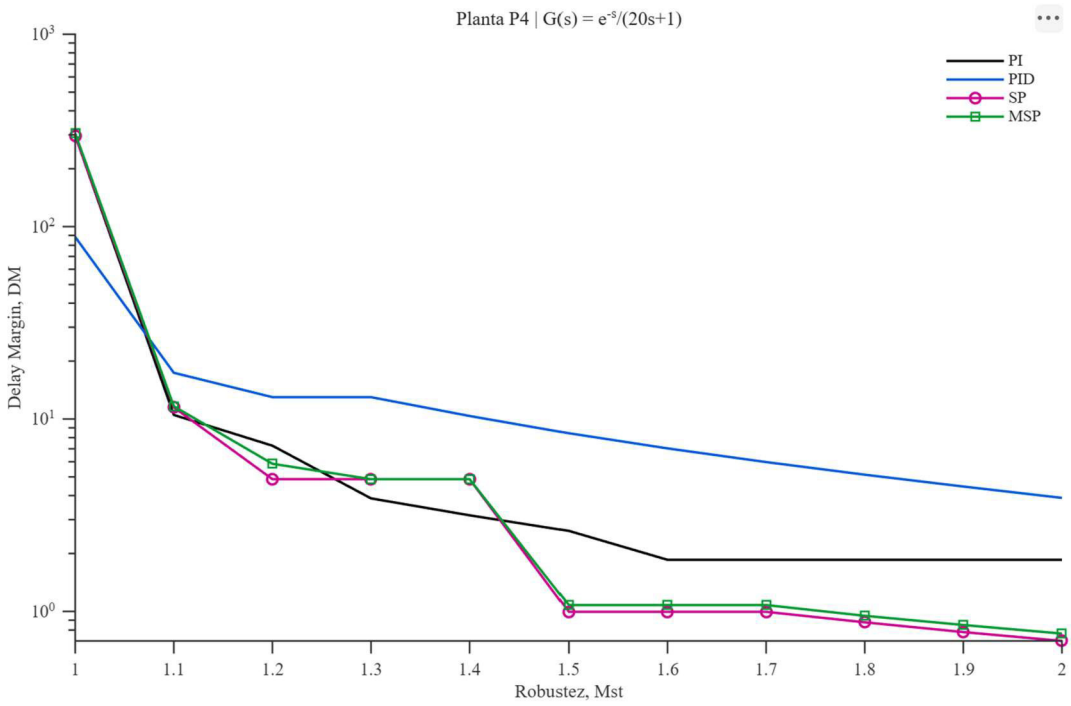

Figura 6-22. Margen de retardo en función de M_{ST} para P_4

Siguiendo la **Figura 6-23**, hasta aproximadamente $M_{ST} = 1.6$, SP y MSP mantienen valores de PM relativamente altos, mientras que PID incluso incrementa su margen de fase y se mantiene como la alternativa con mayor reserva angular. Esta región corresponde a una sintonía todavía progresiva.

Después de $M_{ST} = 1.7$, SP y MSP presentan una caída brusca del margen de fase. Esta reducción coincide con el salto de k_c , la disminución de τ_i y la caída del DM. Por tanto, el cambio de régimen no solo mejora J, sino que también reduce las reservas de estabilidad frente a incertidumbres de fase, errores de modelo o dinámica no considerada.

PID, aunque no obtiene el menor J, conserva el mayor margen de fase en la región alta de M_{ST}. Esto lo convierte en una alternativa más robusta desde el punto de vista de estabilidad relativa. Así, para P_4 se observa con claridad el compromiso entre desempeño y robustez: SP/MSP logran mejores valores de J, mientras PID conserva mayores márgenes.

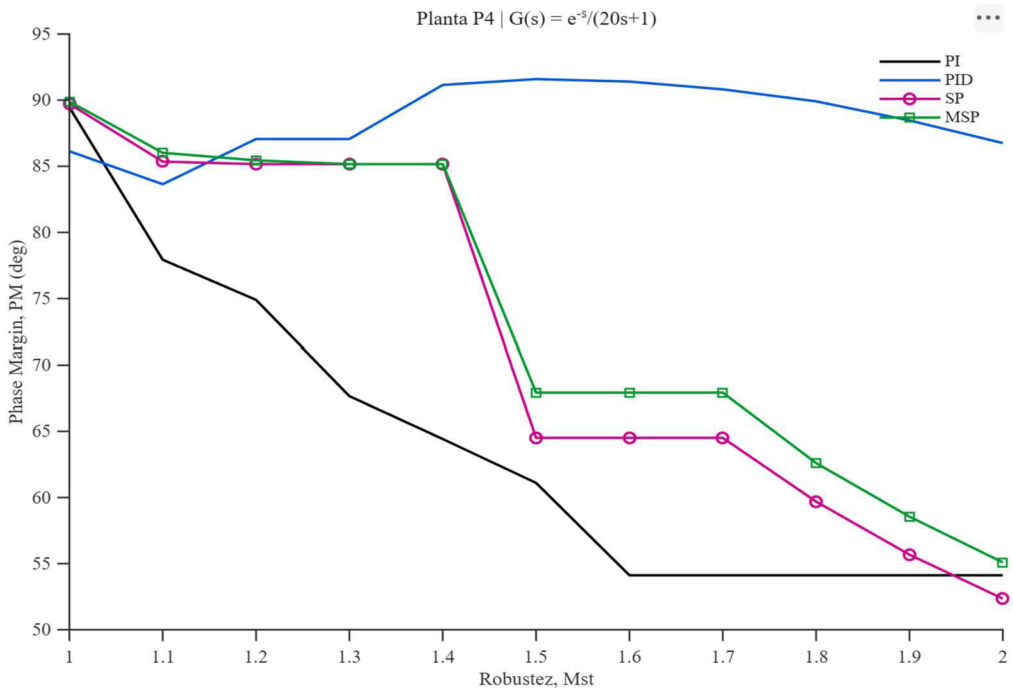

Figura 6-23. Margen de fase en función de M_{ST} para P_4

6.5.5. Simulación del efecto de errores de modelado del tiempo de retardo en la respuesta controlada

Para evaluar el comportamiento de los controladores frente a errores en el modelado del tiempo de retardo, se simuló la respuesta controlada para un valor de $M_{ST} = 1.6$. La **Figura 6-24**, muestra las respuestas obtenidas ante un error de modelado del tiempo de retardo de +30%, correspondiente al escenario más desfavorable dentro de los casos evaluados.

En términos generales, los controladores no presentan una degradación significativa de la respuesta ante este nivel de error, solo una intensificación en los puntos máximos y mínimos, lo cual es consistente con la tendencia observada en la curva DM vs. M_{ST}. MSP, seguido de SP, presenta una mejor respuesta respecto a los controladores convencionales.

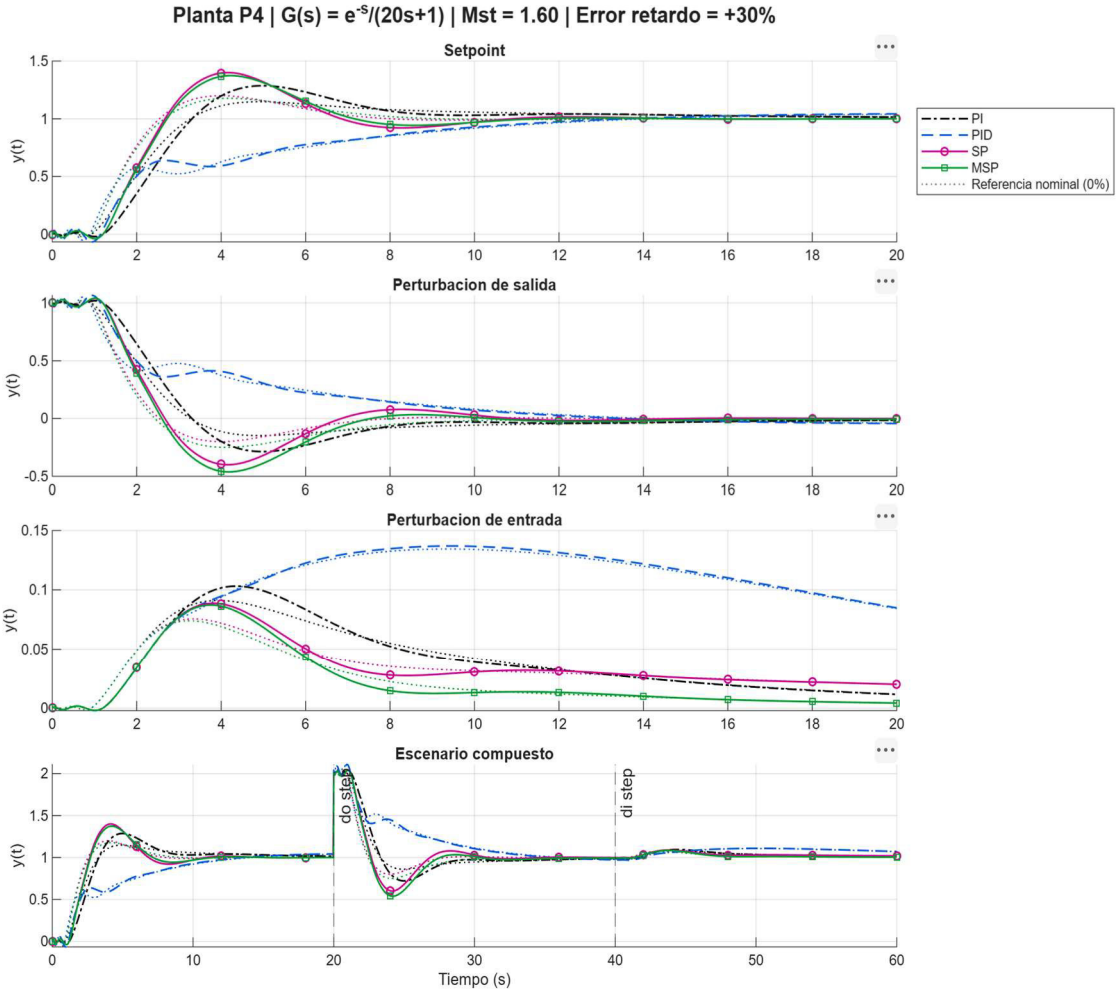

Figura 6-24. Respuestas controladas ante un error de modelado del tiempo de retardo de +30% para P4 ($M_{ST} = 1.6$)

6.5.6. Síntesis de resultados para P4

En síntesis, para la planta P4, correspondiente a un proceso estable próximo al integrador, las estrategias predictivas presentan una ventaja clara en términos de desempeño nominal frente a los controladores convencionales. La elevada constante de tiempo de la planta hace que la respuesta esté fuertemente condicionada por la dinámica lenta del proceso, por lo que la mejora del rendimiento requiere sintonías más exigentes. En este contexto, SP y MSP alcanzan los menores valores de J, especialmente en las regiones altas de M_{ST},donde ambos controladores incrementan significativamente k_c y reducen τ_i, evidenciando una sintonización más agresiva.

La comparación entre SP y MSP muestra un intercambio entre canales de perturbación. El SP obtiene mejores resultados frente a perturbaciones de salida, mientras que el MSP logra una mayor reducción del error asociado a perturbaciones de entrada. Debido a que

el índice J es especialmente sensible a IAE_{di} en esta planta, la mejora del MSP en la perturbación de entrada compensa su menor desempeño en perturbación de salida, permitiéndole alcanzar un valor de J apenas menor que el SP. Por ello, la superioridad del MSP sobre el SP debe interpretarse como una ventaja leve, pero técnicamente relevante, asociada principalmente al rechazo de perturbaciones de entrada.

Frente a los controladores convencionales, el PI presenta un comportamiento más aceptable que el PID. Aunque su respuesta es más lenta, logra un desempeño más cercano al de las estrategias predictivas. En cambio, el PID no resulta conveniente cuando se prioriza el rendimiento nominal, ya que la acción derivativa no compensa adecuadamente la dinámica próxima al integrador y termina generando un peor desempeño global. No obstante, desde el punto de vista de robustez clásica, el PID conserva mayores márgenes de retardo y de fase, lo que confirma que su limitación principal no es la estabilidad, sino su baja eficiencia para reducir el error acumulado en este caso.

El análisis de robustez muestra que la mejora de SP y MSP no es gratuita. En las regiones donde alcanzan los menores valores de J, ambos controladores trabajan cerca de los límites de sintonización explorados y presentan una reducción importante de los márgenes de estabilidad, especialmente del margen de retardo. Esto indica que las estrategias predictivas mejoran el desempeño nominal, pero a costa de una menor tolerancia frente a errores de modelado, particularmente en el tiempo de retardo.

Por tanto, para P_4, el MSP se posiciona como la mejor alternativa global bajo modelado ideal, aunque con una ventaja muy estrecha respecto al SP. Su principal fortaleza está en mejorar el rechazo de perturbaciones de entrada, que es el componente más influyente en el índice J para esta planta. El SP permanece como una opción prácticamente equivalente y más favorable frente a perturbaciones de salida. El PID, aunque más robusto en términos de márgenes, queda claramente por debajo en rendimiento. En consecuencia, para una dinámica próxima al integrador, el MSP representa la opción más equilibrada entre rendimiento nominal y capacidad de regulación, siempre que se reconozca que dicha ventaja depende de una sintonización agresiva y de una calidad adecuada del modelo del proceso.

Tabla 6-22. Resumen de resultados P_4

Bloque	Criterio	P4 – Próximo al integrador
Rendimiento	J nominal	MSP ≈ SP > PI > PID
	Seguimiento de referencia ($M_{ST} \approx 1.6$)	SP ≈ MSP > PI > PID
	Perturbación de salida ($M_{ST} \approx 1.6$)	SP > MSP > PI > PID
	Perturbación de entrada ($M_{ST} \approx 1.6$)	MSP > SP > PI > PID
Subtotal rendimiento		**MSP ≈ SP dominan. SP favorece perturbación de salida; MSP destaca en perturbación de entrada.**
Robustez	Margen de retardo, DM	PID > PI > MSP > SP
	Margen de fase, PM	PID > MSP > SP > PI
	Error de retardo +30% ($M_{ST} \approx 1.6$)	MSP > SP > PI > PID
Subtotal robustez		**PID presenta los mayores márgenes; MSP responde mejor ante el error de retardo evaluado.**
Lectura general		- MSP: alternativa preferente cuando se prioriza rendimiento, especialmente desde $M_{ST} \approx 1.5$; alcanza el menor J y destaca en perturbación de entrada. - SP: prácticamente equivalente al MSP en J y preferible si se prioriza perturbación de salida. - PID: preferible si la prioridad son los márgenes de robustez, pero presenta el peor rendimiento nominal. - PI: opción intermedia; supera al PID en rendimiento, pero queda por debajo de las estrategias predictivas.

6.6. Planta P_5: Integrador

La planta P_5 corresponde al proceso integrador $P_5(s) = e^{-s}/s$. representa un caso más exigente que las plantas estables P_1-P_4, porque incorpora un integrador puro. Esto implica que las perturbaciones, especialmente las aplicadas en la entrada del proceso, pueden acumularse en el tiempo si el controlador no las rechaza adecuadamente. Por ello, en esta planta el análisis de J debe interpretarse con especial atención al balance entre IAE_{di} e IAE_{do}. Como se observa en la tabla de pesos base, el peso asociado a la perturbación de entrada es considerablemente mayor que el de salida, lo que anticipa que el rechazo de d_i será un aspecto crítico en esta planta.

6.6.1. Pesos base resultantes

A diferencia de las plantas estables anteriores, esta planta no posee autorregulación, por lo que las perturbaciones de entrada tienen un efecto más severo sobre el desempeño. Esto se refleja directamente en los pesos base reportados en la **Tabla 6-23**, donde se obtiene $IAE_{do}^0 = 3.675$ e $IAE_{di}^0 = 32.331$. La diferencia entre ambos pesos indica que, para esta planta, el rechazo de perturbaciones de entrada representa una condición mucho más exigente.

Tabla 6-23. Pesos base para la normalización del índice J para P_5

Variable	Controlador	k_c	τ_i	τ_d	M_{ST}	IAE
IAE_{do}^0	PI	0.481	19.991	0	1.594	3.675
IAE_{di}^0	PI	0.481	19.391	0	1.595	32.331

6.6.2. Relación J, M_{ST} y parámetros de control óptimos

Siguiendo lo indicado en la **Figura 6-25**, en la región de M_{ST} igual a 1 a1.2, las soluciones no aparecen para todos los controladores. Esto no debe interpretarse como un valor nulo de desempeño, sino como ausencia de soluciones válidas dentro de los criterios de búsqueda. En los puntos disponibles, los valores de J son todavía altos, especialmente para PI y SP. Esto es coherente con una sintonía conservadora aplicada sobre una planta integradora: el controlador aún no tiene suficiente acción proporcional-integral para reducir de forma efectiva el error acumulado. En esta región, MSP ya muestra una ventaja inicial, con menor J que SP y PID.

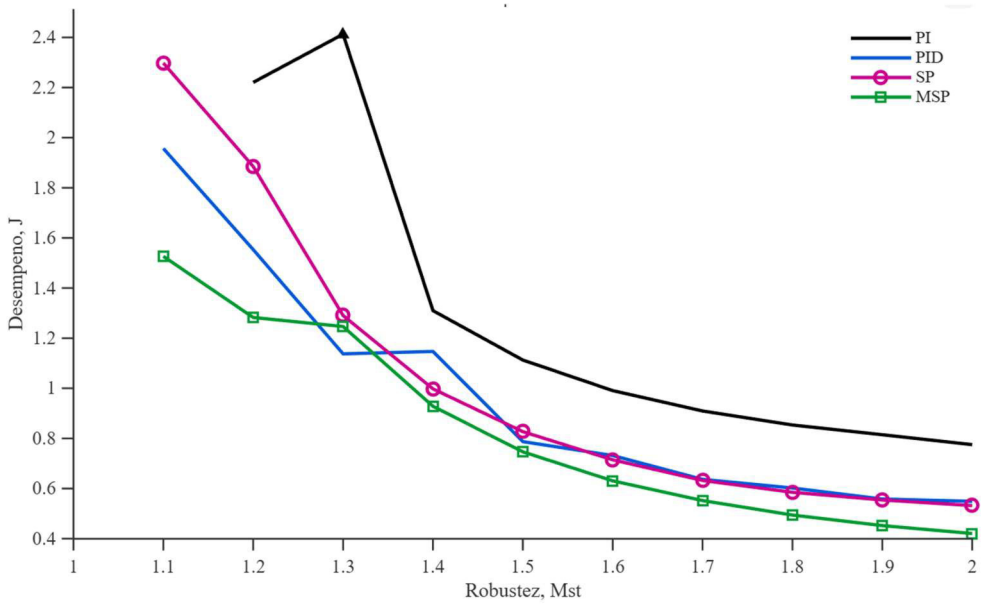

Figura 6-25. Curva de desempeño-robustez J vs. M_{ST} para P_5

Entre M_{ST} igual a 1.2 y 1.4, se observa una zona de transición. El PI presenta un comportamiento irregular, con un incremento de J alrededor de $M_{ST} = 1.3$, asociado a una combinación poco favorable de k_c bajo y τ_i alto. En una planta integradora, una acción integral demasiado lenta limita la capacidad de corregir errores acumulados, lo que penaliza directamente el IAE. En cambio, PID, SP y MSP reducen progresivamente J, apoyándose en una disminución de τ_i y un aumento gradual de k_c, ver **Figura 6-26.**

En la región de M_{ST} igual a 1.4 a 1.7, la separación entre controladores se vuelve más clara. MSP mantiene los menores valores de J, seguido de SP y PID. El PI queda rezagado porque conserva valores elevados de τ_i, alrededor de 18 a 19, lo que vuelve lenta la corrección del error. Por el contrario, MSP reduce τ_i de forma más marcada y aumenta k_c, generando una acción de control más efectiva frente a la acumulación propia del integrador.

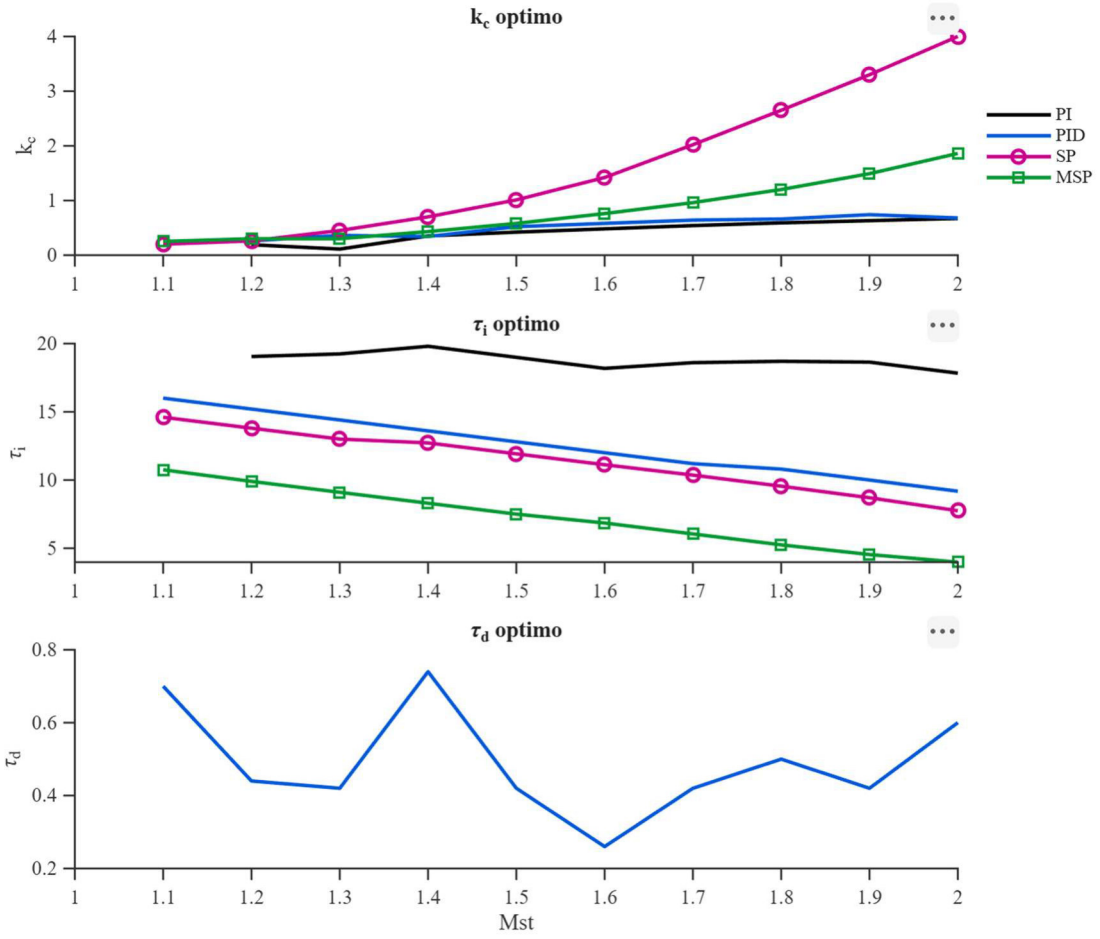

Figura 6-26. Parámetros de sintonía óptimos en función de M_{ST} para P_5

Para M_{ST} igual a 1.7 a 2, las curvas tienden a asentarse. MSP conserva la mejor posición y alcanza el menor valor global de J, MSP alcanza $J = 0.421$, mientras que SP obtiene $J = 0.532$, PID $J = 0.549$ y PI $J = 0.775$, ver **Tabla 6-24**. En este caso, SP reduce de manera muy efectiva el error ante perturbación de salida, mientras que MSP logra una reducción mucho más importante del error ante perturbación de entrada. Esto es decisivo en P_5, porque el canal de entrada es especialmente sensible por la presencia del integrador.

Tabla 6-24. Puntos de mínimo J por controlador para P_5

Planta	Tipo	Controlador	J	M_{ST}	IAE_{do}	IAE_{di}
P_5	Integrador	PI	0.775	1.992	3.200	21.966
P_5	Integrador	PID	0.549	2.009	2.538	13.166
P_5	Integrador	SP	0.532	1.871	0.454	30.410

Planta	Tipo	Controlador	J	M_{ST}	IAE_{do}	IAE_{di}
P_5	Integrador	MSP	0.421	2.005	2.516	5.058

Para mostrar el comportamiento de estos controladores, se simuló la respuesta controlada para un valor de $M_{ST} = 1.6$, considerando un modelado ideal de la planta. Como se observa en la **Figura 6-27** y se resume en la **Tabla 6-25**, los controladores PI/PID logran estabilizar la respuesta; sin embargo, presentan dificultades para eliminar completamente el offset, especialmente ante perturbaciones de entrada. Este comportamiento evidencia la dificultad de regular una planta integradora con retardo, ya que una sintonización más agresiva de la acción integral podría reducir el error estacionario, pero también comprometer los márgenes de estabilidad.

Por otro lado, el SP presenta un mejor comportamiento que el MSP ante cambios en la referencia y perturbaciones de salida, mientras que el MSP mejora el desempeño frente a perturbaciones de entrada. No obstante, frente a perturbaciones de entrada, tanto el SP como el MSP muestran dificultades importantes para reducir el offset, alcanzando incluso valores superiores a los obtenidos con los controladores convencionales. Esto sugiere que, para este escenario, las estructuras predictivas mejoran la respuesta en algunos casos, pero no eliminan completamente las limitaciones asociadas a la dinámica integradora. Además, como se muestra en la **Figura 6-26**, los parámetros de sintonización se encuentran próximos a sus límites de búsqueda, lo que indica que una mayor exigencia sobre el controlador podría conducir a regiones poco robustas o inestables.

Tabla 6-25. Indicadores de rendimiento de las respuestas controladas para P_5 ($M_{ST} = 1.6$)

Controlador	Tiempo de subida	Pico absoluto	Tiempo de estabilización	Sobrepaso	Offset	IAE
Referencia						
PI	3.10	-	20.30	1.109E+01	-2.447E-02	3.74
PID	2.42	-	19.84	1.257E+01	-1.166E-02	3.18
SP	2.63	-	3.76	2.220E-14	5.773E-15	1.70
MSP	4.03	-	6.15	0.000E+00	2.603E-10	2.31
Perturbación de entrada						
PI	-	2.06	*	-	4.158E-01	31.18
PID	-	1.65	*	-	1.301E-01	19.36
SP	-	1.70	*	-	1.704E+00	47.73
MSP	-	1.72	*	-	1.314E+00	39.30
Perturbación de salida						
PI	-	1.01	*	-	2.447E-02	3.74
PID	-	1.05	24.06	-	1.166E-02	3.18

Controlador	Tiempo de subida	Pico absoluto	Tiempo de estabilización	Sobrepaso	Offset	IAE
SP	-	1.04	3.73	-	7.328E-15	1.70
MSP	-	1.03	9.88	-	5.205E-06	2.14

(*) Nota: Valores fueron omitidos por el programa al no encontrar el punto de estabilización en el tiempo de simulación.

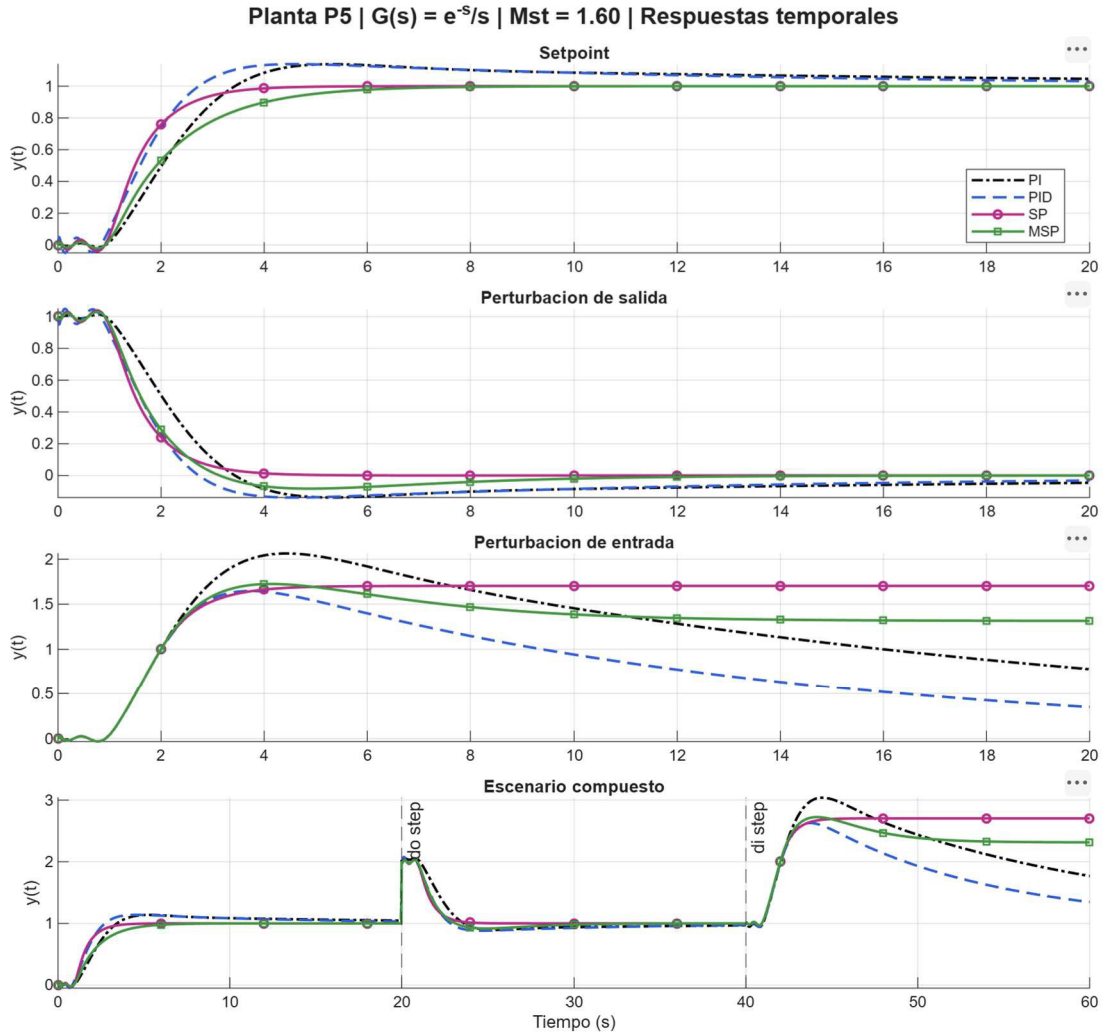

Figura 6-27. Respuestas controladas ante cambio de referencia y perturbaciones de entrada y salida para P_5 ($M_{ST} = 1.6$)

6.6.3. Interpretación estructural

La ventaja de MSP en P_5 tiene una justificación más clara que en las plantas puramente estables de bajo orden. En una planta integradora, una perturbación de entrada pasa por el término $1/s$, por lo que su efecto puede acumularse si no existe una compensación suficientemente efectiva. El SP mejora la respuesta al compensar el retardo nominal, pero

no modifica de forma tan favorable el efecto acumulativo del integrador sobre perturbación de entrada.

$$H_{di}^{MSP}(s) = \frac{e^{-s}}{s}\left[\frac{1 + C(s)\frac{1}{s}\left(1 + F_p(s)e^{-s}\right)}{1 + C(s)\frac{1}{s}}\right]$$

El MSP, en cambio, introduce el filtro $F_{p(s)}$, que en plantas integradoras tiene mayor sentido estructural porque permite modificar la forma en que el sistema rechaza la perturbación antes de que esta se acumule en la salida. Por eso, aunque SP obtiene el menor IAE_{do}, MSP logra un IAE_{di} mucho menor. Según la **Tabla 6-24**, SP alcanza $IAE_{do} = 0.454$, pero mantiene $IAE_{di} = 30.4$; en cambio, MSP obtiene $IAE_{do} = 2.516$, pero reduce IAE_{di} hasta 5.058. Debido a este balance, MSP obtiene el menor J global.

El PID también mejora claramente respecto al PI, lo que indica que la acción derivativa sí aporta cierto beneficio dinámico en esta planta. Sin embargo, su efecto no es suficiente para superar a las estrategias predictivas, porque el problema dominante no es solo anticipar cambios, sino rechazar una perturbación que se acumula por la acción integradora del proceso.

6.6.4. Márgenes de retardo y fase

El análisis del margen de retardo confirma que, para P_5, la mejora de desempeño obtenida por MSP no se alcanza a costa de una pérdida inmediata de robustez frente al retardo. Como se observa en la **Figura 6-28**, MSP mantiene un margen de retardo sostenido al menos hasta $M_{ST} = 1.6$. En esta región, su DM es mayor que el de SP en prácticamente todo el intervalo y resulta comparable con el obtenido por PID. Sin embargo, a diferencia del PID, MSP logra simultáneamente menores valores de J. Esto indica que, para la planta integradora, el filtro MSP permite mejorar el rendimiento sin deteriorar de forma abrupta la tolerancia al retardo.

A partir de $M_{ST} > 1.6$, el DM de MSP comienza a disminuir con mayor claridad, lo cual es coherente con una sintonía más exigente y con el aumento progresivo de k_c. No obstante, incluso en esta región, MSP conserva una ventaja de desempeño frente a PID y PI, por lo que el compromiso entre rendimiento y robustez sigue siendo favorable. En cambio, SP presenta una caída más marcada del margen de retardo en todo el intervalo,

especialmente hacia la zona de mayor M_{ST}, lo que sugiere que su mejora de desempeño se obtiene con una menor tolerancia frente a errores o incrementos adicionales del retardo.

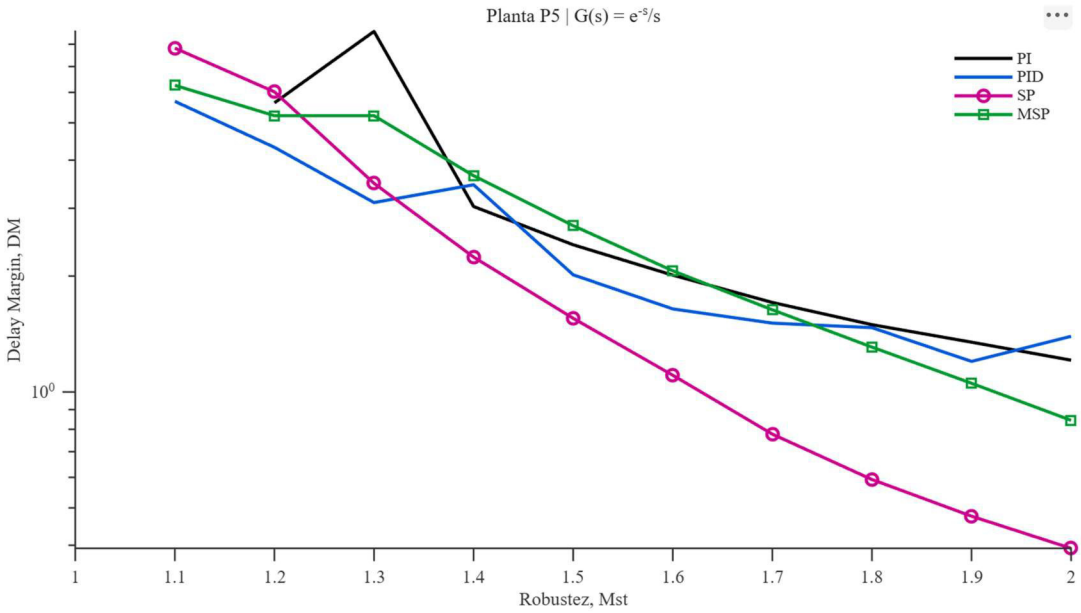

Figura 6-28. Margen de retardo en función de M_{ST} para P_5

El margen de fase complementa esta interpretación, como se muestra en la **Figura 6-29**. MSP mantiene valores cercanos a 90°, lo que indica una reserva de fase elevada. Esto significa que la estrategia MSP conserva una distancia angular amplia respecto a la condición crítica de inestabilidad, aun cuando el DM disminuye en la región de mayor M_{ST}. Esta aparente diferencia entre PM y DM no es contradictoria: el margen de retardo depende de la relación entre el margen de fase y la frecuencia de cruce. Por tanto, una sintonía más rápida puede conservar un buen PM, pero reducir el DM si desplaza la frecuencia de cruce hacia valores mayores.

En conjunto, para P5, MSP ofrece el compromiso más atractivo: mejora el rendimiento global, principalmente por la reducción del IAE_{di}, y mantiene un margen de retardo competitivo hasta $M_{ST} = 1.6$. Esto refuerza la conveniencia de MSP para plantas integradoras, donde el rechazo de perturbaciones de entrada es crítico y donde una pérdida temprana de robustez frente al retardo sería especialmente perjudicial.

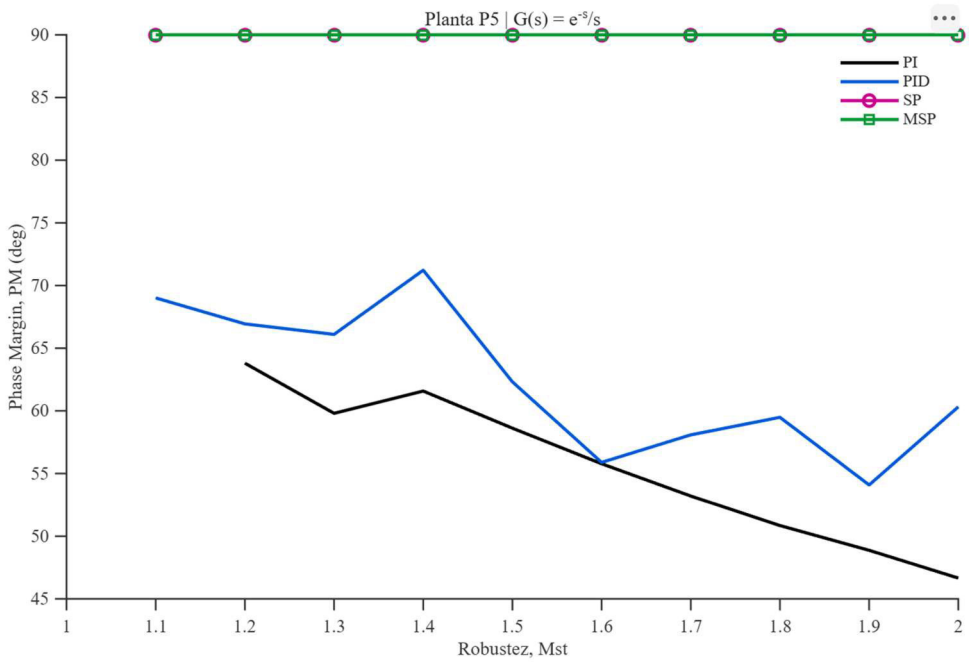

Figura 6-29. Margen de fase en función de M_{ST} para P5

6.6.5. Simulación del efecto de errores de modelado del tiempo de retardo en la respuesta controlada

Para evaluar el comportamiento de los controladores frente a errores en el modelado del tiempo de retardo, se simuló la respuesta controlada para un valor de $M_{ST} = 1.6$. La **Figura 6-30**, muestra las respuestas obtenidas ante un error de modelado del tiempo de retardo de +30%, correspondiente al escenario más desfavorable dentro de los casos evaluados (Escenario de -30% en Anexos).

En términos generales, las respuestas controladas presentan intensificación en los puntos máximos y mínimos, lo cual es consistente con la tendencia observada en la curva DM vs. M_{ST}. MSP, seguido de SP, presenta una mejor respuesta respecto a los controladores convencionales, excepto por el control de la perturbación de entrada, donde no pueden llegar a reducir el offset completamente.

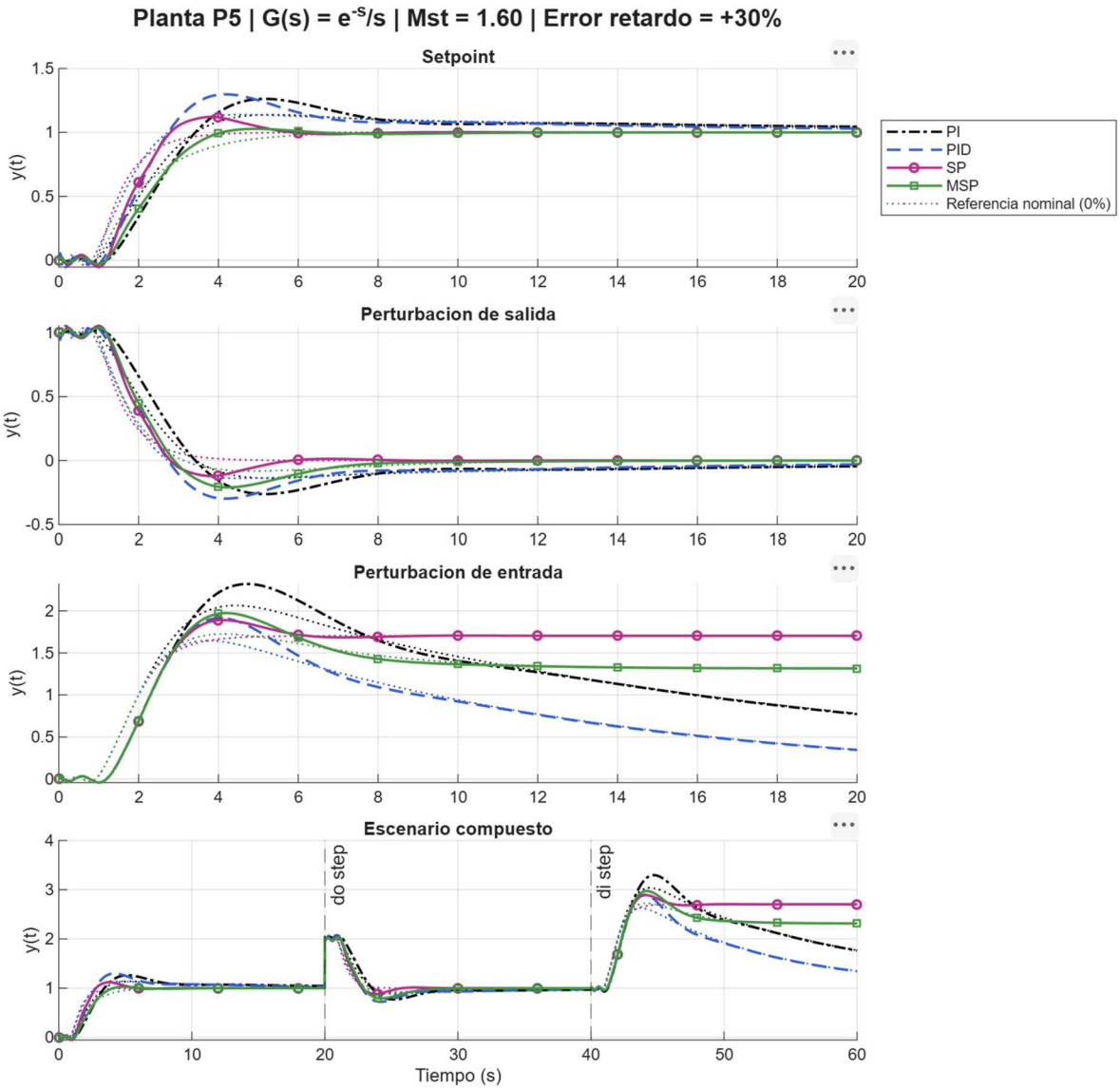

Figura 6-30. Respuestas controladas ante un error de modelado del tiempo de retardo de +30% para P5 ($M_{ST} = 1.6$)

6.6.6. Síntesis de resultados para P5

En síntesis, para la planta P5, correspondiente a un proceso integrador con retardo, el MSP presenta la mejor posición global entre las estrategias evaluadas. A diferencia de las plantas estables, en este caso la perturbación de entrada representa el escenario más exigente, debido a que su efecto puede acumularse por la presencia del integrador. Por ello, el índice J queda fuertemente condicionado por la capacidad del controlador para reducir IAE_{di}.

Bajo condiciones de modelado ideal y considerando los mínimos globales obtenidos en el barrido de M_{ST}, el MSP alcanza el menor valor de J, superando al SP, al PID y al PI.

Esta ventaja se explica principalmente por su capacidad para reducir de forma significativa el error acumulado ante perturbaciones de entrada. Mientras que el SP obtiene el menor IAE_{do}, su desempeño frente a perturbaciones de entrada es considerablemente más limitado. En cambio, el MSP sacrifica parte del rechazo de perturbaciones de salida, pero mejora de manera decisiva el canal de entrada, lo que resulta más relevante para una planta integradora.

La comparación con los controladores convencionales también favorece al MSP. El PID mejora claramente al PI, lo que indica que la acción derivativa aporta cierto beneficio dinámico; sin embargo, no logra compensar adecuadamente el efecto acumulativo de la planta. Los controladores PI/PID estabilizan la respuesta, pero presentan dificultades para eliminar completamente el offset, especialmente ante perturbaciones de entrada. Por tanto, en P_5, la ventaja del MSP no proviene de una respuesta más rápida en todos los escenarios, sino de una mejor distribución del desempeño frente a la dinámica integradora, particularmente en el rechazo de perturbaciones de entrada

El análisis de robustez refuerza esta conclusión. Hasta $M_{ST} = 1.6$, el MSP mantiene un margen de retardo competitivo, superior al del SP y comparable al del PID, pero con un mejor desempeño global. Esto indica que, para la planta integradora, el filtro predictivo permite mejorar el rendimiento sin generar una pérdida temprana de robustez frente al retardo. Aunque para valores más altos de M_{ST} el margen de retardo del MSP disminuye, su compromiso entre rendimiento y robustez sigue siendo más favorable que el de las demás estrategias.

Por tanto, para P_5, el MSP se posiciona como la alternativa más conveniente. Frente al SP, ofrece una mejora clara en el rechazo de perturbaciones de entrada, que es el aspecto crítico en una planta integradora. Frente al PID, logra un mejor desempeño global sin perder competitividad en robustez para la región de evaluación considerada. No obstante, debe reconocerse que ninguna estrategia elimina completamente las dificultades asociadas al offset ante perturbaciones de entrada, por lo que la ventaja del MSP debe interpretarse como una mejora sustancial del compromiso rendimiento-robustez, más que como una solución completa a las limitaciones propias de la dinámica integradora.

Tabla 6-26. Resumen de resultados P_5

Bloque	Criterio	P5 – Integrador
Rendimiento	J nominal	MSP > SP > PID > PI
	Seguimiento de referencia ($M_{ST} \approx 1.6$)	SP > MSP > PID > PI
	Perturbación de salida ($M_{ST} \approx 1.6$)	SP > MSP > PID > PI
	Perturbación de entrada ($M_{ST} \approx 1.6$)	PID > PI > MSP > SP
Subtotal rendimiento		**MSP alcanza el menor J global, principalmente por mejorar el canal de entrada al aumentar M_{ST}; a $M_{ST} \approx 1.6$, SP domina referencia/salida y PID la perturbación de entrada.**
Robustez	Margen de retardo, DM ($M_{ST} \approx 1.6$)	PI ≈ MSP > PID > SP
	Margen de fase, PM	MSP ≈ SP > PID ≈ PI
	Error de retardo +30% ($M_{ST} \approx 1.6$)	MSP ≈ SP en referencia/salida; PID > PI > MSP > SP en perturbación de entrada
Subtotal robustez		**MSP mantiene PM alto y DM competitivo, claramente superior al SP; la perturbación de entrada sigue siendo el escenario limitante.**
Lectura general		- MSP: alternativa preferente para P_5 cuando se considera el desempeño global; desde $M_{ST} \approx 1.4$ presenta la mejor tendencia de J y mantiene robustez competitiva. - SP: preferible si se prioriza seguimiento o perturbación de salida alrededor de $M_{ST} \approx 1.6$, pero pierde competitividad por su menor DM y peor rechazo de perturbación de entrada. - PID: preferible si se prioriza específicamente la perturbación de entrada alrededor de $M_{ST} \approx 1.6$; su J global queda por encima de MSP y SP. - PI: conserva buen DM, pero presenta el peor J global; no resulta competitivo si se prioriza rendimiento.

6.7. Planta P_6: Dinámica balanceada

Para P_6, correspondiente al proceso inestable $P_6(s) = e^{-s}/s - 1$, no se obtuvo una familia consistente de soluciones válidas y comparables. Esta planta combina un polo inestable con retardo, por lo que la sintonía debe estabilizar una dinámica naturalmente divergente y, simultáneamente, cumplir restricciones de robustez y desempeño.

Como se reporta en los resultados del anexo, PI y PID no presentaron candidatos aceptables dentro de los límites de búsqueda definidos. En las estructuras predictivas aparecieron algunos puntos aislados, pero asociados a valores extraordinariamente altos de IAE_{di}, por lo que no pueden considerarse soluciones útiles, aunque numéricamente se aproximen a un valor de M_{ST}. Esto confirma que alcanzar una condición de robustez no es suficiente si el rechazo de perturbaciones queda dominado por errores integrales excesivos.

Este resultado no implica que P_6 sea imposible de estabilizar, sino que, bajo la discretización, los rangos de k_c, τ_i, τ_d y los criterios de aceptación empleados en esta investigación, no se encontró una región factible robusta y comparable. Para obtener soluciones podría requerirse ampliar los límites de búsqueda, usar optimización continua, explorar otros rangos de M_{ST}, o incorporar una etapa de preestabilización. Sin embargo, aplicar ajustes particulares solo para P_6 rompería la consistencia metodológica usada en las demás plantas.

En consecuencia, P6 se reporta como un caso extraordinario y no se selecciona un valor de M_{ST} para el análisis posterior en el dominio del tiempo. Su resultado delimita el alcance práctico de la metodología propuesta frente a procesos inestables con retardo y dinámica rápida.

6.8. Planta P_7: Inestable con dinámica dominante

La planta P_7, que corresponde al proceso inestable $P_7(s) = e^{-s}/8s - 1$, representa un caso inestable con una dinámica menos rápida que P_6. Esto permite encontrar soluciones válidas para varios controladores, aunque no en todo el intervalo de M_{ST}. Por ello, los espacios en blanco o tramos ausentes en las curvas no deben interpretarse como valores nulos, sino como puntos donde la búsqueda no encontró una sintonía aceptable bajo los criterios de estabilidad, robustez y desempeño definidos.

6.8.1. Pesos base resultantes

Como se muestra en la **Tabla 6-27**, los pesos base para P_7 se obtuvieron alrededor de $M_{ST} = 1.630$, con $IAE_{do}^{0} = 9.119$ e $IAE_{di}^{0} = 5.236$. Estos valores indican que la referencia de normalización no penaliza de la misma forma las perturbaciones de salida y de entrada. Por ello, en esta planta resulta especialmente importante analizar IAE_{do}, IAE_{di} y J de forma conjunta.

Tabla 6-27. Pesos base para la normalización del índice J para P_7

Variable	Controlador	k_c	ti	t_d	M_{ST}	IAE
IAE_{do}^{0}	PI	3.451	19.851	0	1.630	9.119
IAE_{di}^{0}	PI	3.401	19.201	0	1.630	5.236

6.8.2. Relación J, M_{ST} y parámetros de control óptimos

Según la **Figura 6-31**, en la región de baja robustez, aproximadamente M_{ST} igual a 1 a 1.3, la factibilidad es limitada. PI no presenta soluciones y PID aparece recién alrededor de $M_{ST} = 1.3$. Las estrategias predictivas sí muestran algunos puntos, pero SP presenta valores de J elevados, del orden de 3.5, debido principalmente al mal rechazo de perturbaciones de entrada. Esto indica que, aunque el predictor puede mejorar ciertos aspectos de la respuesta nominal, no logra controlar adecuadamente el efecto de la dinámica inestable cuando la perturbación ingresa por la planta.

Entre M_{ST} igual a 1.3 y 1.7, las curvas de PI, PID y MSP se ubican en una zona cercana, con valores de J menores que los de SP. En esta región, PID comienza a mostrar la mejor tendencia, porque combina una ganancia moderada con una acción derivativa pequeña pero útil. La derivada no busca cancelar directamente la dinámica dominante, sino aportar adelanto de fase y mejorar la capacidad de estabilizar una planta inestable con retardo.

MSP también se mantiene competitivo, porque el filtro ayuda a reducir el efecto de la perturbación de entrada respecto a SP.

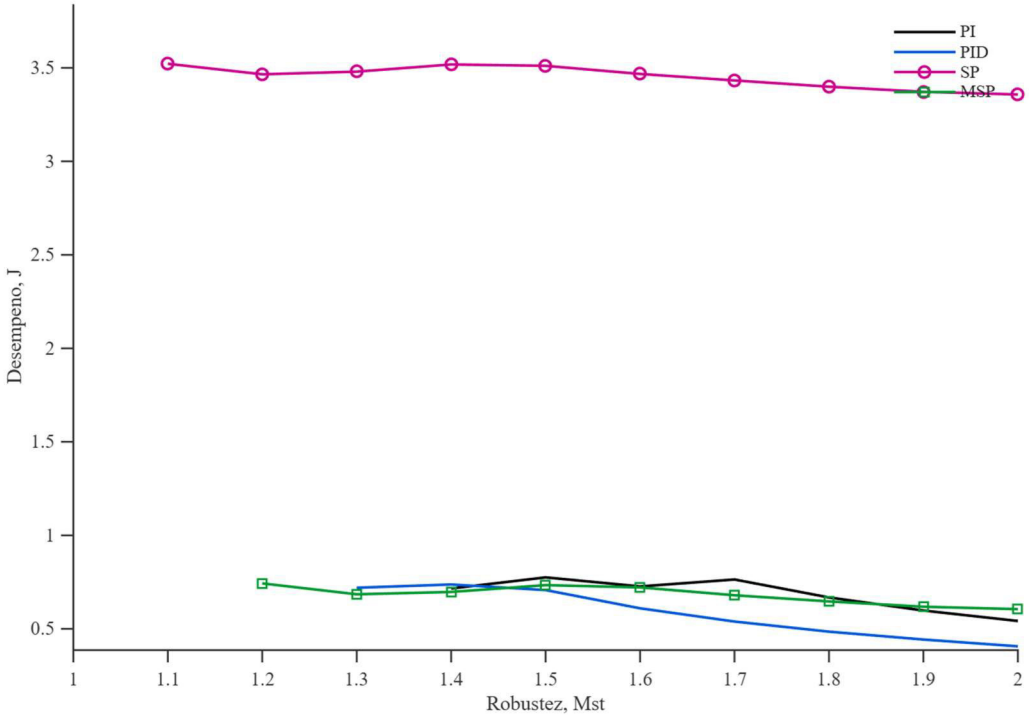

Figura 6-31. Curva de desempeño-robustez J vs. M_{ST} para P_7

En la región alta, M_{ST} igual a 1.7 a 2.0, PID se consolida como la mejor alternativa en desempeño global, alcanzando $J = 0.406$, ver **Tabla 6-28**. PI también obtiene un resultado aceptable, con $J = 0.542$, mientras que MSP queda ligeramente por encima, con $J = 0.605$. SP, en cambio, mantiene el peor desempeño global, pese a reducir IAE_{do}, porque su IAE_{di} permanece excesivamente alto. Este resultado muestra que, para P_7, mejorar solo el canal de perturbación de salida no es suficiente: la dinámica inestable amplifica fuertemente el efecto de la perturbación de entrada.

Como se muestra en la **Figura 6-32**, desde los parámetros óptimos se observa que PI y PID trabajan con k_c más moderados y τ_i altos, lo que refleja una sintonía más prudente frente a la inestabilidad. En cambio, SP y MSP requieren mayores valores de k_c y menores valores de τ_i, especialmente en la zona alta de M_{ST}. Esta diferencia indica que las estructuras predictivas intentan compensar la dificultad del proceso mediante una acción más intensa, pero en SP esa agresividad no se traduce en buen desempeño integral debido al canal de perturbación de entrada.

Tabla 6-28. Puntos de mínimo J por controlador para P_7

Planta	Tipo	Controlador	J	M_{ST}	IAE_{do}	IAE_{di}
P_7	Inestable	PI	0.542	2.003	5.864	2.309
P_7	Inestable	PID	0.406	2.007	4.354	1.757
P_7	Inestable	SP	3.357	2.002	1.307	34.408
P_7	Inestable	MSP	0.605	1.901	5.704	3.063

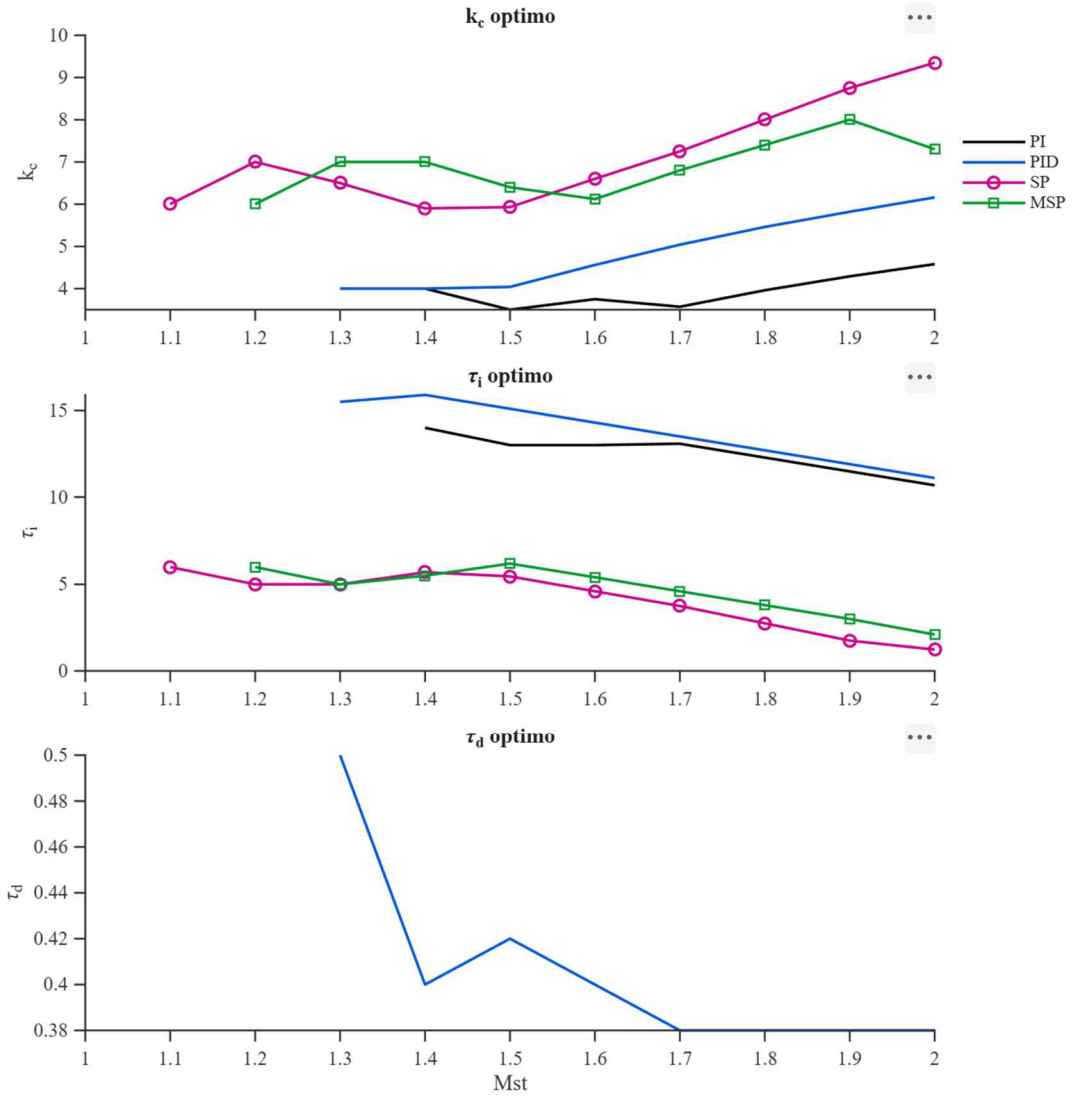

Figura 6-32. Parámetros de sintonía óptimos en función de M_{ST} para P_7

Para mostrar el comportamiento de estos controladores, se simuló la respuesta controlada para un valor de $M_{ST} = 1.6$, considerando un modelado ideal de la planta. Como se observa en la **Figura 6-33** y la **Tabla 6-29**, todos los controladores se ven afectados por

la dinámica más lenta de la planta, haciendo que el tiempo en conseguir el estado estable sea mayor y provocando un offset remanente.

Respecto al control de cambios en la referencia y perturbación de salida, el SP y MSP presentan sobrepasos y picos absolutos menores, llegando al valor deseado mucho más rápido que los controladores convencionales. Sin embargo, respecto a la perturbación de entrada, el SP claramente es incapaz de presentar un control efectivo y es el MSP quien presente el comportamiento sobresaliente.

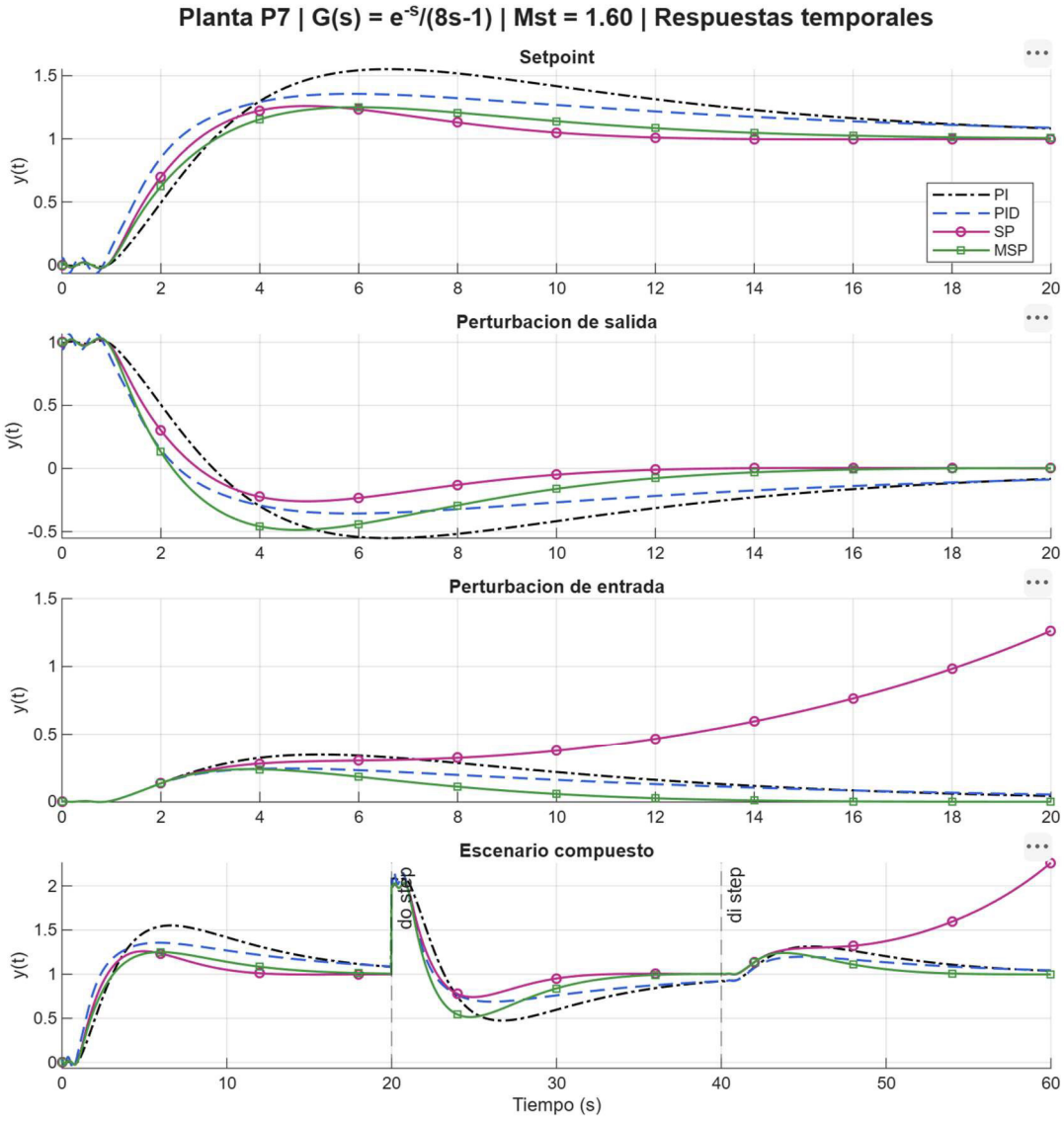

Figura 6-33. Respuestas controladas ante cambio de referencia y perturbaciones de entrada y salida para P_7 ($M_{ST} = 1.6$)

Tabla 6-29. Indicadores de rendimiento de las respuestas controladas para P_7 ($M_{ST} = 1.6$)

Controlador	Tiempo de subida	Pico absoluto	Tiempo de estabilización	Sobrepaso	Offset	IAE
Referencia						
PI	2.86	-	24.95	5.299E+01	-1.419E-02	7.62
PID	2.17	-	25.18	3.197E+01	-2.776E-02	5.90
SP	2.47	-	11.30	2.596E+01	-2.309E-05	3.01
MSP	2.74	-	16.62	2.503E+01	3.490E-05	3.67
Perturbación de entrada						
PI	-	0.35	-	-	7.028E-03	3.63
PID	-	0.24	-	-	1.622E-02	3.00
SP	-	4.41	-	-	4.409E+00	34.79
MSP	-	0.24	14.62	-	2.786E-04	1.49
Perturbación de salida						
PI	-	1.01	27.98	-	1.419E-02	7.62
PID	-	1.07	-	-	2.776E-02	5.90
SP	-	1.03	11.27	-	2.309E-05	3.01
MSP	-	1.03	14.69	-	6.974E-04	4.58

6.8.3. Interpretación estructural

La diferencia principal entre SP y MSP se entiende a partir del tratamiento de las dinámicas no deseadas. En una planta inestable, el predictor de Smith clásico puede mejorar el comportamiento asociado al retardo nominal, pero no necesariamente atenúa de forma suficiente el efecto del polo inestable para una perturbación de entrada. Por eso, SP obtiene un IAE_{do} bajo, pero un IAE_{di} muy alto.

El MSP introduce el filtro $F_{p(s)}$, cuyo objetivo es modificar la forma en que las dinámicas críticas de la planta aparecen en las funciones de transferencia de perturbación. En P_7, esto sí genera una mejora clara frente a SP en el canal de entrada: MSP reduce el IAE_{di} de 34.408 a 3.063. Sin embargo, esa mejora no es suficiente para superar al PID, porque el filtro también modifica la dinámica efectiva del sistema y no reduce simultáneamente IAE_{do} con la misma eficacia. En consecuencia, MSP corrige una debilidad importante del SP, pero no alcanza el mejor compromiso global.

6.8.4. Márgenes de estabilidad y robustez

Como muestra la **Figura 6-34**, los márgenes de retardo muestran que P7 es una planta exigente: los valores de DM son mucho menores que en las plantas estables y tienden a disminuir conforme aumenta M_{ST}. En la región intermedia, PID presenta los mejores márgenes de retardo, especialmente entre M_{ST} igual a 1.3 y 1.5. MSP mantiene valores comparables con PI y superiores a SP en parte del rango alto, pero pierde margen conforme la sintonía se vuelve más agresiva.

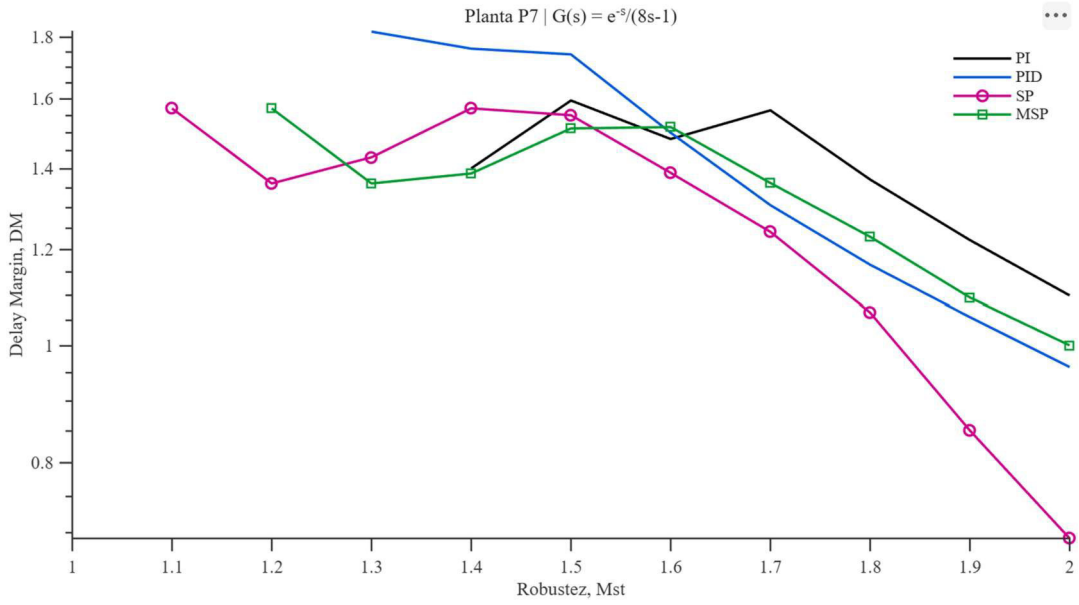

Figura 6-34. Margen de retardo en función de M_{ST} para P_7

El margen de fase muestra una lectura complementaria, ver Figura 6-35. SP y MSP presentan mayores valores de PM que PI y PID, lo cual indica mayor reserva de fase nominal. Sin embargo, esto no implica automáticamente mejor desempeño ni mayor margen de retardo, porque DM también depende de la frecuencia de cruce. En P7, las estructuras predictivas pueden conservar mayor margen de fase, pero si la sintonía desplaza la frecuencia de cruce o amplifica el canal de perturbación de entrada, el desempeño global puede deteriorarse. Por eso SP tiene buen PM, pero mal J.

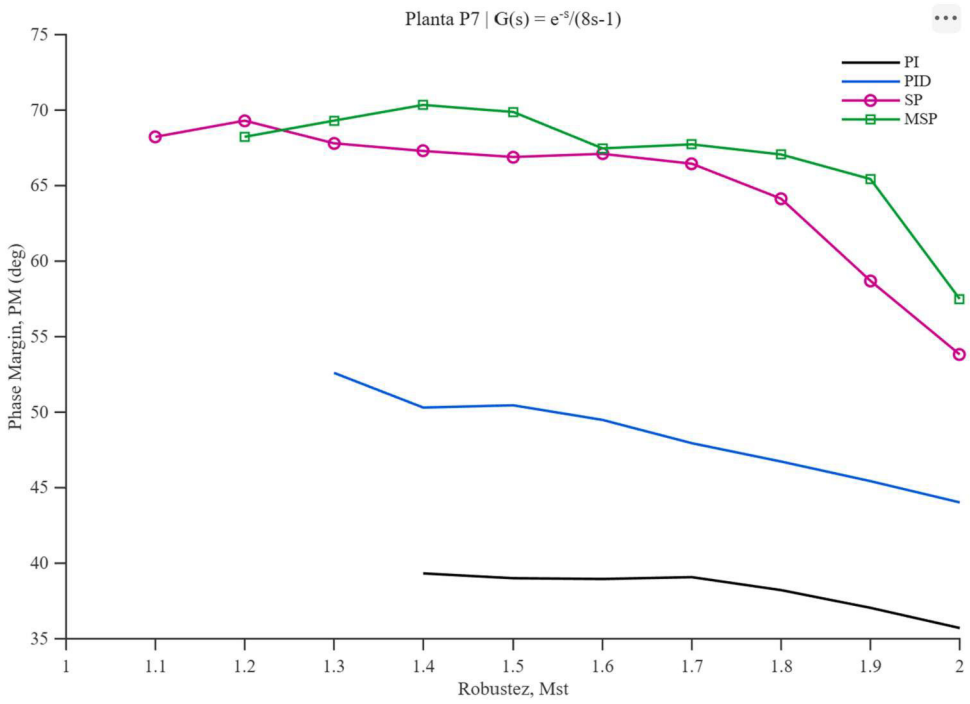

Figura 6-35. Margen de fase en función de M_{ST} para P_7

6.8.5. Simulación del efecto de errores de modelado del tiempo de retardo en la respuesta controlada

Para evaluar el comportamiento de los controladores frente a errores en el modelado del tiempo de retardo, se simuló la respuesta controlada para un valor de $M_{ST} = 1.6$. La **Figura 6-36**, muestra las respuestas obtenidas ante un error de modelado del tiempo de retardo de +30%, correspondiente al escenario más desfavorable dentro de los casos evaluados (Escenario de -30% en Anexos).

En términos generales, las respuestas controladas presentan intensificación en los puntos máximos y mínimos, lo cual es consistente con la tendencia observada en la curva DM vs. M_{ST}. MSP presenta el comportamiento más equilibrado frente al error de retardo evaluado. El SP mantiene un buen comportamiento ante cambios de referencia y perturbaciones de salida, pero no consigue una respuesta aceptable frente a perturbaciones de entrada.

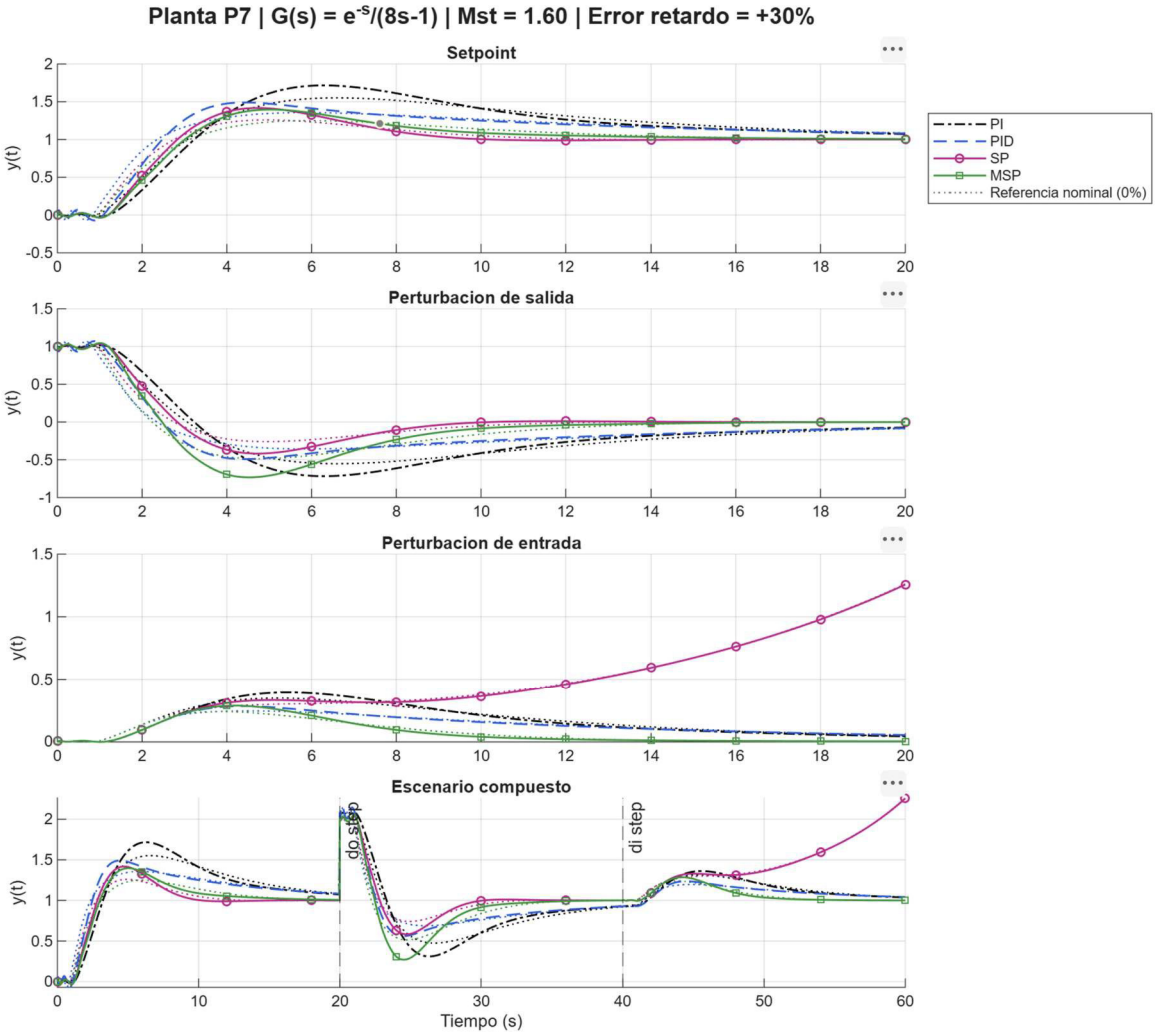

Figura 6-36. Respuestas controladas ante un error de modelado del tiempo de retardo de +30% para P_7 ($M_{ST} = 1.6$)

6.8.6. Síntesis de resultados para P_7

En síntesis, para la planta P_7, correspondiente a un proceso inestable con retardo dominante, el PID se posiciona como la alternativa con mejor desempeño global. A diferencia de P_6, la dinámica inestable de P_7 es más lenta, lo que permite encontrar soluciones válidas para varios controladores; sin embargo, la presencia del polo inestable sigue imponiendo restricciones importantes sobre la sintonización. En este contexto, el índice J debe interpretarse considerando simultáneamente IAE_{do} e IAE_{di}, ya que una mejora aislada en uno de los canales no garantiza un mejor desempeño global.

Bajo condiciones de modelado ideal, el PID alcanza el menor valor de J, seguido por el PI y el MSP. Este resultado indica que, para P_7, una acción derivativa moderada permite mejorar la estabilización de la planta inestable sin recurrir a sintonías tan agresivas como

las requeridas por las estructuras predictivas. El PI también presenta un desempeño aceptable, aunque inferior al PID, debido a que mantiene una sintonía más prudente frente a la inestabilidad. En cambio, el SP obtiene el peor desempeño global, pese a lograr un bajo IAE_{do}, porque su rechazo de perturbaciones de entrada resulta deficiente.

La comparación entre SP y MSP es especialmente relevante en esta planta. El SP mejora el rechazo de perturbaciones de salida, pero no controla adecuadamente el efecto de la perturbación de entrada sobre la dinámica inestable, lo que genera valores elevados de IAE_{di}. El MSP corrige en gran medida esta debilidad mediante el filtro $F_p(s)$, reduciendo de forma significativa el error acumulado ante perturbaciones de entrada. No obstante, esta mejora se obtiene sacrificando parte del desempeño frente a perturbaciones de salida, por lo que el MSP no alcanza el mejor valor global de J.

Desde el punto de vista de robustez, P_7 confirma que los márgenes deben analizarse junto con el desempeño. Aunque SP y MSP pueden presentar mayores márgenes de fase que PI y PID, esto no se traduce necesariamente en mejor robustez efectiva ni en menor J. En particular, el SP muestra que una buena reserva de fase nominal no es suficiente si el controlador amplifica el canal de perturbación de entrada. El PID, en cambio, mantiene una relación más favorable entre desempeño, margen de retardo y estabilidad relativa.

Por tanto, para P_7, el PID es la estrategia más conveniente, al ofrecer el mejor compromiso entre estabilización, rechazo de perturbaciones y desempeño global. El MSP mejora ampliamente al SP, especialmente frente a perturbaciones de entrada, pero mantiene un desempeño global inferior al PID. En consecuencia, para procesos inestables con dinámica dominante como P_7, el MSP resulta útil como mejora estructural frente al Predictor de Smith clásico; sin embargo, no constituye la alternativa dominante cuando se compara con un PID adecuadamente sintonizado.

Tabla 6-30. Resumen de resultados P_7

Bloque	Criterio	P7 – Inestable con dinámica dominante
Rendimiento	J nominal	PID > PI > MSP >>> SP
	Seguimiento de referencia ($M_{ST} \approx 1.6$)	SP > MSP > PID > PI
	Perturbación de salida ($M_{ST} \approx 1.6$)	SP > MSP > PID > PI
	Perturbación de entrada ($M_{ST} \approx 1.6$)	MSP > PID > PI >>> SP
Subtotal rendimiento		**PID domina en J global. SP destaca en referencia/salida, pero falla en entrada; MSP**

Bloque	Criterio	P7 – Inestable con dinámica dominante
		corrige esta debilidad y logra el mejor rechazo de perturbación de entrada.
Robustez	Margen de retardo, DM ($M_{ST} \approx 1.6$)	MSP > PID ≈ PI > SP
	Margen de fase, PM	SP ≈ MSP > PI ≈ PID
	Error de retardo +30% ($M_{ST} \approx 1.6$)	MSP > PID ≈ PI > SP
Subtotal robustez		**MSP presenta el comportamiento más equilibrado ante el error de retardo; SP pierde robustez efectiva por su deficiente control de la perturbación de entrada.**
Lectura general		- PID: alternativa preferente global, especialmente hacia $M_{ST} \approx 1.7–2.0$; alcanza el menor J y mantiene DM competitivo. - MSP: preferible cuando la perturbación de entrada es crítica; a $M_{ST} \approx 1.6$ presenta el mejor rechazo de este canal y mejora ampliamente al SP, aunque no supera al PID en J global. - SP: conveniente solo si se prioriza referencia o perturbación de salida; no resulta favorable cuando la perturbación de entrada es relevante. - PI: segunda mejor alternativa según J, pero sin una ventaja específica frente al PID.

7. CONCLUSIONES Y RECOMENDACIONES

7.1. Conclusiones

1. Se construyó una estrategia de control basada en el Predictor de Smith Modificado para procesos con retardo, integradores e inestables. La incorporación del filtro $F_p(s)$ modificó la respuesta del Predictor de Smith clásico; sin embargo, su aporte quedó condicionado por la dinámica de la planta, el tipo de perturbación dominante y la incertidumbre asociada al tiempo de retardo.

2. Se propuso una metodología de evaluación apoyada en un programa computacional codificado para comparar PI, PID, SP y MSP bajo un mismo procedimiento. La metodología integró la búsqueda de sintonías, la construcción de curvas J vs. M_{ST}, el cálculo de márgenes de estabilidad, la evaluación temporal a robustez fija y la prueba frente a errores en el retardo. De esta forma, el aporte principal de la investigación se concentra en el procedimiento comparativo desarrollado, el cual permitió evaluar no solo el desempeño nominal, sino también la consistencia de cada estrategia ante incertidumbre del modelo.

3. Respecto al rendimiento ante perturbaciones, el Predictor de Smith Modificado no mostró una superioridad general frente al controlador PID ni frente al Predictor de Smith clásico. Su principal ventaja se observó en el proceso integrador P_5, donde alcanzó el mejor desempeño global al mejorar de forma significativa el rechazo de perturbaciones de entrada. En P_4 también presentó ventajas cuando la perturbación de entrada dominó la respuesta. En cambio, para P1 el SP mantuvo el mejor desempeño nominal, mientras que en P_2, P_3 y P_7 el PID conservó un comportamiento competitivo o superior. Para P_6 no se obtuvo una familia consistente de soluciones válidas. Por tanto, el MSP mejora el desempeño principalmente en procesos integradores, donde el efecto de las perturbaciones de entrada puede acumularse, y en condiciones donde este canal limita el desempeño de las estrategias convencionales o del SP clásico.

4. Respecto a la robustez, el Predictor de Smith Modificado presentó beneficios cuando la incertidumbre en el tiempo de retardo afectó significativamente la respuesta del Predictor de Smith clásico. Esta mejora fue más relevante en P_1, donde el MSP mostró una respuesta más amortiguada ante errores de retardo, y

en P_5, donde mantuvo un comportamiento favorable dentro de la región intermedia de robustez. Sin embargo, en plantas estables simples o con dinámica dominante, el PID conservó márgenes y respuestas suficientemente competitivas, por lo que la mayor complejidad del MSP no siempre se justificó. En plantas inestables, los resultados no permiten establecer una recomendación general, debido a la ausencia de soluciones consistentes en P_6 y a que, en P_7, el MSP mejoró ampliamente al SP frente a perturbaciones de entrada, pero no superó el desempeño global del PID.

5. El SP se justifica en procesos dominados por retardo y con baja incertidumbre del modelo, como P_1 ($\phi = 0$). El MSP se justifica cuando la incertidumbre del retardo o la perturbación de entrada condicionan la respuesta, principalmente en plantas próximas al integrador, como P_4 ($\phi = 20$), e integradoras, como P_5 ($\phi = \infty$). En dinámicas estables simples, como P_2 ($\phi = 1$), y con dinámica dominante, como P_3 ($\phi = 8$), la estrategia convencional debe conservarse cuando ofrece un desempeño equivalente con menor complejidad. Para plantas inestables, no se establece una recomendación general: en P_6 ($\phi = 1$) no se obtuvo una familia consistente de soluciones comparables, mientras que en P_7 ($\phi = 8$) el PID alcanzó el mejor desempeño global y el MSP mostró su principal ventaja frente al SP en el rechazo de perturbaciones de entrada. Por ello, se requiere un estudio específico con una sintonía más especializada para este tipo de planta.

7.2. Recomendaciones

1. Se recomienda ampliar el análisis de incertidumbre del modelo, incorporando variaciones no solo en el tiempo de retardo, sino también en la ganancia, constantes de tiempo y ubicación de polos. Esto permitiría verificar si las ventajas observadas del MSP se mantienen cuando el desajuste afecta la dinámica completa de la planta y no únicamente el retardo.

2. Se recomienda desarrollar una evaluación específica para plantas inestables, especialmente para casos donde no se obtuvo una familia consistente de soluciones válidas. Para ello, debería considerarse una estrategia de sintonización más especializada, rangos de búsqueda ampliados o una etapa previa de estabilización que permita comparar de forma más concluyente el PID, SP y MSP.

3. Se recomienda complementar la evaluación por simulación con condiciones más cercanas a una implementación real, incluyendo ruido de medición, saturación del actuador, discretización del controlador y restricciones operativas. Esto permitiría determinar si el mejor desempeño observado en el dominio continuo se conserva en escenarios prácticos de control industrial.

8. REFERENCIAS

Åström, K. J., & Hägglund, T. (1995). *PID controllers: Theory, design, and tuning* (2nd ed.). Instrument Society of America.

Bequette, B. W. (2003). *Process control: Modeling, design, and simulation*. Prentice Hall PTR.

Bhattacharyya, S. P., Datta, A., & Keel, L. H. (2009). *Linear control theory: Structure, robustness, and optimization*. CRC Press.

Borase, R. P., Maghade, D. K., Sondkar, S. Y., & Pawar, S. N. (2021). A review of PID control, tuning methods and applications. *International Journal of Dynamics and Control, 9*, 818–827. https://doi.org/10.1007/s40435-020-00665-4

Camacho, C., Alvarez, H., Espin, J., & Camacho, O. (2023). An internal model based—sliding mode control for open-loop unstable chemical processes with time delay. *ChemEngineering, 7*(3), Article 53. https://doi.org/10.3390/chemengineering7030053

da Silva, L. R., Flesch, R. C. C., & Normey-Rico, J. E. (2020). Controlling industrial dead-time systems: When to use a PID or an advanced controller. *ISA Transactions, 99*, 339–350. https://doi.org/10.1016/j.isatra.2019.09.008

del Real Torres, A., Andreiana, D. S., Ojeda Roldán, Á., Hernández Bustos, A., & Acevedo Galicia, L. E. (2022). A review of deep reinforcement learning approaches for smart manufacturing in Industry 4.0 and 5.0 framework. *Applied Sciences, 12*(23), Article 12377. https://doi.org/10.3390/app122312377

Desborough, L., & Miller, R. (2002). Increasing customer value of industrial control performance monitoring—Honeywell's experience. En *Chemical Process Control VI*.

Díaz-Rodríguez, I. D., Han, S., & Bhattacharyya, S. P. (2019). *Analytical design of PID controllers*. Springer. https://doi.org/10.1007/978-3-030-18228-1

Freitas Machado, G., Olímpio Pereira, R. D., Torrico, B. C., & Normey-Rico, J. E. (2019). Por que não deveríamos esquecer o preditor de Smith. *Anais do Simpósio Brasileiro de Automação Inteligente*. https://doi.org/10.17648/sbai-2019-111455

Grimholt, C., & Skogestad, S. (2012). Optimal PI-control and verification of the SIMC tuning rule. *IFAC Proceedings Volumes, 45*(3), 11–22.

Grimholt, C., & Skogestad, S. (2013). Optimal PID-control on first order plus time delay systems & verification of the SIMC rules. *IFAC Proceedings Volumes, 46*(32), 265–270.

Holene, A. L. (2013). *Performance and robustness of Smith predictor control and comparison with PID control* [Master's thesis, Norwegian University of Science and Technology].

Huang, C. J., Chicoma, E. D. T., & Huang, Y. H. (2019). Evaluating the factors that are affecting the implementation of Industry 4.0 technologies in manufacturing MSMEs, the case of Peru. *Processes, 7*(3), Article 161. https://doi.org/10.3390/pr7030161

Korupu, V. L., & Muthukumarasamy, M. (2021). A comparative study of various Smith predictor configurations for industrial delay processes. *Chemical Product and Process Modeling*. https://doi.org/10.1515/cppm-2021-0026

Lennartson, B., & Kristiansson, B. (2009). Evaluation and tuning of robust PID controllers. *IET Control Theory & Applications, 3*(3), 294–302. https://doi.org/10.1049/iet-cta:20060450

Mariátegui, L. (2024, 19 de agosto). Industria 5.0: el desafío de los nuevos profesionales. *Conexión Universitaria, Gestión*. https://blogs.gestion.pe/conexion-universitaria/2024/08/industria-5-0-el-desafio-de-los-nuevos-profesionales.html

Maxim, A., Copot, D., Copot, C., & Ionescu, C. M. (2019). The 5W's for control as part of Industry 4.0: Why, what, where, who, and when—A PID and MPC control perspective. *Inventions, 4*(1), Article 10. https://doi.org/10.3390/inventions4010010

Mendoza, M. (2017). El Perú solo tiene un 1% de industrias 4.0. *El Comercio*. https://elcomercio.pe/economia/peru-1-industrias-4-0-noticia-467091-noticia/

Moeuf, A., Pellerin, R., Lamouri, S., Tamayo-Giraldo, S., & Barbaray, R. (2018). The industrial management of SMEs in the era of Industry 4.0. *International Journal of Production Research, 56*(3), 1118–1136. https://doi.org/10.1080/00207543.2017.1372647

Normey-Rico, J. E., & Camacho, E. F. (2007). *Control of dead-time processes*. Springer. https://doi.org/10.1007/978-1-84628-829-6

Normey-Rico, J. E., & Camacho, E. F. (2008a). Dead-time compensators: A survey. *Control Engineering Practice, 16*(4), 407–428. https://doi.org/10.1016/j.conengprac.2007.05.006

Normey-Rico, J. E., & Camacho, E. F. (2008b). Control of dead-time processes. *IEEE Control Systems Magazine, 28*, 136–137.

Normey-Rico, J. E., & Camacho, E. F. (2009). Unified approach for robust dead-time compensator design. *Journal of Process Control, 19*(1), 38–47. https://doi.org/10.1016/j.jprocont.2008.02.003

Normey-Rico, J. E., & Guzmán, J. L. (2013). Unified PID tuning approach for stable, integrative, and unstable dead-time processes. *Industrial & Engineering Chemistry Research, 52*(47), 16811–16819. https://doi.org/10.1021/ie401722y

Pérez Aviles, D., Ildefonso Linares, L. M., Estrada Mogolbay, M. C., & Menéndez Ramos, R. M. (2022). *Negocia.Pe: Plataforma financiera de factoring colaborativo para MIPYMES* [Tesis de maestría, Pontificia Universidad Católica del Perú]. Repositorio de Tesis PUCP. http://hdl.handle.net/20.500.12404/22043

Reznik, L., Ghanayem, O., & Bourmistrov, A. (2000). PID plus fuzzy controller structures as a design base for industrial applications. *Engineering Applications of Artificial Intelligence, 13*(4), 419–430. https://doi.org/10.1016/S0952-1976(00)00013-0

Rivera, D. E., Morari, M., & Skogestad, S. (1986). Internal model control: PID controller design. *Industrial & Engineering Chemistry Process Design and Development, 25*(1), 252–265. https://doi.org/10.1021/i200032a041

Samad, T. (2017). A survey on industry impact and challenges thereof. *IEEE Control Systems Magazine, 37*, 17–18.

Schumacher, A., Erol, S., & Sihn, W. (2016). A maturity model for assessing Industry 4.0 readiness and maturity of manufacturing enterprises. *Procedia CIRP, 52*, 161–166. https://doi.org/10.1016/j.procir.2016.07.040

Shinskey, F. G. (1967). *Process-control systems: Application, design, adjustment*. McGraw-Hill.

Skogestad, S. (2003). Simple analytic rules for model reduction and PID controller tuning. *Journal of Process Control, 13*(4), 291–309. https://doi.org/10.1016/S0959-1524(02)00062-8

Skogestad, S., & Grimholt, C. (2018). Should we forget the Smith Predictor? *IFAC-PapersOnLine, 51*(4), 769–774. https://doi.org/10.1016/j.ifacol.2018.06.203

Skogestad, S., & Postlethwaite, I. (2005). *Multivariable feedback control: Analysis and design* (2nd ed.). Wiley.

Smith, O. J. M. (1957). Closer control of loops with dead time. *Chemical Engineering Progress, 53*(5), 217–219.

9. ANEXOS

Anexo	Contenido	Resumen
Anexo 1	Matriz de consistencia	Relaciona problema, objetivos, hipótesis y variables de la investigación; permite verificar la coherencia metodológica del estudio.
Anexo 2	Matriz de operacionalización de variables	Define las variables independiente y dependiente, sus dimensiones e indicadores, incluyendo estrategia de control, desempeño dinámico y robustez.
Anexo 3	Matriz de hallazgos de antecedentes	Resume los principales estudios revisados sobre PID, Predictor de Smith, Predictor de Smith Modificado y control de procesos con retardo.
Anexo 4	Tablas de resultados	Presenta los resultados detallados de las simulaciones para las plantas P1, P2, P3, P4, P5 y P7, incluyendo sintonía, índices de desempeño y métricas de robustez.
Anexo 5	Scripts de evaluación	Incluye los códigos desarrollados para ejecutar las etapas de evaluación, generación de curvas, validación temporal y exportación de resultados.

9.1. Anexo 1. Matriz de consistencia

Nivel	Variables	Problema	Objetivo	Hipótesis
General	**Independiente:** Estrategia de control evaluada, considerando PI/PID, Predictor de Smith clásico (SP) y Predictor de Smith Modificado (MSP). **Dependiente:** Desempeño del sistema de control, entendido como la medición conjunta del rendimiento dinámico y la robustez del lazo cerrado.	¿Cómo diseñar y evaluar sistemáticamente una estrategia de control basada en el Predictor de Smith Modificado que permita determinar su aplicabilidad en procesos industriales con retardo dominante, comportamiento integrador o dinámica inestable, en comparación con el controlador PID y el Predictor de Smith clásico, en términos de desempeño y robustez?	Diseñar y evaluar sistemáticamente una estrategia de control basada en el Predictor de Smith Modificado que permita determinar su aplicabilidad en procesos industriales con retardo dominante, comportamiento integrador o dinámica inestable, en comparación con el controlador PID y el Predictor de Smith clásico, en términos de desempeño y robustez	El Predictor de Smith Modificado permite obtener un mejor compromiso entre rendimiento y robustez que el controlador PID y el Predictor de Smith clásico en procesos con retardo dominante, especialmente en plantas integradoras e inestables, donde la compensación del retardo y la preservación de la estabilidad del lazo cerrado resultan más exigentes.
Especifico 1	**Independiente:** Formulación de la estructura del Predictor de Smith Modificado. **Dependiente:** Capacidad de obtener lazos cerrados estables y evaluables para plantas estables autorreguladas, integradoras e inestables con retardo dominante.	¿Cómo formular una estructura de control basada en el Predictor de Smith Modificado que permita abordar las dinámicas de procesos industriales con retardo dominante, integradores e inestables?	Formular una estructura de control basada en el Predictor de Smith Modificado que permita abordar las dinámicas de procesos industriales con retardo dominante, integradores e inestables.	La formulación del Predictor de Smith Modificado, mediante una estructura predictiva ajustada y el uso de un filtro predictivo, permite adaptar el Predictor de Smith clásico a procesos con retardo dominante, integradores e inestables, obteniendo lazos cerrados estables y evaluables bajo los criterios definidos en la investigación.
Específico 2	**Independiente:** Procedimiento de evaluación de estrategias de control. **Dependiente:** Comparabilidad y consistencia de los resultados obtenidos para distintos modelos de proceso, estructuras de control y niveles de robustez y rendimiento deseados.	¿Cómo establecer un procedimiento sistemático que permita realizar la búsqueda de sintonías, la evaluación y la comparación del rendimiento y la robustez de las estrategias de control consideradas?	Establecer un procedimiento sistemático que permita realizar la búsqueda de sintonías, la evaluación y la comparación del rendimiento y la robustez de las estrategias de control consideradas.	El procedimiento propuesto permite evaluar estrategias de control de manera reproducible en distintos escenarios de análisis, al combinar diferentes modelos de proceso, estructuras de control y niveles de robustez y rendimiento deseados bajo una misma lógica de búsqueda, validación y comparación. Esto permite obtener una base consistente para analizar el desempeño relativo de cada estrategia y extender el procedimiento a otros procesos, controladores o condiciones de estudio.
Específico 3	**Independiente:** Estrategia de control evaluada: PID, SP y MSP. **Dependiente:** Rendimiento dinámico ante cambios de referencia y perturbaciones de entrada y salida.	¿En qué medida el Predictor de Smith Modificado mejora el desempeño ante cambios de referencia y perturbaciones frente al controlador PID y al Predictor de Smith clásico en procesos industriales con retardo dominante, integradores e inestables?	Evaluar en qué medida el Predictor de Smith Modificado mejora el desempeño ante cambios de referencia y perturbaciones frente al controlador PID y al Predictor de Smith clásico en procesos industriales con retardo dominante, integradores e inestables.	El Predictor de Smith Modificado mejora el desempeño ante cambios de referencia y perturbaciones de entrada y salida frente al controlador PID y al Predictor de Smith clásico, lo cual se refleja en una reducción del error acumulado de la respuesta en el dominio del tiempo, cuantificado mediante el índice J, principalmente en plantas integradoras e inestables con retardo dominante.
Específico 4	**Independiente:** Estrategia de control evaluada: PID, SP y MSP. **Dependiente:** Robustez del lazo cerrado frente a incertidumbres en el tiempo de retardo.	¿En qué medida el Predictor de Smith Modificado mejora la robustez frente al controlador PID y al Predictor de Smith clásico en procesos industriales con retardo dominante, integradores e inestables?	Evaluar en qué medida el Predictor de Smith Modificado mejora la robustez frente al controlador PID y al Predictor de Smith clásico en procesos industriales con retardo dominante, integradores e inestables.	El Predictor de Smith Modificado presenta mayor robustez frente al controlador PID y al Predictor de Smith clásico, lo cual se refleja en valores adecuados de M_{ST}, mayores márgenes de retardo y una menor degradación de la respuesta dinámica ante errores en el tiempo de retardo, especialmente en plantas integradoras e inestables con retardo dominante.

9.2. Anexo 2. Matriz de operacionalización de variables

Problema	Variables	Definición	Dimensiones	Indicadores
¿Cómo diseñar y evaluar sistemáticamente una estrategia de control basada en el Predictor de Smith Modificado que permita abordar las limitaciones de desempeño y robustez que pueden presentar los controladores PID y el Predictor de Smith clásico en procesos industriales con retardo dominante, integradores e inestables?	**Independiente:** Estrategia de control evaluada	Arquitectura de control empleada para regular la respuesta del proceso ante cambios en la referencia o perturbaciones. En la investigación se consideran controladores PI/PID, Predictor de Smith clásico y Predictor de Smith Modificado, aplicados a modelos representativos con retardo dominante.	Arquitectura de control	PI, PID, SP, MSP
			Parámetros de sintonía	Ganancia proporcional, tiempo integral, tiempo derivativo y parámetros asociados a la estructura de control
			Condición de aplicación	Tipo de planta evaluada: estable autorregulada, integradora e inestable con retardo dominante
	Dependiente: Desempeño del sistema de control	Medición conjunta del rendimiento y la robustez del lazo cerrado. El rendimiento se asocia con la calidad de la respuesta dinámica ante cambios de referencia o perturbaciones; la robustez, con la capacidad del sistema para conservar un comportamiento estable y aceptable frente a incertidumbres, especialmente en el tiempo de retardo.	Rendimiento dinámico	Error acumulado mediante el índice J; IAE ante cambios de referencia y perturbaciones de entrada y salida; tiempo de subida; tiempo de estabilización; sobrepaso
			Robustez del lazo cerrado	Valor real de M_{ST}; margen de retardo; margen de fase; margen de ganancia; ancho de banda; respuesta ante errores en el tiempo de retardo

9.3. Anexo 3. Matriz de hallazgos – Revisión de antecedentes

Ítem	Fuente	Modelo de planta	Algoritmo de control	Metodología de evaluación	Principales hallazgos
1	O. J. M. Smith (1957)	-	▪ SP	-	▪ Se propuso el primero compensador de tiempos de retardo (DTC), un mecanismo predictivo simple conocido como el Predictor de Smith ▪ Presenta inconvenientes de estabilidad y robustez que complicarían su aplicación en la práctica, en particular, la alta sensibilidad al error de la estimación del tiempo de retardo
2	Lennartson & Kristiansson (2009)	▪ SOPDT	▪ PI ▪ PID ▪ SP ▪ PID+SP	▪ Índices de rendimiento y robustez en el dominio de la frecuencia	▪ PID bien sintonizado mejor que PI+ SP ▪ SP genera una disminución de la robustez para tiempos de retardo grandes ▪ PID+SP, mejoras de 15-16% para retardos bajos y medios.
3	Normey-Rico & Camacho (2008a)	▪ FOPDT ▪ IPDT ▪ UFOPDT	▪ SP ▪ MSP	▪ Características de la curva de respuesta ▪ Análisis de estabilidad en el dominio de la frecuencia	▪ Se analizaron varias estructuras y reglas de sintonía para el control de procesos estables y estructuras modificadas para estabilizar el control de las plantas integradores e inestables con tiempo de retardo. ▪ El SP original puede ofrecer mejores respuestas que un controlador PID, principalmente, en el caso donde el tiempo de retardo es dominante y bien conocido, obteniendo un buen equilibrio entre robustez y rendimiento, ajustando adecuadamente el controlador principal y el predictor.
4	Normey-Rico & Camacho (2009)	▪ FOPDT ▪ IPDT ▪ UFOPDT	▪ SP ▪ MSP	▪ Características de la curva de respuesta ▪ Análisis de estabilidad en el dominio de la frecuencia	▪ Propusieron un procedimiento de sintonización unificado del SP modificado con filtros de referencia y predictivos para plantas estables, inestables e integradoras con tiempo de retardo. ▪ La sintonización se basaba en dos términos que permitían agregar polos en lazo cerrado a conveniencia y obtener una buena compensación de rendimiento-robustez.
5	Normey-Rico & Guzmán (2013)	▪ FOPDT ▪ IPDT ▪ UFOPDT	▪ PID ▪ FSP	▪ Características de la curva de respuesta ▪ Minimización del índice IAE ▪ Análisis de estabilidad en el dominio de la frecuencia (M_{ST}, DM)	▪ El controlador PID propuesto es simple de analizar y sintonizar y se basa en una aproximación de baja frecuencia de la estructura ideal de compensación de tiempo de retardo del predictor de Smith filtrado (FSP) ▪ Solo se utiliza un parámetro para obtener un compromiso entre rendimiento y robustez, y se desarrolla un procedimiento de ajuste para seleccionarlo

Ítem	Fuente	Modelo de planta	Algoritmo de control	Metodología de evaluación	Principales hallazgos
6	Holene (2013)	▪ FOPDT ▪ IPDT ▪ SOPDT	▪ SP ▪ MSP	▪ Función de rendimiento objetivo basado en IAE ▪ Criterio del máximo pico de sensibilidad (M_{ST}) ▪ Margen de retardo (DM)	▪ El SP produce pequeñas mejoras de rendimiento en comparación con el PI y PID óptimos en procesos de segundo orden. ▪ Para procesos de primer orden, los controladores PI y PID óptimos tienen un rendimiento superior al predictor de Smith. ▪ Cuando el tiempo de retardo del proceso variaba, incluso los SP con valores modestos de M_S tienden a desestabilizarse.
7	Skogestad & Grimholt (2018)	▪ FOPDT	▪ PI ▪ PID ▪ SP	▪ Función de rendimiento objetivo basado en IAE ▪ Criterio del máximo pico de sensibilidad (M_{ST}) ▪ Margen de retardo (DM) ▪ BODE	▪ SP, mejora no significativa respecto a PI y PID ▪ PID presenta un mejor desempeño. ▪ No evalúa PID +SP
8	da Silva et al. (2019)	▪ FOPDT	▪ PID ▪ PID+ Anti Windup ▪ FSP	▪ Características de la curva de respuesta ▪ Función de rendimiento objetivo basado en IAE ▪ Criterio del máximo pico de sensibilidad (M_{ST}) ▪ Margen de retardo (DM)	▪ Escoger entre PID o un algoritmo más complejo depende más de la calidad del modelo ▪ Para un entorno industrial (soluciones robustas), un PID bien sintonizado con AW puede proporcionar mejores resultados que estrategias más complejas (DTC o MPC).
9	Machado et al. (2019)	▪ FOPDT	▪ SP ▪ MSP	▪ Índice de rendimiento objetivo basado en IAE ▪ Criterio del máximo pico de sensibilidad (M_{ST}) ▪ Margen de retardo (DM)	▪ Se comparó una estructura modificada del Smith Predictor, denominada Compensador de tiempo de retardo simplificado (SDTC), y controladores PI/PID en términos de rendimiento y robustez. ▪ Los principales resultados mostraron que el SDTC es mejor o similar que los controladores PI/PID en términos de rendimiento. ▪ Se reportó la misma caída de estabilidad ante incertidumbres en el tiempo de retardo que mostro Skogestad & Grimholt (2018)

9.4. Anexo 4. Tablas de resultados

9.4.1. Planta 1

Tabla 9-1. Resultados de la búsqueda de sintonías que minimizan J para cada nivel objetivo de M_{ST} en P_1

Planta	Tipo	Controlador	k_c	τ_i	τ_d	M_{ST}	IAE_{do}	IAE_{di}	J
P_1	Estable	PI	0.001	0.171	0	1.005	27.637	26.974	13.247
P_1	Estable	PI	0.071	0.551	0	1.098	7.632	7.799	3.739
P_1	Estable	PI	0.161	1.051	0	1.205	6.456	6.637	3.172
P_1	Estable	PI	0.211	0.751	0	1.309	3.559	3.755	1.771
P_1	Estable	PI	0.241	0.611	0	1.404	2.535	2.735	1.275
P_1	Estable	PI	0.241	0.461	0	1.497	1.913	2.113	0.973
P_1	Estable	PI	0.281	0.481	0	1.603	1.712	1.915	0.877
P_1	Estable	PI	0.351	0.641	0	1.699	1.826	2.036	0.934
P_1	Estable	PI	0.401	0.771	0	1.805	1.923	2.138	0.982
P_1	Estable	PI	0.461	1.241	0	1.919	2.692	2.914	1.356
P_1	Estable	PI	0.491	1.751	0	2.001	3.561	3.785	1.778
P_1	Estable	PID	0.001	0.101	0.06	1.010	26.141	25.572	12.543
P_1	Estable	PID	0.081	0.581	0	1.110	7.088	7.262	3.477
P_1	Estable	PID	0.161	0.981	0	1.207	6.048	6.232	2.975
P_1	Estable	PID	0.201	0.741	0	1.292	3.687	3.882	1.832
P_1	Estable	PID	0.241	0.601	0	1.408	2.495	2.694	1.255
P_1	Estable	PID	0.241	0.461	0	1.497	1.914	2.113	0.974
P_1	Estable	PID	0.281	0.481	0	1.603	1.713	1.916	0.877
P_1	Estable	PID	0.341	0.601	0	1.693	1.764	1.973	0.903
P_1	Estable	PID	0.401	0.761	0	1.810	1.899	2.114	0.970
P_1	Estable	PID	0.461	1.241	0	1.919	2.693	2.915	1.357
P_1	Estable	PID	0.501	1.801	0	2.040	3.590	3.815	1.793
P_1	Estable	SP	0.001	0.171	0	1.005	27.491	26.986	13.212
P_1	Estable	SP	0.071	0.391	0	1.103	5.473	6.645	2.924
P_1	Estable	SP	0.131	0.321	0	1.205	2.450	3.640	1.462
P_1	Estable	SP	0.231	0.331	0	1.304	1.433	2.628	0.970
P_1	Estable	SP	0.371	0.341	0	1.405	0.919	2.120	0.722
P_1	Estable	SP	0.491	0.301	0	1.505	0.613	1.819	0.575
P_1	Estable	SP	0.711	0.291	0	1.605	0.409	1.622	0.478

Planta	Tipo	Controlador	k_c	τ_i	τ_d	M_{ST}	IAE_{do}	IAE_{di}	J
P_1	Estable	SP	1.021	0.271	0	1.704	0.265	1.485	0.410
P_1	Estable	SP	1.531	0.241	0	1.804	0.157	1.386	0.359
P_1	Estable	SP	1.861	0.151	0	1.905	0.081	1.316	0.324
P_1	Estable	SP	1.861	0.001	0	2.000	0.002	1.280	0.295
P_1	Estable	MSP	0.001	0.231	0	1.005	28.277	27.574	13.548
P_1	Estable	MSP	0.111	0.741	0	1.104	7.953	8.116	3.893
P_1	Estable	MSP	0.211	0.591	0	1.204	4.220	4.403	2.088
P_1	Estable	MSP	0.391	0.591	0	1.305	2.931	3.115	1.463
P_1	Estable	MSP	0.681	0.591	0	1.405	2.287	2.472	1.152
P_1	Estable	MSP	1.101	0.531	0	1.505	1.902	2.088	0.965
P_1	Estable	MSP	1.651	0.391	0	1.605	1.656	1.843	0.846
P_1	Estable	MSP	1.771	0.151	0	1.705	1.505	1.692	0.773
P_1	Estable	MSP	1.801	0.001	0	1.728	1.420	1.611	0.732
P_1	Estable	MSP	2.001	0.001	0	1.728	1.420	1.611	0.732
P_1	Estable	MSP	2.001	0.001	0	1.728	1.420	1.611	0.732

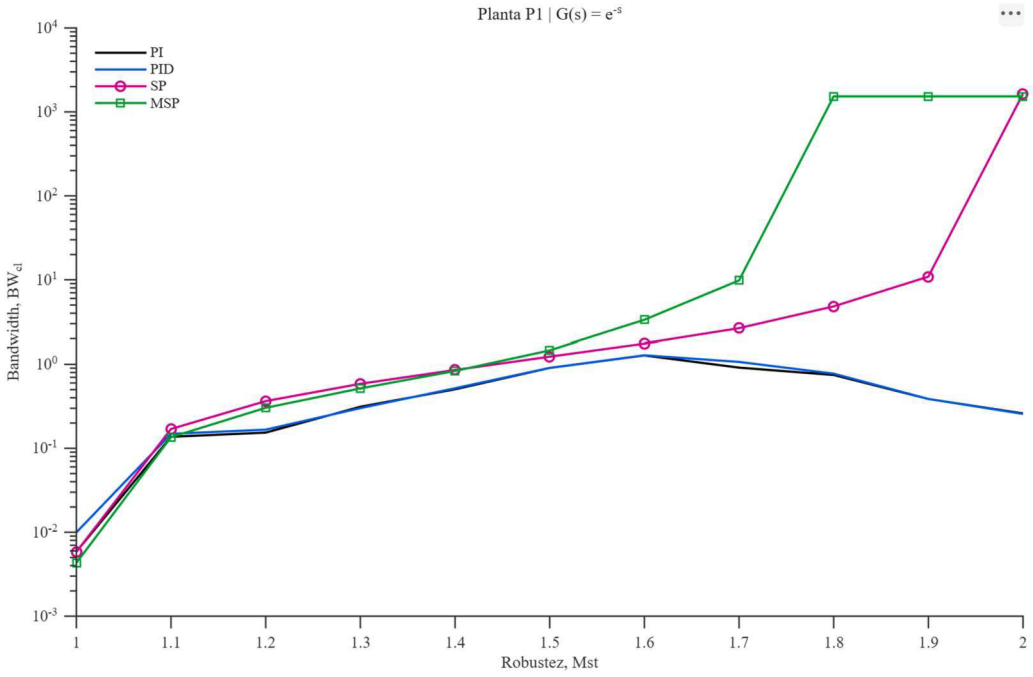

Figura 9-1. Ancho de banda en función de M_{ST} para P_1

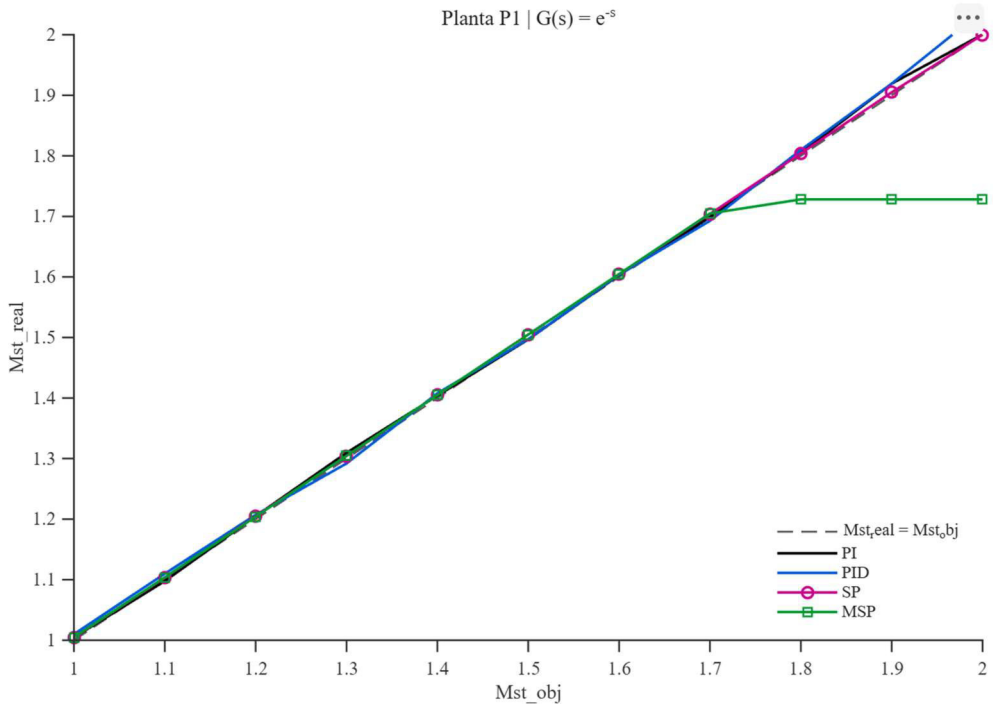

Figura 9-2. Comparación entre M_{ST} real y M_{ST} objetivo para P_1

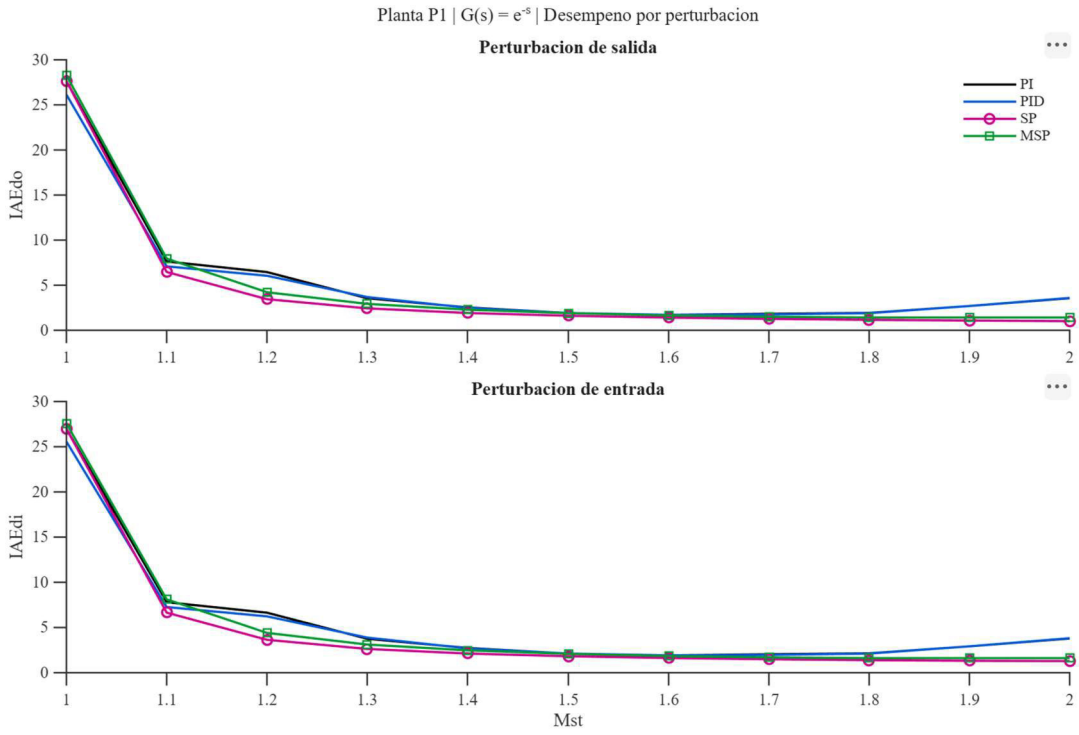

Figura 9-3. Índices IAE_{di} e IAE_{do} en función de M_{ST} para P_1

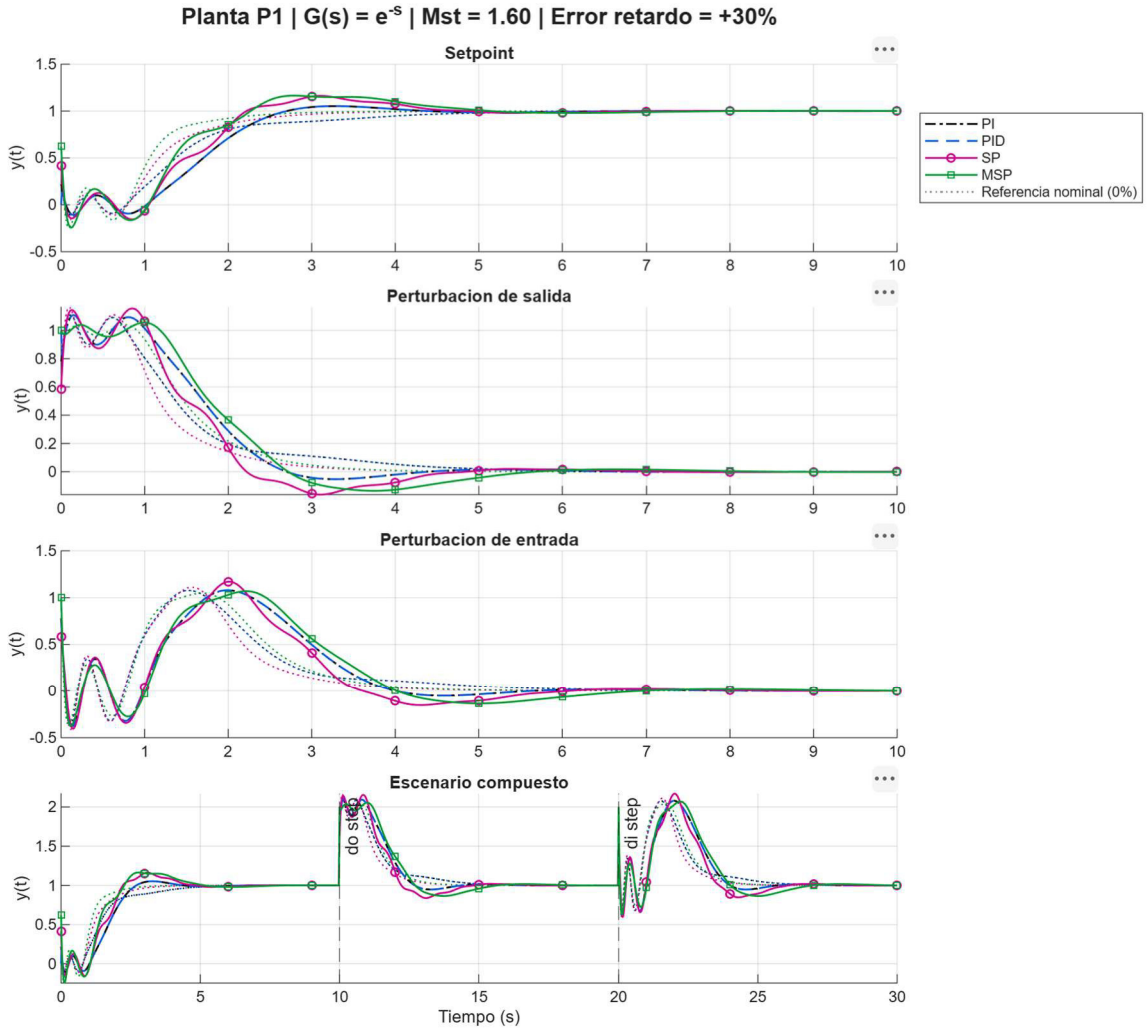

Figura 9-4. Respuestas controladas ante un error de modelado del tiempo de retardo de +30% para P_1 ($M_{ST} = 1.6$)

9.4.2. Planta 2

Tabla 9-2. Resultados de la búsqueda de sintonías que minimizan J para cada nivel objetivo de M_{ST} en P_2

Planta	Tipo	Controlador	k_c	τ_i	τ_d	M_{ST}	IAE_{do}	IAE_{di}	J
P_2	Estable	PI	0.001	0.321	0	1.005	28.780	26.959	9.641
P_2	Estable	PI	0.121	1.071	0	1.105	8.654	8.612	2.986
P_2	Estable	PI	0.221	1.051	0	1.205	4.754	4.770	1.647
P_2	Estable	PI	0.301	1.011	0	1.305	3.359	3.376	1.165
P_2	Estable	PI	0.371	0.991	0	1.402	2.671	2.689	0.927
P_2	Estable	PI	0.461	1.041	0	1.504	2.269	2.276	0.786
P_2	Estable	PI	0.541	1.081	0	1.604	2.080	2.016	0.708
P_2	Estable	PI	0.611	1.111	0	1.704	2.005	1.837	0.664
P_2	Estable	PI	0.691	1.181	0	1.804	1.955	1.727	0.637
P_2	Estable	PI	0.841	1.561	0	1.905	1.936	1.874	0.659
P_2	Estable	PI	0.981	2.351	0	2.003	2.396	2.413	0.832
P_2	Estable	PID	0.001	0.101	1.6	1.010	26.060	24.568	8.757
P_2	Estable	PID	0.061	0.601	1.3	1.101	9.411	9.324	3.240
P_2	Estable	PID	0.181	0.761	0.8	1.208	4.203	4.220	1.457
P_2	Estable	PID	0.361	0.981	0.44	1.308	2.717	2.736	0.943
P_2	Estable	PID	0.361	0.701	0.5	1.410	2.011	1.988	0.692
P_2	Estable	PID	0.421	0.701	0.52	1.510	1.776	1.721	0.605
P_2	Estable	PID	0.481	0.721	0.52	1.604	1.632	1.554	0.551
P_2	Estable	PID	0.561	0.781	0.5	1.705	1.522	1.430	0.511
P_2	Estable	PID	0.701	0.981	0.42	1.802	1.438	1.419	0.494
P_2	Estable	PID	0.701	1.001	0.52	1.907	1.438	1.448	0.499
P_2	Estable	PID	0.701	0.821	0.5	2.003	1.467	1.198	0.461
P_2	Estable	SP	0.001	0.321	0	1.005	28.697	26.962	9.627
P_2	Estable	SP	0.151	1.131	0	1.105	7.343	8.287	2.703
P_2	Estable	SP	0.301	1.021	0	1.205	3.392	4.408	1.348
P_2	Estable	SP	0.501	1.011	0	1.304	2.018	3.036	0.873
P_2	Estable	SP	0.711	0.941	0	1.404	1.323	2.341	0.633
P_2	Estable	SP	0.961	0.881	0	1.504	0.925	1.935	0.494
P_2	Estable	SP	1.381	0.891	0	1.604	0.660	1.663	0.401
P_2	Estable	SP	1.971	0.891	0	1.705	0.473	1.470	0.336
P_2	Estable	SP	2.571	0.791	0	1.804	0.364	1.326	0.292

Planta	Tipo	Controlador	k_c	τ_i	τ_d	M_{ST}	IAE_{do}	IAE_{di}	J
P_2	Estable	SP	3.181	0.651	0	1.904	0.303	1.224	0.264
P_2	Estable	SP	3.781	0.511	0	2.001	0.265	1.155	0.245
P_2	Estable	MSP	0.001	0.301	0	1.005	28.693	26.884	9.613
P_2	Estable	MSP	0.141	1.061	0	1.105	8.218	8.185	2.837
P_2	Estable	MSP	0.281	0.991	0	1.204	4.380	4.397	1.518
P_2	Estable	MSP	0.421	0.901	0	1.305	2.995	3.012	1.039
P_2	Estable	MSP	0.571	0.831	0	1.405	2.312	2.328	0.802
P_2	Estable	MSP	0.851	0.891	0	1.505	1.910	1.920	0.662
P_2	Estable	MSP	1.181	0.921	0	1.605	1.657	1.653	0.572
P_2	Estable	MSP	1.601	0.941	0	1.705	1.488	1.461	0.510
P_2	Estable	MSP	2.151	0.941	0	1.805	1.378	1.311	0.465
P_2	Estable	MSP	2.751	0.871	0	1.902	1.328	1.190	0.436
P_2	Estable	MSP	3.351	0.731	0	2.004	1.316	1.092	0.417

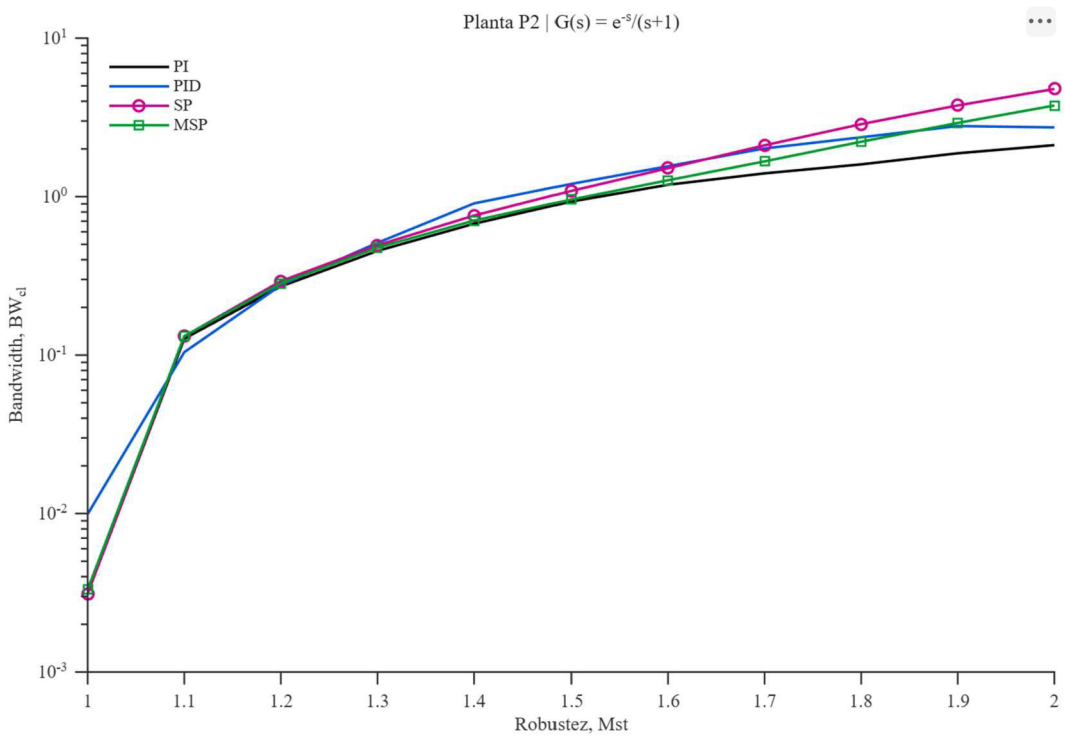

Figura 9-5. Ancho de banda en función de M_{ST} para P_2

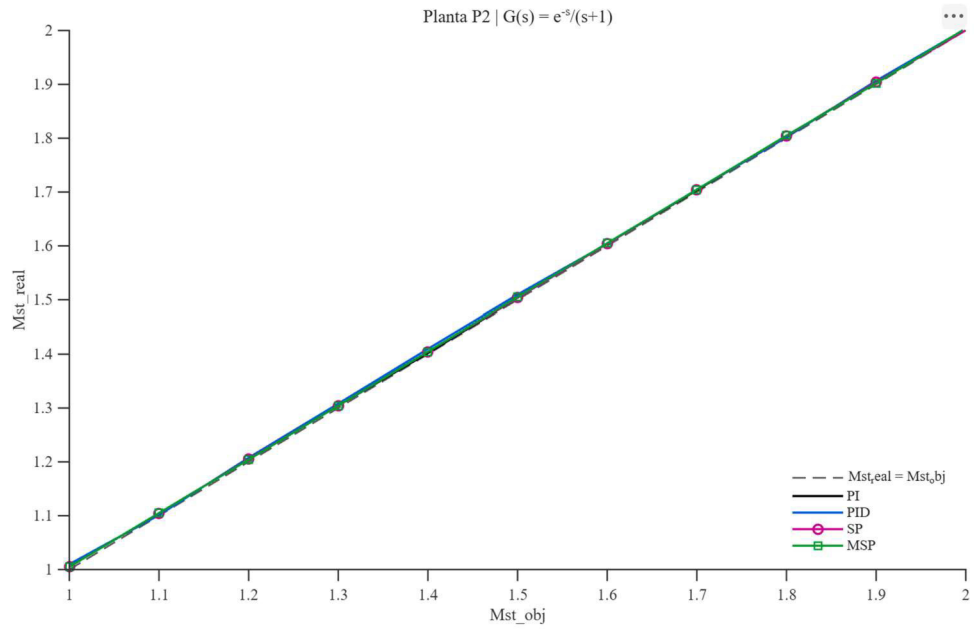

Figura 9-6. Comparación entre M_{ST} real y M_{ST} objetivo para P_2

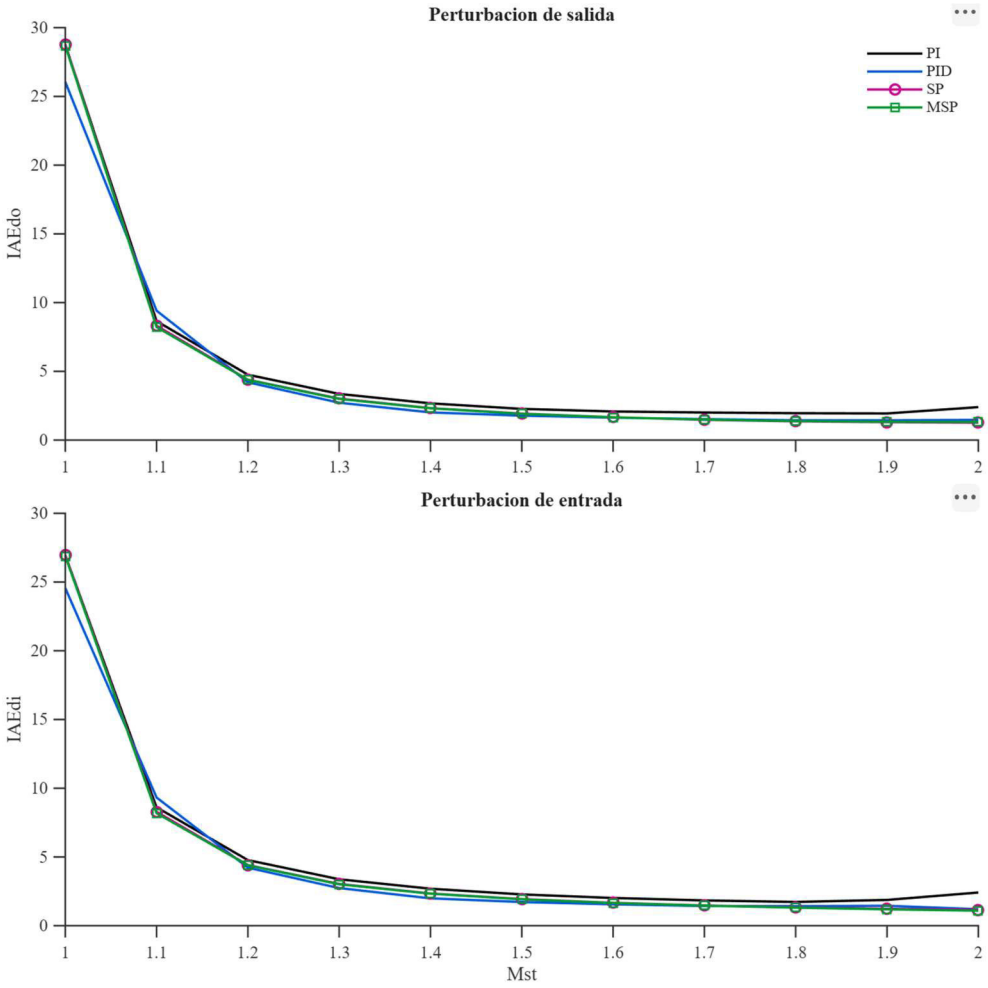

Figura 9-7. Índices IAE_{di} e IAE_{do}en función de M_{ST} para P_2

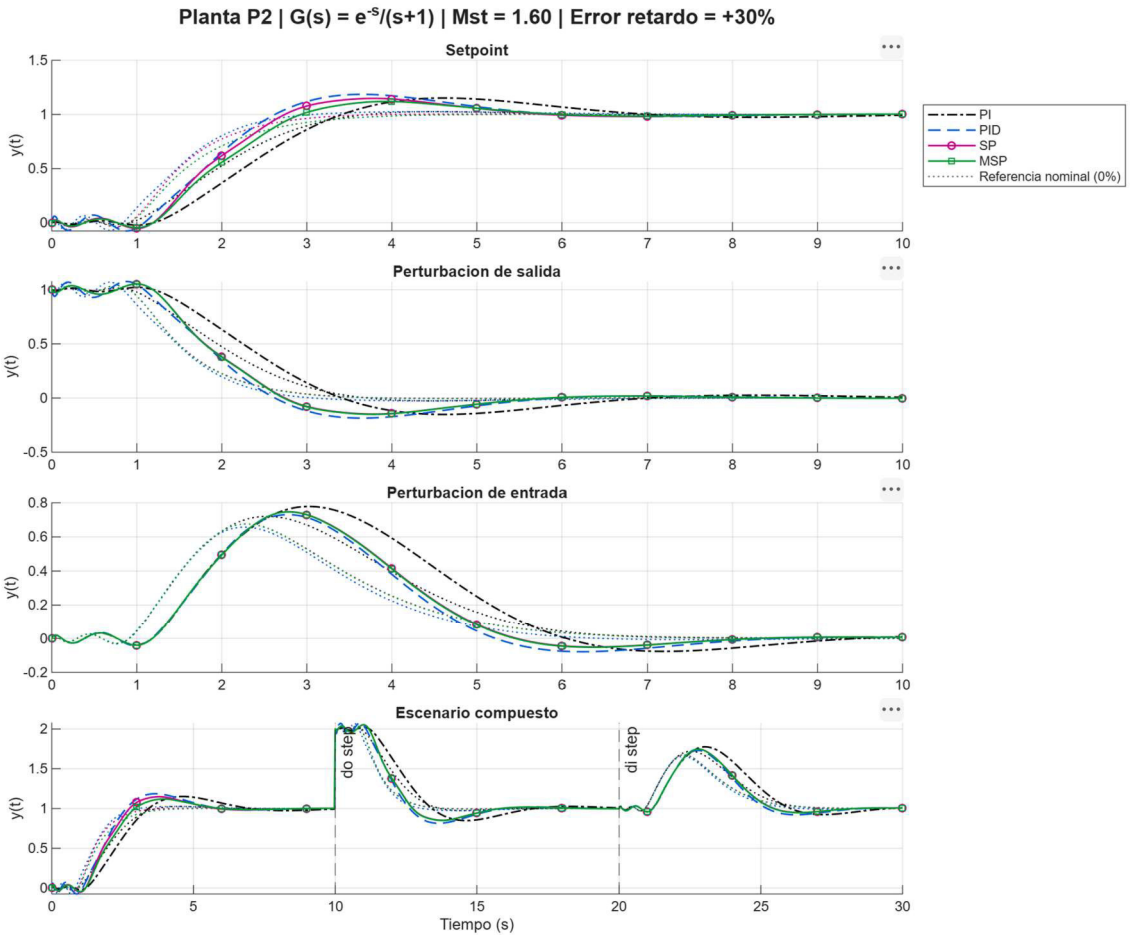

Figura 9-8. Respuestas controladas ante un error de modelado del tiempo de retardo de +30% para P_2 ($M_{ST} = 1.6$)

9.4.3. Planta 3

Tabla 9-3. Resultados de la búsqueda de sintonías que minimizan J para cada nivel objetivo de M_{ST} en P_3

Planta	Tipo	Controlador	k_c	τ_i	τ_d	M_{ST}	IAE_{do}	IAE_{di}	J
P_3	Estable	PI	0.041	7.931	0	1.005	27.928	20.072	15.354
P_3	Estable	PI	0.651	7.101	0	1.079	10.618	9.353	6.601
P_3	Estable	PI	1.301	6.201	0	1.166	5.437	4.748	3.362
P_3	Estable	PI	1.901	5.351	0	1.262	4.156	2.834	2.217
P_3	Estable	PI	2.501	4.501	0	1.380	3.598	1.833	1.644
P_3	Estable	PI	3.101	4.181	0	1.505	3.246	1.359	1.352
P_3	Estable	PI	3.521	3.981	0	1.605	3.089	1.133	1.216
P_3	Estable	PI	3.871	3.761	0	1.705	3.009	0.974	1.127
P_3	Estable	PI	4.211	3.641	0	1.805	2.937	0.867	1.063
P_3	Estable	PI	4.561	3.631	0	1.905	2.862	0.798	1.015
P_3	Estable	PI	4.811	3.481	0	2.005	2.838	0.726	0.977
P_3	Estable	PID	0.101	6.661	0.66	1.010	24.537	18.153	13.719
P_3	Estable	PID	0.701	5.841	1.08	1.108	8.489	7.752	5.399
P_3	Estable	PID	1.301	5.041	1	1.206	4.877	3.933	2.871
P_3	Estable	PID	1.901	4.241	0.92	1.307	3.640	2.320	1.870
P_3	Estable	PID	2.501	3.441	0.84	1.409	3.010	1.476	1.350
P_3	Estable	PID	3.101	2.641	0.74	1.500	2.682	0.963	1.047
P_3	Estable	PID	3.681	2.321	0.68	1.610	2.459	0.725	0.890
P_3	Estable	PID	4.261	2.421	0.62	1.709	2.301	0.625	0.809
P_3	Estable	PID	4.841	2.541	0.56	1.809	2.208	0.554	0.756
P_3	Estable	PID	5.381	2.561	0.5	1.909	2.180	0.491	0.721
P_3	Estable	PID	5.681	2.541	0.5	2.009	2.147	0.458	0.699
P_3	Estable	SP	0.041	7.891	0	1.005	27.790	20.074	15.323
P_3	Estable	SP	0.651	7.051	0	1.071	10.386	9.857	6.771
P_3	Estable	SP	2.001	5.251	0	1.195	3.523	3.611	2.416
P_3	Estable	SP	2.751	4.251	0	1.262	2.720	2.523	1.747
P_3	Estable	SP	3.501	3.001	0	1.351	2.255	1.831	1.333
P_3	Estable	SP	4.251	2.001	0	1.491	1.937	1.445	1.088
P_3	Estable	SP	4.851	1.291	0	1.705	1.741	1.240	0.952
P_3	Estable	SP	4.851	1.291	0	1.705	1.741	1.240	0.952
P_3	Estable	SP	5.451	1.101	0	1.803	1.581	1.176	0.887

Planta	Tipo	Controlador	k_c	τ_i	τ_d	M_{ST}	IAE_{do}	IAE_{di}	J
P_3	Estable	SP	6.051	0.951	0	1.899	1.447	1.131	0.836
P_3	Estable	SP	6.651	0.821	0	1.999	1.335	1.097	0.795
P_3	Estable	MSP	0.041	8.151	0	1.005	27.901	20.047	15.336
P_3	Estable	MSP	0.651	7.301	0	1.076	11.048	9.591	6.806
P_3	Estable	MSP	1.901	5.701	0	1.206	4.339	3.446	2.531
P_3	Estable	MSP	2.501	4.751	0	1.268	3.755	2.369	1.918
P_3	Estable	MSP	4.001	2.751	0	1.446	3.163	1.187	1.257
P_3	Estable	MSP	4.001	2.751	0	1.446	3.163	1.187	1.257
P_3	Estable	MSP	4.751	1.751	0	1.637	3.002	0.845	1.068
P_3	Estable	MSP	4.751	1.751	0	1.637	3.002	0.845	1.068
P_3	Estable	MSP	5.351	1.331	0	1.802	2.865	0.699	0.972
P_3	Estable	MSP	5.951	1.161	0	1.900	2.736	0.636	0.914
P_3	Estable	MSP	6.551	1.011	0	2.003	2.627	0.595	0.871

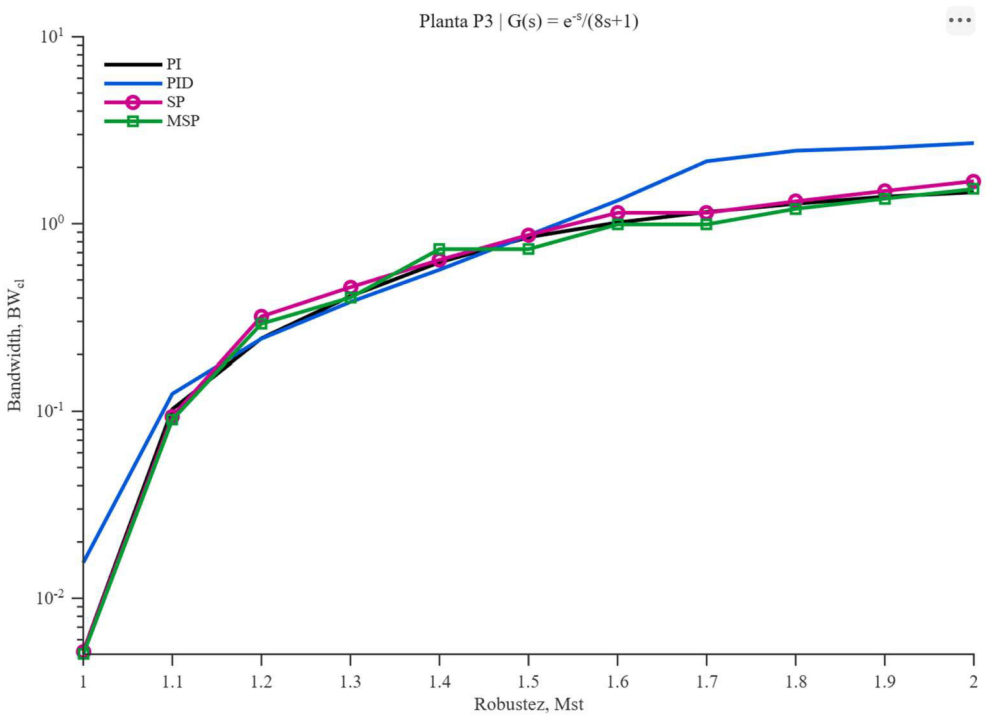

Figura 9-9. Ancho de banda en función de M_{ST} para P_3

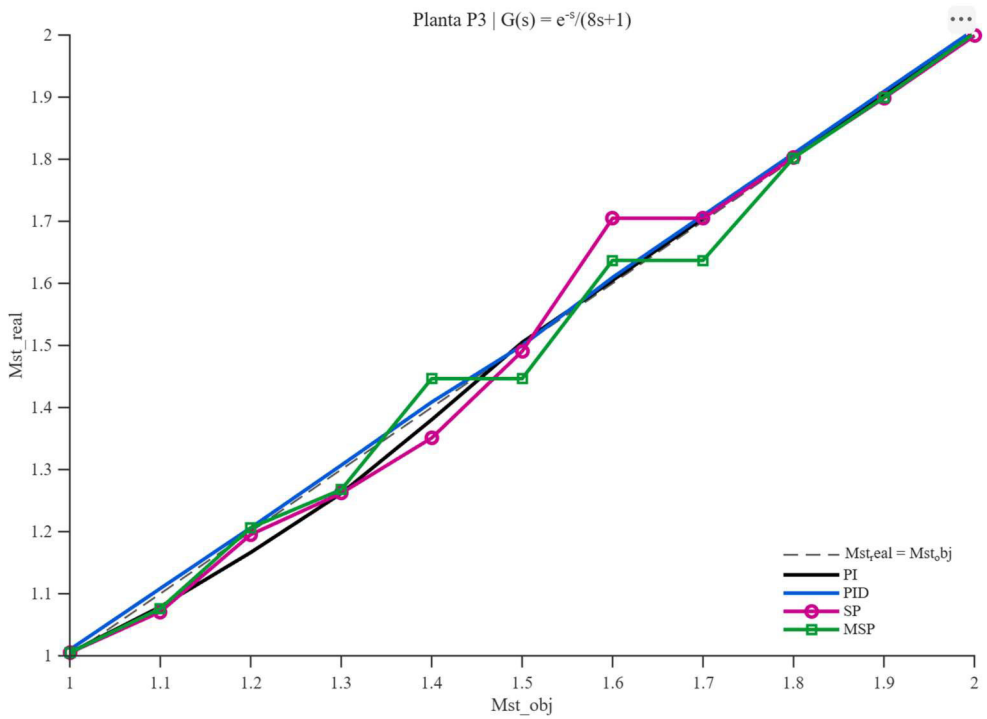

Figura 9-10. Comparación entre M_{ST} real y M_{ST} objetivo para P_3

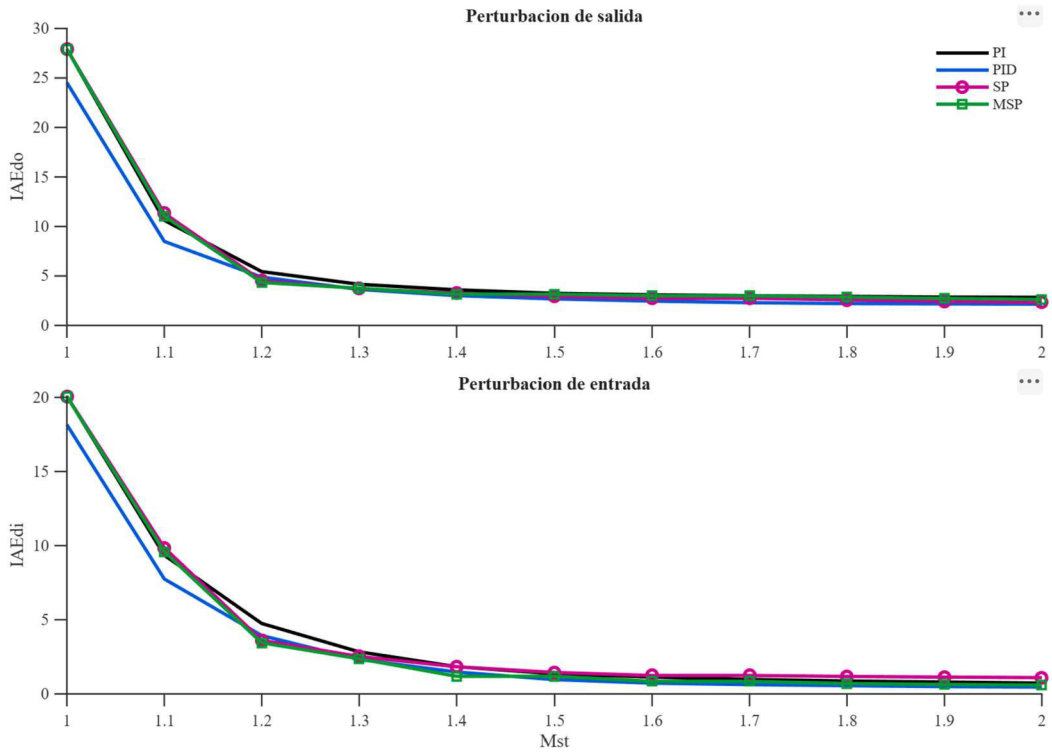

Figura 9-11. Índices IAE_{di} e IAE_{do} en función de M_{ST} para P_3

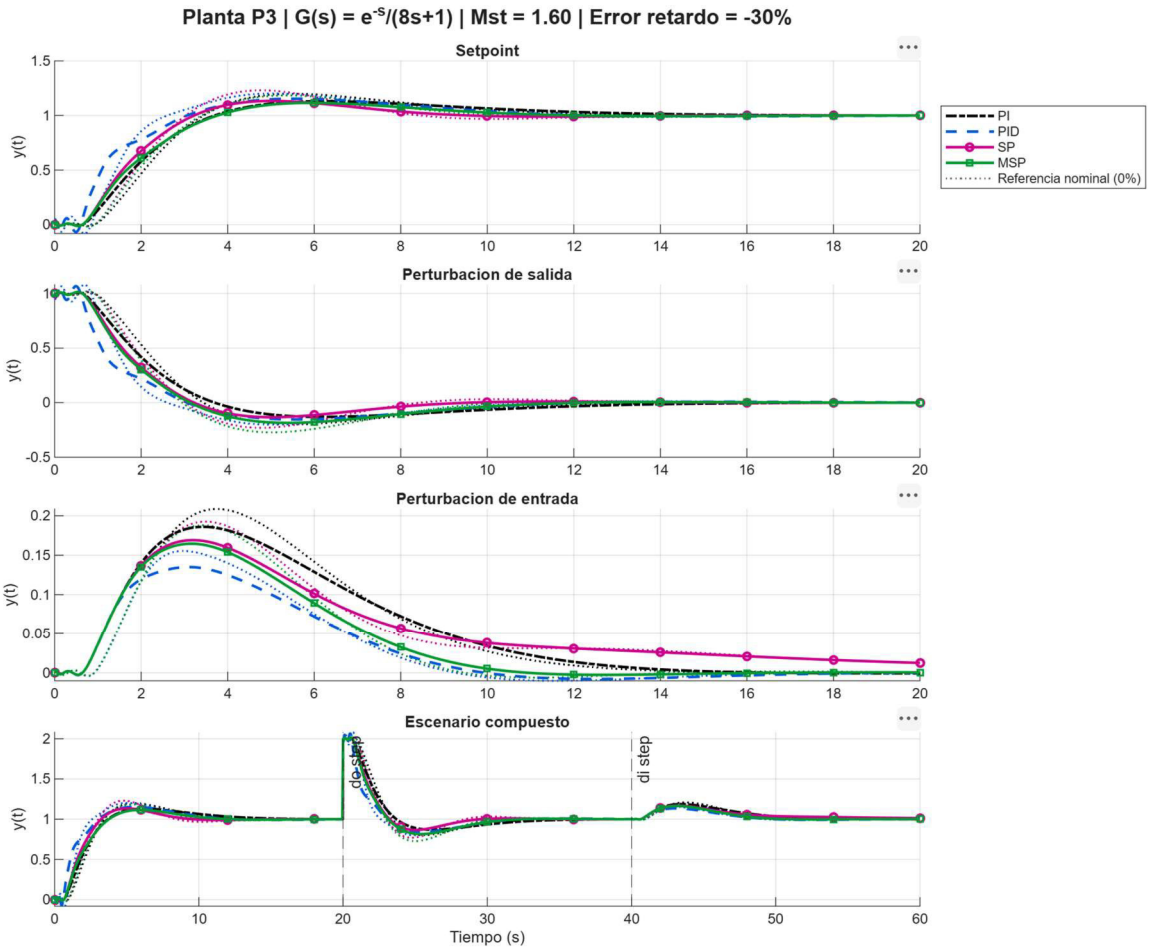

Figura 9-12. Respuestas controladas ante un error de modelado del tiempo de retardo de -30% para P_3 ($M_{ST} = 1.6$)

9.4.4. Planta 4

Tabla 9-4. Resultados de la búsqueda de sintonías que minimizan J para cada nivel objetivo de M_{ST} en P_4

Planta	Tipo	Controlador	k_c	τ_i	τ_d	M_{ST} real	IAE_{do}	IAE_{di}	J
P_4	Estable	PI	0.101	19.191	0	1.005	27.938	13.033	17.397
P_4	Estable	PI	0.701	18.301	0	1.033	18.637	9.839	12.566
P_4	Estable	PI	1.501	17.251	0	1.071	11.503	7.000	8.535
P_4	Estable	PI	2.501	16.001	0	1.121	7.141	4.826	5.702
P_4	Estable	PI	3.501	15.001	0	1.173	5.456	3.527	4.222
P_4	Estable	PI	5.001	14.001	0	1.257	4.212	2.430	3.013
P_4	Estable	PI	6.001	13.001	0	1.319	3.757	1.941	2.498
P_4	Estable	PI	7.001	12.001	0	1.386	3.458	1.577	2.123
P_4	Estable	PI	8.001	11.001	0	1.461	3.269	1.293	1.841
P_4	Estable	PI	9.001	10.001	0	1.544	3.155	1.065	1.623
P_4	Estable	PI	10.001	9.001	0	1.638	3.090	0.876	1.449
P_4	Estable	PID	0.281	16.221	0.54	1.010	24.215	11.799	15.501
P_4	Estable	PID	0.901	15.401	1.1	1.054	15.378	8.546	10.728
P_4	Estable	PID	1.501	14.601	1.6	1.130	10.373	6.389	7.761
P_4	Estable	PID	2.101	13.801	2.1	1.250	7.424	4.901	5.831
P_4	Estable	PID	2.701	13.001	2.46	1.409	5.824	3.871	4.596
P_4	Estable	PID	3.301	12.201	2.3	1.509	4.948	3.166	3.802
P_4	Estable	PID	3.901	11.401	2.14	1.606	4.375	2.623	3.213
P_4	Estable	PID	4.501	10.601	2	1.704	3.960	2.189	2.753
P_4	Estable	PID	5.101	9.801	1.88	1.805	3.638	1.834	2.380
P_4	Estable	PID	5.701	9.001	1.76	1.897	3.380	1.538	2.072
P_4	Estable	PID	6.301	8.181	1.66	1.995	3.153	1.284	1.807
P_4	Estable	SP	0.101	19.061	0	1.005	27.801	13.034	17.367
P_4	Estable	SP	0.701	18.201	0	1.032	18.237	9.936	12.556
P_4	Estable	SP	1.501	17.251	0	1.064	11.280	7.297	8.733
P_4	Estable	SP	2.501	16.001	0	1.102	6.959	5.315	6.070
P_4	Estable	SP	4.001	15.001	0	1.152	4.632	3.752	4.221
P_4	Estable	SP	5.001	14.001	0	1.184	3.876	3.111	3.508
P_4	Estable	SP	6.001	13.001	0	1.213	3.375	2.651	3.005
P_4	Estable	SP	19.991	1.661	0	1.704	1.251	0.843	0.997
P_4	Estable	SP	19.991	1.311	0	1.803	1.241	0.826	0.980

Planta	Tipo	Controlador	k_c	τ_i	τ_d	M_{ST} real	IAE_{do}	IAE_{di}	J
P_4	Estable	SP	19.991	1.081	0	1.905	1.235	0.814	0.969
P_4	Estable	SP	19.991	0.921	0	2.003	1.232	0.806	0.962
P_4	Estable	MSP	0.101	19.641	0	1.005	27.900	13.015	17.373
P_4	Estable	MSP	0.701	18.801	0	1.033	18.686	9.830	12.570
P_4	Estable	MSP	1.501	17.751	0	1.068	11.841	7.110	8.706
P_4	Estable	MSP	2.501	16.501	0	1.109	7.533	5.075	6.001
P_4	Estable	MSP	4.001	15.001	0	1.164	5.340	3.423	4.108
P_4	Estable	MSP	5.001	14.001	0	1.198	4.711	2.767	3.411
P_4	Estable	MSP	6.001	13.001	0	1.230	4.313	2.297	2.925
P_4	Estable	MSP	19.991	1.981	0	1.705	2.691	0.478	1.023
P_4	Estable	MSP	19.991	1.511	0	1.805	2.647	0.455	0.993
P_4	Estable	MSP	19.991	1.241	0	1.903	2.610	0.441	0.973
P_4	Estable	MSP	19.991	1.051	0	2.002	2.583	0.432	0.959

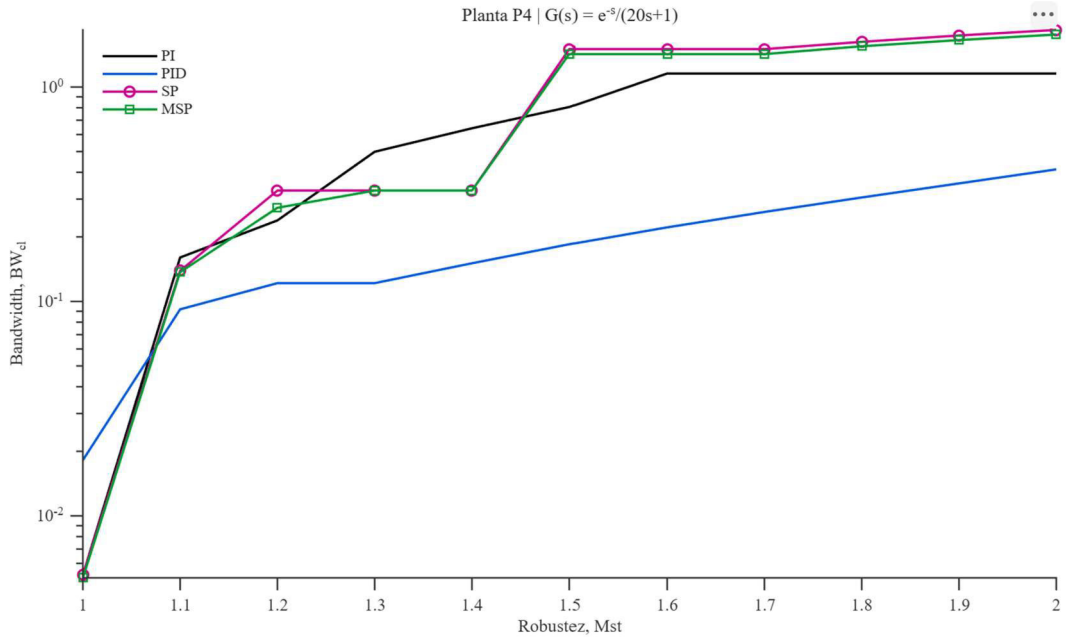

Figura 9-13. Ancho de banda en función de M_{ST} para P_4

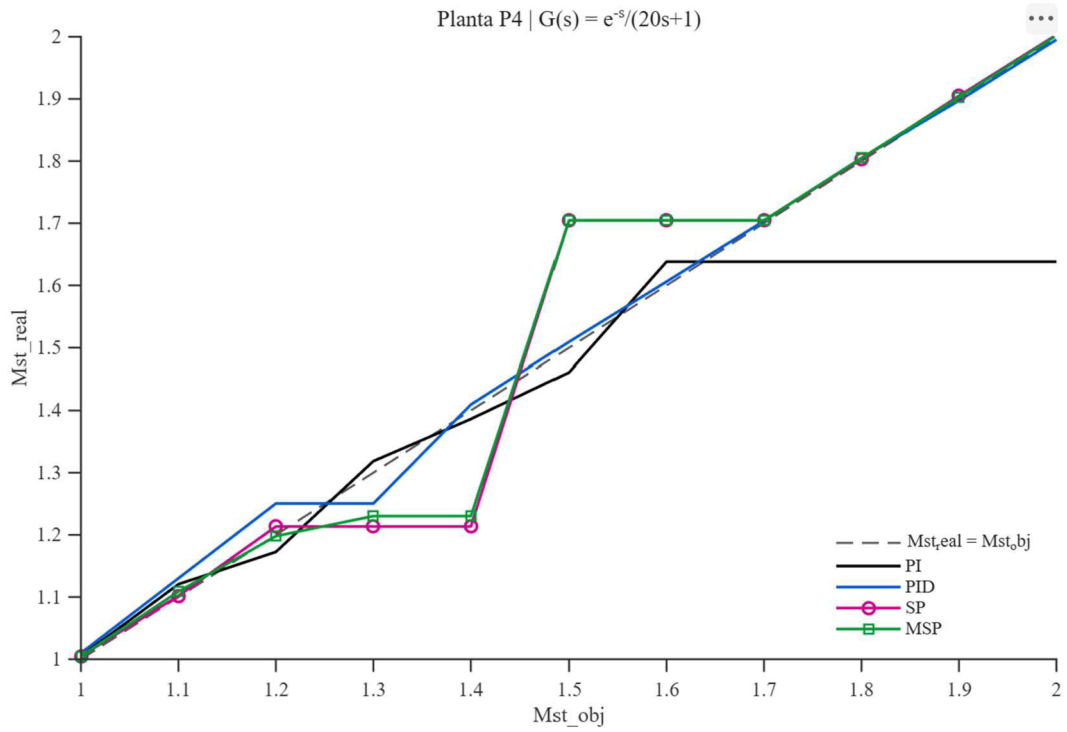

Figura 9-14. Comparación entre M_{ST} real y M_{ST} objetivo para P₄

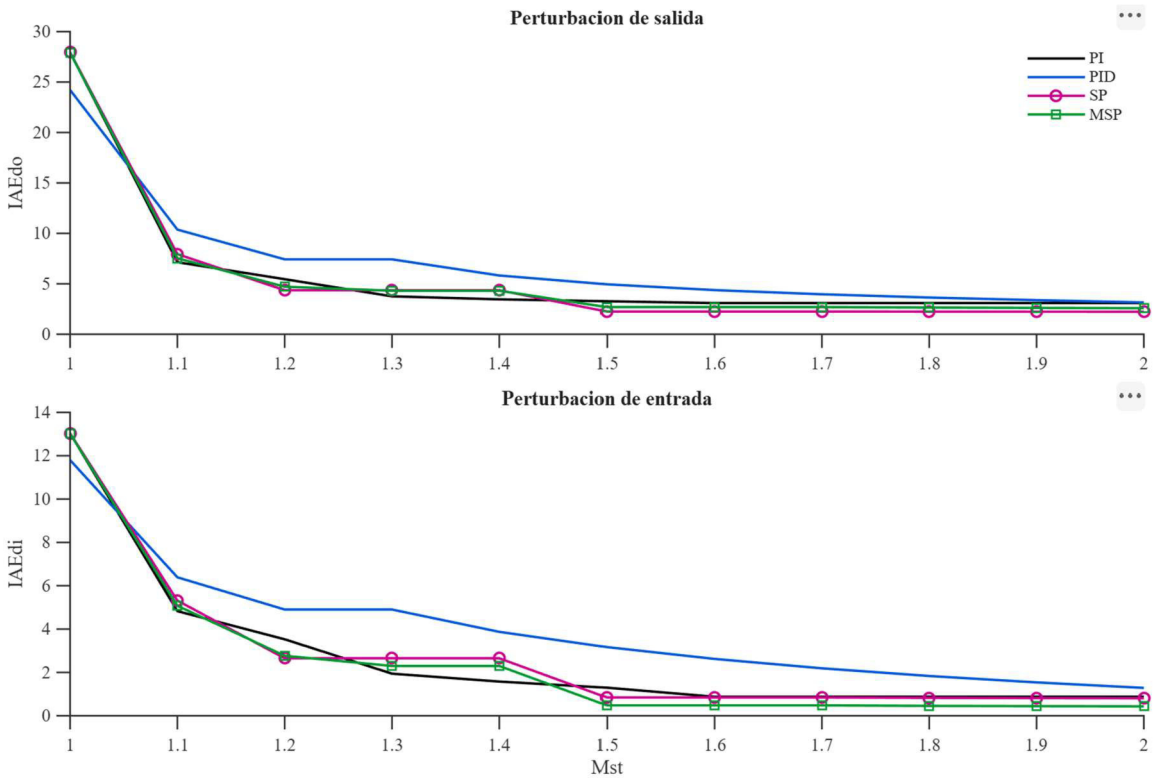

Figura 9-15. Índices IAE_{di} e IAE_{do} en función de M_{ST} para P₄

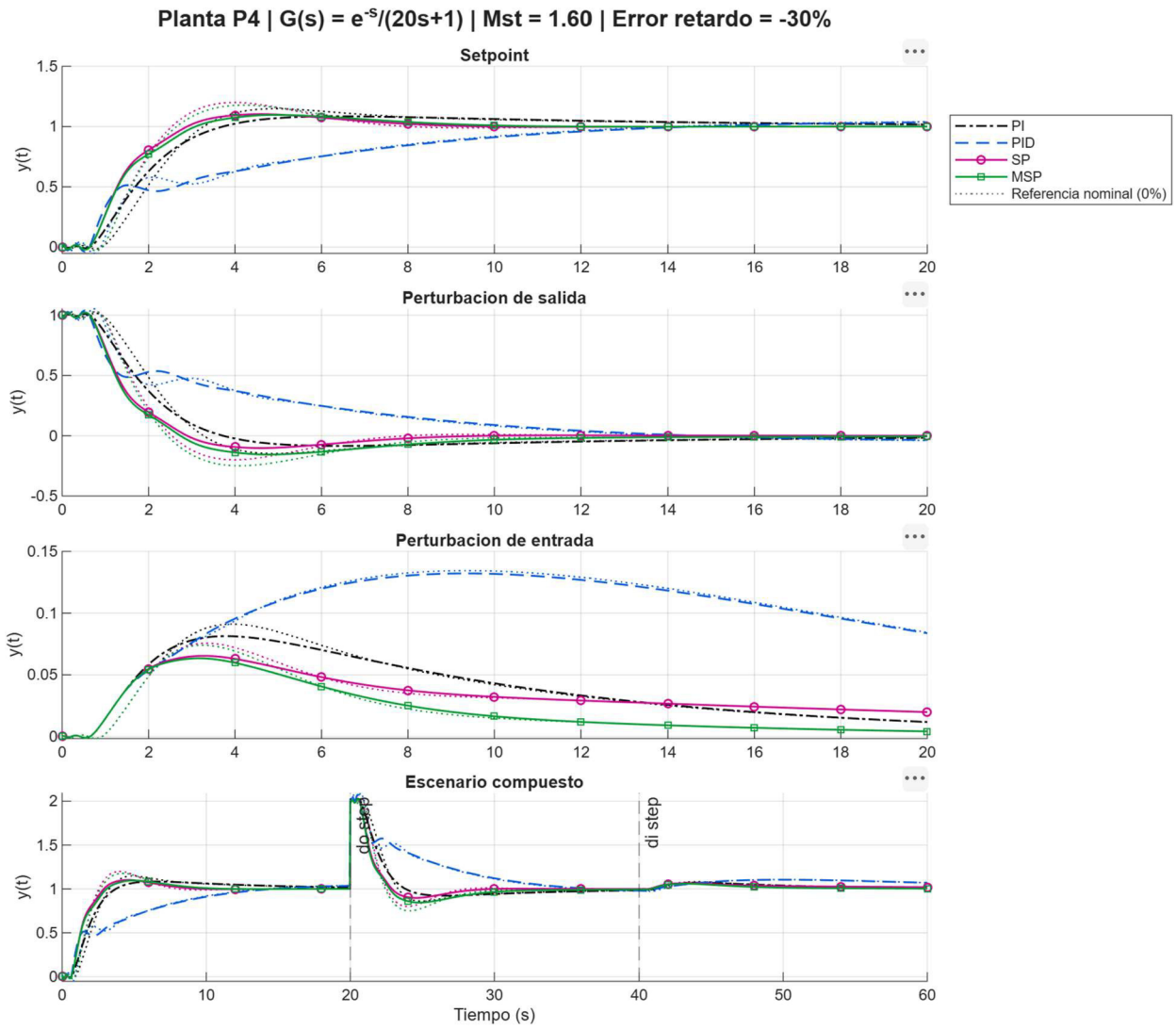

Figura 9-16. Respuestas controladas ante un error de modelado del tiempo de retardo de -30% para P₄ ($M_{ST} = 1.6$)

9.4.5. Planta 5

Tabla 9-5. Resultados de la búsqueda de sintonías que minimizan J para cada nivel objetivo de M_{ST} en P_5

Planta	Tipo	Controlador	k_c	τ_i	τ_d	M_{ST} real	IAE_{do}	IAE_{di}	J
P_5	Integrador	PI							
P_5	Integrador	PI							
P_5	Integrador	PI	0.191	19.051	0	1.202	6.894	82.934	2.221
P_5	Integrador	PI	0.111	19.241	0	1.299	9.001	140.149	3.392
P_5	Integrador	PI	0.351	19.801	0	1.396	4.498	45.124	1.310
P_5	Integrador	PI	0.421	18.991	0	1.499	4.015	36.623	1.113
P_5	Integrador	PI	0.481	18.181	0	1.598	3.737	31.183	0.991
P_5	Integrador	PI	0.541	18.601	0	1.705	3.499	28.024	0.909
P_5	Integrador	PI	0.591	18.701	0	1.805	3.352	25.692	0.853
P_5	Integrador	PI	0.631	18.641	0	1.893	3.260	23.995	0.815
P_5	Integrador	PI	0.671	17.831	0	1.992	3.200	21.966	0.775
P_5	Integrador	PID							
P_5	Integrador	PID	0.201	16.001	0.7	1.199	6.314	70.993	1.957
P_5	Integrador	PID	0.261	15.201	0.44	1.209	5.439	52.595	1.553
P_5	Integrador	PID	0.361	14.401	0.42	1.309	4.256	36.107	1.137
P_5	Integrador	PID	0.341	13.601	0.74	1.401	4.251	36.761	1.147
P_5	Integrador	PID	0.521	12.801	0.42	1.509	3.204	22.716	0.787
P_5	Integrador	PID	0.581	12.001	0.26	1.602	3.177	19.360	0.732
P_5	Integrador	PID	0.641	11.201	0.42	1.704	2.795	16.569	0.637
P_5	Integrador	PID	0.661	10.801	0.5	1.810	2.654	15.586	0.602
P_5	Integrador	PID	0.741	10.001	0.42	1.910	2.623	13.018	0.558
P_5	Integrador	PID	0.681	9.181	0.6	2.009	2.538	13.166	0.549
P_5	Integrador	SP							
P_5	Integrador	SP	0.201	14.601	0	1.200	6.145	94.526	2.298
P_5	Integrador	SP	0.261	13.801	0	1.203	5.095	77.058	1.885
P_5	Integrador	SP	0.451	13.001	0	1.302	3.255	54.801	1.290
P_5	Integrador	SP	0.701	12.721	0	1.405	2.222	44.963	0.998
P_5	Integrador	SP	1.011	11.921	0	1.505	1.608	39.321	0.827
P_5	Integrador	SP	1.421	11.121	0	1.604	1.184	35.778	0.714
P_5	Integrador	SP	2.021	10.361	0	1.705	0.858	33.331	0.632
P_5	Integrador	SP	2.651	9.551	0	1.777	0.668	31.945	0.585
P_5	Integrador	SP	3.301	8.701	0	1.830	0.544	31.058	0.554
P_5	Integrador	SP	4.001	7.751	0	1.871	0.454	30.410	0.532
P_5	Integrador	MSP							

Planta	Tipo	Controlador	k_c	τ_i	τ_d	M_{ST} real	IAE_{do}	IAE_{di}	J
P_5	Integrador	MSP	0.251	10.751	0	1.260	5.999	45.932	1.527
P_5	Integrador	MSP	0.301	9.901	0	1.297	5.348	35.880	1.282
P_5	Integrador	MSP	0.301	9.101	0	1.300	5.351	33.527	1.246
P_5	Integrador	MSP	0.431	8.301	0	1.401	4.306	22.145	0.928
P_5	Integrador	MSP	0.581	7.501	0	1.502	3.693	15.776	0.746
P_5	Integrador	MSP	0.761	6.851	0	1.605	3.284	11.886	0.631
P_5	Integrador	MSP	0.961	6.051	0	1.704	3.011	9.202	0.552
P_5	Integrador	MSP	1.201	5.251	0	1.805	2.804	7.287	0.494
P_5	Integrador	MSP	1.491	4.541	0	1.905	2.645	5.964	0.452
P_5	Integrador	MSP	1.861	3.981	0	2.005	2.516	5.058	0.421

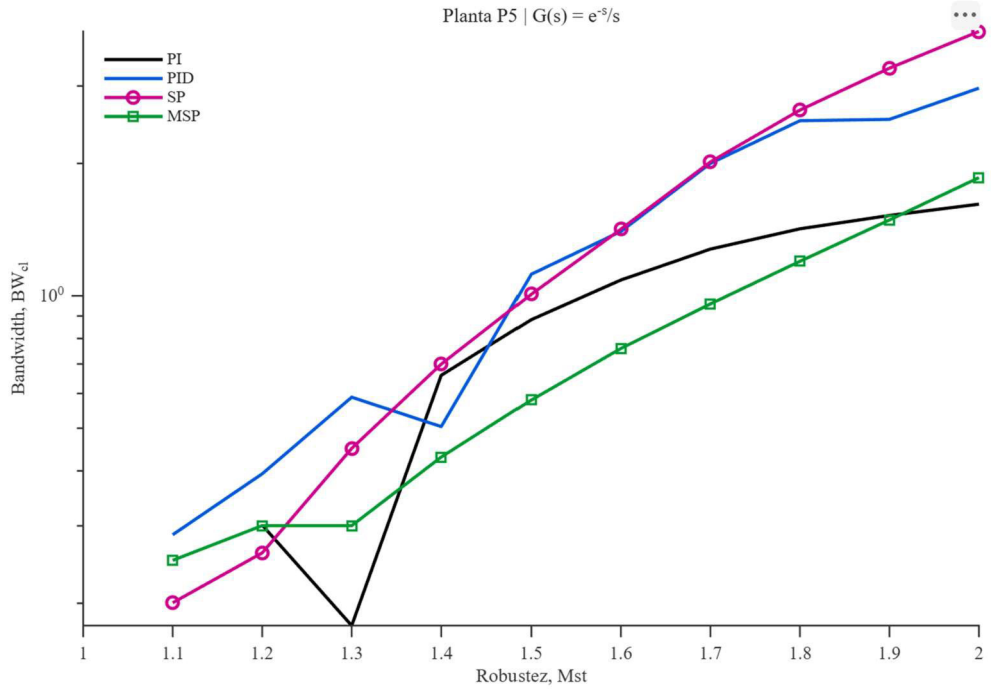

Figura 9-17. Ancho de banda en función de M_{ST} para P_5

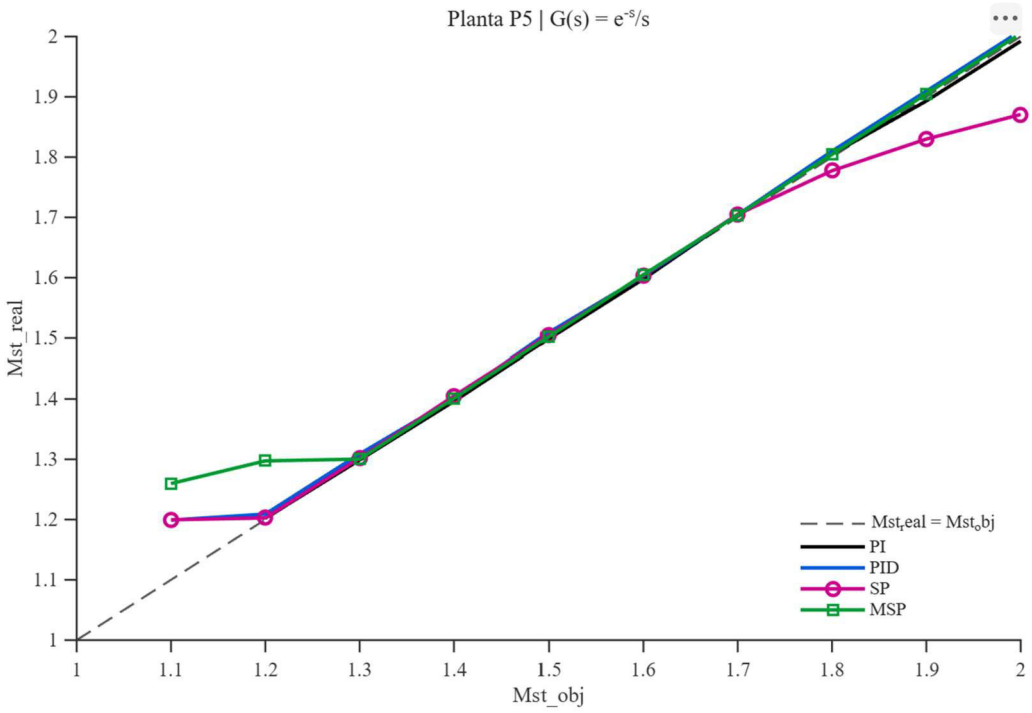

Figura 9-18. Comparación entre M_{ST} real y M_{ST} objetivo para P_5

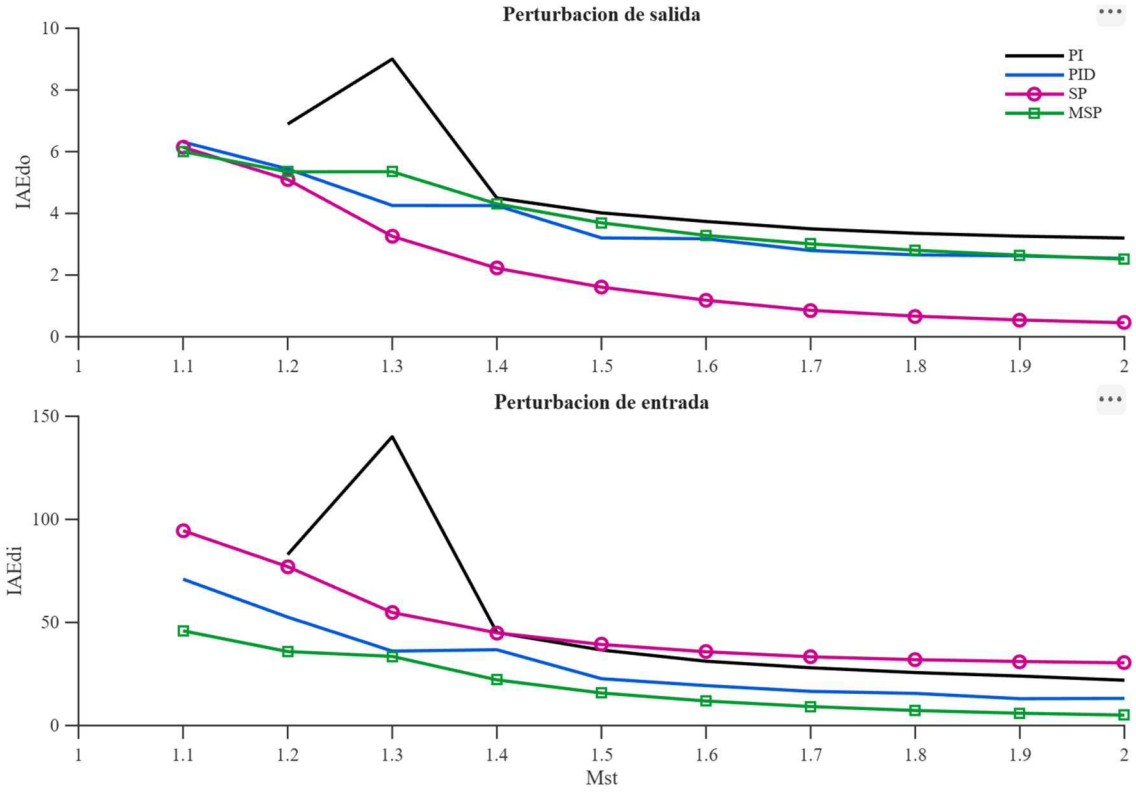

Figura 9-19. Índices IAE_{di} e IAE_{do} en función de M_{ST} para P_5

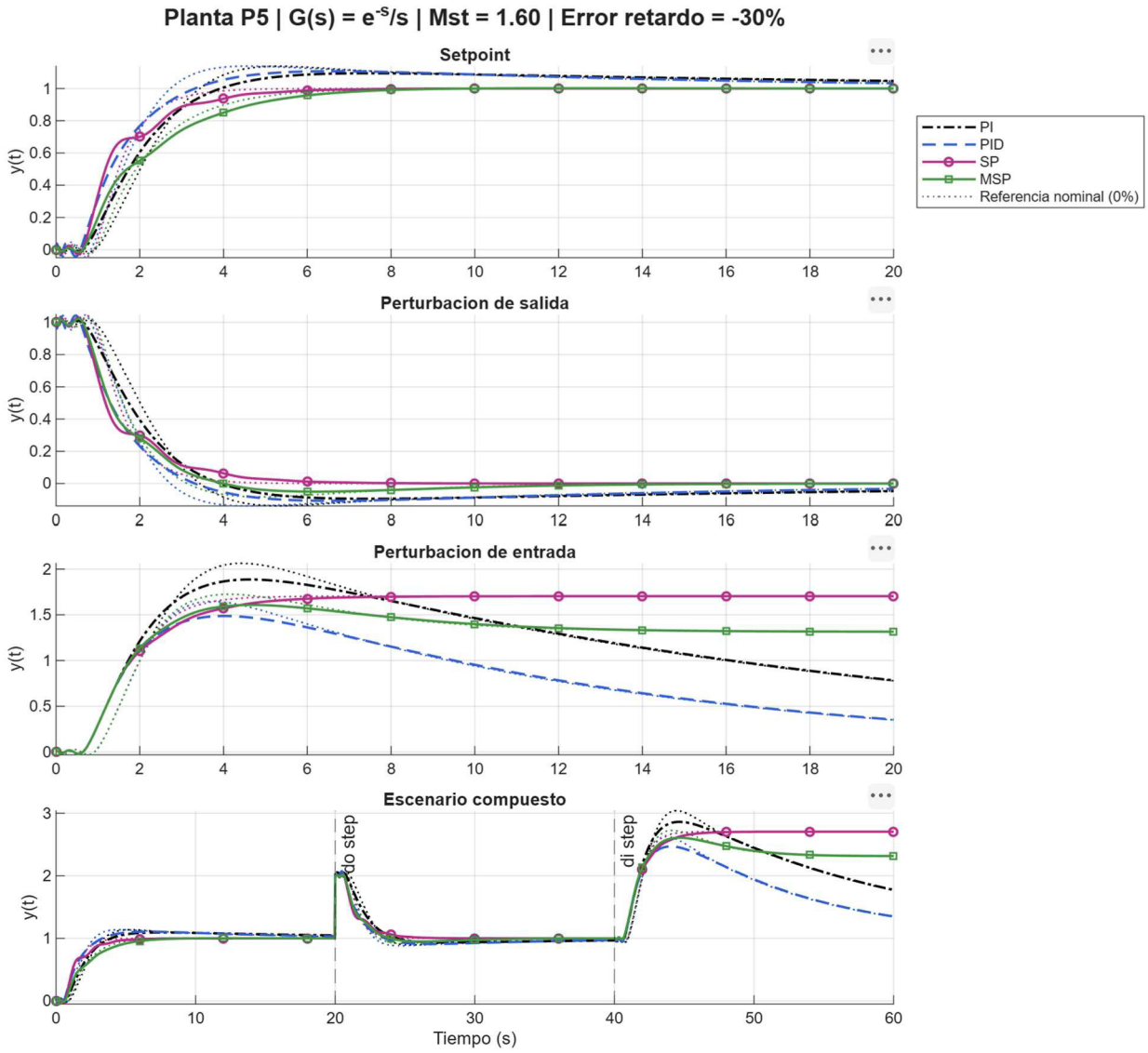

Figura 9-20. Respuestas controladas ante un error de modelado del tiempo de retardo de -30% para P_5 ($M_{ST} = 1.6$)

9.4.6. Planta 7

Tabla 9-6. Resultados de la búsqueda de sintonías que minimizan J para cada nivel objetivo de M_{ST} en P_7

Planta	Tipo	Controlador	k_c	τ_i	τ_d	M_{ST} real	IAE_{do}	IAE_{di}	J
P_7	Inestable	PI							
P_7	Inestable	PI							
P_7	Inestable	PI							
P_7	Inestable	PI							
P_7	Inestable	PI	4.001	14.001	0	1.793	7.136	3.397	0.716
P_7	Inestable	PI	3.501	13.001	0	1.723	7.721	3.684	0.775
P_7	Inestable	PI	3.751	13.001	0	1.744	7.304	3.417	0.727
P_7	Inestable	PI	3.571	13.081	0	1.705	7.616	3.625	0.764
P_7	Inestable	PI	3.961	12.281	0	1.804	6.851	3.061	0.668
P_7	Inestable	PI	4.291	11.481	0	1.903	6.298	2.644	0.598
P_7	Inestable	PI	4.581	10.681	0	2.003	5.864	2.309	0.542
P_7	Inestable	PID							
P_7	Inestable	PID							
P_7	Inestable	PID							
P_7	Inestable	PID	4.001	15.501	0.5	1.528	6.688	3.697	0.720
P_7	Inestable	PID	4.001	15.901	0.4	1.500	6.879	3.770	0.737
P_7	Inestable	PID	4.041	15.101	0.42	1.510	6.664	3.580	0.707
P_7	Inestable	PID	4.561	14.301	0.4	1.607	5.902	2.997	0.610
P_7	Inestable	PID	5.041	13.501	0.38	1.708	5.361	2.565	0.539
P_7	Inestable	PID	5.461	12.701	0.38	1.809	4.945	2.238	0.485
P_7	Inestable	PID	5.821	11.901	0.38	1.905	4.626	1.980	0.443
P_7	Inestable	PID	6.161	11.101	0.38	2.007	4.354	1.757	0.406
P_7	Inestable	SP							
P_7	Inestable	SP	6.001	6.001	0	1.498	2.789	35.277	3.522
P_7	Inestable	SP	7.001	5.001	0	1.550	2.263	34.991	3.466
P_7	Inestable	SP	6.501	5.001	0	1.534	2.424	35.046	3.480
P_7	Inestable	SP	5.901	5.701	0	1.500	2.781	35.243	3.518
P_7	Inestable	SP	5.931	5.461	0	1.505	2.728	35.197	3.511
P_7	Inestable	SP	6.601	4.601	0	1.547	2.327	34.973	3.467
P_7	Inestable	SP	7.251	3.751	0	1.594	2.005	34.794	3.433
P_7	Inestable	SP	8.001	2.751	0	1.673	1.698	34.620	3.399
P_7	Inestable	SP	8.751	1.751	0	1.830	1.449	34.477	3.372
P_7	Inestable	SP	9.351	1.231	0	2.002	1.307	34.408	3.357
P_7	Inestable	MSP							

Planta	Tipo	Controlador	k_c	τ_i	τ_d	M_{ST} real	IAE_{do}	IAE_{di}	J
P_7	Inestable	MSP							
P_7	Inestable	MSP	6.001	6.001	0	1.588	6.858	3.842	0.743
P_7	Inestable	MSP	7.001	5.001	0	1.647	6.342	3.529	0.685
P_7	Inestable	MSP	7.001	5.501	0	1.635	6.433	3.608	0.697
P_7	Inestable	MSP	6.401	6.201	0	1.600	6.751	3.808	0.734
P_7	Inestable	MSP	6.121	5.411	0	1.605	6.690	3.714	0.722
P_7	Inestable	MSP	6.801	4.601	0	1.652	6.320	3.487	0.680
P_7	Inestable	MSP	7.401	3.801	0	1.703	6.029	3.309	0.647
P_7	Inestable	MSP	8.001	3.001	0	1.771	5.774	3.158	0.618
P_7	Inestable	MSP	7.301	2.101	0	1.901	5.704	3.063	0.605

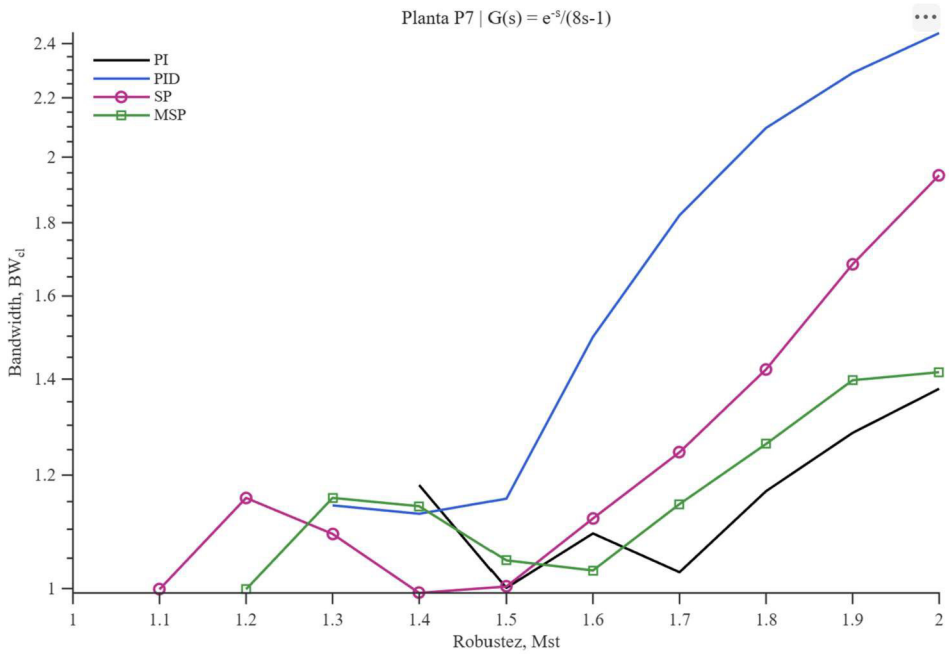

Figura 9-21. Ancho de banda en función de M_{ST} para P_7

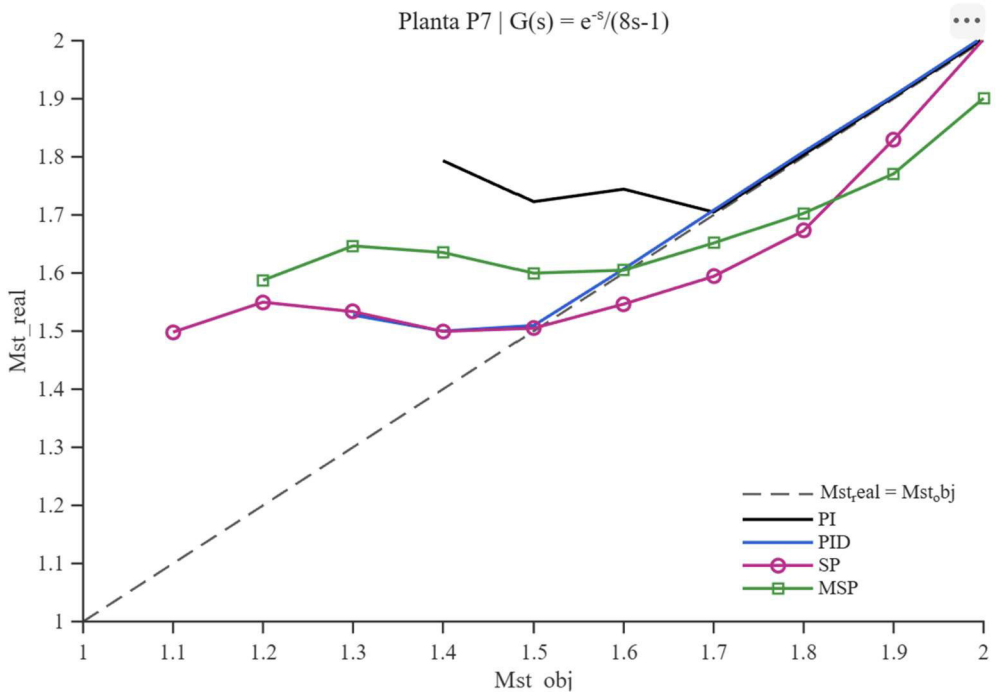

Figura 9-22. Comparación entre M_{ST} real y M_{ST} objetivo para P$_7$

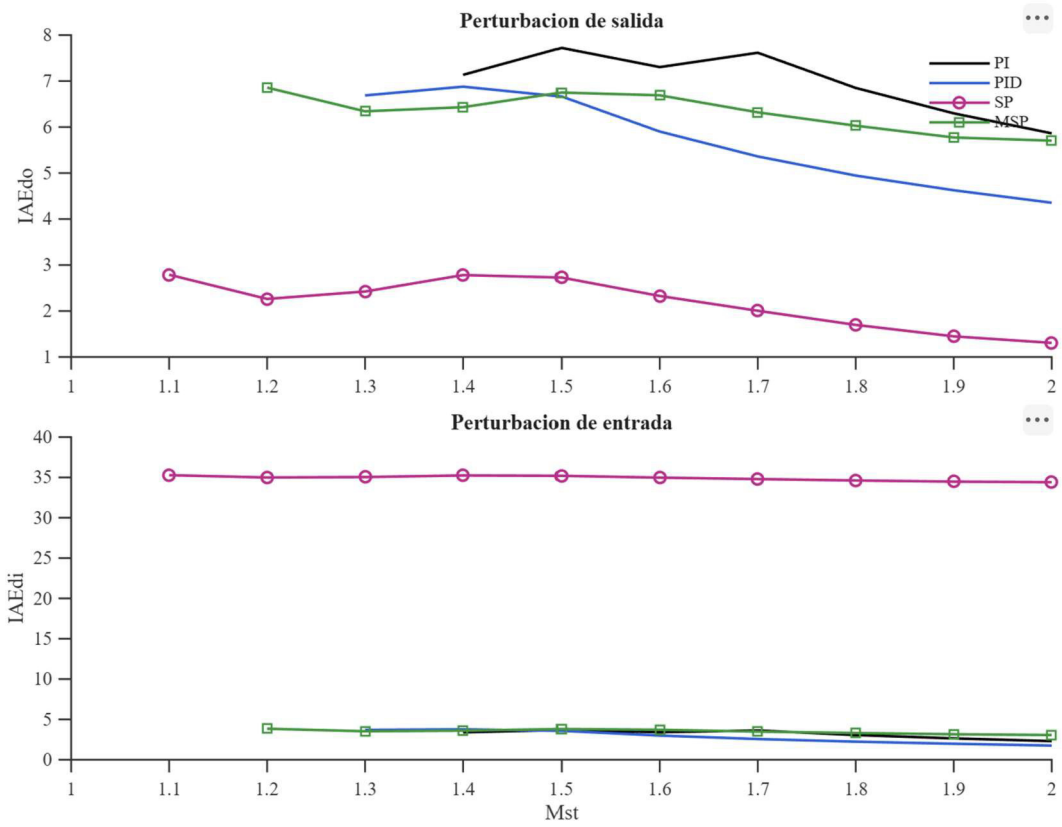

Figura 9-23. Índices IAE_{di} e IAE_{do} en función de M_{ST} para P$_7$

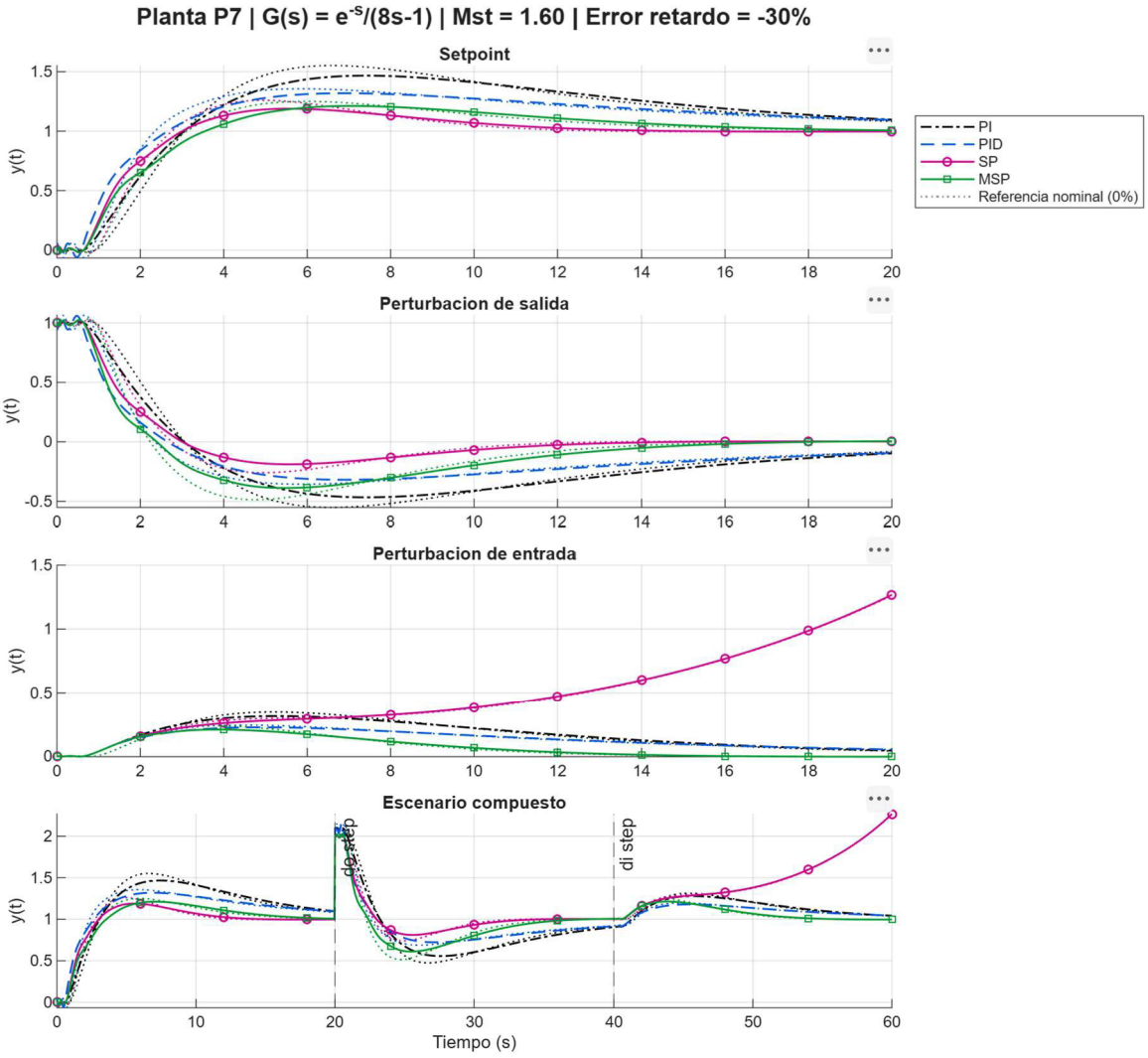

Figura 9-24. Respuestas controladas ante un error de modelado del tiempo de retardo de +30% para P_7 ($M_{ST} = 1.6$)

9.5. Anexo 5. Código MATLAB del Programa de evaluación de estrategias de control

```
%% ========================================================================
%  PROGRAMA DE EVALUACION DE ESTRATEGIAS DE CONTROL
%  Fecha: 2026-06-29
%  Revision: 0
%  Autor: PAUL MARTIN SAAVEDRA CARRASCO
%
%  Descripcion general:
%  - Ejecuta el barrido principal J vs Mst para PI, PID, SP y MSP.
%  - Calcula pesos base, indicadores de robustez y tablas de resultados.
%  - Genera figuras y archivos de exportacion para documentacion de tesis.
%  - Permite reutilizar resultados del barrido en la etapa temporal a Mst fijo.
%% ========================================================================

%% DESCRIPCION DEL ARCHIVO
%  El archivo se organiza en dos bloques:
%  1) Script principal: configuracion, busqueda, reportes, graficas y exportacion.
%  2) Funciones locales: rutinas auxiliares para barrido, evaluacion y postproceso.
%
%  Lectura sugerida en anexo:
%  - Revisar primero el indice funcional y la notacion principal.
%  - Asociar cada seccion del script con la metodologia descrita en la tesis.
%  - Consultar las funciones locales como soporte de implementacion.

%% INDICE FUNCIONAL
%  0. Configuracion general y parametros de presentacion
%  1. Pesos base por planta
%  2. Barrido principal J vs Mst_obj
%  3. Reporte de busqueda
%  4. Graficas de resultados del barrido
%  5. Exportacion para tesis
%  6. Evaluacion temporal a Mst fijo
%  7. Funciones locales

%% INDICE DE FUNCIONES
%  Las funciones locales se presentan al final del archivo para conservar
%  una implementacion autocontenida y facilitar la trazabilidad del flujo.
%
%  Funciones principales del barrido:
%  build_mst_grid()                        % Construye el barrido principal en Mst
%  build_aligned_axis()                    % Alinea las mallas de busqueda al paso usado
%  run_mst_sweep()                         % Ejecuta el barrido J vs Mst para un controlador
%  build_warm_bounds()                     % Crea una region local desde la solucion previa
%  global_mst_map()                        % Hace el scan global del espacio de parametros
%  select_bounds_from_scan()               % Elige regiones prometedoras desde el scan
%  detect_scan_regions()                   % Agrupa puntos vecinos del scan global
%  refinement_over_bounds_list()           % Refina una o varias regiones seleccionadas
%  refinement_search()                     % Refinamiento iterativo dentro de una region
%  evaluate_candidate_iae()                % Evalua IAE de todos los candidatos
%  build_grid_and_eval()                   % Construye una malla y evalua estabilidad y Mst
%  eval_mst_controller()                   % Evalua estabilidad y robustez para un candidato
%  exact_mst_classic()                     % Mst exacto para PI y PID clasico
%  exact_mst_sp()                          % Mst exacto para SP
%  exact_mst_msp()                         % Mst exacto para MSP
%  sp_robustness_tf()                      % S y T equivalentes con Pade para SP
%  msp_robustness_tf()                     % S y T equivalentes con Pade para MSP
%  eval_iae_controller()                   % IAE ante perturbaciones de salida y entrada
%  build_base_result()                     % Construye IAEdo0_base e IAEdi0_base
%  build_neutral_base_result()             % Usa pesos neutros cuando no hay base valida
%  build_opt_result_with_external_weights()% Elige la mejor solucion para cada Mst_obj
%  continuity_cost()                       % Penaliza cambios bruscos entre soluciones vecinas
%  print_sweep_table()                     % Imprime la tabla resumen del barrido
%  print_iter_log()                        % Imprime el log de iteraciones
%  print_iter_status_header()              % Imprime el encabezado compacto por region
%  print_iter_status_compact()             % Imprime el estado por iteracion en formato
compacto
```

```
%  print_eval_mst_distribution_from_iter_log() % Resume el Mst evaluado por iteracion
%  build_mst_edges()                        % Construye bandas de histograma para Mst
%  get_thesis_plot_styles()                 % Define colores y marcadores de tesis
%  compute_clean_j_limits()                 % Calcula limites de J con cap visual
%  plot_sweep_curve_clean()                 % Dibuja una curva J vs Mst
%  build_base_export_table()                % Arma CSV de pesos base
%  build_sweep_export_table()               % Arma CSV resumen del barrido
%  build_extraordinary_export_table()       % Arma CSV de casos extraordinarios
%  build_metadata_lines()                   % Arma metadata de la corrida
%  write_text_lines()                       % Escribe archivos de texto
%  smith_tf()                               % Transferencias equivalentes de Smith Predictor
%  msp_tf()                                 % Transferencias equivalentes de MSP
%  build_Fp_msp()                           % Construye Fp para MSP segun el tipo de planta
%  get_Tplant_by_id()                       % Recupera la constante de tiempo base
%  parse_lista_numerica()                   % Interpreta selecciones numericas del usuario
%  ensure_parallel_pool_quiet()             % Inicia el pool en silencio para no ensuciar la
salida
%  Utilidades: fmt_num, fmt_rng, vec2str, clip_str, ternary_txt, etc.
%
%  Funciones de la etapa temporal a Mst fijo:
%  build_fixed_pick_table_td()              % Selecciona el controlador por Mst fijo
%  build_time_metrics_table_td()            % Calcula indicadores temporales
%  build_deadtime_error_table_td()          % Evalua error en el tiempo de retardo
%  plot_time_domain_plant_figure_td()       % Grafica respuestas temporales por planta
%  plot_deadtime_error_figure_td()          % Grafica sensibilidad al error de retardo

%% NOTACION PRINCIPAL
%  Mst_base  : robustez usada para obtener los pesos base
%  Mst_obj   : robustez deseada en el barrido principal
%  Mst_real  : robustez realmente obtenida
%  Tol_Mst   : tolerancia tomada desde h_used de la ultima iteracion valida
%  IAEdo0    : peso base para perturbacion de salida
%  IAEdi0    : peso base para perturbacion de entrada
%  J         : 0.5 * (IAEdo / IAEdo0 + IAEdi / IAEdi0)

clc; clear; close all;
format compact;
s = tf('s'); %#ok<NASGU>

script_fullpath = mfilename('fullpath');
if isempty(script_fullpath)
    project_root = pwd;
else
    project_root = fileparts(script_fullpath);
end

%% ====================== 0. CONFIGURACION GENERAL ======================
code_version = 'Codigo_tesis_consolidado_v1';

Mst_base = 1.59;
pade_order = 4;

% PID
N_der  = 10;
Tf_min = 1e-3;

% Refinamiento general
refinamientos  = [1, 0.5, 0.25, 0.10, 0.05, 0.02, 0.01];
h_1_vals       = [0.20, 0.12, 0.08, 0.05, 0.02, 0.01, 0.005];

% Refinamiento PID
refinamientos_pid = [1, 0.5, 0.25, 0.10, 0.05, 0.02];
h_1_vals_pid      = [0.20, 0.12, 0.08, 0.05, 0.02, 0.01];

edge_mult      = 2;
```

```
h_relax_factor = 2;

% Scan global de Mst
scan_paso_global = 0.2;
region_gap_mult  = 1.5;
max_regions_keep = 4;
region_min_pts   = 2;
margen_mult      = 2;

% Stop temprano
max_empty_iters = 4;

% IAE
T_end = 30;
dt    = 0.01;
t     = 0:dt:T_end;

% Disturbancias
Go = 1;
Gi = 1;

% Bounds globales
kc_bounds_global = [1e-3 20];
ti_bounds_global = [1e-3 20];
td_bounds_global = [0 5];

% La tolerancia final de Mst se toma de h_used en la ultima iteracion valida

% Mst exacto
use_exact_mst = true;
w_mst = logspace(-3, 3, 4000);

% Guardado de detalle
save_eval_detail = true;

%% ===================== 0.1. PRESENTACION / TESIS =====================
thesis_mode         = true;
verbose_status      = false;
show_iter_logs      = false;
show_iter_status    = true;
compact_console     = true;
export_pub_figs     = true;
export_results_csv  = false;
export_metadata_txt = true;
export_results_xlsx = true;
save_run_workspace  = true;
csv_delimiter       = ';';
print_export_tables = true;
print_search_report = true;
export_annex_reports = true;
print_sweep_tables = false;
export_final_report_txt = true;
report_mst_ref      = 1.6;
prestart_parallel_pool = true;
parallel_profile       = 'Processes';

% Etapa consolidada de Mst fijo / dominio del tiempo
run_time_eval_stage    = true;
ask_mst_ref_eval       = true;
mst_ref_eval           = 1.6;
allow_far_mst_eval     = false;
settling_p_time        = 2;
export_time_eval_figs  = true;
export_time_eval_tables = true;
deadtime_err_list_time = [-0.30 -0.20 -0.10 0 0.10 0.20 0.30];
```

```
194
195 thesis_out_root = fullfile(project_root, 'salida_tesis');
196 pub_fig_folder  = fullfile(thesis_out_root, 'figuras');
197 csv_out_folder  = fullfile(thesis_out_root, 'tablas');
198 meta_out_folder = fullfile(thesis_out_root, 'meta');
199 annex_out_folder = fullfile(thesis_out_root, 'anexos_busqueda');
200 run_stamp       = datestr(now, 'yyyymmdd_HHMMSS');
201 xlsx_book_file  = fullfile(csv_out_folder, sprintf('resultados_corrida_%s.xlsx', run_stamp));
202 annex_xlsx_file = fullfile(annex_out_folder, sprintf('anexo_busqueda_por_planta_%s.xlsx',
    run_stamp));
203 final_report_file = fullfile(annex_out_folder, sprintf('reporte_final_busqueda_%s.txt',
    run_stamp));
204 workspace_mat_file = fullfile(meta_out_folder, sprintf('workspace_corrida_%s.mat',
    run_stamp));
205
206 fig_w_cm         = 13.5;
207 fig_h_cm         = 8.2;
208 fig_res_dpi      = 300;
209 font_name_axes   = 'Times New Roman';
210 font_name_title  = 'Times New Roman';
211 font_size_axes   = 9.5;
212 font_size_title  = 10.5;
213 font_size_legend = 8.5;
214 line_w           = 1.80;
215 marker_sz        = 6.0;
216 grid_alpha_major = 0.14;
217 grid_alpha_minor = 0.07;
218
219 plot_show_title      = false;
220 plot_title_show_base = false;
221 legend_location_single = 'northeast';
222 legend_location_shared = 'southoutside';
223 legend_num_cols      = 4;
224
225 % Cap visual en J
226 plot_j_use_cap  = true;
227 plot_j_hard_cap = 50;
228 plot_j_qref     = 95;
229 plot_j_hide_above = 10000;
230 plot_j_use_zoom_inset = false;
231
232 if export_pub_figs && ~exist(pub_fig_folder,'dir')
233     mkdir(pub_fig_folder);
234 end
235 if (export_results_csv || export_results_xlsx) && ~exist(csv_out_folder,'dir')
236     mkdir(csv_out_folder);
237 end
238 if (export_metadata_txt || save_run_workspace) && ~exist(meta_out_folder,'dir')
239     mkdir(meta_out_folder);
240 end
241 if export_annex_reports && ~exist(annex_out_folder,'dir')
242     mkdir(annex_out_folder);
243 end
244
245 global verbose_status_global show_iter_logs_global show_iter_status_global thesis_mode_global
    compact_console_global
246 verbose_status_global = verbose_status;
247 show_iter_logs_global = show_iter_logs;
248 show_iter_status_global = show_iter_status;
249 thesis_mode_global    = thesis_mode;
250 compact_console_global = compact_console;
251
252 ensure_parallel_pool_quiet(prestart_parallel_pool, parallel_profile); % Inicia el pool antes
    del reporte para evitar mensajes en medio de la salida
253
```

```
%% ==================== 0.2. DEFINICION DE PLANTAS ====================
Plantas = struct([]);

Plantas(1).id='P1'; Plantas(1).tipo='estable';    Plantas(1).P_dt=tf(1,1,'InputDelay',1);
Plantas(2).id='P2'; Plantas(2).tipo='estable';    Plantas(2).P_dt=tf(1,[1 1],'InputDelay',1);
Plantas(3).id='P3'; Plantas(3).tipo='estable';    Plantas(3).P_dt=tf(1,[8 1],'InputDelay',1);
Plantas(4).id='P4'; Plantas(4).tipo='estable';    Plantas(4).P_dt=tf(1,[20 1],'InputDelay',1);
Plantas(5).id='P5'; Plantas(5).tipo='integrador'; Plantas(5).P_dt=tf(1,[1 0],'InputDelay',1);
Plantas(6).id='P6'; Plantas(6).tipo='inestable';  Plantas(6).P_dt=tf(1,[1 -1],'InputDelay',1);
Plantas(7).id='P7'; Plantas(7).tipo='inestable';  Plantas(7).P_dt=tf(1,[8 -1],'InputDelay',1);

%% ==================== 0.3. ENTRADAS DE USUARIO ====================
fprintf('\n=== SELECCION DE PLANTAS A SIMULAR ===\n');
fprintf('  0) Todas las plantas\n');
for ip = 1:numel(Plantas)
    fprintf('  %d) %s (%s)\n', ip, Plantas(ip).id, Plantas(ip).tipo);
end
modoP = input('Ingresa 0 para todas, o uno o varios numeros separados por coma: ','s');
modoP = strtrim(modoP);
if isempty(modoP), modoP = '0'; end

selP = parse_lista_numerica(modoP, 0:numel(Plantas));
if ismember(0, selP)
    if numel(selP) > 1
        error('La opcion 0 para plantas debe usarse sola.');
    end
    idx_run = 1:numel(Plantas);
else
    idx_run = selP;
end
fprintf('Plantas seleccionadas: %s\n\n', strjoin({Plantas(idx_run).id}, ', '));

fprintf('=== SELECCION DE CONTROLADOR BASE PARA PESOS ===\n');
fprintf('  1) PI\n');
fprintf('  2) PID\n');
baseCtrl_in = input('Ingresa el numero de la opcion (default = 1): ','s');
baseCtrl_in = strtrim(baseCtrl_in);
if isempty(baseCtrl_in), baseCtrl_in = '1'; end

selBaseCtrl = parse_lista_numerica(baseCtrl_in, 1:2);
if numel(selBaseCtrl) ~= 1
    error('Debes seleccionar una sola opcion para el controlador base.');
end

if selBaseCtrl == 1
    baseCtrl = 'PI';
else
    baseCtrl = 'PID';
end
fprintf('Controlador base: %s\n\n', baseCtrl);

fprintf('=== MODO DE PESOS BASE ===\n');
fprintf('  1) SEARCH  - buscar pesos base normalmente\n');
fprintf('  2) SKIP    - omitir busqueda base y usar pesos neutros\n');
baseMode_in = input('Ingresa el numero de la opcion (default = 1): ','s');
baseMode_in = strtrim(baseMode_in);
if isempty(baseMode_in), baseMode_in = '1'; end

selBaseMode = parse_lista_numerica(baseMode_in, 1:2);
if numel(selBaseMode) ~= 1
    error('Debes seleccionar una sola opcion para el modo de pesos base.');
end

if selBaseMode == 1
    baseMode = 'SEARCH';
```

```
else
    baseMode = 'SKIP';
end
fprintf('Modo pesos base: %s\n\n', baseMode);

fprintf('=== SELECCION DE CONTROLADORES A COMPARAR ===\n');
fprintf('  0) Todos\n');
fprintf('  1) PI\n');
fprintf('  2) PID\n');
fprintf('  3) SP\n');
fprintf('  4) MSP\n');
modoC = input('Ingresa 0 para todos, o uno o varios numeros separados por coma: ','s');
modoC = strtrim(modoC);
if isempty(modoC), modoC = '0'; end

selCtrl = parse_lista_numerica(modoC, 0:4);
if ismember(0, selCtrl)
    if numel(selCtrl) > 1
        error('La opcion 0 para controladores debe usarse sola.');
    end
    ctrls = {'PI','PID','SP','MSP'};
else
    mapa_ctrls = {'PI','PID','SP','MSP'};
    ctrls = mapa_ctrls(selCtrl);
    if isempty(ctrls)
        error('Debes seleccionar al menos un controlador.');
    end
end

runPI  = ismember('PI', ctrls);
runPID = ismember('PID', ctrls);
runSP  = ismember('SP', ctrls);
runMSP = ismember('MSP', ctrls);

fprintf('Controladores seleccionados: %s\n', strjoin(ctrls, ', '));
fprintf('Se ejecutara: PI=%d | PID=%d | SP=%d | MSP=%d\n\n', runPI, runPID, runSP, runMSP);

fprintf('=== SELECCION DE PASO DE BARRIDO EN Mst ===\n');
mst_step = input('Ingresa el paso para Mst (recomendado 0.2): ');
if isempty(mst_step), mst_step = 0.2; end
if ~isscalar(mst_step) || mst_step <= 0
    error('El paso de Mst debe ser un escalar positivo.');
end

fprintf('=== RANGO DE BARRIDO EN Mst ===\n');
mst_min = input('Ingresa Mst minimo (enter = 1): ');
if isempty(mst_min), mst_min = 1; end
mst_max = input('Ingresa Mst maximo (enter = 2): ');
if isempty(mst_max), mst_max = 2; end
if ~isscalar(mst_min) || ~isscalar(mst_max) || mst_min <= 0 || mst_max <= mst_min
    error('Rango invalido de Mst.');
end

mst_grid = build_mst_grid(mst_min, mst_max, mst_step); % Construye el barrido principal de
robustez
fprintf('Barrido Mst: %s\n\n', vec2str(mst_grid, '%.4f'));

%% ==================== 1. PESOS BASE POR PLANTA ====================
fprintf('\n');
print_block_title('PESOS BASE POR PLANTA');
fprintf('1. CONTROLADOR BASE: %s | Mst_base = %.4f | BaseMode = %s\n', baseCtrl, Mst_base,
baseMode);
fprintf('%s\n', repmat('-',1,112));

for ip = idx_run
```

```
382     fprintf('\n1.%d Planta %s (%s)\n', find(idx_run==ip), Plantas(ip).id, Plantas(ip).tipo);
383     fprintf('%s\n', repmat('-',1,112));
384 
385     P = pade(Plantas(ip).P_dt, pade_order);
386     theta0 = Plantas(ip).P_dt.InputDelay;
387     [num0, den0] = tfdata(Plantas(ip).P_dt,'v');
388     P0 = tf(num0, den0);
389 
390     if strcmpi(baseCtrl,'PI')
391         ctrl_key_base = 'PI';
392         refinamientos_use = refinamientos;
393         h_1_vals_use = h_1_vals;
394     else
395         ctrl_key_base = 'PID';
396         refinamientos_use = refinamientos_pid;
397         h_1_vals_use = h_1_vals_pid;
398     end
399 
400     if strcmp(baseMode,'SKIP')
401         status_msg('WARN', sprintf('Planta %s | Base %s | EXTRAORDINARIO: se omite busqueda
    de pesos base y se usan pesos neutros', ...
402             Plantas(ip).id, ctrl_key_base), 'important');
403 
404         Plantas(ip).base.scan = [];
405         Plantas(ip).base.bounds_status = 'SKIPPED';
406         Plantas(ip).base.bounds_list   = {};
407         Plantas(ip).base.regions       = struct([]);
408         Plantas(ip).base.ctrl_key      = ctrl_key_base;
409         Plantas(ip).base.ref_status    = 'SKIPPED';
410         Plantas(ip).base.candidates    = struct('kc',[],'ti',[],'td',[],'mst',[],'iaedo',
    [],'iaedi',[],'ctrl_key',ctrl_key_base);
411         Plantas(ip).base.iter_log      = {};
412 
413         base_result = build_neutral_base_result(ctrl_key_base); % Usa IAEdo0_base=1 e
    IAEdi0_base=1
414         base_status = 'SKIPPED_W1';
415 
416         Plantas(ip).base.result = base_result;
417         Plantas(ip).base.status = base_status;
418         Plantas(ip).base.extraordinary_note = 'EXTRAORDINARIO: se omitio la busqueda de pesos
    base; se usan pesos neutros IAEdo0_base=1 e IAEdi0_base=1';
419     else
420         status_msg('RUN', sprintf('Planta %s | Base %s | GlobalScan Mst=%.4f',
    Plantas(ip).id, ctrl_key_base, Mst_base));
421 
422         scan_base = global_mst_map( ...
423             ctrl_key_base, P, P0, theta0, Plantas(ip).tipo, Plantas(ip).id, ...
424             scan_paso_global, kc_bounds_global, ti_bounds_global, td_bounds_global, ...
425             Go, Gi, pade_order, N_der, Tf_min, Plantas(ip).id, use_exact_mst, w_mst); %
    Explora globalmente donde viven los Mst estables
426 
427         Plantas(ip).base.scan = scan_base;
428         print_global_scan_summary(scan_base);
429 
430         [bounds_list_base, status_base_scan, regions_base] = select_bounds_from_scan( ...
431             scan_base, ctrl_key_base, Mst_base, h_1_vals_use(1), margen_mult, ...
432             kc_bounds_global, ti_bounds_global, td_bounds_global, ...
433             region_gap_mult, max_regions_keep, region_min_pts); % Selecciona regiones
    prometedoras cerca de Mst_base
434 
435         Plantas(ip).base.bounds_status = status_base_scan;
436         Plantas(ip).base.bounds_list   = bounds_list_base;
437         Plantas(ip).base.regions       = regions_base;
438         Plantas(ip).base.ctrl_key      = ctrl_key_base;
439 
```

```
440         print_selected_regions(bounds_list_base, status_base_scan, ctrl_key_base);
441 
442         if show_iter_logs
443             status_msg('RUN', sprintf('Planta %s | Base %s | Refinement Mst=%.4f',
    Plantas(ip).id, ctrl_key_base, Mst_base));
444             print_subtitle(sprintf('1.%d.%s - BUSQUEDA DE PESOS BASE', find(idx_run==ip),
    ctrl_key_base));
445             print_iter_header();
446         end
447 
448         [cand_base, iter_log_base, ref_base_status] = refinement_over_bounds_list( ...
449             ctrl_key_base, P, P0, theta0, Plantas(ip).tipo, Plantas(ip).id, ...
450             bounds_list_base, Mst_base, refinamientos_use, h_1_vals_use, edge_mult,
    h_relax_factor, ...
451             kc_bounds_global, ti_bounds_global, td_bounds_global, ...
452             Go, Gi, t, pade_order, N_der, Tf_min, Plantas(ip).id, use_exact_mst, w_mst, ...
453             save_eval_detail, max_empty_iters, [], []); % Refina cada region hasta obtener
    candidatos utiles
454 
455         Plantas(ip).base.ref_status = ref_base_status;
456         Plantas(ip).base.candidates = cand_base;
457         Plantas(ip).base.iter_log   = iter_log_base;
458 
459         if show_iter_logs
460             print_iter_log(iter_log_base);
461         end
462 
463         [base_result, base_status] = build_base_result(ctrl_key_base, cand_base); % Construye
    IAEdo0_base e IAEdi0_base
464 
465         if ~strcmp(base_status,'OK')
466             status_msg('WARN', sprintf('Planta %s | Base %s | Sin pesos base validos, se usan
    pesos neutros', ...
467                 Plantas(ip).id, ctrl_key_base), 'important');
468             base_result = build_neutral_base_result(ctrl_key_base);
469             base_status = 'FALLBACK_W1';
470             Plantas(ip).base.extraordinary_note = 'EXTRAORDINARIO: no hubo pesos base
    validos; se usan pesos neutros IAEdo0_base=1 e IAEdi0_base=1';
471         else
472             status_msg('DONE', sprintf('Planta %s | Base %s | Pesos base listos',
    Plantas(ip).id, ctrl_key_base));
473         end
474 
475         Plantas(ip).base.result = base_result;
476         Plantas(ip).base.status = base_status;
477     end
478 
479     fprintf('  BaseCtrl=%-4s | Scan=%-14s | Ref=%-12s | BaseStatus=%-14s\n', ...
480         ctrl_key_base, clip_str(getfield_safe(Plantas(ip).base,'bounds_status','NA'),14), ...
481         clip_str(getfield_safe(Plantas(ip).base,'ref_status','NA'),12), ...
482         clip_str(base_status,14));
483 
484     fprintf('  do0_base | k_c=%-10s | tau_i=%-10s | tau_d=%-10s | Mst=%-10s | IAEdo0=%-10s |
    IAEdi@do0=%-10s\n', ...
485         fmt_num(base_result.do0.kc_do0), fmt_num(base_result.do0.ti_do0),
    fmt_num(base_result.do0.td_do0), ...
486         fmt_num(base_result.do0.mst_do0), fmt_num(base_result.do0.iaedo_do0),
    fmt_num(base_result.do0.iaedi_at_do0));
487 
488     fprintf('  di0_base | k_c=%-10s | tau_i=%-10s | tau_d=%-10s | Mst=%-10s | IAEdo@di0=%-10s
    | IAEdi0=%-10s\n', ...
489         fmt_num(base_result.di0.kc_di0), fmt_num(base_result.di0.ti_di0),
    fmt_num(base_result.di0.td_di0), ...
490         fmt_num(base_result.di0.mst_di0), fmt_num(base_result.di0.iaedo_at_di0),
    fmt_num(base_result.di0.iaedi_di0));
```

```
491
492     if isfield(Plantas(ip).base,'extraordinary_note')
493         fprintf('  %s\n', Plantas(ip).base.extraordinary_note);
494     end
495 end
496
497 %% ==================== 2. BARRIDO J vs Mst ====================
498 fprintf('\n');
499 print_block_title('BARRIDO J vs Mst');
500 fprintf('2. BUSQUEDA POR ROBUSTEZ\n');
501 print_notes_vertical({
502     'Cada Mst_obj arranca con un GlobalScan'
503     'Si el target no existe en la nube estable, se buscan regiones estables cercanas'
504     'Si una region no produce candidatos durante varias iteraciones seguidas, se corta'
505     'Si BaseMode=SKIP o si falla la base, J usa pesos neutros = 1'
506     'La tolerancia final de Mst es la h_used de la ultima iteracion valida'
507     });
508
509 for ip = idx_run
510     fprintf('\n');
511     print_block_title(sprintf('PLANTA %s (%s)', Plantas(ip).id, upper(Plantas(ip).tipo)));
512
513     P = pade(Plantas(ip).P_dt, pade_order);
514     theta0 = Plantas(ip).P_dt.InputDelay;
515     [num0, den0] = tfdata(Plantas(ip).P_dt,'v');
516     P0 = tf(num0, den0);
517
518     iaedo0_base = Plantas(ip).base.result.do0.iaedo_do0;
519     iaedi0_base = Plantas(ip).base.result.di0.iaedi_di0;
520
521     fprintf('Pesos usados: status=%s | IAEdo0_base=%s | IAEdi0_base=%s\n', ...
522         Plantas(ip).base.status, fmt_num(iaedo0_base), fmt_num(iaedi0_base));
523     if isfield(Plantas(ip).base,'extraordinary_note')
524         fprintf('  %s\n', Plantas(ip).base.extraordinary_note);
525     end
526
527     if runPI
528         fprintf('\n2.%d.1 PI\n', find(idx_run==ip));
529         Plantas(ip).PI_sweep = run_mst_sweep( ...
530             'PI', mst_grid, P, P0, theta0, Plantas(ip).tipo, Plantas(ip).id, ...
531             refinamientos, h_1_vals, edge_mult, h_relax_factor, ...
532             scan_paso_global, region_gap_mult, max_regions_keep, region_min_pts, margen_mult,
    ...
533             kc_bounds_global, ti_bounds_global, td_bounds_global, ...
534             Go, Gi, t, pade_order, N_der, Tf_min, ...
535             iaedo0_base, iaedi0_base, Plantas(ip).id, ...
536             use_exact_mst, w_mst, save_eval_detail, max_empty_iters); % Barrido completo J vs
    Mst para PI
537         if print_sweep_tables, print_sweep_table(Plantas(ip).PI_sweep); end
538     end
539
540     if runPID
541         fprintf('\n2.%d.2 PID\n', find(idx_run==ip));
542         Plantas(ip).PID_sweep = run_mst_sweep( ...
543             'PID', mst_grid, P, P0, theta0, Plantas(ip).tipo, Plantas(ip).id, ...
544             refinamientos_pid, h_1_vals_pid, edge_mult, h_relax_factor, ...
545             scan_paso_global, region_gap_mult, max_regions_keep, region_min_pts, margen_mult,
    ...
546             kc_bounds_global, ti_bounds_global, td_bounds_global, ...
547             Go, Gi, t, pade_order, N_der, Tf_min, ...
548             iaedo0_base, iaedi0_base, Plantas(ip).id, ...
549             use_exact_mst, w_mst, save_eval_detail, max_empty_iters); % Barrido completo J vs
    Mst para PID
550         if print_sweep_tables, print_sweep_table(Plantas(ip).PID_sweep); end
551     end
```

```
552
553     if runSP
554         fprintf('\n2.%d.3 SP\n', find(idx_run==ip));
555         Plantas(ip).SP_sweep = run_mst_sweep( ...
556             'SP', mst_grid, P, P0, theta0, Plantas(ip).tipo, Plantas(ip).id, ...
557             refinamientos, h_1_vals, edge_mult, h_relax_factor, ...
558             scan_paso_global, region_gap_mult, max_regions_keep, region_min_pts, margen_mult,
    ...
559             kc_bounds_global, ti_bounds_global, td_bounds_global, ...
560             Go, Gi, t, pade_order, N_der, Tf_min, ...
561             iaedo0_base, iaedi0_base, Plantas(ip).id, ...
562             use_exact_mst, w_mst, save_eval_detail, max_empty_iters); % Barrido completo J vs
    Mst para SP
563         if print_sweep_tables, print_sweep_table(Plantas(ip).SP_sweep); end
564     end
565
566     if runMSP
567         fprintf('\n2.%d.4 MSP\n', find(idx_run==ip));
568         Plantas(ip).MSP_sweep = run_mst_sweep( ...
569             'MSP', mst_grid, P, P0, theta0, Plantas(ip).tipo, Plantas(ip).id, ...
570             refinamientos, h_1_vals, edge_mult, h_relax_factor, ...
571             scan_paso_global, region_gap_mult, max_regions_keep, region_min_pts, margen_mult,
    ...
572             kc_bounds_global, ti_bounds_global, td_bounds_global, ...
573             Go, Gi, t, pade_order, N_der, Tf_min, ...
574             iaedo0_base, iaedi0_base, Plantas(ip).id, ...
575             use_exact_mst, w_mst, save_eval_detail, max_empty_iters); % Barrido completo J vs
    Mst para MSP
576         if print_sweep_tables, print_sweep_table(Plantas(ip).MSP_sweep); end
577     end
578 end
579
580 for ip = idx_run
581     if runPI && isfield(Plantas(ip),'PI_sweep')
582         Plantas(ip).PI_sweep = enrich_sweep_with_freq_metrics(Plantas(ip), 'PI',
    Plantas(ip).PI_sweep, Go, Gi, pade_order, N_der, Tf_min);
583     end
584     if runPID && isfield(Plantas(ip),'PID_sweep')
585         Plantas(ip).PID_sweep = enrich_sweep_with_freq_metrics(Plantas(ip), 'PID',
    Plantas(ip).PID_sweep, Go, Gi, pade_order, N_der, Tf_min);
586     end
587     if runSP && isfield(Plantas(ip),'SP_sweep')
588         Plantas(ip).SP_sweep = enrich_sweep_with_freq_metrics(Plantas(ip), 'SP',
    Plantas(ip).SP_sweep, Go, Gi, pade_order, N_der, Tf_min);
589     end
590     if runMSP && isfield(Plantas(ip),'MSP_sweep')
591         Plantas(ip).MSP_sweep = enrich_sweep_with_freq_metrics(Plantas(ip), 'MSP',
    Plantas(ip).MSP_sweep, Go, Gi, pade_order, N_der, Tf_min);
592         Plantas(ip).MSP_sweep = attach_msp_paper_reference( ...
593             Plantas(ip), Plantas(ip).MSP_sweep, Plantas(ip).base.result, ...
594             Go, Gi, t, pade_order, N_der, Tf_min, use_exact_mst, w_mst);
595     end
596 end
597
598 Plantas = backfill_freq_metrics_if_missing(Plantas, idx_run, runPI, runPID, runSP, runMSP,
    Go, Gi, t, pade_order, N_der, Tf_min, use_exact_mst, w_mst);
599
600 T_base  = build_base_export_table(Plantas, idx_run, baseMode);
601 T_sweep = build_sweep_export_table(Plantas, idx_run, runPI, runPID, runSP, runMSP);
602 T_freq  = build_sweep_freq_export_table(Plantas, idx_run, runPI, runPID, runSP, runMSP);
603 T_extra = build_extraordinary_export_table(Plantas, idx_run, runPI, runPID, runSP, runMSP);
604 T_summary = build_sweep_summary_table(Plantas, idx_run, runPI, runPID, runSP, runMSP);
605 T_winner  = build_sweep_winner_table(T_summary);
606 T_search_report = build_search_report_table(Plantas, idx_run, runPI, runPID, runSP, runMSP,
    report_mst_ref);
```

```
607 T_search_winner = build_search_winner_report_table(T_search_report);
608 T_search_magnitude = build_search_magnitude_table(Plantas, idx_run, runPI, runPID, runSP,
    runMSP);
609 did_run_search_stage = true;
610
611 %% ====================== 3. REPORTE DE BUSQUEDA ======================
612 fprintf('\n');
613 print_block_title('REPORTE DE BUSQUEDA');
614
615 if ~exist('thesis_mode', 'var') || isempty(thesis_mode)
616     thesis_mode = true;
617 end
618 if ~exist('show_iter_status', 'var') || isempty(show_iter_status)
619     show_iter_status = true;
620 end
621 if ~exist('export_pub_figs', 'var') || isempty(export_pub_figs)
622     export_pub_figs = true;
623 end
624 if ~exist('export_results_csv', 'var') || isempty(export_results_csv)
625     export_results_csv = false;
626 end
627 if ~exist('export_results_xlsx', 'var') || isempty(export_results_xlsx)
628     export_results_xlsx = true;
629 end
630 if ~exist('save_run_workspace', 'var') || isempty(save_run_workspace)
631     save_run_workspace = true;
632 end
633 if ~exist('print_export_tables', 'var') || isempty(print_export_tables)
634     print_export_tables = true;
635 end
636 if ~exist('print_search_report', 'var') || isempty(print_search_report)
637     print_search_report = true;
638 end
639 if ~exist('export_final_report_txt', 'var') || isempty(export_final_report_txt)
640     export_final_report_txt = true;
641 end
642 if ~exist('print_sweep_tables', 'var') || isempty(print_sweep_tables)
643     print_sweep_tables = false;
644 end
645 if ~exist('export_annex_reports', 'var') || isempty(export_annex_reports)
646     export_annex_reports = true;
647 end
648 if ~exist('report_mst_ref', 'var') || isempty(report_mst_ref)
649     report_mst_ref = 1.6;
650 end
651 if ~exist('csv_delimiter', 'var') || isempty(csv_delimiter)
652     csv_delimiter = ';';
653 end
654 if ~exist('export_metadata_txt', 'var') || isempty(export_metadata_txt)
655     export_metadata_txt = true;
656 end
657 if ~exist('plot_j_use_cap', 'var') || isempty(plot_j_use_cap)
658     plot_j_use_cap = true;
659 end
660 if ~exist('plot_j_hard_cap', 'var') || isempty(plot_j_hard_cap)
661     plot_j_hard_cap = 50;
662 end
663 if ~exist('plot_j_qref', 'var') || isempty(plot_j_qref)
664     plot_j_qref = 95;
665 end
666 if ~exist('plot_j_hide_above', 'var') || isempty(plot_j_hide_above)
667     plot_j_hide_above = 10000;
668 end
669 if ~exist('plot_j_use_zoom_inset', 'var') || isempty(plot_j_use_zoom_inset)
670     plot_j_use_zoom_inset = false;
```

```
end
if ~exist('did_run_search_stage', 'var') || isempty(did_run_search_stage)
    did_run_search_stage = false;
end
if ~exist('refresh_loaded_sweeps_before_report', 'var') ||
isempty(refresh_loaded_sweeps_before_report)
    refresh_loaded_sweeps_before_report = false;
end

if ~exist('idx_run', 'var') || isempty(idx_run)
    idx_run = 1:numel(Plantas);
end
if ~exist('runPI', 'var') || ~exist('runPID', 'var') || ~exist('runSP', 'var') ||
~exist('runMSP', 'var')
    [runPI, runPID, runSP, runMSP] = infer_run_flags_from_loaded_sweeps_td(Plantas, idx_run);
end
if ~exist('ctrls', 'var') || isempty(ctrls)
    ctrls = {};
    if runPI,  ctrls{end+1} = 'PI'; end %#ok<AGROW>
    if runPID, ctrls{end+1} = 'PID'; end %#ok<AGROW>
    if runSP,  ctrls{end+1} = 'SP'; end %#ok<AGROW>
    if runMSP, ctrls{end+1} = 'MSP'; end %#ok<AGROW>
end
if ~exist('Go', 'var') || isempty(Go)
    Go = 1;
end
if ~exist('Gi', 'var') || isempty(Gi)
    Gi = 1;
end
if ~exist('dt', 'var') || isempty(dt)
    dt = 0.01;
end
if ~exist('T_end', 'var') || isempty(T_end)
    T_end = 30;
end
if ~exist('t', 'var') || isempty(t)
    t = 0:dt:T_end;
end

if refresh_loaded_sweeps_before_report && ~did_run_search_stage
    fprintf('Tie-in: refrescando IAE/J de sweeps cargados con el codigo actual antes del
reporte...\n');
    Plantas = refresh_loaded_sweeps_j_metrics( ...
        Plantas, idx_run, runPI, runPID, runSP, runMSP, ...
        Go, Gi, t, pade_order, N_der, Tf_min);
end

ctrls_sel_txt = strjoin(ctrls, ', ');
plants_sel_txt = strjoin({Plantas(idx_run).id}, ', ');

if print_search_report
    T_search_report = build_search_report_table(Plantas, idx_run, runPI, runPID, runSP,
runMSP, report_mst_ref);
    T_search_winner = build_search_winner_report_table(T_search_report);
    T_search_magnitude = build_search_magnitude_table(Plantas, idx_run, runPI, runPID, runSP,
runMSP);

    fprintf('1. IDENTIFICACION DE LA CORRIDA\n');
    fprintf('%s\n', repmat('-',1,84));
    fprintf('Codigo       : %s\n', code_version);
    fprintf('Plantas      : %s\n', plants_sel_txt);
    fprintf('Controladores: %s\n', ctrls_sel_txt);
    fprintf('Mst barrido  : %.4f a %.4f | paso %.4f | n=%d\n', mst_min, mst_max, mst_step,
numel(mst_grid));
    fprintf('Pesos base   : %s | Mst_base %.4f | modo %s\n', baseCtrl, Mst_base, baseMode);
```

```
730     fprintf('Criterio OK  : Mst_real dentro de h_used de la ultima iteracion valida\n');
731     fprintf('Anexos       : %s\n\n', ternary_txt(export_annex_reports, 'se exportaran por
    planta', 'desactivados'));
732
733     fprintf('2. PESOS BASE USADOS\n');
734     fprintf('%s\n', repmat('-',1,84));
735     print_compact_base_report(Plantas, idx_run);
736
737     fprintf('\n2. INDICE RAPIDO DE COLUMNAS\n');
738     fprintf('%s\n', repmat('-',1,84));
739     print_search_column_guide();
740
741     fprintf('\n3. RESUMEN POR CONTROLADOR (cobertura y mejor punto encontrado)\n');
742     fprintf('%s\n', repmat('-',1,84));
743     fprintf('Mst de referencia para comparar J y DM: %.2f\n', report_mst_ref);
744     fprintf('OK/Tot = soluciones validas / total de Mst pedidos.\n\n');
745
746     print_search_report_table(T_search_report);
747
748     fprintf('\n');
749     fprintf('4. GANADORES POR PLANTA (quien gana cada criterio)\n');
750     fprintf('%s\n', repmat('-',1,84));
751     fprintf('Mejor J = menor J. Mejor DMref = mayor DM en el Mst de referencia. Mejor Track =
    menor error medio de seguimiento de Mst.\n\n');
752     print_search_winner_report_table(T_search_winner);
753
754     fprintf('\n');
755     fprintf('5. MAGNITUD DE LA BUSQUEDA\n');
756     fprintf('%s\n', repmat('-',1,84));
757     fprintf('EvalRef = puntos evaluados en mallas de refinamiento. CandBand = candidatos que
    entraron a banda durante el refinamiento.\n\n');
758     print_search_magnitude_table(T_search_magnitude);
759 else
760     fprintf('Reporte de busqueda compacto desactivado.\n');
761 end
762
763 if print_export_tables
764     fprintf('\n');
765     print_block_title('REPORTE FINAL PARA ANEXO');
766     if ~exist('T_base', 'var') || isempty(T_base)
767         T_base = build_base_export_table(Plantas, idx_run, baseMode);
768     end
769     if ~exist('T_sweep', 'var') || isempty(T_sweep)
770         T_sweep = build_sweep_export_table(Plantas, idx_run, runPI, runPID, runSP, runMSP);
771     end
772     if ~exist('T_freq', 'var') || isempty(T_freq)
773         T_freq = build_sweep_freq_export_table(Plantas, idx_run, runPI, runPID, runSP,
    runMSP);
774     end
775     if ~exist('T_extra', 'var') || isempty(T_extra)
776         T_extra = build_extraordinary_export_table(Plantas, idx_run, runPI, runPID, runSP,
    runMSP);
777     end
778     if ~exist('T_summary', 'var') || isempty(T_summary)
779         T_summary = build_sweep_summary_table(Plantas, idx_run, runPI, runPID, runSP, runMSP);
780     end
781     if ~exist('T_winner', 'var') || isempty(T_winner)
782         T_winner = build_sweep_winner_table(T_summary);
783     end
784     if ~exist('T_search_report', 'var') || isempty(T_search_report)
785         T_search_report = build_search_report_table(Plantas, idx_run, runPI, runPID, runSP,
    runMSP, report_mst_ref);
786     end
787     if ~exist('T_search_winner', 'var') || isempty(T_search_winner)
788         T_search_winner = build_search_winner_report_table(T_search_report);
```

```
    end
    if ~exist('T_search_magnitude', 'var') || isempty(T_search_magnitude)
        T_search_magnitude = build_search_magnitude_table(Plantas, idx_run, runPI, runPID,
runSP, runMSP);
    end
    print_final_search_annex_report( ...
        1, Plantas, idx_run, T_base, T_sweep, T_freq, T_extra, T_summary, T_winner, ...
        T_search_report, T_search_winner, T_search_magnitude, code_version, baseCtrl,
baseMode, ctrls_sel_txt, ...
        plants_sel_txt, Mst_base, mst_min, mst_max, mst_step, mst_grid, report_mst_ref);
end

if export_final_report_txt
    if ~exist('run_stamp', 'var') || isempty(run_stamp)
        run_stamp = datestr(now, 'yyyymmdd_HHMMSS');
    end
    if ~exist('project_root', 'var') || isempty(project_root)
        project_root = resolve_script_root();
    end
    if ~exist('thesis_out_root', 'var') || isempty(thesis_out_root)
        thesis_out_root = fullfile(project_root, 'salida_tesis');
    end
    if ~exist('annex_out_folder', 'var') || isempty(annex_out_folder)
        annex_out_folder = fullfile(thesis_out_root, 'anexos_busqueda');
    end
    if ~exist('final_report_file', 'var') || isempty(final_report_file)
        final_report_file = fullfile(annex_out_folder,
sprintf('reporte_final_busqueda_%s.txt', run_stamp));
    end
    if ~exist(annex_out_folder, 'dir')
        mkdir(annex_out_folder);
    end
    if ~exist('T_base', 'var') || isempty(T_base)
        T_base = build_base_export_table(Plantas, idx_run, baseMode);
    end
    if ~exist('T_sweep', 'var') || isempty(T_sweep)
        T_sweep = build_sweep_export_table(Plantas, idx_run, runPI, runPID, runSP, runMSP);
    end
    if ~exist('T_freq', 'var') || isempty(T_freq)
        T_freq = build_sweep_freq_export_table(Plantas, idx_run, runPI, runPID, runSP,
runMSP);
    end
    if ~exist('T_extra', 'var') || isempty(T_extra)
        T_extra = build_extraordinary_export_table(Plantas, idx_run, runPI, runPID, runSP,
runMSP);
    end
    if ~exist('T_summary', 'var') || isempty(T_summary)
        T_summary = build_sweep_summary_table(Plantas, idx_run, runPI, runPID, runSP, runMSP);
    end
    if ~exist('T_winner', 'var') || isempty(T_winner)
        T_winner = build_sweep_winner_table(T_summary);
    end
    if ~exist('T_search_report', 'var') || isempty(T_search_report)
        T_search_report = build_search_report_table(Plantas, idx_run, runPI, runPID, runSP,
runMSP, report_mst_ref);
    end
    if ~exist('T_search_winner', 'var') || isempty(T_search_winner)
        T_search_winner = build_search_winner_report_table(T_search_report);
    end
    if ~exist('T_search_magnitude', 'var') || isempty(T_search_magnitude)
        T_search_magnitude = build_search_magnitude_table(Plantas, idx_run, runPI, runPID,
runSP, runMSP);
    end
    write_final_search_annex_report( ...
```

```
        final_report_file, Plantas, idx_run, T_base, T_sweep, T_freq, T_extra, T_summary, T_winner, ...
        T_search_report, T_search_winner, T_search_magnitude, code_version, baseCtrl, baseMode, ctrls_sel_txt, ...
        plants_sel_txt, Mst_base, mst_min, mst_max, mst_step, mst_grid, report_mst_ref);
    fprintf('\nReporte final para anexo exportado en:\n  %s\n', final_report_file);
end

%% ==================== 4. GRAFICAS J vs Mst ====================
fprintf('\n');
print_block_title('GRAFICAS J vs Mst');

if ~exist('legend_location_single', 'var') || isempty(legend_location_single)
    legend_location_single = 'northeast';
end
if ~exist('legend_location_shared', 'var') || isempty(legend_location_shared)
    legend_location_shared = 'southoutside';
end
if ~exist('legend_num_cols', 'var') || isempty(legend_num_cols)
    legend_num_cols = 4;
end
if ~exist('font_name_axes', 'var') || isempty(font_name_axes)
    font_name_axes = 'Times New Roman';
end
if ~exist('font_name_title', 'var') || isempty(font_name_title)
    font_name_title = 'Times New Roman';
end
if ~exist('font_size_axes', 'var') || isempty(font_size_axes)
    font_size_axes = 9.5;
end
if ~exist('font_size_title', 'var') || isempty(font_size_title)
    font_size_title = 10.5;
end
if ~exist('font_size_legend', 'var') || isempty(font_size_legend)
    font_size_legend = 8.5;
end
if ~exist('line_w', 'var') || isempty(line_w)
    line_w = 1.80;
end
if ~exist('marker_sz', 'var') || isempty(marker_sz)
    marker_sz = 6.0;
end

Plantas = backfill_freq_metrics_if_missing(Plantas, idx_run, runPI, runPID, runSP, runMSP, Go, Gi, t, pade_order, N_der, Tf_min, use_exact_mst, w_mst);

styles = get_thesis_plot_styles();

for ip = idx_run
    fig = figure( ...
        'Name', sprintf('J vs Mst - %s', Plantas(ip).id), ...
        'NumberTitle', 'off', ...
        'Color', 'w', ...
        'Units', 'centimeters', ...
        'Position', [2 2 fig_w_cm fig_h_cm]);

    ax = axes(fig);
    hold(ax, 'on');
    format_metric_axis(ax, font_name_axes, font_size_axes, grid_alpha_major, grid_alpha_minor);

    y_all_valid = [];
    if runPI && isfield(Plantas(ip),'PI_sweep')
        y_all_valid = [y_all_valid; get_sweep_plotted_values(Plantas(ip).PI_sweep, 'J_opt')];
    end
```

```
907     if runPID && isfield(Plantas(ip),'PID_sweep')
908         y_all_valid = [y_all_valid; get_sweep_plotted_values(Plantas(ip).PID_sweep, 'J_opt')];
909     end
910     if runSP && isfield(Plantas(ip),'SP_sweep')
911         y_all_valid = [y_all_valid; get_sweep_plotted_values(Plantas(ip).SP_sweep, 'J_opt')];
912     end
913     if runMSP && isfield(Plantas(ip),'MSP_sweep')
914         y_all_valid = [y_all_valid; get_sweep_plotted_values(Plantas(ip).MSP_sweep, 'J_opt')];
915     end
916 
917     y_all_valid = y_all_valid(isfinite(y_all_valid) & (y_all_valid <= plot_j_hide_above));
918     [y_min_plot, y_top_plot] = compute_clean_j_limits(y_all_valid, plot_j_use_cap,
    plot_j_hard_cap, plot_j_qref); % Ajusta la escala visual de J
919 
920     hmain = gobjects(0);
921     if runPI && isfield(Plantas(ip),'PI_sweep')
922         h = plot_sweep_curve_clean(ax, Plantas(ip).PI_sweep, styles.PI, 'PI', y_top_plot,
    line_w, marker_sz, plot_j_hide_above, 'static_markers');
923         if ~isempty(h) && all(isgraphics(h)), hmain(end+1) = h; end %#ok<AGROW>
924     end
925     if runPID && isfield(Plantas(ip),'PID_sweep')
926         h = plot_sweep_curve_clean(ax, Plantas(ip).PID_sweep, styles.PID, 'PID', y_top_plot,
    line_w, marker_sz, plot_j_hide_above, 'static_markers');
927         if ~isempty(h) && all(isgraphics(h)), hmain(end+1) = h; end %#ok<AGROW>
928     end
929     if runSP && isfield(Plantas(ip),'SP_sweep')
930         h = plot_sweep_curve_clean(ax, Plantas(ip).SP_sweep, styles.SP, 'SP', y_top_plot,
    line_w, marker_sz, plot_j_hide_above, 'static_markers');
931         if ~isempty(h) && all(isgraphics(h)), hmain(end+1) = h; end %#ok<AGROW>
932     end
933     if runMSP && isfield(Plantas(ip),'MSP_sweep')
934         h = plot_sweep_curve_clean(ax, Plantas(ip).MSP_sweep, styles.MSP, 'MSP', y_top_plot,
    line_w, marker_sz, plot_j_hide_above, 'static_markers');
935         if ~isempty(h) && all(isgraphics(h)), hmain(end+1) = h; end %#ok<AGROW>
936     end
937 
938     xlabel(ax, 'Robustez, Mst', 'FontName', font_name_axes, 'FontSize', font_size_axes + 0.5,
    'Interpreter', 'none');
939     ylabel(ax, 'Desempeno, J', 'FontName', font_name_axes, 'FontSize', font_size_axes + 0.5,
    'Interpreter', 'tex');
940 
941     title(ax, sprintf('Planta %s | J vs Robustez', Plantas(ip).id), ...
942         'FontName', font_name_title, ...
943         'FontSize', font_size_title, ...
944         'FontWeight', 'bold', ...
945         'Interpreter', 'none');
946 
947     xlim(ax, [min(mst_grid) max(mst_grid)]);
948     ylim(ax, [y_min_plot y_top_plot * 1.04]);
949 
950     if ~isempty(hmain)
951         labels_j = get(hmain, {'DisplayName'});
952         if ischar(labels_j), labels_j = {labels_j}; end
953         lgd = legend(ax, hmain, labels_j, 'Location', legend_location_single, 'Orientation',
    'vertical');
954         apply_publication_legend(lgd, font_name_axes, font_size_legend, 1);
955     end
956 
957     if isfield(Plantas(ip).base,'extraordinary_note')
958         text(ax, 0.99, 0.02, 'Base extraordinaria', ...
959             'Units', 'normalized', ...
960             'HorizontalAlignment', 'right', ...
961             'VerticalAlignment', 'bottom', ...
962             'FontName', font_name_axes, ...
963             'FontSize', max(7.5, font_size_axes - 1), ...
```

```
964             'Color', [0.35 0.35 0.35]);
965     end
966
967     if export_pub_figs
968         exportgraphics(fig, fullfile(pub_fig_folder, sprintf('J_vs_Mst_%s.png',
    Plantas(ip).id)), 'Resolution', fig_res_dpi);
969         exportgraphics(fig, fullfile(pub_fig_folder, sprintf('J_vs_Mst_%s.pdf',
    Plantas(ip).id)), 'ContentType', 'vector');
970     end
971 end
972
973 fprintf('\n');
974 print_block_title('GRAFICAS DELAY MARGIN vs Mst');
975
976 figDM = figure( ...
977     'Name', 'Delay Margin vs Mst', ...
978     'NumberTitle', 'off', ...
979     'Color', 'w', ...
980     'Units', 'centimeters', ...
981     'Position', [2 2 max(15, fig_w_cm * 1.55) max(10, fig_h_cm * 1.45)]);
982
983 nPlants = numel(idx_run);
984 nColsDM = min(2, max(1, nPlants));
985 nRowsDM = ceil(nPlants / nColsDM);
986 tlDM = tiledlayout(figDM, nRowsDM, nColsDM, 'TileSpacing', 'compact', 'Padding', 'compact');
987 title(tlDM, 'Delay Margin vs Robustez', 'FontName', font_name_title, 'FontSize',
    font_size_title + 0.3, 'FontWeight', 'bold');
988
989 for kp = 1:nPlants
990     ip = idx_run(kp);
991     axDM = nexttile(tlDM, kp);
992     hold(axDM, 'on');
993     format_metric_axis(axDM, font_name_axes, font_size_axes, grid_alpha_major,
    grid_alpha_minor);
994     axDM.YScale = 'log';
995
996     htmp = gobjects(0);
997     if runPI && isfield(Plantas(ip),'PI_sweep')
998         h = plot_sweep_metric(axDM, Plantas(ip).PI_sweep, 'DM_opt', styles.PI, 'PI', line_w,
    marker_sz, 'static_markers');
999         if ~isempty(h) && all(isgraphics(h)), htmp(end+1) = h; %#ok<AGROW>
1000        end
1001    end
1002    if runPID && isfield(Plantas(ip),'PID_sweep')
1003        h = plot_sweep_metric(axDM, Plantas(ip).PID_sweep, 'DM_opt', styles.PID, 'PID',
    line_w, marker_sz, 'static_markers');
1004        if ~isempty(h) && all(isgraphics(h)), htmp(end+1) = h; %#ok<AGROW>
1005        end
1006    end
1007    if runSP && isfield(Plantas(ip),'SP_sweep')
1008        h = plot_sweep_metric(axDM, Plantas(ip).SP_sweep, 'DM_opt', styles.SP, 'SP', line_w,
    marker_sz, 'static_markers');
1009        if ~isempty(h) && all(isgraphics(h)), htmp(end+1) = h; %#ok<AGROW>
1010        end
1011    end
1012    if runMSP && isfield(Plantas(ip),'MSP_sweep')
1013        h = plot_sweep_metric(axDM, Plantas(ip).MSP_sweep, 'DM_opt', styles.MSP, 'MSP',
    line_w, marker_sz, 'static_markers');
1014        if ~isempty(h) && all(isgraphics(h)), htmp(end+1) = h; %#ok<AGROW>
1015        end
1016    end
1017
1018    xlim(axDM, [min(mst_grid) max(mst_grid)]);
1019    xlabel(axDM, 'Robustez, Mst', 'FontName', font_name_axes, 'FontSize', font_size_axes +
    0.3, 'Interpreter', 'none');
```

```
    ylabel(axDM, 'Delay Margin, DM', 'FontName', font_name_axes, 'FontSize', font_size_axes + 0.3, 'Interpreter', 'tex');
    title(axDM, plant_title_with_model(Plantas(ip)), 'FontName', font_name_title, 'FontSize', font_size_title - 0.2, 'FontWeight', 'normal', 'Interpreter', 'tex');
    if ~isempty(htmp)
        labels_dm = get(htmp, {'DisplayName'});
        if ischar(labels_dm), labels_dm = {labels_dm}; end
        apply_publication_legend(legend(axDM, htmp, labels_dm, 'Location', legend_location_single, 'Orientation', 'vertical'), font_name_axes, font_size_legend, 1);
    else
        text(axDM, 0.5, 0.5, 'Sin datos validos para DM', ...
            'Units', 'normalized', 'HorizontalAlignment', 'center', 'VerticalAlignment', 'middle', ...
            'FontName', font_name_axes, 'FontSize', font_size_axes, 'Color', [0.35 0.35 0.35]);
    end
end

if export_pub_figs
    dm_file_base = file_base_with_plant_suffix('DelayMargin_vs_Mst', Plantas, idx_run);
    drawnow;
    export_target_dm = tlDM;
    if ~isgraphics(export_target_dm)
        export_target_dm = figDM;
    end
    exportgraphics(export_target_dm, fullfile(pub_fig_folder, [dm_file_base '.png']), 'Resolution', fig_res_dpi);
    exportgraphics(export_target_dm, fullfile(pub_fig_folder, [dm_file_base '.pdf']), 'ContentType', 'vector');
end

fprintf('\n');
print_block_title('GRAFICAS PHASE MARGIN vs Mst');

    create_metric_panel_figure( ...
        Plantas, idx_run, mst_grid, runPI, runPID, runSP, runMSP, styles, ...
        'PM_deg_opt', 'Phase Margin vs Mst', 'Phase Margin vs Robustez', 'Phase Margin, PM (deg)', ...
        'PhaseMargin_vs_Mst', false, fig_w_cm, fig_h_cm, font_name_axes, font_name_title, ...
        font_size_axes, font_size_title, font_size_legend, line_w, marker_sz, ...
        grid_alpha_major, grid_alpha_minor, legend_location_single, legend_num_cols, ...
        export_pub_figs, pub_fig_folder, fig_res_dpi, 'static_markers');

fprintf('\n');
print_block_title('GRAFICAS PARAMETROS OPTIMOS vs Mst');

create_params_figures( ...
    Plantas, idx_run, mst_grid, runPI, runPID, runSP, runMSP, styles, ...
    fig_w_cm, fig_h_cm, font_name_axes, font_name_title, ...
    font_size_axes, font_size_title, font_size_legend, line_w, marker_sz, ...
    grid_alpha_major, grid_alpha_minor, legend_location_single, ...
    export_pub_figs, pub_fig_folder, fig_res_dpi, 'static_markers');

plot_extra_figs = input('Deseas imprimir tambien las graficas de anexo (BW, tracking e IAE)? (1=SI, 0=NO, default=0): ');
if isempty(plot_extra_figs), plot_extra_figs = 0; end

if plot_extra_figs == 1
    fprintf('\n');
    print_block_title('GRAFICAS ANEXO');

    create_metric_panel_figure( ...
        Plantas, idx_run, mst_grid, runPI, runPID, runSP, runMSP, styles, ...
        'BW_cl_opt', 'Bandwidth vs Mst', 'Bandwidth vs Robustez', 'Bandwidth, BW_{cl}', ...
        'Bandwidth_vs_Mst', true, fig_w_cm, fig_h_cm, font_name_axes, font_name_title, ...
```

```
1076          font_size_axes, font_size_title, font_size_legend, line_w, marker_sz, ...
1077          grid_alpha_major, grid_alpha_minor, 'northwest', legend_num_cols, ...
1078          export_pub_figs, pub_fig_folder, fig_res_dpi, 'static_markers');
1079
1080      fprintf('\n');
1081      print_block_title('GRAFICAS COMPLEMENTARIAS DE TESIS');
1082
1083      for ip = idx_run
1084      figTrack = figure( ...
1085          'Name', sprintf('Mst_real vs Mst_obj - %s', Plantas(ip).id), ...
1086          'NumberTitle', 'off', ...
1087          'Color', 'w', ...
1088          'Units', 'centimeters', ...
1089          'Position', [2 2 fig_w_cm fig_h_cm]);
1090
1091      axT = axes(figTrack);
1092      hold(axT, 'on');
1093      format_metric_axis(axT, font_name_axes, font_size_axes, grid_alpha_major,
      grid_alpha_minor);
1094
1095      plot(axT, mst_grid, mst_grid, '--', 'Color', [0.35 0.35 0.35], 'LineWidth', 1.0,
      'DisplayName', 'Mst_real = Mst_obj');
1096
1097      if runPI && isfield(Plantas(ip),'PI_sweep')
1098          plot_sweep_metric(axT, Plantas(ip).PI_sweep, 'Mst_real', styles.PI, 'PI', line_w,
      marker_sz, 'static_markers');
1099      end
1100      if runPID && isfield(Plantas(ip),'PID_sweep')
1101          plot_sweep_metric(axT, Plantas(ip).PID_sweep, 'Mst_real', styles.PID, 'PID', line_w,
      marker_sz, 'static_markers');
1102      end
1103      if runSP && isfield(Plantas(ip),'SP_sweep')
1104          plot_sweep_metric(axT, Plantas(ip).SP_sweep, 'Mst_real', styles.SP, 'SP', line_w,
      marker_sz, 'static_markers');
1105      end
1106      if runMSP && isfield(Plantas(ip),'MSP_sweep')
1107          plot_sweep_metric(axT, Plantas(ip).MSP_sweep, 'Mst_real', styles.MSP, 'MSP', line_w,
      marker_sz, 'static_markers');
1108      end
1109
1110      xlabel(axT, 'Mst_obj', 'FontName', font_name_axes, 'FontSize', font_size_axes + 0.5,
      'Interpreter', 'none');
1111      ylabel(axT, 'Mst_real', 'FontName', font_name_axes, 'FontSize', font_size_axes + 0.5,
      'Interpreter', 'none');
1112      title(axT, plant_title_with_model(Plantas(ip)), 'FontName', font_name_title, 'FontSize',
      font_size_title - 0.1, 'FontWeight', 'normal', 'Interpreter', 'tex');
1113      xlim(axT, [min(mst_grid) max(mst_grid)]);
1114      ylim(axT, [min(mst_grid) max(mst_grid)]);
1115      apply_publication_legend(legend(axT, 'Location', 'southeast', 'Orientation', 'vertical'),
      font_name_axes, font_size_legend, 1);
1116
1117      if export_pub_figs
1118          drawnow;
1119          export_target_track = figTrack;
1120          if ~isgraphics(export_target_track)
1121              export_target_track = axT;
1122          end
1123          exportgraphics(export_target_track, fullfile(pub_fig_folder,
      sprintf('Mst_tracking_%s.png', Plantas(ip).id)), 'Resolution', fig_res_dpi);
1124          exportgraphics(export_target_track, fullfile(pub_fig_folder,
      sprintf('Mst_tracking_%s.pdf', Plantas(ip).id)), 'ContentType', 'vector');
1125      end
1126
1127      figIAE = figure( ...
1128          'Name', sprintf('IAE vs Mst - %s', Plantas(ip).id), ...
```

```
1129          'NumberTitle', 'off', ...
1130          'Color', 'w', ...
1131          'Units', 'centimeters', ...
1132          'Position', [2 2 fig_w_cm fig_h_cm * 1.15]);
1133
1134      tlI = tiledlayout(figIAE, 2, 1, 'TileSpacing', 'compact', 'Padding', 'compact');
1135      title(tlI, sprintf('%s | Desempeno por perturbacion',
      plant_title_with_model(Plantas(ip))), 'FontName', font_name_title, 'FontSize',
      font_size_title - 0.1, 'FontWeight', 'normal', 'Interpreter', 'tex');
1136
1137      axIdo = nexttile(tlI, 1); hold(axIdo, 'on'); box(axIdo, 'on'); grid(axIdo, 'on');
1138      axIdi = nexttile(tlI, 2); hold(axIdi, 'on'); box(axIdi, 'on'); grid(axIdi, 'on');
1139      format_metric_axis(axIdo, font_name_axes, font_size_axes, grid_alpha_major,
      grid_alpha_minor);
1140      format_metric_axis(axIdi, font_name_axes, font_size_axes, grid_alpha_major,
      grid_alpha_minor);
1141
1142      if runPI && isfield(Plantas(ip),'PI_sweep')
1143          plot_sweep_metric(axIdo, Plantas(ip).PI_sweep, 'IAEdo_opt', styles.PI, 'PI', line_w,
      marker_sz, 'static_markers');
1144          plot_sweep_metric(axIdi, Plantas(ip).PI_sweep, 'IAEdi_opt', styles.PI, 'PI', line_w,
      marker_sz, 'static_markers');
1145      end
1146      if runPID && isfield(Plantas(ip),'PID_sweep')
1147          plot_sweep_metric(axIdo, Plantas(ip).PID_sweep, 'IAEdo_opt', styles.PID, 'PID',
      line_w, marker_sz, 'static_markers');
1148          plot_sweep_metric(axIdi, Plantas(ip).PID_sweep, 'IAEdi_opt', styles.PID, 'PID',
      line_w, marker_sz, 'static_markers');
1149      end
1150      if runSP && isfield(Plantas(ip),'SP_sweep')
1151          plot_sweep_metric(axIdo, Plantas(ip).SP_sweep, 'IAEdo_opt', styles.SP, 'SP', line_w,
      marker_sz, 'static_markers');
1152          plot_sweep_metric(axIdi, Plantas(ip).SP_sweep, 'IAEdi_opt', styles.SP, 'SP', line_w,
      marker_sz, 'static_markers');
1153      end
1154      if runMSP && isfield(Plantas(ip),'MSP_sweep')
1155          plot_sweep_metric(axIdo, Plantas(ip).MSP_sweep, 'IAEdo_opt', styles.MSP, 'MSP',
      line_w, marker_sz, 'static_markers');
1156          plot_sweep_metric(axIdi, Plantas(ip).MSP_sweep, 'IAEdi_opt', styles.MSP, 'MSP',
      line_w, marker_sz, 'static_markers');
1157      end
1158
1159      ylabel(axIdo, 'IAEdo');
1160      ylabel(axIdi, 'IAEdi');
1161      xlabel(axIdi, 'Mst', 'Interpreter', 'none');
1162      title(axIdo, 'Perturbacion de salida');
1163      title(axIdi, 'Perturbacion de entrada');
1164      xlim(axIdo, [min(mst_grid) max(mst_grid)]);
1165      xlim(axIdi, [min(mst_grid) max(mst_grid)]);
1166      apply_publication_legend(legend(axIdo, 'Location', legend_location_single, 'Orientation',
      'vertical'), font_name_axes, font_size_legend, 1);
1167
1168      if export_pub_figs
1169          drawnow;
1170          export_target_iae = tlI;
1171          if ~isgraphics(export_target_iae)
1172              export_target_iae = figIAE;
1173          end
1174          exportgraphics(export_target_iae, fullfile(pub_fig_folder,
      sprintf('IAE_vs_Mst_%s.png', Plantas(ip).id)), 'Resolution', fig_res_dpi);
1175          exportgraphics(export_target_iae, fullfile(pub_fig_folder,
      sprintf('IAE_vs_Mst_%s.pdf', Plantas(ip).id)), 'ContentType', 'vector');
1176      end
1177      end
1178  else
```

```
1179      fprintf('Se omitieron las graficas adicionales; se imprimieron J, DM, PM y parametros
      optimos vs Mst.\n');
1180  end
1181
1182  %% ==================== 5. EXPORT PARA TESIS ====================
1183  fprintf('\n');
1184  print_block_title('EXPORT PARA TESIS');
1185
1186  if ~exist('run_stamp', 'var') || isempty(run_stamp)
1187      run_stamp = datestr(now, 'yyyymmdd_HHMMSS');
1188  end
1189  if ~exist('project_root', 'var') || isempty(project_root)
1190      project_root = resolve_script_root();
1191  end
1192  if ~exist('thesis_out_root', 'var') || isempty(thesis_out_root)
1193      thesis_out_root = fullfile(project_root, 'salida_tesis');
1194  end
1195  if ~exist('pub_fig_folder', 'var') || isempty(pub_fig_folder)
1196      pub_fig_folder = fullfile(thesis_out_root, 'figuras');
1197  end
1198  if ~exist('csv_out_folder', 'var') || isempty(csv_out_folder)
1199      csv_out_folder = fullfile(thesis_out_root, 'tablas');
1200  end
1201  if ~exist('meta_out_folder', 'var') || isempty(meta_out_folder)
1202      meta_out_folder = fullfile(thesis_out_root, 'meta');
1203  end
1204  if ~exist('xlsx_book_file', 'var') || isempty(xlsx_book_file)
1205      xlsx_book_file = fullfile(csv_out_folder, sprintf('resultados_corrida_%s.xlsx',
      run_stamp));
1206  end
1207  if ~exist('annex_out_folder', 'var') || isempty(annex_out_folder)
1208      annex_out_folder = fullfile(thesis_out_root, 'anexos_busqueda');
1209  end
1210  if ~exist('annex_xlsx_file', 'var') || isempty(annex_xlsx_file)
1211      annex_xlsx_file = fullfile(annex_out_folder, sprintf('anexo_busqueda_por_planta_%s.xlsx',
      run_stamp));
1212  end
1213  if ~exist('final_report_file', 'var') || isempty(final_report_file)
1214      final_report_file = fullfile(annex_out_folder, sprintf('reporte_final_busqueda_%s.txt',
      run_stamp));
1215  end
1216  if ~exist('workspace_mat_file', 'var') || isempty(workspace_mat_file)
1217      workspace_mat_file = fullfile(meta_out_folder, sprintf('workspace_corrida_%s.mat',
      run_stamp));
1218  end
1219  if ~exist(csv_out_folder, 'dir')
1220      mkdir(csv_out_folder);
1221  end
1222  if ~exist(meta_out_folder, 'dir')
1223      mkdir(meta_out_folder);
1224  end
1225  if export_annex_reports && ~exist(annex_out_folder, 'dir')
1226      mkdir(annex_out_folder);
1227  end
1228
1229  T_base  = build_base_export_table(Plantas, idx_run, baseMode); % Tabla de pesos base por
      planta
1230  T_sweep = build_sweep_export_table(Plantas, idx_run, runPI, runPID, runSP, runMSP); % Tabla
      resumen del barrido
1231  T_freq  = build_sweep_freq_export_table(Plantas, idx_run, runPI, runPID, runSP, runMSP); %
      Tabla de margenes y BW
1232  T_extra = build_extraordinary_export_table(Plantas, idx_run, runPI, runPID, runSP, runMSP); %
      Casos sin solucion o extraordinarios
1233  T_summary = build_sweep_summary_table(Plantas, idx_run, runPI, runPID, runSP, runMSP);
1234  T_winner  = build_sweep_winner_table(T_summary);
```

```
if ~exist('T_search_report', 'var') || isempty(T_search_report)
    T_search_report = build_search_report_table(Plantas, idx_run, runPI, runPID, runSP, runMSP, report_mst_ref);
end
if ~exist('T_search_winner', 'var') || isempty(T_search_winner)
    T_search_winner = build_search_winner_report_table(T_search_report);
end

if print_export_tables
    fprintf('Reporte final para anexo ya impreso en la seccion REPORTE FINAL PARA ANEXO.\n');
end

if export_results_xlsx
    workbook_tables = { ...
        T_base, 'Pesos base'; ...
        T_sweep, 'Barrido completo'; ...
        T_freq, 'Margenes y BW'; ...
        T_extra, 'Casos extraordinarios'; ...
        T_summary, 'Resumen tesis'; ...
        T_winner, 'Ganadores por planta'; ...
        T_search_report, 'Reporte busqueda'; ...
        T_search_winner, 'Ganadores busqueda'};

    xlsx_book_file = write_tables_to_workbook_safely(workbook_tables, xlsx_book_file);

    fprintf('Workbook XLSX exportado en:\n  %s\n', xlsx_book_file);
    fprintf('Sheets: Pesos base | Barrido completo | Margenes y BW | Casos extraordinarios | Resumen tesis | Ganadores por planta | Reporte busqueda | Ganadores busqueda\n');
end

if export_results_csv
    writetable(T_base,  fullfile(csv_out_folder, 'tabla_pesos_base.csv'), 'Delimiter', csv_delimiter);
    writetable(T_sweep, fullfile(csv_out_folder, 'tabla_sweep_resumen.csv'), 'Delimiter', csv_delimiter);
    writetable(T_freq,  fullfile(csv_out_folder, 'tabla_margenes_bw.csv'), 'Delimiter', csv_delimiter);
    writetable(T_extra, fullfile(csv_out_folder, 'tabla_casos_extraordinarios.csv'), 'Delimiter', csv_delimiter);
    writetable(T_summary, fullfile(csv_out_folder, 'tabla_resumen_tesis.csv'), 'Delimiter', csv_delimiter);
    writetable(T_winner,  fullfile(csv_out_folder, 'tabla_ganadores_planta.csv'), 'Delimiter', csv_delimiter);
    writetable(T_search_report, fullfile(csv_out_folder, 'tabla_reporte_busqueda.csv'), 'Delimiter', csv_delimiter);
    writetable(T_search_winner, fullfile(csv_out_folder, 'tabla_ganadores_busqueda.csv'), 'Delimiter', csv_delimiter);

    fprintf('CSV exportados en: %s\n', csv_out_folder);
    fprintf('  - tabla_pesos_base.csv\n');
    fprintf('  - tabla_sweep_resumen.csv\n');
    fprintf('  - tabla_margenes_bw.csv\n');
    fprintf('  - tabla_casos_extraordinarios.csv\n');
    fprintf('  - tabla_resumen_tesis.csv\n');
    fprintf('  - tabla_ganadores_planta.csv\n');
    fprintf('  - tabla_reporte_busqueda.csv\n');
    fprintf('  - tabla_ganadores_busqueda.csv\n');
end

if export_annex_reports
    annex_xlsx_file = write_search_annex_reports( ...
        Plantas, idx_run, T_search_report, T_sweep, ...
        code_version, baseCtrl, baseMode, ctrls_sel_txt, Mst_base, mst_min, mst_max, mst_step, mst_grid, ...
```

```
        refinamientos, h_1_vals, refinamientos_pid, h_1_vals_pid, edge_mult, h_relax_factor, ...
        scan_paso_global, region_gap_mult, max_regions_keep, region_min_pts, margen_mult, max_empty_iters, ...
        kc_bounds_global, ti_bounds_global, td_bounds_global, ...
        use_exact_mst, w_mst, plot_j_hide_above, report_mst_ref, ...
        annex_out_folder, annex_xlsx_file);

    fprintf('Anexos de busqueda por planta exportados en:\n  %s\n', annex_out_folder);
    fprintf('Workbook de anexos:\n  %s\n', annex_xlsx_file);
end

if ~export_results_csv
    fprintf('Export CSV desactivado.\n');
end
if ~export_results_xlsx
    fprintf('Export XLSX desactivado.\n');
end

if export_metadata_txt
    meta_lines = build_metadata_lines( ...
        code_version, baseCtrl, baseMode, ctrls_sel_txt, plants_sel_txt, ...
        Mst_base, mst_min, mst_max, mst_step, mst_grid, ...
        scan_paso_global, ...
        use_exact_mst, save_eval_detail, show_iter_status, export_pub_figs, export_results_csv, export_results_xlsx, save_run_workspace, print_export_tables, csv_delimiter, thesis_mode, ...
        pub_fig_folder, csv_out_folder, xlsx_book_file, workspace_mat_file); % Metadata completa de la corrida

    write_text_lines(fullfile(meta_out_folder, 'run_metadata.txt'), meta_lines);
    fprintf('Metadata exportada en: %s\n', fullfile(meta_out_folder, 'run_metadata.txt'));
else
    fprintf('Export metadata desactivado.\n');
end

if save_run_workspace
    save(workspace_mat_file, ...
        'Plantas', 'idx_run', 'ctrls', 'baseCtrl', 'baseMode', ...
        'Mst_base', 'mst_min', 'mst_max', 'mst_step', 'mst_grid', ...
        'T_base', 'T_sweep', 'T_freq', 'T_extra', 'T_summary', 'T_winner', ...
        'T_search_report', 'T_search_winner', ...
        'code_version', 'run_stamp', ...
        'pade_order', 'N_der', 'Tf_min', 'Go', 'Gi', 'T_end', 'dt', ...
        '-v7.3');
    fprintf('Workspace de la corrida exportado en:\n  %s\n', workspace_mat_file);
else
    fprintf('Guardado de workspace desactivado.\n');
end

%% ==================== 6. EVALUACION TEMPORAL A Mst FIJO ====================
if ~exist('run_time_eval_stage', 'var') || isempty(run_time_eval_stage)
    run_time_eval_stage = true;
end
if ~exist('ask_mst_ref_eval', 'var') || isempty(ask_mst_ref_eval)
    ask_mst_ref_eval = true;
end
if ~exist('mst_ref_eval', 'var') || isempty(mst_ref_eval)
    mst_ref_eval = 1.6;
end
if ~exist('allow_far_mst_eval', 'var') || isempty(allow_far_mst_eval)
    allow_far_mst_eval = false;
end
if ~exist('settling_p_time', 'var') || isempty(settling_p_time)
    settling_p_time = 2;
```

```
1348  end
1349  if ~exist('export_time_eval_figs', 'var') || isempty(export_time_eval_figs)
1350      export_time_eval_figs = true;
1351  end
1352  if ~exist('export_time_eval_tables', 'var') || isempty(export_time_eval_tables)
1353      export_time_eval_tables = true;
1354  end
1355  if ~exist('deadtime_err_list_time', 'var') || isempty(deadtime_err_list_time)
1356      deadtime_err_list_time = [-0.30 -0.20 -0.10 0 0.10 0.20 0.30];
1357  end
1358  if ~exist('T_end', 'var') || isempty(T_end)
1359      T_end = 30;
1360  end
1361  if ~exist('dt', 'var') || isempty(dt)
1362      dt = 0.01;
1363  end
1364  if ~exist('t', 'var') || isempty(t)
1365      t = 0:dt:T_end;
1366  end
1367  if ~exist('Go', 'var') || isempty(Go)
1368      Go = 1;
1369  end
1370  if ~exist('Gi', 'var') || isempty(Gi)
1371      Gi = 1;
1372  end
1373  if ~exist('pade_order', 'var') || isempty(pade_order)
1374      pade_order = 4;
1375  end
1376  if ~exist('N_der', 'var') || isempty(N_der)
1377      N_der = 10;
1378  end
1379  if ~exist('Tf_min', 'var') || isempty(Tf_min)
1380      Tf_min = 1e-3;
1381  end
1382  if ~exist('thesis_out_root', 'var') || isempty(thesis_out_root)
1383      thesis_out_root = 'salida_tesis';
1384  end
1385  if ~exist('run_stamp', 'var') || isempty(run_stamp)
1386      run_stamp = datestr(now, 'yyyymmdd_HHMMSS');
1387  end
1388  if ~exist('line_w', 'var') || isempty(line_w)
1389      line_w = 1.80;
1390  end
1391  if ~exist('styles', 'var') || isempty(styles)
1392      styles = get_thesis_plot_styles();
1393  end
1394
1395  if run_time_eval_stage
1396      fprintf('\n');
1397      print_block_title('EVALUACION TEMPORAL A Mst FIJO');
1398
1399      if ~exist('Plantas', 'var') || isempty(Plantas)
1400          error(['No existe la variable Plantas en memoria. ' ...
1401              'Corre primero la busqueda J vs Mst o carga un workspace previo antes de ejecutar
      esta seccion.']);
1402      end
1403      if ~exist('idx_run', 'var') || isempty(idx_run)
1404          idx_run = 1:numel(Plantas);
1405      end
1406
1407      [runPI_td, runPID_td, runSP_td, runMSP_td] =
      infer_run_flags_from_loaded_sweeps_td(Plantas, idx_run);
1408      mst_eval_options = collect_available_mst_options_td(Plantas, idx_run, runPI_td,
      runPID_td, runSP_td, runMSP_td);
1409      if isempty(mst_eval_options)
```

```
1410          error('No se encontraron valores de Mst_obj en los sweeps cargados para la etapa
      temporal.');
1411      end
1412
1413      try
1414          [mst_ref_eval_use, mst_ref_default] = resolve_mst_eval_choice_td(mst_ref_eval,
      mst_eval_options, ask_mst_ref_eval);
1415      catch err
1416          if strcmp(err.identifier, 'CodigoTesis:CanceladoPorUsuario')
1417              fprintf('\n%s\n', err.message);
1418              fprintf('Se cancelo la etapa de evaluacion temporal a Mst fijo. Fin de la
      ejecucion.\n');
1419              return;
1420          end
1421          rethrow(err);
1422      end
1423
1424      fprintf('Opciones de Mst disponibles: %s\n', vec2str(mst_eval_options, '%.4f'));
1425      fprintf('Mst default de evaluacion: %.4f\n', mst_ref_default);
1426      fprintf('Mst seleccionado: %.4f\n', mst_ref_eval_use);
1427      fprintf('Permitir FAR_MST: %d | Banda settling: %.2f%%\n', allow_far_mst_eval,
      settling_p_time);
1428
1429      if ~exist('run_stamp', 'var') || isempty(run_stamp)
1430          run_stamp = datestr(now, 'yyyymmdd_HHMMSS');
1431      end
1432      if ~exist('project_root', 'var') || isempty(project_root)
1433          project_root = resolve_script_root();
1434      end
1435      if ~exist('thesis_out_root', 'var') || isempty(thesis_out_root)
1436          thesis_out_root = fullfile(project_root, 'salida_tesis');
1437      end
1438
1439      time_eval_root = fullfile(thesis_out_root, 'mst_fijo_eval');
1440      time_eval_fig_folder = fullfile(time_eval_root, 'figuras');
1441      time_eval_tab_folder = fullfile(time_eval_root, 'tablas');
1442      time_eval_meta_folder = fullfile(time_eval_root, 'meta');
1443      time_eval_xlsx_file = fullfile(time_eval_tab_folder, sprintf('tabla_mst_fijo_%s.xlsx',
      run_stamp));
1444      time_eval_meta_file = fullfile(time_eval_meta_folder, sprintf('metadata_mst_fijo_%s.txt',
      run_stamp));
1445
1446      if export_time_eval_figs && ~exist(time_eval_fig_folder, 'dir')
1447          mkdir(time_eval_fig_folder);
1448      end
1449      if export_time_eval_tables && ~exist(time_eval_tab_folder, 'dir')
1450          mkdir(time_eval_tab_folder);
1451      end
1452      if ~exist(time_eval_meta_folder, 'dir')
1453          mkdir(time_eval_meta_folder);
1454      end
1455
1456      [Plantas_time_eval, T_fixed_pick] = build_fixed_pick_table_td( ...
1457          Plantas, idx_run, runPI_td, runPID_td, runSP_td, runMSP_td, mst_ref_eval_use,
      allow_far_mst_eval);
1458
1459      fprintf('\n6.1 Seleccion a Mst fijo\n');
1460      disp(T_fixed_pick);
1461
1462      T_time_metrics = build_time_metrics_table_td( ...
1463          Plantas_time_eval, idx_run, runPI_td, runPID_td, runSP_td, runMSP_td, ...
1464          Go, Gi, t, dt, settling_p_time, pade_order, N_der, Tf_min);
1465      fprintf('Nota tie-in: J temporal se recalcula con IAEdo/IAEdi de esta etapa y pesos base
      cargados; J_sweep se conserva para auditoria.\n');
```

```
    [T_time_ref_all, T_time_di_all, T_time_do_all] = build_time_scenario_tables_td(T_time_metrics, '', true);
    T_time_winner = build_time_winner_table_td(T_time_metrics);
    T_deadtime_eval = build_deadtime_error_table_td( ...
        Plantas_time_eval, idx_run, runPI_td, runPID_td, runSP_td, runMSP_td, Go, Gi, t, settling_p_time, ...
        pade_order, N_der, Tf_min, deadtime_err_list_time);

    fprintf('\n6.2 Indicadores temporales\n');
    for k = 1:numel(idx_run)
        ip = idx_run(k);
        [T_time_ref_plant, T_time_di_plant, T_time_do_plant] = build_time_scenario_tables_td(T_time_metrics, Plantas_time_eval(ip).id, false);
        fprintf('\n6.2.%d Planta %s - Referencia\n', k, Plantas_time_eval(ip).id);
        disp(T_time_ref_plant);
        fprintf('\n6.2.%d Planta %s - Perturbacion de entrada\n', k, Plantas_time_eval(ip).id);
        disp(T_time_di_plant);
        fprintf('\n6.2.%d Planta %s - Perturbacion de salida\n', k, Plantas_time_eval(ip).id);
        disp(T_time_do_plant);
    end

    fprintf('\n6.3 Ganadores temporales por planta\n');
    disp(T_time_winner);

    fprintf('\n6.4 Evaluacion con error en tiempo de retardo\n');
    disp(T_deadtime_eval);

    fprintf('\n6.4.1 Configuracion del tiempo visible en graficas temporales\n');
    fprintf('Nota: este ajuste solo cambia lo que se muestra en las figuras; no cambia IAE ni las metricas ya calculadas.\n');
    time_plot_short_by_plant = 10 * ones(numel(idx_run), 1);
    for k = 1:numel(idx_run)
        ip = idx_run(k);
        prompt_time = sprintf('Tiempo visible para paneles temporales de %s (s, default=10): ', Plantas_time_eval(ip).id);
        val_time = input(prompt_time);
        if isempty(val_time)
            val_time = 10;
        end
        if ~isscalar(val_time) || ~isfinite(val_time) || val_time <= 0
            error('El tiempo visible por planta debe ser un escalar positivo.');
        end
        time_plot_short_by_plant(k) = val_time;
    end

    fprintf('\n6.5 Graficas temporales por planta\n');
    for k = 1:numel(idx_run)
        ip = idx_run(k);
        plot_time_domain_plant_figure_td( ...
            Plantas_time_eval(ip), runPI_td, runPID_td, runSP_td, runMSP_td, ...
            styles, Go, Gi, t, dt, pade_order, N_der, Tf_min, ...
            max(1.15, line_w - 0.20), export_time_eval_figs, time_eval_fig_folder, mst_ref_eval_use, ...
            time_plot_short_by_plant(k));
    end

    fprintf('\n6.6 Graficas de error en tiempo de retardo\n');
    fprintf('Se omiten las figuras individuales por controlador; se mantienen las comparativas conjuntas.\n');

    fprintf('\n6.7 Graficas comparativas por %% de error\n');
    if runPI_td || runPID_td || runSP_td || runMSP_td
        plot_deadtime_error_response_panels_td( ...
```

```
1522              Plantas_time_eval, idx_run, runPI_td, runPID_td, runSP_td, runMSP_td, styles, Go,
      Gi, dt, pade_order, N_der, Tf_min, ...
1523              deadtime_err_list_time, max(1.05, line_w - 0.30), export_time_eval_figs,
      time_eval_fig_folder, mst_ref_eval_use, ...
1524              time_plot_short_by_plant);
1525      else
1526          fprintf('Se omiten porque no hay controladores disponibles para esta comparacion.\n');
1527      end
1528
1529      if export_time_eval_tables
1530          time_workbook_tables = { ...
1531              T_fixed_pick, 'Seleccion_Mst_fijo'; ...
1532              T_time_ref_all, 'Tiempo_referencia'; ...
1533              T_time_di_all, 'Tiempo_di'; ...
1534              T_time_do_all, 'Tiempo_do'; ...
1535              T_time_winner, 'Ganadores_tiempo'; ...
1536              T_deadtime_eval, 'Error_tiempo_muerto'};
1537          time_eval_xlsx_file = write_tables_to_workbook_safely(time_workbook_tables,
      time_eval_xlsx_file);
1538          fprintf('\nWorkbook Mst fijo exportado en:\n  %s\n', time_eval_xlsx_file);
1539      end
1540
1541      time_meta_lines = build_time_eval_metadata_lines_td( ...
1542          mst_ref_eval_use, allow_far_mst_eval, settling_p_time, T_end, dt, Go, Gi, pade_order,
      N_der, Tf_min, ...
1543          deadtime_err_list_time, idx_run, Plantas, runPI_td, runPID_td, runSP_td, runMSP_td,
      ...
1544          time_eval_xlsx_file, time_eval_fig_folder);
1545      write_text_lines(time_eval_meta_file, time_meta_lines);
1546      fprintf('Metadata de Mst fijo exportada en:\n  %s\n', time_eval_meta_file);
1547  end
1548
1549  % Las utilidades complementarias de comparacion y auditoria permanecen
1550  % disponibles como funciones auxiliares, aunque no forman parte del flujo
1551  % principal de ejecucion utilizado en esta version consolidada.
1552
1553  %% ==================== FUNCIONES LOCALES ====================
1554  %  Organizacion del bloque de funciones:
1555  %  - Construccion de mallas y normalizacion de parametros
1556  %  - Barrido y refinamiento sobre Mst
1557  %  - Evaluacion de robustez, IAE y margenes
1558  %  - Reportes, tablas y exportacion
1559  %  - Etapa temporal a Mst fijo
1560  %  - Modelos equivalentes SP y MSP
1561  %
1562  %  Las funciones se conservan en un unico archivo para facilitar la
1563  %  trazabilidad de la implementacion y la reproduccion de resultados.
1564
1565  function mst_grid = build_mst_grid(a, b, step)
1566      mst_grid = a:step:b;
1567      if isempty(mst_grid) || abs(mst_grid(end) - b) > 1e-12
1568          mst_grid = [mst_grid, b];
1569      end
1570      mst_grid = unique(round(mst_grid, 10));
1571  end
1572
1573  function vals = build_aligned_axis(range_in, global_range, paso)
1574      lo = max(range_in(1), global_range(1));
1575      hi = min(range_in(2), global_range(2));
1576      if hi < lo
1577          vals = [];
1578          return;
1579      end
1580
1581      base = global_range(1);
```

```
1582      startv = base + floor((lo - base)/paso) * paso;
1583      endv   = base + ceil((hi - base)/paso)  * paso;
1584
1585      startv = max(startv, global_range(1));
1586      endv   = min(endv, global_range(2));
1587
1588      if endv < startv
1589          vals = [];
1590          return;
1591      end
1592
1593      vals = startv:paso:endv;
1594      vals = unique(round(vals, 10));
1595  end
1596
1597  function vals = build_search_ti_axis(ctrl_key, plant_tipo, plant_id, theta0, range_in,
      global_range, paso)
1598      if ~ismember(upper(ctrl_key), {'SP','MSP'})
1599          vals = build_aligned_axis(range_in, global_range, paso);
1600          return;
1601      end
1602
1603      [predictor_mode, ti_fixed] = get_predictor_paper_mode(ctrl_key, plant_tipo, plant_id,
      theta0);
1604      switch predictor_mode
1605          case 'P_ONLY'
1606              vals = 0;
1607
1608          case 'FIXED_TI'
1609              lo = max(range_in(1), global_range(1));
1610              hi = min(range_in(2), global_range(2));
1611              if ti_fixed < lo || ti_fixed > hi
1612                  vals = [];
1613              else
1614                  vals = ti_fixed;
1615              end
1616
1617          otherwise
1618              vals = build_aligned_axis(range_in, global_range, paso);
1619      end
1620
1621      vals = unique(round(vals, 10));
1622  end
1623
1624  function [predictor_mode, ti_fixed] = get_predictor_paper_mode(ctrl_key, plant_tipo,
      plant_id, theta0)
1625      if ismember(upper(ctrl_key), {'SP','MSP'}) && strcmpi(plant_tipo, 'integrador')
1626          predictor_mode = 'P_ONLY';
1627          ti_fixed = 0;
1628          return;
1629      end
1630
1631      if strcmpi(ctrl_key, 'MSP')
1632          [predictor_mode, ti_fixed] = get_msp_paper_mode(plant_tipo, plant_id, theta0);
1633      else
1634          predictor_mode = 'FREE_PI';
1635          ti_fixed = NaN;
1636      end
1637  end
1638
1639  function [msp_mode, ti_fixed] = get_msp_paper_mode(plant_tipo, plant_id, theta0)
1640      switch lower(plant_tipo)
1641          case 'integrador'
1642              msp_mode = 'P_ONLY';
1643              ti_fixed = 0;
```

```
1644
1645          case 'inestable'
1646              % Para el barrido volvemos a buscar Ti libremente.
1647              % La sintonia analitica del paper se calcula aparte como referencia.
1648              msp_mode = 'FREE_PI';
1649              ti_fixed = NaN;
1650
1651          otherwise
1652              msp_mode = 'FREE_PI';
1653              ti_fixed = NaN;
1654      end
1655  end
1656
1657  function ti = normalize_search_ti(ctrl_key, plant_tipo, plant_id, theta0, ti_raw)
1658      if ~ismember(upper(ctrl_key), {'SP','MSP'})
1659          ti = max(ti_raw, 1e-3);
1660          return;
1661      end
1662
1663      [predictor_mode, ti_fixed] = get_predictor_paper_mode(ctrl_key, plant_tipo, plant_id,
      theta0);
1664      switch predictor_mode
1665          case 'P_ONLY'
1666              ti = 0;
1667
1668          case 'FIXED_TI'
1669              ti = ti_fixed;
1670
1671          otherwise
1672              ti = max(ti_raw, 1e-3);
1673      end
1674  end
1675
1676  function C = build_controller_tf(ctrl_key, kc, ti, td, plant_tipo, plant_id, theta0, N_der,
      Tf_min)
1677      s = tf('s');
1678
1679      switch upper(ctrl_key)
1680          case 'PI'
1681              C = kc * (1 + 1/(s * max(ti, 1e-6)));
1682
1683          case 'PID'
1684              Tf = max(td / N_der, Tf_min);
1685              C = kc * (1 + 1/(s * max(ti, 1e-6))) * ((s * max(td, 0) + 1) / (s * Tf + 1));
1686
1687          case 'SP'
1688              [predictor_mode, ~] = get_predictor_paper_mode(ctrl_key, plant_tipo, plant_id,
      theta0);
1689              if strcmpi(predictor_mode, 'P_ONLY')
1690                  C = tf(kc, 1);
1691              else
1692                  C = kc * (1 + 1/(s * max(ti, 1e-6)));
1693              end
1694
1695          case 'MSP'
1696              [predictor_mode, ti_fixed] = get_predictor_paper_mode(ctrl_key, plant_tipo,
      plant_id, theta0);
1697              switch predictor_mode
1698                  case 'P_ONLY'
1699                      C = tf(kc, 1);
1700
1701                  case 'FIXED_TI'
1702                      C = kc * (1 + 1/(s * max(ti_fixed, 1e-6)));
1703
1704                  otherwise
```

```
                    C = kc * (1 + 1/(s * max(ti, 1e-6)));
            end

        otherwise
            error('Controlador no reconocido: %s', ctrl_key);
    end
end

function sweep = run_mst_sweep(ctrl_key, mst_grid, P, P0, theta0, plant_tipo, plant_id, ...
    refinamientos, h_1_vals, edge_mult, h_relax_factor, ...
    scan_paso_global, region_gap_mult, max_regions_keep, region_min_pts, margen_mult, ...
    kc_bounds_global, ti_bounds_global, td_bounds_global, ...
    Go, Gi, t, pade_order, N_der, Tf_min, ...
    iaedo0_base, iaedi0_base, plant_label, ...
    use_exact_mst, w_mst, save_eval_detail, max_empty_iters)

    nM = numel(mst_grid);
    sweep = struct();
    sweep.ctrl_key      = ctrl_key;
    sweep.Mst_obj       = mst_grid(:);
    sweep.status        = repmat({''}, nM, 1);
    sweep.nCand         = zeros(nM,1);
    sweep.Kc_opt        = NaN(nM,1);
    sweep.Ti_opt        = NaN(nM,1);
    sweep.Td_opt        = NaN(nM,1);
    sweep.Mst_real      = NaN(nM,1);
    sweep.Mst_err       = NaN(nM,1);
    sweep.Mst_tol_used  = NaN(nM,1);
    sweep.IAEdo_opt     = NaN(nM,1);
    sweep.IAEdi_opt     = NaN(nM,1);
    sweep.J_opt         = NaN(nM,1);
    sweep.scan_status   = repmat({''}, nM, 1);
    sweep.bounds_mode   = repmat({''}, nM, 1);
    sweep.iter_log      = cell(nM,1);

    prev_ok = false;
    prev_opt = struct();

    for im = 1:nM
        Mst_r = mst_grid(im);
        used_warm = false;
        iter_log = {};
        cand = struct('kc',[],'ti',[],'td',[],'mst',[],'tol_mst',[],'iaedo',[],'iaedi',
[],'ctrl_key',ctrl_key);

        if prev_ok
            warm_bounds = build_warm_bounds(ctrl_key, prev_opt, kc_bounds_global,
ti_bounds_global, td_bounds_global); % Busca primero cerca de la solucion anterior
            status_msg('RUN', sprintf('Planta %s | Ctrl %s | Mst=%.4f | WarmBounds', ...
                plant_label, ctrl_key_to_user(ctrl_key), Mst_r));

            [cand_warm, iter_log_warm, ref_status_warm] = refinement_over_bounds_list( ...
                ctrl_key, P, P0, theta0, plant_tipo, plant_id, {warm_bounds}, Mst_r, ...
                refinamientos, h_1_vals, edge_mult, h_relax_factor, ...
                kc_bounds_global, ti_bounds_global, td_bounds_global, ...
                Go, Gi, t, pade_order, N_der, Tf_min, plant_label, use_exact_mst, w_mst, ...
                save_eval_detail, max_empty_iters, iaedo0_base, iaedi0_base);

            if ~isempty(cand_warm.kc)
                cand = cand_warm;
                iter_log = iter_log_warm;
                sweep.bounds_mode{im} = 'WARM';
                sweep.scan_status{im} = ref_status_warm;
                used_warm = true;
```

```
1767                 status_msg('DONE', sprintf('Planta %s | Ctrl %s | Mst=%.4f | Warm success |
      nCand=%d', ...
1768                     plant_label, ctrl_key_to_user(ctrl_key), Mst_r, numel(cand.kc)));
1769             else
1770                 status_msg('WARN', sprintf('Planta %s | Ctrl %s | Mst=%.4f | Warm failed', ...
1771                     plant_label, ctrl_key_to_user(ctrl_key), Mst_r));
1772             end
1773         end
1774
1775         if ~used_warm
1776             status_msg('RUN', sprintf('Planta %s | Ctrl %s | Mst=%.4f | GlobalScan', ...
1777                 plant_label, ctrl_key_to_user(ctrl_key), Mst_r));
1778
1779             scan_res = global_mst_map( ...
1780                 ctrl_key, P, P0, theta0, plant_tipo, plant_id, ...
1781                 scan_paso_global, kc_bounds_global, ti_bounds_global, td_bounds_global, ...
1782                 Go, Gi, pade_order, N_der, Tf_min, plant_label, use_exact_mst, w_mst); % Si
      no hay solucion previa, explora globalmente
1783
1784             print_global_scan_summary(scan_res);
1785
1786             [bounds_list, scan_status, ~] = select_bounds_from_scan( ...
1787                 scan_res, ctrl_key, Mst_r, h_1_vals(1), margen_mult, ...
1788                 kc_bounds_global, ti_bounds_global, td_bounds_global, ...
1789                 region_gap_mult, max_regions_keep, region_min_pts); % Selecciona regiones
      cercanas al Mst_obj deseado
1790
1791             sweep.scan_status{im} = scan_status;
1792             sweep.bounds_mode{im} = 'SCAN';
1793             print_selected_regions(bounds_list, scan_status, ctrl_key);
1794
1795             status_msg('RUN', sprintf('Planta %s | Ctrl %s | Mst=%.4f | Refinement', ...
1796                 plant_label, ctrl_key_to_user(ctrl_key), Mst_r));
1797
1798             [cand, iter_log, ref_status] = refinement_over_bounds_list( ...
1799                 ctrl_key, P, P0, theta0, plant_tipo, plant_id, bounds_list, Mst_r, ...
1800                 refinamientos, h_1_vals, edge_mult, h_relax_factor, ...
1801                 kc_bounds_global, ti_bounds_global, td_bounds_global, ...
1802                 Go, Gi, t, pade_order, N_der, Tf_min, plant_label, use_exact_mst, w_mst, ...
1803                 save_eval_detail, max_empty_iters, iaedo0_base, iaedi0_base); % Refina las
      regiones prometedoras
1804
1805             sweep.scan_status{im} = sprintf('%s|%s', scan_status, ref_status);
1806         end
1807
1808         sweep.iter_log{im} = iter_log;
1809
1810         if isempty(cand.kc)
1811             sweep.status{im} = 'NO_CAND';
1812             sweep.nCand(im)  = 0;
1813             prev_ok = false;
1814             status_msg('WARN', sprintf('Planta %s | Ctrl %s | Mst=%.4f | status=NO_CAND', ...
1815                 plant_label, ctrl_key_to_user(ctrl_key), Mst_r));
1816             continue;
1817         end
1818
1819         sweep.nCand(im) = numel(cand.kc);
1820         if prev_ok
1821             prev_sel = prev_opt;
1822         else
1823             prev_sel = [];
1824         end
1825
1826         [opt_res, opt_status] = build_opt_result_with_external_weights( ...
```

```
1827            ctrl_key, cand, iaedo0_base, iaedi0_base, Mst_r, prev_sel); % Elige la mejor
    solucion usando la tolerancia de la ultima iteracion valida
1828
1829        sweep.status{im} = opt_status;
1830
1831        if ismember(opt_status, {'OK','FAR_MST'})
1832            sweep.Kc_opt(im)    = opt_res.kc_opt;
1833            sweep.Ti_opt(im)    = opt_res.ti_opt;
1834            sweep.Td_opt(im)    = opt_res.td_opt;
1835            sweep.Mst_real(im)  = opt_res.mst_opt;
1836            sweep.Mst_err(im)   = opt_res.mst_err;
1837            sweep.Mst_tol_used(im) = opt_res.mst_tol_used;
1838            sweep.IAEdo_opt(im) = opt_res.iaedo_opt;
1839            sweep.IAEdi_opt(im) = opt_res.iaedi_opt;
1840            sweep.J_opt(im)     = opt_res.J_opt;
1841
1842            if strcmp(opt_status,'OK')
1843                prev_ok = true;
1844                prev_opt = opt_res;
1845            else
1846                prev_ok = false;
1847            end
1848
1849            status_msg('DONE', sprintf('Planta %s | Ctrl %s | Mst=%.4f | status=%s | nCand=%d
    | J=%s | err=%s | tol=%s', ...
1850                plant_label, ctrl_key_to_user(ctrl_key), Mst_r, opt_status, sweep.nCand(im),
    ...
1851                fmt_num(sweep.J_opt(im)), fmt_num(sweep.Mst_err(im)),
    fmt_num(sweep.Mst_tol_used(im))));
1852        else
1853            prev_ok = false;
1854            status_msg('WARN', sprintf('Planta %s | Ctrl %s | Mst=%.4f | status=%s |
    nCand=%d', ...
1855                plant_label, ctrl_key_to_user(ctrl_key), Mst_r, opt_status, sweep.nCand(im)));
1856        end
1857    end
1858
1859    sweep = reselect_sweep_globally(sweep);
1860 end
1861
1862 function sweep = reselect_sweep_globally(sweep)
1863    mask_pool = isfinite(sweep.Kc_opt) & isfinite(sweep.Ti_opt) & isfinite(sweep.Td_opt) & ...
1864        isfinite(sweep.Mst_real) & isfinite(sweep.IAEdo_opt) & isfinite(sweep.IAEdi_opt) &
    isfinite(sweep.J_opt);
1865
1866    if ~any(mask_pool)
1867        return;
1868    end
1869
1870    if ~isfield(sweep, 'source_mst_obj')
1871        sweep.source_mst_obj = sweep.Mst_obj(:);
1872    end
1873    if ~isfield(sweep, 'source_idx')
1874        sweep.source_idx = (1:numel(sweep.Mst_obj)).';
1875    end
1876
1877    if ~isfield(sweep, 'local_selection')
1878        sweep.local_selection = struct();
1879        sweep.local_selection.status = sweep.status;
1880        sweep.local_selection.Kc_opt = sweep.Kc_opt;
1881        sweep.local_selection.Ti_opt = sweep.Ti_opt;
1882        sweep.local_selection.Td_opt = sweep.Td_opt;
1883        sweep.local_selection.Mst_real = sweep.Mst_real;
1884        sweep.local_selection.Mst_err = sweep.Mst_err;
1885        sweep.local_selection.Mst_tol_used = sweep.Mst_tol_used;
```

```
1886          sweep.local_selection.IAEdo_opt = sweep.IAEdo_opt;
1887          sweep.local_selection.IAEdi_opt = sweep.IAEdi_opt;
1888          sweep.local_selection.J_opt = sweep.J_opt;
1889      end
1890
1891      pool_idx = find(mask_pool);
1892      pool = struct();
1893      pool.kc    = sweep.Kc_opt(pool_idx);
1894      pool.ti    = sweep.Ti_opt(pool_idx);
1895      pool.td    = sweep.Td_opt(pool_idx);
1896      pool.mst   = sweep.Mst_real(pool_idx);
1897      pool.tol   = sweep.Mst_tol_used(pool_idx);
1898      pool.iaedo = sweep.IAEdo_opt(pool_idx);
1899      pool.iaedi = sweep.IAEdi_opt(pool_idx);
1900      pool.J     = sweep.J_opt(pool_idx);
1901      pool.src_mst_obj = sweep.Mst_obj(pool_idx);
1902      pool.src_idx = pool_idx;
1903
1904      prev_sel = [];
1905      for im = 1:numel(sweep.Mst_obj)
1906          target = sweep.Mst_obj(im);
1907          err = abs(pool.mst - target);
1908          min_err = min(err);
1909          mask_best_err = err <= (min_err + 1e-9);
1910
1911          kc_s     = pool.kc(mask_best_err);
1912          ti_s     = pool.ti(mask_best_err);
1913          td_s     = pool.td(mask_best_err);
1914          mst_s    = pool.mst(mask_best_err);
1915          tol_s    = pool.tol(mask_best_err);
1916          iaedo_s  = pool.iaedo(mask_best_err);
1917          iaedi_s  = pool.iaedi(mask_best_err);
1918          J_s      = pool.J(mask_best_err);
1919          err_s    = err(mask_best_err);
1920          src_mst_obj_s = pool.src_mst_obj(mask_best_err);
1921          src_idx_s     = pool.src_idx(mask_best_err);
1922
1923          cont_s = continuity_cost(sweep.ctrl_key, kc_s, ti_s, td_s, prev_sel);
1924          score = J_s + 1e-4 * cont_s;
1925          [~, idx_sel] = min(score);
1926
1927          sweep.Kc_opt(im)       = kc_s(idx_sel);
1928          sweep.Ti_opt(im)       = ti_s(idx_sel);
1929          sweep.Td_opt(im)       = td_s(idx_sel);
1930          sweep.Mst_real(im)     = mst_s(idx_sel);
1931          sweep.Mst_err(im)      = err_s(idx_sel);
1932          sweep.Mst_tol_used(im) = tol_s(idx_sel);
1933          sweep.IAEdo_opt(im)    = iaedo_s(idx_sel);
1934          sweep.IAEdi_opt(im)    = iaedi_s(idx_sel);
1935          sweep.J_opt(im)        = J_s(idx_sel);
1936          sweep.nCand(im)        = max(sweep.nCand(im), sum(mask_best_err));
1937          sweep.source_mst_obj(im) = src_mst_obj_s(idx_sel);
1938          sweep.source_idx(im)     = src_idx_s(idx_sel);
1939
1940          if err_s(idx_sel) <= (tol_s(idx_sel) + 1e-12)
1941              sweep.status{im} = 'OK';
1942          else
1943              sweep.status{im} = 'FAR_MST';
1944          end
1945
1946          prev_sel = struct( ...
1947              'kc_opt', sweep.Kc_opt(im), ...
1948              'ti_opt', sweep.Ti_opt(im), ...
1949              'td_opt', sweep.Td_opt(im));
1950      end
```

```
1951 end
1952
1953 function warm_bounds = build_warm_bounds(ctrl_key, prev_opt, kc_bounds_global,
     ti_bounds_global, td_bounds_global)
1954     warm_bounds = struct();
1955
1956     kc = prev_opt.kc_opt;
1957     ti = prev_opt.ti_opt;
1958     td = prev_opt.td_opt;
1959
1960     warm_bounds.kc = [max(kc-0.6, kc_bounds_global(1)), min(kc+0.6, kc_bounds_global(2))];
1961     warm_bounds.ti = [max(ti-0.8, ti_bounds_global(1)), min(ti+0.8, ti_bounds_global(2))];
1962
1963     if strcmpi(ctrl_key,'PID')
1964         warm_bounds.td = [max(td-0.5, td_bounds_global(1)), min(td+0.5, td_bounds_global(2))];
1965     else
1966         warm_bounds.td = [0 0];
1967     end
1968
1969     if diff(warm_bounds.kc) < 0.1
1970         c = mean(warm_bounds.kc);
1971         warm_bounds.kc = [max(c-0.2, kc_bounds_global(1)), min(c+0.2, kc_bounds_global(2))];
1972     end
1973     if diff(warm_bounds.ti) < 0.1
1974         c = mean(warm_bounds.ti);
1975         warm_bounds.ti = [max(c-0.2, ti_bounds_global(1)), min(c+0.2, ti_bounds_global(2))];
1976     end
1977     if strcmpi(ctrl_key,'PID') && diff(warm_bounds.td) < 0.1
1978         c = mean(warm_bounds.td);
1979         warm_bounds.td = [max(c-0.1, td_bounds_global(1)), min(c+0.1, td_bounds_global(2))];
1980     end
1981 end
1982
1983 function scan = global_mst_map(ctrl_key, P, P0, theta0, plant_tipo, plant_id, paso_scan, ...
1984     kc_bounds_global, ti_bounds_global, td_bounds_global, ...
1985     Go, Gi, pade_order, N_der, Tf_min, plant_label, use_exact_mst, w_mst)
1986
1987     parfor_min_grid = 200;
1988
1989     scan = struct();
1990     scan.ctrl_key = ctrl_key;
1991     scan.paso     = paso_scan;
1992     scan.status   = 'SCAN_OK';
1993
1994     status_msg('RUN', sprintf('Planta %s | Ctrl %s | GlobalScan paso=%.3f', ...
1995         plant_label, ctrl_key_to_user(ctrl_key), paso_scan));
1996
1997     kc_vals = build_aligned_axis(kc_bounds_global, kc_bounds_global, paso_scan);
1998     ti_vals = build_search_ti_axis(ctrl_key, plant_tipo, plant_id, theta0, ...
1999         ti_bounds_global, ti_bounds_global, paso_scan);
2000
2001     if strcmpi(ctrl_key,'PID')
2002         td_vals = build_aligned_axis(td_bounds_global, td_bounds_global, paso_scan);
2003         [Kc, Ti, Td] = ndgrid(kc_vals, ti_vals, td_vals);
2004         n_t = numel(Kc);
2005
2006         kc_m = zeros(n_t,1); ti_m = zeros(n_t,1); td_m = zeros(n_t,1);
2007         mst  = inf(n_t,1);
2008         es_estable = false(n_t,1);
2009
2010         if n_t >= parfor_min_grid
2011             parfor b = 1:n_t
2012                 kc = max(Kc(b), 1e-3);
2013                 ti = normalize_search_ti(ctrl_key, plant_tipo, plant_id, theta0, Ti(b));
2014                 td = max(Td(b), 0);
```

```
2015                 [es_estable(b), mst(b)] = eval_mst_controller( ...
2016                     ctrl_key, kc, ti, td, P, P0, theta0, plant_tipo, plant_id, ...
2017                     Go, Gi, pade_order, N_der, Tf_min, use_exact_mst, w_mst);
2018                 kc_m(b)=kc; ti_m(b)=ti; td_m(b)=td;
2019             end
2020         else
2021             for b = 1:n_t
2022                 kc = max(Kc(b), 1e-3);
2023                 ti = normalize_search_ti(ctrl_key, plant_tipo, plant_id, theta0, Ti(b));
2024                 td = max(Td(b), 0);
2025                 [es_estable(b), mst(b)] = eval_mst_controller( ...
2026                     ctrl_key, kc, ti, td, P, P0, theta0, plant_tipo, plant_id, ...
2027                     Go, Gi, pade_order, N_der, Tf_min, use_exact_mst, w_mst);
2028                 kc_m(b)=kc; ti_m(b)=ti; td_m(b)=td;
2029             end
2030         end
2031     else
2032         [Kc, Ti] = ndgrid(kc_vals, ti_vals);
2033         n_t = numel(Kc);
2034 
2035         kc_m = zeros(n_t,1); ti_m = zeros(n_t,1); td_m = zeros(n_t,1);
2036         mst  = inf(n_t,1);
2037         es_estable = false(n_t,1);
2038 
2039         if n_t >= parfor_min_grid
2040             parfor b = 1:n_t
2041                 kc = max(Kc(b), 1e-3);
2042                 ti = normalize_search_ti(ctrl_key, plant_tipo, plant_id, theta0, Ti(b));
2043                 td = 0;
2044                 [es_estable(b), mst(b)] = eval_mst_controller( ...
2045                     ctrl_key, kc, ti, td, P, P0, theta0, plant_tipo, plant_id, ...
2046                     Go, Gi, pade_order, N_der, Tf_min, use_exact_mst, w_mst);
2047                 kc_m(b)=kc; ti_m(b)=ti; td_m(b)=td;
2048             end
2049         else
2050             for b = 1:n_t
2051                 kc = max(Kc(b), 1e-3);
2052                 ti = normalize_search_ti(ctrl_key, plant_tipo, plant_id, theta0, Ti(b));
2053                 td = 0;
2054                 [es_estable(b), mst(b)] = eval_mst_controller( ...
2055                     ctrl_key, kc, ti, td, P, P0, theta0, plant_tipo, plant_id, ...
2056                     Go, Gi, pade_order, N_der, Tf_min, use_exact_mst, w_mst);
2057                 kc_m(b)=kc; ti_m(b)=ti; td_m(b)=td;
2058             end
2059         end
2060     end
2061 
2062     scan.kc = kc_m;
2063     scan.ti = ti_m;
2064     scan.td = td_m;
2065     scan.mst = mst;
2066     scan.stable = es_estable;
2067     scan.n_eval = n_t;
2068     scan.n_stable = sum(es_estable & isfinite(mst));
2069 
2070     stable_mst = mst(es_estable & isfinite(mst));
2071     if isempty(stable_mst)
2072         scan.mst_stable_rng = [NaN NaN];
2073         scan.status = 'SCAN_NO_STABLE';
2074         status_msg('WARN', sprintf('Planta %s | Ctrl %s | GlobalScan no stable points |
     nEval=%d', ...
2075             plant_label, ctrl_key_to_user(ctrl_key), n_t));
2076     else
2077         scan.mst_stable_rng = [min(stable_mst) max(stable_mst)];
```

```
2078          status_msg('DONE', sprintf('Planta %s | Ctrl %s | GlobalScan done | nEval=%d |
      nStable=%d | MstStable=%s', ...
2079              plant_label, ctrl_key_to_user(ctrl_key), n_t, scan.n_stable,
      fmt_rng(scan.mst_stable_rng)));
2080      end
2081  end
2082
2083  function [bounds_list, status, regions] = select_bounds_from_scan(scan, ctrl_key, Mst_obj,
      h_target, margen_mult, ...
2084      kc_bounds_global, ti_bounds_global, td_bounds_global, ...
2085      region_gap_mult, max_regions_keep, region_min_pts)
2086
2087      stable_mask = scan.stable & isfinite(scan.mst);
2088      stable_kc = scan.kc(stable_mask);
2089      stable_ti = scan.ti(stable_mask);
2090      stable_td = scan.td(stable_mask);
2091      stable_mst = scan.mst(stable_mask);
2092
2093      bounds_list = {};
2094      regions = struct([]);
2095
2096      if isempty(stable_mst)
2097          status = 'SCAN_NO_STABLE';
2098          bounds_list = {default_bounds(ctrl_key, kc_bounds_global, ti_bounds_global,
      td_bounds_global)};
2099          return;
2100      end
2101
2102      mask_target = abs(stable_mst - Mst_obj) <= h_target;
2103
2104      if any(mask_target)
2105          status = 'SCAN_TARGET';
2106          regions = detect_scan_regions( ...
2107              stable_kc(mask_target), stable_ti(mask_target), stable_td(mask_target),
      stable_mst(mask_target), ...
2108              ctrl_key, scan.paso, region_gap_mult, region_min_pts);
2109      else
2110          err = abs(stable_mst - Mst_obj);
2111          min_err = min(err);
2112          mask_near = err <= (min_err + max(h_target, 0.5*scan.paso));
2113          status = 'TARGET_OUTSIDE';
2114          regions = detect_scan_regions( ...
2115              stable_kc(mask_near), stable_ti(mask_near), stable_td(mask_near),
      stable_mst(mask_near), ...
2116              ctrl_key, scan.paso, region_gap_mult, 1);
2117      end
2118
2119      if isempty(regions)
2120          [~, idx_best] = min(abs(stable_mst - Mst_obj));
2121          single.kc_rng  = [stable_kc(idx_best) stable_kc(idx_best)];
2122          single.ti_rng  = [stable_ti(idx_best) stable_ti(idx_best)];
2123          single.td_rng  = [stable_td(idx_best) stable_td(idx_best)];
2124          single.mst_rng = [stable_mst(idx_best) stable_mst(idx_best)];
2125          single.npts = 1;
2126          regions = single;
2127      end
2128
2129      scores = [[regions.npts].' -abs(mean_rng_array(regions,'mst_rng') - Mst_obj)];
2130      [~, ord] = sortrows(scores, [-1 -2]);
2131      regions = regions(ord);
2132
2133      nkeep = min(max_regions_keep, numel(regions));
2134      regions = regions(1:nkeep);
2135
2136      for ir = 1:numel(regions)
```

```
2137          margen = margen_mult * scan.paso;
2138          b = struct();
2139          b.kc = [max(regions(ir).kc_rng(1)-margen, kc_bounds_global(1)),
      min(regions(ir).kc_rng(2)+margen, kc_bounds_global(2))];
2140          b.ti = [max(regions(ir).ti_rng(1)-margen, ti_bounds_global(1)),
      min(regions(ir).ti_rng(2)+margen, ti_bounds_global(2))];
2141
2142          if strcmpi(ctrl_key,'PID')
2143              b.td = [max(regions(ir).td_rng(1)-margen, td_bounds_global(1)),
      min(regions(ir).td_rng(2)+margen, td_bounds_global(2))];
2144          else
2145              b.td = [0 0];
2146          end
2147
2148          bounds_list{end+1} = b; %#ok<AGROW>
2149      end
2150  end
2151
2152  function regions = detect_scan_regions(kc, ti, td, mst, ctrl_key, paso_scan, region_gap_mult,
      region_min_pts)
2153      regions = struct([]);
2154      if isempty(kc)
2155          return;
2156      end
2157
2158      n = numel(kc);
2159      used = false(n,1);
2160      gap = region_gap_mult * paso_scan;
2161
2162      comps = {};
2163      for i = 1:n
2164          if used(i), continue; end
2165
2166          queue = i;
2167          used(i) = true;
2168          comp = [];
2169
2170          while ~isempty(queue)
2171              j = queue(1);
2172              queue(1) = [];
2173              comp(end+1) = j; %#ok<AGROW>
2174
2175              if strcmpi(ctrl_key,'PID')
2176                  neigh = ~used & abs(kc-kc(j)) <= gap & abs(ti-ti(j)) <= gap & abs(td-td(j))
      <= gap;
2177              else
2178                  neigh = ~used & abs(kc-kc(j)) <= gap & abs(ti-ti(j)) <= gap;
2179              end
2180
2181              idxn = find(neigh);
2182              if ~isempty(idxn)
2183                  used(idxn) = true;
2184                  queue = [queue; idxn(:)]; %#ok<AGROW>
2185              end
2186          end
2187
2188          comps{end+1} = comp(:); %#ok<AGROW>
2189      end
2190
2191      kk = 0;
2192      for ic = 1:numel(comps)
2193          idx = comps{ic};
2194          if numel(idx) < region_min_pts
2195              continue;
2196          end
```

```
2197
2198          kk = kk + 1;
2199          regions(kk).npts    = numel(idx);
2200          regions(kk).kc_rng  = [min(kc(idx)) max(kc(idx))];
2201          regions(kk).ti_rng  = [min(ti(idx)) max(ti(idx))];
2202          regions(kk).td_rng  = [min(td(idx)) max(td(idx))];
2203          regions(kk).mst_rng = [min(mst(idx)) max(mst(idx))];
2204      end
2205
2206      if isempty(regions)
2207          kk = 0;
2208          for ic = 1:numel(comps)
2209              idx = comps{ic};
2210              kk = kk + 1;
2211              regions(kk).npts    = numel(idx);
2212              regions(kk).kc_rng  = [min(kc(idx)) max(kc(idx))];
2213              regions(kk).ti_rng  = [min(ti(idx)) max(ti(idx))];
2214              regions(kk).td_rng  = [min(td(idx)) max(td(idx))];
2215              regions(kk).mst_rng = [min(mst(idx)) max(mst(idx))];
2216          end
2217      end
2218  end
2219
2220  function m = mean_rng_array(regions, fieldname)
2221      m = zeros(numel(regions),1);
2222      for i = 1:numel(regions)
2223          r = regions(i).(fieldname);
2224          m(i) = mean(r);
2225      end
2226  end
2227
2228  function b = default_bounds(ctrl_key, kc_bounds_global, ti_bounds_global, td_bounds_global)
2229      b.kc = kc_bounds_global;
2230      b.ti = ti_bounds_global;
2231      if strcmpi(ctrl_key,'PID')
2232          b.td = td_bounds_global;
2233      else
2234          b.td = [0 0];
2235      end
2236  end
2237
2238  function print_global_scan_summary(scan)
2239      fprintf('  Scan global     : %-14s | Estables=%-6d | Rango Mst=%-18s\n', ...
2240          clip_str(scan.status,14), scan.n_stable, fmt_rng(scan.mst_stable_rng));
2241  end
2242
2243  function print_selected_regions(bounds_list, status, ctrl_key)
2244      fprintf('  Regiones        : %-14s | nReg=%d\n', clip_str(status,14), numel(bounds_list));
2245      for ir = 1:numel(bounds_list)
2246          b = bounds_list{ir};
2247          if strcmpi(ctrl_key,'PID')
2248              fprintf('    R%-2d -> k_c=%-18s | tau_i=%-18s | tau_d=%-18s\n', ir,
      fmt_rng(b.kc), fmt_rng(b.ti), fmt_rng(b.td));
2249          else
2250              fprintf('    R%-2d -> k_c=%-18s | tau_i=%-18s\n', ir, fmt_rng(b.kc),
      fmt_rng(b.ti));
2251          end
2252      end
2253  end
2254
2255  function [cand, iter_log_all, ref_status] = refinement_over_bounds_list( ...
2256      ctrl_key, P, P0, theta0, plant_tipo, plant_id, bounds_list, Mst_r, ...
2257      refinamientos, h_1_vals, edge_mult, h_relax_factor, ...
2258      kc_bounds_global, ti_bounds_global, td_bounds_global, ...
2259      Go, Gi, t, pade_order, N_der, Tf_min, plant_label, use_exact_mst, w_mst, ...
```

```
2260      save_eval_detail, max_empty_iters, iaedo0_base, iaedi0_base)
2261
2262      iter_log_all = {};
2263      raw_all = struct('kc',[],'ti',[],'td',[],'mst',[],'tol_mst',[],'ctrl_key',ctrl_key);
2264
2265      for ir = 1:numel(bounds_list)
2266          [cand_raw_r, iter_log_r, status_r] = refinement_search( ...
2267              ctrl_key, P, P0, theta0, plant_tipo, plant_id, bounds_list{ir}, Mst_r, ...
2268              refinamientos, h_1_vals, edge_mult, h_relax_factor, ...
2269              kc_bounds_global, ti_bounds_global, td_bounds_global, ...
2270              Go, Gi, pade_order, N_der, Tf_min, plant_label, use_exact_mst, w_mst, ...
2271              save_eval_detail, max_empty_iters, ir);
2272
2273          for k = 1:numel(iter_log_r)
2274              iter_log_all{end+1} = iter_log_r{k}; %#ok<AGROW>
2275          end
2276
2277          raw_all.kc  = [raw_all.kc; cand_raw_r.kc(:)];
2278          raw_all.ti  = [raw_all.ti; cand_raw_r.ti(:)];
2279          raw_all.td  = [raw_all.td; cand_raw_r.td(:)];
2280          raw_all.mst = [raw_all.mst; cand_raw_r.mst(:)];
2281          raw_all.tol_mst = [raw_all.tol_mst; cand_raw_r.tol_mst(:)];
2282
2283          status_msg('DONE', sprintf('Planta %s | Ctrl %s | Region %d/%d | status=%s |
      nCand=%d', ...
2284              plant_label, ctrl_key_to_user(ctrl_key), ir, numel(bounds_list), status_r,
      numel(cand_raw_r.kc)));
2285      end
2286
2287      if isempty(raw_all.kc)
2288          cand.kc = [];
2289          cand.ti = [];
2290          cand.td = [];
2291          cand.mst = [];
2292          cand.tol_mst = [];
2293          cand.iaedo = [];
2294          cand.iaedi = [];
2295          cand.ctrl_key = ctrl_key;
2296          ref_status = 'NO_RAW_CAND';
2297          return;
2298      end
2299
2300      A = [round(raw_all.kc,10), round(raw_all.ti,10), round(raw_all.td,10)];
2301      [A_unique, ia] = unique(A, 'rows', 'stable');
2302
2303      tol_unique = NaN(size(ia));
2304      mst_unique = NaN(size(ia));
2305      for iu = 1:size(A_unique,1)
2306          mask_u = ismember(A, A_unique(iu,:), 'rows');
2307          tol_unique(iu) = min(raw_all.tol_mst(mask_u));
2308          mst_unique(iu) = raw_all.mst(find(mask_u, 1, 'first'));
2309      end
2310
2311      cand_raw.kc  = raw_all.kc(ia);
2312      cand_raw.ti  = raw_all.ti(ia);
2313      cand_raw.td  = raw_all.td(ia);
2314      cand_raw.mst = mst_unique(:);
2315      cand_raw.tol_mst = tol_unique(:);
2316      cand_raw.ctrl_key = ctrl_key;
2317
2318      cand = evaluate_candidate_iae( ...
2319          ctrl_key, cand_raw, P, P0, theta0, plant_tipo, plant_id, ...
2320          Go, Gi, t, pade_order, N_der, Tf_min); % Calcula IAEdo e IAEdi para los candidatos
2321
2322      if isempty(cand.kc)
```

```
2323          ref_status = 'NO_IAE';
2324      else
2325          ref_status = 'OK';
2326      end
2327
2328      if ~isempty(iaedo0_base) && ~isempty(iaedi0_base) && ~isempty(cand.kc)
2329          % Lugar reservado si luego se desea enriquecer el log con J por iteracion.
2330      end
2331  end
2332
2333  function [cand_raw, iter_log, status] = refinement_search(ctrl_key, P, P0, theta0,
      plant_tipo, plant_id, ...
2334      bounds_init, Mst_r, refinamientos, h_1_vals, edge_mult, h_relax_factor, ...
2335      kc_bounds_global, ti_bounds_global, td_bounds_global, ...
2336      Go, Gi, pade_order, N_der, Tf_min, plant_label, use_exact_mst, w_mst, ...
2337      save_eval_detail, max_empty_iters, region_id)
2338
2339      global show_iter_status_global show_iter_logs_global
2340
2341      rango_kc = bounds_init.kc;
2342      rango_ti = bounds_init.ti;
2343      rango_td = bounds_init.td;
2344
2345      kc_cell  = cell(length(refinamientos),1);
2346      ti_cell  = cell(length(refinamientos),1);
2347      td_cell  = cell(length(refinamientos),1);
2348      mst_cell = cell(length(refinamientos),1);
2349
2350      iter_log = cell(length(refinamientos),1);
2351      empty_streak = 0;
2352      last_idx = length(refinamientos);
2353
2354      if show_iter_status_global && ~show_iter_logs_global
2355          print_iter_status_header(region_id, Mst_r);
2356      end
2357
2358      for idx = 1:length(refinamientos)
2359          paso = refinamientos(idx);
2360          h_1  = h_1_vals(idx);
2361
2362          mode_iter = ternary_txt(idx==1, 'REGION', 'LOCAL');
2363
2364          [kc_m, ti_m, td_m, Mst, es_estable, n_t] = build_grid_and_eval( ...
2365              ctrl_key, rango_kc, rango_ti, rango_td, paso, ...
2366              kc_bounds_global, ti_bounds_global, td_bounds_global, ...
2367              P, P0, theta0, plant_tipo, plant_id, Go, Gi, pade_order, N_der, Tf_min,
      use_exact_mst, w_mst); % Evalua toda la malla local
2368
2369          mask_nom = es_estable & isfinite(Mst) & (Mst >= (Mst_r-h_1)) & (Mst <= (Mst_r+h_1));
2370          if ~any(mask_nom)
2371              h_used = h_1 * h_relax_factor;
2372              mask = es_estable & isfinite(Mst) & (Mst >= (Mst_r-h_used)) & (Mst <=
      (Mst_r+h_used));
2373              used_relax = true;
2374          else
2375              h_used = h_1;
2376              mask = mask_nom;
2377              used_relax = false;
2378          end
2379
2380          kc_f  = kc_m(mask);
2381          ti_f  = ti_m(mask);
2382          td_f  = td_m(mask);
2383          Mst_f = Mst(mask);
2384
```

```
2385          kc_cell{idx}  = kc_f(:);
2386          ti_cell{idx}  = ti_f(:);
2387          td_cell{idx}  = td_f(:);
2388          mst_cell{idx} = Mst_f(:);
2389
2390          if isempty(kc_f)
2391              empty_streak = empty_streak + 1;
2392              if used_relax
2393                  status_iter = 'ZERO_RELAX';
2394              else
2395                  status_iter = 'ZERO_NOM';
2396              end
2397          else
2398              empty_streak = 0;
2399              if used_relax
2400                  status_iter = 'FOUND_RELAX';
2401              else
2402                  status_iter = 'FOUND_NOM';
2403              end
2404          end
2405
2406          if ~isempty(kc_f)
2407              edge_margin = edge_mult * paso;
2408
2409              new_low_kc  = min(kc_f) - paso;
2410              new_high_kc = max(kc_f) + paso;
2411              if max(kc_f) > (rango_kc(2)-edge_margin), new_high_kc = rango_kc(2); end
2412              if min(kc_f) < (rango_kc(1)+edge_margin), new_low_kc  = rango_kc(1); end
2413              rango_kc = [max(new_low_kc, kc_bounds_global(1)), min(new_high_kc,
      kc_bounds_global(2))];
2414
2415              new_low_ti  = min(ti_f) - paso;
2416              new_high_ti = max(ti_f) + paso;
2417              if max(ti_f) > (rango_ti(2)-edge_margin), new_high_ti = rango_ti(2); end
2418              if min(ti_f) < (rango_ti(1)+edge_margin), new_low_ti  = rango_ti(1); end
2419              rango_ti = [max(new_low_ti, ti_bounds_global(1)), min(new_high_ti,
      ti_bounds_global(2))];
2420
2421              if strcmpi(ctrl_key,'PID')
2422                  new_low_td  = min(td_f) - paso;
2423                  new_high_td = max(td_f) + paso;
2424                  if max(td_f) > (rango_td(2)-edge_margin), new_high_td = rango_td(2); end
2425                  if min(td_f) < (rango_td(1)+edge_margin), new_low_td  = rango_td(1); end
2426                  rango_td = [max(new_low_td, td_bounds_global(1)), min(new_high_td,
      td_bounds_global(2))];
2427              else
2428                  rango_td = [0 0];
2429              end
2430          end
2431
2432          obs = get_iter_obs(numel(kc_f), used_relax);
2433
2434          if save_eval_detail
2435              eval_detail = struct();
2436              eval_detail.kc          = kc_m(:);
2437              eval_detail.ti          = ti_m(:);
2438              eval_detail.td          = td_m(:);
2439              eval_detail.mst         = Mst(:);
2440              eval_detail.estable     = es_estable(:);
2441              eval_detail.in_band     = mask(:);
2442              eval_detail.in_band_nom = mask_nom(:);
2443          else
2444              eval_detail = [];
2445          end
2446
```

```
2447          iter_log{idx} = struct( ...
2448              'region_id', region_id, ...
2449              'idx', idx, ...
2450              'niter', length(refinamientos), ...
2451              'mode', mode_iter, ...
2452              'paso', paso, ...
2453              'h_nom', h_1, ...
2454              'h_used', h_used, ...
2455              'used_relax', used_relax, ...
2456              'status_iter', status_iter, ...
2457              'cand', numel(kc_f), ...
2458              'n_eval', n_t, ...
2459              'n_stable', sum(es_estable & isfinite(Mst)), ...
2460              'n_in_band', sum(mask), ...
2461              'kc_rng', [min_or_nan(kc_f) max_or_nan(kc_f)], ...
2462              'ti_rng', [min_or_nan(ti_f) max_or_nan(ti_f)], ...
2463              'td_rng', [min_or_nan(td_f) max_or_nan(td_f)], ...
2464              'mst_rng', [min_or_nan(Mst_f) max_or_nan(Mst_f)], ...
2465              'obs', obs, ...
2466              'eval_detail', eval_detail);
2467
2468          if show_iter_status_global && ~show_iter_logs_global
2469              print_iter_status_compact(iter_log{idx});
2470          end
2471
2472          if empty_streak >= max_empty_iters
2473              last_idx = idx;
2474              iter_log{idx}.obs = 'STOP_EMPTY';
2475              if show_iter_status_global && ~show_iter_logs_global
2476                  fprintf('    -> Stop temprano: %d iteraciones vacias consecutivas\n',
      empty_streak);
2477              else
2478                  status_msg('WARN', sprintf('Planta %s | Ctrl %s | Region %d | stop temprano
      por %d iter vacias', ...
2479                      plant_label, ctrl_key_to_user(ctrl_key), region_id, empty_streak),
      'summary');
2480              end
2481              break;
2482          end
2483      end
2484
2485      iter_log = iter_log(1:last_idx);
2486      kc_cell  = kc_cell(1:last_idx);
2487      ti_cell  = ti_cell(1:last_idx);
2488      td_cell  = td_cell(1:last_idx);
2489      mst_cell = mst_cell(1:last_idx);
2490
2491      last_valid_idx = find(~cellfun(@isempty, kc_cell), 1, 'last');
2492
2493      if isempty(last_valid_idx)
2494          cand_raw.kc = [];
2495          cand_raw.ti = [];
2496          cand_raw.td = [];
2497          cand_raw.mst = [];
2498          cand_raw.tol_mst = [];
2499          cand_raw.ctrl_key = ctrl_key;
2500          status = 'NO_CAND';
2501          return;
2502      end
2503
2504      sel_h_used = iter_log{last_valid_idx}.h_used;
2505      sel_kc  = kc_cell{last_valid_idx}(:);
2506      sel_ti  = ti_cell{last_valid_idx}(:);
2507      sel_td  = td_cell{last_valid_idx}(:);
2508      sel_mst = mst_cell{last_valid_idx}(:);
```

```
2509
2510      A = [round(sel_kc,10), round(sel_ti,10), round(sel_td,10)];
2511      [~, ia] = unique(A, 'rows', 'stable');
2512
2513      cand_raw.kc  = sel_kc(ia);
2514      cand_raw.ti  = sel_ti(ia);
2515      cand_raw.td  = sel_td(ia);
2516      cand_raw.mst = sel_mst(ia);
2517      cand_raw.tol_mst = sel_h_used * ones(numel(ia),1);
2518      cand_raw.ctrl_key = ctrl_key;
2519
2520      status = 'OK_RAW';
2521  end
2522
2523  function cand = evaluate_candidate_iae(ctrl_key, cand_raw, P, P0, theta0, plant_tipo,
      plant_id, ...
2524      Go, Gi, t, pade_order, N_der, Tf_min)
2525
2526      parfor_min_iae = 40;
2527
2528      if isempty(cand_raw.kc)
2529          cand.kc = [];
2530          cand.ti = [];
2531          cand.td = [];
2532          cand.mst = [];
2533          cand.tol_mst = [];
2534          cand.iaedo = [];
2535          cand.iaedi = [];
2536          cand.ctrl_key = ctrl_key;
2537          return;
2538      end
2539
2540      nC = numel(cand_raw.kc);
2541      iaedo = inf(nC,1);
2542      iaedi = inf(nC,1);
2543
2544      if nC >= parfor_min_iae
2545          parfor k = 1:nC
2546              [iaedo(k), iaedi(k)] = eval_iae_controller( ...
2547                  ctrl_key, cand_raw.kc(k), cand_raw.ti(k), cand_raw.td(k), ...
2548                  P, P0, theta0, plant_tipo, plant_id, Go, Gi, t, pade_order, N_der, Tf_min);
2549          end
2550      else
2551          for k = 1:nC
2552              [iaedo(k), iaedi(k)] = eval_iae_controller( ...
2553                  ctrl_key, cand_raw.kc(k), cand_raw.ti(k), cand_raw.td(k), ...
2554                  P, P0, theta0, plant_tipo, plant_id, Go, Gi, t, pade_order, N_der, Tf_min);
2555          end
2556      end
2557
2558      mask_iae = isfinite(iaedo) & isfinite(iaedi);
2559
2560      cand.kc       = cand_raw.kc(mask_iae);
2561      cand.ti       = cand_raw.ti(mask_iae);
2562      cand.td       = cand_raw.td(mask_iae);
2563      cand.mst      = cand_raw.mst(mask_iae);
2564      cand.tol_mst  = cand_raw.tol_mst(mask_iae);
2565      cand.iaedo    = iaedo(mask_iae);
2566      cand.iaedi    = iaedi(mask_iae);
2567      cand.ctrl_key = ctrl_key;
2568  end
2569
2570  function [kc_m, ti_m, td_m, Mst, es_estable, n_t] = build_grid_and_eval( ...
2571      ctrl_key, rango_kc, rango_ti, rango_td, paso, ...
2572      kc_bounds_global, ti_bounds_global, td_bounds_global, ...
```

```
2573      P, P0, theta0, plant_tipo, plant_id, Go, Gi, pade_order, N_der, Tf_min, use_exact_mst,
      w_mst)
2574
2575      parfor_min_grid = 200;
2576
2577      if strcmpi(ctrl_key,'PID')
2578          kc_vals = build_aligned_axis(rango_kc, kc_bounds_global, paso);
2579          ti_vals = build_aligned_axis(rango_ti, ti_bounds_global, paso);
2580          td_vals = build_aligned_axis(rango_td, td_bounds_global, paso);
2581
2582          if isempty(kc_vals) || isempty(ti_vals) || isempty(td_vals)
2583              kc_m=[]; ti_m=[]; td_m=[]; Mst=[]; es_estable=[]; n_t=0; return;
2584          end
2585
2586          [Kc, Ti, Td] = ndgrid(kc_vals, ti_vals, td_vals);
2587          n_t = numel(Kc);
2588
2589          kc_m = zeros(n_t,1);
2590          ti_m = zeros(n_t,1);
2591          td_m = zeros(n_t,1);
2592          Mst = inf(n_t,1);
2593          es_estable = false(n_t,1);
2594
2595          if n_t >= parfor_min_grid
2596              parfor b = 1:n_t
2597                  kc = max(Kc(b),1e-3);
2598                  ti = normalize_search_ti(ctrl_key, plant_tipo, plant_id, theta0, Ti(b));
2599                  td = max(Td(b),0);
2600                  [es_estable(b), Mst(b)] = eval_mst_controller( ...
2601                      ctrl_key, kc, ti, td, P, P0, theta0, plant_tipo, plant_id, ...
2602                      Go, Gi, pade_order, N_der, Tf_min, use_exact_mst, w_mst);
2603                  kc_m(b)=kc;
2604                  ti_m(b)=ti;
2605                  td_m(b)=td;
2606              end
2607          else
2608              for b = 1:n_t
2609                  kc = max(Kc(b),1e-3);
2610                  ti = normalize_search_ti(ctrl_key, plant_tipo, plant_id, theta0, Ti(b));
2611                  td = max(Td(b),0);
2612                  [es_estable(b), Mst(b)] = eval_mst_controller( ...
2613                      ctrl_key, kc, ti, td, P, P0, theta0, plant_tipo, plant_id, ...
2614                      Go, Gi, pade_order, N_der, Tf_min, use_exact_mst, w_mst);
2615                  kc_m(b)=kc;
2616                  ti_m(b)=ti;
2617                  td_m(b)=td;
2618              end
2619          end
2620      else
2621          kc_vals = build_aligned_axis(rango_kc, kc_bounds_global, paso);
2622          ti_vals = build_search_ti_axis(ctrl_key, plant_tipo, plant_id, theta0, ...
2623              rango_ti, ti_bounds_global, paso);
2624
2625          if isempty(kc_vals) || isempty(ti_vals)
2626              kc_m=[]; ti_m=[]; td_m=[]; Mst=[]; es_estable=[]; n_t=0; return;
2627          end
2628
2629          [Kc, Ti] = ndgrid(kc_vals, ti_vals);
2630          n_t = numel(Kc);
2631
2632          kc_m = zeros(n_t,1);
2633          ti_m = zeros(n_t,1);
2634          td_m = zeros(n_t,1);
2635          Mst = inf(n_t,1);
2636          es_estable = false(n_t,1);
```

```
2637
2638        if n_t >= parfor_min_grid
2639            parfor b = 1:n_t
2640                kc = max(Kc(b),1e-3);
2641                ti = normalize_search_ti(ctrl_key, plant_tipo, plant_id, theta0, Ti(b));
2642                td = 0;
2643                [es_estable(b), Mst(b)] = eval_mst_controller( ...
2644                    ctrl_key, kc, ti, td, P, P0, theta0, plant_tipo, plant_id, ...
2645                    Go, Gi, pade_order, N_der, Tf_min, use_exact_mst, w_mst);
2646                kc_m(b)=kc;
2647                ti_m(b)=ti;
2648                td_m(b)=td;
2649            end
2650        else
2651            for b = 1:n_t
2652                kc = max(Kc(b),1e-3);
2653                ti = normalize_search_ti(ctrl_key, plant_tipo, plant_id, theta0, Ti(b));
2654                td = 0;
2655                [es_estable(b), Mst(b)] = eval_mst_controller( ...
2656                    ctrl_key, kc, ti, td, P, P0, theta0, plant_tipo, plant_id, ...
2657                    Go, Gi, pade_order, N_der, Tf_min, use_exact_mst, w_mst);
2658                kc_m(b)=kc;
2659                ti_m(b)=ti;
2660                td_m(b)=td;
2661            end
2662        end
2663    end
2664end
2665
2666function [ok, mst_val] = eval_mst_controller(ctrl_key, kc, ti, td, P, P0, theta0, plant_tipo,
    plant_id, ...
2667    Go, Gi, pade_order, N_der, Tf_min, use_exact_mst, w_mst)
2668
2669    s = tf('s');
2670    ok = false;
2671    mst_val = inf;
2672
2673    try
2674        switch upper(ctrl_key)
2675            case 'PI'
2676                C = build_controller_tf(ctrl_key, kc, ti, td, plant_tipo, plant_id, theta0,
    N_der, Tf_min);
2677                Tlc = feedback(P*C, 1);
2678                ok = isstable(Tlc);
2679                if ~ok, return; end
2680
2681                if use_exact_mst
2682                    [~, ~, mst_val] = exact_mst_classic(P0, theta0, C, w_mst);
2683                else
2684                    L = P*C;
2685                    S = minreal(1/(1+L), 1e-6);
2686                    T = minreal(L/(1+L), 1e-6);
2687                    [Ms, ~] = getPeakGain(S);
2688                    [Mt, ~] = getPeakGain(T);
2689                    mst_val = max(Ms, Mt);
2690                end
2691
2692            case 'PID'
2693                C = build_controller_tf(ctrl_key, kc, ti, td, plant_tipo, plant_id, theta0,
    N_der, Tf_min);
2694                Tlc = feedback(P*C, 1);
2695                ok = isstable(Tlc);
2696                if ~ok, return; end
2697
2698                if use_exact_mst
```

```
                    [~, ~, mst_val] = exact_mst_classic(P0, theta0, C, w_mst);
                else
                    L = P*C;
                    S = minreal(1/(1+L), 1e-6);
                    T = minreal(L/(1+L), 1e-6);
                    [Ms, ~] = getPeakGain(S);
                    [Mt, ~] = getPeakGain(T);
                    mst_val = max(Ms, Mt);
                end

            case 'SP'
                C = build_controller_tf(ctrl_key, kc, ti, td, plant_tipo, plant_id, theta0, N_der, Tf_min);
                [~, Gry_sp, ~, ~, is_stable] = smith_tf(P0, theta0, theta0, C, Go, Gi, pade_order);
                ok = is_stable;
                if ~ok, return; end

                if use_exact_mst
                    [ok, ~, ~, mst_val] = exact_mst_sp(P0, theta0, theta0, C, w_mst);
                else
                    [Srob, Trob, ok] = sp_robustness_tf(P0, theta0, theta0, C, pade_order);
                    if ~ok, return; end
                    [Ms, ~] = getPeakGain(Srob);
                    [Mt, ~] = getPeakGain(Trob);
                    mst_val = max(Ms, Mt);
                end
                ok = ok && isstable(Gry_sp);

            case 'MSP'
                C = build_controller_tf(ctrl_key, kc, ti, td, plant_tipo, plant_id, theta0, N_der, Tf_min);
                [~, Hr_msp, ~, ~, is_stable, ~] = msp_tf(P0, theta0, theta0, C, plant_tipo, plant_id, Go, Gi, pade_order);
                ok = is_stable;
                if ~ok, return; end

                if use_exact_mst
                    [ok, ~, ~, mst_val] = exact_mst_msp(P0, theta0, theta0, C, plant_tipo, plant_id, w_mst);
                else
                    [Srob, Trob, ok] = msp_robustness_tf(P0, theta0, theta0, C, plant_tipo, plant_id, pade_order);
                    if ~ok, return; end
                    [Ms, ~] = getPeakGain(Srob);
                    [Mt, ~] = getPeakGain(Trob);
                    mst_val = max(Ms, Mt);
                end
                ok = ok && isstable(Hr_msp);

            otherwise
                ok = false;
                return;
        end

        ok = ok && isfinite(mst_val) && isscalar(mst_val);
    catch
        ok = false;
        mst_val = inf;
    end
end

function [Ms, Mt, Mst] = exact_mst_classic(P0, theta, C, w)
    jw = 1j*w(:).';
    P0jw = squeeze(freqresp(P0, w)); P0jw = P0jw(:).';
```

```
    Cjw  = squeeze(freqresp(C,  w)); Cjw  = Cjw(:).';
    e_th = exp(-jw*theta);

    L = Cjw .* P0jw .* e_th;
    S = 1 ./ (1 + L);
    T = L ./ (1 + L);

    Ms = max(abs(S));
    Mt = max(abs(T));
    Mst = max(Ms, Mt);
end

function [ok, Ms, Mt, Mst] = exact_mst_sp(P0, theta0, L_real, C, w)
    ok = false;
    Ms = inf; Mt = inf; Mst = inf;

    try
        jw = 1j*w(:).';
        P0jw = squeeze(freqresp(P0, w)); P0jw = P0jw(:).';
        Cjw  = squeeze(freqresp(C,  w)); Cjw  = Cjw(:).';
        e_th0 = exp(-jw*theta0);
        e_L   = exp(-jw*L_real);

        den_c  = 1 + Cjw .* P0jw .* (1 - e_th0);
        den_cl = 1 + Cjw .* P0jw .* (1 - e_th0 + e_L);

        S = den_c ./ den_cl;
        T = (Cjw .* P0jw .* e_L) ./ den_cl;

        Ms = max(abs(S));
        Mt = max(abs(T));
        Mst = max(Ms, Mt);
        ok = isfinite(Mst) && isscalar(Mst);
    catch
        ok = false;
    end
end

function [ok, Ms, Mt, Mst] = exact_mst_msp(P0, theta0, L_real, C, tipo, plant_id, w)
    ok = false;
    Ms = inf; Mt = inf; Mst = inf;

    try
        jw = 1j*w(:).';
        P0jw = squeeze(freqresp(P0, w)); P0jw = P0jw(:).';
        Cjw  = squeeze(freqresp(C,  w)); Cjw  = Cjw(:).';
        [Fp, ~, ~, ~] = build_Fp_msp(tipo, plant_id, theta0);
        Fpjw = squeeze(freqresp(Fp, w)); Fpjw = Fpjw(:).';

        e_th0 = exp(-jw*theta0);
        e_L   = exp(-jw*L_real);

        den_c  = 1 + Cjw .* P0jw .* (1 - Fpjw .* e_th0);
        den_cl = 1 + Cjw .* P0jw .* (1 + Fpjw .* (e_L - e_th0));

        S = den_c ./ den_cl;
        T = (Cjw .* P0jw .* Fpjw .* e_L) ./ den_cl;

        Ms = max(abs(S));
        Mt = max(abs(T));
        Mst = max(Ms, Mt);
        ok = isfinite(Mst) && isscalar(Mst);
    catch
        ok = false;
    end
```

```
2823 end
2824
2825 function [Srob, Trob, ok] = sp_robustness_tf(P0, theta0, L_real, C, pade_order)
2826     tol = 1e-6;
2827     ok = false;
2828     Srob = tf(1);
2829     Trob = tf(0);
2830
2831     try
2832         [numL, denL]   = pade(L_real, pade_order);
2833         [numTh, denTh] = pade(theta0, pade_order);
2834
2835         e_L   = tf(numL, denL);
2836         e_th0 = tf(numTh, denTh);
2837
2838         den_c  = minreal(1 + C*P0*(1 - e_th0), tol);
2839         den_cl = minreal(1 + C*P0*(1 - e_th0 + e_L), tol);
2840
2841         Srob = minreal(den_c / den_cl, tol);
2842         Trob = minreal((C*P0*e_L) / den_cl, tol);
2843
2844         ok = isstable(Trob);
2845     catch
2846         ok = false;
2847     end
2848 end
2849
2850 function [Srob, Trob, ok] = msp_robustness_tf(P0, theta0, L_real, C, tipo, plant_id,
    pade_order)
2851     tol = 1e-6;
2852     ok = false;
2853     Srob = tf(1);
2854     Trob = tf(0);
2855
2856     try
2857         [Fp, ~, ~, ~] = build_Fp_msp(tipo, plant_id, theta0);
2858         [numL, denL]   = pade(L_real, pade_order);
2859         [numTh, denTh] = pade(theta0, pade_order);
2860
2861         e_L   = tf(numL, denL);
2862         e_th0 = tf(numTh, denTh);
2863
2864         den_c  = minreal(1 + C*P0*(1 - Fp*e_th0), tol);
2865         den_cl = minreal(1 + C*P0*(1 + Fp*(e_L - e_th0)), tol);
2866
2867         Srob = minreal(den_c / den_cl, tol);
2868         Trob = minreal((C*P0*Fp*e_L) / den_cl, tol);
2869
2870         ok = isstable(Trob);
2871     catch
2872         ok = false;
2873     end
2874 end
2875
2876 function [iaedo, iaedi] = eval_iae_controller(ctrl_key, kc, ti, td, P, P0, theta0,
    plant_tipo, plant_id, Go, Gi, t, pade_order, N_der, Tf_min)
2877     s = tf('s');
2878     iaedo = inf;
2879     iaedi = inf;
2880
2881     try
2882         switch upper(ctrl_key)
2883             case 'PI'
2884                 C = build_controller_tf(ctrl_key, kc, ti, td, plant_tipo, plant_id, theta0,
    N_der, Tf_min);
```

```
2885                 Tlc = feedback(P*C, 1);
2886                 if ~isstable(Tlc), return; end
2887                 L = P*C;
2888                 S = 1/(1 + L);
2889                 G_y_do = Go * S;
2890                 G_y_di = P * Gi * S;
2891 
2892             case 'PID'
2893                 C = build_controller_tf(ctrl_key, kc, ti, td, plant_tipo, plant_id, theta0,
    N_der, Tf_min);
2894                 Tlc = feedback(P*C, 1);
2895                 if ~isstable(Tlc), return; end
2896                 L = P*C;
2897                 S = 1/(1 + L);
2898                 G_y_do = Go * S;
2899                 G_y_di = P * Gi * S;
2900 
2901             case 'SP'
2902                 C = build_controller_tf(ctrl_key, kc, ti, td, plant_tipo, plant_id, theta0,
    N_der, Tf_min);
2903                 [Gdo_sp, ~, Gdi_sp, ~, is_stable] = smith_tf(P0, theta0, theta0, C, Go, Gi,
    pade_order);
2904                 if ~is_stable, return; end
2905                 G_y_do = Gdo_sp;
2906                 G_y_di = Gdi_sp;
2907 
2908             case 'MSP'
2909                 C = build_controller_tf(ctrl_key, kc, ti, td, plant_tipo, plant_id, theta0,
    N_der, Tf_min);
2910                 [Gdo_msp, ~, Gdi_msp, ~, is_stable, ~] = msp_tf(P0, theta0, theta0, C,
    plant_tipo, plant_id, Go, Gi, pade_order);
2911                 if ~is_stable, return; end
2912                 G_y_do = Gdo_msp;
2913                 G_y_di = Gdi_msp;
2914 
2915             otherwise
2916                 return;
2917         end
2918 
2919         y_do = step(G_y_do, t);
2920         y_di = step(G_y_di, t);
2921 
2922         iaedo = trapz(t, abs(y_do));
2923         iaedi = trapz(t, abs(y_di));
2924     catch
2925         iaedo = inf;
2926         iaedi = inf;
2927     end
2928 end
2929 
2930 function [base_result, status] = build_base_result(ctrl_key, cand)
2931     base_result = struct();
2932 
2933     if isempty(cand.kc)
2934         status = 'NO_CAND';
2935         return;
2936     end
2937 
2938     mask_ok = isfinite(cand.iaedo) & isfinite(cand.iaedi);
2939     idx_ok = find(mask_ok);
2940 
2941     if isempty(idx_ok)
2942         status = 'NO_IAE';
2943         return;
2944     end
```

```
2945
2946      [iaedo0, rel_do0] = min(cand.iaedo(mask_ok));
2947      idx_do0 = idx_ok(rel_do0);
2948
2949      [iaedi0, rel_di0] = min(cand.iaedi(mask_ok));
2950      idx_di0 = idx_ok(rel_di0);
2951
2952      base_result.ctrl_key = ctrl_key;
2953
2954      base_result.do0.kc_do0 = cand.kc(idx_do0);
2955      base_result.do0.ti_do0 = cand.ti(idx_do0);
2956      base_result.do0.td_do0 = cand.td(idx_do0);
2957      base_result.do0.mst_do0 = cand.mst(idx_do0);
2958      base_result.do0.iaedo_do0 = iaedo0;
2959      base_result.do0.iaedi_at_do0 = cand.iaedi(idx_do0);
2960
2961      base_result.di0.kc_di0 = cand.kc(idx_di0);
2962      base_result.di0.ti_di0 = cand.ti(idx_di0);
2963      base_result.di0.td_di0 = cand.td(idx_di0);
2964      base_result.di0.mst_di0 = cand.mst(idx_di0);
2965      base_result.di0.iaedi_di0 = iaedi0;
2966      base_result.di0.iaedo_at_di0 = cand.iaedo(idx_di0);
2967
2968      status = 'OK';
2969  end
2970
2971  function base_result = build_neutral_base_result(ctrl_key)
2972      base_result = struct();
2973      base_result.ctrl_key = ctrl_key;
2974
2975      base_result.do0.kc_do0 = NaN;
2976      base_result.do0.ti_do0 = NaN;
2977      base_result.do0.td_do0 = NaN;
2978      base_result.do0.mst_do0 = NaN;
2979      base_result.do0.iaedo_do0 = 1;
2980      base_result.do0.iaedi_at_do0 = NaN;
2981
2982      base_result.di0.kc_di0 = NaN;
2983      base_result.di0.ti_di0 = NaN;
2984      base_result.di0.td_di0 = NaN;
2985      base_result.di0.mst_di0 = NaN;
2986      base_result.di0.iaedi_di0 = 1;
2987      base_result.di0.iaedo_at_di0 = NaN;
2988  end
2989
2990  function [opt_res, status] = build_opt_result_with_external_weights(ctrl_key, cand,
      iaedo0_base, iaedi0_base, Mst_obj, prev_opt)
2991      opt_res = struct();
2992
2993      if nargin < 6
2994          prev_opt = [];
2995      end
2996
2997      if isempty(cand.kc)
2998          status = 'NO_CAND';
2999          return;
3000      end
3001
3002      mask_ok = isfinite(cand.iaedo) & isfinite(cand.iaedi) & isfinite(cand.mst) &
      isfinite(cand.tol_mst);
3003      if ~any(mask_ok)
3004          status = 'NO_IAE';
3005          return;
3006      end
3007
```

```
    kc_ok    = cand.kc(mask_ok);
    ti_ok    = cand.ti(mask_ok);
    td_ok    = cand.td(mask_ok);
    mst_ok   = cand.mst(mask_ok);
    tol_ok   = cand.tol_mst(mask_ok);
    iaedo_ok = cand.iaedo(mask_ok);
    iaedi_ok = cand.iaedi(mask_ok);

    J_ok = 0.5 * ((iaedo_ok/max(iaedo0_base,1e-12)) + (iaedi_ok/max(iaedi0_base,1e-12)));
    Mst_err_ok = abs(mst_ok - Mst_obj);
    feasible = Mst_err_ok <= (tol_ok + 1e-12);

    if any(feasible)
        kc_s    = kc_ok(feasible);
        ti_s    = ti_ok(feasible);
        td_s    = td_ok(feasible);
        mst_s   = mst_ok(feasible);
        tol_s   = tol_ok(feasible);
        iaedo_s = iaedo_ok(feasible);
        iaedi_s = iaedi_ok(feasible);
        J_s     = J_ok(feasible);
        err_s   = Mst_err_ok(feasible);

        cont_s = continuity_cost(ctrl_key, kc_s, ti_s, td_s, prev_opt);
        score = J_s + 0.02*cont_s + 1e-6*err_s;
        [~, idx_opt] = min(score);

        opt_res.ctrl_key   = ctrl_key;
        opt_res.kc_opt     = kc_s(idx_opt);
        opt_res.ti_opt     = ti_s(idx_opt);
        opt_res.td_opt     = td_s(idx_opt);
        opt_res.mst_opt    = mst_s(idx_opt);
        opt_res.mst_err    = err_s(idx_opt);
        opt_res.mst_tol_used = tol_s(idx_opt);
        opt_res.iaedo_opt  = iaedo_s(idx_opt);
        opt_res.iaedi_opt  = iaedi_s(idx_opt);
        opt_res.J_opt      = J_s(idx_opt);

        status = 'OK';
    else
        min_err = min(Mst_err_ok);
        mask_closest = Mst_err_ok <= (min_err + 1e-9);

        kc_c    = kc_ok(mask_closest);
        ti_c    = ti_ok(mask_closest);
        td_c    = td_ok(mask_closest);
        mst_c   = mst_ok(mask_closest);
        tol_c   = tol_ok(mask_closest);
        iaedo_c = iaedo_ok(mask_closest);
        iaedi_c = iaedi_ok(mask_closest);
        J_c     = J_ok(mask_closest);
        err_c   = Mst_err_ok(mask_closest);

        cont_c = continuity_cost(ctrl_key, kc_c, ti_c, td_c, prev_opt);
        score = J_c + 0.02*cont_c + 1e-6*err_c;
        [~, idx_opt] = min(score);

        opt_res.ctrl_key   = ctrl_key;
        opt_res.kc_opt     = kc_c(idx_opt);
        opt_res.ti_opt     = ti_c(idx_opt);
        opt_res.td_opt     = td_c(idx_opt);
        opt_res.mst_opt    = mst_c(idx_opt);
        opt_res.mst_err    = err_c(idx_opt);
        opt_res.mst_tol_used = tol_c(idx_opt);
        opt_res.iaedo_opt  = iaedo_c(idx_opt);
```

```
3073          opt_res.iaedi_opt  = iaedi_c(idx_opt);
3074          opt_res.J_opt      = J_c(idx_opt);
3075
3076          status = 'FAR_MST';
3077      end
3078  end
3079
3080  function c = continuity_cost(ctrl_key, kc, ti, td, prev_opt)
3081      if isempty(prev_opt)
3082          c = zeros(size(kc));
3083          return;
3084      end
3085
3086      c = abs(kc - prev_opt.kc_opt) ./ max(abs(prev_opt.kc_opt), 1) + ...
3087          abs(ti - prev_opt.ti_opt) ./ max(abs(prev_opt.ti_opt), 1);
3088
3089      if strcmpi(ctrl_key,'PID')
3090          c = c + abs(td - prev_opt.td_opt) ./ max(abs(prev_opt.td_opt), 1);
3091      end
3092  end
3093
3094  function print_sweep_table(sweep)
3095      global thesis_mode_global
3096
3097      if isempty(thesis_mode_global) || ~thesis_mode_global
3098          fmt = '%-8s | %-10s | %-6s | %-6s | %-18s | %-8s | %-8s | %-8s | %-8s | %-8s | %-8s |
      %-10s | %-10s | %-10s | %-10s\n';
3099          fprintf(fmt, 'Mst_obj', 'status', 'nCand', 'mode', 'scan/ref', 'k_c_opt',
      'tau_i_opt', 'tau_d_opt', 'Mst_real', 'Mst_err', 'Mst_tol', 'IAEdo_opt', 'IAEdi_opt',
      'J_opt', 'DM_opt');
3100          fprintf('%s\n', repmat('-',1,190));
3101
3102          dm_vec = getfield_safe(sweep, 'DM_opt', NaN(size(sweep.Mst_obj)));
3103
3104          for i = 1:numel(sweep.Mst_obj)
3105              fprintf(fmt, ...
3106                  fmt_num(sweep.Mst_obj(i)), ...
3107                  clip_str(sweep.status{i},10), ...
3108                  num2str(sweep.nCand(i)), ...
3109                  clip_str(sweep.bounds_mode{i},6), ...
3110                  clip_str(sweep.scan_status{i},18), ...
3111                  fmt_num(sweep.Kc_opt(i)), ...
3112                  fmt_num(sweep.Ti_opt(i)), ...
3113                  fmt_num(sweep.Td_opt(i)), ...
3114                  fmt_num(sweep.Mst_real(i)), ...
3115                  fmt_num(sweep.Mst_err(i)), ...
3116                  fmt_num(sweep.Mst_tol_used(i)), ...
3117                  fmt_num(sweep.IAEdo_opt(i)), ...
3118                  fmt_num(sweep.IAEdi_opt(i)), ...
3119                  fmt_num(sweep.J_opt(i)), ...
3120                  fmt_num(dm_vec(i)));
3121          end
3122      else
3123          fmt = '  %-8s | %-10s | %-8s | %-8s | %-8s | %-8s | %-8s | %-10s | %-10s\n';
3124          fprintf(fmt, 'Mst_obj', 'status', 'k_c_opt', 'tau_i_opt', 'Mst_real', 'Mst_err',
      'Mst_tol', 'J_opt', 'DM_opt');
3125          fprintf('  %s\n', repmat('-',1,104));
3126
3127          dm_vec = getfield_safe(sweep, 'DM_opt', NaN(size(sweep.Mst_obj)));
3128
3129          for i = 1:numel(sweep.Mst_obj)
3130              fprintf(fmt, ...
3131                  fmt_num(sweep.Mst_obj(i)), ...
3132                  clip_str(sweep.status{i},10), ...
3133                  fmt_num(sweep.Kc_opt(i)), ...
```

```
                fmt_num(sweep.Ti_opt(i)), ...
                fmt_num(sweep.Mst_real(i)), ...
                fmt_num(sweep.Mst_err(i)), ...
                fmt_num(sweep.Mst_tol_used(i)), ...
                fmt_num(sweep.J_opt(i)), ...
                fmt_num(dm_vec(i)));
        end
    end
end

function print_iter_log(iter_log)
    for i = 1:numel(iter_log)
        print_iter_row(iter_log{i});
    end
end

function print_iter_status_header(region_id, Mst_obj)
    fprintf('  Refinamiento    : Region=%d | Mst_obj=%s\n', region_id, fmt_num(Mst_obj));
    fprintf('    %-8s | %-6s | %-11s | %-11s | %-11s | %-7s | %-7s\n', ...
        'Iter', 'Modo', 'Paso', 'h_nom', 'h_used', 'nEval', 'nCand');
    fprintf('    %s\n', repmat('-',1,76));
end

function print_iter_status_compact(row)
    fprintf('    %-8s | %-6s | %-11s | %-11s | %-11s | %-7d | %-7d | %s\n', ...
        sprintf('%d/%d', row.idx, row.niter), ...
        clip_str(row.mode,6), ...
        fmt_num(row.paso), ...
        fmt_num(row.h_nom), ...
        fmt_num(row.h_used), ...
        row.n_eval, ...
        row.cand, ...
        row.status_iter);
end

function print_iter_header()
    fmt = '%-4s | %-4s | %-6s | %-6s | %-6s | %-6s | %-11s | %-5s | %-7s | %-18s | %-18s | %-18s | %-18s | %-10s\n';
    fprintf(fmt, 'Iter', 'Reg', 'Mode', 'Paso', 'h_nom', 'h_used', 'Status', 'Cand', 'N_eval', 'k_c_rng', 'tau_i_rng', 'tau_d_rng', 'Mst_rng', 'Obs');
    fprintf('%s\n', repmat('-',1,170));
end

function print_iter_row(row)
    fmt = '%-4s | %-4s | %-6s | %-6s | %-6s | %-6s | %-11s | %-5d | %-7d | %-18s | %-18s | %-18s | %-18s | %-10s\n';
    fprintf(fmt, ...
        sprintf('%d/%d', row.idx, row.niter), ...
        num2str(row.region_id), ...
        clip_str(row.mode,6), ...
        fmt_num(row.paso), ...
        fmt_num(row.h_nom), ...
        fmt_num(row.h_used), ...
        clip_str(row.status_iter,11), ...
        row.cand, ...
        row.n_eval, ...
        fmt_rng(row.kc_rng), ...
        fmt_rng(row.ti_rng), ...
        fmt_rng(row.td_rng), ...
        fmt_rng(row.mst_rng), ...
        clip_str(row.obs,10));
end

function obs = get_iter_obs(ncand, used_relax)
    if ncand == 0
```

```
3196          if used_relax, obs = 'ZERO+REL'; else, obs = 'ZERO'; end
3197      elseif ncand == 1
3198          if used_relax, obs = '1+RELAX'; else, obs = '1_CAND'; end
3199      else
3200          if used_relax, obs = 'RELAX'; else, obs = 'REFINED'; end
3201      end
3202  end
3203
3204  function print_eval_mst_distribution_from_iter_log(iter_log, Mst_obj, only_stable,
      only_in_band, print_raw)
3205      if isempty(iter_log)
3206          fprintf('Sin iter_log.\n');
3207          return;
3208      end
3209
3210      max_raw_rows = 120;
3211
3212      for i = 1:numel(iter_log)
3213          row = iter_log{i};
3214
3215          if ~isfield(row,'eval_detail') || isempty(row.eval_detail)
3216              fprintf('Iter %d: sin detalle guardado.\n', row.idx);
3217              continue;
3218          end
3219
3220          ed = row.eval_detail;
3221          mask = true(size(ed.kc));
3222
3223          if only_stable
3224              mask = mask & ed.estable;
3225          end
3226          if only_in_band
3227              mask = mask & ed.in_band;
3228          end
3229
3230          kc  = ed.kc(mask);
3231          ti  = ed.ti(mask);
3232          td  = ed.td(mask);
3233          mst = ed.mst(mask);
3234          est = ed.estable(mask);
3235          inb = ed.in_band(mask);
3236
3237          fprintf('\n');
3238          fprintf('Iter %d/%d | Reg=%d | mode=%s | paso=%s | h_nom=%s | h_used=%s | status=%s |
      Mst_obj=%s | n_total=%d | n_stable=%d | n_in_band=%d\n', ...
3239              row.idx, row.niter, row.region_id, row.mode, fmt_num(row.paso),
      fmt_num(row.h_nom), fmt_num(row.h_used), ...
3240              row.status_iter, fmt_num(Mst_obj), numel(ed.kc), row.n_stable, row.n_in_band);
3241
3242          fprintf('Rangos filtrados: k_c=%s | tau_i=%s | tau_d=%s | Mst=%s\n', ...
3243              fmt_rng([min_or_nan(kc) max_or_nan(kc)]), ...
3244              fmt_rng([min_or_nan(ti) max_or_nan(ti)]), ...
3245              fmt_rng([min_or_nan(td) max_or_nan(td)]), ...
3246              fmt_rng([min_or_nan(mst(isfinite(mst))) max_or_nan(mst(isfinite(mst)))]));
3247
3248          mst_fin = mst(isfinite(mst));
3249          if isempty(mst_fin)
3250              fprintf('Percentiles Mst: p0=-- | p5=-- | p25=-- | p50=-- | p75=-- | p95=-- |
      p100=--\n');
3251              fprintf('Distribucion por bandas de Mst: sin valores finitos.\n');
3252              continue;
3253          end
3254
3255          q = prctile(mst_fin, [0 5 25 50 75 95 100]);
```

```
3256          fprintf('Percentiles Mst: p0=%s | p5=%s | p25=%s | p50=%s | p75=%s | p95=%s |
      p100=%s\n', ...
3257              fmt_num(q(1)), fmt_num(q(2)), fmt_num(q(3)), fmt_num(q(4)), fmt_num(q(5)),
      fmt_num(q(6)), fmt_num(q(7)));
3258
3259          edges = build_mst_edges(max(mst_fin));
3260          counts = histcounts(mst_fin, edges);
3261
3262          fprintf('Distribucion por bandas de Mst\n');
3263          fprintf('%-18s | %-8s\n', 'Banda', 'nPts');
3264          fprintf('%s\n', repmat('-',1,32));
3265          for k = 1:numel(counts)
3266              if isinf(edges(k+1))
3267                  fprintf('[%.1f, inf)        | %-8d\n', edges(k), counts(k));
3268              else
3269                  fprintf('[%.1f, %.1f)        | %-8d\n', edges(k), edges(k+1), counts(k));
3270              end
3271          end
3272
3273          if print_raw == 1 && numel(kc) <= max_raw_rows
3274              fprintf('\nTabla cruda filtrada\n');
3275              fmt = '%-10s | %-10s | %-10s | %-10s | %-8s | %-8s\n';
3276              fprintf(fmt, 'k_c', 'tau_i', 'tau_d', 'Mst', 'Stable', 'InBand');
3277              fprintf('%s\n', repmat('-',1,70));
3278              for j = 1:numel(kc)
3279                  fprintf(fmt, ...
3280                      fmt_num(kc(j)), fmt_num(ti(j)), fmt_num(td(j)), fmt_num(mst(j)), ...
3281                      ternary_txt(est(j),'YES','NO'), ternary_txt(inb(j),'YES','NO'));
3282              end
3283          elseif print_raw == 1
3284              fprintf('Tabla cruda omitida: n=%d > %d\n', numel(kc), max_raw_rows);
3285          end
3286      end
3287  end
3288
3289  function edges = build_mst_edges(mst_max)
3290      if isempty(mst_max) || ~isscalar(mst_max) || ~isfinite(mst_max)
3291          edges = [0 1];
3292          return;
3293      end
3294
3295      mst_max = max(mst_max, 1);
3296      hi = min(max(ceil(mst_max), 2), 200);
3297      edges = 0:1:hi;
3298
3299      if edges(end) < mst_max
3300          edges = [edges inf];
3301      end
3302
3303      if numel(edges) < 2
3304          edges = [0 1];
3305      end
3306  end
3307
3308  function styles = get_thesis_plot_styles()
3309      styles = struct();
3310      % Paleta de alto contraste para distinguir mejor cada controlador.
3311      styles.PI.color  = [0.00 0.00 0.00]; % negro
3312      styles.PID.color = [0.00 0.35 0.85]; % azul
3313      styles.SP.color  = [0.80 0.00 0.55]; % magenta
3314      styles.MSP.color = [0.00 0.60 0.20]; % verde
3315
3316      styles.PI.marker  = 'o';
3317      styles.PID.marker = 's';
3318      styles.SP.marker  = 'd';
```

```
3319      styles.MSP.marker = '^';
3320  end
3321
3322  function [y_min_plot, y_top_plot] = compute_clean_j_limits(y_all_valid, plot_j_use_cap,
      plot_j_hard_cap, plot_j_qref)
3323      y_all_valid = y_all_valid(isfinite(y_all_valid));
3324      if isempty(y_all_valid)
3325          y_min_plot = 0;
3326          y_top_plot = 1;
3327          return;
3328      end
3329
3330      y_ref = max(y_all_valid);
3331      if plot_j_use_cap
3332          y_q = prctile(y_all_valid, min(max(plot_j_qref, 0), 100));
3333          y_ref = min([y_ref, plot_j_hard_cap, max(y_q * 1.20, y_q + 0.1)]);
3334      end
3335
3336      y_keep = y_all_valid(y_all_valid <= y_ref);
3337      if isempty(y_keep)
3338          y_keep = min(max(y_all_valid), plot_j_hard_cap);
3339      else
3340          y_keep = max(y_keep);
3341      end
3342
3343      y_min_plot = max(0, min(y_all_valid) * 0.95);
3344      y_top_plot = max(y_keep * 1.05, y_min_plot + 1e-3);
3345  end
3346
3347  function hmain = plot_sweep_curve_clean(ax, sweep, style, label_txt, y_cap, line_w,
      marker_sz, hide_above, convention_mode)
3348      if nargin < 9 || isempty(convention_mode)
3349          convention_mode = 'default';
3350      end
3351      hmain = gobjects(0);
3352      x = sweep.Mst_obj(:);
3353      y = get_sweep_plot_field(sweep, 'J_opt');
3354      mask_status = get_sweep_plot_mask(sweep, numel(x));
3355
3356      y_plot = y;
3357      hide_mask = isfinite(y_plot) & (y_plot > hide_above);
3358      y_plot(hide_mask) = NaN;
3359      clip_mask = isfinite(y) & (y > y_cap);
3360      clip_mask = clip_mask & ~hide_mask;
3361      y_plot(clip_mask) = y_cap;
3362
3363      mask_plot = mask_status & isfinite(x) & isfinite(y_plot);
3364      if any(mask_plot)
3365          hmain = plot_editorial_series(ax, x(mask_plot), y_plot(mask_plot), style, label_txt,
      line_w, marker_sz, convention_mode);
3366      end
3367
3368      clip_mask = clip_mask & mask_status;
3369      if any(clip_mask)
3370          plot(ax, x(clip_mask), y_cap * ones(sum(clip_mask),1), '^', ...
3371              'Color', style.color, ...
3372              'MarkerFaceColor', style.color, ...
3373              'MarkerSize', max(5, marker_sz - 1), ...
3374              'HandleVisibility', 'off');
3375      end
3376  end
3377
3378  function add_j_zoom_inset(ax_main, plant, mst_grid, runPI, runPID, runSP, runMSP, styles, ...
3379      line_w, marker_sz, hide_above, font_name_axes, font_size_axes)
3380
```

```
3381      x_zoom_min = mst_grid(2);
3382      x_zoom_max = mst_grid(end);
3383      if x_zoom_max <= x_zoom_min
3384          return;
3385      end
3386
3387      y_zoom = [];
3388      if runPI && isfield(plant, 'PI_sweep')
3389          y_zoom = [y_zoom; collect_j_subset_for_zoom(plant.PI_sweep, x_zoom_min, x_zoom_max,
      hide_above)]; %#ok<AGROW>
3390      end
3391      if runPID && isfield(plant, 'PID_sweep')
3392          y_zoom = [y_zoom; collect_j_subset_for_zoom(plant.PID_sweep, x_zoom_min, x_zoom_max,
      hide_above)]; %#ok<AGROW>
3393      end
3394      if runSP && isfield(plant, 'SP_sweep')
3395          y_zoom = [y_zoom; collect_j_subset_for_zoom(plant.SP_sweep, x_zoom_min, x_zoom_max,
      hide_above)]; %#ok<AGROW>
3396      end
3397      if runMSP && isfield(plant, 'MSP_sweep')
3398          y_zoom = [y_zoom; collect_j_subset_for_zoom(plant.MSP_sweep, x_zoom_min, x_zoom_max,
      hide_above)]; %#ok<AGROW>
3399      end
3400
3401      y_zoom = y_zoom(isfinite(y_zoom));
3402      if isempty(y_zoom)
3403          return;
3404      end
3405
3406      [y_zoom_min, y_zoom_top] = compute_clean_j_limits(y_zoom, false, max(y_zoom), 95);
3407
3408      ax_inset = axes('Parent', ancestor(ax_main, 'figure'), ...
3409          'Position', [0.18 0.58 0.28 0.24]);
3410      hold(ax_inset, 'on');
3411      box(ax_inset, 'on');
3412      format_metric_axis(ax_inset, font_name_axes, max(7.5, font_size_axes - 1.2), 0.10, 0.05);
3413
3414      if runPI && isfield(plant, 'PI_sweep')
3415          plot_sweep_curve_subset(ax_inset, plant.PI_sweep, styles.PI, y_zoom_top, line_w -
      0.15, marker_sz - 1, hide_above, x_zoom_min, x_zoom_max);
3416      end
3417      if runPID && isfield(plant, 'PID_sweep')
3418          plot_sweep_curve_subset(ax_inset, plant.PID_sweep, styles.PID, y_zoom_top, line_w -
      0.15, marker_sz - 1, hide_above, x_zoom_min, x_zoom_max);
3419      end
3420      if runSP && isfield(plant, 'SP_sweep')
3421          plot_sweep_curve_subset(ax_inset, plant.SP_sweep, styles.SP, y_zoom_top, line_w -
      0.15, marker_sz - 1, hide_above, x_zoom_min, x_zoom_max);
3422      end
3423      if runMSP && isfield(plant, 'MSP_sweep')
3424          plot_sweep_curve_subset(ax_inset, plant.MSP_sweep, styles.MSP, y_zoom_top, line_w -
      0.15, marker_sz - 1, hide_above, x_zoom_min, x_zoom_max);
3425      end
3426
3427      xlim(ax_inset, [x_zoom_min x_zoom_max]);
3428      ylim(ax_inset, [y_zoom_min y_zoom_top * 1.05]);
3429      title(ax_inset, 'Zoom', 'FontName', font_name_axes, 'FontSize', max(7.5, font_size_axes -
      1.0), 'FontWeight', 'normal');
3430  end
3431
3432  function y = collect_j_subset_for_zoom(sweep, x_min, x_max, hide_above)
3433      y = [];
3434      if isempty(sweep) || ~isstruct(sweep) || ~isfield(sweep, 'Mst_obj') || ~isfield(sweep,
      'J_opt')
3435          return;
```

```
    end
    x = sweep.Mst_obj(:);
    yv = get_sweep_plot_field(sweep, 'J_opt');
    mask_status = get_sweep_plot_mask(sweep, numel(x));
    mask = mask_status & isfinite(x) & isfinite(yv) & (x >= x_min) & (x <= x_max) & (yv <= hide_above);
    y = yv(mask);
end

function plot_sweep_curve_subset(ax, sweep, style, y_cap, line_w, marker_sz, hide_above, x_min, x_max)
    if isempty(sweep) || ~isstruct(sweep) || ~isfield(sweep, 'Mst_obj') || ~isfield(sweep, 'J_opt')
        return;
    end

    x = sweep.Mst_obj(:);
    y = get_sweep_plot_field(sweep, 'J_opt');
    mask_status = get_sweep_plot_mask(sweep, numel(x));
    mask_range = mask_status & isfinite(x) & (x >= x_min) & (x <= x_max);
    x = x(mask_range);
    y = y(mask_range);
    if isempty(x)
        return;
    end

    y_plot = y;
    hide_mask = isfinite(y_plot) & (y_plot > hide_above);
    y_plot(hide_mask) = NaN;
    clip_mask = isfinite(y) & (y > y_cap) & ~hide_mask;
    y_plot(clip_mask) = y_cap;

    mask_plot = isfinite(x) & isfinite(y_plot);
    if any(mask_plot)
        plot_editorial_series(ax, x(mask_plot), y_plot(mask_plot), style, '', line_w, max(3.5, marker_sz), 'default');
    end
end

function h = plot_sweep_metric(ax, sweep, yfield, style, label_txt, line_w, marker_sz, convention_mode)
    if nargin < 8 || isempty(convention_mode)
        convention_mode = 'default';
    end
    if isempty(sweep) || ~isstruct(sweep) || ~isfield(sweep, 'Mst_obj') || ~isfield(sweep, yfield)
        h = gobjects(0);
        return;
    end

    x = sweep.Mst_obj(:);
    y = get_sweep_plot_field(sweep, yfield);
    mask_status = get_sweep_plot_mask(sweep, numel(x));
    mask = mask_status & isfinite(x) & isfinite(y);
    if strcmp(get(ax, 'YScale'), 'log')
        mask = mask & (y > 0);
    end
    if ~any(mask)
        h = gobjects(0);
        return;
    end

    h = plot_editorial_series(ax, x(mask), y(mask), style, label_txt, line_w, marker_sz, convention_mode);
end
```

```
3494
3495  function mask = get_sweep_plot_mask(sweep, n_expected)
3496      mask = true(n_expected, 1);
3497      if isempty(sweep) || ~isstruct(sweep)
3498          return;
3499      end
3500
3501      status = get_sweep_plot_status(sweep);
3502      if isempty(status)
3503          return;
3504      end
3505
3506      if numel(status) ~= n_expected
3507          return;
3508      end
3509
3510      % En figuras solo se muestran puntos efectivamente encontrados y aceptados.
3511      mask = status == "OK";
3512  end
3513
3514  function vals = get_sweep_plotted_values(sweep, fieldname)
3515      vals = [];
3516      if isempty(sweep) || ~isstruct(sweep) || ~isfield(sweep, fieldname)
3517          return;
3518      end
3519
3520      raw = get_sweep_plot_field(sweep, fieldname);
3521      mask = get_sweep_plot_mask(sweep, numel(raw));
3522      vals = raw(mask);
3523  end
3524
3525  function vals = get_sweep_plot_field(sweep, fieldname)
3526      vals = [];
3527      if isempty(sweep) || ~isstruct(sweep)
3528          return;
3529      end
3530
3531      if isfield(sweep, fieldname)
3532          vals = sweep.(fieldname)(:);
3533          if ~isempty(vals)
3534              return;
3535          end
3536      end
3537
3538      if isfield(sweep, 'local_selection') && isstruct(sweep.local_selection) ...
3539              && isfield(sweep.local_selection, fieldname)
3540          vals = sweep.local_selection.(fieldname)(:);
3541          return;
3542      end
3543  end
3544
3545  function status = get_sweep_plot_status(sweep)
3546      status = string.empty(0,1);
3547      if isempty(sweep) || ~isstruct(sweep)
3548          return;
3549      end
3550
3551      if isfield(sweep, 'status')
3552          status = string(sweep.status(:));
3553          if ~isempty(status)
3554              return;
3555          end
3556      end
3557
3558      if isfield(sweep, 'local_selection') && isstruct(sweep.local_selection) ...
```

```
3559             && isfield(sweep.local_selection, 'status')
3560         status = string(sweep.local_selection.status(:));
3561         return;
3562     end
3563 end
3564
3565 function create_metric_panel_figure( ...
3566     Plantas, idx_run, mst_grid, runPI, runPID, runSP, runMSP, styles, ...
3567     yfield, fig_name, super_title, y_label_txt, file_base, use_log_y, ...
3568     fig_w_cm, fig_h_cm, font_name_axes, font_name_title, ...
3569     font_size_axes, font_size_title, font_size_legend, line_w, marker_sz, ...
3570     grid_alpha_major, grid_alpha_minor, legend_location, legend_num_cols, ...
3571     export_pub_figs, pub_fig_folder, fig_res_dpi, convention_mode)
3572
3573     if nargin < 31 || isempty(convention_mode)
3574         convention_mode = 'default';
3575     end
3576
3577     fig = figure( ...
3578         'Name', fig_name, ...
3579         'NumberTitle', 'off', ...
3580         'Color', 'w', ...
3581         'Units', 'centimeters', ...
3582         'Position', [2 2 max(15, fig_w_cm * 1.55) max(10, fig_h_cm * 1.45)]);
3583
3584     nPlants = numel(idx_run);
3585     nCols = min(2, max(1, nPlants));
3586     nRows = ceil(nPlants / nCols);
3587     tl = tiledlayout(fig, nRows, nCols, 'TileSpacing', 'compact', 'Padding', 'loose');
3588     title(tl, super_title, 'FontName', font_name_title, 'FontSize', font_size_title + 0.3,
    'FontWeight', 'bold');
3589
3590     for kp = 1:nPlants
3591         ip = idx_run(kp);
3592         ax = nexttile(tl, kp);
3593         hold(ax, 'on');
3594         format_metric_axis(ax, font_name_axes, font_size_axes, grid_alpha_major,
    grid_alpha_minor);
3595         if use_log_y
3596             ax.YScale = 'log';
3597         end
3598
3599         htmp = gobjects(0);
3600         if runPI && isfield(Plantas(ip),'PI_sweep')
3601             h = plot_sweep_metric(ax, Plantas(ip).PI_sweep, yfield, styles.PI, 'PI', line_w,
    marker_sz, convention_mode);
3602             if ~isempty(h) && all(isgraphics(h)), htmp(end+1) = h; end %#ok<AGROW>
3603         end
3604         if runPID && isfield(Plantas(ip),'PID_sweep')
3605             h = plot_sweep_metric(ax, Plantas(ip).PID_sweep, yfield, styles.PID, 'PID',
    line_w, marker_sz, convention_mode);
3606             if ~isempty(h) && all(isgraphics(h)), htmp(end+1) = h; end %#ok<AGROW>
3607         end
3608         if runSP && isfield(Plantas(ip),'SP_sweep')
3609             h = plot_sweep_metric(ax, Plantas(ip).SP_sweep, yfield, styles.SP, 'SP', line_w,
    marker_sz, convention_mode);
3610             if ~isempty(h) && all(isgraphics(h)), htmp(end+1) = h; end %#ok<AGROW>
3611         end
3612         if runMSP && isfield(Plantas(ip),'MSP_sweep')
3613             h = plot_sweep_metric(ax, Plantas(ip).MSP_sweep, yfield, styles.MSP, 'MSP',
    line_w, marker_sz, convention_mode);
3614             if ~isempty(h) && all(isgraphics(h)), htmp(end+1) = h; end %#ok<AGROW>
3615         end
3616
3617         xlim(ax, [min(mst_grid) max(mst_grid)]);
```

```
3618          xlabel(ax, 'Robustez, Mst', 'FontName', font_name_axes, 'FontSize', font_size_axes +
      0.3, 'Interpreter', 'none');
3619          ylabel(ax, y_label_txt, 'FontName', font_name_axes, 'FontSize', font_size_axes + 0.3,
      'Interpreter', 'tex');
3620          title(ax, plant_title_with_model(Plantas(ip)), 'FontName', font_name_title,
      'FontSize', font_size_title - 0.2, 'FontWeight', 'normal', 'Interpreter', 'tex');
3621          if ~isempty(htmp)
3622              legend_labels = get(htmp, {'DisplayName'});
3623              if ischar(legend_labels), legend_labels = {legend_labels}; end
3624              apply_publication_legend(legend(ax, htmp, legend_labels, 'Location',
      legend_location, 'Orientation', 'vertical'), font_name_axes, font_size_legend, 1);
3625          else
3626              text(ax, 0.5, 0.5, sprintf('Sin datos validos para %s', yfield), ...
3627                  'Units', 'normalized', 'HorizontalAlignment', 'center', 'VerticalAlignment',
      'middle', ...
3628                  'FontName', font_name_axes, 'FontSize', font_size_axes, 'Color', [0.35 0.35
      0.35]);
3629          end
3630      end
3631
3632      if export_pub_figs
3633          export_base = file_base_with_plant_suffix(file_base, Plantas, idx_run);
3634          safe_export_dual(fig, fig, fullfile(pub_fig_folder, [export_base '.png']),
      fullfile(pub_fig_folder, [export_base '.pdf']), fig_res_dpi);
3635      end
3636  end
3637
3638  function create_params_figures( ...
3639      Plantas, idx_run, mst_grid, runPI, runPID, runSP, runMSP, styles, ...
3640      fig_w_cm, fig_h_cm, font_name_axes, font_name_title, ...
3641      font_size_axes, font_size_title, font_size_legend, line_w, marker_sz, ...
3642      grid_alpha_major, grid_alpha_minor, legend_location, ...
3643      export_pub_figs, pub_fig_folder, fig_res_dpi, convention_mode)
3644
3645      if nargin < 24 || isempty(convention_mode)
3646          convention_mode = 'default';
3647      end
3648
3649      for ip = idx_run
3650          figParams = figure( ...
3651              'Name', sprintf('Parametros vs Mst - %s', Plantas(ip).id), ...
3652              'NumberTitle', 'off', ...
3653              'Color', 'w', ...
3654              'Units', 'centimeters', ...
3655              'Position', [2 2 fig_w_cm fig_h_cm * 1.35]);
3656
3657          tlP = tiledlayout(figParams, 3, 1, 'TileSpacing', 'compact', 'Padding', 'compact');
3658          title(tlP, sprintf('%s | Parametros optimos vs robustez',
      plant_title_with_model(Plantas(ip))), ...
3659              'FontName', font_name_title, 'FontSize', font_size_title - 0.1, ...
3660              'FontWeight', 'normal', 'Interpreter', 'tex');
3661
3662          axPk = nexttile(tlP, 1); hold(axPk, 'on');
3663          axPt = nexttile(tlP, 2); hold(axPt, 'on');
3664          axPd = nexttile(tlP, 3); hold(axPd, 'on');
3665
3666          format_metric_axis(axPk, font_name_axes, font_size_axes, grid_alpha_major,
      grid_alpha_minor);
3667          format_metric_axis(axPt, font_name_axes, font_size_axes, grid_alpha_major,
      grid_alpha_minor);
3668          format_metric_axis(axPd, font_name_axes, font_size_axes, grid_alpha_major,
      grid_alpha_minor);
3669
3670          if runPI && isfield(Plantas(ip),'PI_sweep')
```

```
3671            plot_sweep_metric(axPk, Plantas(ip).PI_sweep, 'Kc_opt', styles.PI, 'PI', line_w,
    marker_sz, convention_mode);
3672            plot_sweep_metric(axPt, Plantas(ip).PI_sweep, 'Ti_opt', styles.PI, 'PI', line_w,
    marker_sz, convention_mode);
3673        end
3674        if runPID && isfield(Plantas(ip),'PID_sweep')
3675            plot_sweep_metric(axPk, Plantas(ip).PID_sweep, 'Kc_opt', styles.PID, 'PID',
    line_w, marker_sz, convention_mode);
3676            plot_sweep_metric(axPt, Plantas(ip).PID_sweep, 'Ti_opt', styles.PID, 'PID',
    line_w, marker_sz, convention_mode);
3677            plot_sweep_metric(axPd, Plantas(ip).PID_sweep, 'Td_opt', styles.PID, 'PID',
    line_w, marker_sz, convention_mode);
3678        end
3679        if runSP && isfield(Plantas(ip),'SP_sweep')
3680            plot_sweep_metric(axPk, Plantas(ip).SP_sweep, 'Kc_opt', styles.SP, 'SP', line_w,
    marker_sz, convention_mode);
3681            plot_sweep_metric(axPt, Plantas(ip).SP_sweep, 'Ti_opt', styles.SP, 'SP', line_w,
    marker_sz, convention_mode);
3682        end
3683        if runMSP && isfield(Plantas(ip),'MSP_sweep')
3684            plot_sweep_metric(axPk, Plantas(ip).MSP_sweep, 'Kc_opt', styles.MSP, 'MSP',
    line_w, marker_sz, convention_mode);
3685            plot_sweep_metric(axPt, Plantas(ip).MSP_sweep, 'Ti_opt', styles.MSP, 'MSP',
    line_w, marker_sz, convention_mode);
3686        end
3687
3688        ylabel(axPk, 'k_c', 'Interpreter', 'tex');
3689        ylabel(axPt, '\tau_i', 'Interpreter', 'tex');
3690        ylabel(axPd, '\tau_d', 'Interpreter', 'tex');
3691        xlabel(axPd, 'Mst', 'Interpreter', 'none');
3692        title(axPk, 'k_c optimo', 'Interpreter', 'tex');
3693        title(axPt, '\tau_i optimo', 'Interpreter', 'tex');
3694        title(axPd, '\tau_d optimo', 'Interpreter', 'tex');
3695        xlim(axPk, [min(mst_grid) max(mst_grid)]);
3696        xlim(axPt, [min(mst_grid) max(mst_grid)]);
3697        xlim(axPd, [min(mst_grid) max(mst_grid)]);
3698        apply_publication_legend(legend(axPk, 'Location', 'eastoutside', 'Orientation',
    'vertical'), font_name_axes, font_size_legend, 1);
3699
3700        if export_pub_figs
3701            drawnow;
3702            export_target_params = tlP;
3703            if ~isgraphics(export_target_params)
3704                export_target_params = figParams;
3705            end
3706            safe_export_dual(figParams, export_target_params, ...
3707                fullfile(pub_fig_folder, sprintf('Params_vs_Mst_%s.png', Plantas(ip).id)), ...
3708                fullfile(pub_fig_folder, sprintf('Params_vs_Mst_%s.pdf', Plantas(ip).id)),
    fig_res_dpi);
3709        end
3710    end
3711 end
3712
3713 function create_metric_compare_panel_figure( ...
3714     Plantas, idx_run, mst_grid, runPI, runPID, runSP, runMSP, styles, ...
3715     yfield_exact, yfield_paper, fig_name, super_title, y_label_txt, file_base, use_log_y, ...
3716     fig_w_cm, fig_h_cm, font_name_axes, font_name_title, ...
3717     font_size_axes, font_size_title, font_size_legend, line_w, marker_sz, ...
3718     grid_alpha_major, grid_alpha_minor, legend_location, legend_num_cols, ...
3719     export_pub_figs, pub_fig_folder, fig_res_dpi)
3720
3721     fig = figure( ...
3722         'Name', fig_name, ...
3723         'NumberTitle', 'off', ...
3724         'Color', 'w', ...
```

```
3725          'Units', 'centimeters', ...
3726          'Position', [2 2 max(15, fig_w_cm * 1.55) max(10, fig_h_cm * 1.45)]);
3727
3728      nPlants = numel(idx_run);
3729      nCols = min(2, max(1, nPlants));
3730      nRows = ceil(nPlants / nCols);
3731      tl = tiledlayout(fig, nRows, nCols, 'TileSpacing', 'compact', 'Padding', 'loose');
3732      title(tl, super_title, 'FontName', font_name_title, 'FontSize', font_size_title + 0.3,
      'FontWeight', 'bold');
3733
3734      for kp = 1:nPlants
3735          ip = idx_run(kp);
3736          ax = nexttile(tl, kp);
3737          hold(ax, 'on');
3738          format_metric_axis(ax, font_name_axes, font_size_axes, grid_alpha_major,
      grid_alpha_minor);
3739          if use_log_y
3740              ax.YScale = 'log';
3741          end
3742
3743          htmp = gobjects(0);
3744          htmp = add_compare_series(ax, Plantas(ip), 'PI_sweep', runPI, styles.PI, 'PI', ...
3745              yfield_exact, yfield_paper, line_w, marker_sz, htmp);
3746          htmp = add_compare_series(ax, Plantas(ip), 'PID_sweep', runPID, styles.PID, 'PID', ...
3747              yfield_exact, yfield_paper, line_w, marker_sz, htmp);
3748          htmp = add_compare_series(ax, Plantas(ip), 'SP_sweep', runSP, styles.SP, 'SP', ...
3749              yfield_exact, yfield_paper, line_w, marker_sz, htmp);
3750          htmp = add_compare_series(ax, Plantas(ip), 'MSP_sweep', runMSP, styles.MSP, 'MSP', ...
3751              yfield_exact, yfield_paper, line_w, marker_sz, htmp);
3752
3753          xlim(ax, [min(mst_grid) max(mst_grid)]);
3754          xlabel(ax, 'Robustez, Mst', 'FontName', font_name_axes, 'FontSize', font_size_axes +
      0.3, 'Interpreter', 'none');
3755          ylabel(ax, y_label_txt, 'FontName', font_name_axes, 'FontSize', font_size_axes + 0.3,
      'Interpreter', 'tex');
3756          title(ax, plant_title_with_model(Plantas(ip)), 'FontName', font_name_title,
      'FontSize', font_size_title - 0.2, 'FontWeight', 'normal', 'Interpreter', 'tex');
3757          if ~isempty(htmp)
3758              legend_labels = get(htmp, {'DisplayName'});
3759              if ischar(legend_labels), legend_labels = {legend_labels}; end
3760              apply_publication_legend(legend(ax, htmp, legend_labels, 'Location',
      legend_location, 'Orientation', 'vertical'), font_name_axes, font_size_legend, 1);
3761          end
3762      end
3763
3764      if export_pub_figs
3765          export_base = file_base_with_plant_suffix(file_base, Plantas, idx_run);
3766          safe_export_dual(fig, fig, fullfile(pub_fig_folder, [export_base '.png']),
      fullfile(pub_fig_folder, [export_base '.pdf']), fig_res_dpi);
3767      end
3768  end
3769
3770  function htmp = add_compare_series(ax, plant_struct, fieldn, runFlag, style, label_txt, ...
3771      yfield_exact, yfield_paper, line_w, marker_sz, htmp)
3772
3773      if ~runFlag || ~isfield(plant_struct, fieldn)
3774          return;
3775      end
3776
3777      sw = plant_struct.(fieldn);
3778      h1 = plot_sweep_metric(ax, sw, yfield_exact, style, [label_txt ' exacto'], line_w,
      marker_sz);
3779      h2 = plot_sweep_metric_overlay(ax, sw, yfield_paper, style, [label_txt ' paper'],
      max(1.0, line_w - 0.2), '--');
3780
```

```
3781      for h = [h1(:); h2(:)].'
3782          if ~isempty(h) && all(isgraphics(h))
3783              htmp(end+1) = h; %#ok<AGROW>
3784          end
3785      end
3786  end
3787
3788  function h = plot_sweep_metric_overlay(ax, sweep, yfield, style, label_txt, line_w,
      line_style)
3789      if isempty(sweep) || ~isstruct(sweep) || ~isfield(sweep, 'Mst_obj') || ~isfield(sweep,
      yfield)
3790          h = gobjects(0);
3791          return;
3792      end
3793
3794      x = sweep.Mst_obj(:);
3795      y = sweep.(yfield)(:);
3796      mask = isfinite(x) & isfinite(y) & (y > 0 | ~strcmp(get(ax, 'YScale'), 'log'));
3797      if ~any(mask)
3798          h = gobjects(0);
3799          return;
3800      end
3801
3802      h = plot(ax, x(mask), y(mask), ...
3803          'LineStyle', line_style, ...
3804          'Color', style.color, ...
3805          'LineWidth', line_w, ...
3806          'DisplayName', label_txt);
3807  end
3808
3809  function format_metric_axis(ax, font_name_axes, font_size_axes, grid_alpha_major,
      grid_alpha_minor)
3810      grid(ax, 'off');
3811      box(ax, 'off');
3812      ax.XMinorGrid = 'off';
3813      ax.YMinorGrid = 'off';
3814      ax.GridAlpha = grid_alpha_major;
3815      ax.MinorGridAlpha = grid_alpha_minor;
3816      ax.LineWidth = 0.9;
3817      ax.FontName = font_name_axes;
3818      ax.FontSize = font_size_axes;
3819      ax.TickDir = 'out';
3820      ax.TickLength = [0.012 0.012];
3821      ax.Layer = 'top';
3822      ax.XColor = [0.20 0.20 0.20];
3823      ax.YColor = [0.20 0.20 0.20];
3824  end
3825
3826  function apply_publication_legend(lgd, font_name_axes, font_size_legend, ncols)
3827      if isempty(lgd) || ~isgraphics(lgd)
3828          return;
3829      end
3830
3831      lgd.Box = 'off';
3832      lgd.Color = 'none';
3833      lgd.FontName = font_name_axes;
3834      lgd.FontSize = font_size_legend;
3835      lgd.NumColumns = ncols;
3836
3837      try
3838          if isprop(lgd, 'Location') && strcmpi(lgd.Location, 'southoutside')
3839              lgd.Units = 'normalized';
3840              pos = lgd.Position;
3841              pos(2) = max(0.005, pos(2) - 0.02);
3842              lgd.Position = pos;
```

```
3843          end
3844      catch
3845          % Mantener comportamiento por defecto si la version de MATLAB
3846          % no permite ajustar esta propiedad.
3847      end
3848  end
3849
3850  function lgd = create_compatible_legend(layout_or_parent, handles, labels, location_txt,
      orientation_txt)
3851      if nargin < 5 || isempty(orientation_txt)
3852          orientation_txt = 'horizontal';
3853      end
3854
3855      lgd = gobjects(0);
3856      if isempty(handles) || ~all(isgraphics(handles))
3857          return;
3858      end
3859
3860      use_shared_dummy = true;
3861      try
3862          parent_type = get(layout_or_parent, 'Type');
3863          use_shared_dummy = ~strcmpi(parent_type, 'axes');
3864      catch
3865          use_shared_dummy = true;
3866      end
3867
3868      if use_shared_dummy
3869          lgd = create_shared_dummy_legend(layout_or_parent, handles, labels, location_txt,
      orientation_txt);
3870          return;
3871      end
3872
3873      try
3874          lgd = legend(layout_or_parent, handles, labels, 'Location', location_txt,
      'Orientation', orientation_txt);
3875      catch
3876          lgd = legend(handles, labels, 'Location', location_txt, 'Orientation',
      orientation_txt);
3877      end
3878  end
3879
3880  function lgd = create_shared_dummy_legend(layout_or_parent, handles, labels, location_txt,
      orientation_txt)
3881      lgd = gobjects(0);
3882      fig = ancestor(layout_or_parent, 'figure');
3883      if isempty(fig) || ~isgraphics(fig)
3884          return;
3885      end
3886
3887      delete(findall(fig, 'Type', 'axes', 'Tag', 'codex_shared_legend_axes'));
3888
3889      ax_dummy = axes( ...
3890          'Parent', fig, ...
3891          'Units', 'normalized', ...
3892          'Position', [0.04 0.005 0.92 0.12], ...
3893          'Visible', 'off', ...
3894          'Color', 'none', ...
3895          'XColor', 'none', ...
3896          'YColor', 'none', ...
3897          'HitTest', 'off', ...
3898          'HandleVisibility', 'off', ...
3899          'Tag', 'codex_shared_legend_axes');
3900
3901      hold(ax_dummy, 'on');
3902      h_dummy = gobjects(0);
```

```
    labels_ok = {};

    for k = 1:numel(handles)
        hsrc = handles(k);
        if ~isgraphics(hsrc)
            continue;
        end

        line_style = get_graphic_prop_safe(hsrc, 'LineStyle', '-');
        color_val = get_graphic_prop_safe(hsrc, 'Color', [0 0 0]);
        line_w = get_graphic_prop_safe(hsrc, 'LineWidth', 1.0);
        marker_val = get_graphic_prop_safe(hsrc, 'Marker', 'none');
        marker_sz = get_graphic_prop_safe(hsrc, 'MarkerSize', 6);
        marker_face = get_graphic_prop_safe(hsrc, 'MarkerFaceColor', 'none');
        marker_edge = get_graphic_prop_safe(hsrc, 'MarkerEdgeColor', color_val);

        h_dummy(end+1) = plot(ax_dummy, NaN, NaN, ... %#ok<AGROW>
            'LineStyle', line_style, ...
            'Color', color_val, ...
            'LineWidth', line_w, ...
            'Marker', marker_val, ...
            'MarkerSize', marker_sz, ...
            'MarkerFaceColor', marker_face, ...
            'MarkerEdgeColor', marker_edge, ...
            'DisplayName', labels{k});
        labels_ok{end+1} = labels{k}; %#ok<AGROW>
    end

    if isempty(h_dummy)
        delete(ax_dummy);
        return;
    end

    lgd = legend(ax_dummy, h_dummy, labels_ok, 'Location', location_txt, 'Orientation',
orientation_txt);
    lgd.Units = 'normalized';
    switch lower(location_txt)
        case 'southoutside'
            lgd.Position = [0.06 0.005 0.88 0.075];
        case 'northoutside'
            lgd.Position = [0.06 0.925 0.88 0.075];
        case 'eastoutside'
            lgd.Position = [0.84 0.18 0.14 0.62];
        case 'westoutside'
            lgd.Position = [0.02 0.18 0.14 0.62];
    end
end

function val = get_graphic_prop_safe(h, prop_name, default_val)
    try
        val = get(h, prop_name);
        if isempty(val)
            val = default_val;
        end
    catch
        val = default_val;
    end
end

function h = plot_editorial_series(ax, x, y, style, label_txt, line_w, marker_sz,
convention_mode)
    if nargin < 8 || isempty(convention_mode)
        convention_mode = 'default';
    end

```

```
3966      [line_style, marker_sym, marker_face] = get_plot_convention_style(label_txt,
      convention_mode);
3967
3968      h = plot(ax, x, y, ...
3969          'LineStyle', line_style, ...
3970          'Color', style.color, ...
3971          'LineWidth', line_w, ...
3972          'Marker', marker_sym, ...
3973          'MarkerSize', marker_sz, ...
3974          'MarkerFaceColor', marker_face, ...
3975          'DisplayName', label_txt);
3976  end
3977
3978  function [line_style, marker_sym, marker_face] = get_plot_convention_style(label_txt,
      convention_mode)
3979      line_style = '-';
3980      marker_sym = 'none';
3981      marker_face = 'none';
3982
3983      if strcmpi(convention_mode, 'static_markers')
3984          switch upper(strtrim(char(string(label_txt))))
3985              case 'SP'
3986                  marker_sym = 'o';
3987              case 'MSP'
3988                  marker_sym = 's';
3989          end
3990          return;
3991      end
3992
3993      if ~strcmpi(convention_mode, 'temporal_like')
3994          return;
3995      end
3996
3997      switch upper(strtrim(char(string(label_txt))))
3998          case 'PI'
3999              line_style = '-.';
4000          case 'PID'
4001              line_style = '--';
4002          case 'SP'
4003              line_style = '-';
4004              marker_sym = 'o';
4005              marker_face = 'none';
4006          case 'MSP'
4007              line_style = '-';
4008              marker_sym = 's';
4009              marker_face = 'none';
4010      end
4011  end
4012
4013  function sweep = enrich_sweep_with_freq_metrics(plant, ctrl_key, sweep, Go, Gi, pade_order,
      N_der, Tf_min)
4014      fm = compute_sweep_freq_metrics(plant, ctrl_key, sweep, Go, Gi, pade_order, N_der,
      Tf_min);
4015      sweep.DM_opt      = fm.dm;
4016      sweep.GM_opt      = fm.gm;
4017      sweep.GM_dB_opt   = fm.gm_db;
4018      sweep.PM_deg_opt  = fm.pm_deg;
4019      sweep.Wcg_opt     = fm.wcg;
4020      sweep.Wcp_opt     = fm.wcp;
4021      sweep.BW_cl_opt   = fm.bw_cl;
4022      sweep.GM_paper_min_opt    = fm.gm_paper_min;
4023      sweep.GM_paper_min_dB_opt = fm.gm_paper_min_db;
4024      sweep.PM_paper_min_deg_opt = fm.pm_paper_min_deg;
4025      sweep.DM_paper_aprox_opt  = fm.dm_paper_aprox;
4026      sweep.GM_dB_gap_opt       = fm.gm_db_gap;
```

```
4027      sweep.PM_deg_gap_opt      = fm.pm_deg_gap;
4028      sweep.DM_gap_opt          = fm.dm_gap;
4029  end
4030
4031  function sweep = attach_msp_paper_reference(plant, sweep, base_result, Go, Gi, t, pade_order,
      N_der, Tf_min, use_exact_mst, w_mst)
4032      nPts = numel(sweep.Mst_obj);
4033      paper_res = evaluate_paper_msp_benchmark( ...
4034          plant, base_result, Go, Gi, t, pade_order, N_der, Tf_min, use_exact_mst, w_mst);
4035
4036      sweep.Kc_paper_ref = paper_res.kc_opt * ones(nPts, 1);
4037      sweep.Ti_paper_ref = paper_res.ti_opt * ones(nPts, 1);
4038      sweep.Mst_paper_ref = paper_res.mst_opt * ones(nPts, 1);
4039      sweep.J_paper_ref = paper_res.J_opt * ones(nPts, 1);
4040      sweep.paper_status_ref = repmat({paper_res.status}, nPts, 1);
4041      sweep.paper_case_ref = repmat({paper_res.paper_case}, nPts, 1);
4042  end
4043
4044  function Plantas = backfill_freq_metrics_if_missing(Plantas, idx_run, runPI, runPID, runSP,
      runMSP, Go, Gi, t, pade_order, N_der, Tf_min, use_exact_mst, w_mst)
4045      for k = 1:numel(idx_run)
4046          ip = idx_run(k);
4047
4048          if runPI && isfield(Plantas(ip), 'PI_sweep') &&
      sweep_needs_freq_metrics(Plantas(ip).PI_sweep)
4049              Plantas(ip).PI_sweep = enrich_sweep_with_freq_metrics(Plantas(ip), 'PI',
      Plantas(ip).PI_sweep, Go, Gi, pade_order, N_der, Tf_min);
4050          end
4051          if runPID && isfield(Plantas(ip), 'PID_sweep') &&
      sweep_needs_freq_metrics(Plantas(ip).PID_sweep)
4052              Plantas(ip).PID_sweep = enrich_sweep_with_freq_metrics(Plantas(ip), 'PID',
      Plantas(ip).PID_sweep, Go, Gi, pade_order, N_der, Tf_min);
4053          end
4054          if runSP && isfield(Plantas(ip), 'SP_sweep') &&
      sweep_needs_freq_metrics(Plantas(ip).SP_sweep)
4055              Plantas(ip).SP_sweep = enrich_sweep_with_freq_metrics(Plantas(ip), 'SP',
      Plantas(ip).SP_sweep, Go, Gi, pade_order, N_der, Tf_min);
4056          end
4057          if runMSP && isfield(Plantas(ip), 'MSP_sweep') &&
      sweep_needs_freq_metrics(Plantas(ip).MSP_sweep)
4058              Plantas(ip).MSP_sweep = enrich_sweep_with_freq_metrics(Plantas(ip), 'MSP',
      Plantas(ip).MSP_sweep, Go, Gi, pade_order, N_der, Tf_min);
4059          end
4060          if runMSP && isfield(Plantas(ip), 'MSP_sweep') &&
      ~sweep_has_paper_reference(Plantas(ip).MSP_sweep)
4061              Plantas(ip).MSP_sweep = attach_msp_paper_reference( ...
4062                  Plantas(ip), Plantas(ip).MSP_sweep, Plantas(ip).base.result, ...
4063                  Go, Gi, t, pade_order, N_der, Tf_min, use_exact_mst, w_mst);
4064          end
4065      end
4066  end
4067
4068  function tf_need = sweep_needs_freq_metrics(sweep)
4069      tf_need = true;
4070
4071      req_core = {'Mst_obj','Kc_opt','Ti_opt','Td_opt'};
4072      for i = 1:numel(req_core)
4073          if ~isfield(sweep, req_core{i})
4074              return;
4075          end
4076      end
4077
4078      if numel(sweep.Mst_obj) == 0
4079          return;
4080      end
```

```
4081
4082      req_freq = {'DM_opt','GM_opt','GM_dB_opt','PM_deg_opt','Wcg_opt','Wcp_opt','BW_cl_opt',
    ...
4083          'GM_paper_min_opt','GM_paper_min_dB_opt','PM_paper_min_deg_opt','DM_paper_aprox_opt',
    ...
4084          'GM_dB_gap_opt','PM_deg_gap_opt','DM_gap_opt'};
4085      tf_need = false;
4086      n = numel(sweep.Mst_obj);
4087      for i = 1:numel(req_freq)
4088          if ~isfield(sweep, req_freq{i}) || numel(sweep.(req_freq{i})) ~= n
4089              tf_need = true;
4090              return;
4091          end
4092      end
4093  end
4094
4095  function tf_has = sweep_has_paper_reference(sweep)
4096      tf_has = isstruct(sweep) ...
4097          && isfield(sweep, 'Kc_paper_ref') ...
4098          && isfield(sweep, 'Ti_paper_ref') ...
4099          && isfield(sweep, 'Mst_paper_ref') ...
4100          && isfield(sweep, 'J_paper_ref') ...
4101          && numel(sweep.Kc_paper_ref) == numel(sweep.Mst_obj) ...
4102          && numel(sweep.Ti_paper_ref) == numel(sweep.Mst_obj);
4103  end
4104
4105  function fm = compute_sweep_freq_metrics(plant, ctrl_key, sweep, Go, Gi, pade_order, N_der,
    Tf_min)
4106      s = tf('s');
4107      nPts = numel(sweep.Mst_obj);
4108      fm = struct();
4109      fm.gm    = NaN(nPts,1);
4110      fm.gm_db = NaN(nPts,1);
4111      fm.pm_deg = NaN(nPts,1);
4112      fm.wcg   = NaN(nPts,1);
4113      fm.wcp   = NaN(nPts,1);
4114      fm.bw_cl = NaN(nPts,1);
4115      fm.dm    = NaN(nPts,1);
4116      fm.gm_paper_min = NaN(nPts,1);
4117      fm.gm_paper_min_db = NaN(nPts,1);
4118      fm.pm_paper_min_deg = NaN(nPts,1);
4119      fm.dm_paper_aprox = NaN(nPts,1);
4120      fm.gm_db_gap = NaN(nPts,1);
4121      fm.pm_deg_gap = NaN(nPts,1);
4122      fm.dm_gap = NaN(nPts,1);
4123
4124      theta0 = plant.P_dt.InputDelay;
4125      [num0, den0] = tfdata(plant.P_dt, 'v');
4126      P0 = tf(num0, den0);
4127      P = pade(plant.P_dt, pade_order);
4128
4129      for im = 1:numel(sweep.Mst_obj)
4130          kc = sweep.Kc_opt(im);
4131          ti = sweep.Ti_opt(im);
4132          td = sweep.Td_opt(im);
4133
4134          if ~(isfinite(kc) && isfinite(ti) && isfinite(td))
4135              continue;
4136          end
4137
4138          try
4139              switch upper(ctrl_key)
4140                  case 'PI'
4141                      C = build_controller_tf(ctrl_key, kc, ti, td, plant.tipo, plant.id,
    theta0, N_der, Tf_min);
```

```
4142                    L = P * C;
4143                    T = feedback(L, 1);
4144
4145                case 'PID'
4146                    C = build_controller_tf(ctrl_key, kc, ti, td, plant.tipo, plant.id,
    theta0, N_der, Tf_min);
4147                    L = P * C;
4148                    T = feedback(L, 1);
4149
4150                case 'SP'
4151                    C = build_controller_tf(ctrl_key, kc, ti, td, plant.tipo, plant.id,
    theta0, N_der, Tf_min);
4152                    [~, T, ~, L, is_stable] = smith_tf(P0, theta0, theta0, C, Go, Gi,
    pade_order);
4153                    if ~is_stable, continue; end
4154
4155                case 'MSP'
4156                    C = build_controller_tf(ctrl_key, kc, ti, td, plant.tipo, plant.id,
    theta0, N_der, Tf_min);
4157                    [~, T, ~, L, is_stable, ~] = msp_tf(P0, theta0, theta0, C, plant.tipo,
    plant.id, Go, Gi, pade_order);
4158                    if ~is_stable, continue; end
4159
4160                otherwise
4161                    continue;
4162            end
4163
4164            fm_i = local_freq_metrics_from_loop(L, T);
4165            fm.gm(im)     = fm_i.gm;
4166            fm.gm_db(im)  = fm_i.gm_db;
4167            fm.pm_deg(im) = fm_i.pm_deg;
4168            fm.wcg(im)    = fm_i.wcg;
4169            fm.wcp(im)    = fm_i.wcp;
4170            fm.bw_cl(im)  = fm_i.bw_cl;
4171            fm.dm(im)     = fm_i.dm;
4172        catch
4173            % Mantener NaN si el calculo frecuencial falla para un punto.
4174        end
4175    end
4176
4177    mst_vec = getfield_safe(sweep, 'Mst_real', NaN(nPts,1));
4178    mask_mst = isfinite(mst_vec) & mst_vec > 1;
4179    if any(mask_mst)
4180        gm_min = mst_vec(mask_mst) ./ max(mst_vec(mask_mst) - 1, 1e-12);
4181        fm.gm_paper_min(mask_mst) = gm_min;
4182        fm.gm_paper_min_db(mask_mst) = 20 * log10(gm_min);
4183
4184        pm_min_rad = 2 * asin(1 ./ (2 * mst_vec(mask_mst)));
4185        fm.pm_paper_min_deg(mask_mst) = rad2deg(pm_min_rad);
4186
4187        mask_dm = mask_mst & isfinite(fm.wcp) & fm.wcp > 0;
4188        if any(mask_dm)
4189            pm_min_rad_dm = 2 * asin(1 ./ (2 * mst_vec(mask_dm)));
4190            fm.dm_paper_aprox(mask_dm) = pm_min_rad_dm ./ fm.wcp(mask_dm);
4191        end
4192    end
4193
4194    mask_gm_gap = isfinite(fm.gm_db) & isfinite(fm.gm_paper_min_db);
4195    fm.gm_db_gap(mask_gm_gap) = fm.gm_db(mask_gm_gap) - fm.gm_paper_min_db(mask_gm_gap);
4196
4197    mask_pm_gap = isfinite(fm.pm_deg) & isfinite(fm.pm_paper_min_deg);
4198    fm.pm_deg_gap(mask_pm_gap) = fm.pm_deg(mask_pm_gap) - fm.pm_paper_min_deg(mask_pm_gap);
4199
4200    mask_dm_gap = isfinite(fm.dm) & isfinite(fm.dm_paper_aprox);
4201    fm.dm_gap(mask_dm_gap) = fm.dm(mask_dm_gap) - fm.dm_paper_aprox(mask_dm_gap);
```

```
end

function fm = local_freq_metrics_from_loop(L, T)
    fm = struct('gm',NaN,'gm_db',NaN,'pm_deg',NaN,'wcg',NaN,'wcp',NaN,'bw_cl',NaN,'dm',NaN);

    try
        [gm, pm, wcg, wcp] = margin(L);
        fm.gm     = gm;
        fm.pm_deg = pm;
        fm.wcg    = wcg;
        fm.wcp    = wcp;

        if isempty(gm) || ~isfinite(gm) || gm <= 0
            if isinf(gm)
                fm.gm_db = Inf;
            else
                fm.gm_db = NaN;
            end
        else
            fm.gm_db = 20*log10(gm);
        end

        am = allmargin(L);
        [gm_alt, wcg_alt] = extract_margin_pair(am, 'GainMargin', 'GMFrequency');
        [pm_alt, wcp_alt] = extract_margin_pair(am, 'PhaseMargin', 'PMFrequency');

        if (~isfinite(fm.gm) || fm.gm <= 0) && isfinite(gm_alt) && gm_alt > 0
            fm.gm = gm_alt;
            fm.gm_db = 20*log10(gm_alt);
        end
        if (~isfinite(fm.wcg) || fm.wcg <= 0) && isfinite(wcg_alt) && wcg_alt > 0
            fm.wcg = wcg_alt;
        end
        if ~isfinite(fm.pm_deg) && isfinite(pm_alt)
            fm.pm_deg = pm_alt;
        end
        if (~isfinite(fm.wcp) || fm.wcp <= 0) && isfinite(wcp_alt) && wcp_alt > 0
            fm.wcp = wcp_alt;
        end

        vals = am.DelayMargin(:);
        vals = vals(~isnan(vals) & vals >= 0);
        if ~isempty(vals)
            vals_finite = vals(isfinite(vals) & vals > 0);
            if ~isempty(vals_finite)
                fm.dm = min(vals_finite);
            elseif any(isinf(vals))
                fm.dm = Inf;
            elseif any(vals == 0)
                fm.dm = 0;
            end
        end

        try
            fm.bw_cl = bandwidth(T);
        catch
            fm.bw_cl = NaN;
        end

        if ~isfinite(fm.wcp) || fm.wcp <= 0
            fm.wcp = estimate_gain_crossover_from_loop(L);
        end
    catch
        fm =
struct('gm',NaN,'gm_db',NaN,'pm_deg',NaN,'wcg',NaN,'wcp',NaN,'bw_cl',NaN,'dm',NaN);
```

```
4266      end
4267  end
4268
4269  function [margin_val, freq_val] = extract_margin_pair(am, margin_field, freq_field)
4270      margin_val = NaN;
4271      freq_val = NaN;
4272
4273      if ~isstruct(am) || ~isfield(am, margin_field) || ~isfield(am, freq_field)
4274          return;
4275      end
4276
4277      margins = am.(margin_field);
4278      freqs = am.(freq_field);
4279      margins = margins(:);
4280      freqs = freqs(:);
4281
4282      n = min(numel(margins), numel(freqs));
4283      if n == 0
4284          return;
4285      end
4286
4287      margins = margins(1:n);
4288      freqs = freqs(1:n);
4289      mask = isfinite(freqs) & freqs > 0 & ~isnan(margins);
4290      if ~any(mask)
4291          return;
4292      end
4293
4294      idx = find(mask, 1, 'first');
4295      margin_val = margins(idx);
4296      freq_val = freqs(idx);
4297  end
4298
4299  function wc = estimate_gain_crossover_from_loop(L)
4300      wc = NaN;
4301      try
4302          w = logspace(-4, 4, 4000);
4303          Ljw = squeeze(freqresp(L, w));
4304          mag = abs(Ljw(:));
4305          mask = isfinite(mag) & mag > 0;
4306          if nnz(mask) < 2
4307              return;
4308          end
4309
4310          w = w(mask);
4311          mag = mag(mask);
4312          cross_idx = find((mag(1:end-1) - 1) .* (mag(2:end) - 1) <= 0, 1, 'first');
4313          if isempty(cross_idx)
4314              return;
4315          end
4316
4317          w1 = w(cross_idx);
4318          w2 = w(cross_idx + 1);
4319          m1 = mag(cross_idx);
4320          m2 = mag(cross_idx + 1);
4321
4322          if ~isfinite(m1) || ~isfinite(m2) || m1 <= 0 || m2 <= 0 || abs(log10(m2) - log10(m1))
      < 1e-12
4323              wc = w1;
4324              return;
4325          end
4326
4327          logw = [log10(w1), log10(w2)];
4328          logm = [log10(m1), log10(m2)];
4329          wc = 10^(interp1(logm, logw, 0, 'linear', 'extrap'));
```

```
4330      catch
4331          wc = NaN;
4332      end
4333  end
4334
4335  function txt = plant_display_model(plant)
4336      switch upper(plant.id)
4337          case 'P1'
4338              txt = 'G(s) = e^{-s}';
4339          case 'P2'
4340              txt = 'G(s) = e^{-s}/(s+1)';
4341          case 'P3'
4342              txt = 'G(s) = e^{-s}/(8s+1)';
4343          case 'P4'
4344              txt = 'G(s) = e^{-s}/(20s+1)';
4345          case 'P5'
4346              txt = 'G(s) = e^{-s}/s';
4347          case 'P6'
4348              txt = 'G(s) = e^{-s}/(s-1)';
4349          case 'P7'
4350              txt = 'G(s) = e^{-s}/(8s-1)';
4351          otherwise
4352              txt = sprintf('Planta %s', plant.id);
4353      end
4354  end
4355
4356  function txt = plant_title_with_model(plant)
4357      txt = sprintf('Planta %s | %s', plant.id, plant_display_model(plant));
4358  end
4359
4360  function export_base = file_base_with_plant_suffix(file_base, Plantas, idx_run)
4361      export_base = file_base;
4362      if numel(idx_run) == 1
4363          export_base = sprintf('%s_%s', file_base, Plantas(idx_run).id);
4364      end
4365  end
4366
4367  function T = build_search_report_table(Plantas, idx_run, runPI, runPID, runSP, runMSP,
      report_mst_ref)
4368      vars = {'planta','tipo','ctrl','n_mst','n_ok','n_far_mst','n_no_cand','cobertura_ok_pct',
      ...
4369          'mst_real_min_ok','mst_real_max_ok','mean_mst_err_ok','max_mst_err_ok', ...
4370          'j_best','mst_at_j_best','dm_at_j_best','gm_db_at_j_best','pm_deg_at_j_best', ...
4371          'j_at_mst_ref','mst_real_at_ref','dm_at_ref','estado_busqueda'};
4372      rows = cell(0, numel(vars));
4373
4374      ctrl_list = {};
4375      if runPI,  ctrl_list(end+1,:) = {'PI',  'PI_sweep'};  end %#ok<AGROW>
4376      if runPID, ctrl_list(end+1,:) = {'PID', 'PID_sweep'}; end %#ok<AGROW>
4377      if runSP,  ctrl_list(end+1,:) = {'SP',  'SP_sweep'};  end %#ok<AGROW>
4378      if runMSP, ctrl_list(end+1,:) = {'MSP', 'MSP_sweep'}; end %#ok<AGROW>
4379
4380      for k = 1:numel(idx_run)
4381          ip = idx_run(k);
4382          for ic = 1:size(ctrl_list,1)
4383              ctrl_key = ctrl_list{ic,1};
4384              fieldn = ctrl_list{ic,2};
4385              if ~isfield(Plantas(ip), fieldn)
4386                  continue;
4387              end
4388
4389              sw = Plantas(ip).(fieldn);
4390              n_mst = numel(sw.Mst_obj);
4391              status = string(sw.status(:));
4392              mask_ok = status == "OK";
```

```
            mask_far = status == "FAR_MST";
            mask_no_cand = status == "NO_CAND";
            mask_ok_finite = mask_ok & isfinite(sw.J_opt(:)) & isfinite(sw.Mst_real(:));

            n_ok = sum(mask_ok);
            n_far = sum(mask_far);
            n_no = sum(mask_no_cand);
            cobertura = 100 * n_ok / max(n_mst, 1);

            mst_min = NaN; mst_max = NaN; mean_err = NaN; max_err = NaN;
            j_best = NaN; mst_best = NaN; dm_best = NaN; gm_best = NaN; pm_best = NaN;
            j_ref = NaN; mst_ref_real = NaN; dm_ref = NaN;

            dm_vec = getfield_safe(sw, 'DM_opt', NaN(size(sw.Mst_obj)));
            gm_db_vec = getfield_safe(sw, 'GM_dB_opt', NaN(size(sw.Mst_obj)));
            pm_deg_vec = getfield_safe(sw, 'PM_deg_opt', NaN(size(sw.Mst_obj)));

            if any(mask_ok_finite)
                mst_ok = sw.Mst_real(mask_ok_finite);
                err_ok = sw.Mst_err(mask_ok_finite);
                mst_min = min(mst_ok);
                mst_max = max(mst_ok);
                mean_err = mean(err_ok, 'omitnan');
                max_err = max(err_ok);

                idx_ok = find(mask_ok_finite);
                [j_best, rel_best] = min(sw.J_opt(mask_ok_finite));
                idx_best = idx_ok(rel_best);
                mst_best = sw.Mst_real(idx_best);
                if numel(dm_vec) >= idx_best, dm_best = dm_vec(idx_best); end
                if numel(gm_db_vec) >= idx_best, gm_best = gm_db_vec(idx_best); end
                if numel(pm_deg_vec) >= idx_best, pm_best = pm_deg_vec(idx_best); end

                [~, rel_ref] = min(abs(sw.Mst_real(mask_ok_finite) - report_mst_ref));
                idx_ref = idx_ok(rel_ref);
                j_ref = sw.J_opt(idx_ref);
                mst_ref_real = sw.Mst_real(idx_ref);
                if numel(dm_vec) >= idx_ref, dm_ref = dm_vec(idx_ref); end
            end

            if n_ok == n_mst && n_mst > 0
                estado = 'COMPLETA';
            elseif n_ok > 0
                estado = 'PARCIAL';
            elseif n_far > 0
                estado = 'SOLO_FAR_MST';
            else
                estado = 'SIN_SOL';
            end

            rows(end+1,:) = { ...
                Plantas(ip).id, Plantas(ip).tipo, ctrl_key_to_user(ctrl_key), ...
                n_mst, n_ok, n_far, n_no, cobertura, ...
                mst_min, mst_max, mean_err, max_err, ...
                j_best, mst_best, dm_best, gm_best, pm_best, ...
                j_ref, mst_ref_real, dm_ref, estado}; %#ok<AGROW>
        end
    end

    T = cell2table(rows, 'VariableNames', vars);
end

function T = build_search_winner_report_table(T_search_report)
    vars = {'planta','tipo','ctrl_best_j','j_best','ctrl_best_dm_ref','dm_at_ref', ...
        'ctrl_best_tracking','mean_mst_err_ok','ctrl_best_coverage','cobertura_ok_pct'};
```

```
4458      rows = cell(0, numel(vars));
4459
4460      if isempty(T_search_report)
4461          T = cell2table(rows, 'VariableNames', vars);
4462          return;
4463      end
4464
4465      plantas = unique(T_search_report.planta, 'stable');
4466      for i = 1:numel(plantas)
4467          mask_p = strcmp(T_search_report.planta, plantas{i});
4468          Tp = T_search_report(mask_p,:);
4469          tipo = Tp.tipo{1};
4470
4471          [ctrl_best_j, j_best] = pick_best_row(Tp.ctrl, Tp.j_best);
4472          [ctrl_best_dm_ref, dm_ref] = pick_max_row(Tp.ctrl, Tp.dm_at_ref);
4473          [ctrl_best_tracking, mean_err] = pick_best_row(Tp.ctrl, Tp.mean_mst_err_ok);
4474          [ctrl_best_coverage, coverage] = pick_max_row(Tp.ctrl, Tp.cobertura_ok_pct);
4475
4476          rows(end+1,:) = {plantas{i}, tipo, ctrl_best_j, j_best, ctrl_best_dm_ref, dm_ref, ...
4477              ctrl_best_tracking, mean_err, ctrl_best_coverage, coverage}; %#ok<AGROW>
4478      end
4479
4480      T = cell2table(rows, 'VariableNames', vars);
4481  end
4482
4483  function T = build_search_magnitude_table(Plantas, idx_run, runPI, runPID, runSP, runMSP)
4484      vars =
     {'planta','tipo','ctrl','n_mst','n_ok','eval_ref_total','iter_total','cand_band_total','scan_u
     ses','warm_uses'};
4485      rows = cell(0, numel(vars));
4486
4487      ctrl_list = {};
4488      if runPI,  ctrl_list(end+1,:) = {'PI',  'PI_sweep'};  end %#ok<AGROW>
4489      if runPID, ctrl_list(end+1,:) = {'PID', 'PID_sweep'}; end %#ok<AGROW>
4490      if runSP,  ctrl_list(end+1,:) = {'SP',  'SP_sweep'};  end %#ok<AGROW>
4491      if runMSP, ctrl_list(end+1,:) = {'MSP', 'MSP_sweep'}; end %#ok<AGROW>
4492
4493      for k = 1:numel(idx_run)
4494          ip = idx_run(k);
4495          for ic = 1:size(ctrl_list,1)
4496              ctrl_key = ctrl_list{ic,1};
4497              fieldn = ctrl_list{ic,2};
4498              if ~isfield(Plantas(ip), fieldn)
4499                  continue;
4500              end
4501
4502              sw = Plantas(ip).(fieldn);
4503              [eval_ref_total, iter_total, cand_band_total] =
     summarize_sweep_iter_logs(getfield_safe(sw, 'iter_log', {}));
4504              status = string(sw.status(:));
4505
4506              rows(end+1,:) = { ...
4507                  Plantas(ip).id, ...
4508                  Plantas(ip).tipo, ...
4509                  ctrl_key_to_user(ctrl_key), ...
4510                  numel(sw.Mst_obj), ...
4511                  sum(status == "OK"), ...
4512                  eval_ref_total, ...
4513                  iter_total, ...
4514                  cand_band_total, ...
4515                  sum(strcmp(sw.bounds_mode, 'SCAN')), ...
4516                  sum(strcmp(sw.bounds_mode, 'WARM')) ...
4517                  }; %#ok<AGROW>
4518          end
4519      end
```

```
4520
4521      T = cell2table(rows, 'VariableNames', vars);
4522  end
4523
4524  function [eval_total, iter_total, cand_total] = summarize_sweep_iter_logs(iter_log_by_mst)
4525      eval_total = 0;
4526      iter_total = 0;
4527      cand_total = 0;
4528
4529      if isempty(iter_log_by_mst)
4530          return;
4531      end
4532
4533      for im = 1:numel(iter_log_by_mst)
4534          iter_log = iter_log_by_mst{im};
4535          if isempty(iter_log)
4536              continue;
4537          end
4538          for ir = 1:numel(iter_log)
4539              row = iter_log{ir};
4540              if ~isstruct(row)
4541                  continue;
4542              end
4543              eval_total = eval_total + getfield_safe(row, 'n_eval', 0);
4544              iter_total = iter_total + 1;
4545              cand_total = cand_total + getfield_safe(row, 'cand', 0);
4546          end
4547      end
4548  end
4549
4550  function print_search_report_table(T)
4551      if isempty(T)
4552          fprintf('  Sin datos de busqueda para reportar.\n');
4553          return;
4554      end
4555
4556      fmt = '  %-6s | %-10s | %-7s | %-7s | %-6s | %-8s | %-8s | %-8s | %-8s | %-8s | %-9s\n';
4557      fprintf(fmt, 'Planta', 'Tipo', 'Ctrl', 'OK/Tot', 'Cob%', 'Estado', 'MejorJ', 'Mst@J',
    'DM@J', 'J@Mref', 'ErrMed');
4558      fprintf('  %s\n', repmat('-',1,112));
4559      for i = 1:height(T)
4560          ok_txt = sprintf('%s/%s', fmt_int_or_na(table_value_or_nan(T, 'n_ok', i)),
    fmt_int_or_na(table_value_or_nan(T, 'n_mst', i)));
4561          fprintf(fmt, ...
4562              table_text_or_na(T, 'planta', i), ...
4563              clip_str(table_text_or_na(T, 'tipo', i),10), ...
4564              clip_str(table_text_or_na(T, 'ctrl', i),7), ...
4565              ok_txt, ...
4566              fmt_num_short(table_value_or_nan(T, 'cobertura_ok_pct', i), 1), ...
4567              clip_str(table_text_or_na(T, 'estado_busqueda', i),8), ...
4568              fmt_num_short(table_value_or_nan(T, 'j_best', i), 3), ...
4569              fmt_num_short(table_value_or_nan(T, 'mst_at_j_best', i), 3), ...
4570              fmt_num_short(table_value_or_nan(T, 'dm_at_j_best', i), 3), ...
4571              fmt_num_short(table_value_or_nan(T, 'j_at_mst_ref', i), 3), ...
4572              fmt_num_short(table_value_or_nan(T, 'mean_mst_err_ok', i), 3));
4573      end
4574  end
4575
4576  function print_search_column_guide()
4577      lines = {
4578          'OK/Tot: soluciones OK sobre total de Mst pedidos'
4579          'Cob%: porcentaje de Mst con solucion OK'
4580          'MejorJ: menor J hallado en todo el barrido'
4581          'Mst@J: Mst_real del punto donde ocurrio MejorJ'
4582          'DM@J: Delay Margin del punto donde ocurrio MejorJ'
```

```
4583          'J@Mref: J del punto OK mas cercano al Mst de referencia'
4584          'ErrMed: error medio |Mst_real - Mst_obj| en puntos OK'
4585          'Mejor DMref: controlador con mayor DM en el Mst de referencia'
4586          'Mejor Track: controlador con menor ErrMed'
4587          'Mas Cobertura: controlador con mayor Cob%'
4588          };
4589      for i = 1:numel(lines)
4590          fprintf('  - %s\n', lines{i});
4591      end
4592  end
4593
4594  function print_compact_base_report(Plantas, idx_run)
4595      fmt = '  %-6s | %-10s | %-8s | %-12s | %-8s | %-8s | %-8s | %-8s | %-10s | %-10s\n';
4596      fprintf(fmt, 'Planta', 'Tipo', 'Base', 'Status', 'k_c', 'tau_i', 'tau_d', 'Mst',
    'IAEdo0', 'IAEdi0');
4597      fprintf('  %s\n', repmat('-',1,112));
4598
4599      for k = 1:numel(idx_run)
4600          ip = idx_run(k);
4601          if ~isfield(Plantas(ip), 'base') || ~isfield(Plantas(ip).base, 'result')
4602              fprintf(fmt, Plantas(ip).id, clip_str(Plantas(ip).tipo,10), 'NA', 'NA', 'NA',
    'NA', 'NA', 'NA', 'NA', 'NA');
4603              continue;
4604          end
4605
4606          base_result = Plantas(ip).base.result;
4607          fprintf(fmt, ...
4608              Plantas(ip).id, ...
4609              clip_str(Plantas(ip).tipo,10), ...
4610              clip_str(getfield_safe(Plantas(ip).base,'ctrl_key','NA'),8), ...
4611              clip_str(getfield_safe(Plantas(ip).base,'status','NA'),12), ...
4612              fmt_num(base_result.do0.kc_do0), ...
4613              fmt_num(base_result.do0.ti_do0), ...
4614              fmt_num(base_result.do0.td_do0), ...
4615              fmt_num(base_result.do0.mst_do0), ...
4616              fmt_num(base_result.do0.iaedo_do0), ...
4617              fmt_num(base_result.di0.iaedi_di0));
4618
4619          if isfield(Plantas(ip).base,'extraordinary_note')
4620              fprintf('    Nota %s: %s\n', Plantas(ip).id, Plantas(ip).base.extraordinary_note);
4621          end
4622      end
4623  end
4624
4625  function print_search_winner_report_table(T)
4626      if isempty(T)
4627          fprintf('  Sin ganadores de busqueda para reportar.\n');
4628          return;
4629      end
4630
4631      fmt = '  %-6s | %-10s | %-12s | %-8s | %-12s | %-8s | %-14s | %-8s | %-14s | %-6s\n';
4632      fprintf(fmt, 'Planta', 'Tipo', 'Mejor J', 'J', 'Mejor DMref', 'DMref', 'Mejor Track',
    'ErrMed', 'Mas Cobertura', 'Cob%');
4633      fprintf('  %s\n', repmat('-',1,126));
4634      for i = 1:height(T)
4635          fprintf(fmt, ...
4636              table_text_or_na(T, 'planta', i), ...
4637              clip_str(table_text_or_na(T, 'tipo', i),10), ...
4638              clip_str(table_text_or_na(T, 'ctrl_best_j', i),12), ...
4639              fmt_num_short(table_value_or_nan(T, 'j_best', i), 3), ...
4640              clip_str(table_text_or_na(T, 'ctrl_best_dm_ref', i),12), ...
4641              fmt_num_short(table_value_or_nan(T, 'dm_at_ref', i), 3), ...
4642              clip_str(table_text_or_na(T, 'ctrl_best_tracking', i),14), ...
4643              fmt_num_short(table_value_or_nan(T, 'mean_mst_err_ok', i), 3), ...
4644              clip_str(table_text_or_na(T, 'ctrl_best_coverage', i),14), ...
```

```
4645              fmt_num_short(table_value_or_nan(T, 'cobertura_ok_pct', i), 1));
4646      end
4647  end
4648
4649  function print_search_magnitude_table(T)
4650      if isempty(T)
4651          fprintf('  Sin datos de magnitud de busqueda para reportar.\n');
4652          return;
4653      end
4654
4655      fmt = '  %-6s | %-10s | %-7s | %-6s | %-6s | %-8s | %-8s | %-8s | %-6s | %-6s\n';
4656      fprintf(fmt, 'Planta', 'Tipo', 'Ctrl', 'Mst', 'OK', 'EvalRef', 'Iter', 'CandBand',
    'SCAN', 'WARM');
4657      fprintf('  %s\n', repmat('-',1,100));
4658      for i = 1:height(T)
4659          fprintf(fmt, ...
4660              table_text_or_na(T, 'planta', i), ...
4661              clip_str(table_text_or_na(T, 'tipo', i),10), ...
4662              clip_str(table_text_or_na(T, 'ctrl', i),7), ...
4663              fmt_int_or_na(table_value_or_nan(T, 'n_mst', i)), ...
4664              fmt_int_or_na(table_value_or_nan(T, 'n_ok', i)), ...
4665              fmt_int_or_na(table_value_or_nan(T, 'eval_ref_total', i)), ...
4666              fmt_int_or_na(table_value_or_nan(T, 'iter_total', i)), ...
4667              fmt_int_or_na(table_value_or_nan(T, 'cand_band_total', i)), ...
4668              fmt_int_or_na(table_value_or_nan(T, 'scan_uses', i)), ...
4669              fmt_int_or_na(table_value_or_nan(T, 'warm_uses', i)));
4670      end
4671  end
4672
4673  function write_final_search_annex_report( ...
4674      filename, Plantas, idx_run, T_base, T_sweep, T_freq, T_extra, T_summary, T_winner, ...
4675      T_search_report, T_search_winner, T_search_magnitude, code_version, baseCtrl, baseMode,
    ctrls_sel_txt, ...
4676      plants_sel_txt, Mst_base, mst_min, mst_max, mst_step, mst_grid, report_mst_ref)
4677
4678      fid = fopen(filename, 'w');
4679      if fid == -1
4680          error('No se pudo abrir el reporte final para escritura: %s', filename);
4681      end
4682      cleaner = onCleanup(@() fclose(fid)); %#ok<NASGU>
4683
4684      print_final_search_annex_report( ...
4685          fid, Plantas, idx_run, T_base, T_sweep, T_freq, T_extra, T_summary, T_winner, ...
4686          T_search_report, T_search_winner, T_search_magnitude, code_version, baseCtrl,
    baseMode, ctrls_sel_txt, ...
4687          plants_sel_txt, Mst_base, mst_min, mst_max, mst_step, mst_grid, report_mst_ref);
4688  end
4689
4690  function print_final_search_annex_report( ...
4691      fid, Plantas, idx_run, T_base, T_sweep, T_freq, T_extra, T_summary, T_winner, ...
4692      T_search_report, T_search_winner, T_search_magnitude, code_version, baseCtrl, baseMode,
    ctrls_sel_txt, ...
4693      plants_sel_txt, Mst_base, mst_min, mst_max, mst_step, mst_grid, report_mst_ref)
4694
4695      fprintf(fid, 'REPORTE FINAL DE BUSQUEDA PARA ANEXO\n');
4696      fprintf(fid, '%s\n', repmat('=',1,110));
4697      fprintf(fid, 'Codigo       : %s\n', code_version);
4698      fprintf(fid, 'Plantas      : %s\n', plants_sel_txt);
4699      fprintf(fid, 'Controladores: %s\n', ctrls_sel_txt);
4700      fprintf(fid, 'Mst barrido  : %.4f a %.4f | paso %.4f | n=%d | grid=%s\n', ...
4701          mst_min, mst_max, mst_step, numel(mst_grid), vec2str(mst_grid, '%.4f'));
4702      fprintf(fid, 'Pesos base   : %s | Mst_base %.4f | modo %s\n', baseCtrl, Mst_base,
    baseMode);
4703      fprintf(fid, 'Mst ref      : %.4f\n', report_mst_ref);
4704      fprintf(fid, 'Criterio OK  : Mst_real dentro de la tolerancia final h_used\n');
```

```
4705      fprintf(fid, 'Criterio FAR : solucion estable calculada, pero fuera de tolerancia\n');
4706      fprintf(fid, 'Criterio NO  : no hubo candidatos validos\n\n');
4707
4708      fprintf(fid, '1. PESOS BASE\n');
4709      fprintf(fid, '%s\n', repmat('-',1,110));
4710      print_base_table_for_annex(fid, T_base);
4711
4712      fprintf(fid, '\n1. INDICE RAPIDO DE COLUMNAS\n');
4713      fprintf(fid, '%s\n', repmat('-',1,110));
4714      print_search_column_guide_to_fid(fid);
4715
4716      fprintf(fid, '\n2. RESUMEN POR CONTROLADOR (cobertura y mejor punto encontrado)\n');
4717      fprintf(fid, '%s\n', repmat('-',1,110));
4718      fprintf(fid, 'Mst de referencia para comparar J y DM: %.2f\n', report_mst_ref);
4719      fprintf(fid, 'OK/Tot = soluciones validas / total de Mst pedidos.\n\n');
4720      print_search_report_table_to_fid(fid, T_search_report);
4721
4722      fprintf(fid, '\n3. GANADORES POR PLANTA (quien gana cada criterio)\n');
4723      fprintf(fid, '%s\n', repmat('-',1,110));
4724      fprintf(fid, 'Mejor J = menor J. Mejor DMref = mayor DM en el Mst de referencia. Mejor
      Track = menor error medio de seguimiento de Mst.\n\n');
4725      print_search_winner_report_table_to_fid(fid, T_search_winner);
4726
4727      fprintf(fid, '\n4. MAGNITUD DE LA BUSQUEDA\n');
4728      fprintf(fid, '%s\n', repmat('-',1,110));
4729      fprintf(fid, 'EvalRef = puntos evaluados en mallas de refinamiento. CandBand = candidatos
      que entraron a banda durante el refinamiento.\n\n');
4730      print_search_magnitude_table_to_fid(fid, T_search_magnitude);
4731
4732      for k = 1:numel(idx_run)
4733          ip = idx_run(k);
4734          plant_id = Plantas(ip).id;
4735          fprintf(fid, '\n%s\n', repmat('=',1,110));
4736          fprintf(fid, '5.%d RESULTADOS DE BUSQUEDA - PLANTA %s | %s\n', k, plant_id,
      plant_display_model(Plantas(ip)));
4737          fprintf(fid, '%s\n', repmat('=',1,110));
4738
4739          Tp_summary = T_summary(strcmp(T_summary.planta, plant_id), :);
4740          Tp_winner = T_winner(strcmp(T_winner.planta, plant_id), :);
4741          Tp_sweep = T_sweep(strcmp(T_sweep.planta, plant_id), :);
4742          Tp_extra = T_extra(strcmp(T_extra.planta, plant_id), :);
4743
4744          fprintf(fid, '\n5.%d.1 Resumen por controlador\n', k);
4745          fprintf(fid, '%s\n', repmat('-',1,110));
4746          print_summary_table_for_annex(fid, Tp_summary);
4747
4748          fprintf(fid, '\n5.%d.2 Ganadores de metricas\n', k);
4749          fprintf(fid, '%s\n', repmat('-',1,110));
4750          print_winner_table_for_annex(fid, Tp_winner);
4751
4752          fprintf(fid, '\n5.%d.3 Resultados validos OK\n', k);
4753          fprintf(fid, '%s\n', repmat('-',1,110));
4754          print_sweep_results_for_annex(fid, Tp_sweep(strcmp(Tp_sweep.status, 'OK'), :), true);
4755
4756          fprintf(fid, '\n5.%d.4 Resultados no validos o fuera de tolerancia\n', k);
4757          fprintf(fid, '%s\n', repmat('-',1,110));
4758          print_sweep_results_for_annex(fid, Tp_sweep(~strcmp(Tp_sweep.status, 'OK'), :),
      false);
4759
4760          fprintf(fid, '\n5.%d.5 Casos extraordinarios\n', k);
4761          fprintf(fid, '%s\n', repmat('-',1,110));
4762          print_extra_table_for_annex(fid, Tp_extra);
4763      end
4764  end
4765
```

```
4766  function print_base_table_for_annex(fid, T)
4767      fmt = '%-7s | %-10s | %-8s | %-12s | %-8s | %-8s | %-8s | %-8s | %-10s | %-10s\n';
4768      fprintf(fid, fmt, 'Planta', 'Tipo', 'Base', 'Status', 'k_c', 'tau_i', 'tau_d', 'Mst',
      'IAEdo0', 'IAEdi0');
4769      fprintf(fid, '%s\n', repmat('-',1,114));
4770      for i = 1:height(T)
4771          fprintf(fid, fmt, table_text_or_na(T, 'planta', i), table_text_or_na(T, 'tipo', i),
      ...
4772              table_text_or_na(T, 'base_ctrl', i), table_text_or_na(T, 'base_status', i), ...
4773              fmt_num(table_value_or_nan(T, 'kc_do0', i)), fmt_num(table_value_or_nan(T,
      'ti_do0', i)), ...
4774              fmt_num(table_value_or_nan(T, 'td_do0', i)), fmt_num(table_value_or_nan(T,
      'mst_do0', i)), ...
4775              fmt_num(table_value_or_nan(T, 'iaedo0_base', i)), fmt_num(table_value_or_nan(T,
      'iaedi0_base', i)));
4776      end
4777  end
4778
4779  function print_search_report_table_to_fid(fid, T)
4780      fmt = '%-7s | %-10s | %-8s | %-7s | %-6s | %-12s | %-8s | %-8s | %-8s | %-8s | %-9s\n';
4781      fprintf(fid, fmt, 'Planta', 'Tipo', 'Ctrl', 'OK/Tot', 'Cob%', 'Estado', 'MejorJ',
      'Mst@J', 'DM@J', 'J@Mref', 'ErrMed');
4782      fprintf(fid, '%s\n', repmat('-',1,114));
4783      for i = 1:height(T)
4784          ok_txt = sprintf('%s/%s', fmt_int_or_na(table_value_or_nan(T, 'n_ok', i)),
      fmt_int_or_na(table_value_or_nan(T, 'n_mst', i)));
4785          fprintf(fid, fmt, table_text_or_na(T, 'planta', i), table_text_or_na(T, 'tipo', i),
      ...
4786              table_text_or_na(T, 'ctrl', i), ok_txt, fmt_num_short(table_value_or_nan(T,
      'cobertura_ok_pct', i), 1), ...
4787              table_text_or_na(T, 'estado_busqueda', i), fmt_num_short(table_value_or_nan(T,
      'j_best', i), 3), ...
4788              fmt_num_short(table_value_or_nan(T, 'mst_at_j_best', i), 3),
      fmt_num_short(table_value_or_nan(T, 'dm_at_j_best', i), 3), ...
4789              fmt_num_short(table_value_or_nan(T, 'j_at_mst_ref', i), 3),
      fmt_num_short(table_value_or_nan(T, 'mean_mst_err_ok', i), 3));
4790      end
4791  end
4792
4793  function print_search_column_guide_to_fid(fid)
4794      lines = {
4795          'OK/Tot: soluciones OK sobre total de Mst pedidos'
4796          'Cob%: porcentaje de Mst con solucion OK'
4797          'MejorJ: menor J hallado en todo el barrido'
4798          'Mst@J: Mst_real del punto donde ocurrio MejorJ'
4799          'DM@J: Delay Margin del punto donde ocurrio MejorJ'
4800          'J@Mref: J del punto OK mas cercano al Mst de referencia'
4801          'ErrMed: error medio |Mst_real - Mst_obj| en puntos OK'
4802          'Mejor DMref: controlador con mayor DM en el Mst de referencia'
4803          'Mejor Track: controlador con menor ErrMed'
4804          'Mas Cobertura: controlador con mayor Cob%'
4805          };
4806      for i = 1:numel(lines)
4807          fprintf(fid, '  - %s\n', lines{i});
4808      end
4809  end
4810
4811  function print_search_winner_report_table_to_fid(fid, T)
4812      fmt = '%-7s | %-10s | %-10s | %-8s | %-11s | %-8s | %-12s | %-8s | %-12s | %-6s\n';
4813      fprintf(fid, fmt, 'Planta', 'Tipo', 'MejorJ', 'J', 'MejorDMref', 'DMref', 'MejorTrack',
      'ErrMed', 'Cobertura', 'Cob%');
4814      fprintf(fid, '%s\n', repmat('-',1,112));
4815      for i = 1:height(T)
4816          fprintf(fid, fmt, table_text_or_na(T, 'planta', i), table_text_or_na(T, 'tipo', i),
      ...
```

```
            table_text_or_na(T, 'ctrl_best_j', i), fmt_num_short(table_value_or_nan(T, 'j_best', i), 3), ...
            table_text_or_na(T, 'ctrl_best_dm_ref', i), fmt_num_short(table_value_or_nan(T, 'dm_at_ref', i), 3), ...
            table_text_or_na(T, 'ctrl_best_tracking', i), fmt_num_short(table_value_or_nan(T, 'mean_mst_err_ok', i), 3), ...
            table_text_or_na(T, 'ctrl_best_coverage', i), fmt_num_short(table_value_or_nan(T, 'cobertura_ok_pct', i), 1));
    end
end

function print_search_magnitude_table_to_fid(fid, T)
    if isempty(T)
        fprintf(fid, '  Sin datos de magnitud de busqueda para reportar.\n');
        return;
    end

    fmt = '%-7s | %-10s | %-8s | %-6s | %-6s | %-8s | %-8s | %-8s | %-6s | %-6s\n';
    fprintf(fid, fmt, 'Planta', 'Tipo', 'Ctrl', 'Mst', 'OK', 'EvalRef', 'Iter', 'CandBand', 'SCAN', 'WARM');
    fprintf(fid, '%s\n', repmat('-',1,102));
    for i = 1:height(T)
        fprintf(fid, fmt, ...
            table_text_or_na(T, 'planta', i), ...
            table_text_or_na(T, 'tipo', i), ...
            table_text_or_na(T, 'ctrl', i), ...
            fmt_int_or_na(table_value_or_nan(T, 'n_mst', i)), ...
            fmt_int_or_na(table_value_or_nan(T, 'n_ok', i)), ...
            fmt_int_or_na(table_value_or_nan(T, 'eval_ref_total', i)), ...
            fmt_int_or_na(table_value_or_nan(T, 'iter_total', i)), ...
            fmt_int_or_na(table_value_or_nan(T, 'cand_band_total', i)), ...
            fmt_int_or_na(table_value_or_nan(T, 'scan_uses', i)), ...
            fmt_int_or_na(table_value_or_nan(T, 'warm_uses', i)));
    end
end

function print_summary_table_for_annex(fid, T)
    if isempty(T)
        fprintf(fid, 'Sin resumen disponible.\n');
        return;
    end
    fmt = '%-8s | %-3s | %-3s | %-7s | %-8s | %-8s | %-8s | %-10s | %-10s | %-8s | %-8s\n';
    fprintf(fid, fmt, 'Ctrl', 'OK', 'FAR', 'NO_CAND', 'Jbest', 'Mst@J', 'Err@J', 'IAEdo', 'IAEdi', 'DMbest', 'BWbest');
    fprintf(fid, '%s\n', repmat('-',1,108));
    for i = 1:height(T)
        fprintf(fid, fmt, table_text_or_na(T, 'ctrl', i), ...
            fmt_int_or_na(table_value_or_nan(T, 'n_ok', i)), ...
            fmt_int_or_na(table_value_or_nan(T, 'n_far_mst', i)), ...
            fmt_int_or_na(table_value_or_nan(T, 'n_no_cand', i)), ...
            fmt_num_brief(table_value_or_nan(T, 'j_best', i)), ...
            fmt_num_brief(table_value_or_nan(T, 'mst_at_j_best', i)), ...
            fmt_num_brief(table_value_or_nan(T, 'mst_err_best', i)), ...
            fmt_num_brief(table_value_or_nan(T, 'iaedo_best', i)), ...
            fmt_num_brief(table_value_or_nan(T, 'iaedi_best', i)), ...
            fmt_num_brief(table_value_or_nan(T, 'dm_best', i)), ...
            fmt_num_brief(table_value_or_nan(T, 'bw_cl_best', i)));
    end
end

function print_winner_table_for_annex(fid, T)
    if isempty(T)
        fprintf(fid, 'Sin ganadores disponibles.\n');
        return;
    end
```

```
4876      fmt = '%-10s | %-8s | %-11s | %-8s | %-11s | %-10s | %-11s | %-10s\n';
4877      fprintf(fid, fmt, 'BestJ', 'J', 'BestTrack', 'Err', 'BestIAEdo', 'IAEdo', 'BestIAEdi',
      'IAEdi');
4878      fprintf(fid, '%s\n', repmat('-',1,96));
4879      for i = 1:height(T)
4880          fprintf(fid, fmt, table_text_or_na(T, 'best_ctrl_j', i),
      fmt_num_brief(table_value_or_nan(T, 'best_j', i)), ...
4881              table_text_or_na(T, 'best_ctrl_tracking', i), fmt_num_brief(table_value_or_nan(T,
      'best_tracking_err', i)), ...
4882              table_text_or_na(T, 'best_ctrl_iaedo', i), fmt_num_brief(table_value_or_nan(T,
      'best_iaedo', i)), ...
4883              table_text_or_na(T, 'best_ctrl_iaedi', i), fmt_num_brief(table_value_or_nan(T,
      'best_iaedi', i)));
4884      end
4885  end
4886
4887  function print_sweep_results_for_annex(fid, T, show_perf)
4888      if isempty(T)
4889          fprintf(fid, 'Sin filas en esta categoria.\n');
4890          return;
4891      end
4892
4893      if show_perf
4894          fmt = '%-8s | %-7s | %-8s | %-8s | %-8s | %-8s | %-8s | %-8s | %-10s | %-10s | %-10s
      | %-8s | %-8s\n';
4895          fprintf(fid, fmt, 'Ctrl', 'MstObj', 'k_c', 'tau_i', 'tau_d', 'MstReal', 'Err', 'Tol',
      'J', 'IAEdo', 'IAEdi', 'DM', 'GMdB');
4896          fprintf(fid, '%s\n', repmat('-',1,126));
4897          for i = 1:height(T)
4898              fprintf(fid, fmt, table_text_or_na(T, 'ctrl', i),
      fmt_num_brief(table_value_or_nan(T, 'mst_obj', i)), ...
4899                  fmt_num_brief(table_value_or_nan(T, 'kc_opt', i)),
      fmt_num_brief(table_value_or_nan(T, 'ti_opt', i)), ...
4900                  fmt_num_brief(table_value_or_nan(T, 'td_opt', i)),
      fmt_num_brief(table_value_or_nan(T, 'mst_real', i)), ...
4901                  fmt_num_brief(table_value_or_nan(T, 'mst_err', i)),
      fmt_num_brief(table_value_or_nan(T, 'mst_tol_used', i)), ...
4902                  fmt_num_brief(table_value_or_nan(T, 'j_opt', i)),
      fmt_num_brief(table_value_or_nan(T, 'iaedo_opt', i)), ...
4903                  fmt_num_brief(table_value_or_nan(T, 'iaedi_opt', i)),
      fmt_num_brief(table_value_or_nan(T, 'dm_opt', i)), ...
4904                  fmt_num_brief(table_value_or_nan(T, 'gm_db_opt', i)));
4905          end
4906      else
4907          fmt = '%-8s | %-7s | %-10s | %-8s | %-8s | %-8s | %-9s | %-14s | %-8s\n';
4908          fprintf(fid, fmt, 'Ctrl', 'MstObj', 'Status', 'nCand', 'MstReal', 'Err', 'J', 'Scan',
      'Mode');
4909          fprintf(fid, '%s\n', repmat('-',1,104));
4910          for i = 1:height(T)
4911              fprintf(fid, fmt, table_text_or_na(T, 'ctrl', i), fmt_num(table_value_or_nan(T,
      'mst_obj', i)), ...
4912                  table_text_or_na(T, 'status', i), fmt_num(table_value_or_nan(T, 'ncand', i)),
      ...
4913                  fmt_num(table_value_or_nan(T, 'mst_real', i)), fmt_num(table_value_or_nan(T,
      'mst_err', i)), ...
4914                  fmt_num(table_value_or_nan(T, 'j_opt', i)), clip_str(table_text_or_na(T,
      'scan_status', i),14), ...
4915                  table_text_or_na(T, 'bounds_mode', i));
4916          end
4917      end
4918  end
4919
4920  function print_extra_table_for_annex(fid, T)
4921      if isempty(T)
4922          fprintf(fid, 'Sin casos extraordinarios.\n');
```

```
4923          return;
4924      end
4925      fmt = '%-8s | %-8s | %-12s | %s\n';
4926      fprintf(fid, fmt, 'Etapa', 'Ctrl', 'Codigo', 'Detalle');
4927      fprintf(fid, '%s\n', repmat('-',1,110));
4928      for i = 1:height(T)
4929          fprintf(fid, fmt, table_text_or_na(T, 'etapa', i), table_text_or_na(T, 'ctrl', i), ...
4930              table_text_or_na(T, 'codigo', i), table_text_or_na(T, 'detalle', i));
4931      end
4932  end
4933
4934  function val = table_value_or_nan(T, varname, idx)
4935      if istable(T) && ismember(varname, T.Properties.VariableNames) && height(T) >= idx
4936          val = T.(varname)(idx);
4937          if iscell(val)
4938              val = val{1};
4939          end
4940          if isstring(val)
4941              val = char(val);
4942          end
4943      else
4944          val = NaN;
4945      end
4946  end
4947
4948  function txt = table_text_or_na(T, varname, idx)
4949      if istable(T) && ismember(varname, T.Properties.VariableNames) && height(T) >= idx
4950          val = T.(varname)(idx);
4951          if iscell(val)
4952              val = val{1};
4953          end
4954          if isstring(val)
4955              val = char(val);
4956          elseif isnumeric(val)
4957              if isfinite(val)
4958                  val = num2str(val);
4959              else
4960                  val = 'NA';
4961              end
4962          else
4963              try
4964                  if any(ismissing(val))
4965                      val = 'NA';
4966                  end
4967              catch
4968                  % Algunas versiones/tipos no admiten ismissing fuera de tablas.
4969              end
4970          end
4971          txt = char(string(val));
4972      else
4973          txt = 'NA';
4974      end
4975  end
4976
4977  function txt = fmt_num_short(x, nd)
4978      if nargin < 2
4979          nd = 3;
4980      end
4981      if isempty(x) || ~isscalar(x) || ~isfinite(x)
4982          txt = 'NA';
4983      else
4984          txt = sprintf(['%.' num2str(nd) 'f'], x);
4985      end
4986  end
4987
```

```
4988  function txt = fmt_int_or_na(x)
4989      if isempty(x) || ~isscalar(x) || ~isfinite(x)
4990          txt = 'NA';
4991      else
4992          txt = sprintf('%.0f', x);
4993      end
4994  end
4995
4996  function txt = fmt_num_brief(x)
4997      if isempty(x) || ~isscalar(x) || ~isfinite(x)
4998          txt = 'NA';
4999          return;
5000      end
5001
5002      ax = abs(x);
5003      if (ax >= 1e4) || (ax > 0 && ax < 1e-3)
5004          txt = sprintf('%.2e', x);
5005      else
5006          txt = sprintf('%.3f', x);
5007      end
5008  end
5009
5010  function annex_xlsx_file = write_search_annex_reports( ...
5011      Plantas, idx_run, T_search_report, T_sweep, ...
5012      code_version, baseCtrl, baseMode, ctrls_sel_txt, Mst_base, mst_min, mst_max, mst_step,
      mst_grid, ...
5013      refinamientos, h_1_vals, refinamientos_pid, h_1_vals_pid, edge_mult, h_relax_factor, ...
5014      scan_paso_global, region_gap_mult, max_regions_keep, region_min_pts, margen_mult,
      max_empty_iters, ...
5015      kc_bounds_global, ti_bounds_global, td_bounds_global, ...
5016      use_exact_mst, w_mst, plot_j_hide_above, report_mst_ref, ...
5017      annex_out_folder, annex_xlsx_file)
5018
5019      if ~exist(annex_out_folder, 'dir')
5020          mkdir(annex_out_folder);
5021      end
5022
5023      workbook_tables = cell(0, 2);
5024
5025      for k = 1:numel(idx_run)
5026          ip = idx_run(k);
5027          plant = Plantas(ip);
5028
5029          T_criteria = build_annex_criteria_table( ...
5030              plant, code_version, baseCtrl, baseMode, Mst_base, mst_min, mst_max, mst_step,
      mst_grid, ...
5031              refinamientos, h_1_vals, refinamientos_pid, h_1_vals_pid, edge_mult,
      h_relax_factor, ...
5032              scan_paso_global, region_gap_mult, max_regions_keep, region_min_pts, margen_mult,
      max_empty_iters, ...
5033              kc_bounds_global, ti_bounds_global, td_bounds_global, ...
5034              use_exact_mst, w_mst, plot_j_hide_above, report_mst_ref);
5035
5036          T_summary_plant = T_search_report(strcmp(T_search_report.planta, plant.id), :);
5037          T_results_plant = T_sweep(strcmp(T_sweep.planta, plant.id), :);
5038
5039          txt_file = fullfile(annex_out_folder, sprintf('anexo_busqueda_%s.txt', plant.id));
5040          write_search_annex_text(txt_file, plant, T_criteria, T_summary_plant,
      T_results_plant);
5041
5042          workbook_tables(end+1,:) = {T_criteria, safe_sheet_name([plant.id '_criterios'])};
      %#ok<AGROW>
5043          workbook_tables(end+1,:) = {T_summary_plant, safe_sheet_name([plant.id '_resumen'])};
      %#ok<AGROW>
```

```
5044          workbook_tables(end+1,:) = {T_results_plant, safe_sheet_name([plant.id
      '_resultados'])}; %#ok<AGROW>
5045      end
5046
5047      if ~isempty(workbook_tables)
5048          annex_xlsx_file = write_tables_to_workbook_safely(workbook_tables, annex_xlsx_file);
5049      end
5050  end
5051
5052  function T = build_annex_criteria_table( ...
5053      plant, code_version, baseCtrl, baseMode, Mst_base, mst_min, mst_max, mst_step, mst_grid,
      ...
5054      refinamientos, h_1_vals, refinamientos_pid, h_1_vals_pid, edge_mult, h_relax_factor, ...
5055      scan_paso_global, region_gap_mult, max_regions_keep, region_min_pts, margen_mult,
      max_empty_iters, ...
5056      kc_bounds_global, ti_bounds_global, td_bounds_global, ...
5057      use_exact_mst, w_mst, plot_j_hide_above, report_mst_ref)
5058
5059      ctrls_plant_txt = get_annex_ctrls_for_plant(plant);
5060
5061      rows = {
5062          'Planta',                  plant.id
5063          'Tipo de planta',          plant.tipo
5064          'Modelo',                  plant_display_model(plant)
5065          'Version de codigo',       code_version
5066          'Controladores evaluados', ctrls_plant_txt
5067          'Controlador base pesos',  baseCtrl
5068          'Modo pesos base',         baseMode
5069          'Mst base pesos',          num2str(Mst_base, '%.4f')
5070          'Mst minimo barrido',      num2str(mst_min, '%.4f')
5071          'Mst maximo barrido',      num2str(mst_max, '%.4f')
5072          'Paso Mst barrido',        num2str(mst_step, '%.4f')
5073          'Malla Mst',               vec2str(mst_grid, '%.4f')
5074          'Mst referencia reporte',  num2str(report_mst_ref, '%.4f')
5075          'Refinamientos generales', vec2str(refinamientos, '%.3f')
5076          'h generales',             vec2str(h_1_vals, '%.3f')
5077          'Refinamientos PID',       vec2str(refinamientos_pid, '%.3f')
5078          'h PID',                   vec2str(h_1_vals_pid, '%.3f')
5079          'Edge mult',               num2str(edge_mult, '%.4f')
5080          'Relax factor h',          num2str(h_relax_factor, '%.4f')
5081          'Paso scan global',        num2str(scan_paso_global, '%.4f')
5082          'Gap regiones',            num2str(region_gap_mult, '%.4f')
5083          'Max regiones retenidas',  num2str(max_regions_keep)
5084          'Min puntos por region',   num2str(region_min_pts)
5085          'Margen region',           num2str(margen_mult, '%.4f')
5086          'Max iteraciones vacias',  num2str(max_empty_iters)
5087          'Bounds k_c',              fmt_rng(kc_bounds_global)
5088          'Bounds tau_i',            fmt_rng(ti_bounds_global)
5089          'Bounds tau_d',            fmt_rng(td_bounds_global)
5090          'Mst exacto',              ternary_txt(use_exact_mst, 'YES', 'NO')
5091          'Puntos w Mst exacto',     num2str(numel(w_mst))
5092          'Ocultar J mayor que',     num2str(plot_j_hide_above, '%.4f')
5093          'Criterio OK',             'Mst_real dentro de la tolerancia final h_used'
5094          'Criterio FAR_MST',        'Solucion estable calculada, pero fuera de tolerancia de
      Mst'
5095          'Criterio NO_CAND',        'No hubo candidatos validos para ese Mst_obj'
5096          };
5097
5098      T = cell2table(rows, 'VariableNames', {'criterio','valor'});
5099  end
5100
5101  function write_search_annex_text(filename, plant, T_criteria, T_summary_plant,
      T_results_plant)
5102      fid = fopen(filename, 'w');
5103      if fid == -1
```

```
5104          error('No se pudo abrir el archivo de anexo: %s', filename);
5105      end
5106      cleaner = onCleanup(@() fclose(fid)); %#ok<NASGU>
5107
5108      fprintf(fid, 'ANEXO DE BUSQUEDA POR PLANTA\n');
5109      fprintf(fid, 'Planta %s | %s\n', plant.id, plant_display_model(plant));
5110      fprintf(fid, '%s\n\n', repmat('=',1,90));
5111
5112      fprintf(fid, '1. CRITERIOS DE BUSQUEDA\n');
5113      fprintf(fid, '%s\n', repmat('-',1,90));
5114      for i = 1:height(T_criteria)
5115          fprintf(fid, '%-32s : %s\n', T_criteria.criterio{i}, T_criteria.valor{i});
5116      end
5117
5118      fprintf(fid, '\n2. RESUMEN DE LA BUSQUEDA\n');
5119      fprintf(fid, '%s\n', repmat('-',1,90));
5120      if isempty(T_summary_plant)
5121          fprintf(fid, 'Sin datos de resumen para esta planta.\n');
5122      else
5123          fprintf(fid, '%-8s | %-8s | %-8s | %-10s | %-10s | %-10s | %-12s\n', ...
5124              'Ctrl', 'OK/Mst', 'CobOK%', 'J_best', 'Mst@J', 'DM@J', 'Estado');
5125          fprintf(fid, '%s\n', repmat('-',1,90));
5126          for i = 1:height(T_summary_plant)
5127              ok_txt = sprintf('%d/%d', T_summary_plant.n_ok(i), T_summary_plant.n_mst(i));
5128              fprintf(fid, '%-8s | %-8s | %-8s | %-10s | %-10s | %-10s | %-12s\n', ...
5129                  T_summary_plant.ctrl{i}, ...
5130                  ok_txt, ...
5131                  fmt_num(T_summary_plant.cobertura_ok_pct(i)), ...
5132                  fmt_num(T_summary_plant.j_best(i)), ...
5133                  fmt_num(T_summary_plant.mst_at_j_best(i)), ...
5134                  fmt_num(T_summary_plant.dm_at_j_best(i)), ...
5135                  T_summary_plant.estado_busqueda{i});
5136          end
5137      end
5138
5139      fprintf(fid, '\n3. RESULTADOS DE LA BUSQUEDA\n');
5140      fprintf(fid, '%s\n', repmat('-',1,90));
5141      if isempty(T_results_plant)
5142          fprintf(fid, 'Sin resultados de barrido para esta planta.\n');
5143      else
5144          fprintf(fid, '%-8s | %-8s | %-10s | %-8s | %-8s | %-8s | %-8s | %-10s | %-10s\n', ...
5145              'Ctrl', 'Mst_obj', 'Status', 'k_c', 'tau_i', 'tau_d', 'Mst', 'J', 'DM');
5146          fprintf(fid, '%s\n', repmat('-',1,108));
5147          for i = 1:height(T_results_plant)
5148              fprintf(fid, '%-8s | %-8s | %-10s | %-8s | %-8s | %-8s | %-8s | %-10s | %-10s\n',
    ...
5149                  T_results_plant.ctrl{i}, ...
5150                  fmt_num(T_results_plant.mst_obj(i)), ...
5151                  clip_str(T_results_plant.status{i},10), ...
5152                  fmt_num(T_results_plant.kc_opt(i)), ...
5153                  fmt_num(T_results_plant.ti_opt(i)), ...
5154                  fmt_num(T_results_plant.td_opt(i)), ...
5155                  fmt_num(T_results_plant.mst_real(i)), ...
5156                  fmt_num(T_results_plant.j_opt(i)), ...
5157                  fmt_num(T_results_plant.dm_opt(i)));
5158          end
5159      end
5160  end
5161
5162  function out = safe_sheet_name(txt)
5163      out = regexprep(txt, '[\\/\?\*\[\]:]', '_');
5164      if numel(out) > 31
5165          out = out(1:31);
5166      end
5167  end
```

```
5168
5169  function workbook_file_out = write_tables_to_workbook_safely(workbook_tables,
      workbook_file_in)
5170      workbook_file_out = workbook_file_in;
5171      last_err = [];
5172
5173      for attempt = 0:20
5174          if attempt == 0
5175              candidate_file = workbook_file_in;
5176          else
5177              [fdir, fname, fext] = fileparts(workbook_file_in);
5178              candidate_file = fullfile(fdir, sprintf('%s_copy%d%s', fname, attempt, fext));
5179          end
5180
5181          try
5182              for i = 1:size(workbook_tables,1)
5183                  writetable(workbook_tables{i,1}, candidate_file, 'Sheet',
      workbook_tables{i,2});
5184              end
5185              if attempt > 0
5186                  fprintf('Aviso: el workbook XLSX original estaba bloqueado o en uso. Se
      exporto una copia alternativa:\n  %s\n', candidate_file);
5187              end
5188              workbook_file_out = candidate_file;
5189              return;
5190          catch err
5191              last_err = err;
5192          end
5193      end
5194
5195      if ~isempty(last_err)
5196          rethrow(last_err);
5197      end
5198  end
5199
5200  function root_dir = resolve_script_root()
5201      script_fullpath = mfilename('fullpath');
5202      if isempty(script_fullpath)
5203          root_dir = pwd;
5204      else
5205          root_dir = fileparts(script_fullpath);
5206      end
5207  end
5208
5209  function txt = get_annex_ctrls_for_plant(plant)
5210      ctrl_labels = {};
5211      if isfield(plant, 'PI_sweep')
5212          ctrl_labels{end+1} = 'PI'; %#ok<AGROW>
5213      end
5214      if isfield(plant, 'PID_sweep')
5215          ctrl_labels{end+1} = 'PID'; %#ok<AGROW>
5216      end
5217      if isfield(plant, 'SP_sweep')
5218          ctrl_labels{end+1} = 'SP'; %#ok<AGROW>
5219      end
5220      if isfield(plant, 'MSP_sweep')
5221          ctrl_labels{end+1} = 'MSP'; %#ok<AGROW>
5222      end
5223
5224      if isempty(ctrl_labels)
5225          txt = 'Sin controladores exportados en esta corrida';
5226      else
5227          txt = strjoin(ctrl_labels, ', ');
5228      end
5229  end
```

```
5230
5231  function T = build_sweep_summary_table(Plantas, idx_run, runPI, runPID, runSP, runMSP)
5232      vars = {'planta','tipo','ctrl','n_ok','n_far_mst','n_no_cand', ...
5233          'j_best','mst_at_j_best','mst_err_best','iaedo_best','iaedi_best', ...
5234          'gm_db_best','pm_deg_best','dm_best','bw_cl_best', ...
5235
      'mean_j_ok','mean_mst_err_ok','mean_gm_db_ok','mean_pm_deg_ok','mean_dm_ok','mean_bw_cl_ok'};
5236      rows = cell(0, numel(vars));
5237
5238      ctrl_list = {};
5239      if runPI,  ctrl_list(end+1,:) = {'PI',  'PI_sweep'};  end %#ok<AGROW>
5240      if runPID, ctrl_list(end+1,:) = {'PID', 'PID_sweep'}; end %#ok<AGROW>
5241      if runSP,  ctrl_list(end+1,:) = {'SP',  'SP_sweep'};  end %#ok<AGROW>
5242      if runMSP, ctrl_list(end+1,:) = {'MSP', 'MSP_sweep'}; end %#ok<AGROW>
5243
5244      for k = 1:numel(idx_run)
5245          ip = idx_run(k);
5246          for ic = 1:size(ctrl_list,1)
5247              ctrl_key = ctrl_list{ic,1};
5248              fieldn = ctrl_list{ic,2};
5249              if ~isfield(Plantas(ip), fieldn)
5250                  continue;
5251              end
5252
5253              sw = Plantas(ip).(fieldn);
5254              status = string(sw.status(:));
5255              mask_ok = status == "OK";
5256              mask_far = status == "FAR_MST";
5257              mask_no_cand = status == "NO_CAND";
5258              mask_valid = mask_ok | mask_far;
5259
5260              j_best = NaN; mst_best = NaN; err_best = NaN; iaedo_best = NaN; iaedi_best = NaN;
5261              gm_db_best = NaN; pm_deg_best = NaN; dm_best = NaN; bw_cl_best = NaN;
5262              mean_j_ok = NaN; mean_err_ok = NaN; mean_gm_db_ok = NaN; mean_pm_deg_ok = NaN;
      mean_dm_ok = NaN; mean_bw_cl_ok = NaN;
5263
5264              if any(mask_valid)
5265                  jv = sw.J_opt(mask_valid);
5266                  [~, idx_rel] = min(jv);
5267                  idx_all = find(mask_valid);
5268                  idx_best = idx_all(idx_rel);
5269                  j_best = sw.J_opt(idx_best);
5270                  mst_best = sw.Mst_real(idx_best);
5271                  err_best = sw.Mst_err(idx_best);
5272                  iaedo_best = sw.IAEdo_opt(idx_best);
5273                  iaedi_best = sw.IAEdi_opt(idx_best);
5274                  gm_db_vec = getfield_safe(sw, 'GM_dB_opt', NaN(size(sw.J_opt)));
5275                  pm_deg_vec = getfield_safe(sw, 'PM_deg_opt', NaN(size(sw.J_opt)));
5276                  dm_best = getfield_safe(sw, 'DM_opt', NaN(size(sw.J_opt)));
5277                  bw_cl_vec = getfield_safe(sw, 'BW_cl_opt', NaN(size(sw.J_opt)));
5278                  if numel(gm_db_vec) >= idx_best, gm_db_best = gm_db_vec(idx_best); end
5279                  if numel(pm_deg_vec) >= idx_best, pm_deg_best = pm_deg_vec(idx_best); end
5280                  if numel(dm_best) >= idx_best
5281                      dm_best = dm_best(idx_best);
5282                  else
5283                      dm_best = NaN;
5284                  end
5285                  if numel(bw_cl_vec) >= idx_best, bw_cl_best = bw_cl_vec(idx_best); end
5286              end
5287
5288              if any(mask_ok)
5289                  mean_j_ok = mean(sw.J_opt(mask_ok), 'omitnan');
5290                  mean_err_ok = mean(sw.Mst_err(mask_ok), 'omitnan');
5291                  gm_db_ok = getfield_safe(sw, 'GM_dB_opt', NaN(size(sw.J_opt)));
5292                  pm_deg_ok = getfield_safe(sw, 'PM_deg_opt', NaN(size(sw.J_opt)));
```

```
5293                 dm_ok = getfield_safe(sw, 'DM_opt', NaN(size(sw.J_opt)));
5294                 bw_cl_ok = getfield_safe(sw, 'BW_cl_opt', NaN(size(sw.J_opt)));
5295                 mean_gm_db_ok = mean(gm_db_ok(mask_ok), 'omitnan');
5296                 mean_pm_deg_ok = mean(pm_deg_ok(mask_ok), 'omitnan');
5297                 mean_dm_ok = mean(dm_ok(mask_ok), 'omitnan');
5298                 mean_bw_cl_ok = mean(bw_cl_ok(mask_ok), 'omitnan');
5299             end
5300 
5301             rows(end+1,:) = { ...
5302                 Plantas(ip).id, Plantas(ip).tipo, ctrl_key_to_user(ctrl_key), ...
5303                 sum(mask_ok), sum(mask_far), sum(mask_no_cand), ...
5304                 j_best, mst_best, err_best, iaedo_best, iaedi_best, ...
5305                 gm_db_best, pm_deg_best, dm_best, bw_cl_best, ...
5306                 mean_j_ok, mean_err_ok, mean_gm_db_ok, mean_pm_deg_ok, mean_dm_ok,
    mean_bw_cl_ok}; %#ok<AGROW>
5307         end
5308     end
5309 
5310     T = cell2table(rows, 'VariableNames', vars);
5311 end
5312 
5313 function T = build_sweep_winner_table(T_summary)
5314     vars = {'planta','tipo','best_ctrl_j','best_j','best_ctrl_tracking','best_tracking_err',
    ...
5315         'best_ctrl_iaedo','best_iaedo','best_ctrl_iaedi','best_iaedi', ...
5316 
    'best_ctrl_gm','best_gm_db','best_ctrl_pm','best_pm_deg','best_ctrl_dm','best_dm','best_ctrl_b
    w','best_bw_cl'};
5317     rows = cell(0, numel(vars));
5318 
5319     if isempty(T_summary)
5320         T = cell2table(rows, 'VariableNames', vars);
5321         return;
5322     end
5323 
5324     plantas = unique(T_summary.planta, 'stable');
5325     for i = 1:numel(plantas)
5326         mask_p = strcmp(T_summary.planta, plantas{i});
5327         Tp = T_summary(mask_p,:);
5328         tipo = Tp.tipo{1};
5329 
5330         [best_ctrl_j, best_j] = pick_best_row(Tp.ctrl, Tp.j_best);
5331         [best_ctrl_tracking, best_tracking_err] = pick_best_row(Tp.ctrl, Tp.mst_err_best);
5332         [best_ctrl_iaedo, best_iaedo] = pick_best_row(Tp.ctrl, Tp.iaedo_best);
5333         [best_ctrl_iaedi, best_iaedi] = pick_best_row(Tp.ctrl, Tp.iaedi_best);
5334         [best_ctrl_gm, best_gm_db] = pick_max_row(Tp.ctrl, Tp.gm_db_best);
5335         [best_ctrl_pm, best_pm_deg] = pick_max_row(Tp.ctrl, Tp.pm_deg_best);
5336         [best_ctrl_dm, best_dm] = pick_max_row(Tp.ctrl, Tp.dm_best);
5337         [best_ctrl_bw, best_bw_cl] = pick_max_row(Tp.ctrl, Tp.bw_cl_best);
5338 
5339         rows(end+1,:) = { ...
5340             plantas{i}, tipo, ...
5341             best_ctrl_j, best_j, ...
5342             best_ctrl_tracking, best_tracking_err, ...
5343             best_ctrl_iaedo, best_iaedo, ...
5344             best_ctrl_iaedi, best_iaedi, ...
5345             best_ctrl_gm, best_gm_db, ...
5346             best_ctrl_pm, best_pm_deg, ...
5347             best_ctrl_dm, best_dm, ...
5348             best_ctrl_bw, best_bw_cl}; %#ok<AGROW>
5349     end
5350 
5351     T = cell2table(rows, 'VariableNames', vars);
5352 end
5353 
```

```
function [best_label, best_value] = pick_best_row(labels, values)
    values = values(:);
    mask = isfinite(values);
    if ~any(mask)
        best_label = 'NA';
        best_value = NaN;
        return;
    end
    idx = find(mask);
    [best_value, idx_rel] = min(values(mask));
    best_label = labels{idx(idx_rel)};
end

function [best_label, best_value] = pick_max_row(labels, values)
    values = values(:);
    mask = ~isnan(values);
    if ~any(mask)
        best_label = 'NA';
        best_value = NaN;
        return;
    end

    idx = find(mask);
    [best_value, idx_rel] = max(values(mask));
    best_label = labels{idx(idx_rel)};
end

function T = build_base_export_table(Plantas, idx_run, baseMode)
    vars = {'planta','tipo','base_ctrl','base_mode','scan_status','ref_status','base_status',
...
        'kc_do0','ti_do0','td_do0','mst_do0','iaedo0_base', ...
        'kc_di0','ti_di0','td_di0','mst_di0','iaedi0_base','nota'};
    rows = cell(0, numel(vars));

    for k = 1:numel(idx_run)
        ip = idx_run(k);
        nota = '';
        if isfield(Plantas(ip).base, 'extraordinary_note')
            nota = Plantas(ip).base.extraordinary_note;
        end

        rows(end+1,:) = { ...
            Plantas(ip).id, ...
            Plantas(ip).tipo, ...
            Plantas(ip).base.ctrl_key, ...
            baseMode, ...
            getfield_safe(Plantas(ip).base, 'bounds_status', 'NA'), ...
            getfield_safe(Plantas(ip).base, 'ref_status', 'NA'), ...
            Plantas(ip).base.status, ...
            Plantas(ip).base.result.do0.kc_do0, ...
            Plantas(ip).base.result.do0.ti_do0, ...
            Plantas(ip).base.result.do0.td_do0, ...
            Plantas(ip).base.result.do0.mst_do0, ...
            Plantas(ip).base.result.do0.iaedo_do0, ...
            Plantas(ip).base.result.di0.kc_di0, ...
            Plantas(ip).base.result.di0.ti_di0, ...
            Plantas(ip).base.result.di0.td_di0, ...
            Plantas(ip).base.result.di0.mst_di0, ...
            Plantas(ip).base.result.di0.iaedi_di0, ...
            nota ...
            }; %#ok<AGROW>
    end

    T = cell2table(rows, 'VariableNames', vars);
end
```

```
5418
5419  function T = build_sweep_export_table(Plantas, idx_run, runPI, runPID, runSP, runMSP)
5420      vars = {'planta','tipo','ctrl','mst_obj','status','ncand','kc_opt','ti_opt','td_opt', ...
5421          'mst_real','mst_err','mst_tol_used','iaedo_opt','iaedi_opt','j_opt', ...
5422          'gm_opt','gm_db_opt','pm_deg_opt','dm_opt','bw_cl_opt', ...
5423
      'kc_paper_ref','ti_paper_ref','mst_paper_ref','j_paper_ref','paper_status_ref','paper_case_ref
      ', ...
5424          'scan_status','bounds_mode'};
5425      rows = cell(0, numel(vars));
5426
5427      for k = 1:numel(idx_run)
5428          ip = idx_run(k);
5429
5430          ctrl_list = {};
5431          if runPI,  ctrl_list(end+1,:) = {'PI',  'PI_sweep'};  end %#ok<AGROW>
5432          if runPID, ctrl_list(end+1,:) = {'PID', 'PID_sweep'}; end %#ok<AGROW>
5433          if runSP,  ctrl_list(end+1,:) = {'SP',  'SP_sweep'};  end %#ok<AGROW>
5434          if runMSP, ctrl_list(end+1,:) = {'MSP', 'MSP_sweep'}; end %#ok<AGROW>
5435
5436          for ic = 1:size(ctrl_list,1)
5437              ctrl_key = ctrl_list{ic,1};
5438              fieldn   = ctrl_list{ic,2};
5439
5440              if ~isfield(Plantas(ip), fieldn)
5441                  continue;
5442              end
5443
5444              sw = Plantas(ip).(fieldn);
5445              gm_vec = getfield_safe(sw, 'GM_opt', NaN(size(sw.Mst_obj)));
5446              gm_db_vec = getfield_safe(sw, 'GM_dB_opt', NaN(size(sw.Mst_obj)));
5447              pm_vec = getfield_safe(sw, 'PM_deg_opt', NaN(size(sw.Mst_obj)));
5448              dm_vec = getfield_safe(sw, 'DM_opt', NaN(size(sw.Mst_obj)));
5449              bw_vec = getfield_safe(sw, 'BW_cl_opt', NaN(size(sw.Mst_obj)));
5450              kc_paper_vec = getfield_safe(sw, 'Kc_paper_ref', NaN(size(sw.Mst_obj)));
5451              ti_paper_vec = getfield_safe(sw, 'Ti_paper_ref', NaN(size(sw.Mst_obj)));
5452              mst_paper_vec = getfield_safe(sw, 'Mst_paper_ref', NaN(size(sw.Mst_obj)));
5453              j_paper_vec = getfield_safe(sw, 'J_paper_ref', NaN(size(sw.Mst_obj)));
5454              paper_status_vec = getfield_safe(sw, 'paper_status_ref', repmat({''},
      size(sw.Mst_obj)));
5455              paper_case_vec = getfield_safe(sw, 'paper_case_ref', repmat({''},
      size(sw.Mst_obj)));
5456              for im = 1:numel(sw.Mst_obj)
5457                  rows(end+1,:) = { ...
5458                      Plantas(ip).id, ...
5459                      Plantas(ip).tipo, ...
5460                      ctrl_key_to_user(ctrl_key), ...
5461                      sw.Mst_obj(im), ...
5462                      sw.status{im}, ...
5463                      sw.nCand(im), ...
5464                      sw.Kc_opt(im), ...
5465                      sw.Ti_opt(im), ...
5466                      sw.Td_opt(im), ...
5467                      sw.Mst_real(im), ...
5468                      sw.Mst_err(im), ...
5469                      sw.Mst_tol_used(im), ...
5470                      sw.IAEdo_opt(im), ...
5471                      sw.IAEdi_opt(im), ...
5472                      sw.J_opt(im), ...
5473                      gm_vec(im), ...
5474                      gm_db_vec(im), ...
5475                      pm_vec(im), ...
5476                      dm_vec(im), ...
5477                      bw_vec(im), ...
5478                      kc_paper_vec(im), ...
```

```
                    ti_paper_vec(im), ...
                    mst_paper_vec(im), ...
                    j_paper_vec(im), ...
                    paper_status_vec{im}, ...
                    paper_case_vec{im}, ...
                    sw.scan_status{im}, ...
                    sw.bounds_mode{im} ...
                    }; %#ok<AGROW>
            end
        end
    end

    T = cell2table(rows, 'VariableNames', vars);
end

function T = build_sweep_freq_export_table(Plantas, idx_run, runPI, runPID, runSP, runMSP)
    vars =
{'planta','tipo','ctrl','mst_obj','status','mst_real','gm_opt','gm_db_opt','pm_deg_opt','dm_op
t','bw_cl_opt','wcg_opt','wcp_opt'};
    rows = cell(0, numel(vars));

    for k = 1:numel(idx_run)
        ip = idx_run(k);

        ctrl_list = {};
        if runPI,  ctrl_list(end+1,:) = {'PI',  'PI_sweep'};  end %#ok<AGROW>
        if runPID, ctrl_list(end+1,:) = {'PID', 'PID_sweep'}; end %#ok<AGROW>
        if runSP,  ctrl_list(end+1,:) = {'SP',  'SP_sweep'};  end %#ok<AGROW>
        if runMSP, ctrl_list(end+1,:) = {'MSP', 'MSP_sweep'}; end %#ok<AGROW>

        for ic = 1:size(ctrl_list,1)
            ctrl_key = ctrl_list{ic,1};
            fieldn   = ctrl_list{ic,2};
            if ~isfield(Plantas(ip), fieldn)
                continue;
            end

            sw = Plantas(ip).(fieldn);
            gm_vec = getfield_safe(sw, 'GM_opt', NaN(size(sw.Mst_obj)));
            gm_db_vec = getfield_safe(sw, 'GM_dB_opt', NaN(size(sw.Mst_obj)));
            pm_vec = getfield_safe(sw, 'PM_deg_opt', NaN(size(sw.Mst_obj)));
            dm_vec = getfield_safe(sw, 'DM_opt', NaN(size(sw.Mst_obj)));
            bw_vec = getfield_safe(sw, 'BW_cl_opt', NaN(size(sw.Mst_obj)));
            wcg_vec = getfield_safe(sw, 'Wcg_opt', NaN(size(sw.Mst_obj)));
            wcp_vec = getfield_safe(sw, 'Wcp_opt', NaN(size(sw.Mst_obj)));

            for im = 1:numel(sw.Mst_obj)
                rows(end+1,:) = { ...
                    Plantas(ip).id, ...
                    Plantas(ip).tipo, ...
                    ctrl_key_to_user(ctrl_key), ...
                    sw.Mst_obj(im), ...
                    sw.status{im}, ...
                    sw.Mst_real(im), ...
                    gm_vec(im), ...
                    gm_db_vec(im), ...
                    pm_vec(im), ...
                    dm_vec(im), ...
                    bw_vec(im), ...
                    wcg_vec(im), ...
                    wcp_vec(im) ...
                    }; %#ok<AGROW>
            end
        end
    end
```

```
5542
5543      T = cell2table(rows, 'VariableNames', vars);
5544  end
5545
5546  function T = build_extraordinary_export_table(Plantas, idx_run, runPI, runPID, runSP, runMSP)
5547      vars = {'planta','tipo','etapa','ctrl','codigo','detalle'};
5548      rows = cell(0, numel(vars));
5549
5550      for k = 1:numel(idx_run)
5551          ip = idx_run(k);
5552
5553          if isfield(Plantas(ip), 'base') && isfield(Plantas(ip).base, 'extraordinary_note')
5554              rows(end+1,:) = { ...
5555                  Plantas(ip).id, ...
5556                  Plantas(ip).tipo, ...
5557                  'BASE', ...
5558                  ctrl_key_to_user(Plantas(ip).base.ctrl_key), ...
5559                  Plantas(ip).base.status, ...
5560                  Plantas(ip).base.extraordinary_note ...
5561                  }; %#ok<AGROW>
5562          end
5563
5564          ctrl_list = {};
5565          if runPI,  ctrl_list(end+1,:) = {'PI',  'PI_sweep'};  end %#ok<AGROW>
5566          if runPID, ctrl_list(end+1,:) = {'PID', 'PID_sweep'}; end %#ok<AGROW>
5567          if runSP,  ctrl_list(end+1,:) = {'SP',  'SP_sweep'};  end %#ok<AGROW>
5568          if runMSP, ctrl_list(end+1,:) = {'MSP', 'MSP_sweep'}; end %#ok<AGROW>
5569
5570          for ic = 1:size(ctrl_list,1)
5571              ctrl_key = ctrl_list{ic,1};
5572              fieldn   = ctrl_list{ic,2};
5573
5574              if ~isfield(Plantas(ip), fieldn)
5575                  continue;
5576              end
5577
5578              sw = Plantas(ip).(fieldn);
5579              for im = 1:numel(sw.Mst_obj)
5580                  if ~strcmp(sw.status{im}, 'OK')
5581                      dm_vec = getfield_safe(sw, 'DM_opt', NaN(size(sw.Mst_obj)));
5582                      detalle = sprintf('Mst_obj=%.4f | nCand=%d | scan=%s | mode=%s |
      Mst_real=%s | J=%s | DM=%s', ...
5583                          sw.Mst_obj(im), sw.nCand(im), sw.scan_status{im}, sw.bounds_mode{im},
      ...
5584                          fmt_num(sw.Mst_real(im)), fmt_num(sw.J_opt(im)), fmt_num(dm_vec(im)));
5585
5586                      rows(end+1,:) = { ...
5587                          Plantas(ip).id, ...
5588                          Plantas(ip).tipo, ...
5589                          'SWEEP', ...
5590                          ctrl_key_to_user(ctrl_key), ...
5591                          sw.status{im}, ...
5592                          detalle ...
5593                          }; %#ok<AGROW>
5594                  end
5595              end
5596          end
5597      end
5598
5599      T = cell2table(rows, 'VariableNames', vars);
5600  end
5601
5602  function lines = build_metadata_lines( ...
5603      code_version, baseCtrl, baseMode, ctrls_sel_txt, plants_sel_txt, ...
5604      Mst_base, mst_min, mst_max, mst_step, mst_grid, ...
```

```
5605      scan_paso_global, ...
5606      use_exact_mst, save_eval_detail, show_iter_status, export_pub_figs, export_results_csv,
      export_results_xlsx, save_run_workspace, print_export_tables, csv_delimiter, thesis_mode, ...
5607      pub_fig_folder, csv_out_folder, xlsx_book_file, workspace_mat_file)
5608
5609      lines = {
5610          'SCRIPT J vs Mst - Metadata de corrida'
5611          ['Fecha_hora = ' datestr(now, 31)]
5612          ['code_version = ' code_version]
5613          ['baseCtrl = ' baseCtrl]
5614          ['baseMode = ' baseMode]
5615          ['controladores = ' ctrls_sel_txt]
5616          ['plantas = ' plants_sel_txt]
5617          ['Mst_base = ' num2str(Mst_base, '%.6f')]
5618          ['mst_min = ' num2str(mst_min, '%.6f')]
5619          ['mst_max = ' num2str(mst_max, '%.6f')]
5620          ['mst_step = ' num2str(mst_step, '%.6f')]
5621          ['mst_grid = ' vec2str(mst_grid, '%.4f')]
5622          ['scan_paso_global = ' num2str(scan_paso_global, '%.6f')]
5623          'tol_mst_final = h_used de la ultima iteracion valida'
5624          ['use_exact_mst = ' ternary_txt(use_exact_mst, 'YES', 'NO')]
5625          ['save_eval_detail = ' ternary_txt(save_eval_detail, 'YES', 'NO')]
5626          ['show_iter_status = ' ternary_txt(show_iter_status, 'YES', 'NO')]
5627          ['thesis_mode = ' ternary_txt(thesis_mode, 'YES', 'NO')]
5628          ['export_pub_figs = ' ternary_txt(export_pub_figs, 'YES', 'NO')]
5629          ['export_results_csv = ' ternary_txt(export_results_csv, 'YES', 'NO')]
5630          ['export_results_xlsx = ' ternary_txt(export_results_xlsx, 'YES', 'NO')]
5631          ['save_run_workspace = ' ternary_txt(save_run_workspace, 'YES', 'NO')]
5632          ['print_export_tables = ' ternary_txt(print_export_tables, 'YES', 'NO')]
5633          ['csv_delimiter = ' csv_delimiter]
5634          ['fig_folder = ' pub_fig_folder]
5635          ['csv_folder = ' csv_out_folder]
5636          ['xlsx_workbook = ' xlsx_book_file]
5637          ['workspace_mat = ' workspace_mat_file]
5638          };
5639  end
5640
5641  function write_text_lines(filename, lines)
5642      fid = fopen(filename, 'w');
5643      if fid == -1
5644          error('No se pudo abrir el archivo para escritura: %s', filename);
5645      end
5646
5647      cleaner = onCleanup(@() fclose(fid)); %#ok<NASGU>
5648      for i = 1:numel(lines)
5649          fprintf(fid, '%s\n', lines{i});
5650      end
5651  end
5652
5653  function vals = parse_lista_numerica(txt, valid_vals)
5654      txt = strtrim(txt);
5655      if isempty(txt)
5656          vals = [];
5657          return;
5658      end
5659
5660      txt = regexprep(txt, '[;|/]', ',');
5661      txt = regexprep(txt, '\s+', ',');
5662      txt = regexprep(txt, ',+', ',');
5663      txt = regexprep(txt, '^,|,$', '');
5664
5665      partes = strsplit(txt, ',');
5666      nums = str2double(partes);
5667
5668      if any(isnan(nums)) || any(~isfinite(nums)) || any(abs(nums - round(nums)) > 1e-12)
```

```
5669          error('La seleccion debe ingresarse solo con numeros enteros validos.');
5670      end
5671
5672      vals = unique(nums, 'stable');
5673      if any(~ismember(vals, valid_vals))
5674          error('Se ingreso una opcion numerica fuera del rango permitido.');
5675      end
5676  end
5677
5678  function ensure_parallel_pool_quiet(prestart_parallel_pool, parallel_profile)
5679      if ~prestart_parallel_pool
5680          return;
5681      end
5682
5683      try
5684          p = gcp('nocreate');
5685          if isempty(p)
5686              if nargin < 2 || isempty(parallel_profile)
5687                  evalc('parpool');
5688              else
5689                  cmd = sprintf('parpool(''%s'')', parallel_profile);
5690                  evalc(cmd);
5691              end
5692          end
5693      catch
5694          % Si no hay Parallel Computing Toolbox o falla el arranque, el script
5695          % continua y MATLAB decidira si ejecuta en serial o intenta abrir el pool.
5696      end
5697  end
5698
5699  function print_block_title(txt)
5700      fprintf('\n%s\n', repmat('=',1,84));
5701      fprintf('%s\n', txt);
5702      fprintf('%s\n', repmat('=',1,84));
5703  end
5704
5705  function print_subtitle(txt)
5706      fprintf('%s\n', repmat('-',1,84));
5707      fprintf('%s\n', txt);
5708      fprintf('%s\n', repmat('-',1,84));
5709  end
5710
5711  function print_notes_vertical(lines)
5712      fprintf('Notas:\n');
5713      for i = 1:numel(lines)
5714          fprintf('  * %s\n', lines{i});
5715      end
5716  end
5717
5718  function print_paper_tuning_comparison(Plantas, idx_run, runMSP, Go, Gi, t, pade_order,
      N_der, Tf_min, use_exact_mst, w_mst)
5719      fprintf('1. CRITERIO DE COMPARACION\n');
5720      fprintf('%s\n', repmat('-',1,118));
5721      fprintf('  Se evalua la sintonia MSP del paper con las mismas metricas del script: Mst,
      IAEdo, IAEdi y J.\n');
5722      fprintf('  Luego se compara contra la mejor solucion MSP hallada por nuestro barrido para
      cada planta.\n');
5723
5724      if ~runMSP
5725          fprintf('\nMSP no fue seleccionado en la corrida; no hay comparacion contra el
      barrido.\n');
5726          return;
5727      end
5728
5729      fprintf('\n2. COMPARACION PAPER vs BARRIDO MSP\n');
```

```
5730      fprintf('%s\n', repmat('-',1,154));
5731      fmt = '  %-6s | %-9s | %-9s | %-9s | %-9s | %-9s | %-9s | %-9s | %-9s | %-9s\n';
5732      fprintf(fmt, 'Planta', 'Kc_paper', 'Ti_paper', 'Mst_paper', 'J_paper', 'Kc_our',
      'Ti_our', 'Mst_our', 'J_our', 'DeltaJ', 'DeltaMst');
5733      fprintf('  %s\n', repmat('-',1,154));
5734
5735      for k = 1:numel(idx_run)
5736          ip = idx_run(k);
5737          plant = Plantas(ip);
5738
5739          if ~isfield(plant, 'MSP_sweep')
5740              fprintf(fmt, plant.id, 'NA', 'NA', 'NA', 'NA', 'NA', 'NA', 'NA', 'NA', 'NA',
      'NA');
5741              continue;
5742          end
5743
5744          paper_res = evaluate_paper_msp_benchmark( ...
5745              plant, Plantas(ip).base.result, Go, Gi, t, pade_order, N_der, Tf_min,
      use_exact_mst, w_mst);
5746          our_res = extract_best_sweep_point(plant.MSP_sweep);
5747
5748          delta_j = paper_res.J_opt - our_res.J_opt;
5749          delta_mst = paper_res.mst_opt - our_res.mst_opt;
5750
5751          fprintf(fmt, ...
5752              plant.id, ...
5753              fmt_num(paper_res.kc_opt), ...
5754              fmt_num(paper_res.ti_opt), ...
5755              fmt_num(paper_res.mst_opt), ...
5756              fmt_num(paper_res.J_opt), ...
5757              fmt_num(our_res.kc_opt), ...
5758              fmt_num(our_res.ti_opt), ...
5759              fmt_num(our_res.mst_opt), ...
5760              fmt_num(our_res.J_opt), ...
5761              fmt_num(delta_j), ...
5762              fmt_num(delta_mst));
5763
5764          fprintf('    Paper: status=%s | caso=%s | IAEdo=%s | IAEdi=%s\n', ...
5765              paper_res.status, paper_res.paper_case, fmt_num(paper_res.iaedo_opt),
      fmt_num(paper_res.iaedi_opt));
5766          fprintf('    Nuestro barrido: status=%s | Mst_obj=%s | IAEdo=%s | IAEdi=%s\n', ...
5767              our_res.status, fmt_num(our_res.mst_obj), fmt_num(our_res.iaedo_opt),
      fmt_num(our_res.iaedi_opt));
5768      end
5769  end
5770
5771  function res = evaluate_paper_msp_benchmark(plant, base_result, Go, Gi, t, pade_order, N_der,
      Tf_min, use_exact_mst, w_mst)
5772      P = pade(plant.P_dt, pade_order);
5773      theta0 = plant.P_dt.InputDelay;
5774      [num0, den0] = tfdata(plant.P_dt, 'v');
5775      P0 = tf(num0, den0);
5776
5777      [kc_paper, ti_paper, paper_case] = get_msp_paper_tuning(plant.tipo, plant.id, theta0);
5778      td_paper = 0;
5779
5780      res = struct( ...
5781          'paper_case', paper_case, ...
5782          'status', 'UNSTABLE', ...
5783          'kc_opt', kc_paper, ...
5784          'ti_opt', ti_paper, ...
5785          'td_opt', td_paper, ...
5786          'mst_opt', NaN, ...
5787          'iaedo_opt', NaN, ...
5788          'iaedi_opt', NaN, ...
```

```
5789          'J_opt', NaN);
5790
5791      [ok, mst_val] = eval_mst_controller('MSP', kc_paper, ti_paper, td_paper, P, P0, theta0,
      ...
5792          plant.tipo, plant.id, Go, Gi, pade_order, N_der, Tf_min, use_exact_mst, w_mst);
5793      if ~ok
5794          return;
5795      end
5796
5797      [iaedo, iaedi] = eval_iae_controller('MSP', kc_paper, ti_paper, td_paper, P, P0, theta0,
      ...
5798          plant.tipo, plant.id, Go, Gi, t, pade_order, N_der, Tf_min);
5799      if ~(isfinite(iaedo) && isfinite(iaedi))
5800          res.status = 'NO_IAE';
5801          res.mst_opt = mst_val;
5802          return;
5803      end
5804
5805      iaedo0_base = max(base_result.do0.iaedo_do0, 1e-12);
5806      iaedi0_base = max(base_result.di0.iaedi_di0, 1e-12);
5807
5808      res.status = 'OK';
5809      res.mst_opt = mst_val;
5810      res.iaedo_opt = iaedo;
5811      res.iaedi_opt = iaedi;
5812      res.J_opt = 0.5 * ((iaedo / iaedo0_base) + (iaedi / iaedi0_base));
5813  end
5814
5815  function [kc_paper, ti_paper, paper_case] = get_msp_paper_tuning(plant_tipo, plant_id, theta0)
5816      [~, ~, To, Tplant] = build_Fp_msp(plant_tipo, plant_id, theta0);
5817      Kp = 1;
5818
5819      switch lower(plant_tipo)
5820          case 'estable'
5821              kc_paper = Tplant / (To * Kp);
5822              ti_paper = Tplant;
5823              paper_case = 'FOPDT';
5824
5825          case 'integrador'
5826              kc_paper = 1 / (To * Kp);
5827              ti_paper = 0;
5828              paper_case = 'IPDT';
5829
5830          case 'inestable'
5831              kc_paper = (To + 2 * Tplant) / (To * Kp);
5832              ti_paper = To * (2 + To / Tplant);
5833              paper_case = 'UFOPDT';
5834
5835          otherwise
5836              kc_paper = NaN;
5837              ti_paper = NaN;
5838              paper_case = 'NA';
5839      end
5840  end
5841
5842  function res = extract_best_sweep_point(sweep)
5843      res = struct('status','NO_DATA','mst_obj',NaN,'kc_opt',NaN,'ti_opt',NaN,'td_opt',NaN, ...
5844          'mst_opt',NaN,'iaedo_opt',NaN,'iaedi_opt',NaN,'J_opt',NaN);
5845
5846      if isempty(sweep) || ~isfield(sweep, 'J_opt')
5847          return;
5848      end
5849
5850      status = string(sweep.status(:));
5851      mask_valid = (status == "OK" | status == "FAR_MST") & isfinite(sweep.J_opt(:));
```

```
    if ~any(mask_valid)
        return;
    end

    idx_all = find(mask_valid);
    [~, idx_rel] = min(sweep.J_opt(mask_valid));
    idx = idx_all(idx_rel);

    res.status = char(status(idx));
    res.mst_obj = sweep.Mst_obj(idx);
    res.kc_opt = sweep.Kc_opt(idx);
    res.ti_opt = sweep.Ti_opt(idx);
    res.td_opt = sweep.Td_opt(idx);
    res.mst_opt = sweep.Mst_real(idx);
    res.iaedo_opt = sweep.IAEdo_opt(idx);
    res.iaedi_opt = sweep.IAEdi_opt(idx);
    res.J_opt = sweep.J_opt(idx);
end

function status_msg(tag, txt, level)
    global verbose_status_global show_iter_logs_global compact_console_global

    if nargin < 3
        level = 'detail';
    end

    if verbose_status_global
        show_msg = true;
    elseif show_iter_logs_global
        show_msg = ~strcmpi(tag,'RUN');
    elseif compact_console_global
        show_msg = strcmpi(level, 'important');
    else
        show_msg = strcmpi(level, 'important') || strcmpi(level, 'summary');
        if strcmpi(tag,'RUN')
            show_msg = false;
        end
    end

    if show_msg
        fprintf('[%-4s] %s\n', upper(tag), txt);
    end
end

function out = fmt_num(x)
    if isempty(x) || ~isscalar(x) || ~isfinite(x)
        out = 'NA';
    else
        out = sprintf('%.4f', x);
    end
end

function out = fmt_rng(v)
    if isempty(v) || numel(v) < 2 || any(~isfinite(v))
        out = '[- -]';
    else
        out = sprintf('[%.4f %.4f]', v(1), v(2));
    end
end

function out = vec2str(v, fmt)
    c = arrayfun(@(x) sprintf(fmt, x), v, 'UniformOutput', false);
    out = ['[' strjoin(c, ' ') ']'];
end
```

```
function out = clip_str(s, n)
    s = string(s);
    s = char(s);
    if numel(s) > n
        out = s(1:n);
    else
        out = s;
    end
end

function out = ternary_txt(cond, a, b)
    if cond
        out = a;
    else
        out = b;
    end
end

function out = ctrl_key_to_user(ctrl_key)
    switch upper(ctrl_key)
        case 'PI'
            out = 'PI';
        case 'PID'
            out = 'PID';
        case 'SP'
            out = 'SP';
        case 'MSP'
            out = 'MSP';
        otherwise
            out = ctrl_key;
    end
end

function x = min_or_nan(v)
    if isempty(v)
        x = NaN;
    else
        x = min(v);
    end
end

function x = max_or_nan(v)
    if isempty(v)
        x = NaN;
    else
        x = max(v);
    end
end

function val = getfield_safe(S, fieldname, default_value)
    if isstruct(S) && isfield(S, fieldname)
        val = S.(fieldname);
    else
        val = default_value;
    end
end

function [runPI, runPID, runSP, runMSP] = infer_run_flags_from_loaded_sweeps_td(Plantas,
idx_run)
    runPI = false; runPID = false; runSP = false; runMSP = false;
    for ip = idx_run
        runPI  = runPI  || isfield(Plantas(ip), 'PI_sweep');
        runPID = runPID || isfield(Plantas(ip), 'PID_sweep');
        runSP  = runSP  || isfield(Plantas(ip), 'SP_sweep');
        runMSP = runMSP || isfield(Plantas(ip), 'MSP_sweep');
```

```
5981      end
5982  end
5983
5984  function mst_opts = collect_available_mst_options_td(Plantas, idx_run, runPI, runPID, runSP,
      runMSP)
5985      mst_opts = [];
5986      ctrl_map = {
5987          runPI,  'PI_sweep'
5988          runPID, 'PID_sweep'
5989          runSP,  'SP_sweep'
5990          runMSP, 'MSP_sweep'
5991          };
5992
5993      for k = 1:numel(idx_run)
5994          ip = idx_run(k);
5995          for ic = 1:size(ctrl_map,1)
5996              runFlag = ctrl_map{ic,1};
5997              fieldn = ctrl_map{ic,2};
5998              if ~runFlag || ~isfield(Plantas(ip), fieldn)
5999                  continue;
6000              end
6001              sw = Plantas(ip).(fieldn);
6002              if isstruct(sw) && isfield(sw, 'Mst_obj')
6003                  vals = sw.Mst_obj(:);
6004                  vals = vals(isfinite(vals));
6005                  mst_opts = [mst_opts; vals]; %#ok<AGROW>
6006              end
6007          end
6008      end
6009      mst_opts = unique(round(mst_opts, 10), 'stable');
6010  end
6011
6012  function [mst_ref_eval, mst_ref_default] = resolve_mst_eval_choice_td(mst_ref_eval_in,
      mst_eval_options, ask_mst_ref_eval)
6013      [~, idx_default] = min(abs(mst_eval_options - mst_ref_eval_in));
6014      mst_ref_default = mst_eval_options(idx_default);
6015      mst_ref_eval = mst_ref_default;
6016
6017      if ~ask_mst_ref_eval
6018          return;
6019      end
6020
6021      fprintf('\n=== SELECCION DE Mst PARA EVALUACION TEMPORAL ===\n');
6022      fprintf('Valores disponibles: %s\n', vec2str(mst_eval_options, '%.4f'));
6023      fprintf('Escribe ENTER para usar el valor default, o escribe 0 / salir / exit para
      cancelar.\n');
6024
6025      while true
6026          mst_in = input(sprintf('Ingresa Mst a evaluar (enter = %.4f): ', mst_ref_default),
      's');
6027          mst_in = strtrim(mst_in);
6028
6029          if isempty(mst_in)
6030              return;
6031          end
6032
6033          if any(strcmpi(mst_in, {'0','salir','exit','quit','cancelar','cancel'}))
6034              error('CodigoTesis:CanceladoPorUsuario', 'Ejecucion cancelada por el usuario en
      la seleccion de Mst.');
6035          end
6036
6037          mst_num = str2double(mst_in);
6038          if ~isscalar(mst_num) || ~isfinite(mst_num)
6039              fprintf('Entrada invalida. Ingresa uno de los valores disponibles, ENTER para el
      default o 0/salir para cancelar.\n');
```

```
6040            continue;
6041        end
6042
6043        idx_match = find(abs(mst_eval_options - mst_num) <= 1e-9, 1, 'first');
6044        if isempty(idx_match)
6045            fprintf('El Mst ingresado no pertenece a los valores barridos en los sweeps
    cargados. Intenta de nuevo o escribe 0 para cancelar.\n');
6046            continue;
6047        end
6048
6049        mst_ref_eval = mst_eval_options(idx_match);
6050        return;
6051    end
6052 end
6053
6054 function [Plantas_eval, T] = build_fixed_pick_table_td(Plantas, idx_run, runPI, runPID,
    runSP, runMSP, mst_ref_eval, allow_far_mst_eval)
6055     Plantas_eval = Plantas;
6056     vars =
    {'planta','tipo','ctrl','pick_status','mst_ref','mst_obj_src','mst_real','mst_err', ...
6057         'kc','ti','td','j_opt','iaedo_opt','iaedi_opt','source_idx'};
6058     rows = cell(0, numel(vars));
6059
6060     ctrl_map = {
6061         'PI',  runPI,  'PI_sweep',  'PI'
6062         'PID', runPID, 'PID_sweep', 'PID'
6063         'SP',  runSP,  'SP_sweep',  'SP'
6064         'MSP', runMSP, 'MSP_sweep', 'MSP'
6065         };
6066
6067     for k = 1:numel(idx_run)
6068         ip = idx_run(k);
6069         Plantas_eval(ip).fixed_eval = struct();
6070         for ic = 1:size(ctrl_map,1)
6071             ctrl_key = ctrl_map{ic,1};
6072             runFlag = ctrl_map{ic,2};
6073             sweep_field = ctrl_map{ic,3};
6074             target_field = ctrl_map{ic,4};
6075
6076             if ~runFlag || ~isfield(Plantas(ip), sweep_field)
6077                 rows(end+1,:) = {Plantas(ip).id, Plantas(ip).tipo,
    ctrl_key_to_user(ctrl_key), ...
6078                     'NO_SWEEP', mst_ref_eval, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN,
    NaN}; %#ok<AGROW>
6079                 continue;
6080             end
6081
6082             [pick, pick_status] = select_fixed_from_sweep_td(Plantas(ip).(sweep_field),
    mst_ref_eval, allow_far_mst_eval);
6083             Plantas_eval(ip).fixed_eval.(target_field) = pick;
6084
6085             if strcmp(pick_status, 'OK')
6086                 Plantas_eval(ip).(target_field).opt = pick.opt;
6087                 rows(end+1,:) = {Plantas(ip).id, Plantas(ip).tipo,
    ctrl_key_to_user(ctrl_key), ...
6088                     'OK', mst_ref_eval, pick.mst_obj_src, pick.opt.mst_opt, pick.opt.mst_err,
    ...
6089                     pick.opt.kc_opt, pick.opt.ti_opt, pick.opt.td_opt, pick.opt.J_opt, ...
6090                     pick.opt.iaedo_opt, pick.opt.iaedi_opt, pick.source_idx}; %#ok<AGROW>
6091             else
6092                 rows(end+1,:) = {Plantas(ip).id, Plantas(ip).tipo,
    ctrl_key_to_user(ctrl_key), ...
6093                     pick_status, mst_ref_eval, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN,
    NaN}; %#ok<AGROW>
6094             end
```

```
6095          end
6096      end
6097
6098      T = cell2table(rows, 'VariableNames', vars);
6099  end
6100
6101  function [pick, status] = select_fixed_from_sweep_td(sweep, mst_ref_eval, allow_far_mst_eval)
6102      pick = struct();
6103      status = 'NO_VALID';
6104
6105      if isempty(sweep) || ~isstruct(sweep) || ~isfield(sweep, 'status') || ~isfield(sweep,
      'Mst_obj')
6106          status = 'NO_SWEEP';
6107          return;
6108      end
6109
6110      status_str = string(sweep.status(:));
6111      mask = status_str == "OK";
6112      if ~any(mask) && allow_far_mst_eval
6113          mask = status_str == "OK" | status_str == "FAR_MST";
6114      end
6115
6116      core_mask = mask ...
6117          & isfinite(sweep.Kc_opt(:)) ...
6118          & isfinite(sweep.Ti_opt(:)) ...
6119          & isfinite(sweep.Td_opt(:)) ...
6120          & isfinite(sweep.Mst_real(:)) ...
6121          & isfinite(sweep.Mst_err(:)) ...
6122          & isfinite(sweep.J_opt(:));
6123
6124      if ~any(core_mask)
6125          status = 'NO_OK_AT_REF';
6126          return;
6127      end
6128
6129      err = abs(sweep.Mst_real(core_mask) - mst_ref_eval);
6130      min_err = min(err);
6131      idx_local = find(core_mask);
6132      idx_candidates = idx_local(err <= (min_err + 1e-12));
6133      [~, rel_best] = min(sweep.J_opt(idx_candidates));
6134      idx = idx_candidates(rel_best);
6135
6136      pick.ctrl_key = getfield_safe(sweep, 'ctrl_key', '');
6137      pick.mst_ref = mst_ref_eval;
6138      pick.mst_obj_src = sweep.Mst_obj(idx);
6139      pick.source_idx = idx;
6140      pick.opt = struct( ...
6141          'kc_opt', sweep.Kc_opt(idx), ...
6142          'ti_opt', sweep.Ti_opt(idx), ...
6143          'td_opt', sweep.Td_opt(idx), ...
6144          'mst_opt', sweep.Mst_real(idx), ...
6145          'mst_err', sweep.Mst_err(idx), ...
6146          'mst_tol_used', getfield_safe(sweep, 'Mst_tol_used', NaN(size(sweep.Mst_obj))), ...
6147          'iaedo_opt', sweep.IAEdo_opt(idx), ...
6148          'iaedi_opt', sweep.IAEdi_opt(idx), ...
6149          'J_opt', sweep.J_opt(idx));
6150      if numel(pick.opt.mst_tol_used) > 1
6151          pick.opt.mst_tol_used = pick.opt.mst_tol_used(idx);
6152      end
6153      status = 'OK';
6154  end
6155
6156  function Plantas_out = refresh_loaded_sweeps_j_metrics(Plantas_in, idx_run, runPI, runPID,
      runSP, runMSP, Go, Gi, t, pade_order, N_der, Tf_min)
6157      Plantas_out = Plantas_in;
```

```
6158
6159      ctrl_map = {
6160          'PI',  runPI,  'PI_sweep'
6161          'PID', runPID, 'PID_sweep'
6162          'SP',  runSP,  'SP_sweep'
6163          'MSP', runMSP, 'MSP_sweep'
6164          };
6165
6166      for k = 1:numel(idx_run)
6167          ip = idx_run(k);
6168          theta0 = Plantas_out(ip).P_dt.InputDelay;
6169          [num0, den0] = tfdata(Plantas_out(ip).P_dt, 'v');
6170          P = pade(Plantas_out(ip).P_dt, pade_order);
6171          P0 = tf(num0, den0);
6172          [iae_do0_base, iae_di0_base, has_base] = get_base_weights_td(Plantas_out(ip));
6173
6174          for ic = 1:size(ctrl_map,1)
6175              ctrl_key = ctrl_map{ic,1};
6176              runFlag = ctrl_map{ic,2};
6177              sweep_field = ctrl_map{ic,3};
6178
6179              if ~runFlag || ~isfield(Plantas_out(ip), sweep_field)
6180                  continue;
6181              end
6182
6183              sw = Plantas_out(ip).(sweep_field);
6184              if isempty(sw) || ~isstruct(sw) || ~isfield(sw, 'Kc_opt') || ~isfield(sw,
      'Ti_opt') || ~isfield(sw, 'Td_opt')
6185                  continue;
6186              end
6187
6188              nM = numel(sw.Mst_obj);
6189              if ~isfield(sw, 'IAEdo_opt') || numel(sw.IAEdo_opt) ~= nM
6190                  sw.IAEdo_opt = NaN(nM,1);
6191              end
6192              if ~isfield(sw, 'IAEdi_opt') || numel(sw.IAEdi_opt) ~= nM
6193                  sw.IAEdi_opt = NaN(nM,1);
6194              end
6195              if ~isfield(sw, 'J_opt') || numel(sw.J_opt) ~= nM
6196                  sw.J_opt = NaN(nM,1);
6197              end
6198
6199              mask_eval = isfinite(sw.Kc_opt(:)) & isfinite(sw.Ti_opt(:)) &
      isfinite(sw.Td_opt(:));
6200
6201              for im = find(mask_eval).'
6202                  has_iaedo = isfinite(sw.IAEdo_opt(im));
6203                  has_iaedi = isfinite(sw.IAEdi_opt(im));
6204                  has_j     = isfinite(sw.J_opt(im));
6205
6206                  if has_iaedo && has_iaedi && has_j
6207                      continue;
6208                  end
6209
6210                  [iaedo, iaedi] = eval_iae_controller( ...
6211                      ctrl_key, sw.Kc_opt(im), sw.Ti_opt(im), sw.Td_opt(im), ...
6212                      P, P0, theta0, Plantas_out(ip).tipo, Plantas_out(ip).id, Go, Gi, t,
      pade_order, N_der, Tf_min);
6213
6214                  if ~has_iaedo
6215                      sw.IAEdo_opt(im) = iaedo;
6216                  end
6217                  if ~has_iaedi
6218                      sw.IAEdi_opt(im) = iaedi;
6219                  end
```

```
6220
6221                 if ~has_j && has_base && isfinite(iaedo) && isfinite(iaedi)
6222                     sw.J_opt(im) = 0.5 * ((iaedo / max(iae_do0_base, 1e-12)) + (iaedi /
      max(iae_di0_base, 1e-12)));
6223                 elseif ~has_j
6224                     sw.J_opt(im) = NaN;
6225                 end
6226             end
6227
6228             if ~isfield(sw, 'source_mst_obj') || numel(sw.source_mst_obj) ~= nM
6229                 sw.source_mst_obj = sw.Mst_obj(:);
6230             end
6231             if ~isfield(sw, 'source_idx') || numel(sw.source_idx) ~= nM
6232                 sw.source_idx = (1:nM).';
6233             end
6234
6235             Plantas_out(ip).(sweep_field) = sw;
6236         end
6237     end
6238 end
6239
6240 function T = build_time_metrics_table_td(Plantas, idx_run, runPI, runPID, runSP, runMSP, Go,
      Gi, t, dt, settling_p, pade_order, N_der, Tf_min)
6241     vars =
      {'planta','tipo','ctrl','pick_status','mst_ref','mst_obj_src','mst_real','mst_err','kc','ti','
      td','j_opt','j_sweep','j_source', ...
6242         'tr_90','ts','os_pct','os_ratio','tp','yp','yss','offset','iae_ref', ...
6243         'do_peak','do_tp','do_ts','do_yss','do_offset','iae_do', ...
6244         'di_peak','di_tp','di_ts','di_yss','di_offset','iae_di', ...
6245         'seq_peak','seq_tp','seq_yfinal','seq_iae'};
6246     rows = cell(0, numel(vars));
6247
6248     ctrl_map = {
6249         'PI',  runPI,  'PI'
6250         'PID', runPID, 'PID'
6251         'SP',  runSP,  'SP'
6252         'MSP', runMSP, 'MSP'
6253         };
6254
6255     for k = 1:numel(idx_run)
6256         ip = idx_run(k);
6257         for ic = 1:size(ctrl_map,1)
6258             ctrl_key = ctrl_map{ic,1};
6259             runFlag = ctrl_map{ic,2};
6260             plant_field = ctrl_map{ic,3};
6261             if ~runFlag
6262                 continue;
6263             end
6264
6265             pick = getfield_safe(getfield_safe(Plantas(ip), 'fixed_eval', struct()),
      plant_field, struct());
6266             if ~isstruct(pick) || ~isfield(pick, 'opt')
6267                 rows(end+1,:) = {Plantas(ip).id, Plantas(ip).tipo,
      ctrl_key_to_user(ctrl_key), ...
6268                     'NO_PICK', NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, 'NO_SOURCE', ...
6269                     NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, ...
6270                     NaN, NaN, NaN, NaN, NaN, NaN, ...
6271                     NaN, NaN, NaN, NaN, NaN, NaN, ...
6272                     NaN, NaN, NaN, NaN}; %#ok<AGROW>
6273                 continue;
6274             end
6275
6276             [has_r, mr] = get_opt_metrics_y_r_td(Plantas(ip), ctrl_key, Go, Gi, t,
      settling_p, pade_order, N_der, Tf_min);
```

```
6277            [has_do, mdo] = get_opt_metrics_y_do_td(Plantas(ip), ctrl_key, Go, Gi, t,
    settling_p, pade_order, N_der, Tf_min);
6278            [has_di, mdi] = get_opt_metrics_y_di_td(Plantas(ip), ctrl_key, Go, Gi, t,
    settling_p, pade_order, N_der, Tf_min);
6279            [has_seq, mseq] = get_opt_metrics_combined_td(Plantas(ip), ctrl_key, Go, Gi, dt,
    settling_p, pade_order, N_der, Tf_min);
6280
6281            if ~has_r, mr = local_nan_metrics_td(); end
6282            if ~has_do, mdo = local_nan_dist_metrics_td(); end
6283            if ~has_di, mdi = local_nan_dist_metrics_td(); end
6284            if ~has_seq, mseq = local_nan_comb_metrics_td(); end
6285            [j_tiein, j_source] = recompute_tiein_j_td(Plantas(ip), pick.opt.J_opt,
    mdo.iae_abs, mdi.iae_abs);
6286
6287            rows(end+1,:) = { ...
6288                Plantas(ip).id, Plantas(ip).tipo, ctrl_key_to_user(ctrl_key), ...
6289                'OK', pick.mst_ref, pick.mst_obj_src, pick.opt.mst_opt, pick.opt.mst_err, ...
6290                pick.opt.kc_opt, pick.opt.ti_opt, pick.opt.td_opt, j_tiein, pick.opt.J_opt,
    j_source, ...
6291                mr.rise90, mr.settle, mr.overshoot_pct, mr.overshoot_ratio, mr.peak_time,
    mr.peak_value, mr.yss, mr.ess, mr.iae_ref, ...
6292                mdo.peak_abs, mdo.peak_time, mdo.settle0, mdo.yss, mdo.ess0, mdo.iae_abs, ...
6293                mdi.peak_abs, mdi.peak_time, mdi.settle0, mdi.yss, mdi.ess0, mdi.iae_abs, ...
6294                mseq.peak_abs, mseq.peak_time, mseq.yfinal, mseq.iae_seq}; %#ok<AGROW>
6295        end
6296    end
6297
6298    T = cell2table(rows, 'VariableNames', vars);
6299 end
6300
6301 function [j_eval, j_source] = recompute_tiein_j_td(plant, j_sweep, iae_do, iae_di)
6302     j_eval = j_sweep;
6303     j_source = 'SWEEP_FALLBACK';
6304
6305     [iae_do0_base, iae_di0_base, has_base] = get_base_weights_td(plant);
6306     if ~has_base
6307         return;
6308     end
6309     if ~isscalar(iae_do) || ~isscalar(iae_di) || ~isfinite(iae_do) || ~isfinite(iae_di)
6310         return;
6311     end
6312
6313     j_eval = 0.5 * ((iae_do / max(iae_do0_base, 1e-12)) + (iae_di / max(iae_di0_base,
    1e-12)));
6314     j_source = 'TIEIN_RECALC';
6315 end
6316
6317 function [iae_do0_base, iae_di0_base, has_base] = get_base_weights_td(plant)
6318     iae_do0_base = NaN;
6319     iae_di0_base = NaN;
6320     has_base = false;
6321
6322     if ~isstruct(plant) || ~isfield(plant, 'base') || ~isfield(plant.base, 'result')
6323         return;
6324     end
6325     if ~isfield(plant.base.result, 'do0') || ~isfield(plant.base.result, 'di0')
6326         return;
6327     end
6328     if ~isfield(plant.base.result.do0, 'iaedo_do0') || ~isfield(plant.base.result.di0,
    'iaedi_di0')
6329         return;
6330     end
6331
6332     iae_do0_base = plant.base.result.do0.iaedo_do0;
6333     iae_di0_base = plant.base.result.di0.iaedi_di0;
```

```
6334      has_base = isfinite(iae_do0_base) && isfinite(iae_di0_base) && iae_do0_base > 0 &&
      iae_di0_base > 0;
6335  end
6336
6337  function [T_ref, T_di, T_do] = build_time_scenario_tables_td(T_metrics, plant_id,
      include_plant_cols)
6338      if nargin < 2
6339          plant_id = '';
6340      end
6341      if nargin < 3
6342          include_plant_cols = false;
6343      end
6344
6345      if isempty(T_metrics)
6346          T_ref = table();
6347          T_di = table();
6348          T_do = table();
6349          return;
6350      end
6351
6352      mask = true(height(T_metrics), 1);
6353      if ~isempty(plant_id)
6354          mask = string(T_metrics.planta) == string(plant_id);
6355      end
6356      Tp = T_metrics(mask, :);
6357      ctrl_disp = string(Tp.ctrl);
6358      ctrl_disp(ctrl_disp == "PI+SP") = "SP";
6359      ctrl_disp(ctrl_disp == "PI+MSP") = "MSP";
6360      Tp.ctrl = cellstr(ctrl_disp);
6361
6362      base_vars = {'ctrl','kc','ti','td','mst_real','j_sweep'};
6363      base_names = {'Controlador','k_c','tau_i','tau_d','Mst','J'};
6364
6365      if include_plant_cols
6366          base_vars = [{'planta','tipo'}, base_vars];
6367          base_names = [{'Planta','Tipo'}, base_names];
6368      end
6369
6370      ref_vars = [base_vars, {'tr_90','ts','os_pct','offset','iae_ref'}];
6371      ref_names = [base_names,
      {'Tiempo_subida_90','Tiempo_estabilizacion','Sobrepaso_pct','Offset','IAE_referencia'}];
6372
6373      di_vars = [base_vars, {'di_peak','di_ts','di_offset','iae_di'}];
6374      di_names = [base_names,
      {'Pico_absoluto_di','Tiempo_estabilizacion_di','Offset_di','IAE_perturbacion_entrada'}];
6375
6376      do_vars = [base_vars, {'do_peak','do_ts','do_offset','iae_do'}];
6377      do_names = [base_names,
      {'Pico_absoluto_do','Tiempo_estabilizacion_do','Offset_do','IAE_perturbacion_salida'}];
6378
6379      T_ref = Tp(:, ref_vars);
6380      T_ref.Properties.VariableNames = ref_names;
6381
6382      T_di = Tp(:, di_vars);
6383      T_di.Properties.VariableNames = di_names;
6384
6385      T_do = Tp(:, do_vars);
6386      T_do.Properties.VariableNames = do_names;
6387  end
6388
6389  function T = build_time_winner_table_td(T_metrics)
6390      vars =
      {'planta','tipo','best_j_ctrl','best_j','best_tr_ctrl','best_tr','best_ts_ctrl','best_ts', ...
6391
      'best_os_ctrl','best_os','best_offset_ctrl','best_offset','best_seq_iae_ctrl','best_seq_iae'};
```

```
    rows = cell(0, numel(vars));
    if isempty(T_metrics)
        T = cell2table(rows, 'VariableNames', vars);
        return;
    end

    plantas = unique(string(T_metrics.planta), 'stable');
    for i = 1:numel(plantas)
        Tp = T_metrics(string(T_metrics.planta) == plantas(i), :);
        tipo = string(Tp.tipo(1));
        [best_j_ctrl, best_j] = pick_best_row_td(Tp.ctrl, Tp.j_sweep, 'min');
        [best_tr_ctrl, best_tr] = pick_best_row_td(Tp.ctrl, Tp.tr_90, 'min');
        [best_ts_ctrl, best_ts] = pick_best_row_td(Tp.ctrl, Tp.ts, 'min');
        [best_os_ctrl, best_os] = pick_best_row_td(Tp.ctrl, Tp.os_pct, 'min');
        [best_offset_ctrl, best_offset] = pick_best_row_td(Tp.ctrl, abs(Tp.offset), 'min');
        [best_seq_ctrl, best_seq] = pick_best_row_td(Tp.ctrl, Tp.seq_iae, 'min');
        rows(end+1,:) = {char(plantas(i)), char(tipo), best_j_ctrl, best_j, best_tr_ctrl,
best_tr, ...
            best_ts_ctrl, best_ts, best_os_ctrl, best_os, best_offset_ctrl, best_offset, ...
            best_seq_ctrl, best_seq}; %#ok<AGROW>
    end
    T = cell2table(rows, 'VariableNames', vars);
end

function T = build_deadtime_error_table_td(Plantas, idx_run, runPI, runPID, runSP, runMSP,
Go, Gi, t, settling_p, pade_order, N_der, Tf_min, deadtime_err_list)
    vars =
{'planta','tipo','ctrl','err_pct','is_stable','gm','gm_db','pm_deg','wcg','wcp','bw_cl','dm','
tr_90','ts','os_pct','yss','offset'};
    rows = cell(0, numel(vars));
    s = tf('s');

    for k = 1:numel(idx_run)
        ip = idx_run(k);
        theta0 = Plantas(ip).P_dt.InputDelay;
        [num0, den0] = tfdata(Plantas(ip).P_dt,'v');
        P0 = tf(num0, den0);

        if runPI && isfield(Plantas(ip),'PI') && isfield(Plantas(ip).PI,'opt') &&
~isempty(fieldnames(Plantas(ip).PI.opt))
            C = build_controller_tf('PI', Plantas(ip).PI.opt.kc_opt,
Plantas(ip).PI.opt.ti_opt, Plantas(ip).PI.opt.td_opt, Plantas(ip).tipo, Plantas(ip).id,
theta0, N_der, Tf_min);
            for e = deadtime_err_list
                theta_real = theta0*(1+e);
                try
                    P_real = pade(tf(num0, den0, 'InputDelay', theta_real), pade_order);
                    L = P_real*C;
                    T_cl = feedback(L, 1);
                    if isstable(T_cl)
                        y = step(T_cl, t);
                        mt = local_step_metrics_td(y, t, settling_p);
                        mf = local_freq_metrics_from_loop(L, T_cl);
                        rows(end+1,:) = {Plantas(ip).id, Plantas(ip).tipo, 'PI', 100*e,
true, mf.gm, mf.gm_db, mf.pm_deg, mf.wcg, mf.wcp, mf.bw_cl, mf.dm, mt.rise90, mt.settle,
mt.overshoot_pct, mt.yss, mt.ess}; %#ok<AGROW>
                    else
                        rows(end+1,:) = {Plantas(ip).id, Plantas(ip).tipo, 'PI', 100*e,
false, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN}; %#ok<AGROW>
                    end
                catch
                    rows(end+1,:) = {Plantas(ip).id, Plantas(ip).tipo, 'PI', 100*e, false,
NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN}; %#ok<AGROW>
                end
            end
```

```
6446         end
6447 
6448         if runPID && isfield(Plantas(ip),'PID') && isfield(Plantas(ip).PID,'opt') &&
     ~isempty(fieldnames(Plantas(ip).PID.opt))
6449             C = build_controller_tf('PID', Plantas(ip).PID.opt.kc_opt,
     Plantas(ip).PID.opt.ti_opt, Plantas(ip).PID.opt.td_opt, Plantas(ip).tipo, Plantas(ip).id,
     theta0, N_der, Tf_min);
6450             for e = deadtime_err_list
6451                 theta_real = theta0*(1+e);
6452                 try
6453                     P_real = pade(tf(num0, den0, 'InputDelay', theta_real), pade_order);
6454                     L = P_real*C;
6455                     T_cl = feedback(L, 1);
6456                     if isstable(T_cl)
6457                         y = step(T_cl, t);
6458                         mt = local_step_metrics_td(y, t, settling_p);
6459                         mf = local_freq_metrics_from_loop(L, T_cl);
6460                         rows(end+1,:) = {Plantas(ip).id, Plantas(ip).tipo, 'PID', 100*e,
     true, mf.gm, mf.gm_db, mf.pm_deg, mf.wcg, mf.wcp, mf.bw_cl, mf.dm, mt.rise90, mt.settle,
     mt.overshoot_pct, mt.yss, mt.ess}; %#ok<AGROW>
6461                     else
6462                         rows(end+1,:) = {Plantas(ip).id, Plantas(ip).tipo, 'PID', 100*e,
     false, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN}; %#ok<AGROW>
6463                     end
6464                 catch
6465                     rows(end+1,:) = {Plantas(ip).id, Plantas(ip).tipo, 'PID', 100*e, false,
     NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN}; %#ok<AGROW>
6466                 end
6467             end
6468         end
6469 
6470         if runSP && isfield(Plantas(ip),'SP') && isfield(Plantas(ip).SP,'opt') &&
     ~isempty(fieldnames(Plantas(ip).SP.opt))
6471             C = Plantas(ip).SP.opt.kc_opt * (1 + 1/(s*max(Plantas(ip).SP.opt.ti_opt,1e-6)));
6472             for e = deadtime_err_list
6473                 L_real = theta0*(1+e);
6474                 try
6475                     [~, Hr_sp, ~, Leq_sp, is_stable] = smith_tf(P0, theta0, L_real, C, Go,
     Gi, pade_order);
6476                     if is_stable
6477                         y = step(Hr_sp, t);
6478                         mt = local_step_metrics_td(y, t, settling_p);
6479                         mf = local_freq_metrics_from_loop(Leq_sp, Hr_sp);
6480                         rows(end+1,:) = {Plantas(ip).id, Plantas(ip).tipo, 'SP', 100*e,
     true, mf.gm, mf.gm_db, mf.pm_deg, mf.wcg, mf.wcp, mf.bw_cl, mf.dm, mt.rise90, mt.settle,
     mt.overshoot_pct, mt.yss, mt.ess}; %#ok<AGROW>
6481                     else
6482                         rows(end+1,:) = {Plantas(ip).id, Plantas(ip).tipo, 'SP', 100*e,
     false, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN}; %#ok<AGROW>
6483                     end
6484                 catch
6485                     rows(end+1,:) = {Plantas(ip).id, Plantas(ip).tipo, 'SP', 100*e, false,
     NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN}; %#ok<AGROW>
6486                 end
6487             end
6488         end
6489 
6490         if runMSP && isfield(Plantas(ip),'MSP') && isfield(Plantas(ip).MSP,'opt') &&
     ~isempty(fieldnames(Plantas(ip).MSP.opt))
6491             C = Plantas(ip).MSP.opt.kc_opt * (1 + 1/(s*max(Plantas(ip).MSP.opt.ti_opt,1e-6)));
6492             for e = deadtime_err_list
6493                 L_real = theta0*(1+e);
6494                 try
6495                     [~, Hr_msp, ~, Leq_msp, is_stable, ~] = msp_tf(P0, theta0, L_real, C,
     Plantas(ip).tipo, Plantas(ip).id, Go, Gi, pade_order);
```

```
                    if is_stable
                        y = step(Hr_msp, t);
                        mt = local_step_metrics_td(y, t, settling_p);
                        mf = local_freq_metrics_from_loop(Leq_msp, Hr_msp);
                        rows(end+1,:) = {Plantas(ip).id, Plantas(ip).tipo, 'MSP', 100*e, true, mf.gm, mf.gm_db, mf.pm_deg, mf.wcg, mf.wcp, mf.bw_cl, mf.dm, mt.rise90, mt.settle, mt.overshoot_pct, mt.yss, mt.ess}; %#ok<AGROW>
                    else
                        rows(end+1,:) = {Plantas(ip).id, Plantas(ip).tipo, 'MSP', 100*e, false, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN}; %#ok<AGROW>
                    end
                catch
                    rows(end+1,:) = {Plantas(ip).id, Plantas(ip).tipo, 'MSP', 100*e, false, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN}; %#ok<AGROW>
                end
            end
        end
    end

    T = cell2table(rows, 'VariableNames', vars);
end

function plot_time_domain_plant_figure_td(plant, runPI, runPID, runSP, runMSP, styles, Go, Gi, t, dt, pade_order, N_der, Tf_min, line_w_plot, export_flag, out_folder, mst_ref_eval, t_plot_short_end)
    if nargin < 18 || isempty(t_plot_short_end)
        t_plot_short_end = 10;
    end
    seq_do_time = t_plot_short_end;
    seq_di_time = 2 * t_plot_short_end;
    seq_end_time = 3 * t_plot_short_end;
    t_plot_short = 0:dt:min(t_plot_short_end, t(end));
    [respPI, hasPI] = simulate_selected_controller_responses_td(plant, 'PI', runPI, Go, Gi, t_plot_short, dt, pade_order, N_der, Tf_min, 0, seq_do_time, seq_di_time, seq_end_time);
    [respPID, hasPID] = simulate_selected_controller_responses_td(plant, 'PID', runPID, Go, Gi, t_plot_short, dt, pade_order, N_der, Tf_min, 0, seq_do_time, seq_di_time, seq_end_time);
    [respSP, hasSP] = simulate_selected_controller_responses_td(plant, 'SP', runSP, Go, Gi, t_plot_short, dt, pade_order, N_der, Tf_min, 0, seq_do_time, seq_di_time, seq_end_time);
    [respMSP, hasMSP] = simulate_selected_controller_responses_td(plant, 'MSP', runMSP, Go, Gi, t_plot_short, dt, pade_order, N_der, Tf_min, 0, seq_do_time, seq_di_time, seq_end_time);

    fig = figure('Name', sprintf('Dominio del tiempo - %s', plant.id), 'NumberTitle', 'off', 'Color', 'w', ...
        'Units', 'centimeters', 'Position', [2 2 15.5 17.5]);

    tl = tiledlayout(fig, 4, 1, 'TileSpacing', 'compact', 'Padding', 'compact');
    title(tl, sprintf('%s | Mst = %.2f | Respuestas temporales', plant_title_with_model(plant), mst_ref_eval), ...
        'FontWeight', 'bold', 'Interpreter', 'tex');

    ax1 = nexttile(tl,1); hold(ax1,'on'); grid(ax1,'on');
    plot_resp_if_available_td(ax1, hasPI, respPI.t, respPI.y_r, styles.PI, 'PI', line_w_plot);
    plot_resp_if_available_td(ax1, hasPID, respPID.t, respPID.y_r, styles.PID, 'PID', line_w_plot);
    plot_resp_if_available_td(ax1, hasSP, respSP.t, respSP.y_r, styles.SP, 'SP', line_w_plot);
    plot_resp_if_available_td(ax1, hasMSP, respMSP.t, respMSP.y_r, styles.MSP, 'MSP', line_w_plot);
    ylabel(ax1, 'y(t)'); title(ax1, 'Setpoint');
    xlim(ax1, [0 t_plot_short_end]);
    legend(ax1, 'Location', 'best');

    ax2 = nexttile(tl,2); hold(ax2,'on'); grid(ax2,'on');
    plot_resp_if_available_td(ax2, hasPI, respPI.t, respPI.y_do, styles.PI, 'PI', line_w_plot);
```

```
6545      plot_resp_if_available_td(ax2, hasPID, respPID.t, respPID.y_do, styles.PID, 'PID',
      line_w_plot);
6546      plot_resp_if_available_td(ax2, hasSP, respSP.t, respSP.y_do, styles.SP, 'SP',
      line_w_plot);
6547      plot_resp_if_available_td(ax2, hasMSP, respMSP.t, respMSP.y_do, styles.MSP, 'MSP',
      line_w_plot);
6548      ylabel(ax2, 'y(t)'); title(ax2, 'Perturbacion de salida');
6549      xlim(ax2, [0 t_plot_short_end]);
6550
6551      ax3 = nexttile(tl,3); hold(ax3,'on'); grid(ax3,'on');
6552      plot_resp_if_available_td(ax3, hasPI, respPI.t, respPI.y_di, styles.PI, 'PI',
      line_w_plot);
6553      plot_resp_if_available_td(ax3, hasPID, respPID.t, respPID.y_di, styles.PID, 'PID',
      line_w_plot);
6554      plot_resp_if_available_td(ax3, hasSP, respSP.t, respSP.y_di, styles.SP, 'SP',
      line_w_plot);
6555      plot_resp_if_available_td(ax3, hasMSP, respMSP.t, respMSP.y_di, styles.MSP, 'MSP',
      line_w_plot);
6556      ylabel(ax3, 'y(t)'); title(ax3, 'Perturbacion de entrada');
6557      xlim(ax3, [0 t_plot_short_end]);
6558
6559      ax4 = nexttile(tl,4); hold(ax4,'on'); grid(ax4,'on');
6560      plot_resp_if_available_td(ax4, hasPI, respPI.t_seq, respPI.y_seq, styles.PI, 'PI',
      line_w_plot);
6561      plot_resp_if_available_td(ax4, hasPID, respPID.t_seq, respPID.y_seq, styles.PID, 'PID',
      line_w_plot);
6562      plot_resp_if_available_td(ax4, hasSP, respSP.t_seq, respSP.y_seq, styles.SP, 'SP',
      line_w_plot);
6563      plot_resp_if_available_td(ax4, hasMSP, respMSP.t_seq, respMSP.y_seq, styles.MSP, 'MSP',
      line_w_plot);
6564      xline(ax4, seq_do_time, '--', 'do step');
6565      xline(ax4, seq_di_time, '--', 'di step');
6566      xlim(ax4, [0 seq_end_time]);
6567      xlabel(ax4, 'Tiempo (s)'); ylabel(ax4, 'y(t)'); title(ax4, 'Escenario compuesto');
6568
6569      if export_flag
6570          drawnow;
6571          tag = strrep(sprintf('%.2f', mst_ref_eval), '.', 'p');
6572          safe_export_dual(fig, tl, ...
6573              fullfile(out_folder, sprintf('TimeDomain_Mst%s_%s.png', tag, plant.id)), ...
6574              fullfile(out_folder, sprintf('TimeDomain_Mst%s_%s.pdf', tag, plant.id)), 220);
6575      end
6576  end
6577
6578  function plot_deadtime_error_figure_td(Plantas, idx_run, ctrl_key, Go, Gi, dt, pade_order,
      N_der, Tf_min, deadtime_err_list, line_w_plot, export_flag, out_folder, mst_ref_eval)
6579      nPlants = numel(idx_run);
6580      nCols = min(2, max(1, nPlants));
6581      nRows = ceil(nPlants / nCols);
6582
6583      fig = figure( ...
6584          'Name', sprintf('Error tiempo muerto - %s', ctrl_key_to_user(ctrl_key)), ...
6585          'NumberTitle', 'off', ...
6586          'Color', 'w', ...
6587          'Units', 'centimeters', ...
6588          'Position', [2 2 max(15, 7.4*nCols) max(9, 4.9*nRows)]);
6589
6590      tl = tiledlayout(fig, nRows, nCols, 'TileSpacing', 'compact', 'Padding', 'compact');
6591      title(tl, sprintf('%s | Escenario compuesto vs error de tiempo de retardo | Mst = %.2f',
      ...
6592          ctrl_key_to_user(ctrl_key), mst_ref_eval), 'FontWeight', 'bold', 'Interpreter',
      'none');
6593
6594      for kp = 1:nPlants
6595          ip = idx_run(kp);
```

```
6596          ax = nexttile(tl, kp);
6597          hold(ax, 'on');
6598          grid(ax, 'on');
6599
6600          has_any = false;
6601          for e = deadtime_err_list
6602              [resp, has_resp] = simulate_selected_controller_responses_td( ...
6603                  Plantas(ip), ctrl_key, true, Go, Gi, 0:dt:10, dt, pade_order, N_der, Tf_min,
      e, 10, 20, 30);
6604              if ~has_resp || isempty(resp.y_seq)
6605                  continue;
6606              end
6607
6608              plot(ax, resp.t_seq, resp.y_seq, ...
6609                  'LineWidth', line_w_plot, ...
6610                  'DisplayName', sprintf('%+.0f%%', 100*e));
6611              has_any = true;
6612          end
6613
6614          xline(ax, 10, '--', 'do step');
6615          xline(ax, 20, '--', 'di step');
6616          xlim(ax, [0 30]);
6617          xlabel(ax, 'Tiempo (s)');
6618          ylabel(ax, 'y(t)');
6619          title(ax, plant_title_with_model(Plantas(ip)), 'Interpreter', 'tex', 'FontWeight',
      'normal');
6620
6621          if has_any
6622              legend(ax, 'Location', 'bestoutside');
6623          else
6624              text(ax, 0.5, 0.5, sprintf('Sin datos validos para %s',
      ctrl_key_to_user(ctrl_key)), ...
6625                  'Units', 'normalized', 'HorizontalAlignment', 'center', 'VerticalAlignment',
      'middle', ...
6626                  'Color', [0.35 0.35 0.35]);
6627          end
6628      end
6629
6630      if export_flag
6631          drawnow;
6632          tag = strrep(sprintf('%.2f', mst_ref_eval), '.', 'p');
6633          base_name = sprintf('DeadtimeError_%s_Mst%s', strrep(ctrl_key_to_user(ctrl_key), '+',
      '_'), tag);
6634          safe_export_dual(fig, tl, ...
6635              fullfile(out_folder, [base_name '.png']), ...
6636              fullfile(out_folder, [base_name '.pdf']), 220);
6637      end
6638  end
6639
6640  function plot_deadtime_error_response_panels_td(Plantas, idx_run, runPI, runPID, runSP,
      runMSP, styles, Go, Gi, dt, pade_order, N_der, Tf_min, deadtime_err_list, line_w_plot,
      export_flag, out_folder, mst_ref_eval, time_plot_short_by_plant)
6641      for ie = 1:numel(deadtime_err_list)
6642          e = deadtime_err_list(ie);
6643          for kp = 1:numel(idx_run)
6644              ip = idx_run(kp);
6645              t_plot_short_end = 10;
6646              if nargin >= 19 && ~isempty(time_plot_short_by_plant) &&
      numel(time_plot_short_by_plant) >= kp
6647                  t_plot_short_end = time_plot_short_by_plant(kp);
6648              end
6649              seq_do_time = t_plot_short_end;
6650              seq_di_time = 2 * t_plot_short_end;
6651              seq_end_time = 3 * t_plot_short_end;
6652              t_short = 0:dt:t_plot_short_end;
```

```
6653
6654            [respPI, hasPI] = simulate_selected_controller_responses_td( ...
6655                Plantas(ip), 'PI', runPI, Go, Gi, t_short, dt, pade_order, N_der, Tf_min, e,
    seq_do_time, seq_di_time, seq_end_time);
6656            [respPID, hasPID] = simulate_selected_controller_responses_td( ...
6657                Plantas(ip), 'PID', runPID, Go, Gi, t_short, dt, pade_order, N_der, Tf_min,
    e, seq_do_time, seq_di_time, seq_end_time);
6658            [respSP, hasSP] = simulate_selected_controller_responses_td( ...
6659                Plantas(ip), 'SP', runSP, Go, Gi, t_short, dt, pade_order, N_der, Tf_min, e,
    seq_do_time, seq_di_time, seq_end_time);
6660            [respMSP, hasMSP] = simulate_selected_controller_responses_td( ...
6661                Plantas(ip), 'MSP', runMSP, Go, Gi, t_short, dt, pade_order, N_der, Tf_min,
    e, seq_do_time, seq_di_time, seq_end_time);
6662
6663            show_nominal_ref = abs(e) > 1e-12;
6664            if show_nominal_ref
6665                [respPI0, hasPI0] = simulate_selected_controller_responses_td( ...
6666                    Plantas(ip), 'PI', runPI, Go, Gi, t_short, dt, pade_order, N_der, Tf_min,
    0, seq_do_time, seq_di_time, seq_end_time);
6667                [respPID0, hasPID0] = simulate_selected_controller_responses_td( ...
6668                    Plantas(ip), 'PID', runPID, Go, Gi, t_short, dt, pade_order, N_der,
    Tf_min, 0, seq_do_time, seq_di_time, seq_end_time);
6669                [respSP0, hasSP0] = simulate_selected_controller_responses_td( ...
6670                    Plantas(ip), 'SP', runSP, Go, Gi, t_short, dt, pade_order, N_der, Tf_min,
    0, seq_do_time, seq_di_time, seq_end_time);
6671                [respMSP0, hasMSP0] = simulate_selected_controller_responses_td( ...
6672                    Plantas(ip), 'MSP', runMSP, Go, Gi, t_short, dt, pade_order, N_der,
    Tf_min, 0, seq_do_time, seq_di_time, seq_end_time);
6673            else
6674                respPI0 = struct(); hasPI0 = false;
6675                respPID0 = struct(); hasPID0 = false;
6676                respSP0 = struct(); hasSP0 = false;
6677                respMSP0 = struct(); hasMSP0 = false;
6678            end
6679
6680            if ~(hasPI || hasPID || hasSP || hasMSP)
6681                continue;
6682            end
6683
6684            fig = figure( ...
6685                'Name', sprintf('Error retardo %s | %s | %+.0f%%', Plantas(ip).id,
    plant_title_with_model(Plantas(ip)), 100*e), ...
6686                'NumberTitle', 'off', ...
6687                'Color', 'w', ...
6688                'Units', 'centimeters', ...
6689                'Position', [2 2 15.8 17.8]);
6690
6691            tl = tiledlayout(fig, 4, 1, 'TileSpacing', 'compact', 'Padding', 'compact');
6692            title(tl, sprintf('Planta %s | %s | Mst = %.2f | Error retardo = %+.0f%%', ...
6693                Plantas(ip).id, plant_display_model(Plantas(ip)), mst_ref_eval, 100*e), ...
6694                'FontWeight', 'bold', 'Interpreter', 'tex');
6695
6696            ax1 = nexttile(tl, 1); hold(ax1, 'on'); grid(ax1, 'on');
6697            plot_nominal_ref_td(ax1, hasPI0,  local_resp_with_field_td(respPI0,  't', 'y_r'),
    styles.PI,  'PI',  line_w_plot);
6698            plot_nominal_ref_td(ax1, hasPID0, local_resp_with_field_td(respPID0, 't', 'y_r'),
    styles.PID, 'PID', line_w_plot);
6699            plot_nominal_ref_td(ax1, hasSP0,  local_resp_with_field_td(respSP0,  't', 'y_r'),
    styles.SP,  'SP',  line_w_plot);
6700            plot_nominal_ref_td(ax1, hasMSP0, local_resp_with_field_td(respMSP0, 't', 'y_r'),
    styles.MSP, 'MSP', line_w_plot);
6701            plot_resp_series_td(ax1, hasPI,  respPI.t,  respPI.y_r,  styles.PI,  'PI',
    line_w_plot);
6702            plot_resp_series_td(ax1, hasPID, respPID.t, respPID.y_r, styles.PID, 'PID',
    line_w_plot);
```

```
6703            plot_resp_series_td(ax1, hasSP,  respSP.t,  respSP.y_r,  styles.SP,  'SP',
    line_w_plot);
6704            plot_resp_series_td(ax1, hasMSP, respMSP.t, respMSP.y_r, styles.MSP, 'MSP',
    line_w_plot);
6705            if show_nominal_ref
6706                plot(ax1, nan, nan, ...
6707                    'Color', [0.45 0.45 0.45], ...
6708                    'LineStyle', ':', ...
6709                    'LineWidth', max(0.8, line_w_plot - 0.45), ...
6710                    'DisplayName', 'Referencia nominal (0%)');
6711            end
6712            ylabel(ax1, 'y(t)');
6713            title(ax1, 'Setpoint');
6714            xlim(ax1, [0 t_plot_short_end]);
6715            legend(ax1, 'Location', 'eastoutside');
6716
6717            ax2 = nexttile(tl, 2); hold(ax2, 'on'); grid(ax2, 'on');
6718            plot_nominal_ref_td(ax2, hasPI0,  local_resp_with_field_td(respPI0,  't',
    'y_do'), styles.PI,  'PI',  line_w_plot);
6719            plot_nominal_ref_td(ax2, hasPID0, local_resp_with_field_td(respPID0, 't',
    'y_do'), styles.PID, 'PID', line_w_plot);
6720            plot_nominal_ref_td(ax2, hasSP0,  local_resp_with_field_td(respSP0,  't',
    'y_do'), styles.SP,  'SP',  line_w_plot);
6721            plot_nominal_ref_td(ax2, hasMSP0, local_resp_with_field_td(respMSP0, 't',
    'y_do'), styles.MSP, 'MSP', line_w_plot);
6722            plot_resp_series_td(ax2, hasPI,  respPI.t,  respPI.y_do,  styles.PI,  'PI',
    line_w_plot);
6723            plot_resp_series_td(ax2, hasPID, respPID.t, respPID.y_do, styles.PID, 'PID',
    line_w_plot);
6724            plot_resp_series_td(ax2, hasSP,  respSP.t,  respSP.y_do,  styles.SP,  'SP',
    line_w_plot);
6725            plot_resp_series_td(ax2, hasMSP, respMSP.t, respMSP.y_do, styles.MSP, 'MSP',
    line_w_plot);
6726            ylabel(ax2, 'y(t)');
6727            title(ax2, 'Perturbacion de salida');
6728            xlim(ax2, [0 t_plot_short_end]);
6729
6730            ax3 = nexttile(tl, 3); hold(ax3, 'on'); grid(ax3, 'on');
6731            plot_nominal_ref_td(ax3, hasPI0,  local_resp_with_field_td(respPI0,  't',
    'y_di'), styles.PI,  'PI',  line_w_plot);
6732            plot_nominal_ref_td(ax3, hasPID0, local_resp_with_field_td(respPID0, 't',
    'y_di'), styles.PID, 'PID', line_w_plot);
6733            plot_nominal_ref_td(ax3, hasSP0,  local_resp_with_field_td(respSP0,  't',
    'y_di'), styles.SP,  'SP',  line_w_plot);
6734            plot_nominal_ref_td(ax3, hasMSP0, local_resp_with_field_td(respMSP0, 't',
    'y_di'), styles.MSP, 'MSP', line_w_plot);
6735            plot_resp_series_td(ax3, hasPI,  respPI.t,  respPI.y_di,  styles.PI,  'PI',
    line_w_plot);
6736            plot_resp_series_td(ax3, hasPID, respPID.t, respPID.y_di, styles.PID, 'PID',
    line_w_plot);
6737            plot_resp_series_td(ax3, hasSP,  respSP.t,  respSP.y_di,  styles.SP,  'SP',
    line_w_plot);
6738            plot_resp_series_td(ax3, hasMSP, respMSP.t, respMSP.y_di, styles.MSP, 'MSP',
    line_w_plot);
6739            ylabel(ax3, 'y(t)');
6740            title(ax3, 'Perturbacion de entrada');
6741            xlim(ax3, [0 t_plot_short_end]);
6742
6743            ax4 = nexttile(tl, 4); hold(ax4, 'on'); grid(ax4, 'on');
6744            plot_nominal_ref_td(ax4, hasPI0,  local_resp_with_field_td(respPI0,  't_seq',
    'y_seq'), styles.PI,  'PI',  line_w_plot);
6745            plot_nominal_ref_td(ax4, hasPID0, local_resp_with_field_td(respPID0, 't_seq',
    'y_seq'), styles.PID, 'PID', line_w_plot);
6746            plot_nominal_ref_td(ax4, hasSP0,  local_resp_with_field_td(respSP0,  't_seq',
    'y_seq'), styles.SP,  'SP',  line_w_plot);
```

```
6747            plot_nominal_ref_td(ax4, hasMSP0, local_resp_with_field_td(respMSP0, 't_seq',
    'y_seq'), styles.MSP, 'MSP', line_w_plot);
6748            plot_resp_series_td(ax4, hasPI,  respPI.t_seq,  respPI.y_seq,  styles.PI,  'PI',
    line_w_plot);
6749            plot_resp_series_td(ax4, hasPID, respPID.t_seq, respPID.y_seq, styles.PID, 'PID',
    line_w_plot);
6750            plot_resp_series_td(ax4, hasSP,  respSP.t_seq,  respSP.y_seq,  styles.SP,  'SP',
    line_w_plot);
6751            plot_resp_series_td(ax4, hasMSP, respMSP.t_seq, respMSP.y_seq, styles.MSP, 'MSP',
    line_w_plot);
6752            xline(ax4, seq_do_time, '--', 'do step');
6753            xline(ax4, seq_di_time, '--', 'di step');
6754            xlim(ax4, [0 seq_end_time]);
6755            xlabel(ax4, 'Tiempo (s)');
6756            ylabel(ax4, 'y(t)');
6757            title(ax4, 'Escenario compuesto');
6758
6759            if export_flag
6760                drawnow;
6761                tag = strrep(sprintf('%.2f', mst_ref_eval), '.', 'p');
6762                err_tag = local_deadtime_err_tag_td(e);
6763                base_name = sprintf('DeadtimePanels_AllCtrls_%s_%s_Mst%s', Plantas(ip).id,
    err_tag, tag);
6764                safe_export_dual(fig, tl, ...
6765                    fullfile(out_folder, [base_name '.png']), ...
6766                    fullfile(out_folder, [base_name '.pdf']), 220);
6767            end
6768        end
6769    end
6770 end
6771
6772 function plot_resp_if_available_td(ax, has_resp, tx, y, style, label_txt, line_w_plot)
6773     if ~has_resp || isempty(y)
6774         return;
6775     end
6776     plot_resp_series_td(ax, true, tx, y, style, label_txt, line_w_plot);
6777 end
6778
6779 function plot_resp_series_td(ax, has_resp, tx, y, style, label_txt, line_w_plot)
6780     if ~has_resp || isempty(y)
6781         return;
6782     end
6783     line_style = get_time_line_style_td(label_txt);
6784     [marker_sym, marker_idx] = get_time_marker_style_td(tx, label_txt);
6785     plot(ax, tx, y, ...
6786         'Color', style.color, ...
6787         'LineStyle', line_style, ...
6788         'LineWidth', line_w_plot, ...
6789         'Marker', marker_sym, ...
6790         'MarkerIndices', marker_idx, ...
6791         'MarkerSize', 4.5, ...
6792         'DisplayName', label_txt);
6793 end
6794
6795 function plot_nominal_ref_td(ax, has_resp, resp_like, style, label_txt, line_w_plot)
6796     if ~has_resp || ~isstruct(resp_like)
6797         return;
6798     end
6799     if ~isfield(resp_like, 'x') || ~isfield(resp_like, 'y') || isempty(resp_like.x) ||
    isempty(resp_like.y)
6800         return;
6801     end
6802     plot(ax, resp_like.x, resp_like.y, ...
6803         'Color', style.color, ...
6804         'LineStyle', ':', ...
```

```
6805          'LineWidth', max(0.8, line_w_plot - 0.45), ...
6806          'HandleVisibility', 'off');
6807  end
6808
6809  function out = local_resp_with_field_td(resp, xfield, yfield)
6810      out = struct('x', [], 'y', []);
6811      if ~isstruct(resp)
6812          return;
6813      end
6814      if isfield(resp, xfield)
6815          out.x = resp.(xfield);
6816      end
6817      if isfield(resp, yfield)
6818          out.y = resp.(yfield);
6819      end
6820  end
6821
6822  function line_style = get_time_line_style_td(label_txt)
6823      switch upper(string(label_txt))
6824          case "PI"
6825              line_style = '-.';
6826          case "PID"
6827              line_style = '--';
6828          case "SP"
6829              line_style = '-';
6830          case "MSP"
6831              line_style = '-';
6832          otherwise
6833              line_style = '-';
6834      end
6835  end
6836
6837  function [marker_sym, marker_idx] = get_time_marker_style_td(tx, label_txt)
6838      npts = numel(tx);
6839      step_idx = max(1, round(npts / 10));
6840      marker_idx = 1:step_idx:npts;
6841      if marker_idx(end) ~= npts
6842          marker_idx = [marker_idx npts]; %#ok<AGROW>
6843      end
6844
6845      switch upper(string(label_txt))
6846          case "SP"
6847              marker_sym = 'o';
6848          case "MSP"
6849              marker_sym = 's';
6850          otherwise
6851              marker_sym = 'none';
6852              marker_idx = [];
6853      end
6854  end
6855
6856  function [resp, has_resp] = simulate_selected_controller_responses_td(plant, ctrl_key,
      runFlag, Go, Gi, t, dt, pade_order, N_der, Tf_min, deadtime_rel_error, seq_do_time,
      seq_di_time, seq_end_time)
6857      resp = struct('t',t(:), 'y_r',[], 'y_do',[], 'y_di',[], 't_seq',[], 'y_seq',[]);
6858      has_resp = false;
6859      if ~runFlag
6860          return;
6861      end
6862
6863      if nargin < 12 || isempty(seq_do_time)
6864          seq_do_time = 20;
6865      end
6866      if nargin < 13 || isempty(seq_di_time)
6867          seq_di_time = 40;
```

```
6868      end
6869      if nargin < 14 || isempty(seq_end_time)
6870          seq_end_time = 60;
6871      end
6872
6873      s = tf('s');
6874      theta0 = plant.P_dt.InputDelay;
6875      [num0, den0] = tfdata(plant.P_dt, 'v');
6876      P0 = tf(num0, den0);
6877      P  = pade(tf(num0, den0, 'InputDelay', theta0 * (1 + deadtime_rel_error)), pade_order);
6878
6879      t_seq = 0:dt:seq_end_time;
6880      r_sig  = double(t_seq >= 0);
6881      do_sig = double(t_seq >= seq_do_time);
6882      di_sig = double(t_seq >= seq_di_time);
6883      resp.t_seq = t_seq(:);
6884
6885      try
6886          switch upper(ctrl_key)
6887              case 'PI'
6888                  if ~(isfield(plant,'PI') && isfield(plant.PI,'opt') &&
      ~isempty(fieldnames(plant.PI.opt))), return; end
6889                  C = build_controller_tf(ctrl_key, plant.PI.opt.kc_opt, plant.PI.opt.ti_opt,
      plant.PI.opt.td_opt, plant.tipo, plant.id, theta0, N_der, Tf_min);
6890                  L = P*C; S = 1/(1+L); T = L/(1+L);
6891                  Gdo = Go*S;
6892                  Gdi = P*Gi*S;
6893                  resp.y_r  = step(T, t);
6894                  resp.y_do = step(Gdo, t);
6895                  resp.y_di = step(Gdi, t);
6896                  resp.y_seq = lsim(T, r_sig, t_seq) + lsim(Gdo, do_sig, t_seq) + lsim(Gdi,
      di_sig, t_seq);
6897
6898              case 'PID'
6899                  if ~(isfield(plant,'PID') && isfield(plant.PID,'opt') &&
      ~isempty(fieldnames(plant.PID.opt))), return; end
6900                  C = build_controller_tf(ctrl_key, plant.PID.opt.kc_opt, plant.PID.opt.ti_opt,
      plant.PID.opt.td_opt, plant.tipo, plant.id, theta0, N_der, Tf_min);
6901                  L = P*C; S = 1/(1+L); T = L/(1+L);
6902                  Gdo = Go*S;
6903                  Gdi = P*Gi*S;
6904                  resp.y_r  = step(T, t);
6905                  resp.y_do = step(Gdo, t);
6906                  resp.y_di = step(Gdi, t);
6907                  resp.y_seq = lsim(T, r_sig, t_seq) + lsim(Gdo, do_sig, t_seq) + lsim(Gdi,
      di_sig, t_seq);
6908
6909              case 'SP'
6910                  if ~(isfield(plant,'SP') && isfield(plant.SP,'opt') &&
      ~isempty(fieldnames(plant.SP.opt))), return; end
6911                  C = build_controller_tf(ctrl_key, plant.SP.opt.kc_opt, plant.SP.opt.ti_opt,
      plant.SP.opt.td_opt, plant.tipo, plant.id, theta0, N_der, Tf_min);
6912                  L_real = theta0 * (1 + deadtime_rel_error);
6913                  [S_sp, T_sp, Tdi_sp, ~, is_stable] = smith_tf(P0, theta0, L_real, C, Go, Gi,
      pade_order);
6914                  if ~is_stable, return; end
6915                  resp.y_r  = step(T_sp, t);
6916                  resp.y_do = step(S_sp, t);
6917                  resp.y_di = step(Tdi_sp, t);
6918                  resp.y_seq = lsim(T_sp, r_sig, t_seq) + lsim(S_sp, do_sig, t_seq) +
      lsim(Tdi_sp, di_sig, t_seq);
6919
6920              case 'MSP'
6921                  if ~(isfield(plant,'MSP') && isfield(plant.MSP,'opt') &&
      ~isempty(fieldnames(plant.MSP.opt))), return; end
```

```
6922                 C = build_controller_tf(ctrl_key, plant.MSP.opt.kc_opt, plant.MSP.opt.ti_opt,
    plant.MSP.opt.td_opt, plant.tipo, plant.id, theta0, N_der, Tf_min);
6923                 L_real = theta0 * (1 + deadtime_rel_error);
6924                 [Hdo_msp, Hr_msp, Hdi_msp, ~, is_stable, ~] = msp_tf(P0, theta0, L_real, C,
    plant.tipo, plant.id, Go, Gi, pade_order);
6925                 if ~is_stable, return; end
6926                 resp.y_r  = step(Hr_msp, t);
6927                 resp.y_do = step(Hdo_msp, t);
6928                 resp.y_di = step(Hdi_msp, t);
6929                 resp.y_seq = lsim(Hr_msp, r_sig, t_seq) + lsim(Hdo_msp, do_sig, t_seq) +
    lsim(Hdi_msp, di_sig, t_seq);
6930             otherwise
6931                 return;
6932         end
6933         has_resp = true;
6934     catch
6935         has_resp = false;
6936     end
6937 end
6938
6939 function [has_opt, metrics_t] = get_opt_metrics_y_r_td(plant, ctrl_key, Go, Gi, t,
    settling_p, pade_order, N_der, Tf_min)
6940     [resp, has_opt] = simulate_selected_controller_responses_td(plant, ctrl_key, true, Go,
    Gi, t, t(2)-t(1), pade_order, N_der, Tf_min, 0);
6941     if ~has_opt
6942         metrics_t = local_nan_metrics_td();
6943         return;
6944     end
6945     metrics_t = local_step_metrics_td(resp.y_r, t, settling_p);
6946 end
6947
6948 function [has_opt, metrics_d] = get_opt_metrics_y_do_td(plant, ctrl_key, Go, Gi, t,
    settling_p, pade_order, N_der, Tf_min)
6949     [resp, has_opt] = simulate_selected_controller_responses_td(plant, ctrl_key, true, Go,
    Gi, t, t(2)-t(1), pade_order, N_der, Tf_min, 0);
6950     if ~has_opt
6951         metrics_d = local_nan_dist_metrics_td();
6952         return;
6953     end
6954     metrics_d = local_dist_metrics_td(resp.y_do, t, settling_p);
6955 end
6956
6957 function [has_opt, metrics_d] = get_opt_metrics_y_di_td(plant, ctrl_key, Go, Gi, t,
    settling_p, pade_order, N_der, Tf_min)
6958     [resp, has_opt] = simulate_selected_controller_responses_td(plant, ctrl_key, true, Go,
    Gi, t, t(2)-t(1), pade_order, N_der, Tf_min, 0);
6959     if ~has_opt
6960         metrics_d = local_nan_dist_metrics_td();
6961         return;
6962     end
6963     metrics_d = local_dist_metrics_td(resp.y_di, t, settling_p);
6964 end
6965
6966 function [has_opt, metrics_c] = get_opt_metrics_combined_td(plant, ctrl_key, Go, Gi, dt,
    settling_p, pade_order, N_der, Tf_min) %#ok<INUSD>
6967     [resp, has_opt] = simulate_selected_controller_responses_td(plant, ctrl_key, true, Go,
    Gi, 0:dt:30, dt, pade_order, N_der, Tf_min, 0);
6968     if ~has_opt
6969         metrics_c = local_nan_comb_metrics_td();
6970         return;
6971     end
6972     metrics_c = local_combined_metrics_td(resp.y_seq, resp.t_seq);
6973 end
6974
6975 function m = local_step_metrics_td(y, t, settling_p)
```

```
    y = y(:); t = t(:);
    if isempty(y) || numel(y) ~= numel(t)
        m = local_nan_metrics_td();
        return;
    end
    yss = y(end);
    m.yss = yss;
    m.ess = 1 - yss;
    m.iae_ref = trapz(t, abs(1 - y));

    if ~isfinite(yss) || abs(yss) < 1e-12
        m.rise90 = NaN; m.settle = NaN; m.overshoot_pct = NaN; m.overshoot_ratio = NaN;
        m.peak_time = NaN; m.peak_value = NaN;
        return;
    end

    yn = y / yss;
    idx_r = find(yn >= 0.9, 1, 'first');
    if isempty(idx_r), m.rise90 = NaN; else, m.rise90 = t(idx_r); end
    [peak_norm, idx_p] = max(yn);
    m.peak_time = t(idx_p);
    m.peak_value = y(idx_p);
    m.overshoot_ratio = peak_norm;
    m.overshoot_pct = max(0, (peak_norm - 1) * 100);

    band = settling_p / 100;
    low = 1 - band; high = 1 + band;
    outside = (yn < low) | (yn > high);
    last_out = find(outside, 1, 'last');
    if isempty(last_out)
        m.settle = 0;
    elseif last_out == numel(t)
        m.settle = NaN;
    else
        m.settle = t(last_out + 1);
    end
end

function m = local_dist_metrics_td(y, t, settling_p)
    y = y(:); t = t(:);
    if isempty(y) || numel(y) ~= numel(t)
        m = local_nan_dist_metrics_td();
        return;
    end
    ya = abs(y);
    [peak_abs, idxp] = max(ya);
    m.peak_abs = peak_abs;
    m.peak_time = t(idxp);
    m.yss = y(end);
    m.ess0 = abs(y(end));
    m.iae_abs = trapz(t, ya);

    if peak_abs < 1e-12
        m.settle0 = 0;
        return;
    end

    thr = peak_abs * (settling_p / 100);
    outside = ya > thr;
    last_out = find(outside, 1, 'last');
    if isempty(last_out)
        m.settle0 = 0;
    elseif last_out == numel(t)
        m.settle0 = NaN;
    else
```

```
        m.settle0 = t(last_out + 1);
    end
end

function m = local_combined_metrics_td(y, t)
    y = y(:); t = t(:);
    if isempty(y) || numel(y) ~= numel(t)
        m = local_nan_comb_metrics_td();
        return;
    end
    ya = abs(y);
    [m.peak_abs, idxp] = max(ya);
    m.peak_time = t(idxp);
    m.yfinal = y(end);
    m.iae_seq = trapz(t, abs(y));
end

function m = local_nan_metrics_td()
    m = struct('rise90',NaN,'settle',NaN,'overshoot_pct',NaN,'overshoot_ratio',NaN, ...
        'peak_time',NaN,'peak_value',NaN,'yss',NaN,'ess',NaN,'iae_ref',NaN);
end

function m = local_nan_dist_metrics_td()
    m =
struct('peak_abs',NaN,'peak_time',NaN,'settle0',NaN,'yss',NaN,'ess0',NaN,'iae_abs',NaN);
end

function m = local_nan_comb_metrics_td()
    m = struct('peak_abs',NaN,'peak_time',NaN,'yfinal',NaN,'iae_seq',NaN);
end

function [best_label, best_value] = pick_best_row_td(labels, values, mode_txt)
    labels = cellstr(string(labels));
    values = values(:);
    mask = isfinite(values);
    if ~any(mask)
        best_label = 'NA';
        best_value = NaN;
        return;
    end
    idx = find(mask);
    switch lower(mode_txt)
        case 'max'
            [best_value, idx_rel] = max(values(mask));
        otherwise
            [best_value, idx_rel] = min(values(mask));
    end
    best_label = labels{idx(idx_rel)};
end

function tag = local_deadtime_err_tag_td(e)
    if e < 0
        prefix = 'm';
    else
        prefix = 'p';
    end
    tag = sprintf('err_%s%02d', prefix, round(abs(100*e)));
end

function safe_export_dual(fig_handle, preferred_handle, png_file, pdf_file, dpi_val)
    if nargin < 5 || isempty(dpi_val)
        dpi_val = 220;
    end

    if nargin < 2 || isempty(preferred_handle) || ~isgraphics(preferred_handle)
```

```
7105          preferred_handle = fig_handle;
7106      end
7107      if nargin < 1 || isempty(fig_handle) || ~isgraphics(fig_handle)
7108          fig_handle = preferred_handle;
7109      end
7110
7111      % PNG
7112      png_ok = false;
7113      try
7114          exportgraphics(preferred_handle, png_file, 'Resolution', dpi_val);
7115          png_ok = true;
7116      catch
7117      end
7118      if ~png_ok
7119          try
7120              exportgraphics(fig_handle, png_file, 'Resolution', dpi_val);
7121              png_ok = true;
7122          catch
7123          end
7124      end
7125      if ~png_ok
7126          try
7127              saveas(fig_handle, png_file);
7128              png_ok = true;
7129          catch ME
7130              warning('No se pudo exportar PNG: %s', ME.message);
7131          end
7132      end
7133
7134      % PDF
7135      pdf_ok = false;
7136      try
7137          exportgraphics(preferred_handle, pdf_file, 'ContentType', 'vector');
7138          pdf_ok = true;
7139      catch
7140      end
7141      if ~pdf_ok
7142          try
7143              exportgraphics(fig_handle, pdf_file, 'ContentType', 'vector');
7144              pdf_ok = true;
7145          catch
7146          end
7147      end
7148      if ~pdf_ok
7149          try
7150              set(fig_handle, 'PaperPositionMode', 'auto');
7151              print(fig_handle, pdf_file, '-dpdf', '-painters');
7152              pdf_ok = true;
7153          catch ME
7154              warning('No se pudo exportar PDF: %s', ME.message);
7155          end
7156      end
7157  end
7158
7159  function lines = build_time_eval_metadata_lines_td(mst_ref_eval, allow_far_mst_eval,
      settling_p, T_end, dt, Go, Gi, pade_order, N_der, Tf_min, deadtime_err_list, idx_run,
      Plantas, runPI, runPID, runSP, runMSP, xlsx_file, fig_folder)
7160      ctrls = {};
7161      if runPI, ctrls{end+1} = 'PI'; end %#ok<AGROW>
7162      if runPID, ctrls{end+1} = 'PID'; end %#ok<AGROW>
7163      if runSP, ctrls{end+1} = 'SP'; end %#ok<AGROW>
7164      if runMSP, ctrls{end+1} = 'MSP'; end %#ok<AGROW>
7165
7166      lines = {
7167          'CODIGO TESIS CONSOLIDADO - EVALUACION TEMPORAL A Mst FIJO'
```

```
7168          ['Fecha_hora = ' datestr(now, 31)]
7169          ['mst_ref_eval = ' num2str(mst_ref_eval, '%.6f')]
7170          'j_temporal = recalculado desde IAEdo/IAEdi de la etapa temporal usando pesos base
      cargados; fallback = J_sweep si faltan pesos base'
7171          ['allow_far_mst_eval = ' ternary_txt(allow_far_mst_eval, 'YES', 'NO')]
7172          ['settling_p = ' num2str(settling_p, '%.6f')]
7173          ['T_end = ' num2str(T_end, '%.6f')]
7174          ['dt = ' num2str(dt, '%.6f')]
7175          ['Go = ' num2str(Go, '%.6f')]
7176          ['Gi = ' num2str(Gi, '%.6f')]
7177          ['pade_order = ' num2str(pade_order)]
7178          ['N_der = ' num2str(N_der)]
7179          ['Tf_min = ' num2str(Tf_min, '%.6f')]
7180          ['deadtime_err_list = ' vec2str(deadtime_err_list, '%.4f')]
7181          ['plantas = ' strjoin({Plantas(idx_run).id}, ', ')]
7182          ['controladores = ' strjoin(ctrls, ', ')]
7183          ['xlsx_file = ' xlsx_file]
7184          ['fig_folder = ' fig_folder]
7185          };
7186  end
7187
7188  function [Gdo_sp, Gry_sp, Gdi_sp, Leq_sp, is_stable] = smith_tf(P0, theta0, L_real, C, Go,
      Gi, pade_order)
7189      tol = 1e-6;
7190
7191      [numL, denL]   = pade(L_real, pade_order);
7192      [numTh, denTh] = pade(theta0, pade_order);
7193
7194      e_L   = tf(numL, denL);
7195      e_th0 = tf(numTh, denTh);
7196
7197      Leq_sp = minreal(C*P0*(1 - e_th0 + e_L), tol);
7198      D = 1 + Leq_sp;
7199
7200      Gdo_sp = minreal(Go * ((1 + C*P0*(1 - e_th0)) / D), tol);
7201      Gry_sp = minreal((C*P0*e_L) * (1/D), tol);
7202      Gdi_sp = minreal((P0*e_L*(1 + C*P0*(1 - e_th0))*Gi) * (1/D), tol);
7203
7204      is_stable = isstable(Gry_sp);
7205  end
7206
7207  function [Gdo_msp, Gry_msp, Gdi_msp, Leq_msp, is_stable, Fp] = ...
7208      msp_tf(P0, theta0, L_real, C, tipo, plant_id, Go, Gi, pade_order)
7209
7210      tol = 1e-6;
7211      [Fp, ~, ~, ~] = build_Fp_msp(tipo, plant_id, theta0);
7212
7213      [numL, denL]   = pade(L_real, pade_order);
7214      [numTh, denTh] = pade(theta0, pade_order);
7215
7216      e_L   = tf(numL, denL);
7217      e_th0 = tf(numTh, denTh);
7218
7219      Leq_msp = minreal(C*P0*(1 + Fp*(e_L - e_th0)), tol);
7220
7221      num_common = minreal(1 + C*P0*(1 - Fp*e_th0), tol);
7222      den_common = minreal(1 + C*P0*(1 + Fp*(e_L - e_th0)), tol);
7223
7224      base_frac = minreal(num_common / den_common, tol);
7225
7226      Gdo_msp = minreal(Go * base_frac, tol);
7227      Gry_msp = minreal((C*P0*e_L) / den_common, tol);
7228      Gdi_msp = minreal((P0*e_L*Gi) * base_frac, tol);
7229
7230      is_stable = isstable(Gry_msp);
```

```
end

function [Fp, beta, To, Tplant] = build_Fp_msp(tipo, plant_id, theta0)
    s = tf('s');

    switch lower(tipo)
        case 'estable'
            Tplant = get_Tplant_by_id(plant_id, tipo);
            To = 0.42*theta0 + 0.35*Tplant;
            beta = Tplant * (1 - (1 - To/Tplant)*exp(-theta0/Tplant));
            Fp = (1 + s*beta) / (1 + s*To);

        case 'integrador'
            Tplant = NaN;
            To = 2.26*theta0 + 0.14;
            beta = To + theta0;
            Fp = (1 + s*beta) / (1 + s*To);

        case 'inestable'
            Tplant = get_Tplant_by_id(plant_id, tipo);
            To = 0.1342 * Tplant * exp(2.382 * theta0 / Tplant);
            beta = Tplant * ( exp(theta0/Tplant) * (1 + To/Tplant)^2 - 1 );
            Fp = (1 + s*beta) / (1 + s*To*(2 + To/Tplant));

        otherwise
            error('Tipo de planta no reconocido para MSP: %s', tipo);
    end
end

function Tplant = get_Tplant_by_id(plant_id, tipo)
    switch upper(plant_id)
        case 'P1'
            Tplant = 1e-3;
        case {'P2','P6'}
            Tplant = 1;
        case {'P3','P7'}
            Tplant = 8;
        case 'P4'
            Tplant = 20;
        otherwise
            switch lower(tipo)
                case 'estable'
                    Tplant = 1;
                case 'inestable'
                    Tplant = 1;
                otherwise
                    Tplant = NaN;
            end
    end
end
```