Universidad Nacional Mayor de San Marcos
Universidad del Perú. Decana de América
Facultad de Ciencias Matemáticas
Escuela Profesional de Matemática

Extensión del método subgradiente para funciones cuasi-convexas en variedades Riemannianas

TESIS

Para optar el Título Profesional de Licenciado en Matemática

AUTOR

Jhon Erick RIOS CORTEGANA

ASESOR

Dr. Erik Alex PAPA QUIROZ

Lima, Perú

2025



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

Rios, J. (2025). *Extensión del método subgradiente para funciones cuasi-convexas en variedades Riemannianas*. [Tesis de pregrado, Universidad Nacional Mayor de San Marcos, Facultad de Ciencias Matemáticas, Escuela Profesional de Matemática]. Repositorio institucional Cybertesis UNMSM.

Metadatos complementarios

Datos de autor	
Nombres y apellidos	Jhon Erick Rios Cortegana
Tipo de documento de identidad	DNI
Número de documento de identidad	75715119
URL de ORCID	https://orcid.org/0009-0005-5623-7853
Datos de asesor	
Nombres y apellidos	Erik Alex Papa Quiroz
Tipo de documento de identidad	DNI
Número de documento de identidad	10451642
URL de ORCID	https://orcid.org/0000-0002-8678-6918
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	Edinson Raúl Montoro Alegre
Tipo de documento	DNI
Número de documento de identidad	09627181
Miembro del jurado 1	
Nombres y apellidos	Alex Armando Cruz Huallpara
Tipo de documento	DNI
Número de documento de identidad	40949706
Datos de investigación	

Línea de investigación	Optimización y Teoría de juegos
Grupo de investigación	OPTIMACO (Optimización Matemática y Computacional)
Agencia de financiamiento	financiamiento del proyecto PCONFIGI-B24140051-UNMSM
Ubicación geográfica de la investigación	Lima, Perú.
Año o rango de años en que se realizó la investigación	2025
URL de disciplinas OCDE	MATEMATICA PURA https://purl.org/pe-repo/ocde/ford#1.01.00

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
Universidad del Perú. DECANA DE AMÉRICA
FACULTAD DE CIENCIAS MATEMATICAS
Ciudad Universitaria - Av. Venezuela
S/N cuadra 34 Teléfono: 619-7000,
Anexo 1610
Correo Postal: 05-0021, E-mail:
eapmat@unmsm.edu.pe Lima - Perú

ESCUELA PROFESIONAL DE MATEMÁTICA

ACTA DE SUSTENTACIÓN DE TESIS PARA OPTAR EL TÍTULO PROFESIONAL DE LICENCIADO EN MATEMÁTICA
(MODALIDAD PRESENCIAL)

En la UNMSM - Ciudad Universitaria - Facultad de Ciencias Matemáticas, siendo las 12:00 horas del viernes 22 de agosto del 2025, se reunieron los docentes designados como Miembros del Jurado Evaluador de Tesis: Dr. Edinson Raúl Montoro Alegre (PRESIDENTE), Dr. Alex Armando Cruz Huallpara (MIEMBRO) y el Dr. Erick Alex Papa Quiroz (MIEMBRO ASESOR), para la sustentación de la tesis titulada: "EXTENSIÓN DEL MÉTODO SUBGRADIENTE PARA FUNCIONES CUASI-CONVEXAS EN VARIEDADES RIEMANNIANAS", presentado por el señor Bachiller JHON ERICK RIOS CORTEGANA, para optar el Título Profesional de Licenciado en Matemática.

Luego de la exposición de la tesis, el presidente invitó al expositor a dar respuesta a las preguntas formuladas.

Realizada la evaluación correspondiente por los miembros del Jurado Evaluador, el expositor mereció la aprobación de Sobresaliente con Mención con un calificativo promedio de Veinte (20).

A continuación, el Presidente del Jurado, Edinson Raúl Montoro Alegre, manifestó que el señor Bachiller JHON ERICK RIOS CORTEGANA, en vista de haber aprobado la sustentación de su tesis, será propuesto para que se le otorgue el Título Profesional de Licenciado en Matemática.

Siendo las 13:05 horas se levantó la sesión firmando para constancia la presente Acta en tres (3) copias originales.

DR. EDINSON RAÚL MONTORO ALEGRE
PRESIDENTE

DR. ALEX ARMANDO CRUZ HUALLPARA
MIEMBRO

DR. ERICK ALEX PAPA QUIROZ
MIEMBRO ASESOR

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
Universidad del Perú, Decana de América

Anexo 1

CERTIFICADO DE SIMILITUD

Yo, Dr. Erik Alex Papa Quiroz en mi condición de asesor acreditado con el Dictamen de la EP N.º 000891-2024-D-FCM/UNMSM de la tesis cuyo título es "**EXTENSIÓN DE MÉTODO SUBGRADIENTE PARA FUNCIONES CUASI-CONVEXAS EN VARIEDADES RIEMANNIANAS",** presentado por el bachiller Jhon Erick Rios Cortegana para optar el título profesional de Licenciado en Matemática, **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 10% 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 Licenciado en Matemática.

Lima,07 de setiembre de 2025

Firma del Asesor:

UNMSM Firmado digitalmente por PAPA QUIROZ Erik Alex FAU 20148092282 soft
Motivo: Soy el autor del documento
Fecha: 07.09.2025 17:16:09 -05:00

Nombres y Apellidos del asesor: Erik Alex Papa Quiroz

DNI: 10451642

Resumen

El presente trabajo de investigación tiene como finalidad extender el método subgradiente para problemas de optimización donde la función objetivo es cuasiconvexa bajo el contexto de variedades Riemannianas con curvatura seccional limitada superiormente por una constante no negativa. Se aborda en particular dos clases de variedades, el semiespacio hiperbólico de Poincaré y el espacio de las matrices simétricas definidas positivas.

Se presenta una demostración de convergencia del algoritmo subgradiente usando el subdiferenial de Greemberg-Pierskalla y ejemplos numéricos para las dos clases de variedades comparando sus resultados con los algoritmos en espacios Euclidianos. Los resultados numéricos obtenidos nos permiten concluir que la introducción de algoritmos Riemannianos son ventajosos para algunas aplicaciones comparados con los algoritmos Euclidianos .

Palabras clave: Variedades Riemannianas, Método Subgradiente, Funciones cuasi-convexas.

Abstract

The present research aims to extend the subgradient method to optimization problems where the objective function is quasiconvex, within the framework of Riemannian manifolds whose sectional curvature is bounded above by a non-negative constant. In particular, the study focuses on two classes of manifolds: the Poincaré upper half-space and the space of symmetric positive definite matrices.

A convergence proof of the subgradient algorithm is provided using the Greenberg-Pierskalla subdifferential, along with numerical examples for both classes of manifolds. The results are compared with corresponding algorithms in Euclidean spaces. The numerical findings allow us to conclude that the introduction of Riemannian algorithms offers advantages in certain applications when compared to their Euclidean counterparts

Keywords: Riemannian manifolds, Subgradient method, Quasiconvex functions.

Dedicatoria

Esta tesis va dedicada a mis padres, hermanos y a todos los que, con paciencia y sabiduría, me guiaron hasta aquí.

Agradecemientos

Quiero agradecer especialmente a mis padres, por su amor incondicional, su paciencia y su aliento constante: Amado Rios, Magda Cortegana, de igual manera a mis hermanos: Fernando, Morgan y Khaterine, por su apoyo constante durante todo este proceso, de igual manera al prof. Victor Neira, por su formación y guía durante mi etapa académica.
Me gustaría mostrar mi más profundo agradecimiento al Dr. Erik Alex Papa Quiroz , sus observaciones rigurosas y su apoyo constante durante todas las etapas de este trabajo. Su experiencia, paciencia y comprensión me han servido y motivado a seguir desarrollándome académicamente.

Agradezco a la UNMSM por el financiamiento del proyecto PCONFIGI- B24140051 que permitió la utilización de los equipos computacionales para desarrollar las implementaciones de los algoritmos.

Por último, quiero agradecer a mi alma mater, por todas las enseñanzas que me ha brindado, al igual que a sus docentes que han ayudado a tener mi formación académica.

Índice

Resumen I

Abstract II

Dedicatoria III

Agradecimientos IV

Introducción 1

I. Marco Teórico 4

1.1 Introducción a la Geometría Diferencial 4

1.2 Espacio tangente en el marco de una variedad diferenciable 6

1.3 Métricas Riemannianas en variedades diferenciables 7

1.4 Métrica Riemanniana 9

1.5 Geodésicas y Campos Paralelos 11

1.6 Ecuaciones Geodésicas 12

1.7 Relación entre la métrica Riemanniana y los símbolos de Christoffel 14

1.8 Curvatura de una variedad Riemanniana 15

1.9 Curvatura Seccional 17

1.10 Gradiente y hessiana en el espacio Euclidiano 18

1.11 Hessiana y Gradiente en una variedad Riemanniana 19

II. Plano de Poincaré 23

2.1 Variedad Riemanniana del Plano superior de Poincaré 23

2.2 Símbolos de Christofel 23

2.3 Curvatura seccional 24

2.4 Ecuación diferencial por los símbolos de Christofel 27

2.5 Ecuación diferencial de la geodésica por la ecuación de Euler-Lagrange . . 28

2.6 Solución de la ecuación diferencial: 31

2.6.1 Primer caso: $t_1 \neq s_1$ 34

2.6.2 Segundo caso: $t_1 = s_1$ 36

2.7 Velocidad de las geodésicas respecto a la métrica hiperbólica 40

2.8 Relación de las ecuaciones diferenciales de las geodésicas por símbolos de Christoffel y ecuación de Euler-Lagrange 41
2.9 Calculo de $\exp_y^{-1} x$ 45
2.9.1 Para el caso de $t_1 = t_2$ 45
2.9.2 Para el caso $t_1 \neq s_1$ 46
2.10 Aproximación numérica de las geodésicas 48
2.10.1 Solución aproximada del problema de Cauchy en el plano superior de Poincaré. 50
2.11 Distancia hiperbólica Riemanniana 57

III. Matrices simétricas definidas positivas **59**
3.0.1 Geodésica en la variedad SPD 59
3.0.2 Distancia definida en SPD 59
3.0.3 Gradiente Euclideano de $f(X)$ 60
3.0.4 Hessiano aplicado a una dirección V 60
3.0.5 Gradiente y Hessiano en $\mathbb{P}_{++}^n$ 62
3.0.6 Cálculo de la Hessiana Riemanniana 63
3.0.7 Derivada Direccional del Gradiente 63

IV. Métodos subgradientes en variedades Riemannianas **65**
4.0.8 Tamaño de paso de Armijo 65
4.1 Método subgradiente para funciones convexas 66
4.1.1 Definición del Problema y el Algoritmo del método subgradiente para funciones convexas 66
4.2 Método subgradiente para funciones cuasi-convexas 67
4.2.1 Definición del Problema y el Algoritmo para funciones cuasi convexas 69
4.2.2 Convergencia del Algoritmo de subgradiente de Greemberg-Pierskalla 70

V. Análisis de subdiferenciales para funciones matriciales en el espacio Euclidiano y en la variedad SPD **73**
5.0.3 Calculo del subdiferencial convexo en la variedad SPD 73
5.0.4 Calculo del subdiferencial de Greenberg–Pierskalla de $f(X) = (\ln \det X)^2 - \ln \det X$ en el espacio Euclídeo 74

5.0.5 Subdiferencial de Greemberg-Pierskalla de $f(X) = (\ln \det X)^2 - \ln \det X$ en la variedad SPD 76

VI. Ejemplos Computacionales **79**

6.1 Aproximación de geodésicas en el plano superior de Poincaré 79

6.1.1 Geodésicas verticales 79

6.1.2 Geodésicas Semicircunferencias 81

6.2 Método del subgradiente clásico para funciones convexas en la variedad del plano superior de Poincaré 82

6.3 Método gradiente y subgradiente para una función conevxa en el plano superior de Poincaré 90

6.4 Metodo subgradiente Greemberg-Pierskalla para funciones convexas en la variedad SPD 106

6.5 Comparación Gradiente, subgradiente en Variedades Riemannianas y Euclidianas 110

6.5.1 Plano superior de Poincaré 110

6.5.2 Matrices simétricas definidas positivas 112

6.6 Implementación del Subgradiente de Greemberg-Pierskalla 115

6.6.1 Comparacion de Subgradiente en el espacio Euclidiano y espacio Riemanniano 115

VII. Conclusiones **117**

VIII. Anexos **120**

8.1 Códigos Maple 120

8.1.1 Código Maple para las derivadas de la primera componente de las geodésicas 120

8.1.2 Cálculo de la derivada de la segunda componente de la geodésica . . 120

8.1.3 Código de método de Runge-Kutta para aproximar geodésicas . . . 120

8.1.4 Código para el calculo de la matriz hessiana en el plano superior de Poincaré en Maple 122

8.1.5 Código para la determinante Hessiana en Maple 122

8.2 Códigos para la comparación del método Gradiente en la variedad SPD y el espacio Euclidiano 122
8.2.1 main.m 123
8.2.2 Gradient.m 124
8.2.3 GradRiemann 127
8.2.4 evalf.m 129
8.2.5 evalg.m 130
8.2.6 comp.m 130
8.3 Subgradiente de Greember-Pierskalla para la variedad de matrices simétricas definidas positivas (SPD) 132
8.3.1 Caso Euclidiano 132
8.3.2 main.m 132
8.3.3 f_func.m 133
8.3.4 exp_map.m 133
8.3.5 GP_subdiff_f.m 133
8.3.6 metric_norm.m 134
8.3.7 incremental_GP_subgradient_.m 134
8.3.8 comp.m 135
8.4 Subgradiente de Greemberg-Pierskalla en el espacio Euclidiano 137
8.4.1 comp.m 137
8.4.2 Euclid_subdiff_f.m 139
8.4.3 f_func.m 139
8.4.4 generate_psd_matrix.m 139
8.4.5 incremental_Euclid_subgradient_f.m 140
8.4.6 main.m 141
8.5 Gradiente y subgradiente en el espacio superior de Poincaré 142
8.5.1 main.m 142
8.5.2 riemannianGradient.m 143
8.5.3 geodesicPoints.m 143
8.5.4 hyperbolicDistance.m 143
8.5.5 subgradientApprox 144
8.5.6 rk4Step 144

8.5.7 riemannianSubgradientMethod 144
8.5.8 riemannianGradientDescentArmijo.m 145
8.5.9 riemannianGradientDecreasing 146
8.5.10 f_val 146
8.5.11 expMap.m 147
8.5.12 euclideanGradient 147
8.5.13 comp.m 147

INTRODUCCIÒN

La estructura geométrica de las variedades Riemannianas juega un rol protagónico en el estudio de la optimización de funciones cuasi-convexas [26, 23]. En este contexto, el método subgradiente, desde su método clásico aplicado a espacios Euclidianos, ha ido transformándose hasta adaptarse a variedades Riemannianas.

Extender los métodos de optimización de espacios Euclidianos a variedades Riemannianas proporciona varias ventajas. Por ejemplo, un problema de optimización que originalmente tiene restricciones puede ser planteado como uno sin restricciones utilizando la geometría Riemanniana [15]. Este enfoque permite que problemas de optimización con funciones objetivo no convexas se transformen en problemas convexos al aplicar una métrica Riemanniana adecuada [26].

El método clásico del subgradiente es una técnica de optimización empleada para tratar funciones convexas que no son diferenciables en espacios Euclidianos y, de manera más amplia, en espacios de Hilbert. A pesar de esto, su aplicación directa en variedades Riemannianas enfrenta obstáculos debido a la curvatura propia de estas estructuras. Por este motivo, se ha requerido la adaptación del método del subgradiente para su uso en este contexto.

El desarrollo del método del subgradiente comenzó con los trabajos de Polyak et al. [1] y Shor et al. [25], quienes extendieron el método del gradiente a funciones no diferenciables. No obstante, se descubrió que estos métodos presentan características distintas al gradiente clásico, como la falta de un descenso monótono garantizado en la función objetivo, lo que motivó un estudio más profundo de sus propiedades.

Inicialmente, estos métodos fueron desarrollados en espacios Euclidianos, proporcionando nuevas perspectivas y técnicas que permitieron su extensión a espacios de Banach. En este sentido, Loewen et al. [17] investigó la formulación del subgradiente proximal en espacios de Banach, lo que resultó fundamental para la extensión de los métodos de optimización a espacios de mayor complejidad, como los espacios de Hilbert et al. [31]. Además, estos métodos se han extendido al caso de funciones cuasi-convexas [8], ampliando su aplicabilidad en problemas de optimización en estructuras geométricas no Euclidianos.

La primera extensión del método del subgradiente en variedades Riemannianas se debe a [11], quienes analizaron el algoritmo del subgradiente en estas estructuras, abordando las dificultades asociadas a la curvatura seccional. Este trabajo estableció una base teórica sólida para el desarrollo de algoritmos subgradientes en geometrías Riemannianas, explorando cómo la curvatura seccional influye en la convergencia del método.

En investigaciones recientes, [4] extendieron este enfoque para resolver problemas cuasi-convexos, considerando curvaturas seccionales no negativas. No obstante, esta restricción es limitante en diversas aplicaciones, ya que excluye variedades compactas como la esfera y otras variedades importantes en computación científica, como las variedades Grassmannianas, de Stiefel y aquellas definidas por matrices simétricas definidas positivas.

Por otro lado,el trabajo de Wang y Jen-Chih et al. [28] desarrollaron algoritmos de proyección de subgradiente para factibilidad convexa en variedades Riemannianas con curvaturas acotadas inferiormente. Posteriormente, Wang et al. [30], demostró la convergencia lineal del algoritmo de subgradiente en este contexto, lo que representa un avance significativo en el campo.

Los trabajos de Zhang et al. [32], presentan un método incremental que permite abordar problemas de gran escala de manera más eficiente, considerando funciones objetivo definidas como la suma de un gran número de componentes. Se estableció una desigualdad clave sobre la secuencia generada por el método, siempre que la curvatura seccional de la variedad es no negativa.

Finalmente, Ferreira et al. [10], analizó el método del subgradiente para problemas de optimización convexa en variedades Riemannianas completas con curvatura seccional acotada inferiormente. Se establecieron límites de iteración-complejidad del método con tamaño de paso exógeno y tamaño de paso de Polyak, mejorando resultados previos en el área.

Un problema crucial en la minimización de una función cuasi-convexa es la definición del subdiferencial adecuado. En ciertos casos, el subdiferencial puede ser vacío en el contexto de variedades Riemannianas, como se observa en [2], ejemplo 3.6. Para abordar esta dificultad, se ha extendido la definición mediante el subdiferencial de Greenberg-Pierskalla, definido como $\partial^{GP} f(x) := \{u \in T_x M : \langle u, \dot{\gamma}_{xy}(0)\rangle < 0, \forall y \in S_f(x), \forall \gamma_{xy} \in \Gamma_{xy}\}$, lo que permite garantizar la existencia de un subdiferencial no vacío para funciones cuasi-convexas.

Este trabajo se enfoca en la variedad del plano superior de Poincaré, que posee curvatura seccional negativa. Se realizarán cálculos de las geodésicas siguiendo el enfoque del trabajo de Xiangmei Wang y Chong Li et al.[29]. En la parte computacional del método del subgradiente, se aproximarán estas geodésicas mediante el método de Runge-Kutta, resolviendo las ecuaciones diferenciales asociadas. Además, se demostrará la convergencia del algoritmo para alcanzar el valor mínimo de una función cuasi-convexa en una variedad Riemanniana.

Nuestro trabajo se compone de siete capítulos más los anexos. El primer capítulo aborda los conceptos y propiedades asociados a la geometría diferencial y las variedades Riemannianas, presentando así las nociones que utilizaremos en los capítulos siguientes. En el segundo capítulo, introduciremos los conceptos y propiedades que giran en torno al plano superior

de Poincaré , abordaremos temas como su curvatura, los símbolos de Christoffel en dicha variedad. Además, una de nuestras contribuciones será el cálculo de las geodésicas, así como el de $exp_y^{-1}x$, donde proporcionaremos todos los detalles necesarios; de igual manera, brindaremos una aproximación de las geodésicas haciendo uso del método de Runge-Kutta de orden cuatro, que será de vital importancia para la implementación de nuestros algoritmos.En el capítulo tres,trataremos sobre la variedad de matrices simétricas definidas positivas, presentando sus conceptos y alcances de esta variedad con curvatura seccional negativa.De igual modo, proporcionaremos un ejemplo de función convexa dentro de este contexto y realizaremos los cálculos de la Hessiana y el gradiente en los espacios Euclidiano y Riemanniano, para compararlos posteriormente en ejemplos computacionales. En el capítulo cuatro, discutiremos el método del subgradiente en el contexto de variedades Riemannianas, tanto para funciones convexas como cuasi-convexas(subdiferencial de Greemberg-Pierskalla), junto con sus algoritmos correspondientes, y los tamaños de paso comúnmente utilizados en estos procedimientos. En el capítulo cinco proporcionaremos los cálculos necesarios para determinar el subdiferencial de Greenberg-Pierskalla para una función convexa tanto en el espacio Euclidiano como en la variedad de matrices simétricas definidas positivas(SPD).En el Capítulo Seis, discutiremos ejemplos computacionales en los cuales se presentan los resultados de los algoritmos expuestos en el capítulo precedente.En el séptimo capítulo se presentan nuestras conclusiones y sugerencias para investigaciones futuras.

I. Marco Teórico

Este capítulo tiene como objetivo exponer de manera rigurosa los conceptos y herramientas teóricas de los cuales haremos uso en capítulos posteriores. Se introducen conceptos de geometría diferencial orientada a variedades diferenciables; de este modo, abarcaremos conceptos como el de superficie, variedad diferenciable, espacio tangente, métricas Riemannianas, geodésicas, campos paralelos, conexión afín, símbolos de Christoffel, curvatura en una variedad, curvatura seccional, gradiente y hessiana en variedades.
Para los lectores interesados en profundizar sobre estos temas, los invitamos a poder revisar como referencias generales [9], [6], [26], [24].

1.1. Introducción a la Geometría Diferencial

Definición 1.1.1 *(Superficie Regular de $\mathbb{R}^n$, Definición 2.2-1 página 54 [9])*
Considerando un subconjunto $\mathfrak{S} \subset \mathbb{R}^n$, decimos que $\mathfrak{S}$ es una superficie regular en $\mathbb{R}^n$ con dimensión $m \leq n$ si, para cada punto $p \in \mathfrak{S}$, existe un entorno B de p en $\mathbb{R}^n$, un subconjunto abierto $\mathcal{U} \subset \mathbb{R}^m$, y una aplicación biyectiva $\mathfrak{X} : \mathcal{U} \to \mathfrak{S} \cap \mathcal{V}$ tal que:>

1) *$\mathfrak{X}$ es diferenciable en $\mathcal{U}$.*

2) *$\mathfrak{X}$ es homeomorfismo.*

3) *Para todo $q \in \mathcal{U}$, $d\mathfrak{X}_q : \mathbb{R}^m \to \mathbb{R}^n$ es inyectiva , donde $d\mathfrak{X}_q$ es la diferencia de $\mathfrak{X}$ en el punto q.*

Proposición 1.1.1 *(Proposición 2.2.1-2 página 13[22])*
Dado un subconjunto $\mathcal{U}$ abierto de $\mathbb{R}^n$ y una función $\mathcal{F} : \mathcal{U} \to \mathbb{R}^m$ que es diferenciable en U, donde $p \in \mathbb{R}^m$ constituye un valor regular de $\mathcal{F}$, se concluye que $\mathcal{F}^{-1}(p)$ configura una superficie regular de dimensión $n - m$.

Definición 1.1.2 *(Cambio de parámetros, Definición 2.2.2-2 página 14 [22])*
Supongamos que $\mathcal{G} \longrightarrow S$ y $\mathcal{J} \longrightarrow S$ son dos parametrizaciones de S en el entorno del punto p tal que

$$W = \mathcal{G}(U) \cap \mathcal{J}(V) \neq \emptyset.$$

La aplicación $\mathcal{G}^{-1} \circ \mathcal{J} : \mathcal{G}^{-1}(W) \longrightarrow \mathcal{J}^{-1}(W)$ es llamada cambio de parámetros.

Proposición 1.1.2 *(Proposición 2.2.2-2 página 14 [22]) Considere una superficie regular S en $\mathbb{R}^n$ con dimensión k. El cambio de parámetros definido por $\mathcal{G}^{-1} \circ \mathcal{J} : \mathcal{G}^{-1}(W) \longrightarrow \mathcal{J}^{-1}(W)$ constituye un difeomorfismo.*

La definición de variedad diferenciable que se presentará a continuación desempeña un papel fundamental en la generalización de las técnicas del cálculo diferencial a espacios de dimensión superior. Posteriormente, se ilustrará que una superficie regular constituye un ejemplo claro de tal variedad.

Definición 1.1.3 *(Variedad diferenciable, Definición 2.1 página 2 [6]) Una variedad diferenciable de dimensión n se describe por un conjunto $\mathfrak{M}$ y una colección de aplicaciones inyectivas $\mathfrak{X}_\alpha : U_\alpha \longrightarrow \mathfrak{M}$, con $\alpha \in I$ siendo un conjunto de parámetros. Estas aplicaciones están definidas en los conjuntos abiertos U_α de $\mathbb{R}^n$ y cumplen ciertas condiciones:*

1. $\mathfrak{M} = \bigcup_{\alpha \in I} \mathfrak{X}_\alpha(U_\alpha)$.
2. *Para todo par $\mathfrak{X}_\alpha, \mathfrak{X}_\beta$ con $\mathfrak{X}_\alpha(U_\alpha) \cap \mathfrak{X}_\beta(U_\beta) = W \neq \emptyset$, los conjuntos $\mathfrak{X}_\alpha^{-1}(W)$ y $\mathfrak{X}_\beta^{-1}(W)$ son abiertos en $\mathbb{R}^n$ y las aplicaciones $\mathfrak{X}_\beta^{-1} \circ \mathfrak{X}_\alpha : \mathfrak{X}_\alpha^{-1}(W) \longrightarrow \mathfrak{X}_\beta^{-1}(W)$ son diferenciables.*
3. *La familia $\{(U_\alpha, \mathfrak{X}_\alpha)\}$ es máxima con respecto a las condiciones (1) y (2).*

El par $(U_\alpha, \mathfrak{X}_\alpha)$ con $p \in \mathfrak{X}_\alpha(U_\alpha)$ se denomina una *parametrización*. Una colección $\{(U_\alpha, \mathfrak{X}_\alpha)\}$ que cumple con las condiciones 1 y 2 se denomina *estructura diferenciable de* $\mathfrak{M}$. Así, una Variedad es el conjunto $\mathfrak{M}$ equipado con una estructura diferenciable. Una estructura diferenciable sobre una variedad diferenciable $\mathfrak{M}$ inducirá de forma natural una topología en M, la cual se define a partir de: Observemos que esta topología es establecida de manera que los conjuntos $\mathfrak{X}_\alpha(U_\alpha)$ son abiertos y las aplicaciones $\mathfrak{X}_\alpha$ son continuas.

Proposición 1.1.3 *(Proposición 2.2.3 página 15 [22])*
Toda superficie regular en $\mathbb{R}^n$ con dimensión k constituye una variedad diferenciable de dimensión equivalente.

Proposición 1.1.4 *(Proposición 2.2.4 página 15 [22])*
Si $\mathfrak{M}_1$ y $\mathfrak{M}_2$ representan dos variedades diferenciables con dimensiones m_1 y m_2 respectivamente, entonces el producto cartesiano $\mathfrak{M}_1 \times \mathfrak{M}_2$ constituye una variedad cuya dimensión es $m_1 + m_2$.

Definición 1.1.4 *(Variedad de Hausdorff con base numerable, Definición 2.2.4 página 15 [22]).*
Una variedad diferenciable $\mathfrak{M}$ se denomina variedad de Hausdorff si $\mathfrak{M}$, con su topología dada, es un espacio de Hausdorff. La variedad diferenciable $\mathfrak{M}$ posee una base numerable si es posible cubrirla con un número numerable de vecindades coordenadas, es decir, si existe

una secuencia $\{\mathfrak{X}_n(U_n)\}, n \in \mathbb{N}$, *de vecindades coordenadas tal que:*

$$\mathfrak{M} = \bigcup_{n\in\mathbb{N}} \mathfrak{X}_n(U_n).$$

1.2. Espacio tangente en el marco de una variedad diferenciable

Las siguientes consideraciones conducen a una definición que extiende el concepto de vector tangente dentro del ámbito de las variedades diferenciables. En el caso de superficies dentro de $\mathbb{R}^3$, un vector tangente en un punto p de la superficie se caracteriza como el "vector de velocidad.[en] $\mathbb{R}^3$ de una curva que pertenece a la superficie y pasa por p. No obstante, en el marco de las variedades diferenciables, no se dispone de un espacio ambiente que sostenga esta interpretación. Por lo tanto, resulta indispensable establecer una propiedad definitoria del vector tangente que sustituya la idea de velocidad.

Recordemos algunas formalidades en el contexto de superficies regulares. (véase Página 7 [6]), Sea $\varepsilon > 0$ lo suficientemente pequeño y considere una curva $\eta : (-\varepsilon, \varepsilon) \to \mathbb{R}^n$ tal que:

$$\eta(t) = (\eta_1(t), \ \dots \ , \eta_n(t)),$$

con $\eta(0) = p$ y $\eta'(0) = (\eta_1'(0), \dots, \eta_n'(0)) = v \in \mathbb{R}^n$. Considere una función $\mathsf{f} : \mathbb{R}^n \to \mathbb{R}$ que es diferenciable en las proximidades de p. La función f puede restringirse a la trayectoria η para determinar la *derivada direccional* de f en la dirección del vector $v \in \mathbb{R}^n$:

$$\left.\frac{d(\mathsf{f}\circ\eta)}{dt}\right|_{t=0} = \sum_{i=1}^{n} \frac{\partial \mathsf{f}}{\partial \eta_i}(\eta(0))\frac{d\eta_i}{dt}(0) = \left(\sum_{i=1}^{n} \eta_i'(0)\left(\frac{\partial}{\partial \eta_i}\right)_p\right)\mathsf{f}.$$

En consecuencia, la derivada direccional en la dirección de v funciona como un operador aplicado a funciones diferenciables, dependiendo únicamente de v. Esta característica particular se empleará para establecer un vector tangente en variedades.

Definición 1.2.1 *(Vector tangente en un punto en el contexto de variedad diferenciable, Definición 2.6 página 7 [6])*
Sea $\mathfrak{M}$ *una variedad diferenciable. Analicemos una curva diferenciable*

$$\eta : (-\varepsilon, \varepsilon) \to \mathfrak{M},$$

tal que $\eta(0) = p$. *La derivada direccional de una función diferenciable* f *en* $\mathfrak{M}$ *a lo largo*

de η define un vector tangente en p.

donde $\eta(0) = p$ y se define $\mathscr{D}_p = \{\mathsf{f} : \mathfrak{M} \to \mathbb{R} : \mathsf{f} \text{ es diferenciable en } p\}$. Definimos el vector tangente a la trayectoria η en $t = 0$ como la aplicación $\eta'(0) : \mathscr{D}_p \to \mathbb{R}$ especificada por:

$$\eta'(0)(\mathsf{f}) \equiv \eta'(0)(\mathsf{f}) = \frac{d(\mathsf{f} \circ \eta)}{dt}\bigg|_{t=0}, \quad \mathsf{f} \in \mathscr{D}_p.$$

Un vector tangente en p es el vector tangente en $t = 0$ de alguna curva $\eta : (-\varepsilon, \varepsilon) \to \mathfrak{M}$ con $\eta(0) = p$.

Si $\mathfrak{M}$ es una superficie regular con dimensión $k \leq n$, es decir, $\mathfrak{M} \subset \mathbb{R}^n$, definimos el vector tangente en el punto p como el vector de velocidad en $\mathbb{R}^n$, es decir,

$$\eta'(0) = (\eta_1'(0), \eta_2'(0), \ldots, \eta_n'(0)).$$

Definición 1.2.2 *(Espacio tangente a una variedad diferenciable, Definición 2.4.2 página 19 [22]). El espacio tangente en un punto p a una variedad $\mathfrak{M}$, denotado como $T_p\mathfrak{M}$, se define como el conjunto de todos los vectores que son tangentes a $\mathfrak{M}$ en el punto p. Así, $T_p\mathfrak{M} = \{v \in \mathbb{R}^n : v \text{ es un vector tangente en } p\}$.*

Proposición 1.2.1 *(Proposición 2.4.1 página 20 [22])*
En el caso de que una variedad diferenciable se configure como un subconjunto abierto de $\mathbb{R}^n$, su espacio tangente en cada punto es idéntico a $\mathbb{R}^n$ mismo.

Como consecuencia de esta proposición, se tiene:

a). $T_p\mathbb{R}^n = \mathbb{R}^n, \quad T_p\mathbb{R}^n_+ = \mathbb{R}^n$.

b). Si $\mathfrak{M} = \{(\lambda, s) \in \mathbb{R}^m \times \mathbb{R}^n : s > 0\}$, entonces $T_p\mathfrak{M} = \mathbb{R}^{m+n}$.

1.3. Métricas Riemannianas en variedades diferenciables

En este subcapítulo, examinamos las métricas dentro de un espacio, pues son esenciales para medir distancias, evaluar errores y determinar la longitud de curvas, entre otros aspectos. Para una curva parametrizada en $\mathbb{R}^n$, descrita por $\eta(t) = (\eta_1(t), \eta_2(t), \ldots, \eta_n(t))$, con t contenido en un intervalo I de $\mathbb{R}$, la longitud del arco de la curva generada por $\eta(t)$ se determina mediante:

$$\ell(\eta) = \int_I \|v(t)\| dt$$

donde $v(t) = (\eta_1'(t), \eta_2'(t), \ldots, \eta_n'(t))$ denotan la norma Euclidiana. Consecuentemente, la extensión de la curva es determinada por la norma del vector de velocidad de acuerdo con la métrica estándar en $\mathbb{R}^n$.

Analicemos la situación en la cual nuestro espacio está representado por una variedad diferenciable $\mathfrak{M}$ y se define una curva sobre esta. La longitud de arco de dicha curva se determina mediante la *medición* realizada en el vector ubicado en el espacio tangente correspondiente a cada punto. Es, por tanto, imprescindible definir una métrica en el espacio tangente $T_p\mathfrak{M}$ para cada $p \in \mathfrak{M}$.

Recordemos que, además del habitual producto interno clásico:

$$\langle v, w \rangle_p = \sum_i v_i w_i,$$

se considera la matriz $G = (g_{ij})$, la cual es simétrica y definida positiva. Esta formulación surge de forma natural ante un cambio de coordenadas. Específicamente, consideremos $x = (x_1, x_2, \ldots, x_n)$ y $z = (z_1, z_2, \ldots, z_n)$ tal que $x(t) = x(z(t))$, es decir,

$$x(t) = (x_n(z_1(t), z_2(t), \ldots, z_n(t)), x_2(z_1(t), z_2(t), \ldots, z_n(t)), \ldots, x_n(z_1(t), z_2(t), \ldots, z_n(t)),$$

entonces:

$$\frac{dx_i}{dt} = \sum_{j=1}^{n} \left(\frac{\partial x_i}{\partial z_j} \frac{\partial z_j}{\partial t} \right), \quad \text{para todo } i = 1, 2, \ldots, n.$$

Denotando $v^x = (v_1^x, v_2^x, \ldots, v_n^x)$ y $v^z = (v_1^z, v_2^z, \ldots, v_n^z)$, donde $v_i^x = \frac{dx_i}{dt}$ y $v_i^z = \frac{dz_i}{dt}$, tenemos:

$$\|v^x\|^2 = (v^x, v^x) = \sum_{i=1}^{n} \left(\frac{dx_i}{dt} \right)^2.$$

Como:

$$\left(\frac{dx_i}{dt} \right)^2 = \left(\sum_{j=1}^{n} \frac{\partial x_i}{\partial z_j} v_j^z \right)^2 = \sum_{j=1}^{n} \sum_{k=1}^{n} \frac{\partial x_i}{\partial z_j} v_j^z \frac{\partial x_i}{\partial z_k} v_k^z = \sum_{j=1}^{n} \left(\sum_{k=1}^{n} \frac{\partial x_i}{\partial z_k} \frac{\partial x_i}{\partial z_j} \right) v_j^z v_k^z,$$

entonces:

$$\|v^x\|^2 = \sum_{i=1}^{n}\sum_{j=1}^{n}\left(\sum_{k=1}^{n}\frac{\partial x_i}{\partial z_j}\frac{\partial x_i}{\partial z_k}v_k^z\right)v_j^z,$$

conmutando sumandos:

$$\|v^x\|^2 = \sum_{i=1}^{n}\sum_{k=1}^{n}\left(\sum_{j=1}^{n}\frac{\partial x_i}{\partial z_j}\frac{\partial x_i}{\partial z_k}v_k^z v_j^z\right),$$

Haciendo un cambio k por i e i por k obtenemos:

$$\|v^x\|^2 = \sum_{i,j=1}^{n}\left(\sum_{k=1}^{n}\frac{\partial x_k}{\partial z_j}\frac{\partial x_k}{\partial z_i}\right)v_i^z v_j^z.$$

Definiendo $g_{ij} = \left(\sum_{k=1}^{n}\frac{\partial x_k}{\partial z_j}\frac{\partial x_k}{\partial z_i}\right)$ se tiene finalmente que:

$$(v^x, v^x) = \|v^x\|^2 = \sum_{i,j=1}^{n} g_{ij}v_i^z v_j^z = (\mathfrak{G}v^z, v^z).$$

Queda claro que un cambio en el sistema de coordenadas no altera las métricas. En efecto, si $v = G^{1/2}w$ tenemos que:

$$(v, v)_p = (\mathfrak{G}^{1/2}w, \mathfrak{G}^{1/2}w)_p = (\mathfrak{G}w, w)_p = \langle w, w\rangle_p.$$

1.4. Métrica Riemanniana

Definición 1.4.1 (Definición 2.5.1 página 23 [22]) *Dada $\mathbb{S}$,una variedad diferenciable. Una métrica Riemanniana es un mapeo que asocia a cada punto $p \in \mathfrak{M}$ un funcional $\langle\cdot,\cdot\rangle_p$,*

$$\langle\cdot,\cdot\rangle_p : T_p\mathfrak{M} \times T_p\mathfrak{M} \longrightarrow \mathbb{R},$$

satisfaciendo los siguientes criterios:

1. Para cada $p \in M$, $\langle\cdot,\cdot\rangle_p$ es un producto interno, que es bilineal, simétrico y definido positivo para cada $p \in M$.

2. $\langle\cdot,\cdot\rangle_p$ varía diferenciablemente de la siguiente manera: Consideremos $\mathfrak{X} : U \subset \mathbb{R}^n \longrightarrow \mathfrak{M}$ como un sistema de coordenadas alrededor de p, tal que $\mathfrak{X}(x_1, x_2, x_3, \ldots, x_n) = q \in \mathfrak{X}(U)$ y $\frac{\partial}{\partial x_i}(q) = d\mathfrak{X}_q(0, 0, \ldots, 1, 0, \ldots, 0)$. En este contexto, la función $g_{ij} : U \longrightarrow \mathbb{R}$ está definida

por:

$$g_{ij}(x_1, x_2, \ldots, x_n) = \left\langle \frac{\partial}{\partial x_i}(q), \frac{\partial}{\partial x_j}(q) \right\rangle_q ,$$

es diferenciable.

Las funciones g_{ij} se denominan coeficientes de la métrica Riemanniana en el sistema de coordenadas $\mathfrak{X}$, y la matriz $G = (g_{ij})$ constituye la representación matricial de dicha métrica. Dado que $\mathfrak{X}$ es un difeomorfismo, se sigue que

$$d\mathfrak{X}_q : \mathbb{R}^n \longrightarrow T_q\mathfrak{M}$$

es un isomorfismo lineal. En consecuencia, la matriz $\mathfrak{G} = (g_{ij})$ es invertible. Por lo tanto, toda métrica Riemanniana posee una representación matricial invertible.

Definición 1.4.2 (*Variedad Riemanniana*, Definición 2.5.2 página 24 [22]) *Una variedad diferenciable que admite la definición de una métrica Riemanniana recibe el nombre de variedad Riemanniana.*

En el segundo capítulo del artículo de Ferreira y Oliveira et al. [11], se presentan los siguientes conceptos.

Definición 1.4.3 *[] Sea $\mathfrak{M}$ una variedad Riemanniana completa. Sea $T_x\mathfrak{M}$ el espacio tangente de $\mathfrak{M}$ en x. La aplicación exponencial exp_x se define en $T_x\mathfrak{M}$ como:*

$$v \mapsto exp_x v := \eta(1, x, v)$$

donde η es la geodésica de $\mathfrak{M}$ tal que

$$\eta(0, x, v) = x \text{ y } \eta'(0, x, v) = v$$

Cuando la referencia al punto x no es necesaria o está implícita, la notación η_v significa que :

$$\eta'(0) = v$$

Si $||v|| = 1$ la geodésica η_v está parametrizada por longitud de arco o normalizada.

Definición 1.4.4 *Cuando $\mathfrak{M}$ está dotado de una métrica Riemanniana $\langle , \rangle$, con la norma correspondiente denotada por $||.||$, entonces $\mathfrak{M}$ es una variedad de Riemann.*

B B Recuerde que la métrica se puede utilizar para definir la longitud de curvas suaves por tramos

$\eta : [a, b] \to M$ que une x con y es decir $\eta(a) = x$ y $\eta(b) = y$ por

$$l(\eta) = \int_a^b ||\eta'(t)||dt$$

y, además, al minimizar este funcional de longitud sobre el conjunto de todas esas curvas, se obtiene una distancia Riemanniana $d(x, y)$, que induce la topología original en $\mathfrak{M}$.

Definición 1.4.5 *Considerando los puntos p y q en $\mathfrak{M}$, la distancia Riemanniana de p a q en la variedad , denotada por $d(x, y)$ es definida por :*

$$d(p, q) = \inf_{\eta} \int_a^b ||\eta'(t)||dt \tag{1}$$

donde $\eta : [a, b] \to \mathfrak{M}$ es una curva diferenciable tal que $\eta(a) = p$ y $\eta(b) = q$.

Definición 1.4.6 *Una variedad Riemanniana $\mathfrak{M}$ se denomina completa (en el sentido geodésico) si, para cada punto $p \in \mathfrak{M}$, las geodésicas emanando desde p están bien definidas para cualquier valor del parámetro $t \in \mathbb{R}$.*

Definición 1.4.7 *Además, $(\mathfrak{M}, d)$ es un espacio métrico completo y acotado y los subconjuntos cerrados son compactos.*
Sea la bola normal con centro en $p \in \mathfrak{M}$ y radio ϵ, definida por

$$B(p, \epsilon) = exp_p B(0, \epsilon), \text{ donde } B(0, \epsilon) = \left\{v \in T_p\mathfrak{M} \backslash ||v|| < \epsilon\right\}$$

y su límite, llamada esfera normal, denotada por $S(p, \epsilon)$.Además, sea $X \subset \mathfrak{M}$ un conjunto, denotamos por $\overline{X}$ y $Bd(X)$, como cierre y frontera de X, respectivamente.

1.5. Geodésicas y Campos Paralelos

La derivada covariante es fundamental para establecer el transporte paralelo a lo largo de curvas que dependen de la métrica, implicando que al modificar la métrica, se altera generalmente el método de derivar campos vectoriales; esto es crucial especialmente para identificar geodésicas, las cuales son curvas en las que el vector tangente es paralelo o cuya derivada covariante es cero. Dada una curva $\alpha : [a, b] \to \mathfrak{M}$, tal que $\eta(a) = p$ y $\alpha(b) = q$, el transporte paralelo está representado por $P_\alpha(t) : T_p\mathfrak{M} \to T_q\mathfrak{M}$, un isomorfismo lineal que satisface $P_\eta(t)(v) = V(b)$, que representa el único transporte paralelo a lo largo de η. En contexto de la métrica euclidiana, la geodésica es meramente el segmento de línea recta que vincula dos puntos p y q, definiendo así la trayectoria de mínima distancia entre ellos.

Si $\eta : [a, b] \to \mathbb{R}^n$ representa una curva diferenciable pasando por: $p = \eta(a)$ y $q = \eta(b)$, siendo el campo $\frac{d\eta}{dt}$ asociado físicamente a la velocidad, tenemos la aceleración $\frac{d}{dt}\left(\frac{d\eta}{dt}\right)$ en cada punto $\eta(t)$ con la propiedad de η ser geodésica dada por

$$\frac{d}{dt}\left(\frac{d\eta}{dt}\right) = 0.$$

La generalización de este concepto a variedades requiere únicamente que la componente covariante de la derivada sea nula.

Geodésicas

Definición 1.5.1 (Definición 2.6.5 página 30[22]) *Una curva parametrizada $\eta : I \longrightarrow \mathfrak{M}$ se denomina **geodésica** si el campo tangente $\dfrac{d\eta}{dt}$ satisface:*

$$\frac{D}{dt}\left(\frac{d\eta}{dt}\right) = 0.$$

Campos Paralelos

Dada $\mathfrak{M}$ como una variedad diferenciable, una conexión afín ∇ y un campo V definido a lo largo de una curva diferenciable $\eta : I \to \mathfrak{M}$, se dice que V es un **campo paralelo** si se cumple que $\frac{DV}{dt} = 0$ para todo $t \in I$. En consecuencia, si α es una geodésica, entonces $\dfrac{d\eta}{dt}$ es un campo paralelo.

1.6. Ecuaciones Geodésicas

La determinación de un campo paralelo V se lleva a cabo mediante las ecuaciones

$$\sum_{k=1}^{n}\left(\frac{dv^k}{dt} + \sum_{i,j=1}^{n} v^j \frac{d\eta^i}{dt}\eta_{ij}^k\right) X_k = 0,$$

o, equivalentemente,

$$\frac{dv^k}{dt} + \sum_{i,j=1}^{n} v^j \frac{d\eta^i}{dt}\eta_{ij}^k = 0, \quad k = 1, \ldots, n.$$

Cuando se trata de una geodésica $\eta(t) = (\eta_1(t), \ldots, \eta_n(t))$, se tiene $v^i = \frac{d\eta^i}{dt}$, entonces esta última ecuación se transforma en

$$\frac{d^2\eta^k}{dt^2} + \sum_{i,j=1}^{n} \eta_{ij}^k \frac{d\eta^i}{dt}\frac{d\eta^j}{dt} = 0, \quad k = 1, \ldots, n, \tag{2}$$

que representa un sistema formado por n ecuaciones diferenciales de segundo orden, asegurando una solución única en el intervalo $I = [a, b]$, cumpliendo con las condiciones $x(0) = \eta(0) = p$ y $\frac{dx}{dt}(0) = \eta'(0) = v$.

Conexión afín en variedades Riemannianas

Definición 1.6.1 (Definición 2.6.6 página 31 [22]) *Sea $\mathfrak{M}$ una variedad diferenciable equipada con una conexión afín ∇ y una métrica Riemanniana $\langle , \rangle$. Se dice que la conexión ∇ es compatible con la métrica $\langle , \rangle$ si, para cada par de campos vectoriales V y W a lo largo de la curva diferenciable $\eta : I \longrightarrow \mathfrak{M}$, cumple:*

$$\frac{d}{dt}\langle V, W\rangle = \left\langle \frac{DV}{dt}, W \right\rangle + \left\langle V, \frac{DW}{dt} \right\rangle. \tag{2.5}$$

Proposición 1.6.1 (Proposición 2.6.2 página 31 [22]) *Si la conexión afín ∇ es compatible con $\langle , \rangle$ y V, W son campos paralelos a lo largo de una curva diferenciable $\eta : I \longrightarrow \mathfrak{M}$, entonces $\langle V, W\rangle$ es constante.*

En particular, si $\eta(t) = (\eta_1(t), \ldots, \eta_n(t))$ es una geodésica, $\left\langle \frac{d\eta}{dt}, \frac{d\eta}{dt} \right\rangle$ es constante.

Proposición 1.6.2 (Proposición 2.6.3 página 31 [22]) *Sea $\mathfrak{M}$ una variedad Riemanniana. La compatibilidad de una conexión afín ∇ con $\langle , \rangle$ se establece si, y solo si:*

$$X\langle Y, Z\rangle = \langle \nabla_X Y, Z\rangle + \langle Y, \nabla_X Z\rangle, \quad \text{para todo } X, Y, Z \in \mathbf{H}.$$

Definición 1.6.2 (Definición 2.6.7 página 31 [22]) *Una conexión afín ∇ sobre una variedad diferenciable $\mathfrak{M}$ se denomina **simétrica** si:*

$$\nabla_X Y - \nabla_Y X = [X, Y],$$

donde $[X, Y] = XY - YX$.

Observaciones

1. En un sistema de coordenadas $(U, \mathfrak{X})$, la simetría de la conexión afín sugiere que$\nabla_{\frac{\partial}{\partial x_i}} \frac{\partial}{\partial x_j} = \nabla_{\frac{\partial}{\partial x_j}} \frac{\partial}{\partial x_i}$. En efecto, para todo $f \in D$,

$$X_i X_j(f) - X_j X_i(f) = \frac{\partial^2 f}{\partial x_i \partial x_j} - \frac{\partial^2 f}{\partial x_j \partial x_i} = 0.$$

2. En consecuencia, se tiene que:

$$\nabla_{X_i} X_j - \nabla_{X_j} X_i = \sum_{k=1}^{n} \left(\Gamma_{ij}^k - \Gamma_{ji}^k \right) X_k = 0.$$

   Debido a la independencia lineal de $\{X_k\}$, obtenemos:

$$\Gamma_{ij}^k = \Gamma_{ji}^k.$$

La recíproca es inmediata.

El siguiente teorema asegura tanto la existencia como la unicidad de una conexión que es simultáneamente simétrica y compatible con la métrica en una Variedad Riemanniana.

Teorema 1.6.3 (Levi-Civita, Teorema 2.6.1 página 32 [22]) *Dada una variedad Riemanniana $\mathfrak{M}$, se establece la existencia de una única conexión afín ∇ sobre $\mathfrak{M}$ que cumple las siguientes condiciones:*

(a) La conexión afín ∇ es simétrica.

(b) La conexión afín ∇ es compatible con la métrica Riemanniana.

*Esta conexión es referida como la **conexión de Riemann**.*

1.7. Relación entre la métrica Riemanniana y los símbolos de Christoffel

Para un sistema de coordenadas $(U, \mathfrak{X})$, se denominan **símbolos de Christoffel** a las funciones $\Gamma_{ij}^k : U \to \mathbb{R}$ definen los coeficientes de conexión $\nabla_{X_i} X_j = \sum_k \Gamma_{ij}^k X_k$. Se muestra que

$$\Gamma_{ij}^m = \frac{1}{2} \sum_{k=1}^{n} \left(\frac{\partial}{\partial x_i} g_{jk} + \frac{\partial}{\partial x_j} g_{ik} - \frac{\partial}{\partial x_k} g_{ij} \right) g^{km},$$

donde $g_{ij} = \left\langle \frac{\partial}{\partial x_i}, \frac{\partial}{\partial x_j} \right\rangle$ son los componentes de la matriz $\mathfrak{G}(x)$, mientras que g^{ij} representan las entradas de su inversa $\mathfrak{G}^{-1}(x)$, respectivamente.

De este modo, tomemos $\frac{\partial}{\partial x_i} = X_i$, $\frac{\partial}{\partial x_j} = X_j$ y $\frac{\partial}{\partial x_k} = X_k$. Usando el resultado siguiente:

$$\langle Z, \nabla_Y X \rangle = \frac{1}{2}\{X\langle Y, Z\rangle + Y\langle Z, X\rangle - Z\langle X, Y\rangle - \langle [X, Z], Y\rangle - \langle [Y, Z], X\rangle - \langle [X, Y], Z\rangle\} \quad (2.6)$$

tenemos:

$$\langle X_k, \nabla_{X_i} X_j \rangle = \frac{1}{2}\{X_i\langle X_j, X_k\rangle + X_j\langle X_k, X_i\rangle - X_k\langle X_i, X_j\rangle\}.$$

Como $\nabla_{X_i} X_j = \nabla_{X_j} X_i = \nabla_{X_j} X_i$ y usando la linealidad del producto interno, se tiene:

$$\sum_{l=1}^{n} \Gamma_{ij}^{l} \langle X_k, X_l \rangle = \frac{1}{2}\{X_i\langle X_j, X_k\rangle + X_j\langle X_k, X_i\rangle - X_k\langle X_i, X_j\rangle\},$$

y así:

$$\sum_{l=1}^{n} \Gamma_{ij}^{l} g_{kl} = \frac{1}{2}\left\{\frac{\partial}{\partial x_i} g_{jk} + \frac{\partial}{\partial x_j} g_{ik} - \frac{\partial}{\partial x_k} g_{ij}\right\}.$$

Denotando $b_k = \frac{1}{2}\left(\frac{\partial}{\partial x_i} g_{jk} + \frac{\partial}{\partial x_j} g_{ik} - \frac{\partial}{\partial x_k} g_{ij}\right)$, $k = 1, 2, \ldots, n$, obtenemos un sistema lineal $Gy = b$ con $y = (\Gamma_{ij}^1, \Gamma_{ij}^2, \ldots, \Gamma_{ij}^n)$ y $b = (b_1, b_2, \ldots, b_n)$. Como $G(x)$ si es invertible (consulte la definición de métrica Riemanniana), entonces $y = G^{-1}b$. Por lo tanto, obtenemos:

$$\Gamma_{ij}^{m} = \frac{1}{2} \sum_{k=1}^{n} g^{mk} b_k.$$

Finalmente sustituyendo el valor de b_k en la expresión anterior se tiene:

$$\Gamma_{ij}^{m} = \frac{1}{2} \sum_{k=1}^{n} g^{km} \left(\frac{\partial}{\partial x_i} g_{jk} + \frac{\partial}{\partial x_j} g_{ik} - \frac{\partial}{\partial x_k} g_{ij}\right). \quad (2.7)$$

1.8. Curvatura de una variedad Riemanniana

En esta parte, presentamos la definición de la curvatura asociada a una variedad Riemanniana, la cual, de manera intuitiva, mide su desvío respecto a la geometría Euclidiana. Desde el enfoque aplicado, esta sección establece que las variedades $\mathbb{R}_+^n$ y C_b^0, con la métrica definida por $\mathfrak{G}(x) = \text{diag}\left(\frac{1}{(x^1)^2}, \frac{1}{(x^2)^2}, \ldots, \frac{1}{(x^n)^2}\right)$ para cualquier función diferenciable $h : \mathbb{R}_+ \to \mathbb{R}_+$ y $h : (0, 1)^n \to \mathbb{R}_+$ respectivamente, presentan curvatura nula.

Definición 1.8.1 (Aplicación curvatura, Definición 2.7.1 página 38 [22]) *Considere* $\mathbf{A}(\mathbf{H},\mathbf{H})$ *como el conjunto de mapeos de* $\mathbf{H}$ *a* $\mathbf{H}$*, y designe* ∇ *como la conexión afín en una variedad Riemanniana* $\mathfrak{M}$*, la cual está unívocamente definida por el teorema de Levi-Civita.*
La ***curvatura*** *K en una variedad Riemanniana* $\mathfrak{M}$ *se describe como una función*

$$K : \mathbf{H} \times \mathbf{H} \longrightarrow \mathbf{A}(\mathbf{H},\mathbf{H})$$

definida por:

$$K(X,Y)Z = \nabla_X \nabla_Y Z - \nabla_Y \nabla_X Z + \nabla_{[X,Y]} Z.$$

Observación 1.8.1 (Observación 2.7.1 página 38 [22]) *Si la variedad* $M = \mathbb{R}^n$*, entonces* $K(X,Y)Z = 0$*, para todo* $X, Y, Z \in \mathbf{H}$*. En efecto, basta indicar* $Z = (z_1, z_2, \ldots, z_n)$ *las componentes del campo Z en las coordenadas naturales de* $\mathbb{R}^n$ *y la conexión definida por:*

$$\nabla_X Z = (Xz_1, Xz_2, \ldots, Xz_n)\,.$$

Observación 1.8.2 (Observación 2.7.2 página 38 [22]) *Si consideramos un sistema de coordenadas* $(U, \mathfrak{X})$ *en torno del punto p y* $\{X_i\}$*,* $i = 1, 2, \ldots, n$ *es una base de* $T_p\mathfrak{M}$*, obtenemos:*

$$K(X_i, X_j)X_k = \nabla_{X_i} \nabla_{X_j} X_k - \nabla_{X_j} \nabla_{X_i} X_k.$$

Observación 1.8.3 (Observación 2.7.3 página 38 [22]) *La curvatura K es antisimétrica. En efecto,*

$$K(X,Y)Z + K(Y,X)Z = \nabla_X \nabla_Y Z + \nabla_Y \nabla_X Z,$$

para todo $Z \in \mathbf{H}$*.*

Como $[X,Y] = -[Y,X]$*, entonces:*

$$K(X,Y)Z + K(Y,X)Z = 0, \quad \text{para todo } Z \in \mathbf{H},$$

y así,

$$K(X,Y)Z = -K(Y,X)Z.$$

Proposición 1.8.1 (Proposición 2.7.1 página 39 [22]) *En el contexto de una variedad Riemanniana, la curvatura K se caracteriza por su propiedad trilineal, en el siguiente sentido:*

a. K es bilineal en $\mathbf{H} \times \mathbf{H}$*, esto es,*

$$K(\mathrm{f}X_1 + \mathrm{g}X_2, Y_1) = \mathrm{f}K(X_1, Y_1) + gK(X_2, Y_1),$$

$$K(X_1, \mathsf{f}Y_1 + gY_2) = \mathsf{f}K(X_1, Y_1) + \mathsf{g}K(X_1, Y_2),$$

donde $\mathsf{f}, \mathsf{g} \in D(M)$ *y* $X_1, X_2, Y_1, Y_2 \in \mathbf{H}$.

b. Para todo par $X, Y \in \mathbf{H}$*, el operador curvatura* $K(X, Y) : \mathbf{H} \to \mathbf{H}$ *es lineal, esto es,*

$$K(X, Y)(Z + W) = K(X, Y)Z + K(X, Y)W,$$

$$K(X, Y)(\mathsf{f}Z) = \mathsf{f}K(X, Y)Z,$$

donde $\mathsf{f} \in D(M)$ *y* $Z, W \in \mathbf{H}$.

Proposición 1.8.2 (Proposición 2.7.2 página 38 [22]) *Consideremos* $(U, \mathfrak{X})$ *como un sistema de coordenadas alrededor de* $p \in \mathfrak{M}$ *y sea* $\{X_i\}$ *una base de* $T_p\mathfrak{M}$ *en este sistema de coordenadas. Entonces:*

$$K(X_i, X_j)X_k = \sum_{l=1}^{n} K^l_{ijk} X_l,$$

donde las componentes K^l_{ijk} *son dadas por:*

$$K^l_{ijk} = X_i\Gamma^l_{jk} - X_j\Gamma^l_{ik} + \sum_{m=1}^{n} \Gamma^m_{jk}\Gamma^l_{im} - \sum_{m=1}^{n} \Gamma^m_{ik}\Gamma^l_{jm}.$$

Observación 1.8.4 (Observación 2.7.4 página 38 [22]) *Si en las coordenadas* $(U, \mathfrak{X})$ *escribimos:* $X = \sum_{i=1}^{n} X^i X_i$, $Y = \sum_{j=1}^{n} Y^j X_j$, $Z = \sum_{k=1}^{n} Z^k X_k$, *por la linealidad de K tenemos:*

$$K(X, Y)Z = \sum_{l=1}^{n} K^l_{ijk} Z^k X_l.$$

1.9. Curvatura Seccional

Intimamente asociado con el operador de curvatura K se encuentra la curvatura *seccional* (también conocida como Riemanniana), la cual definiremos a continuación.

Sea M una variedad Riemanniana y σ un subespacio bidimensional de $T_p\mathfrak{M}$. Definimos la forma cuadrática como $Q : \sigma \to \mathbb{R}$ tal que:

$$Q(x, y) = \langle x, x\rangle\langle y, y\rangle - \langle x, y\rangle^2.$$

En términos geométricos, $\sqrt{Q(x, y)}$ denota el área del paralelogramo determinado por x y y.

Proposición 1.9.1 *[Proposición 2.7.3 página 41 [22]] Considere que $\sigma \subset T_p\mathfrak{M}$ es un subespacio de dos dimensiones, y $x, y \in \sigma$ representan dos vectores linealmente independientes. En tal caso,*

$$K(x, y) = \frac{\langle R(x, y)x, y\rangle}{Q(x, y)},$$

permanece invariante respecto a la elección de los vectores x y y.

Definición 1.9.1 (Curvatura Seccional, Definición 2.7.2 página 41[22]) *Considerando un punto $p \in M$ y $\sigma \subset T_p\mathfrak{M}$, el número*

$$K(x, y) = K(\sigma),$$

*donde $\{x, y\}$ constituye una base de σ, se denomina*curvatura seccional de M.

Si $K(x, y) \leq 0$ para todos los puntos $x, y \in \sigma$, entonces la curvatura seccional dentro de la variedad Riemanniana es **no positiva**.

Si $K(x, y) \geq 0$ para todo $x, y \in \sigma$, entonces, la curvatura seccional de la variedad Riemanniana es **no negativa**.

1.10. Gradiente y hessiana en el espacio Euclidiano

Las definiciones de gradiente y hessiano vistas en cursos de cálculo o análisis matemático se añadirán con el objetivo de poder usarlas más adelante en las aplicaciones de los algoritmos y de esta manera realizar una comparación entre los métodos gradiente y subgradiente en espacios Euclidianos y variedades Riemannianas, de este modo apoyándonos en el trabajo de Martina Magliocca [18] definimos la gradiente y el hessiano en el caso Euclidiano de la siguiente manera:

Definición 1.10.1 *(Gradiente) Sean $D \subseteq \mathbb{R}^n$ abierto, $\mathsf{h} : D \to \mathbb{R}$ derivable en $\vec{x}$. El vector cuyas componentes son las n derivadas parciales de h en $\vec{x}$ se llama **gradiente** de h en $\vec{x}$ y se escribe como*

$$\nabla \mathsf{h}(\vec{x}) = \begin{pmatrix} \mathsf{h}_{x_1}(\vec{x}) \\ \vdots \\ \mathsf{h}_{x_n}(\vec{x}) \end{pmatrix}.$$

Definición 1.10.2 *(Matriz hessiana y hessiano) La matriz $n \times n$ de segundas derivadas de h*

en $\vec{x}$ definida como

$$H\mathsf{h}(\vec{x}) = \begin{pmatrix} \frac{\partial^2 \mathsf{h}(\vec{x})}{\partial x_1^2} & \frac{\partial^2 \mathsf{h}(\vec{x})}{\partial x_2 \partial x_1} & \cdots & \frac{\partial^2 \mathsf{h}(\vec{x})}{\partial x_n \partial x_1} \\ \frac{\partial^2 \mathsf{h}(\vec{x})}{\partial x_1 \partial x_2} & \frac{\partial^2 \mathsf{h}(\vec{x})}{\partial x_2^2} & \cdots & \frac{\partial^2 \mathsf{h}(\vec{x})}{\partial x_n \partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 \mathsf{h}(\vec{x})}{\partial x_1 \partial x_n} & \frac{\partial^2 \mathsf{h}(\vec{x})}{\partial x_2 \partial x_n} & \cdots & \frac{\partial^2 \mathsf{h}(\vec{x})}{\partial x_n^2} \end{pmatrix}$$

*se llama **matriz hessiana** de f en $\vec{x}$, y a veces se denota por $D^2 f(\vec{x})$. Se llama **hessiano** de f en $\vec{x}$ al determinante de la matriz hessiana.*

1.11. Hessiana y Gradiente en una variedad Riemanniana

Dada una variedad Riemanniana $\mathfrak{M}$ y una función diferenciable $g : \mathfrak{M} \longrightarrow \mathbb{R}$, para un punto $p \in \mathfrak{M}$, la diferencial de g en p actúa como un funcional lineal sobre $T_p\mathfrak{M}$. En base al teorema de representación de Riesz, existe un único elemento, representado como $\nabla_{\mathfrak{M}} g(p) \in T_p\mathfrak{M}$, que satisface para todo $v \in T_p\mathfrak{M}$ la condición siguiente

$$dg_p(v) = \langle \nabla_{\mathfrak{M}} g(p), v \rangle \tag{3}$$

y

$$\|\nabla_M g(p)\| = \|dg_p\|,$$

En efecto, la aplicación diferencial puede describirse a través del operador del producto interno. Consecuentemente, es posible definir un campo vectorial $grad\, g : \mathfrak{M} \longrightarrow T\mathfrak{M}$ tal que

$$grad\, g(p) = \nabla_{\mathfrak{M}} g(p).$$

La expresión (2) puede ser escrita como:

$$df_p(X(p)) = \langle grad\, g(p), X(p) \rangle, \quad \text{para todo } X \in \mathrm{H},$$

y así también podemos definir una aplicación $dg : \mathrm{H} \longrightarrow \mathfrak{M}^* = \mathcal{L}(M, \mathbb{R})$, donde $\mathcal{L}(\mathfrak{M}, \mathbb{R})$ es el conjunto de funciones en $\mathfrak{M}$ en $\mathbb{R}$, tal que:

$$dg(X) = \langle grad\, g, X\rangle.$$

Además, $df_p(X(p)) = \frac{d}{dt}(f \circ \eta)\big|_{t=0}$ para alguna curva $\eta : I \longrightarrow \mathfrak{M}$ con $\eta(0) = p$ y $\eta'(0) = X(p)$, luego tenemos que

$$dg_p(X(p)) = \sum_{i=1}^{n} \eta_i'(0)\frac{\partial g}{\partial x_i}(p) = X(g)(p),$$

por tanto

$$dg(X) = \langle grad\, g, X\rangle = X(g).$$

Así llegamos a la siguiente definición.

Definición 1.11.1 (Gradiente,Definición 2.8.1 página 42 [22]) *El gradiente de una función diferenciable $g : \mathfrak{M} \to \mathbb{R}$ es un campo vectorial*

$$\operatorname{grad} g : \mathfrak{M} \to T\mathfrak{M}$$

métricamente equivalente a la diferencial, esto es,

$$dg_p(X(p)) = \langle \operatorname{grad} g(p), X(p)\rangle = X(p)g, \quad \textit{para todo } X \in \mathrm{H}.$$

Observación 1.11.1 *[Observación 2.8.1 página 38 [22]] Sea $\mathfrak{M} \subset \mathbb{R}^n$ una variedad Riemanniana con la métrica definida por*

$$\langle v, w\rangle_x = v^T \mathfrak{G}(x) w,$$

donde $\mathfrak{G}(x)$ es una matriz simétrica definida positiva. Se puede caracterizar el campo gradiente como:

$$\operatorname{grad} f(q) = \mathfrak{G}^{-1}(q) f'(q),$$

donde $\mathfrak{G}^{-1}(q) = (g^{ij}(q))$ es la matriz inversa de $\mathfrak{G}(q)$ y

$$f' = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n}\right)$$

es el vector de derivadas parciales de la función f con respecto a $\mathcal{X}$. En efecto,

$$df_q(v) = f'(q)^T v = f'(q)^T(\mathfrak{G}(q)^{-1})^T\mathfrak{G}(q)v = (\mathfrak{G}(q)^{-1}f'(q))^T\mathfrak{G}(q)v = \langle \mathfrak{G}(q)^{-1}f'(q), v\rangle_q.$$

Definición 1.11.2 (Punto crítico, Definición 2.8.2 página 44[22]) *Sea $\mathfrak{M}$ una variedad Riemanniana y $p \in \mathfrak{G}$. Decimos que p es punto crítico si $gradf(p) = 0$.*

Definición 1.11.3 (Hessiana de una función, Definición 2.8.3 página 44[22]) *Considerando $f : \mathfrak{M} \longrightarrow \mathbb{R}$ como una función de clase C^k, con $k \geq 2$, la hessiana de f, simbolizada por H^f, se define como la derivada covariante del campo del gradiente, es decir,*

$$H^f = \frac{D}{dt}(gradf).$$

De esta manera, la matriz Hessiana en el punto p, en la dirección del vector $v \in T_p\mathfrak{M}$, se expresa como:

$$H_p^f(v) = \frac{D}{dt}(gradf)\bigg|_p = \nabla_v gradf(p).$$

A partir del concepto de hessiana, podemos establecer las aplicaciones $H_p^f : T_p\mathfrak{M} \to T_p\mathfrak{M}$ y $H^f : \mathfrak{M} \to \mathcal{L}(T\mathfrak{M}, T\mathfrak{M})$. Aquí, $\mathcal{L}(T\mathfrak{M}, T\mathfrak{M})$ representa el conjunto de transformaciones lineales de $T\mathfrak{M}$ en sí mismo, y se tiene que $H^f(p) = H_p^f \in \mathcal{L}(T_p\mathfrak{M}, T_p\mathfrak{M})$.

Proposición 1.11.1 (Proposición 2.8.1 página 44[22]) *Para cada $p \in \mathfrak{M}$, el operador $H_p^f : T_p\mathfrak{M} \longrightarrow T_p\mathfrak{M}$ es lineal y autoadjunto, esto es,*

$$\langle H_p^f(v), w\rangle_p = \langle v, H_p^f(w)\rangle_p.$$

Basándonos en la Proposición anterior, para cada $p \in \mathfrak{M}$, es posible definir una forma cuadrática $q_p^f : T_p\mathfrak{M} \times T_p\mathfrak{M} \longrightarrow \mathbb{R}$ representada por:

$$q_p^f(v, w) = \langle H_p^f v, w\rangle_p.$$

De manera más general, podemos definir el mapeo $q^f : \mathrm{H} \times \mathrm{H} \to \mathcal{L}(\mathfrak{M}, \mathbb{R})$ como sigue:

$$q^f(X, Y) = \langle \nabla_X gradf, Y\rangle. \tag{2.13}$$

La función especificada en (2.13) muestra la desventaja de depender de la métrica y la conexión. Puesto que la métrica determina una conexión afín, como se describe en el Teorema de Levi Civita, la subsiguiente proposición resulta fundamental para alcanzar una

caracterización adecuada.

Proposición 1.11.2 *[Proposición 2.8.2 página 44[22]] Para todo* $X, Y \in \mathrm{H}$*,*

$$q^f(X,Y) = (XY - \nabla_X Y)f = (YX - \nabla_Y X)f.$$

Observación 1.11.2 *[Observación 2.8.2 página 45[22]] Para un sistema de coordenadas* $(\mathcal{X}, U)$ *asociados a la base* $\{X_k\}$*, tenemos:*

$$q^f(X_i, X_j) = \left(X_i X_j - \sum_{m=1}^{n} \Gamma_{ij}^m X_m \right) f,$$

esto es:

$$q^f(X_i, X_j) = \left(H_p^f X_i, X_j \right) = \left(\frac{\partial^2 f}{\partial x_i \partial x_j} - \sum_{m=1}^{n} \Gamma_{ij}^m \frac{\partial f}{\partial x_m} \right). \tag{2.14}$$

Corolario 1 *[Corolario 2.8.1 página 46[22]] Si* $p \in M$ *es un punto crítico de* f *y* $X, Y \in \mathrm{H}$*, entonces:*

$$H_p^f(X(p), Y(p)) = X(p)Y(p)f.$$

Demostración 1.11.1

$$H_p^f(X(p), Y(p)) = X(p)(Y(p)f) - \langle \nabla_{X(p)} Y(p), \operatorname{grad} f(p) \rangle,$$

y como $\operatorname{grad} f(p) = 0$*.*

II. Plano de Poincaré

2.1. Variedad Riemanniana del Plano superior de Poincaré

Consideremos el modelo del semiplano superior de Poincaré

$$\mathbb{H}^2 = \{(u, v) \in \mathbb{R}^2 : v > 0\}$$

dotado de la métrica Riemanniana definida para cada $(u, v) \in \mathbb{H}^2$ por

$$g_{ij}(u, v) = \frac{1}{v^2}\delta_{ij}, \quad i, j = 1, 2$$

Es fácil probar que esta variedad $(\mathbb{H}^2, g)$ es una variedad Riemanniana, donde $T_x\mathbb{H}^2 = \mathbb{R}^2$.

2.2. Símbolos de Christofel

Sea $x = (x_1, x_2)$, entonces:

$$g_{ij}(x) = \left\langle \frac{\partial}{\partial x_i}, \frac{\partial}{\partial x_j} \right\rangle_x = \left\langle e_i,\ e_j \right\rangle_x = \frac{1}{x_2^2}\delta_{ij}$$

Las componentes de la métrica son:

$$g_{11}(x) = \frac{1}{x_2^2}, \quad g_{12}(x) = 0, \quad g_{21}(x) = 0, \quad g_{22}(x) = \frac{1}{x_2^2}$$

La matriz de la métrica es:

$$\mathfrak{G} = \begin{pmatrix} \frac{1}{x_2^2} & 0 \\ 0 & \frac{1}{x_2^2} \end{pmatrix}$$

y su inversa:

$$\mathfrak{G}^{-1} = \begin{pmatrix} x_2^2 & 0 \\ 0 & x_2^2 \end{pmatrix}$$

Para calcular los símbolos de Christoffel, utilizamos la fórmula:

$$\Gamma_{ij}^m = \frac{1}{2}\sum_{k=1}^{2}\left(\frac{\partial g_{jk}}{\partial x^i} + \frac{\partial g_{ik}}{\partial x^j} - \frac{\partial g_{ij}}{\partial x^k}\right) g^{km}$$

Calculamos los símbolos de Christoffel de la siguiente manera:

1. Para Γ^1_{11}:

$$\Gamma^1_{11} = \frac{1}{2}\left(\frac{\partial g_{11}}{\partial x^1} + \cancel{\frac{\partial g_{11}}{\partial x^1}} - \cancel{\frac{\partial g_{11}}{\partial x^1}}\right) g^{11} + \frac{1}{2}\left(\cancel{\frac{\partial g_{12}}{\partial x^1}} + \cancel{\frac{\partial g_{21}}{\partial x^1}} - \frac{\partial g_{11}}{\partial x^2}\right) g^{21}$$

Notamos que:

$$\frac{\partial g_{11}}{\partial x^1} = 0 \quad \text{y} \quad \frac{\partial g_{11}}{\partial x^2} = -\frac{2}{x_2^3}, \quad g^{21} = 0$$

Por lo tanto:

$$\Gamma^1_{11} = 0$$

2. Para Γ^2_{11}:

$$\Gamma^2_{11} = \frac{1}{2}\left(\frac{\partial g_{12}}{\partial x^1} + \frac{\partial g_{12}}{\partial x^1} - \frac{\partial g_{11}}{\partial x^2}\right) g^{12} + \frac{1}{2}\left(\frac{\partial g_{11}}{\partial x^1} + \frac{\partial g_{11}}{\partial x^1} - \frac{\partial g_{11}}{\partial x^2}\right) g^{22}$$

Notamos que:

$$\frac{\partial g_{11}}{\partial x^2} = -\frac{2}{x_2^3}, \quad g^{12} = 0, \quad g^{22} = x_2^2$$

Por lo tanto:

$$\Gamma^2_{11} = \frac{1}{2} \cdot \frac{2}{x_2^3} \cdot x_2^2 = \frac{1}{x_2}$$

Los otros símbolos de Christoffel se calculan de manera similar:

$$\Gamma^1_{22} = 0, \quad \Gamma^2_{22} = -\frac{1}{x_2}, \quad \Gamma^1_{21} = -\frac{1}{x_2}, \quad \Gamma^2_{21} = 0, \quad \Gamma^1_{12} = -\frac{1}{x_2}, \quad \Gamma^2_{12} = 0$$

2.3. Curvatura seccional

El tensor de curvatura se define como:

$$K(X,Y)Z = \nabla_Y \nabla_X Z - \nabla_X \nabla_Y Z + \nabla_{[X,Y]} Z,$$

donde ∇ es la conexión afin en una variedad Riemanniana M, dada por el teorema de *Levi-Civita*.

Dado un espacio vectorial V, denotamos por $Q(x, y)$ la expresión

$$\langle x, x\rangle\langle y, y\rangle - \langle x, y\rangle^2,$$

la cual representa el área de un paralelogramo bidimensional determinado por el par de vectores $x, y \in V$.

Sea $\sigma \subset T_{(x_1,x_2)}\mathbb{H}^2$ un subespacio bidimensional del espacio tangente $T_{(x_1,x_2)}\mathbb{H}^2$, y sean $x, y \in \sigma$ dos vectores linealmente independientes. Entonces, la curvatura seccional de $\mathbb{H}^2$ es dada por

$$k(x,y) = \frac{\langle K(x,y)x, y\rangle}{Q(x,y)}, \tag{4}$$

donde $\{x, y\}$ es una base de $\sigma \subset T_{(x_1,x_2)}\mathbb{H}^2$, ver

Recordemos que:

$$Q(x,y) = \langle x,x\rangle\langle y,y\rangle - \langle x,y\rangle^2$$

Como $\mathbb{H}^2$ es abierto de $\mathbb{R}^2$ y estamos considerando la parametrización identidad:

$$\Rightarrow \quad x = \frac{\partial}{\partial x_1}, \quad y = \frac{\partial}{\partial x_2} \quad \text{es una base de } T_{(x_1,x_2)}\mathbb{H}^2$$

Luego:

$$K\left(\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}\right)\frac{\partial}{\partial x_1} = \underbrace{\nabla_{\frac{\partial}{\partial x_2}}\nabla_{\frac{\partial}{\partial x_1}}\frac{\partial}{\partial x_1}}_{A} - \underbrace{\nabla_{\frac{\partial}{\partial x_1}}\nabla_{\frac{\partial}{\partial x_2}}\frac{\partial}{\partial x_1}}_{B}$$

$$\Rightarrow \quad \nabla_{\frac{\partial}{\partial x_1}}\frac{\partial}{\partial x_1} = \Gamma_{11}^1\frac{\partial}{\partial x_1} + \Gamma_{11}^2\frac{\partial}{\partial x_2} = \frac{1}{x_2}\frac{\partial}{\partial x_2}$$

$$\nabla_{\frac{\partial}{\partial x_2}}\frac{\partial}{\partial x_1} = \Gamma_{21}^1\frac{\partial}{\partial x_1} + \Gamma_{21}^2\frac{\partial}{\partial x_2} = -\frac{1}{x_2}\frac{\partial}{\partial x_1}$$

Ahora:

$$A = \nabla_{\frac{\partial}{\partial x_2}}\left(\frac{1}{x_2}\frac{\partial}{\partial x_2}\right) = \frac{1}{x_2}\nabla_{\frac{\partial}{\partial x_2}}\frac{\partial}{\partial x_2} + \frac{\partial}{\partial x_2}\left(\frac{1}{x_2}\right)\frac{\partial}{\partial x_2}$$

$$= \frac{1}{x_2}\left(-\frac{1}{x_2}\right)\frac{\partial}{\partial x_2} - \frac{1}{x_2^2}\frac{\partial}{\partial x_2}$$

$$= -\frac{2}{x_2^2}\frac{\partial}{\partial x_2}$$

$$B = \nabla_{\frac{\partial}{\partial x_1}}\left(-\frac{1}{x_2}\frac{\partial}{\partial x_1}\right) = -\frac{1}{x_2}\nabla_{\frac{\partial}{\partial x_1}}\frac{\partial}{\partial x_1} + \frac{\partial}{\partial x_1}\left(-\frac{1}{x_2}\right)\frac{\partial}{\partial x_1}$$

$$= -\frac{1}{x_2}\left(\frac{1}{x_2}\frac{\partial}{\partial x_2}\right) = -\frac{1}{x_2}\left(\frac{1}{x_2}\frac{\partial}{\partial x_2}\right)$$

$$= -\frac{1}{x_2}\frac{\partial}{\partial x_2}$$

Por lo tanto:

$$K\left(\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}\right)\frac{\partial}{\partial x_1} = A - B = -\frac{2}{x_2^2}\frac{\partial}{\partial x_2} - \left(-\frac{1}{x_2^2}\frac{\partial}{\partial x_2}\right)$$

$$= -\frac{1}{x_2^2}\frac{\partial}{\partial x_2}$$

Ahora calculamos:

$$\left\langle K\left(\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}\right)\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}\right\rangle = -\frac{1}{x_2^2}\left\langle \frac{\partial}{\partial x_2}, \frac{\partial}{\partial x_2}\right\rangle = -\frac{1}{x_2^2}\cdot\frac{1}{x_2^2} \tag{I}$$

Calculamos:

$$Q\left(\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}\right) = \left\|\frac{\partial}{\partial x_1}\right\|^2\left\|\frac{\partial}{\partial x_2}\right\|^2 - \left\langle \frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}\right\rangle^2$$

$$= \frac{1}{x_2^2}\cdot\frac{1}{x_2^2} \tag{II}$$

Reemplazando en ($\triangle$) se tiene que:

$$K\left(\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}\right) = -\frac{1}{x_2^2}\cdot\left(\frac{1}{x_2^2}\right)^{-1} = -1$$

2.4. Ecuación diferencial por los símbolos de Christofel

Para hallar las geodésicas se necesita resolver la ecuación diferencial visto en 2, vinculados a los símbolos de Christoffel. Así, considerando que la matriz inversa de la matriz asociada al tensor métrico en un punto p y una base B en la superficie es proporcionada por

$$\big(MB(g_p)\big)^{-1} = \begin{bmatrix} y^2 & 0 \\ 0 & y^2 \end{bmatrix}$$

Entonces, $g^{11} = g^{22} = y^2$, $g^{12} = g^{21} = 0$ y como $g_{11} = g_{22} = \frac{1}{y^2}$, $g_{12} = g_{21} = 0$; los símbolos de Christoffel quedan de la siguiente manera:

$$\Gamma^2_{22} = \frac{1}{2y^2} y^2 \left[\frac{\partial}{\partial y} \left(\frac{1}{y^2} \right) \right] = -\frac{1}{y}$$

y de manera similar se obtienen:

$$\Gamma^1_{12} = \Gamma^1_{21} = \Gamma^2_{11} = \Gamma^2_{12} = \frac{1}{y} \quad \text{y} \quad \Gamma^1_{11} = \Gamma^2_{11} = 0$$

Por lo tanto, aplicando la definición siguiente: Considérese $\eta(t) = (x^1(t), x^2(t))$, que representa una parametrización de una curva en un entorno coordenado U de una Superficie S. Se dice que α es una geodésica si y solo si

$$\frac{d^2x^i}{dt^2} + \sum_{j,k} \Gamma^i_{jk} \frac{dx^j}{dt} \frac{dx^k}{dt} = 0, \quad i = 1, 2 \tag{5}$$

donde

$$\Gamma^i_{jk} = \frac{1}{2} \sum_{m=1}^{n} g^{im} \left(\frac{\partial g_{mj}}{\partial x^k} + \frac{\partial g_{mk}}{\partial x^j} - \frac{\partial g_{jk}}{\partial x^m} \right),$$

(con $i, j, k = 1, 2$) son denominados dos símbolos de Christoffel y los g^{im}_p son los componentes de la matriz inversa a $M_B(g_p)$. Observe que Γ^i_{jk} es una función diferenciable.

Las geodésicas se hallan resolviendo el sistema de ecuaciones:

$$\frac{d^2x}{dt^2} - \frac{2}{y} \cdot \frac{dx}{dt} \cdot \frac{dy}{dt} = 0 \tag{6}$$

y

$$\frac{d^2y}{dt^2} + \frac{1}{y} \cdot \left(\frac{dx}{dt} \right)^2 - \frac{1}{y} \cdot \left(\frac{dy}{dt} \right)^2 = 0 \tag{7}$$

Para verificar la equivalencia entre el sistema anterior y la ecuación (17), se considera que

$$\frac{dy}{dx} = \frac{\frac{dy}{dt}}{\frac{dx}{dt}}, \quad \frac{d^2y}{dx^2} = \frac{\frac{d^2y}{dt^2}}{\left(\frac{dx}{dt}\right)^2} + \frac{d}{dt}\left(\frac{dy}{dx}\right)\frac{dy}{dt}$$

Consecuentemente, al sustituir ambas ecuaciones iguales y reemplazar el valor de $\frac{d^2x}{dt^2}$ en la segunda ecuación, el sistema original se transforma en:

$$\frac{d^2y}{dx^2} + \frac{2}{y} \cdot \left(\frac{dy}{dx}\right) \cdot \frac{dx}{dt} + \left(\frac{dy}{dx}\right)^2 + \frac{1}{y} = 0$$

dado que $\frac{dx}{dt}$ no se anula para ningún $t \in \mathbb{R}$, por consiguiente, simplificando los términos semejantes, la ecuación se simplifica a:

$$\frac{d^2y}{dx^2} + \frac{1}{y} \cdot \left(\frac{dy}{dx}\right)^2 + \frac{1}{y} = 0$$

2.5. Ecuación diferencial de la geodésica por la ecuación de Euler-Lagrange

A partir de la investigación desarrollada en esta tesis, he identificado una oportunidad para extender sus resultados al contexto del trabajo de Wang, Xiangmei and Li, Chong and Yao en *"On some basic results related to affine functions on Riemannian manifolds"* [29] , lo cual constituye el principal aporte de esta investigación. El enfoque adoptado permite ampliar estos cálculos para aplicarlos a capítulos posteriores.
Estudiaremos las ecuaciones de Euler-Lagrange para el funcional f que ayudará al cálculo de las geodésicas del modelo del plano superior de Poincaré. De esta manera, definimos las ecuaciones de Euler-Lagrange como:

$$\frac{\partial f}{\partial y} - \frac{d}{dx}\left(\frac{\partial f}{\partial y'}\right) = 0 \tag{8}$$

A continuación, presentaremos los conceptos básicos relacionados con los funcionales, ya que para abordar problemas de cálculo variacional es de suma importancia identificar los extremos de estos funcionales. Un funcional se define como una función que toma como dominio un conjunto de funciones. Examinamos, por ejemplo, la siguiente integral:

$$L = \int_a^b f(x)dx$$

La función L depende exclusivamente de la función $f(x)$, por lo tanto, $L = L(f)$ constituye un funcional, cuyo dominio comprende el conjunto de todas las funciones que son diferenciables. De igual manera, la función,

$$g = g(y, z)$$

donde $y = y(x)$ y $z = z(x)$ para todo $x \in \mathbb{R}$. Esto sugiere que g no es meramente una función, sino más bien un funcional que depende de dos funciones.

Ahora bien, para calcular las geodésicas en $\mathbb{H}^2$, hay que partir de la definición de longitud de arco:

$$s = \int_a^b \sqrt{\sum_{i,j=1}^{2} |g_{ij}(p)x_i'(t)x_j'(t)|}\, dt \tag{9}$$

y la caracterización del espacio hiperbólico bidimensional o Plano Hiperbólico $\mathbb{H}^2$ como el conjunto de todos los puntos (x_1, x_2) donde $x_2 > 0$, $x_i \in \mathbb{R}$, con la métrica definida por la expresión:

$$ds^2 = \frac{1}{x_2^2}\left(dx_1^2 + dx_2^2\right) \tag{10}$$

Observamos que los componentes de la métrica g presentan dependencia respecto a la coordenada x_2 del punto p al cual se está aplicando. En otras palabras, $M_B(g_p)$ es diagonal con respecto a la base

$$B = \left\{ \frac{\partial}{\partial x^1}\Big|_p, \frac{\partial}{\partial x^2}\Big|_p \right\}$$

del espacio tangente a $\mathbb{H}^2$ en p:

$$M_B(g_p) = \begin{bmatrix} \frac{1}{x_2^2} & 0 \\ 0 & \frac{1}{x_2^2} \end{bmatrix},$$

y se puede notar que $g_{12} = 0$ y $g_{11} = g_{22} = \frac{1}{x_2^2}$.
Por lo tanto, considerando (9) y (10) y la curva C^2 diferenciable $\beta(t) = (x(t), y(t))$, la ecuación (5) puede ser escrita como sigue:

$$s = \int_a^b \frac{\sqrt{1 + \left(\frac{dy/dt}{dx/dt}\right)^2}}{y(t)} \frac{dx}{dt} \cdot dt \tag{11}$$

o de forma equivalente:

$$s = \int_{x_0}^{x_1} \frac{\sqrt{1 + (y'(x))^2}}{y} dx \tag{12}$$

donde $x_0 = x(a)$, $x_1 = x(b)$ y $y'(x) = \frac{dy}{dx}(x)$.

Partiendo de la ecuación obtenida, se establece $f(y, y')$ de la siguiente manera:

$$f(y, y') = \frac{\sqrt{1 + (y')^2}}{y} \tag{13}$$

Por lo tanto, la ecuación (13) se reduce a:

$$s = \int_{x_0}^{x_1} f(y, y') dx$$

$$f(y, y') = \frac{\sqrt{1 + (y')^2}}{y}$$

depende únicamente de y y y', permitiendo el cálculo de sus derivadas parciales respectivas, resultando en

$$\frac{\partial f}{\partial y} = -\frac{\sqrt{1 + (y')^2}}{y^2} \tag{14}$$

$$\frac{\partial f}{\partial y'} = \frac{1}{y} \frac{y'}{\sqrt{1 + (y')^2}} \tag{15}$$

Debido a que $y > 0$, se afirma que el funcional f, así como sus derivadas parciales, están bien definidas en todos los puntos de sus respectivos dominios.

Aplicando la definición de la ecuación de *Euler-Lagrange* (8) las soluciones de la ecuación

$$\frac{\partial f}{\partial y} - \frac{d}{dx}\left(\frac{\partial f}{\partial y'}\right) = 0 \tag{16}$$

constituyen la curva más corta que conecta dos puntos en el plano hiperbólico definido por la métrica (10).

Para abordar la resolución de la ecuación (8) en relación con el funcional descrito en (13), se emplean las equivalencias en (14) y (15).

$$\frac{y''(y')^2}{y(1+(y')^2)^{3/2}} - \frac{y''}{y\sqrt{1+(y')^2}} + \frac{(y')^2}{y^2\sqrt{1+(y')^2}} - \frac{\sqrt{1+(y')^2}}{y^2} = 0$$

Multiplicando ambos lados de la ecuación por $y^2(1+(y')^2)^{3/2}$ y reduciendo términos semejantes, se llega a la ecuación diferencial ordinaria:

$$y'' + \frac{1}{y}(y')^2 + \frac{1}{y} = 0 \tag{17}$$

O de forma equivalente,

$$\frac{d^2y}{dx^2} + \frac{1}{y}\left(\frac{dy}{dx}\right)^2 + \frac{1}{y} = 0 \tag{18}$$

tomemos en cuenta que

$$\frac{dy}{dt} = \frac{dy}{dx}\frac{dx}{dt}$$

y

$$\frac{d^2y}{dt^2} = \frac{d^2y}{dx^2}\left(\frac{dx}{dt}\right)^2 + \frac{dy}{dx}\frac{d^2x}{dt^2}. \tag{19}$$

2.6. Solución de la ecuación diferencial:

La ecuación diferencial que se plantea en (18) se soluciona de la siguiente manera:

$$\frac{d^2y}{dx^2} + \frac{1}{y}\left(\frac{dy}{dx}\right)^2 + \frac{1}{y} = 0$$

Multiplicamos por y para simplificar los términos:

$$y\frac{d^2y}{dx^2} + \left(\frac{dy}{dx}\right)^2 + 1 = 0$$

Definimos $v = \dfrac{dy}{dx}$ y, por tanto, $\dfrac{d^2y}{dx^2} = \dfrac{dv}{dx}$. Esto transforma la ecuación en:

$$y\frac{dv}{dx} + v^2 + 1 = 0$$

Usamos la regla de la cadena para expresar $\dfrac{dv}{dx}$ en función de y:

$$\frac{dv}{dx} = v\frac{dv}{dy}$$

Sustituyendo en la ecuación anterior:

$$yv\frac{dv}{dy} + v^2 + 1 = 0$$

Separamos variables:

$$\frac{v}{v^2+1}dv = -\frac{1}{y}dy$$

Integramos ambos lados:

$$\int \frac{v}{v^2+1}dv = \frac{1}{2}\ln(v^2+1)$$

$$\int -\frac{1}{y}dy = -\ln|y|$$

Por lo tanto, obtenemos:

$$\frac{1}{2}\ln(v^2+1) = -\ln|y| + C$$

Multiplicamos ambos lados por 2:

$$\ln(v^2+1) = -2\ln|y| + 2C$$

Exponenciamos ambos lados:

$$v^2 + 1 = e^{2C} \cdot y^{-2}$$

Denotamos $e^{2C} = A$, por lo que:

$$v^2 + 1 = \frac{A}{y^2}$$

Despejamos v^2:

$$v^2 = \frac{A}{y^2} - 1$$

Por lo tanto:

$$v = \frac{dy}{dx} = \pm\sqrt{\frac{A}{y^2} - 1}$$

Separamos las variables:

$$\frac{dy}{\sqrt{\frac{A}{y^2} - 1}} = \pm dx$$

Multiplicamos ambos lados por y para simplificar el denominador:

$$\frac{y\,dy}{\sqrt{A - y^2}} = \pm dx$$

Integramos ambos lados:

$$\int \frac{y\,dy}{\sqrt{A - y^2}} = -\sqrt{A - y^2}$$

Obtenemos:

$$-\sqrt{A - y^2} = \pm x + C_1$$

Elevamos ambos lados al cuadrado:

$$A - y^2 = (x + C_1)^2$$

Finalmente, la solución es:

$$y^2 = A - (x + C_1)^2$$

Expandiendo, obtenemos la forma cuadrática:

$$x^2 + y^2 + 2C_1x - A + C_1^2 = 0$$

Cuya solución es

$$x^2 + y^2 - 2bx - a = 0 \tag{20}$$

, donde $b = -C_1$ y $a = A - C_1^2$. donde a y b son constantes que se establecen según las condiciones iniciales. Procederemos a resolver el caso en el que se proporcionan dos puntos a través de los cuales pasa la geodésica. Por tanto, al considerar dichos puntos $x = (t_1, t_2)$ y $y = (s_1, s_2) \in \mathbb{H}^2$

2.6.1. Primer caso: $t_1 \neq s_1$

Para este caso si definimos la ecuación cuadrática de la semicircunferencias (20), a su forma canónica tenemos:

Reemplazando (t_1, t_2) en (20):

$$t_1^2 + t_2^2 - 2bt_1 - a = 0$$

Reemplazando (s_1, s_2) en (20):

$$s_1^2 + s_2^2 - 2bs_1 - a = 0$$

Finalmente, tenemos el sistema de ecuaciones:

$$\begin{cases} t_1^2 + t_2^2 - 2bt_1 - a = 0 & \text{(I)} \\ s_1^2 + s_2^2 - 2bs_1 - a = 0 & \text{(II)} \end{cases}$$

Restando (II)-(I)

$$s_1^2 + s_2^2 - \left(t_1^2 + t_2^2\right) - 2bs_1 + 2bt_1 = 0$$

$$s_1^2 + s_2^2 - \left(t_1^2 + t_2^2\right) = 2b(s_1 - t_1)$$

$$\frac{s_1^2 + s_2^2 - \left(t_1^2 + t_2^2\right)}{2(s_1 - t_1)} = b_{xy}$$

Calculando r:

$$r_{xy} = \sqrt{(x - b_{xy})^2 + y^2}$$

para $x = t_1$ y $y = t_2$

$$\Rightarrow r_{xy} = \sqrt{(t_1 - b_{xy})^2 + t_2^2}$$

Por lo que, se reescribe la ecuación (20), se tiene que:

$$(x - b)^2 + y^2 = a + b^2 \tag{21}$$

En otras palabras, las curvas que minimizan la distancia entre dos puntos, (t_1, t_2) y (s_1, s_2), con la condición de que $t_1 \neq s_1$, en el Plano de Poincaré, son semicircunferencias cuyo centro está en $(b, 0)$ y con un radio que corresponde a:

$$r_{xy} = \sqrt{(b_{xy} - t_1)^2 + t_2^2}$$

Al resolver para obtener el valor de t_2, obtenemos:

$$t_2^2 = rxy^2 - (b_{xy} - t_1)^2 \tag{22}$$

empleando una parametrización en coordenadas hiperbólicas obtenemos:

- $x = b_{xy} - r_{xy} \tanh(t)$
- $y = \dfrac{r_{xy}}{\cosh(t)}$

Definamos $s = s' \wedge s = s''$ tal que:



$$\Rightarrow \eta^1(s') = b_{xy} - r_{xy}\tanh s' = t_1 \quad \Rightarrow \quad s' = \operatorname{arctanh}\left(\frac{b_{xy} - t_1}{r_{xy}}\right) \quad \cdots (I)$$

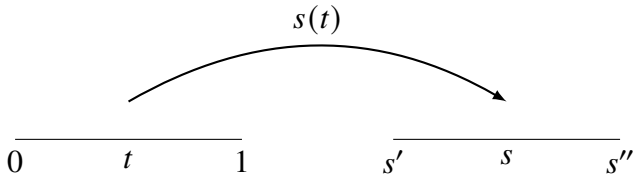
$$\Rightarrow \eta^2(s') = \frac{r_{xy}}{\cosh(s')} = t_2$$

Además:

$$\Rightarrow \eta^1(s'') = b_{xy} - r_{xy}\tanh s'' = s_1 \quad \Rightarrow \quad s'' = \operatorname{arctanh}\left(\frac{b_{xy} - s_1}{r_{xy}}\right) \quad \cdots (II)$$

$$\Rightarrow \eta^2(s'') = \frac{r_{xy}}{\cosh(s'')} = s_2$$

Para un $t \in [0, 1]$, tenemos:



$$\Rightarrow \frac{s - s'}{t - 0} = \frac{s'' - s'}{1 - 0} \quad \Rightarrow \quad s - s' = t(s'' - s')$$

$$s = s' + ts'' - ts' \quad \Rightarrow \quad s = (1 - t)s' + ts'' \tag{23}$$

Reemplazando s' y s'' de (I) y (II) tenemos:

$$\Rightarrow s = (1 - t)s' + ts'' = (1 - t)\operatorname{arctanh}\left(\frac{b_{xy} - t_1}{r_{xy}}\right) + t \cdot \operatorname{arctanh}\left(\frac{b_{xy} - s_1}{r_{xy}}\right)$$

como t es una variable sorda entonces $t = s$ y nos queda:

- $x = b_{xy} - r_{xy}\tanh\left((1 - s)\operatorname{arctanh}\left(\frac{b_{xy} - t_1}{r_{xy}}\right) + s \cdot \operatorname{arctanh}\left(\frac{b_{xy} - s_1}{r_{xy}}\right)\right)$
- $y = \dfrac{r_{xy}}{\cosh\left((1 - s)\operatorname{arctanh}\left(\frac{b_{xy} - t_1}{r_{xy}}\right) + s \cdot \operatorname{arctanh}\left(\frac{b_{xy} - s_1}{r_{xy}}\right)\right)}$

2.6.2. Segundo caso: $t_1 = s_1$

En el caso donde $t_1 = s_1$, al sustituir nuevamente los valores de a y b en la ecuación (20), se obtiene la siguiente relación:

$$(s_1 - t_1)(x^2 + y^2) - (s_1^2 - t_1^2)x - (s_2^2 - t_2^2)x - (t_1^2 + t_2^2)s_1 + (s_1^2 + s_2^2)t_1 = 0 \qquad (24)$$

Partiendo de la condición $t_1 = s_1$ y combinando los términos semejantes, se obtiene la ecuación:

$$0 - 0 - (s_2^2 - t_2^2)x - (t_1^2 + t_2^2)t_1 + (t_1^2 + s_2^2)t_1 = 0$$

$$-(s_2^2 - t_2^2)x + (-t_1^2 - t_2^2 + t_1^2 + s_2^2)t_1 = 0$$

$$-(s_2^2 - t_2^2)x + (s_2^2 - t_2^2)t_1 = 0$$

$$(s_2^2 - t_2^2)(-x + t_1) = 0$$

$$x = t_1 \qquad (25)$$

es decir, una recta perpendicular al eje horizontal que intersecta en t_1. Esto nos permite observar que:

$$x = t_1$$

$$\Rightarrow \frac{dx}{dt} = 0$$

reemplazando en (19), tenemos:

$$\frac{d^2y}{dt^2} + \frac{1}{y}\left(\frac{dx}{dt}\right)^2 - \frac{1}{y}\left(\frac{dy}{dt}\right)^2 = 0$$

$$\Rightarrow \frac{d^2y}{dt^2} + \frac{1}{y}(0)^2 - \frac{1}{y}\left(\frac{dy}{dt}\right)^2 = 0$$

$$\Rightarrow \frac{d^2y}{dt^2} - \frac{1}{y}\left(\frac{dy}{dt}\right)^2 = 0$$

$$\Rightarrow y\frac{d^2y}{dt^2} - \left(\frac{dy}{dt}\right)^2 = 0$$

Dividimos por y^2 y reescribimos $\frac{d^2y}{dt^2} = \ddot{y}$, $\frac{dy}{dt} = \dot{y}$, nos quedaría de la siguiente manera:

$$\frac{y\ddot{y} - (\dot{y})^2}{y^2} = 0$$

$$\frac{d}{dt}\left(\frac{\dot{y}}{y}\right) = \frac{y\ddot{y} - (\dot{y})^2}{y^2} = 0$$

$$\Rightarrow \frac{\dot{y}}{y} = c_1$$

donde c_1 es una constante.

$$\int \frac{\dot{y}}{y} dt = \int c_1 dt$$

$$\Rightarrow \ln(y) = c_1.t - c_1.m_0$$

, donde m_0 es una constante

$$\Rightarrow y = e^{c_1.(t-m_0)}$$

dado que $y = e^{c_1(t-m_0)}$ para alguna constante real m_0. Si el parámetro t representa la longitud de arco, entonces la constante c_1 está restringida por la condición (10)

$$1 = \frac{1}{x_2^2}\left(dx_1^2 + dx_2^2\right) = \frac{dx_1^2}{x_2^2} + \frac{dx_2^2}{x_2^2} = 0 + \frac{(e^{c_1(t-m_0)}.c_1)^2}{(e^{c_1(t-m_0)})^2} = 0 + c_1^2$$

$$\Rightarrow c_1 = \pm 1$$

que corresponde a una línea vertical en $\mathbb{H}^2$, recorrida hacia arriba o hacia abajo dependiendo de la elección del signo ±, sin perdida de generalidad usaremos el signo +,y haciendo $c_1.m_0 = m_1$ donde m_1 es una constante, entonces esta solución quedaría representado por:

$$y = e^{t-m1}$$

De la misma forma y haciendo uso de (23) calcularemos los valores de s' y s'':

$$x = t_1$$

$$y = e^{s-m_1} \quad \Rightarrow \quad t_2 = e^{s'-m_1} \quad \Rightarrow \quad \ln(t_2) = s' - m_1$$

$$s_2 = e^{s''-m_1} \quad \Rightarrow \quad \ln(s_2) = s'' - m_1$$

Reemplazando en (23) tenemos

$$s = (1-t)\left(\ln(t_2) + m_1\right) + t\left(\ln(s_2) + m_1\right) = (1-t)\ln(t_2) + t\ln(s_2) + m_1$$

como t es una variable sorda, reemplazando $t = s$ tenemos

$$s = (1-s)\ln(t_2) + s\ln(s_2) + m_1$$

- $x = t_1$
- $y = e^{s=(1-s)\ln(t_2)+s\ln(s_2)+m_1}$

De donde, recordemos que para $s = 0 \Rightarrow \eta(0) = x = (t_1, t_2)$, entonces:

$$\eta(0) = \left(t_1, e^{(1-0)\ln(t_2)+0\cdot\ln(s_2)+m_1}\right) = (t_1, t_2).$$

Para que esto sea cierto, debe cumplirse:

$$e^{\ln(t_2)+m_1} = t_2 \Rightarrow \ln(t_2) + m_1 = \ln(t_2).$$

De esta manera, se concluye que $m_1 = 0$.

Para finalizar, definimos nuestras geodésicas según los dos casos de la siguiente manera:

$$\eta^1_{xy}(s) := \begin{cases} t_1, & \text{si } t_1 = s_1, \\ b_{xy} - r_{xy}\tanh\left((1-s)\cdot \operatorname{arctanh}\left(\dfrac{b_{xy}-t_1}{r_{xy}}\right) + s\cdot\operatorname{arctanh}\left(\dfrac{b_{xy}-s_1}{r_{xy}}\right)\right), & \text{si } t_1 \neq s_1, \end{cases}$$

y

$$\eta^2_{xy}(s) := \begin{cases} e^{(1-s)\ln t_2 + s\ln s_2}, & \text{si } t_1 = s_1, \\ \dfrac{r_{xy}}{\cosh\left((1-s)\cdot \operatorname{arctanh}\left(\dfrac{b_{xy}-t_1}{r_{xy}}\right) + s\cdot\operatorname{arctanh}\left(\dfrac{b_{xy}-s_1}{r_{xy}}\right)\right)}, & \text{si } t_1 \neq s_1, \end{cases}$$

para todo $s \in [0, 1]$

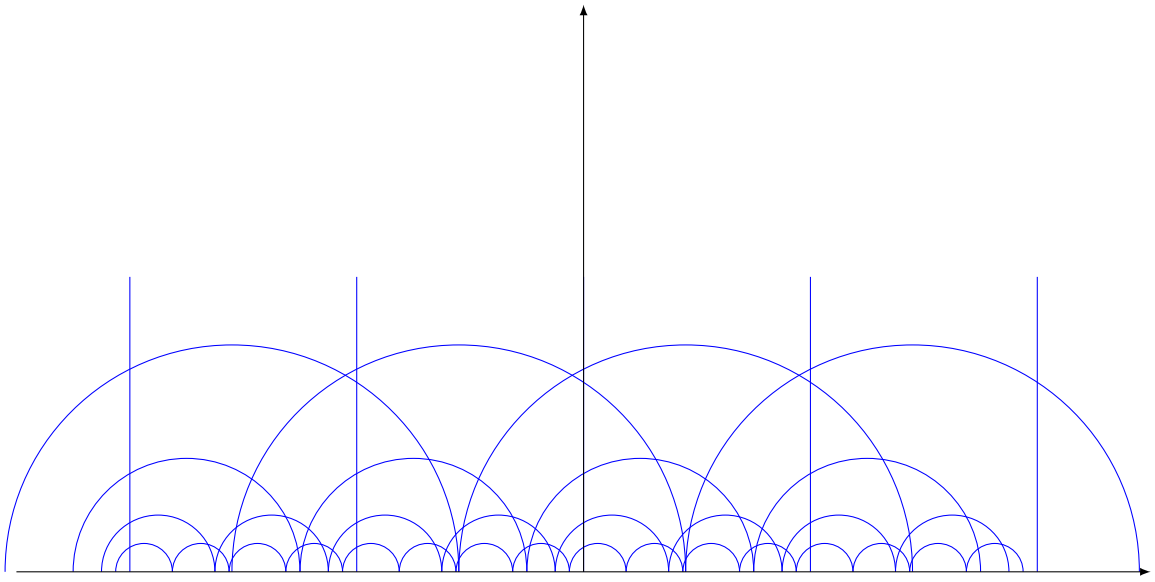


Figura 1: Geodesicas en el plano superior de Poincaré

2.7. Velocidad de las geodésicas respecto a la métrica hiperbólica

Como hemos podido ver en la subsección anterior, las geodésicas en $\mathbb{H}^2$ son líneas verticales

$$\eta^1(t) = (t_1, e^t)$$

y semicírculos

$$\eta^2(t) = \left(b_{xy} - r_{xy}\tanh(t), \frac{r_{xy}}{\cosh(t)}\right),$$

Donde por [16, p. 17], define la velocidad de la geodésica respecto a la métrica Riemanniana.

$$\|\dot{\eta}(t)\|_g = \sqrt{g_{\eta(t)}\left(\dot{\eta}(t), \dot{\eta}(t)\right)} = \sqrt{\frac{\dot{x}(t)^2 + \dot{y}(t)^2}{y(t)^2}}.$$

por lo cual, para la geodésica $\eta^1(t)$,

$$\|\dot{\eta}^1(t)\|_g = \sqrt{\frac{0^2 + e^{2t}}{e^{2t}}} = 1,$$

$$\dot{x}(t) = \frac{d}{dt}\left(b_{xy} - r_{xy}\tanh(t)\right) = -r_{xy}\cdot\operatorname{sech}^2(t),$$

$$\dot{y}(t) = \frac{d}{dt}\left(\frac{r_{xy}}{\cosh(t)}\right) = -r_{xy}\cdot\frac{\sinh(t)}{\cosh^2(t)} = -r_{xy}\cdot\tanh(t)\cdot\operatorname{sech}(t).$$

Cálculo de la norma con la métrica de Poincaré:

$$\|\dot{\eta}(t)\|_g^2 = \frac{\dot{x}(t)^2 + \dot{y}(t)^2}{y(t)^2},$$

donde:

$$\dot{x}(t)^2 = r_{xy}^2\cdot\operatorname{sech}^4(t), \quad \dot{y}(t)^2 = r_{xy}^2\cdot\tanh^2(t)\cdot\operatorname{sech}^2(t), \quad y(t)^2 = \left(\frac{r_{xy}}{\cosh(t)}\right)^2 = r_{xy}^2\cdot\operatorname{sech}^2(t).$$

Sustituyendo:

$$\|\dot{\eta}(t)\|_g^2 = \frac{r_{xy}^2\cdot\operatorname{sech}^4(t) + r_{xy}^2\cdot\tanh^2(t)\cdot\operatorname{sech}^2(t)}{r_{xy}^2\cdot\operatorname{sech}^2(t)}.$$

Factorizando y simplificando:

$$\|\dot{\eta}(t)\|_g^2 = \operatorname{sech}^2(t) + \tanh^2(t).$$

Utilizando la identidad hiperbólica:

$$\operatorname{sech}^2(t) + \tanh^2(t) = 1,$$

concluimos que:

$$\|\dot{\eta}(t)\|_g = 1. \tag{26}$$

La curva $\eta(t)$ tiene velocidad constante unitaria con respecto a la métrica hiperbólica, por lo tanto, es una geodésica parametrizada afínmente en el plano superior de Poincaré.

2.8. Relación de las ecuaciones diferenciales de las geodésicas por símbolos de Christoffel y ecuación de Euler-Lagrange

Para establecer la equivalencia entre las ecuaciones diferenciales que describen las geodésicas, derivadas a partir de los símbolos de Christoffel y aquellas obtenidas mediante la ecuación de Euler-Lagrange, se tomará como punto de partida el sistema dado en (6) y (7). A partir de dichas expresiones, se procederá a deducir la ecuación variacional correspondiente, formulada en (18), con el fin de mostrar rigurosamente su equivalencia.

Consideramos que

$$\frac{dy}{dt} = \frac{dy}{dx} \cdot \frac{dx}{dt}, \tag{27}$$

$$\frac{d^2y}{dt^2} = \frac{d^2y}{dx^2}\left(\frac{dx}{dt}\right)^2 + \frac{dy}{dx} \cdot \frac{d^2x}{dt^2}. \tag{28}$$

Consecuentemente, de la ecuación (6) reemplazando el valor de $\dfrac{d^2x}{dt^2}$ en (28) obtenemos:

$$\frac{d^2y}{dt^2} = \frac{d^2y}{dx^2}\left(\frac{dx}{dt}\right)^2 + \frac{dy}{dx} \cdot \frac{2}{y} \cdot \frac{dx}{dt}\frac{dy}{dt},$$

$$= \frac{d^2y}{dx^2}\left(\frac{dx}{dt}\right)^2 + \frac{2}{y}\left(\frac{dy}{dt}\right)^2. \tag{29}$$

Reemplazando el valor de (27) en (29) obtenemos:

$$\frac{d^2y}{dt^2} = \frac{d^2y}{dx^2}\left(\frac{dx}{dt}\right)^2 + \frac{2}{y}\left(\frac{dy}{dx}\cdot\frac{dx}{dt}\right)^2. \tag{30}$$

De este modo, reemplazando (30) en la ecuación (7) se obtiene

$$\frac{d^2y}{dx^2}\left(\frac{dx}{dt}\right)^2 + \frac{2}{y}\left(\frac{dy}{dx}\cdot\frac{dx}{dt}\right)^2 + \frac{1}{y}\left(\frac{dx}{dt}\right)^2 - \frac{1}{y}\left(\frac{dy}{dt}\right)^2 = 0,$$

Reemplazando el valor de (27), tenemos

$$\frac{d^2y}{dx^2}\left(\frac{dx}{dt}\right)^2 + \frac{2}{y}\left(\frac{dy}{dx}\cdot\frac{dx}{dt}\right)^2 + \frac{1}{y}\left(\frac{dx}{dt}\right)^2 - \frac{1}{y}\left(\frac{dy}{dx}\cdot\frac{dx}{dt}\right)^2.$$

Finalmente nos queda:

$$\frac{d^2y}{dx^2}\left(\frac{dx}{dt}\right)^2 + \frac{2}{y}\left(\frac{dy}{dx}\frac{dx}{dt}\right)^2 - \frac{1}{y}\left(\frac{dy}{dx}.\frac{dx}{dt}\right)^2 + \frac{1}{y}\left(\frac{dx}{dt}\right)^2 = 0.$$

Si $\dfrac{dx}{dt} = 0$ en todo el intervalo, entonces $x(t) = t_0$ es constante y la curva se reduce a la recta vertical

$$\eta(t) = \big(t_0,\ y(t)\big), \qquad y(t) > 0.$$

Estas rectas verticales son, en efecto, una familia de geodésicas de $\mathbb{H}^2$.

Para el otro caso donde $\dfrac{dx}{dt} \neq 0$ tenemos, por consiguiente, simplificando los términos semejantes, la ecuación se simplifica a:

$$\frac{d^2y}{dx^2} + \frac{1}{y}\left(\frac{dy}{dx}\right)^2 + \frac{1}{y} = 0,$$

igual a la ecuación (18),

A fin de demostrar la equivalencia en el sentido inverso, partiremos ahora de las ecuaciones de Euler-Lagrange con el objetivo de deducir las ecuaciones de las geodésicas expresadas mediante los símbolos de Christoffel. Para ello, será necesario introducir previamente algunas definiciones fundamentales que permitirán estructurar el desarrollo matemático de manera rigurosa.

Para simplificar los cálculos denotaremos $\frac{dx}{dt} = \dot{x}$ y $\frac{dy}{dt} = \dot{y}$ y al funcional de longitud asociada a una curva $\eta(t) = (x(t), y(t))$ es

$$L = \int F(x, y, y')\, dt, \qquad F = \frac{\sqrt{\dot{x}^2 + \dot{y}^2}}{y}.$$

En el calculo variacional general para curvas paramétricas se considera la curva como $(x(t), y(t))$ y una función $F(t, x(t), y(t), \dot{x}(t), \dot{y}(t))$ que depende de ambas coordenadas y sus derivadas respecto a un parámetro t. En este caso, las ecuaciones de Euler–Lagrange se escriben en su forma temporal habitual como vemos en [16, p. 18] y [14, p. 6];

$$\frac{d}{dt}\left(\frac{\partial f}{\partial \dot{x}}\right) - \frac{\partial f}{\partial x} = 0, \qquad \frac{d}{dt}\left(\frac{\partial f}{\partial \dot{y}}\right) - \frac{\partial f}{\partial y} = 0, \tag{31}$$

Esta formulación es esencial cuando tratamos las dos coordenadas por igual y dejamos libre la parametrización. En cambio, la forma tradicional vista en 8

$$\frac{\partial F}{\partial y} - \frac{d}{dx}\left(\frac{\partial F}{\partial y'}\right) = 0.$$

resulta del caso especial de una sola función $y = y(x)$ en el plano. Es decir, la forma con x como variable independiente ya está incorporada.

En geometría riemanniana se tiene que para curvas afínmente parametrizadas los funcionales de energía y de longitud comparten los mismos extremos geodésicos. En particular, uno puede usar el funcional de energía.

$$\mathrm{E}(\eta) = \frac{1}{2}\int_a^b g(\dot{\eta}, \dot{\eta})\, dt.$$

en lugar del funcional de longitud

$$\mathrm{L}(\eta) = \int_a^b \sqrt{g(\dot{\eta}, \dot{\eta})}\, dt.$$

donde $g(\dot{\eta}, \dot{\eta})$ es, por tanto, la norma al cuadrado de $\dot{\eta}(t)$ inducida por la métrica g:

$$\mathrm{g}(\dot{\eta}(t), \dot{\eta}(t)) = \|\dot{\eta}(t)\|_g^2.$$

Se muestra que los puntos críticos del funcional energía son precisamente las geodésicas de velocidad constante, y que cualquier curva que minimiza energía con extremos fijos también

minimiza la longitud [16, p. 17] y [19, p. 111].
De este modo E evita la raíz cuadrada y simplifica los cálculos. De ello se sigue que, imponiendo una parametrización afín (velocidad constante), minimizar E o L lleva a las mismas ecuaciones de geodésica con símbolos de Christoffel.
La curva $\eta(t)$ tiene velocidad constante unitaria con respecto a la métrica hiperbólica, por lo tanto, es una geodésica parametrizada afínmente en el plano superior de Poincaré. De este modo para simplificar el tratamiento, utilizamos el funcional de energía, equivalente al funcional de longitud para parametrizaciones afines:

$$\mathrm{E} = \int \frac{\dot{x}^2 + \dot{y}^2}{2y^2}\, dt.$$

Entonces, F se define como:

$$F(x, y, \dot{x}, \dot{y}) = \frac{\dot{x}^2 + \dot{y}^2}{2y^2}.$$

Calculamos las derivadas necesarias:

$$\begin{aligned}
\frac{\partial F}{\partial x} &= 0, \\
\frac{\partial F}{\partial \dot{x}} &= \frac{\dot{x}}{y^2}, \\
\frac{\partial F}{\partial y} &= -\frac{\dot{x}^2 + \dot{y}^2}{y^3}, \\
\frac{\partial F}{\partial \dot{y}} &= \frac{\dot{y}}{y^2}.
\end{aligned}$$

Aplicamos las ecuaciones de Euler–Lagrange de 31

Coordenada x:

$$\frac{d}{dt}\left(\frac{\dot{x}}{y^2}\right) = \frac{\ddot{x}}{y^2} - \frac{2\dot{x}\dot{y}}{y^3} = 0.$$

Multiplicando por y^2, obtenemos:

$$\ddot{x} - \frac{2}{y}\dot{x}\dot{y} = 0. \tag{32}$$

Coordenada y:

$$\frac{d}{dt}\left(\frac{\dot{y}}{y^2}\right) + \frac{\dot{x}^2 + \dot{y}^2}{y^3} = 0.$$

Calculamos la derivada:

$$\frac{d}{dt}\left(\frac{\dot{y}}{y^2}\right) = \frac{\ddot{y}}{y^2} - \frac{2\dot{y}^2}{y^3},$$

entonces:

$$\frac{\ddot{y}}{y^2} - \frac{2\dot{y}^2}{y^3} + \frac{\dot{x}^2 + \dot{y}^2}{y^3} = 0.$$

Multiplicando por y^2, resulta:

$$\ddot{y} - \frac{2\dot{y}^2}{y} + \frac{\dot{x}^2 + \dot{y}^2}{y} = 0,$$

lo cual se simplifica a:

$$\ddot{y} + \frac{\dot{x}^2 - \dot{y}^2}{y} = 0. \tag{33}$$

se observa que de 32 y 33, es el sistema de ecuaciones diferenciales que se genera por los simbolos de Christoffel.
Ambos métodos utilizados para calcular las geodésicas conducen a resultados equivalentes, tal como era de esperarse.

2.9. Calculo de $\exp_y^{-1} x$

Para esta sección se brinda los detalles del desarrollo de las operaciones y los resultados finales se han corroborado haciendo uso del software Maple (ver códigos 7.1-7.2 en anexos), de este modo en base a los resultados anteriores de las geodésicas hallaremos

$$\exp_y^{-1} x = \eta'(0)$$

2.9.1. Para el caso de $t_1 = t_2$

Sea $\eta_{xy}(s) = (\eta_1(s), \eta_2(s))$ donde:

$$\eta_{xy}(s) = \left(t_1,\ e^{(1-s)\ln t_2 + s \ln s_2}\right)$$

Por lo tanto

$$\eta_1(s) = t_1 \Rightarrow \eta_1(s)' = 0$$

Sea η_2 definida como:

$$\eta_2 = e^{(1-s)\ln(t_2)+s\ln(s_2)}.$$

El diferencial de η_2 está dado por:

$$\eta_2' = (-\ln(t_2) + \ln(s_2)) \cdot e^{(1-s)\ln(t_2)+s\ln(s_2)}.$$

evaluando para $s = 0$ tenemos:

$$(-\ln(t_2) + \ln(s_2))\, t_2 = \ln\left(\frac{s_2}{t_2}\right) \cdot t_2$$

$$\therefore \exp_y^{-1}\ x = \left(0, t_2 \ln\frac{t_2}{s_2}\right) \qquad \text{si} \quad t_1 = s_1$$

2.9.2. Para el caso $t_1 \neq s_1$

Sea $\eta_{xy}(s) = (\eta_1(s), \eta_2(s))$ donde:

$$\eta_{xy}(s) = \begin{cases} \eta_1(s) = b_{xy} - r_{xy} \tanh\left((1-s) \cdot \operatorname{arctanh}\left(\dfrac{b_{xy}-t_1}{r_{xy}}\right) + s \cdot \operatorname{arctanh}\left(\dfrac{b_{xy}-s_1}{r_{xy}}\right)\right), \\ \eta_2(s) = \dfrac{r_{xy}}{\cosh\left((1-s) \cdot \operatorname{arctanh}\left(\dfrac{b_{xy}-t_1}{r_{xy}}\right) + s \cdot \operatorname{arctanh}\left(\dfrac{b_{xy}-s_1}{r_{xy}}\right)\right)}. \end{cases}$$

Sea η_1 definido como:

$$\eta_1(s) = b_{xy} - r_{xy} \tanh\left((1-s) \cdot \operatorname{arctanh}\left(\frac{b_{xy}-t_1}{r_{xy}}\right) + s \cdot \operatorname{arctanh}\left(\frac{b_{xy}-s_1}{r_{xy}}\right)\right),$$

y su derivada respecto a s:

$$\eta_1' = -r_{xy}\left(-\operatorname{arctanh}\left(\frac{b_{xy}-t_1}{r_{xy}}\right) + \operatorname{arctanh}\left(\frac{b_{xy}-s_1}{r_{xy}}\right)\right) \cdot \left(1 - \tanh^2\left((1-s) \cdot \operatorname{arctanh}\left(\frac{b_{xy}-t_1}{r_{xy}}\right)\right.\right.$$

$$+s \cdot \operatorname{arctanh}\left(\frac{b_{xy}-s_1}{r_{xy}}\right)\right)\right).$$

Además, evaluando para $s = 0$ tenemos:

$$\Rightarrow -r_{xy}\left(-\operatorname{arctanh}\left(\frac{b_{xy}-t_1}{r_{xy}}\right)+\operatorname{arctanh}\left(\frac{b_{xy}-s_1}{r_{xy}}\right)\right)\cdot\left(1-\left(\frac{b_{xy}-t_1}{r_{xy}}\right)^2\right).$$

$$\Rightarrow r_{xy}\left(\operatorname{arctanh}\left(\frac{b_{xy}-t_1}{r_{xy}}\right)-\operatorname{arctanh}\left(\frac{b_{xy}-s_1}{r_{xy}}\right)\right)\cdot\left(\frac{{r_{xy}}^2-\left(b_{xy}-t_1\right)^2}{r_{xy}^2}\right).$$

De donde haciendo uso de (22), nos queda:

$$\left(\operatorname{arctanh}\left(\frac{b_{xy}-t_1}{r_{xy}}\right)-\operatorname{arctanh}\left(\frac{b_{xy}-s_1}{r_{xy}}\right)\right)\cdot\left(\frac{t_2^2}{r_{xy}}\right).$$

Sea η_2 definido como:

$$\eta_2(s)=\frac{r_{xy}}{\cosh\left((1-s)\cdot\operatorname{arctanh}\left(\frac{b_{xy}-t_1}{r_{xy}}\right)+s\cdot\operatorname{arctanh}\left(\frac{b_{xy}-s_1}{r_{xy}}\right)\right)},$$

y su derivada respecto a s es:

$$\eta_2'=-r_{xy}\left(-\operatorname{arctanh}\left(\frac{b_{xy}-t_1}{r_{xy}}\right)+\operatorname{arctanh}\left(\frac{b_{xy}-s_1}{r_{xy}}\right)\right).P(s)$$

Donde $P(s)$ es:

$$P(s)=\frac{\sinh\left((1-s)\cdot\operatorname{arctanh}\left(\frac{b_{xy}-t_1}{r_{xy}}\right)+s\cdot\operatorname{arctanh}\left(\frac{b_{xy}-s_1}{r_{xy}}\right)\right)}{\cosh^2\left((1-s)\cdot\operatorname{arctanh}\left(\frac{b_{xy}-t_1}{r_{xy}}\right)+s\cdot\operatorname{arctanh}\left(\frac{b_{xy}-s_1}{r_{xy}}\right)\right)}$$

$$\Rightarrow \eta_2'=-r_{xy}\left(-\operatorname{arctanh}\left(\frac{b_{xy}-t_1}{r_{xy}}\right)+\operatorname{arctanh}\left(\frac{b_{xy}-s_1}{r_{xy}}\right)\right)\cdot\frac{\sinh(\dots)}{\cosh^2(\dots)}$$

Evaluando para $s = 0$, nos queda la siguiente expresión:

$$\Rightarrow -\frac{r_{xy}\left(-\operatorname{arctanh}\left(\frac{b_{xy}-t_1}{r_{xy}}\right)+\operatorname{arctanh}\left(\frac{b_{xy}-s_1}{r_{xy}}\right)\right)\sinh\left(\operatorname{arctanh}\left(\frac{b_{xy}-t_1}{r_{xy}}\right)\right)}{\cosh^2\left(\operatorname{arctanh}\left(\frac{b_{xy}-t_1}{r_{xy}}\right)\right)}$$

$$\Rightarrow -\sqrt{1-\left(\frac{b_{xy}-t_1}{r_{xy}}\right)^2}\left(-\operatorname{arctanh}\left(\frac{b_{xy}-t_1}{r_{xy}}\right)+\operatorname{arctanh}\left(\frac{b_{xy}-s_1}{r_{xy}}\right)\right)(b_{xy}-t_1),$$

$$\Rightarrow -\sqrt{\frac{{r_{xy}}^2-\left(b_{xy}-t_1\right)^2}{r_{xy}^2}}\left(-\operatorname{arctanh}\left(\frac{b_{xy}-t_1}{r_{xy}}\right)+\operatorname{arctanh}\left(\frac{b_{xy}-s_1}{r_{xy}}\right)\right)\cdot(b_{xy}-t_1),$$

Haciendo uso de (22) , tenemos:

$$\Rightarrow -\sqrt{\frac{t_2^2}{r_{xy}^2}}\left(-\operatorname{arctanh}\left(\frac{b_{xy}-t_1}{r_{xy}}\right)+\operatorname{arctanh}\left(\frac{b_{xy}-s_1}{r_{xy}}\right)\right)(b_{xy}-t_1),$$

$$\Rightarrow \frac{t_2}{r_{xy}}\left(\operatorname{arctanh}\left(\frac{b_{xy}-t_1}{r_{xy}}\right)-\operatorname{arctanh}\left(\frac{b_{xy}-s_1}{r_{xy}}\right)\right)(b_{xy}-t_1)$$

$$\therefore \exp_y^{-1} x = \frac{t_2}{r_{xy}}\left(\arctan\left(\frac{b_{xy}-t_1}{r_{xy}}\right)-\arctan\left(\frac{b_{xy}-s_1}{r_{xy}}\right)\right)(t_2, b_{xy}-t_1) \qquad \text{si } t_1 \neq s_1$$

$$\exp_y^{-1} x = \begin{cases} \left(0, t_2 \ln \frac{t_2}{s_2}\right), & \text{si } t_1 = s_1, \\ \dfrac{t_2}{r_{xy}}\left(\arctan\dfrac{b_{xy}-t_1}{r_{xy}}-\arctan\dfrac{b_{xy}-s_1}{r_{xy}}\right)(t_2, b_{xy}-t_1), & \text{si } t_1 \neq s_1. \end{cases} \tag{34}$$

2.10. Aproximación numérica de las geodésicas

Hallar de manera analítica una geodésica que una dos puntos dados$(\eta(0) = x\ ,\ \eta(1) = y)$ en una variedad Riemanniana requiere mucho más esfuerzo que determinar una geodésica a partir de datos iniciales. Esto se debe a que es mucho más difícil resolver problemas de valor en la frontera que problemas de valor inicial, como podemos ver en el trabajo de Noakes [21]. De este modo, buscamos aproximar las geodésicas haciendo uso de sus valores iniciales; la

aproximación se realizará por el método de Runge-Kutta para luego implementarlo en los métodos computacionales de gradiente y subgradiente. Para esta aproximación se revisó el trabajo de Udriste [26].

Las geodésicas de $\mathbb{H}^2$ son las soluciones del sistema diferencial:

$$\frac{d^2x^i}{dt^2} + \sum_{j,k} \Gamma^i_{jk} \frac{dx^j}{dt} \frac{dx^k}{dt} = 0, \quad i = 1, 2$$

cómo hemos podido observar en (5)

$$\eta : I \to M, \quad \eta(t) = (\eta^1(t), \eta^2(t))$$

que satisfacen de manera idéntica la ecuación diferencial anterior.

Dado que este sistema diferencial es de orden dos, una geodésica puede determinarse mediante sus condiciones iniciales:
Problema de Cauchy:

$$\eta(0) = x_0, \quad \dot{\eta}(0) = v_{x_0}$$

Existen diversos métodos computacionales para aproximar soluciones de ecuaciones diferenciales ordinarias de segundo orden no lineales, entre los cuales destacan el método de Euler, los métodos de Runge-Kutta y los métodos multipasos como el de Adams-Bashforth-Moulton. En esta sección se implementará el método de Runge-Kutta de orden cuatro (RK4), ya que ofrece ventajas significativas respecto a los otros métodos mencionados:

- A diferencia del método de Euler que puede considerarse un caso particular de Runge-Kutta de primer orden [33, p. 359] lo cual generará aproximaciones no muy precisas, el método RK4 proporciona una mayor precisión en cada paso de integración.

- En comparación con métodos multipaso como Adams-Bashforth-Moulton, RK4 no requiere almacenar ni calcular valores previos de la solución, lo que simplifica su implementación[33, p. 365].

- Generalmente, los métodos de Adams-Bashforth-Moulton requieren una etapa inicial con RK4 para obtener las primeras soluciones necesarias, lo que los hace menos autónomos en la práctica[33, p. 364].

2.10.1. Solución aproximada del problema de Cauchy en el plano superior de Poincaré.

Dado que el problema de Cauchy anterior puede reformularse como un problema de Cauchy en TM (usando el levantamiento natural de una geodésica), se tiene el siguiente sistema de ecuaciones diferenciales:

$$\begin{cases} \dfrac{dx^i}{dt} = y^i, \\ \dfrac{dy^i}{dt} = -\Gamma^i_{jk}(x) y^j y^k, \quad x^i(0) = x^i_o, \quad y^i(0) = y^i_o. \end{cases}$$

Dado este sistema, se puede aplicar el algoritmo de Runge-Kutta para ecuaciones diferenciales de primer orden. Para expresar las fórmulas de Runge-Kutta, definimos:

$$x = (x^i), \quad y = (y^i), \quad X = (x, y),$$

$$F = (F^i, F^{n+1}), \quad F^i = y^i, \quad F^{n+1} = -\Gamma^i_{jk}(x) y^j y^k.$$

Reformulamos el problema de Cauchy en la forma:

$$\frac{dX}{dt} = F(X), \quad X(0) = (x^i_o, y^i_o).$$

Para $t \in [0, T]$, se define el paso $h = T/m$ y se establece $t_\alpha = \alpha h$ para $\alpha = 0, 1, \ldots, m$. Definiendo $X_\alpha \approx X(t_\alpha)$, el algoritmo de Runge-Kutta se expresa como:

$$X_0 = X(0),$$

$$X_{\alpha+1} = X_\alpha + \frac{1}{6}\left(r_1 + 2r_2 + 2r_3 + r_4\right),$$

donde los coeficientes r_i son:

$$r_1 = hF(X_\alpha),$$

$$r_2 = hF\left(X_\alpha + \frac{k_1}{2}\right),$$

$$r_3 = hF\left(X_\alpha + \frac{k_2}{2}\right),$$

$$r_4 = hF\left(X_\alpha + k_3\right).$$

Para $t \in [0, T]$ encontramos el paso $h = T/m$ y, por lo tanto, $t_\alpha = \alpha h$, $\alpha = 0, 1, \ldots, m$. Definiendo $X_\alpha \approx X(t_\alpha)$, el algoritmo de Runge-Kutta se describe como:

$$X_0 = X(0)$$

$$X_{\alpha+1} = X_\alpha + \frac{1}{6}\left(r_1 + 2r_2 + 2r_3 + r_4\right)$$

$$\begin{aligned} r_1 &= hF(X_\alpha), \\ r_2 &= hF\left(X_\alpha + \frac{r_1}{2}\right), \\ r_3 &= hF\left(X_\alpha + \frac{r_2}{2}\right), \\ r_4 &= hF\left(X_\alpha + r_3\right). \end{aligned}$$

El problema de Cauchy anterior también puede resolverse mediante el método de Runge-Kutta adaptado directamente a los sistemas diferenciales de orden dos:

$$I = [0, T], \quad h = \frac{T}{m}, \quad x^i(0) = x_0^i, \quad \frac{dx^i}{dt}(0) = y_0^i, \quad t_\alpha = \alpha h$$
$$\alpha = 0, 1, \ldots, m; \quad i = 1, \ldots, n$$

$$x_\alpha^i = x_\alpha^i, \quad \frac{dx^i}{dt}(t_\alpha) = y_\alpha^i$$

$$x_{\alpha+1}^i = x_\alpha^i + h y_\alpha^i + \frac{h}{6}\left(r_1^i + r_2^i + r_3^i\right)$$

$$y_{\alpha+1}^i = y_\alpha^i + \frac{1}{6}\left(r_1^i + 2r_2^i + 2r_3^i + r_4^i\right)$$

$$r_1^i = hf^i(x_\alpha^j, y_\alpha^j), \quad j = 1, \ldots, n$$

$$r_2^i = hf^i\left(x_\alpha^j + \frac{h}{2}y_\alpha^j, y_\alpha^j + \frac{1}{2}r_1^j\right)$$

$$r_3^i = hf^i\left(x_\alpha^j + \frac{h}{2}y_\alpha^j + \frac{h}{4}r_1^j, y_\alpha^j + \frac{1}{2}r_2^j\right)$$

$$r_4^i = hf^i\left(x_\alpha^j + hy_\alpha^j + \frac{h}{2}r_2^j, y_\alpha^j + r_3^j\right).$$

Las curvas geodésicas en el plano superior de Poincaré corresponden a las líneas verticales:

$$\eta(t) : x = t_0, \quad y = e^t$$

y a los arcos de circunferencia:

$$\eta(t) : x = b - r\tanh t, \quad y = \frac{r}{\cosh t}$$

Para aproximar estas trayectorias, aplicaremos el método de Runge-Kutta de cuarto orden. El sistema diferencial que se representa en (18) que describe estas curvas es:

$$\begin{cases} \dfrac{dx}{dt} = u, \quad \dfrac{dy}{dt} = v \\ \dfrac{du}{dt} = \dfrac{2}{y}uv, \quad \dfrac{dv}{dt} = \dfrac{1}{y}(v^2 - u^2). \end{cases}$$

Teniendo la condición $u^2 + v^2 = 1$ garantiza que t sea parámetro de arco.

Definiendo:

$$X(t) = (x(t),\ y(t),\ u(t),\ v(t))^T, \quad X(0) = (a,\ b,\ c,\ d)^T$$

$$F = (u,\ v,\ 2y^{-1}uv,\ y^{-1}(v^2 - u^2))^T,$$

Usando los símbolos de Christoffel del plano superior de Poincaré y reemplazando en el

sistema de Runge-Kuta.

Para el primer caso de $u(t)$:

$$\frac{du}{dt} + \Gamma^1_{12}uv + \Gamma^1_{21}vu = 0 \quad \text{entonces} \quad \frac{du}{dt} - \frac{1}{y}uv - \frac{1}{y}vu = 0$$

de aquí concluimos que

$$\frac{du}{dt} - \frac{2}{y}uv = 0 \quad \text{entonces} \quad \frac{du}{dt} = \frac{2}{y}uv. \tag{35}$$

por otro lado para $v(t)$

$$\frac{dv}{dt} + \Gamma^2_{11}u^2 + \Gamma^2_{22}v^2 = 0 \quad \text{entonces} \quad \frac{dv}{dt} + \frac{1}{y}u^2 - \frac{1}{y}v^2 = 0$$

de aquí concluimos que

$$\frac{dv}{dt} + \frac{1}{y}u^2 - \frac{1}{y}v^2 = 0 \quad \text{entonces} \quad \frac{dv}{dt} = \frac{1}{y}(v^2 - u^2). \tag{36}$$

De este modo podemos aplicar el método de Runge-Kutta de cuarto orden, donde el sistema diferencial que se representa en el sistema de Runge-Kuta que describe estas curvas reemplazando en las condiciones iniciales, (35) y (36) es:

$$\begin{cases} \dfrac{dx}{dt} = u, \quad \dfrac{dy}{dt} = v \\ \dfrac{du}{dt} = \dfrac{2}{y}uv, \quad \dfrac{dv}{dt} = \dfrac{1}{y}(v^2 - u^2). \end{cases}$$

Teniendo la condición de (26), $\|\dot{\eta}(t)\|_g = 1$ que garantiza que t sea parámetro de arco.

Definimos:

$$X(t) = (x(t),\ y(t),\ u(t),\ v(t))^T, \quad X(0) = (a,\ b,\ c,\ d)^T,$$

Donde $(a, b) = (x_0, y_0)$ que son las posiciones iniciales y $(c, d) = (u_0, v_0)$ las velocidades iniciales.

$$F = (u,\ v,\ 2y^{-1}uv,\ y^{-1}(v^2 - u^2))^T,$$

Podemos aplicar las fórmulas de Runge–Kutta. Para trazar aproximadamente la imagen de

una geodésica, es suficiente seleccionar los puntos

$$(x(t_\alpha),\ y(t_\alpha)) \in \mathbb{H}, \quad \alpha = 0, 1, \ldots, m.$$

Si evaluamos por ejemplo con los siguientes datos numéricos:

$$I = [0, T] = [0, 3], \qquad m = 150, \qquad h = \frac{T}{m} = 0{,}02, \qquad \mathbf{X}(0) = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 1 \\ 0{,}1 \\ 0{,}1 \\ 1 \end{bmatrix}.$$

Aplicando el algoritmo RK4: Para cada paso $t_n \longrightarrow t_{n+1} = t_n + h$,

$$\begin{aligned} \mathbf{k}_1 &= h\, F(\mathbf{X}_n), \\ \mathbf{k}_2 &= h\, F\Big(\mathbf{X}_n + \tfrac{1}{2}\mathbf{k}_1\Big), \\ \mathbf{k}_3 &= h\, F\Big(\mathbf{X}_n + \tfrac{1}{2}\mathbf{k}_2\Big), \\ \mathbf{k}_4 &= h\, F\Big(\mathbf{X}_n + \mathbf{k}_3\Big), \\ \mathbf{X}_{n+1} &= \mathbf{X}_n + \tfrac{1}{6}\big(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4\big). \end{aligned}$$

Primer paso ($n = 0 \longrightarrow 1$).

$$\mathbf{X}_0 = \begin{bmatrix} 1 \\ 0{,}1 \\ 0{,}1 \\ 1 \end{bmatrix}.$$

Calcular $\mathbf{k}_1 = h \cdot F(\mathfrak{X}_0)$

- $u_0 = 0{,}1$, $v_0 = 1$, $y_0 = 0{,}1$

$$F(X_0) = \begin{bmatrix} 0{,}1 \\ 1 \\ \frac{2(0{,}1)(1)}{0{,}1} = 2 \\ \frac{1^2 - 0{,}1^2}{0{,}1} = \frac{0{,}99}{0{,}1} = 9{,}9 \end{bmatrix} \Rightarrow \mathbf{k}_1 = 0{,}02 \cdot \begin{bmatrix} 0{,}1 \\ 1 \\ 2 \\ 9{,}9 \end{bmatrix} = \begin{bmatrix} 0{,}002 \\ 0{,}02 \\ 0{,}04 \\ 0{,}198 \end{bmatrix}$$

Calcular $\mathbf{k}_2 = h \cdot F\left(X_0 + \frac{1}{2}\mathbf{k}_1\right)$

$$X_0 + \frac{1}{2}\mathbf{k}_1 = \begin{bmatrix} 1 + 0{,}001 \\ 0{,}1 + 0{,}01 \\ 0{,}1 + 0{,}02 \\ 1 + 0{,}099 \end{bmatrix} = \begin{bmatrix} 1{,}001 \\ 0{,}11 \\ 0{,}12 \\ 1{,}099 \end{bmatrix}$$

$$F \approx \begin{bmatrix} 0{,}12 \\ 1{,}099 \\ \frac{2(0{,}12)(1{,}099)}{0{,}11} \approx 2{,}3978 \\ \frac{1{,}099^2 - 0{,}12^2}{0{,}11} \approx \frac{1{,}208 - 0{,}0144}{0{,}11} \approx 10{,}716 \end{bmatrix} \Rightarrow \mathbf{k}_2 \approx 0{,}02 \cdot \begin{bmatrix} 0{,}12 \\ 1{,}099 \\ 2{,}3978 \\ 10{,}716 \end{bmatrix} = \begin{bmatrix} 0{,}0024 \\ 0{,}02198 \\ 0{,}047956 \\ 0{,}21432 \end{bmatrix}$$

Calcular $\mathbf{k}_3 = h \cdot F\left(X_0 + \frac{1}{2}\mathbf{k}_2\right)$

$$X_0 + \frac{1}{2}\mathbf{k}_2 \approx \begin{bmatrix} 1{,}0012 \\ 0{,}11099 \\ 0{,}123978 \\ 1{,}10716 \end{bmatrix}$$

$$F \approx \begin{bmatrix} 0{,}123978 \\ 1{,}10716 \\ \frac{2(0{,}123978)(1{,}10716)}{0{,}11099} \approx 2{,}4716 \\ \frac{1{,}2258 - 0{,}01537}{0{,}11099} \approx 10{,}904 \end{bmatrix} \Rightarrow \mathbf{k}_3 \approx 0{,}02 \cdot \begin{bmatrix} 0{,}123978 \\ 1{,}10716 \\ 2{,}4716 \\ 10{,}904 \end{bmatrix} = \begin{bmatrix} 0{,}0024796 \\ 0{,}022143 \\ 0{,}049432 \\ 0{,}21808 \end{bmatrix}$$

Calcular $\mathbf{k}_4 = h \cdot F\left(X_0 + \mathbf{k}_3\right)$

$$X_0 + \mathbf{k}_3 \approx \begin{bmatrix} 1{,}0024796 \\ 0{,}122143 \\ 0{,}149432 \\ 1{,}21808 \end{bmatrix}$$

$$F \approx \begin{bmatrix} 0{,}149432 \\ 1{,}21808 \\ \frac{2(0{,}149432)(1{,}21808)}{0{,}122143} \approx 2{,}976 \\ \frac{1{,}484-0{,}0223}{0{,}122143} \approx 11{,}969 \end{bmatrix} \Rightarrow \mathbf{k}_4 \approx 0{,}02 \cdot \begin{bmatrix} 0{,}149432 \\ 1{,}21808 \\ 2{,}976 \\ 11{,}969 \end{bmatrix} = \begin{bmatrix} 0{,}0029886 \\ 0{,}0243616 \\ 0{,}05952 \\ 0{,}23938 \end{bmatrix}$$

Calcular $\mathfrak{X}_1$

$$X_1 = X_0 + \frac{1}{6}\left(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4\right)$$

$$\mathfrak{X}_1 = \begin{bmatrix} 1 \\ 0{,}1 \\ 0{,}1 \\ 1 \end{bmatrix} + \frac{1}{6}\left(\begin{bmatrix} 0{,}002 \\ 0{,}02 \\ 0{,}04 \\ 0{,}198 \end{bmatrix} + 2 \cdot \begin{bmatrix} 0{,}0024 \\ 0{,}02198 \\ 0{,}047956 \\ 0{,}21432 \end{bmatrix} + 2 \cdot \begin{bmatrix} 0{,}0024796 \\ 0{,}022143 \\ 0{,}049432 \\ 0{,}21808 \end{bmatrix} + \begin{bmatrix} 0{,}0029886 \\ 0{,}0243616 \\ 0{,}05952 \\ 0{,}23938 \end{bmatrix} \right)$$

Componente	$\mathbf{k}_1$	$\mathbf{k}_2$	$\mathbf{k}_3$	$\mathbf{k}_4$
x	0.00200000	0.00240000	0.00247956	0.00299057
y	0.02000000	0.02198000	0.02216982	0.02437248
u	0.04000000	0.04795636	0.04952832	0.05971180
v	0.19800000	0.21698198	0.21862416	0.23949507

Cuadro 1: Incrementos RK4 en el primer paso ($h = 0{,}02$).

Actualización:

$$\mathbf{X}_1 = \begin{bmatrix} 1{,}00245817 \\ 0{,}12211210 \\ 0{,}14910522 \\ 1{,}21811975 \end{bmatrix}.$$

Primeras dos iteraciones:

n	t_n	x_n	y_n	u_n	v_n
0	0.00	1.00000000	0.10000000	0.10000000	1.00000000
1	0.02	1.00245817	0.12211210	0.14910522	1.21811975
2	0.04	1.00612151	0.14902367	0.22205561	1.48111640

Cuadro 2: Evolución inicial con $h = 0{,}02$.

De este modo el método de RK4 se explica con los siguientes pasos:

1. Dividir $[0, T]$ en los nodos $t_\alpha = \alpha h,\ \alpha = 0, \ldots, m$.
2. Establecer $\mathbf{X}_0$ con las condiciones iniciales.
3. Para cada $\alpha = 0, \ldots, m-1$ aplicar las cuatro evaluaciones de F y actualizar a $\mathbf{X}_{\alpha+1}$ mediante la fórmula RK4.
4. Obtener así las 151 aproximaciones $\{\mathbf{X}_\alpha\}_{\alpha=0}^{150}$ que estiman $(x(t), y(t), u(t), v(t))$ en todo el intervalo. Así, este ejemplo inicial se desarrolla en el contexto de una geodésica que corresponde a una semicircunferencia en el plano superior de Poincaré

2.11. Distancia hiperbólica Riemanniana

Finalmente, la distancia entre dos puntos (t_1, t_2) y (s_1, s_2) se puede obtener para los dos casos: $t_1 = s_1$ y $t_1 \neq s_1$. En el primer caso en que $t_1 = s_1$, la distancia se expresa mediante

$$dis((t_1, t_2),\ (s_1, s_2)) = \int_{t^0}^{t^1} \frac{\sqrt{x'(t)^2 + y'(t)^2}}{y(t)}\, dt$$

donde $(x(t^0), y(t^0)) = (t_1, t_2)$ y $(x(t^1), y(t^1)) = (s_1, s_2)$. Dado que $x(t) = t_1 = s_1$, por lo tanto, cuando $x'(t) = 0$, la ecuación anterior se simplifica a:

$$dis((t_1, t_2), (t_1, s_2)) = \int_{t^0}^{t^1} \frac{y'(t)}{y(t)}\, dt = \ln\left|\frac{s_2}{t_2}\right| = \ln\frac{s_2}{t_2}.$$

La última igualdad se da porque $y > 0$.

De manera similar, para el caso $t_1 \neq s_1$, si se considera la parametrización:

$$x(t) = \frac{a}{2} + r\cos(t) \quad \text{y} \quad y(t) = r\sin(t), \quad \text{donde} \quad t \in (0, \pi) \text{ (de tal manera que } y(t) > 0),$$

b y r están definidas anteriormente. Entonces, la ecuación queda:

$$dis((t_1, t_2), (s_1, s_2)) = \int_{t^0}^{t^1} \frac{\sqrt{r^2}}{r\sin(t)}\, dt = \int_{t^0}^{t^1} \frac{1}{\sin(t)}\, dt = \ln\left|\frac{\cot(t^0) + \csc(t^0)}{\cot(t^1) + \csc(t^1)}\right|.$$

Por definición: $\cot(t) = \dfrac{\cos(t)}{\sin(t)}$ y $\csc(t) = \dfrac{1}{\sin(t)}$, además como $\cos^2(t) + \sin^2(t) = 1$, $y(t) = r\sin(t) > 0$, $\sin(t) \leq 1$, y en consecuencia $\dfrac{1}{\sin(t)} \geq 1$. Entonces:

$$dis((t_1, t_2), (s_1, s_2)) = \ln\left(\frac{\sqrt{\dfrac{1}{\sin^2(t^0)} - 1} + \dfrac{1}{\sin(t^0)}}{\sqrt{\dfrac{1}{\sin^2(t^1)} - 1} + \dfrac{1}{\sin(t^1)}}\right) = \operatorname{arcosh}\left(\frac{1}{\sin(t^0)}\right) - \operatorname{arcosh}\left(\frac{1}{\sin(t^1)}\right).$$

Debido a la parametrización para x e y, entonces:

$$dis((t_1, t_2), (s_1, s_2)) = \operatorname{arcosh}\left(\frac{r}{t_2}\right) - \operatorname{arcosh}\left(\frac{r}{s_2}\right).$$

Entre algunas de las identidades de las funciones trigonométricas hiperbólicas, se tiene que $\operatorname{arcosh}(x) + \operatorname{arcosh}(y) = \operatorname{arcosh}\left(xy + \sqrt{(x^2 - 1)(y^2 - 1)}\right)$. Es así que:

$$dis((t_1, t_2), (s_1, s_2)) = \operatorname{arcosh}\left(\frac{r^2}{t_2 s_2}\left(\sqrt{\frac{r^2}{t_2^2} - 1}\right)\left(\frac{r^2}{s_2^2} - 1\right)\right).$$

Al sustituir el valor de r y simplificar los términos iguales, se determina que la distancia entre los puntos $(t_1, t_2), (s_1, s_2)$ es:

$$dis((t_1, t_2), (s_1, s_2)) = \operatorname{arcosh}\left(1 + \frac{(s_1 - t_1)^2 + (s_2 - t_2)^2}{2t_2 s_2}\right).$$

III. Matrices simétricas definidas positivas

Para este capítulo tomamos como literatura los trabajos de Ansari et al.[3] y Nesterov et al.[20], de este modo mostramos los detalles y los cálculos respectivos de la gradiente(caso Euclideano y Riemanniano) al igual que el subgradiente de Greemberg Pierskalla en la variedad SPD de la función definida como: $f = ln^2(det(X)) - ln(det(X))$.

Sea $\mathbb{P}^n$ el conjunto de matrices simétricas y $\mathbb{P}^n_{++}$ el conjunto de matrices simétricas definidas positivas. Sea $\mathfrak{M} = (\mathbb{P}^n_{++}, (\cdot,\cdot))$ la variedad Riemanniana dotada de la métrica Riemanniana dada por

$$\langle U, V\rangle_X = \mathrm{tr}\big(V\, X^{-1}\, U\, X^{-1}\big),$$

para toda $X \in \mathfrak{M}$ y $U, V \in T_X\mathfrak{M} \simeq \mathbb{P}^n$, donde $\mathrm{tr}(A)$ denota la traza de la matriz $A \in \mathbb{P}^n$.

3.0.1. Geodésica en la variedad SPD

El segmento geodésico único que une $X, Y \in \mathfrak{M}$ está dado por

$$\eta(t) = X^{1/2}\left(X^{-1/2}\, Y\, X^{-1/2}\right)^t X^{1/2}, \quad t \in [0,1].$$

La aplicación exponencial $\exp_X : T_X\mathfrak{M} \to \mathfrak{M}$ y su inversa $\exp_X^{-1} : \mathfrak{M} \to T_X\mathfrak{M}$ están dadas, respectivamente, por

$$\exp_X V = X^{1/2}\, \exp\big(X^{-1/2}\, V\, X^{-1/2}\big)\, X^{1/2}$$

y

$$\exp_X^{-1} Y = X^{1/2}\, \ln\big(X^{-1/2}\, Y\, X^{-1/2}\big)\, X^{1/2}.$$

3.0.2. Distancia definida en SPD

La distancia Riemanniana d definida en $\mathfrak{M} = \mathbb{P}^n_{++}$ está dada por

$$d^2(X,Y) = \mathrm{tr}\Big(\ln^2\big(X^{-1/2}\, Y\, X^{-1/2}\big)\Big), \quad \forall X, Y \in \mathbb{P}^n_{++}.$$

Nótese que $\mathfrak{M} := (\mathbb{P}^n_{++}, \langle\cdot,\cdot\rangle)$ es una variedad Riemanniana con curvatura seccional no positiva.

Ejemplo 3.0.1 *Considere el siguiente problema de optimización:*

$$\min_{X \in P_{++}^n} f(X)$$

Donde la función $f : \mathbb{P}_{++}^n \to \mathbb{R}$ *definida por* $f(X) = \ln^2 \det X - \ln \det X$ *En nuestro caso, la función*

$$f(X) = \ln^2 \det X - \ln \det X$$

Analizaremos la convexidad de f *tanto en el contexto Euclidiano como en el Riemanniano.*

Calculamos el gradiente y el hessiano Euclidianos de f.

y deseamos calcular su gradiente $\nabla f(X)$ *y su hessiano aplicado a una dirección* $V \in \mathbb{R}^{n \times n}$, *es decir,* $\nabla^2 f(X)[V]$.

3.0.3. Gradiente Euclideano de $f(X)$

Definimos la función auxiliar $g(X) := \ln \det X$. *Entonces:*

$$f(X) = g(X)^2 - g(X).$$

Aplicamos la regla de la cadena para funciones escalares:

$$\nabla f(X) = 2g(X)\nabla g(X) - \nabla g(X) = (2 \ln \det X - 1)\nabla g(X).$$

donde se sabe que:

$$\nabla g(X) = \nabla \ln \det X = X^{-1},$$

se obtiene:

$$\boxed{\nabla f(X) = (2 \ln \det X - 1)X^{-1}}.$$

3.0.4. Hessiano aplicado a una dirección V

Sea $a(X) := 2 \ln \det X - 1$ *y* $b(X) := X^{-1}$, *de modo que:*

$$\nabla f(X) = a(X)b(X).$$

Aplicamos la regla del producto para derivadas direccionales de funciones matriciales:

$$\nabla^2 f(X)[V] = \nabla a(X)[V] \cdot b(X) + a(X) \cdot \nabla b(X)[V].$$

Dado que $a(X) = 2 \ln \det X - 1$*, se tiene:*

$$\nabla a(X)[V] = 2 \cdot \nabla(\ln \det X)[V].$$

Usando la derivada direccional:

$$\nabla(\ln \det X)[V] = tr(X^{-1}H),$$

obtenemos:

$$\boxed{\nabla a(X)[V] = 2 \cdot tr(X^{-1}H)}.$$

La derivada direccional de la inversa es:

$$\nabla b(X)[V] = \nabla(X^{-1})[V] = -X^{-1}HX^{-1}.$$

entonces de esto obtenemos:

$$\begin{aligned}\nabla^2 f(X)[V] &= \nabla a(X)[V] \cdot b(X) + a(X) \cdot \nabla b(X)[V] \\ &= 2tr(X^{-1}H)X^{-1} - (2 \ln \det X - 1)X^{-1}HX^{-1}.\end{aligned}$$

Por lo tanto, el hessiano aplicado a una dirección H es:

$$\boxed{\nabla^2 f(X)[V] = 2 \cdot tr(X^{-1}H)X^{-1} - (2 \ln \det X - 1)X^{-1}HX^{-1}}.$$

Para analizar la convexidad, restringimos a la curva $X = tI_n$*, para* $t \in \mathbb{R}_{++}$*, y tomamos* $V = I_n$*. Entonces tenemos:*

$$\ln \det(tI_n) = n \ln t, \qquad X^{-1} = \frac{1}{t} I_n, \qquad \operatorname{tr}(X^{-1}V) = \frac{n}{t}.$$

Sustituyendo en la expresión del hessiano obtenemos:

$$\begin{aligned}\nabla^2 f(tI_n)[I_n] &= 2 \cdot \frac{n}{t} \cdot \frac{1}{t} I_n - [2n \ln t - b] \cdot \frac{1}{t^2} I_n \\ &= \left[\frac{2n}{t^2} - \frac{1}{t^2}(2n \ln t - b) \right] I_n \\ &= \frac{1}{t^2}[2n - 2n \ln t + b] I_n.\end{aligned}$$

Para $t \geq 1$, el término logarítmico $\frac{1}{t^2}[2n - 2n\ln t + 1] < 0$, el coeficiente se vuelve negativo. Por lo tanto, el hessiano Euclidiano no es definido positivo para todo t, y concluimos que f no es convexa en el sentido Euclidiano.

El punto crítico de f se obtiene resolviendo $\nabla f(X) = 0$. Dado que $\nabla f(X) = [2\ln\det(X) - 1]X^{-1}$, se debe tener $2\ln\det(X) - 1 = 0$, es decir,

$$\det X = e^{1/2}. \tag{37}$$

Esto implica que cualquier $X \in \mathbb{P}_n^{++}$ cuya determinante sea igual a $e^{1/2}$ es un punto crítico de f.

3.0.5. Gradiente y Hessiano en $\mathbb{P}_{++}^n$

La conexión de Levi-Civita asociada a esta métrica se expresa:

$$\nabla_V W = DW(X)[V] - \tfrac{1}{2}\Big(V\,X^{-1}W + W\,X^{-1}V\Big),$$

donde $DW(X)[V]$ representa la derivada direccional de W en la dirección de V.

De esta manera calcularemos la gradiente con la métrica definida en la variedad de las matrices definidas positivas.

El gradiente obtenido anteriormente en el sentido Euclidiano es

$$\nabla_{\text{Euc}} f(X) = \Big(2a\ \ln(\det X) - b\Big)X^{-1}.$$

De esta manera, el gradiente Riemanniano $\operatorname{grad} f(X)$ *se define mediante la relación*

$$Df(X)[V] = g_X\Big(\operatorname{grad} f(X), V\Big) = \operatorname{tr}\Big(X^{-1}\operatorname{grad} f(X)\,X^{-1}V\Big) \quad \forall V.$$

Donde g_X es la metrica definida anteriormente $g_X(U,V) = \langle U, V\rangle_X = \operatorname{tr}\big(V\,X^{-1}\,U\,X^{-1}\big)$.

Comparando con la expresión Euclidiana, se obtiene la siguiente relación:

$$X^{-1}\operatorname{grad} f(X)\,X^{-1} = \nabla_{\text{Euc}} f(X).$$

De aquí se deduce que

$$\operatorname{grad} f(X) = X\,\nabla_{\text{Euc}} f(X)\,X.$$

Sustituyendo la expresión de $\nabla_{\text{Euc}} f(X)$*:*

$$\operatorname{grad} f(X) = X\Big[\big(2a \cdot \ln\det X - b\big)X^{-1}\Big]X = \big(2a \cdot \ln\det X - b\big)X.$$

3.0.6. Cálculo de la Hessiana Riemanniana

La hessiana Riemanniana se define, para $V \in T_X\mathbb{P}_n^{++}$*, como*

$$\operatorname{Hess} f(X)[V] = \nabla_V\Big(\operatorname{grad} f\Big)(X).$$

3.0.7. Derivada Direccional del Gradiente

Partimos de la expresión:

$$\operatorname{grad} f(X) = \big(2a\ \ln(\det X) - b\big)X.$$

Aplicamos la regla del producto para obtener la derivada direccional en la dirección V*:*

$$D\Big[\operatorname{grad} f(X)\Big][V] = D\Big[\big(2a\ \ln(\det X) - b\big)X\Big][V] = \underbrace{(2a\, D[\ln(\det X)][V])\, X}_{(I)} + \underbrace{\big(2a\ \ln(\det X) - b\big)V}_{(II)}.$$

Recordemos que

$$D[\ln(\det X)][V] = \operatorname{tr}\big(X^{-1}V\big),$$

la parte (I) se convierte en:

$$(I) = 2a\ \operatorname{tr}(X^{-1}V)\, X.$$

De aquì tenemos,

$$D\Big[\operatorname{grad} f(X)\Big][V] = 2a\ \operatorname{tr}(X^{-1}V)\, X + \big(2a\ \ln(\det X) - b\big)V.$$

Aplicando la definiciòn de la Conexión de Levi-Civita

Utilizando la fórmula para la conexión de Levi-Civita, se tiene:

$$\nabla_V\Big(\operatorname{grad} f\Big)(X) = D\Big[\operatorname{grad} f(X)\Big][V] - \tfrac{1}{2}\Big(V\, X^{-1} \operatorname{grad} f(X) + \operatorname{grad} f(X)\, X^{-1}V\Big).$$

Observamos que:

$$V\,X^{-1}\,\mathrm{grad}\,f(X) = V\,X^{-1}\Big[\big(2a\,\ln\det X - b\big)X\Big] = \big(2a\,\ln\det X - b\big)V,$$

y de igual forma,

$$\mathrm{grad}\,f(X)\,X^{-1}V = \big(2a\,\ln(\det X) - b\big)V.$$

Por lo tanto,

$$\tfrac{1}{2}\Big(V\,X^{-1}\,\mathrm{grad}\,f(X) + \mathrm{grad}\,f(X)\,X^{-1}V\Big) = \big(2a\,\ln(\det X) - b\big)V.$$

Sustituyendo en la definición de la derivada covariante obtenemos:

$$\begin{aligned}\mathrm{Hess}\,f(X)[V] &= \Big[2a\,\mathrm{tr}(X^{-1}V)\,X + \big(2a\,\ln(\det X) - b\big)V\Big] - \big(2a\,\ln(\det X) - b\big)V \\ &= 2a\,\mathrm{tr}(X^{-1}V)\,X.\end{aligned}$$

Así, la hessiana Riemanniana es:

$$\boxed{\mathrm{Hess}\,f(X)[V] = 2a\,\mathrm{tr}(X^{-1}V)\,X, \quad \forall\, V \in T_X\mathbb{P}_n^{++}.}$$

entonces el producto interno es

$$\begin{aligned}g_X\Big(\mathrm{Hess}\,f(X)[V],\,V\Big) &= \mathrm{tr}\Big(X^{-1}\Big[2a\,\mathrm{tr}(X^{-1}V)\,X\Big]X^{-1}V\Big) \\ &= 2a\,\mathrm{tr}(X^{-1}V)\,\mathrm{tr}(X^{-1}V) \\ &= 2a\,\Big[\mathrm{tr}(X^{-1}V)\Big]^2.\end{aligned}$$

$$\boxed{2a\,\Big[\mathrm{tr}(X^{-1}V)\Big]^2}$$

Esta cantidad es siempre no negativa para todo V, por lo que concluimos que f es geodésicamente convexa con respecto a la métrica en la variedad $\mathbb{P}_n^{++}$.

IV. Métodos subgradientes en variedades Riemannianas

En el caso que f sea diferenciable, es posible obtener el método gradiente en variedades Riemannianas.
Por razones de brevedad, en este trabajo se presentará únicamente una parte del algoritmo gradiente en variedades Riemannianas y los tamaños de paso (Armijo y Lipchitz), invitamos al lector a revisar el trabajo de Grohs [13] para profundizar.

Algorithm 1 Método del gradiente en la variedad Riemanniana $\mathfrak{M}$

1: **Entrada:** Punto inicial $p_0 \in \mathfrak{M}$, tolerancia $\epsilon > 0$, máximo de iteraciones K
2: **Inicializar:** $k \leftarrow 0$
3: **while** $\| \operatorname{grad} f(p_k)\| > \epsilon$ y $k < K$ **do**
4: Escoger $t_k > 0$ (según una estrategia de tamaño de paso)
5: $p_{k+1} \leftarrow \exp_{p_k}(-t_k \operatorname{grad} f(p_k))$
6: $k \leftarrow k + 1$
7: **Retornar:** p_k

A continuación, presentamos las dos estrategias más comúnmente utilizadas para escoger el tamaño de paso $t_k > 0$ en el método del gradiente.

Tamaño de paso de Lipschitz

Supongamos que grad f es Lipschitz continua con constante $L > 0$. Sea $\varepsilon > 0$ y tomamos

$$\varepsilon < t_k \leq \frac{1}{L}. \tag{18.5}$$

4.0.8. Tamaño de paso de Armijo

Elegir $\delta \in (0, 1)$ y tomar

$$t_k := \max\left\{2^{-i} : f\left(\eta_k(2^{-i})\right) \leq f(p_k) - \delta 2^{-i} \| \operatorname{grad} f(p_k)\|^2, \quad i = 0, 1, \ldots\right\},$$

donde $\eta_k(2^{-i}) := \exp_{p_k}\left(-2^{-i} \operatorname{grad} f(p_k)\right)$.

4.1. Método subgradiente para funciones convexas

A seguir presentamos el método subgradiente para funciones convexas en variedades Riemannianas desarrollado por Ferreira y Oliveira. Bajo ciertas hipótesis y condiciones, se demuestra la convergencia (ver capítulo 5 de [11]). De este modo, definiremos el subgradiente para el caso de funciones convexas en una variedad Riemanniana.

Definición 4.1.1 *Sea f una función convexa y $x \in M$. Un vector $s \in T_xM$ se dice que es un* ***subgradiente*** *de f en x si, para cualquier geodésica η de M con $\eta(0) = x$, se cumple que*

$$(f \circ \eta)(t) \geq f(x) + t\langle s, \eta'(0)\rangle,$$

para todo $t \geq 0$.

El conjunto de todos los subgradientes de f en x, denotado por $\partial f(x)$, se llama el ***subdiferencial*** *de f en x.*

4.1.1. Definición del Problema y el Algoritmo del método subgradiente para funciones convexas

Sea $\mathfrak{M}$ una variedad Riemanniana (conexa) completa, y sea f una función convexa. El conjunto Φ^* denota el conjunto de minimizadores de f, y

$$f^* = \inf_{x \in M} f(x)$$

denota su valor ínfimo. El problema consiste en estimar f^* y también en encontrar un punto de Φ^*, si dicho punto existe, es decir, si $\Phi^* \neq \emptyset$.

Algorithm 2 Algoritmo Conceptual de Subgradiente

Entrada: Secuencia $\{t_k\}$ con $t_k > 0$ para $k = 1, 2, \ldots$.
Inicialización: Elegir $x_1 \in \mathfrak{M}$ y obtener $s_1 \in \partial f(x_1)$. Hacer $k = 1$.
while $s_k \neq 0$ **do**
Calcular la geodésica η_{v_k} con

$$\eta_{v_k}(0) = x_k, \quad \eta'_{v_k}(0) = v_k, \quad v_k = -\frac{s_k}{\|s_k\|}.$$

Actualizar $x_{k+1} = \eta_{v_k}(t_k)$.
Obtener $s_{k+1} \in \partial f(x_{k+1})$.
Hacer $k = k + 1$.
Salida: Punto crítico x_k donde $s_k = 0$.

A continuación, presentamos las dos estrategias más comúnmente utilizadas para escoger el tamaño de paso $t_k > 0$ en el método subgradiente.

Tamaño de paso exógeno

$$t_k > 0, \quad \sum_{k=0}^{\infty} t_k = +\infty, \quad \sigma := \sum_{k=0}^{\infty} t_k^2 < +\infty. \tag{18.13}$$

Un valor de t_k que cumple estas condiciones es $t_k = \dfrac{1}{k}$.
El método del subgradiente con el tamaño de paso exógeno ha sido analizado en varios artículos; véase, por ejemplo, [7, 11, 27].

Tamaño de paso de Polyak

Recordamos que $f : \mathfrak{M} \to \mathbb{R}$ es una función convexa, donde $\mathfrak{M}$ es una variedad Riemanniana con curvatura seccional $K \geq \kappa$ y $\kappa \leq 0$. Recordamos que Φ^*, el conjunto de soluciones del Problema, se asumirá no vacío solo cuando se indique explícitamente.

Supóngase que $p_0 \in \mathfrak{M}$, $\Phi^* \neq \emptyset$ y definamos

$$t_k = \alpha \frac{f(p_k) - f^*}{\|s_k\|}, \quad 0 < \alpha < 2\frac{\tanh(\sqrt{\kappa}|d_0|)}{\sqrt{\kappa}|d_0|}, \quad d_0 := d(p_0, \Phi^*), \tag{18.14}$$

donde

$$d(p_0, \Phi^*) := \text{ínf}\{d(p_0, q) : q \in \Phi^*\} > 0,$$

y $\kappa \neq 0$, y $0 < \alpha < 2$ para $\kappa = 0$. □

4.2. Método subgradiente para funciones cuasi-convexas

Definición 4.2.1 *Sea $\mathfrak{M}$ una variedad completa de Riemann, y considere $f : M \longrightarrow \mathbb{R}$ una función real. La función f se denomina cuasi-convexa en $\mathfrak{M}$ si, para cualesquiera $x, y \in \mathfrak{M}$ y $t \in [0, 1]$, se cumple:*

$$f(\eta(t)) \leq max\{f(x), f(y)\},$$

para toda curva geodésica $\eta : [0, 1] \longrightarrow \mathfrak{M}$, tal que $\eta(0) = x$ y $\eta(1) = y$.

Definición 4.2.2 *Una función $f : C \subseteq M \to \mathbb{R}$ se dice que satisface la condición de Hölder*

con sonstante $L > 0$ y grado β en $x \in C$ si :

$$|f(x) - f(y)| \leqslant L||x - y||^{\beta} \text{ para cualquier } y \in C$$

Diremos que f satisface la condición de Hölder en $N \subseteq C$ sobre C , si la desigualdad anterior es válida para todo $x \in N$.
En particular si $\beta = 1$ y se satisface la desigualdad anterior para toda $c \in C$ entonces f es Lipschitziana en C.

Proposición 4.2.1 *Sea $\mathfrak{M}$ una variedad Riemanniana completa con una curvatura seccional $K \geq 0$. Si η_{u_1} y η_{u_2} son dos geodésicas normalizadas tal que $\eta_{u_1}(0) = \eta_{u_2}(0)$ entonces*

$$d\left(\eta_{u_1}(r_1), \eta_{u_2}(r_2)\right) \leq ||r_2u_2 - r_1u_1||$$

Teorema 4.2.2 *[23] Sea C un subconjunto geodésico convexo de $\mathbf{M}$. Entonces la función $f : C \to \mathbb{R}$ es cuasi-convexa geodésica si y solo si el conjunto de nivel sub $S_{f,\alpha}$ es convexo geodésico para cada $\alpha \in \mathbb{R}$.*

Para cualquier $\alpha \in \mathbb{R}$, el conjunto de nivel sub de la función $f : C \to \mathbb{R}$ se denota por

$$S_{f,\alpha} := \{y \in \mathbf{M} : f(y) < \alpha\}.$$

Por simplicidad, denotamos $S_f(x) := S_{f,f(x)}$.

Definición 4.2.3 *Una función $f : \mathbf{M} \to \mathbb{R}$ es semi-continua inferior en $x \in \mathbf{M}$ si*

$$f(x) \leq \liminf_{n\to\infty} f(x_n) \quad \textit{siempre que} \quad x_n \to x \quad \textit{cuando} \quad n \to \infty.$$

Además, f es semi-continua inferior en $\mathbf{M}$ si es semi-continua inferior en cada punto de $\mathbf{M}$. La función f es semi-continua superior en $\mathbf{M}$ si $-f$ es semi-continua inferior en $\mathbf{M}$.

Observación 1. Como en el caso de los espacios vectoriales, una función $f : \mathbf{M} \to \mathbb{R}$ es superior semi-continua si y solo si el conjunto de nivel superior $S_{f,\alpha}$ es abierto.

Sea C un subconjunto geodésico convexo de $\mathbf{M}$ y $f : C \to \mathbb{R}$ una función convexa geodésica. Entonces, el subdiferencial ∂f de f en x se define como

$$\partial f(x) := \left\{u \in T_x\mathbf{M} : \langle u, \eta'_{xy}(0)\rangle \leq f(y) - f(x), \forall y \in \mathbf{M}\right\}.$$

Se sabe que $\partial f(x)$ es un conjunto no vacío, cerrado y convexo geodésico, ver por ejemplo . El siguiente ejemplo ilustra que $\partial f(x)$ podría estar vacío si la función f es cuasi-convexa

geodésica. Sea $\mathfrak{M}$ una variedad Riemanniana y $f : \mathfrak{M} \to \mathbb{R}$ una función cuasi-convexa geodésica.

4.2.1. Definición del Problema y el Algoritmo para funciones cuasi convexas

Para esta subsecciòn estamos usando el trabajo de Ansari y Qanul et.al [3], para el caso particular de $m = 1$, particularizamos cuando f es cuasi-convexa El cuasi-subdiferencial de Greenberg-Pierskalla de f en $x \in \mathfrak{M}$ se define por

$$\partial^{GP} f(x) := \left\{ u \in T_x\mathfrak{M} : \langle u, \dot{\eta}_{xy}(0) \rangle < 0, \ \forall y \in S_f(x) \text{ y } \forall \eta_{xy} \in \eta_{xy} \right\}.$$

Donde para cualquier $\alpha \in \mathbb{R}$, el conjunto subnivel de la función $f : \mathfrak{M} \to \mathbb{R}$ se denota por

$$S_{f,\alpha} := \{ y \in \mathfrak{M} : f(y) < \alpha \}.$$

Por simplicidad, escribimos $S_f(x) := S_{f,f(x)}$.

Algorithm 3 Sea $\{t_k\}$ una secuencia de números reales tales que $t_k > 0$ para todo$k \in \mathbb{N}$

Paso 1: Elija $x_0 \in \mathfrak{M}$ y fijamos $k = 0$.

Paso 2: Si $f(x^k) = f^*$ se detiene el algoritmo, de lo contrario, si $f(x^k) > f^*$ y pase al Paso 3.

Paso 3: Elija

$$u^k \in \partial^{GP} f(x^k)$$

y definamos un $\tilde{u}^k = u^k / ||u^k||$ y calcule

$$x^{k+1} = \exp_{x^k}\left(-t_k \, \tilde{u}^k\right).$$

Paso 4: reemplace k por $k + 1$ y regrese al Paso 2.

Estudiamos la convergencia de la sucesión generada por el Algoritmo 2 con la siguiente regla de tamaño de paso exógeno:

$$\sum_{k=0}^{\infty} t_k = \infty \quad \text{y} \quad \sum_{k=0}^{\infty} t_k^2 < \infty.$$

4.2.2. Convergencia del Algoritmo de subgradiente de Greemberg-Pierskalla

Por razones prácticas, incluimos la prueba llevada a cabo por Ansari y Uddin et al. [5]. Considere $\mathfrak{M}$ como una variedad Riemanniana completa, donde

$$f : \mathfrak{M} \to \mathbb{R}$$

una función geodésicamente cuasi–convexa. Consideramos el siguiente problema de optimización para f en $\mathfrak{M}$:

$$\min_{x \in \mathfrak{M}} \; f(x) \quad \text{sujeto a} \quad x \in \mathfrak{M}. \tag{38}$$

El resto de este trabajo adopta las siguientes notaciones:

1. $f^* = \min_{x \in \mathfrak{M}} f(x) \quad \text{y} \quad \Phi^* = \arg\min_{x \in \mathfrak{M}} f(x)$;

asumimos que se satisfacen la condición de Holder de orden β.

Lemma 4.2.3 (Lema 2.6 [5]) *Sean $\{a_k\}$ y $\{b_k\}$ dos sucesiones de números reales tales que*

$$b_k > 0 \quad \forall k, \qquad \sum_{j=1}^{k} b_j \;\longrightarrow\; \infty \quad \textit{cuando } k \to \infty.$$

Entonces se cumple:

$$\liminf_{k\to\infty} a_k \;\le\; \liminf_{k\to\infty} \frac{\sum_{j=1}^{k} b_j\, a_j}{\sum_{j=1}^{k} b_j} \;\le\; \limsup_{k\to\infty} \frac{\sum_{j=1}^{k} b_j\, a_j}{\sum_{j=1}^{k} b_j} \;\le\; \limsup_{k\to\infty} a_k.$$

En particular, si existe a tal que $\lim_{k\to\infty} a_k = a$, entonces

$$\lim_{k\to\infty} \frac{\sum_{j=1}^{k} b_j\, a_j}{\sum_{j=1}^{k} b_j} = a.$$

Lemma 4.2.4 (Lema 4.3 [5]) *Consideremos $\mathfrak{M}$ como una Variedad Riemanniana completa con curvatura seccional $K \ge \kappa$. Sea $\{x_k\}$ la sucesión producida mediante el Algoritmo 4.2 con el tamaño de paso especificado en (16). Bajo estas condiciones, para cualquier $z \in \Phi^*$ y para cada $k \in \mathbb{N}_+$,*

$$d\big(\Phi_{i,k},\, z\big) \;\le\; \frac{1}{\hat{\kappa}} \cosh^{-1}\!\Big(\cosh\big(\hat{\kappa}\, d(x_0, z)\big)\, e^{\frac{\hat{\kappa}}{\sqrt{\sigma}} \sinh\left(\hat{\kappa}\sqrt{\sigma}\right)/2}\Big), \quad \forall\, i = 1, 2, \ldots, m.$$

Lemma 4.2.5 (Lema 4.4 [5]) *Si $\mathfrak{M}$ es una Variedad Riemanniana completa cuya curvatura seccional satisface $K \ge \kappa$, y consideremos $\{x_k\}$ como una sucesión producida por el Algoritmo 4.2, empleando el tamaño de paso especificado en la ecuación (16). Entonces, para*

cualquier $z \in \Phi^*$ y cada $k \in \mathbb{N}_+$,

$$d^2(x_{k+1}, z) \;\leq\; d^2(x_k, z) \;+\; m^2 D^2_{\sigma,\hat{\kappa}}\, t_k^2 \;-\; 4m\, t_k\, Z_{\beta,m}\left(f(x_k) - f^*\right)^{\frac{1}{\beta}},$$

donde

$$Z_{\beta,m} = \left(2m\, L_{\text{máx}}\right)^{-\frac{1}{\beta}}.$$

Teorema 4.2.6 (Teorema 4.5 [5]) *Sea $\mathfrak{M}$ una variedad Riemanniana completa con curvatura seccional $K > \kappa$, y sea $\{x_k\}$ una sucesión generada por el Algoritmo 4.2 con el tamaño de paso definido en (16). Entonces se cumple:*
Sea $\{x_k\}$ la sucesión generada por el Algoritmo 4.2. Entonces

$$\liminf_{k\to\infty} f(x_k) = f^*, \quad y \quad x_k \to x^* \in \Phi^*.$$

Demostración 4.2.1 (Demostración teorema 4.5 [5]) *Sea $x^* \in \Phi^*$ arbitrario. A partir del Lema 4.4, para $z = x^*$ y sumando sobre $k = 1, 2, \ldots, n$, obtenemos*

$$d^2(x_{n+1}, x^*) \;\leq\; d^2(x_0, x^*) \;+\; m^2 D^2_{\sigma,\hat{\kappa}} \sum_{k=0}^{n} t_k^2 \;-\; 4mZ_{\beta,m} \sum_{k=0}^{n} t_k \left(f(x_k) - f^*\right)^{\frac{1}{\beta}}. \tag{39}$$

Y por lo tanto,

$$\frac{\displaystyle\sum_{k=0}^{n} t_k \left(f(x_k) - f^*\right)^{\frac{1}{\beta}}}{\displaystyle\sum_{k=0}^{n} t_k} \;\leq\; \frac{d^2(x_0, x^*)}{4mZ_{\beta,m}\ \sum_{k=0}^{n} t_k} \;+\; \frac{m\, D^2_{\sigma,\hat{\kappa}}\ \sum_{k=0}^{n} t_k^2}{4\, Z_{\beta,m}\ \sum_{k=0}^{n} t_k}. \tag{40}$$

Como $\sum_{k=0}^{\infty} t_k = \infty$, se tiene

$$\lim_{n\to\infty} \frac{d^2(x_0, x^*)}{\sum_{k=0}^{n} t_k} = 0.$$

De este modo, aplicando el Lema 2.6 (con $(f(x_k) - f^*)^{\frac{1}{\beta}}$ en lugar de a_k) y la desigualdad (40), obtenemos

$$\liminf_{k\to\infty} \left(f(x_k) - f^*\right)^{\frac{1}{\beta}} \;\leq\; \liminf_{n\to\infty} \frac{\displaystyle\sum_{k=0}^{n} t_k \left(f(x_k) - f^*\right)^{\frac{1}{\beta}}}{\displaystyle\sum_{k=0}^{n} t_k} \;\leq\; 0.$$

Esto demuestra la afirmación deseada.

(b) Del Lema 4.2.5 se tiene

$$d^2(x_{k+1}, z) \;\leq\; d^2(x_k, z) \;+\; m^2 D^z_{\sigma,\hat{k}}\, t_k^2, \quad \forall\, z \in \Phi^*.$$

Como $\sum_{k=0}^{\infty} t_k^2 < \infty$, la sucesión $\{x_k\}$ es cuasi–Fejér convergente a Φ^*. De este modo , $\{x_k\}$ está acotada. Sea $\bar{x}$ un punto de acumulación de $\{x_k\}$ y $\{x_{k_j}\}$ una sub–sucesión tal que

$$\lim_{j\to\infty} x_{k_j} = \bar{x}.$$

Por la parte (a), existe además una sub–sucesión $\{f(x_{k_j})\}$ con

$$\lim_{j\to\infty} f(x_{k_j}) = f^*.$$

Entonces, por continuidad de f,

$$f(\bar{x}) = \lim_{j\to\infty} f(x_{k_j}) = f^*,$$

y por tanto $\bar{x} \in \Phi^*$ la sucesión $\{x_k\}$ converge a una solución del problema (38).

V. Análisis de subdiferenciales para funciones matriciales en el espacio Euclidiano y en la variedad SPD

5.0.3. Calculo del subdiferencial convexo en la variedad SPD

Sea $f\colon \mathcal{M} \to \mathbb{R}$ una función en una variedad Riemanniana $\mathcal{M}$. Su *subdiferencial clásico* en $X \in \mathcal{M}$ se define como

$$\partial f(X) := \left\{ u \in T_X\mathcal{M} \;\middle|\; f\big(\eta(t)\big) \geq f(X) + t\big\langle u, \dot{\eta}(0)\big\rangle_X,\ \forall\, \eta \text{ geodésica con } \eta(0) = X \right\}.$$

Como $f \in C^1$, la diferencial $Df_X\colon T_X\mathcal{M} \to \mathbb{R}$ es lineal y continua, tal que

$$Df_X[v] = \big\langle \nabla f(X),\, v\big\rangle_X \quad \forall\, v \in T_X\mathcal{M}.$$

Sea $\eta\colon [0, \varepsilon) \to \mathcal{M}$ una geodésica con $\eta(0) = X$ y $v = \dot{\eta}(0)$.
Tenemos

$$\left.\frac{d}{dt}\right|_{t=0^+} f\big(\eta(t)\big) = Df_X[v] = \big\langle gradf(X),\, v\big\rangle_X.$$

La convexidad en t implica que para todo $0 \leq t \leq 1$,

$$f\big(\eta(t)\big) \geq f(X) + t\, Df_X[v] = f(X) + t\,\big\langle gradf(X),\, v\big\rangle_X,$$

. Supongamos $u \in \partial f(X)$. Entonces para toda geodésica η con velocidad inicial v,

$$f\big(\eta(t)\big) \geq f(X) + t\,\langle u, v\rangle_X, \quad \forall\, t \geq 0.$$

Dividiendo por $t > 0$ y tomando límite $t \to 0^+$,

$$Df_X[v] = \lim_{t\to 0^+} \frac{f(\eta(t)) - f(X)}{t} \geq \langle u, v\rangle_X.$$

Pero $Df_X[v] = \langle gradf(X), v\rangle_X$, luego $\langle gradf(X) - u, v\rangle_X \geq 0$ para todo v. Aplicando la misma desigualdad con $-v$ se concluye $\langle gradf(X) - u, v\rangle_X = 0$ para todo v, es decir, $u = gradf(X)$.

5.0.4. Calculo del subdiferencial de Greenberg–Pierskalla de $f(X) = (\ln\det X)^2 - \ln\det X$ en el espacio Euclídeo

Sea $f : \mathcal{S}^n \to \mathbb{R}$,

$$f(X) = \big(\ln\det X\big)^2 - \ln\det X,$$

definida sobre el espacio euclídeo de matrices simétricas con producto interno $\langle U, V\rangle = tr(UV)$.

Definición de $\partial_{GP} f(X)$

$$\partial^{GP} f(X) = \big\{ U \in \mathcal{S}^n \;\big|\; f(Y) < f(X) \implies tr\big[U\,(Y - X)\big] < 0\big\}.$$

donde el conjunto subnivel estricto es

$$S_f(X) \;=\; \{Y \in \mathcal{S}^n \mid f(Y) < f(X)\}.$$

Para calcular el subdiferencial de Greenberg-Pierskalla en la función antes mencionada, fijamos $X \in \mathcal{S}^n$ y definimos

$$t_X = \ln\det X, \qquad Y(c) = cX, \quad c > 0.$$

Entonces

$$\ln\det Y(c) = \ln\det(cX) = \ln\big(c^n \det X\big) = n\,\ln c + t_X.$$

Sea $\varphi(t) = t^2 - t$. Se tiene

$$f\big(Y(c)\big) = \varphi\big(n\ln c + t_X\big), \qquad f(X) = \varphi(t_X).$$

La condición $f(Y(c)) < f(X)$ equivale a

$$\varphi\big(n\ln c + t_X\big) < \varphi(t_X).$$

Como φ alcanza su mínimo global en $t = \frac{1}{2}$ y es estrictamente monótona en cada lado, distingimos:

1. **Caso $t_X > \frac{1}{2}$:** φ crece para $t > t_X$ y decrece para $t < t_X$.

$$\varphi(n\ln c + t_X) < \varphi(t_X) \iff n\ln c + t_X < t_X \iff \ln c < 0 \iff 0 < c < 1.$$

Entonces $\{\, c > 0 \mid Y(c) \in S_f(X)\} = (0, 1)$.

2. **Caso** $t_X < \frac{1}{2}$**:** φ decrece para $t < t_X$ y crece para $t > t_X$.

$$\varphi(n \ln c + t_X) < \varphi(t_X) \iff n \ln c + t_X < 1 - t_X \iff \ln c < \frac{1 - 2\,t_X}{n} \iff 1 < c < e^{\frac{1-2t_X}{n}}.$$

Así $\{\, c > 0 \mid Y(c) \in S_f(X)\} = \left(1, e^{\frac{1-2t_X}{n}}\right)$.

3. **Caso** $t_X = \frac{1}{2}$**:** φ alcanza su mínimo y no existe $c \neq 1$ con $\varphi(n \ln c + t_X) < \varphi(t_X)$. Por tanto $S_f(X) = \varnothing$.

Elección de U y comprobación

Por invarianza bajo congruencia y simetría, buscamos

$$U = \alpha\, X^{-1}, \quad \alpha \in \mathbb{R}.$$

Para cada c,

$$tr\left[U\,(Y(c) - X)\right] = tr\left[\alpha\, X^{-1}(cX - X)\right] = \alpha\, tr\left[(c-1)I\right] = \alpha\, n\,(c-1).$$

Queremos que, para todo c tal que $Y(c) \in S_f(X)$, $tr\left[U\,(Y(c) - X)\right] < 0$.

- Si $t_X > 1/2$, entonces $0 < c < 1 \Longrightarrow (c - 1 < 0)$. Para $\alpha\, n\,(c-1) < 0$ se necesita $\alpha > 0$.
- Si $t_X < 1/2$, entonces $1 < c < e^{\frac{1-2t_X}{n}} \Longrightarrow (c - 1 > 0)$. Para $\alpha\, n\,(c-1) < 0$ se necesita $\alpha < 0$.
- Si $t_X = 1/2$, $S_f(X) = \emptyset$ y cualquier α vale.

De esta manera definimos el subdiferencial de GP en el caso Euclidiano para la función

$$\boxed{\partial_{GP} f(X) = \left\{\, \alpha\, X^{-1} \;\middle|\; \alpha \begin{cases} > 0, & \ln\det X > \frac{1}{2} \\ < 0, & \ln\det X < \frac{1}{2} \end{cases}\right.}$$

5.0.5. Subdiferencial de Greemberg-Pierskalla de $f(X) = (\ln\det X)^2 - \ln\det X$ en la variedad SPD

Definicion de los elementos u en $\partial^{GP} f$

Sea $\eta_{XY}(t)$ la geodésica que une X e Y; en $\mathbb{P}^n_{++}$ la forma explícita es

$$\eta_{XY}(t) = X^{1/2}\Big(X^{-1/2}YX^{-1/2}\Big)^t X^{1/2},$$

con velocidad inicial

$$\dot\eta_{XY}(0) = X^{1/2}\ln\Big(X^{-1/2}YX^{-1/2}\Big)X^{1/2}.$$

Utilizando la métrica

$$\langle U, V\rangle_X = \mathrm{tr}\Big(VX^{-1}UX^{-1}\Big),$$

se cumple que:

$$\langle X,\ \dot\eta_{XY}(0)\rangle_X = \mathrm{tr}\Big(\ln\Big(X^{-1/2}YX^{-1/2}\Big)\Big) = \ln\det(Y) - \ln\det(X)$$

Efectivamente si consideramos la variedad Riemanniana $\mathbb{P}^n_{++}$ de matrices simétricas positivas definidas, equipada con la métrica invariante afín definida, para $P \in \mathbb{P}^n_{++}$, y $A, B \in T_P\mathbb{P}^n_{++}$, por:

$$\langle A, B\rangle_P = \mathrm{tr}\left(P^{-1}AP^{-1}B\right).$$

Dada esta métrica, la geodésica $\eta(t)$ que une X con Y está dada por:

$$\eta(t) = X^{1/2}\left(X^{-1/2}YX^{-1/2}\right)^t X^{1/2}.$$

La derivada en $t = 0$ es:

$$\dot\eta(0) = \left.\frac{d}{dt}\right|_{t=0}\eta(t) = X^{1/2}\cdot\ln\left(X^{-1/2}YX^{-1/2}\right)\cdot X^{1/2}.$$

Entonces, el producto interno en X entre X y $\dot\eta(0)$ se calcula como:

$$\begin{aligned}\langle X, \dot{\eta}(0)\rangle_X &= \operatorname{tr}\left(X^{-1}\cdot X\cdot X^{-1}\dot{\eta}(0)\right) = \operatorname{tr}\left(X^{-1}\cdot \eta(0)\right)\\ &= \operatorname{tr}\left(X^{-1}\cdot X^{1/2}\cdot \ln\left(X^{-1/2}\cdot Y\cdot X^{-1/2}\right)\cdot X^{1/2}\right)\\ &= \operatorname{tr}\left(X^{-1/2}\cdot \ln\left(X^{-1/2}\cdot Y\cdot X^{-1/2}\right)\cdot X^{1/2}\right)\\ &= \operatorname{tr}\left(\ln\left(X^{-1/2}\cdot Y\cdot X^{-1/2}\right)\right).\end{aligned}$$

Ahora, utilizamos la propiedad de la fórmula de Jacobi que para toda matriz simétrica positiva definida A, se cumple:

$$\operatorname{tr}(\ln A) = \ln\det(A).$$

Aplicando esta propiedad, obtenemos:

$$\operatorname{tr}\left(\ln\left(X^{-1/2}YX^{-1/2}\right)\right) = \ln\det\left(X^{-1/2}YX^{-1/2}\right).$$

Usando propiedades del determinante:

$$\begin{aligned}\ln\det\left(X^{-1/2}YX^{-1/2}\right) &= \ln\left(\det(X^{-1/2})^2\cdot\det(Y)\right)\\ &= \ln\left(\det(Y)\cdot\det(X)^{-1}\right)\\ &= \ln\det(Y) - \ln\det(X).\end{aligned}$$

Por lo tanto, se concluye que:

$$\langle X, \dot{\eta}_{XY}(0)\rangle_X = \ln\det(Y) - \ln\det(X).$$

Por lo tanto:

- Si $\ln\det(X) > \frac{1}{2}$, entonces para todo $Y \in S_f(X)$ se tiene $\ln\det Y < \ln\det X$, de modo que
$$\langle X,\ \dot{\eta}_{XY}(0)\rangle_X = \ln\det Y - \ln\det X < 0.$$

- Si $\ln\det(X) < \frac{1}{2}$, para $Y \in S_f(X)$ se cumple $\ln\det(Y) > \ln\det(X)$ y, utilizando $u(X) = -X$,
$$\langle -X,\ \dot{\eta}_{XY}(0)\rangle_X = -\big(\ln\det(Y) - \ln\det(X)\big) < 0.$$

En ambos casos, el vector $u(X)$ cumple la condición para pertenecer a $\partial^{GP} f(X)$.

Algoritmo del quasi subdiferencial de Greenberg-Pisrkalla para matrices definidas positivas

1. **Inicialización:**

   - Escoger $X_0 \in \mathbb{P}^n_{++}$.
   - Definir una secuencia de pasos $\{t_k\}_{k\geq 0}$ con $t_k > 0$ que cumpla

   $$\sum_{k=0}^{\infty} t_k = \infty \quad \text{y} \quad \sum_{k=0}^{\infty} t_k^2 < \infty.$$

2. **Criterio de parada:** Detener el algoritmo si $f(X_k)$ está suficientemente cerca del óptimo (por ejemplo, si $|f(X_k) - f^*|$ es menor a una tolerancia).

3. **Selección del elemento del subdiferencial de GP:** Para el iterado actual X_k, definir

   $$u^k = \begin{cases} X_k, & \text{si } \ln\det(X_k) > \frac{1}{2}, \\ -X_k, & \text{si } \ln\det(X_k) < \frac{1}{2}. \end{cases}$$

   Luego, normalizar el vector respecto a la norma inducida por la métrica en $\mathbb{P}^n_{++}$:

   $$\tilde{u}^k = \frac{u^k}{\|u^k\|_{X_k}},$$

   donde

   $$\|u\|_X = \sqrt{\langle u, u\rangle_X}.$$

4. **Actualización del iterado:** Actualizar X_k utilizando el mapeo exponencial en $\mathbb{P}^n_{++}$:

   $$X_{k+1} = \exp_{X_k}\left(-t_k\, \tilde{u}^k\right),$$

   siendo

   $$\exp_{X_k}(V) = X_k^{1/2}\, \exp\left(X_k^{-1/2} \cdot V \cdot\, X_k^{-1/2}\right) X_k^{1/2}.$$

5. **Iterar:** Incrementar k y repetir a partir del Paso 2.

VI. Ejemplos Computacionales

En esta sección se muestra algunos resultados computacionales con el fin de poner en evidencia que los algoritmos de método subgradiente mostrado en las secciones anterios convergen a su punto mínimo. Debemos mencionar que en esta sección se evidenciará la comparación del método gradiente en variedades Riemannianos con el método subgradiente de Ferreira y Oliveira , de igual manera se mostrara la convergencia del subgradiente de GP

Antes de seguir con nuestro cometido informamos que para los siguientes resultados fueron obtenidos a partir de programas escritos en los programas de Matlab, Maple y Python (programas detallados en los anexos) en una láptop con las siguientes características:

- Procesador: 12th Gen Intel(R) Core(TM) i7-1255U (12 CPUs), 1.7 GHz
- Memoria: 16 GB RAM
- Sistema: Sistema operativo de 64 bits, procesador basado en x64.

6.1. Aproximación de geodésicas en el plano superior de Poincaré

Podemos aplicar los métodos de Runge-Kutta. Para obtener una aproximación de la imagen de una geodésica, es suficiente seleccionar los puntos

$$(x(t_\alpha), y(t_\alpha)) \in \mathbb{H}^2, \quad \alpha = 0, 1, \ldots, m.$$

donde (u, v) representa las componentes del vector tangente a la curva geodésica en el plano superior de Poincaré y $\mathfrak{M}$ representa el número de particiones que realizaremos al intervalo $I = [0, \tau]$ y $h = \dfrac{\tau}{m}$ de este modo h es nuestro tamaño de paso , por cuestiones computacionales denotaremos los valores de entrada para las geodésicas de Poincaré: como (x, y, u, v, T, m) para los puntos iniciales, los vectores tangentes y (T, m) para el valor de (τ, m)

6.1.1. Geodésicas verticales

$(x, y, u, v) = (0, 1, 0, 1)$ y $T, m = 3, 15$

N	x	y	u	v	N	x	y	u	v
0	0.0000	1.0000	0.0000	1.0000	8	0.0000	1.7046	0.0000	1.7046
1	0.0000	1.0689	0.0000	1.0689	9	0.0000	1.8221	0.0000	1.8221
2	0.0000	1.1426	0.0000	1.1426	10	0.0000	1.9477	0.0000	1.9477
3	0.0000	1.2214	0.0000	1.2214	11	0.0000	2.0820	0.0000	2.0820
4	0.0000	1.3056	0.0000	1.3056	12	0.0000	2.2255	0.0000	2.2255
5	0.0000	1.3956	0.0000	1.3956	13	0.0000	2.3790	0.0000	2.3790
6	0.0000	1.4918	0.0000	1.4918	14	0.0000	2.5430	0.0000	2.5430
7	0.0000	1.5947	0.0000	1.5947	15	0.0000	2.7183	0.0000	2.7183

Cuadro 3: Resultados del método de Runge-Kutta para aproximar geodésicas, caso de la linea vertical

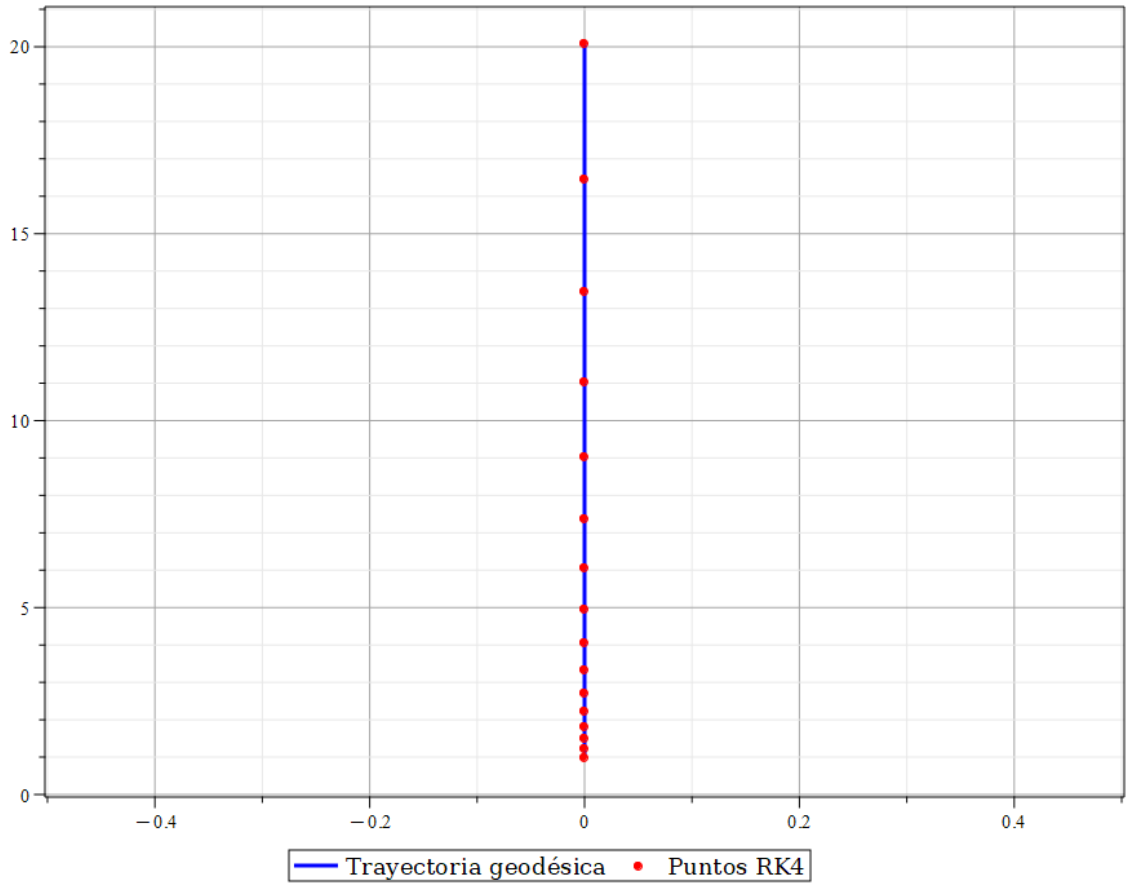


Figura 2: Geodésica de la linea vertical

6.1.2. Geodésicas Semicircunferencias

Geodésicas de Poincaré para: (x, y, u, v) =(1,0.1,0.1,1) y $T, m = 3, 50$

N	x	y	u	v	N	x	y	u	v
0	1.0000	0.1000	0.1000	1.0000	21	2.9861	0.2042	0.4171	-2.0096
1	1.0032	0.1284	0.1647	1.2794	22	2.9943	0.1595	0.2545	-1.5827
2	1.0086	0.1646	0.2708	1.6317	23	2.9993	0.1244	0.1548	-1.2404
3	1.0173	0.2107	0.4436	2.0701	24	3.0024	0.0969	0.0940	-0.9693
4	1.0316	0.2689	0.7225	2.6038	25	3.0042	0.0754	0.0570	-0.7560
5	1.0548	0.3416	1.1662	3.2292	26	3.0053	0.0587	0.0345	-0.5890
6	1.0920	0.4308	1.8547	3.9127	27	3.0060	0.0457	0.0209	-0.4586
7	1.1504	0.5370	2.8813	4.5637	28	3.0064	0.0355	0.0127	-0.3570
8	1.2395	0.6573	4.3163	5.0009	29	3.0067	0.0276	0.0077	-0.2778
9	1.3694	0.7830	6.1244	4.9409	30	3.0068	0.0215	0.0046	-0.2161
10	1.5468	0.8976	8.0486	4.0735	31	3.0069	0.0167	0.0028	-0.1681
11	1.7685	0.9787	9.5696	2.2737	32	3.0070	0.0130	0.0017	-0.1308
12	2.0170	1.0058	10.1070	-0.1597	33	3.0070	0.0101	0.0010	-0.1017
13	2.2634	0.9711	9.4219	-2.5458	34	3.0070	0.0079	0.0006	-0.0791
14	2.4802	0.8845	7.8160	-4.2347	35	3.0071	0.0061	0.0004	-0.0616
15	2.6516	0.7674	5.8825	-4.9874	36	3.0071	0.0048	0.0002	-0.0479
16	2.7759	0.6417	4.1128	-4.9668	37	3.0071	0.0037	0.0001	-0.0372
17	2.8605	0.5228	2.7305	-4.4888	38	3.0071	0.0029	0.0001	-0.0290
18	2.9158	0.4187	1.7514	-3.8257	39	3.0071	0.0022	0.0001	-0.0225
19	2.9508	0.3316	1.0988	-3.1460	40	3.0071	0.0017	0.0000	-0.0175
20	2.9727	0.2608	0.6799	-2.5312					

Cuadro 4: Resultados del método de Runge-Kutta para aproximar geodésicas, caso de la semicircunferencia

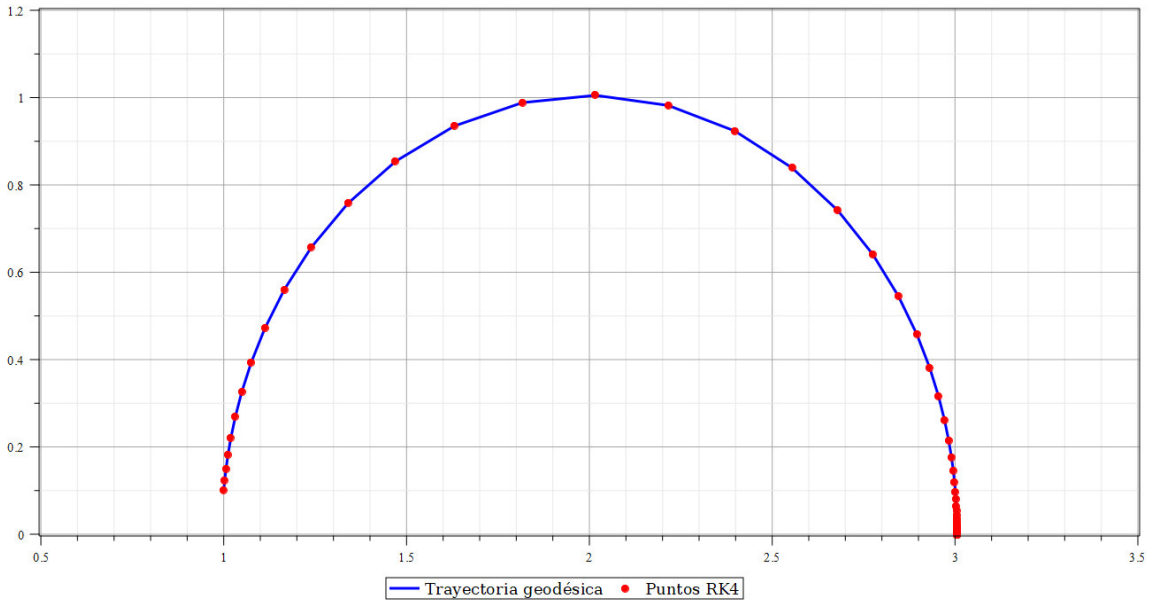


Figura 3: Geodésica de la semicircunferencia

6.2. Método del subgradiente clásico para funciones convexas en la variedad del plano superior de Poincaré

Sea $f : \mathbb{H}^2 \to \mathbb{R}$ tal que:

$$f(x_1, x_2) = 1 + \frac{{x_1}^2}{{x_2}^2}$$

Los símbolos de Christoffel de segunda clase son los siguientes:

$$\Gamma^{x_1} = \begin{bmatrix} 0 & -\frac{1}{x_2} \\ -\frac{1}{x_2} & 0 \end{bmatrix}, \quad \Gamma^{x_2} = \begin{bmatrix} \frac{1}{x_2} & 0 \\ 0 & -\frac{1}{x_2} \end{bmatrix}.$$

Ahora, vemos que el hessiano (hiperbólico), o la derivada covariante de segundo orden, de está dado por la siguiente definición:

$$H^g f(x_1, x_2) = \nabla^2 f(x_1, x_2) - \nabla f(x_1, x_2)\Gamma,$$

Donde:

$$H = \begin{bmatrix} \frac{2}{{x_2}^2} + \frac{2{x_1}^2}{{x_2}^4} & -\frac{2x_1}{{x_2}^3} \\ -\frac{2x_1}{{x_2}^3} & \frac{4{x_1}^2}{{x_2}^4} \end{bmatrix}$$

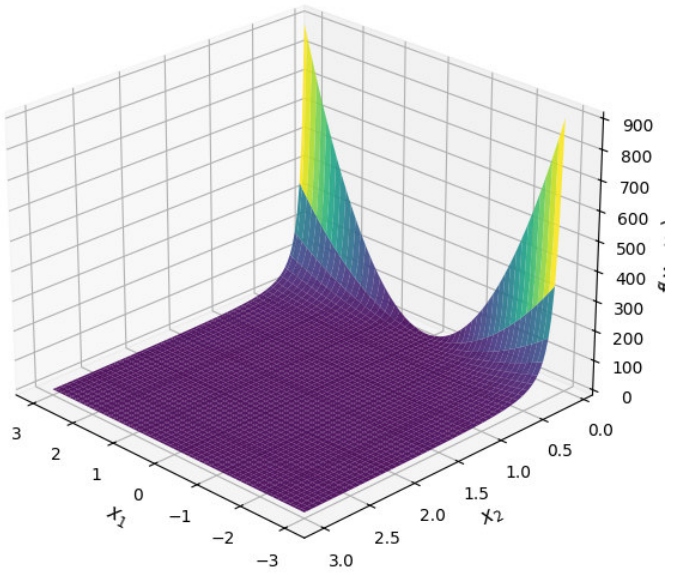


Figura 4: gráfico de la función en el plano superior de Poincaré

podemos notar que $|H_{11}| > 0$ y ademas:

$$|H| = \begin{vmatrix} \frac{2}{{x_2}^2} + \frac{2{x_1}^2}{{x_2}^4} & -\frac{2x_1}{{x_2}^3} \\ -\frac{2x_1}{{x_2}^3} & \frac{4{x_1}^2}{{x_2}^4} \end{vmatrix}$$

$$|H| = \frac{4{x_1}^2(2{x_1}^2 + {x_2}^3)}{{x_2}^8} > 0$$

Por lo tanto $f(x_1, x_2)$ es convexa en el plano superior de Poincaré.
Empleando el método del gradiente en variedades de Riemann bajo la condición de Armijo, junto con un tamaño de paso exógeno y una variante ligera que satisface dichas condiciones, usaremos $t_k = \frac{3}{k+5}$, el método subgradiente con el tamaño de paso $1/k$ y el método subgradiente con el tamaño de paso de Polyak, la tolerancia(tol=10^{-6}) y el maximo de iteraciones será 1000, de este modo mostraremos los resultados obtenidos.

Para el punto $x_0 = (-2, 11)$ con geodésicas aproximadas obtenemos:

Cuadro 5: Descenso por gradiente con paso adaptativo (Armijo)

Iteración	**x**	**y**	**f(x)**	$\|\nabla f(x)\|$
0	-2.0	11.0	1.03305785	0.03359982
1	-0.02450732	11.18031303	1.0000048	4.06557814
2	-0.00081684	11.18033986	1.00000001	0.04901475
3	-2.723e-05	11.18033989	1.0	0.00163367
4	-9.1e-07	11.18033989	1.0	5.446e-05
5	-3e-08	11.18033989	1.0	1.82e-06

Cuadro 6: Descenso por gradiente con paso decreciente $\alpha = \frac{1}{k}$

Iteración	**x**	**y**	**f(x)**	$\|\nabla f(x)\|$
0	-2.0	11.0	1.03305785	4.06557814
1	-0.02450732	11.18031303	1.0000048	4.06557814
2	-0.0042012	11.1803391	1.00000014	0.04901475
3	-0.00115533	11.18033983	1.00000001	0.00840239
4	-0.00041078	11.18033988	1.0	0.00231066
5	-0.00017253	11.18033989	1.0	0.00082157
6	-8.156e-05	11.18033989	1.0	0.00034506
		⋮		
12	-3.52e-06	11.18033989	1.0	1.069e-05
13	-2.39e-06	11.18033989	1.0	7.04e-06
14	-1.66e-06	11.18033989	1.0	4.77e-06
15	-1.18e-06	11.18033989	1.0	3.32e-06
16	-8.5e-07	11.18033989	1.0	2.35e-06
17	-6.3e-07	11.18033989	1.0	1.7e-06
18	-4.7e-07	11.18033989	1.0	1.25e-06

Cuadro 7: Método del subgradiente con paso $\alpha = \frac{1}{k}$

Iteración	**x**	**y**	**f(x)**
0	-2.0	11.0	1.03305785
1	7.33479055	8.43806325	1.7555964
2	3.28255644	10.68759795	1.09433312
3	-0.22044707	11.17816049	1.00038893
4	2.44148031	10.91050262	1.05007454
5	0.31981988	11.1757585	1.00081895
6	-1.47276038	11.08290672	1.01765862
		⋮	
53	0.00385934	11.18033026	1.00000012
54	-0.19626222	11.17860809	1.00030825
55	0.0002204	11.18033092	1.0
56	-0.1927543	11.1786691	1.00029732
57	-0.00316545	11.18033044	1.00000008
58	0.18315701	11.1788306	1.00026844
59	-7.17e-06	11.18033086	1.0

Cuadro 8: Método del subgradiente con paso de Polyak

Iteración	**x**	**y**	**f(x)**
0	-2.0	11.0	1.03305785
1	-1.01891511	11.13381419	1.00837506
2	-0.51193193	11.16861388	1.002101
3	-0.25627687	11.17740287	1.0005257
4	-0.1281753	11.17960576	1.00013145
5	-0.06409015	11.18015684	1.00003286
6	-0.03204327	11.18029463	1.00000821
		⋮	
11	-0.00099667	11.18034052	1.00000001
12	-0.00049591	11.18034055	1
13	-0.00024551	11.18034056	1
14	-0.00012029	11.18034056	1
15	-5.763e-05	11.18034056	1
16	-2.618e-05	11.18034056	1
17	-1.012e-05	11.18034056	1

Para el punto $x_0 = (55555, 55555)$ con geodésicas aproximadas obtenemos:

Cuadro 9: Descenso por gradiente con paso adaptativo (Armijo)

Iteración	x	y	f(x)	$\|\nabla f(x)\|$
0	55555	55555	2	5.091e-05
1	-35417.558	70130.675	1.2550476	157133.27
2	4803.2654	78419.664	1.0037517	79355.806
3	148.68688	78566.487	1.0000036	9624.5342
4	4.9558919	78566.628	1	297.37429
5	0.16519639	78566.628	1	9.9117839
6	0.00550655	78566.628	1	0.33039277
		⋮		
87	7.4e-07	78566.628	1	1.58e-06
88	-6.9e-07	78566.628	1	1.47e-06
89	6.4e-07	78566.628	1	1.37e-06
90	-6e-07	78566.628	1	1.28e-06
91	5.6e-07	78566.628	1	1.2e-06
92	-5.2e-07	78566.628	1	1.12e-06
93	4.9e-07	78566.628	1	1.04e-06

Cuadro 10: Descenso por gradiente con paso decreciente $\alpha = \frac{1}{k}$

Iteración	x	y	f(x)	$\|\nabla f(x)\|$
0	55555	55555	2	157133.27
1	-35417.558	70130.675	1.2550476	157133.27
2	-1329.0428	78555.386	1.0002862	79355.806
3	-365.33513	78565.779	1.0000216	2658.466
4	-129.89436	78566.521	1.0000027	730.67815
5	-54.555534	78566.609	1.0000005	259.78907
6	-25.789883	78566.624	1.0000001	109.11109
		⋮		
173	-5.8e-07	78566.628	1	1.2e-06
174	-5.6e-07	78566.628	1	1.17e-06
175	-5.5e-07	78566.628	1	1.13e-06
176	-5.3e-07	78566.628	1	1.09e-06
177	-5.1e-07	78566.628	1	1.06e-06
178	-5e-07	78566.628	1	1.02e-06

Cuadro 11: Método del subgradiente con paso $\alpha = \frac{1}{k}$

Iteración	x	y	f(x)
0	55555	55555	2
1	-6684.9887	78281.741	1.0072926
2	29719.436	72728.653	1.1669821
3	5945.3583	78341.068	1.0057594
4	-12911.949	77498.058	1.0277589
5	2158.8279	78536.629	1.0007556
6	-10436.504	77870.157	1.0179626
	⋮		
53	27.169791	78564.051	1.0000001
54	-1379.1344	78555.136	1.0003082
55	1.6026726	78567.267	1
56	-1354.4833	78555.563	1.0002973
57	-22.190654	78567.2	1.0000001
58	1287.1414	78556.289	1.0002685
59	0.00341868	78566.793	1

Cuadro 12: Método del subgradiente con paso de Polyak

Iteración	x	y	f(x)
0	55555	55555	2
1	38003.714	68763.617	1.3054462
2	21874.035	75460.028	1.084028
3	11447.939	77727.899	1.0216921
4	5795.4797	78352.369	1.0054711
5	2906.9528	78512.624	1.0013709
6	1454.6506	78552.943	1.0003429
	⋮		
15	2.8357413	78566.347	1
16	1.4178706	78566.347	1
17	0.70893517	78566.347	1
18	0.3544672	78566.347	1
19	0.17723467	78566.347	1
20	0.10338778	78566.347	1
21	0.02799745	78566.347	1

Para el punto $x_0 = (7, 20)$ con geodésicas aproximadas obtenemos:

Cuadro 13: Descenso por gradiente con paso adaptativo (Armijo)

Iteración	x	y	f(x)	$\|\nabla f(x)\|$
0	7	20	1.1225	0.03708184
1	-0.32282398	21.187161	1.0002322	14.832734
2	-0.01071333	21.189617	1.0000003	0.64572291
3	-0.00035711	21.18962	1	0.02142666
4	-1.19e-05	21.18962	1	0.00071422
5	-4e-07	21.18962	1	2.381e-05

Cuadro 14: Descenso por gradiente con paso decreciente $\alpha = \frac{1}{k}$

Iteración	x	y	f(x)	$\|\nabla f(x)\|$
0	7	20	1.1225	14.83273407
1	-0.32282398	21.18716084	1.00023216	14.83273407
2	-0.05530401	21.18954792	1.00000681	0.64572291
3	-0.01520845	21.18961464	1.00000052	0.1106084
4	-0.00540745	21.18961941	1.00000007	0.03041691
5	-0.00227113	21.18961997	1.00000001	0.01081489
6	-0.00107362	21.18962007	1	0.00454226
		⋮		
24	-1.39e-06	21.1896201	1	3.48e-06
25	-1.12e-06	21.1896201	1	2.78e-06
26	-9.1e-07	21.1896201	1	2.25e-06
27	-7.5e-07	21.1896201	1	1.83e-06
28	-6.2e-07	21.1896201	1	1.49e-06
29	-5.1e-07	21.1896201	1	1.23e-06
30	-4.3e-07	21.1896201	1	1.02e-06

Cuadro 15: Método del subgradiente con paso $\alpha = \frac{1}{k}$

Iteración	x	y	f(x)
0	7	20	1.1225
1	-11.72891952	17.64743942	1.44172564
2	-2.94963711	20.98331053	1.0197601
3	3.81675041	20.84303587	1.03353247
4	-1.26048267	21.15208838	1.00355114
5	2.81789587	21.00140846	1.01800334
6	-0.579052	21.18169849	1.00074733
		⋮	
76	0.00356078	21.18960479	1.00000003
77	-0.26244248	21.18797972	1.00015342
78	0.00015031	21.18960508	1
79	-0.25911884	21.18801885	1.00014956
80	-0.00309071	21.18960307	1.00000002
81	0.24977775	21.18813096	1.00013897
82	-6.81e-06	21.18960312	1

Cuadro 16: Método del subgradiente con paso de Polyak

Iteración	x	y	f(x)
0	7	20	1.1225
1	3.73336953	20.85813737	1.03203693
2	1.90092962	21.10417931	1.00811326
3	0.95494423	21.16808892	1.00203513
4	0.47804069	21.18422476	1.00050922
5	0.23909374	21.18826877	1.00012733
6	0.11955816	21.1892804	1.00003184
		⋮	
16	0.00012152	21.18961767	1
17	6.309e-05	21.18961767	1
18	3.379e-05	21.18961767	1
19	1.901e-05	21.18961767	1
20	1.143e-05	21.18961767	1
21	7.41e-06	21.18961767	1
22	5.17e-06	21.18961767	1

6.3. Método gradiente y subgradiente para una función conevxa en el plano superior de Poincaré

Sea la función:

$$f(x_1, x_2) = 1 + \frac{(x_1 - 1)^2 + (x_2 - 1)^2}{x_2}, \quad x_2 > 0$$

Primero demostraremos que esta función es convexa en la variedad, luego se sabe que la composición de una función creciente con una función convexa, es cuasi-convexa, en este caso definiremos $g(x) = \ln x$, donde $h = g \circ f = \ln f(x, y)$
Sea la función

$$f\colon \mathbb{H}^2 \to \mathbb{R}, \qquad f(x_1, x_2) \;=\; 1 + \frac{(x_1 - 1)^2 + (x_2 - 1)^2}{x_2}, \qquad x_2 > 0, \tag{41}$$

definida sobre el plano superior de Poincaré $\mathbb{H}^2 = \{(x_1, x_2) \in \mathbb{R}^2 : x_2 > 0\}$ con la métrica

$$g \;=\; \frac{1}{x_2^2}\left(dx_1^2 + dx_2^2\right). \tag{42}$$

Símbolos de Christoffel

Para la métrica (42), las únicas componentes no nulas de la conexión de Levi-Civita son

$$\Gamma^1_{12} = \Gamma^1_{21} = -\frac{1}{x_2}, \qquad \Gamma^2_{11} = \frac{1}{x_2}, \qquad \Gamma^2_{22} = -\frac{1}{x_2}. \tag{43}$$

Gradiente y Hessiana Euclidiana

Las derivadas parciales de f se calculan directamente a partir de (41):

$$\partial_1 f = \frac{2\,(x_1 - 1)}{x_2}, \qquad \partial_2 f = \frac{-x_1^2 + 2x_1 + x_2^2 - 2}{x_2^2}. \tag{44}$$

La matriz hessiana clásica queda

$$H^{\mathrm{Euc}} f = \begin{pmatrix} \dfrac{2}{x_2} & \dfrac{2\,(1 - x_1)}{x_2^2} \\[2ex] \dfrac{2\,(1 - x_1)}{x_2^2} & \dfrac{2\left(x_1^2 - 2x_1 + 2\right)}{x_2^3} \end{pmatrix}.$$

Fórmula de la Hessiana Riemanniana

Para un campo escalar f y la conexión ∇ asociada a g,

$$(\mathrm{H}^g f)_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j} - \sum_k \Gamma_{ij}^k \frac{\partial f}{\partial x_k} = \partial_i \partial_j f - \Gamma_{ij}^k \, \partial_k f. \tag{45}$$

Sustituyendo (44) y (43) en (45):

$$\begin{aligned}
(\mathrm{H}^g f)_{11} &= \frac{2}{x_2} - \frac{1}{x_2} \frac{-x_1^2 + 2x_1 + x_2^2 - 2}{x_2^2} \;=\; \frac{x_1^2 - 2x_1 + x_2^2 + 2}{x_2^3}, \\
(\mathrm{H}^g f)_{12} &= \frac{2\,(1 - x_1)}{x_2^2} - \left(-\frac{1}{x_2}\right) \frac{2\,(x_1 - 1)}{x_2} \;=\; 0, \\
(\mathrm{H}^g f)_{22} &= \frac{2\left(x_1^2 - 2x_1 + 2\right)}{x_2^3} - \left(-\frac{1}{x_2}\right) \frac{-x_1^2 + 2x_1 + x_2^2 - 2}{x_2^2} \;=\; \frac{x_1^2 - 2x_1 + x_2^2 + 2}{x_2^3}.
\end{aligned}$$

Matriz Hessiana Riemanniana

Todos los elementos fuera de la diagonal son nulos y los elementos diagonales son iguales, por lo que

$$\boxed{\mathrm{H}^g f(x_1, x_2) = \frac{x_1^2 + x_2^2 - 2x_1 + 2}{x_2^3} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}, \qquad x_2 > 0. \tag{46}$$

El numerador puede reescribirse como $(x_1 - 1)^2 + x_2^2 + 1 > 0$, por lo que la Hessiana es definida positiva en todo $\mathbb{H}^2$.

Determinante de la Hessiana

Sea $a(x_1, x_2) = \dfrac{x_1^2 + x_2^2 - 2x_1 + 2}{x_2^3}$. Al ser $\mathrm{H}^g f = a\, I_2$, se tiene

$$\boxed{\det\!\left(\mathrm{H}^g f\right) = \frac{\left(x_1^2 + x_2^2 - 2x_1 + 2\right)^2}{x_2^6} > 0}. \tag{47}$$

De este modo podemos ver que f es convexa en la variedad
Para los puntos iniciales $x_0 = (55555, 55555)$

Cuadro 17: Método del gradiente con paso de Armijo

Iteración	x	y	$f(x)$	$\|\nabla f(x)\|$
0	55555	55555	111107.000036	111108.000018
1	-1.14117562	158.03780704	157.07314437	158.06049149
2	0.99993304	1.33416721	1.08369845	0.58463599
3	0.99999901	1.0057497	1.00003287	0.01146652
4	0.99999997	1.0001911	1.00000004	0.00038215
5	1	1.00000637	1	1.274e-05
6	1	1.00000021	1	4.2e-07

Cuadro 18: Método del gradiente con paso exógeno(1/k)

Iteración	x	y	$f(x)$	$\|\nabla f(x)\|$
0	55555	55555	111107.000036	111108.000018
1	3.75547071	640.86074289	639.87415083	640.87103008
2	0.99999335	0.1658436	5.195621	5.8639338
3	1.03538329	71.99292627	71.00683391	71.97905341
4	0.99999317	0.00122218	817.21139865	818.20895429
5	167462.39823085	71356.19823861	464359.87885032	464360.87884601
⋮	⋮	⋮	⋮	⋮
68	0.99999998	1.00000074	1	1.47e-06
69	0.99999998	1.00000067	1	1.35e-06
70	0.99999998	1.00000062	1	1.24e-06
71	0.99999998	1.00000057	1	1.14e-06
72	0.99999999	1.00000052	1	1.04e-06
73	0.99999999	1.00000048	1	9.6e-07

Cuadro 19: Método del subgradiente con paso exógeno

Iteración	x	y	$f(x)$
0	55555	55555	111107.000036
1	14042.87336079	36919.0760735	42258.78958669
2	3951.62270338	20575.39740979	21332.94512742
3	1326.59637184	12063.59234793	12208.25432741
4	508.94421217	7495.32027759	7528.7428657
5	215.52109314	4877.44652537	4885.88185166
⋮	⋮	⋮	⋮
147	0.99999739	0.99999978	1
148	0.96789496	0.93038098	1.00631735
149	0.99988119	0.99975932	1.00000007
150	1.03675607	1.07273068	1.00619052
151	0.99999541	0.99999484	1

Cuadro 20: Método del subgradiente con paso de Polyak

Iteración	x	y	$f(x)$
0	55555	55555	111107.000036
1	40984.99838228	53610.1094338	84940.6650329
2	28290.50101297	48404.03253112	64936.69360674
3	18491.32780317	41383.18443971	49643.80634962
4	11610.72258895	33987.79706486	37952.50450201
5	7094.47170529	27163.17714793	29014.58717933
⋮	⋮	⋮	⋮
167	0.99999517	0.99999788	1
168	0.99999517	0.9999965	1
169	0.99999517	0.9999929	1
170	0.99999517	0.99999727	1
171	0.99999517	0.9999953	1

Mostraremos los cuadros de los metodos gradiente y subgradiente aplicado en variedades Riemannianas, tomando como punto inicial el (4444, 3333), obtenemos los siguientes cuadros

Cuadro 21: Método de descenso por gradiente Riemanniano (Armijo)

Iter	x	y	f	gradNorm
0	4444	3333	9254.7	9255.7
1	0.87102	1.4854	1.1698	0.8415
2	0.99948	1.0025	1	0.0050131
3	0.99998	1.0001	1	0.0001671
4	1	1	1	5.57e-06
5	1	1	1	1.9e-07

Cuadro 22: Descenso por gradiente Riemanniano (tamaño decreciente $1/k$)

Iter	x	y	f	gradNorm
0	4444	3333	9254.7	9255.7
1	0.68364	0.20126	4.6671	5.3025
2	4.0505	0.011015	934.58	935.57
3	0.71874	0.44074	1.8892	2.085
4	1.6804	1.5149	1.4806	1.4675
5	0.89264	0.84327	1.0428	0.41596
6	1.0009	1.0012	1	0.0029419
7	1.0001	1.0002	1	0.00043262
8	1	1	1	0.00011093
9	1	1	1	3.782e-05
...	...	...	...	...
15	1	1	1	7e-07

Cuadro 23: Método del subgradiente Riemanniano $(1/k)$

Iter	x	y	f
0	4444	3333	9254.7
1	1420.9	2800.5	3519.4
2	427.91	1667.9	1776.2
3	146.85	995.71	1016.1
4	57.207	622.18	626.25
5	24.786	405.7	406.1
⋮	⋮	⋮	⋮
295	0.99999	1	1
296	1.0562	1.0009	1.0032
297	1.0001	1	1
298	0.94432	0.99618	1.0031
299	1	1	1

Cuadro 24: Método del subgradiente Riemanniano (Polyak)

Iter	x	y	f
0	4444	3333	9254.7
1	3285.1	3466.6	6576.7
2	2166.7	3216.6	4673.7
3	1294.6	2703.4	3321.4
4	720.89	2116.6	2360.5
5	384.4	1585.9	1677.6
⋮	⋮	⋮	⋮
104	1	1	1
105	1	0.99999	1
106	1	1	1
107	1	0.99999	1
108	1	1	1

De igual modo mostraremos los metodos de gradiente Riemanniano, con el tamaño de paso de Armijo y tamaño de paso exógeno$(1/k)$, como también el método subgradiente con tamaño de paso exógeno$(1/k)$ y tamaño de paso de Polyak, las tabla para el punto inicial $(7, 21)$ son:

Cuadro 25: Resultados del método de descenso por gradiente Riemann (Armijo).

Iteración	x	y	$f(x)$	$\|\nabla f(x)\|$
0	7	21	21.76190476	22.6738684
1	1.01451909	1.46698594	1.14879943	0.78570924
2	1.00009187	1.00363884	1.0000132	0.0072668
3	1.00000305	1.00012107	1.00000001	0.00024221
4	1.0000001	1.00000404	1	8.07e-06
5	1	1.00000013	1	2.7e-07

Cuadro 26: Resultados del método de descenso por gradiente Riemann con tamaño de paso decreciente $1/k$.

Iteración	x	y	$f(x)$	$\|\nabla f(x)\|$
0	7	21	21.76190476	22.6738684
1	0.96040726	1.64e-06	609484.89211	609485.89210672
2	0.96873118	0.47107345	1.59596023	1.65499532
3	1.23135055	2.58769701	1.99482469	2.22911976
4	0.96896459	0.47720044	1.57477437	1.62156192
5	1.02450741	1.26825762	1.05721453	0.48180039
6	0.99981823	0.99773867	1.00000516	0.0045424
7	0.99997322	0.99966713	1.00000011	0.000668
8	0.99999313	0.99991464	1.00000001	0.00017128
9	0.99999766	0.9999709	1	5.839e-05
10	0.99999904	0.99998812	1	2.383e-05
11	0.99999956	0.9999945	1	1.103e-05
12	0.99999977	0.9999972	1	5.61e-06
13	0.99999988	0.99999847	1	3.07e-06
14	0.99999993	0.99999911	1	1.78e-06
15	0.99999996	0.99999946	1	1.08e-06
16	0.99999997	0.99999966	1	6.8e-07

Cuadro 27: Resultados del método del subgradiente Riemann con tamaño de paso $1/k$.

Iteración	x	y	$f(x)$
0	7	21	21.76190476
1	1.9174845	8.54080633	7.75645082
2	1.22651835	4.34962852	3.59132975
3	1.06587113	2.49431357	1.89696504
4	1.01726112	1.53924679	1.18910876
5	0.99997666	0.99920491	1.00000063
6	1.01402263	1.48242628	1.15712872
7	1.00068274	1.02887983	1.00081109
8	0.99453512	0.73108503	1.09895583
⋮	⋮	⋮	⋮
86	0.99792334	0.90158541	1.01074745
87	0.99999503	1.00001412	1
88	1.00066628	0.90209948	1.01062516
89	0.99999866	0.99943374	1.00000032
90	0.99929497	1.10663845	1.0102764
91	0.99999561	0.99999498	1

Cuadro 28: Resultados del método del subgradiente Riemann con regla de Polyak.

Iteración	x	y	$f(x)$
0	7	21	21.76190476
1	2.02369016	9.00915413	8.23647198
2	1.19915225	4.09388279	3.34783769
3	1.05057356	2.23825347	1.68617311
4	1.01688935	1.52964158	1.18357598
5	1.00682967	1.24177743	1.04711228
6	1.00307152	1.11532032	1.01193219
7	1.00145767	1.05638376	1.00301146
8	1.0007098	1.0279305	1.00075941
⋮	⋮	⋮	⋮
26	0.99999505	0.99999039	1
27	0.99999535	1.00000257	1
28	0.99999531	1.00000067	1
29	0.99999525	0.99999864	1
30	0.99999517	0.99999516	1

Por otro lado mostraremos de manera grafica las iteraciones, que se presentan en el metodo subgradiente Riemanniano, y comparándolas con el método subgradiente Euclidiano, las gráficas nos muestran como se comportan los algoritmos en cada iteración

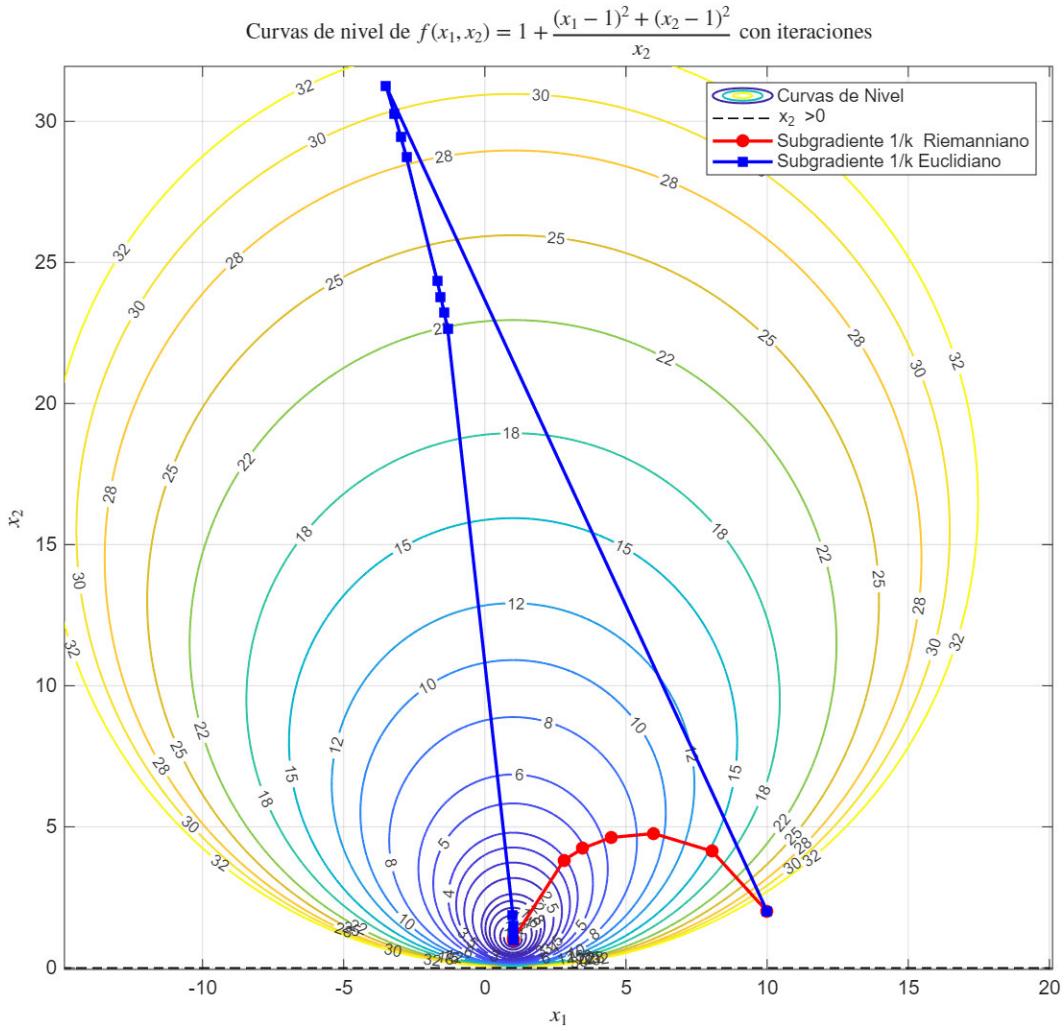


Figura 5: Comparación de los métodos subgradiente en la variedad Riemanniana y el espacio Euclidiano con el tamaño de paso exógeno$(1/k)$.

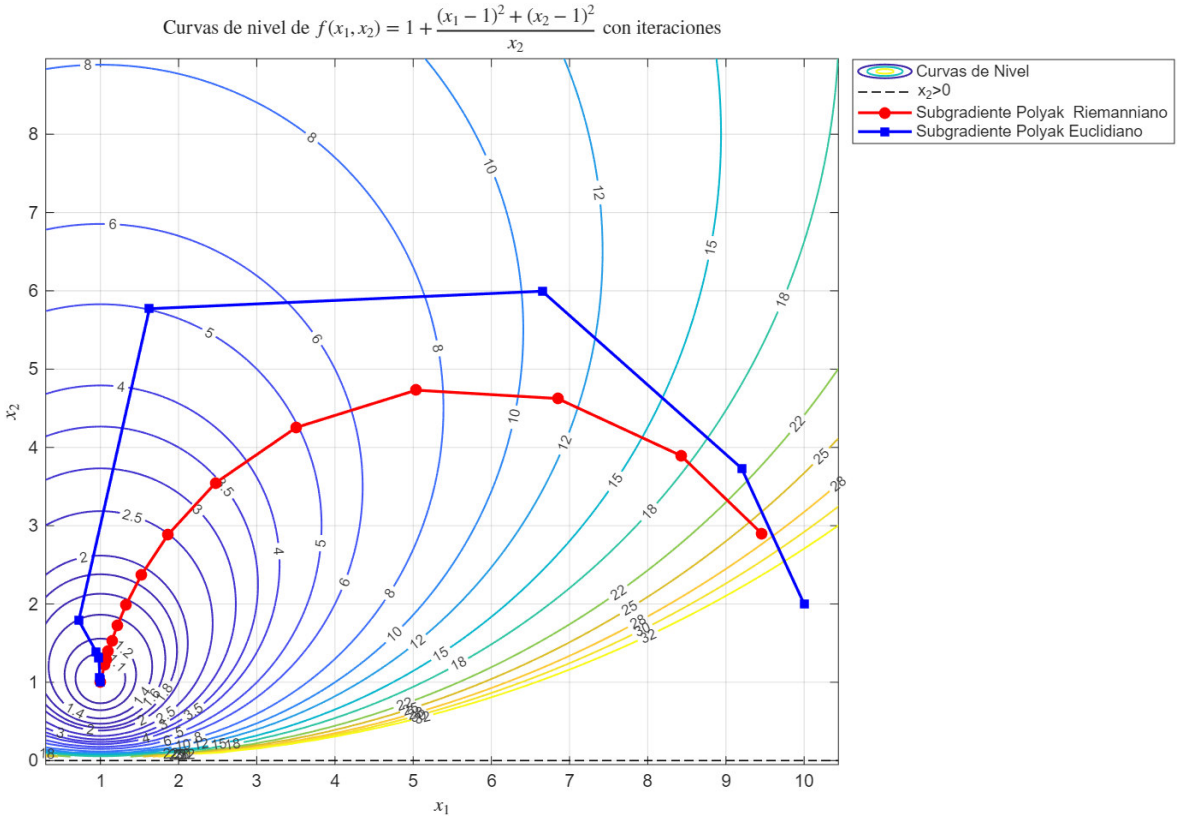


Figura 6: Comparación de los métodos subgradiente en la variedad Riemanniana y el espacio Euclidiano con el tamaño de paso de Polyak

La función $f(x_1,x_2) = 1 + \frac{(x_1-1)^2+(x_2-1)^2}{x_2}, \quad x_2 > 0$ establecida anteriormente,es convexa en la variedad como hemos podido ver , de este modo, si hacemos la composición de una función crecienteln con $h := f(x_1,x_2)$, osea $\ln(f(x_1,x_2))$ se obtiene una función cuasi-convexa, de este modo ahora probaremos que esta nueva función no es convexa.

Dado que los cálculos son extensos para la matriz Hessiana y determinante, se mostrará los resultados del código Maple con los cálculos realizados de la Hessiana en la variedad y la determinante:

Matriz Hessiana Riemanniana:

$$H_{11} = \frac{2}{x_2\left(1+\frac{(x_1-1)^2+(x_2-1)^2}{x_2}\right)} - \frac{(2x_1-2)^2}{x_2^2\left(1+\frac{(x_1-1)^2+(x_2-1)^2}{x_2}\right)^2} - \frac{\frac{2x_2-2}{x_2} - \frac{(x_1-1)^2+(x_2-1)^2}{x_2^2}}{\left(1+\frac{(x_1-1)^2+(x_2-1)^2}{x_2}\right)x_2},$$

$$H_{12} = -\frac{(2x_1-2)\left(\frac{2x_2-2}{x_2} - \frac{(x_1-1)^2+(x_2-1)^2}{x_2^2}\right)}{x_2^2\left(1+\frac{(x_1-1)^2+(x_2-1)^2}{x_2}\right)^2},$$

$$H_{21} = -\frac{(2x_1-2)\left(\frac{2x_2-2}{x_2} - \frac{(x_1-1)^2+(x_2-1)^2}{x_2^2}\right)}{x_2^2\left(1+\frac{(x_1-1)^2+(x_2-1)^2}{x_2}\right)^2},$$

$$H_{22} = \frac{\frac{2}{x_2} - \frac{2(2x_2-2)}{x_2^2} + \frac{2\left((x_1-1)^2+(x_2-1)^2\right)}{x_2^3}}{1+\frac{(x_1-1)^2+(x_2-1)^2}{x_2}} - \frac{\left(\frac{2x_2-2}{x_2} - \frac{(x_1-1)^2+(x_2-1)^2}{x_2^2}\right)^2}{\left(1+\frac{(x_1-1)^2+(x_2-1)^2}{x_2}\right)^2} + \frac{\frac{2x_2-2}{x_2} - \frac{(x_1-1)^2+(x_2-1)^2}{x_2^2}}{\left(1+\frac{(x_1-1)^2+(x_2-1)^2}{x_2}\right)x_2},$$

$$H^g = \begin{pmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{pmatrix}.$$

Determinante de la matriz Hessiana:

$$\det H^g = -\frac{x_1^4 + 2x_1^2x_2^2 + x_2^4 - 4x_1^3 - 4x_1^2x_2 - 4x_1x_2^2 - 4x_2^3 + 8x_1^2 + 8x_1x_2 + 4x_2^2 - 8x_1 - 8x_2 + 4}{x_2^3\left(x_1^2 + x_2^2 - 2x_1 - x_2 + 2\right)^3}$$
$$= -\frac{\left(x_1^2 + x_2^2 - 2x_1 - 4x_2 + 2\right)\left(x_1^2 + x_2^2 - 2x_1 + 2\right)}{x_2^3\left(x_1^2 + x_2^2 - 2x_1 - x_2 + 2\right)^3}.$$

Donde para

$$\det(H^g\ h)\big|_{(x_1,x_2)=(1,5)} = -\frac{\left(1^2 + 5^2 - 2\cdot 1 - 4\cdot 5 + 2\right)\left(1^2 + 5^2 - 2\cdot 1 + 2\right)}{5^3\left(1^2 + 5^2 - 2\cdot 1 - 5 + 2\right)^3}$$
$$= -\frac{6\cdot 26}{125\cdot 21^3}$$
$$= -\frac{156}{1\,157\,625}$$
$$\approx -1{,}3475866537\times 10^{-4} < 0.$$

Con lo cual la función h no es convexa, de este modo concluimos que h es cuasi-convexa pero no convexa.

Se realizaron las siguientes pruebas de gradiente y subgradiente Riemanniano para esta función, obteniendo los siguientes resultados, para el punto inicial $(7, 21)$

Cuadro 29: Método de descenso por gradiente Riemanniano (Armijo)

Iter	x	y	f	gradNorm
0	7	21	3.0802	1.0419
1	1.8459	8.2092	2.0039	1.1023
2	1.0903	2.8556	0.79236	1.136
3	0.99885	0.95339	0.0022771	0.095303
4	0.99996	0.99832	2.82e-06	0.0033578
5	1	0.99994	0	0.00011193
6	1	1	0	3.73e-06
7	1	1	0	1.2e-07

Cuadro 30: Descenso por gradiente Riemanniano (tamaño decreciente $1/k$)

Iter	x	y	f	gradNorm
0	7	21	3.0802	1.0419
1	0.99078	0.52015	0.36661	0.97216
2	1.0657	2.4885	0.63766	1.1041
3	0.99532	0.79376	0.052228	0.44247
4	1.003	1.1125	0.011318	0.21128
5	0.99948	0.97922	0.00044125	0.042
6	1	0.99999	0	1.545e-05
7	1	1	0	2.27e-06
8	1	1	0	5.8e-07

Cuadro 31: Método del subgradiente Riemanniano ($1/k$)

Iter	x	y	f
0	7	21	3.0802
1	1.9175	8.5408	2.0485
2	1.2265	4.3496	1.2785
3	1.0659	2.4943	0.64026
4	1.0173	1.5392	0.1732
5	0.99998	0.9992	6.3e-07
⋮	⋮	⋮	⋮
86	1.0007	0.90157	0.01069
87	0.99999	1	0
88	1.0035	0.90216	0.010569
89	1	0.99943	3.2e-07
90	0.99593	1.1066	0.010224
91	1	1	0

Cuadro 32: Método del subgradiente Riemanniano (Polyak)

Iter	x	y	f
0	7	21	3.0802
1	1.0575	2.3577	0.57843
2	1.0184	1.5682	0.18738
3	1.0088	1.3022	0.067854
4	1.0048	1.1736	0.025373
5	1.0027	1.1031	0.0095948
⋮	⋮	⋮	⋮
38	1	1	0
39	1	1	0
40	1	1	0
41	1	1	0
42	1	1	0

De igual modo, ahora para el punto inicial $(-222, 222)$ se obtiene los siguientes resultados:

Cuadro 33: Método de descenso por gradiente Riemanniano (Armijo)

Iter	x	y	f	gradNorm
0	-222	222	6.0981	1.0022
1	-55.342	147.79	5.1256	1.0059
2	-8.0401	62.714	4.1437	1.0154
3	-0.29019	23.92	3.1369	1.0398
4	0.82869	8.7768	2.0661	1.0978
5	0.98146	3.0381	0.86178	1.1443
6	0.99998	1.0051	2.571e-05	0.010141
7	1	1.0002	3e-08	0.00033824
8	1	1	0	1.127e-05
9	1	1	0	3.8e-07

Cuadro 34: Descenso por gradiente Riemanniano (tamaño decreciente $1/k$)

Iter	x	y	f	gradNorm
0	-222	222	6.0981	1.0022
1	0.69035	11.781	2.3864	1.0764
2	0.99252	2.0815	0.44591	1.0251
3	1.0011	0.71994	0.10341	0.60334
4	0.99932	1.1409	0.017261	0.25994
5	1.0001	0.97512	0.00063472	0.050367
6	1	0.99999	0	2.665e-05
7	1	1	0	3.92e-06
8	1	1	0	1e-06
9	1	1	0	3.4e-07

Cuadro 35: Método del subgradiente Riemanniano ($1/k$)

Iter	x	y	f
0	-222	222	6.0981
1	-55.555	148.03	5.1278
2	-14.923	82.567	4.4385
3	-4.3425	48.42	3.8718
4	-1.0459	30.086	3.3762
5	0.13723	19.579	2.9268
⋮	⋮	⋮	⋮
807	1	1	0
808	0.99775	0.96663	0.0011562
809	0.99999	0.99997	0
810	1.0027	1.0344	0.0011513
811	1	1	0

Cuadro 36: Método del subgradiente Riemanniano (Polyak)

Iter	x	y	f
0	-222	222	6.0981
1	-2.7013	40.383	3.6825
2	0.78188	9.8894	2.1967
3	0.95718	4.4732	1.3076
4	0.98383	2.8597	0.79274
5	0.99161	2.1732	0.49067
⋮	⋮	⋮	⋮
67	1	1	0
68	1	1	0
69	1	1	0
70	1	1	0

6.4. Metodo subgradiente Greemberg-Pierskalla para funciones convexas en la variedad SPD

En esta sección evaluaremos la función $f : \mathbb{P}^n_{++} \to \mathbb{R}$ definida por:

$$f(X) = \ln\big(\det(X)\big)^2 \; - \; \ln\big(\det(X)\big)$$

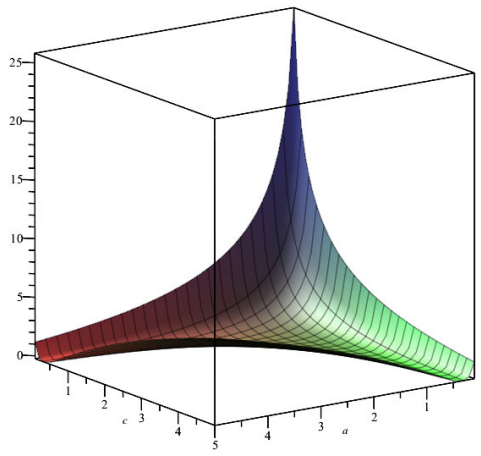


Figura 7: Gràfica de $f(X) = \big(\ln\det(X)\big)^2 - \ln\det(X)$

Ejemplo 1:

Para la matriz inicial de dimensión 3×3 generada de manera aleatoria con elementos en el intervalo $[0, 5]^n$ tenemos:

$$\begin{pmatrix} 1{,}0735 & -0{,}9233 & 0{,}1544 \\ -0{,}9233 & 1{,}8518 & -0{,}6571 \\ 0{,}1544 & -0{,}6571 & 0{,}3795 \end{pmatrix}$$

Iter	$f(X)$	$\det(X)$
0	7.053730	0.110522
1	3.122791	0.262760
2	1.335498	0.468057
3	0.432529	0.721695
4	−0,019847	1.020464
5	−0,213493	1.361970
6	−0,246823	1.744326
⋮	⋮	⋮
64	−0,250000	1.648043
65	−0,249333	1.691865
66	−0,250000	1.648688
67	−0,249352	1.691222
68	−0,250000	1.649297
69	−0,249405	1.608988
70	−0,250000	1.648722

Valor final de $f(X) = -0{,}25, \quad \det(X) = 1{,}648722.$

Ejemplo 2:

Para la matriz inicial de dimensión 5×5 generada de manera aleatoria con elementos en el intervalo $[0, 5]^n$ tenemos:

$$\begin{bmatrix} 1{,}9716 & -0{,}2052 & -0{,}0971 & 0{,}0515 & -2{,}0278 \\ -0{,}2052 & 3{,}1188 & -0{,}9499 & 0{,}7384 & 0{,}2659 \\ -0{,}0971 & -0{,}9499 & 1{,}4940 & -0{,}9736 & 0{,}4078 \\ 0{,}0515 & 0{,}7384 & -0{,}9736 & 1{,}5156 & -0{,}2848 \\ -2{,}0278 & 0{,}2659 & 0{,}4078 & -0{,}2848 & 3{,}0867 \end{bmatrix}$$

Iter	$f(X)$	$\det(X)$
0	1.329482	5.793720
1	−0,230751	1.894094
2	0.117982	0.898871
3	−0,247734	1.572083
4	−0,090308	2.458657
5	−0,249274	1.693737
6	−0,164443	1.230596
⋮	⋮	⋮
65	−0,250000	1.649571
66	−0,248920	1.595427
67	−0,250000	1.648762
68	−0,248951	1.596187
69	−0,250000	1.647999
70	−0,249036	1.700727
71	−0,250000	1.648720

Valor final de $f(X) = -0{,}25 \quad \det(X) = 1{,}648720.$

Ejemplo 3:

Para la matriz inicial de dimensión 8×8 generada de manera aleatoria con elementos en el intervalo $[0, 10]^n$ tenemos:

$$\begin{bmatrix} 4{,}5302 & -0{,}9237 & 0{,}8303 & 0{,}0497 & -0{,}5463 & -1{,}7458 & -0{,}2303 & 0{,}6257 \\ -0{,}9237 & 4{,}8430 & -2{,}2044 & 1{,}5398 & 0{,}1637 & -0{,}9826 & 0{,}1661 & -0{,}2142 \\ 0{,}8303 & -2{,}2044 & 4{,}5211 & 1{,}1094 & 0{,}1608 & 0{,}7904 & 0{,}7060 & -0{,}7785 \\ 0{,}0497 & 1{,}5398 & 1{,}1094 & 4{,}0940 & 0{,}0605 & 0{,}4912 & -1{,}0460 & -2{,}3327 \\ -0{,}5463 & 0{,}1637 & 0{,}1608 & 0{,}0605 & 5{,}9755 & 0{,}7123 & 1{,}0412 & -0{,}6930 \\ -1{,}7458 & -0{,}9826 & 0{,}7904 & 0{,}4912 & 0{,}7123 & 4{,}8769 & 0{,}6494 & 0{,}8510 \\ -0{,}2303 & 0{,}1661 & 0{,}7060 & -1{,}0460 & 1{,}0412 & 0{,}6494 & 3{,}6425 & 0{,}2666 \\ 0{,}6257 & -0{,}2142 & -0{,}7785 & -2{,}3327 & -0{,}6930 & 0{,}8510 & 0{,}2666 & 4{,}3117 \end{bmatrix}$$

Iter	$f(X)$	$\det(X)$
0	81.781940	14145.453856
1	58.164456	3438.996548
2	44.641688	1339.599497
3	35.666279	660.514571
4	29.205952	375.152931
5	24.311235	234.142344
6	20.469504	156.314221
⋮	⋮	⋮
68	−0,250000	1.649707
69	−0,248415	1.584378
70	−0,250000	1.648768
71	−0,248459	1.585254
72	−0,250000	1.647881
73	−0,248578	1.712086
74	−0,250000	1.648721

Valor final de $f(X) = -0{,}25, \quad \det(X) = 1{,}64721.$

6.5. Comparación Gradiente, subgradiente en Variedades Riemannianas y Euclidianas

6.5.1. Plano superior de Poincaré

Se llevó a cabo una comparación de los métodos de gradiente y subgradiente en Variedades Riemannianas y en el espacio Euclidiano. El procedimiento comenzó con la asignación de valores iniciales aleatorios para x e y, donde x está dentro del intervalo $[-10, 10]$ y y en el rango $[1, 10]$. La tolerancia establecida es 10^{-6} y el número máximo de iteraciones se fijó en 1000. Basándose en estos parámetros, se generaron los resultados siguientes. Los 100 puntos seleccionados aleatoriamente dentro del rango de $x \in [-10, 10]$ y $y \in [1, 10]$ se presentan en el cuadro siguiente. Estos puntos se emplearán para realizar las comparaciones respectivas

(x, y)	(x, y)	(x, y)	(x, y)	(x, y)
(2.3877, 6.5380)	(-7.5475, 2.1141)	(-4.3108, 7.6216)	(-1.7738, 8.4608)	(8.7023, 4.5916)
(-8.9558, 6.1407)	(4.9534, 3.8822)	(-0.1413, 2.9949)	(8.7855, 5.3407)	(0.7999, 2.9895)
(-8.0811, 1.5415)	(6.3902, 7.9433)	(-6.0861, 9.0561)	(3.6860, 6.9116)	(9.8076, 1.3032)
(-1.5149, 5.4099)	(1.6701, 1.7494)	(3.2031, 1.4707)	(1.1366, 7.4082)	(-0.2418, 6.5584)
(-5.7244, 6.8109)	(-2.3872, 1.9334)	(-2.4498, 3.3657)	(-5.1743, 6.6063)	(0.4586, 4.7191)
(-5.6442, 8.7270)	(7.2202, 3.5555)	(2.3079, 8.0154)	(9.0969, 9.2764)	(-2.3037, 2.4638)
(5.9350, 2.0244)	(-6.8235, 4.2025)	(6.9552, 6.2450)	(1.7236, 9.3325)	(1.5016, 1.0898)
(6.1875, 6.4793)	(-0.4017, 3.4160)	(-4.8381, 5.3291)	(-5.4532, 1.4374)	(-6.6152, 3.3260)
(-6.0418, 6.4512)	(6.4741, 8.2955)	(6.0447, 7.3730)	(7.1876, 8.0298)	(-5.9241, 9.9401)
(-8.1275, 6.8556)	(-5.6965, 3.1949)	(-3.2062, 2.7808)	(0.1367, 9.5568)	(-2.1079, 6.2602)
(2.1307, 7.4318)	(-1.9693, 8.7282)	(8.4097, 7.7576)	(-4.2882, 8.1713)	(-7.1447, 5.5410)
(2.2138, 7.3342)	(-2.3332, 7.5582)	(7.7457, 1.5026)	(-7.2357, 8.7676)	(-1.5651, 4.7018)
(9.1828, 7.7522)	(9.6199, 3.1016)	(-8.0755, 4.4612)	(0.0055, 6.1323)	(9.5326, 5.4357)
(-1.9823, 9.9549)	(-4.7800, 6.9879)	(9.2851, 7.0404)	(-4.0165, 5.7801)	(-9.9707, 8.9541)
(-1.9123, 3.7109)	(9.0116, 5.1458)	(-4.2470, 1.7616)	(1.6436, 2.3776)	(-8.5381, 6.2251)
(-4.2597, 4.2573)	(4.4965, 8.7248)	(-3.0417, 9.6557)	(9.0714, 2.8543)	(5.3649, 6.5398)
(8.3784, 6.4228)	(4.0428, 7.6931)	(-2.2980, 3.2635)	(-9.2648, 5.2491)	(2.9014, 3.5107)
(0.3572, 3.2110)	(-4.0498, 6.8543)	(7.8279, 8.7500)	(-5.8017, 4.5918)	(7.7577, 3.3088)
(9.3360, 6.5724)	(-6.6931, 8.4358)	(3.1139, 5.9181)	(-4.9734, 1.3614)	(-5.3325, 4.2500)
(2.6691, 9.8749)	(-5.8569, 7.8138)	(7.7266, 5.2501)	(-6.8217, 8.2983)	(-0.4698, 2.0466)

Cuadro 37: Valores aleatorios de x_0 y y_0 para las 100 ejecuciones

Basándonos en estos puntos seleccionados aleatoriamente, procederemos a elaborar el

cuadro comparativo siguiente. Además, designaremos como NC aquellos casos en los que la ejecución haya excedido el número máximo de iteraciones sin lograr convergencia.

Método	**Variedad Riemann**		**Euclídeo**	
	Prom it	%	Prom it	%
Gradiente (Armijo)	5.47	100.00	430.51	99.00
Gradiente (1/k)	96.77	93.00	911.48	17.00
Subgradiente simple (1/k)	83.68	100.00	NC	0,00
Subgradiente Polyak	86.53	97.00	NC	0,00

Cuadro 38: Comparación del método gradiente y subgradiente en la variedad Riemanniana y en el espacio Euclidiano.

hay que considerar que en el caso del método subgradiente utilizando el tamaño de paso de Polyak, aunque es cierto que el programa no excede el número máximo de iteraciones, no siempre logra converger al mínimo. Esto se debe a que, para alcanzar su mínimo, el valor de x debe tender a 0, situación que no ocurre en el contexto Euclidiano, como podemos ver en este ejemplo:

Iter	**x**	**y**	**f**
0	55555.0	55555.0	2.0000
1	41666.0	69444.0	1.3600
2	26348.0	78635.0	1.1123
3	14504.0	82603.0	1.0308
4	7468.7	83839.0	1.0079
5	3763.7	84169.0	1.0020
6	1885.6	84253.0	1.0005

Cuadro 39: Resultados del método del subgradiente (Polyak)

Ahora para los casos para el intervalo de $x \in [-1000, 1000]$ y el intervalo de $y \in [1, 1000]$ se obtuvieron los siguientes resultados

(x_0, y_0)	(x_0, y_0)	(x_0, y_0)	(x_0, y_0)	(x_0, y_0)
(-746.03, 913.46)	(264.72, 98.443)	(-443, 547.33)	(915.01, 964.92)	(-684.77, 970.62)
(914.33, 485.89)	(600.56, 142.74)	(-156.48, 915.82)	(584.41, 959.53)	(311.48, 36.676)
(698.26, 934.06)	(357.47, 757.98)	(486.26, 392.83)	(310.96, 172.02)	(412.09, 32.801)
(-446.15, 47.125)	(-805.74, 823.63)	(389.66, 317.78)	(900.44, 35.412)	(-122.51, 382.18)
(531.03, 795.4)	(-626.25, 490.27)	(-108.83, 646.67)	(418.73, 754.93)	(-447.95, 680.02)
(310.2, 163.45)	(-762, 498.87)	(919.49, 341.05)	(170.54, 224.59)	(502.53, 255.84)
(11.914, 699.38)	(781.81, 959.33)	(94.431, 139.49)	(-701.41, 258.25)	(681.43, 255.03)
(628.57, 244.28)	(858.53, 350.63)	(-606.81, 251.83)	(232.09, 473.82)	(-296.68, 831)
(170.53, 550.17)	(834.39, 286.55)	(514.4, 753.98)	(-239.11, 568.25)	(-848.29, 54.896)
(61.595, 779.39)	(868.02, 130.78)	(137.65, 469.92)	(-976.2, 337.79)	(-675.64, 794.49)
(-377.57, 529)	(-668.7, 602.38)	(-474.06, 654.43)	(378.43, 748.4)	(-98.917, 84.738)
(-542.05, 913.42)	(-695.24, 825.99)	(76.685, 996.14)	(-843.65, 443.24)	(-786.69, 961.94)
(-990.73, 775.14)	(634.61, 868.83)	(-831.13, 400.38)	(-480.26, 800.27)	(-137.17, 910.74)
(-636.31, 264.54)	(-708.92, 136.93)	(738.58, 580.12)	(99.72, 145.81)	(706.06, 622.43)
(-298.1, 513.74)	(-196.38, 76.891)	(-520.17, 124.2)	(-632.18, 240.71)	(-165.47, 50.605)
(805.43, 944.84)	(-18.272, 489.76)	(-324.56, 900.15)	(-261.51, 112.09)	(560.5, 390.35)
(-516.62, 404.51)	(-807.09, 132.84)	(884.1, 956.18)	(150.42, 60.72)	(-530.44, 353.81)
(642.39, 16.388)	(-913.95, 169.82)	(298.23, 731.99)	(295.49, 451.47)	(94.018, 297.02)
(489.39, 189.77)	(373.55, 184.33)	(-263.03, 625.99)	(560.45, 82.045)	(858.77, 775.94)
(-26.417, 436.42)	(-106.43, 307.04)	(17.017, 511.26)	(635.26, 795.04)	(288.64, 379.23)

Cuadro 40: Valores iniciales (x_0, y_0) organizados en 5 columnas y 20 filas

Método	**Variedad Riemann**		**Euclídeo**	
	Prom it	**%**	**Prom it**	**%**
Gradiente (Armijo)	25.41	100.00	*NC*	*NC*
Gradiente (1/k)	344.30	71.00	990.00	1.00
Subgradiente 1/k	114.45	99.00	990.00	1.00
Subgradiente Polyak	98.12	100.00	*NC*	0,00

Cuadro 41: Comparación entre métodos de optimización en variedades Riemannianas y Euclídeas

6.5.2. Matrices simétricas definidas positivas

En esta sección, se llevará a cabo un análisis comparativo entre los métodos gradiente y subgradiente de Greemberg-Pierskalla en la Variedad Riemanniana y en el Espacio Euclídeo.

Por restricciones de tiempo de ejecución, se ha fijado una tolerancia de 10^{-4}. Para matrices simétricas definidas positivas aleatorias con dimensiones n = 5, 10, 15, 20, 25, 35, 50, 100, se denotará con it el promedio de iteraciones. El promedio de funciones evaluadas que satisfacen la condición de Armijo y alcanzan su punto crítico se indica con $nfev$, y % representa el porcentaje de funciones exitosas. Se realizaron 100 compilaciones, y los resultados comparativos del Gradiente en el Espacio Euclídeo frente al Gradiente en la Variedad de Riemann se presentan en el cuadro siguiente. Esto se aplica a los casos donde los elementos de las matrices están en los intervalos $[0, 5]^n$, $[0, 10]^n$, y $[0, 20]^n$.

Caso para matrices con elementos en $[0, 5]^n$

	Gradiente Riemanniano			Gradiente Euclideano		
n	%	it	nfev	%	it	nfev
5	100.0	10.58	15.58	100.0	10.17	72.59
10	100.0	12	18	100.0	16.16	170.19
15	100.0	4.98	10.98	100.0	20.79	469.61
20	100.0	13.8	20.8	86.0	53.34	1843.06
25	100.0	9.45	16.45	65.0	54.29	1949.26
30	100.0	5.50	12.50	30.0	67.47	2441.20
50	100.0	10.44	18.44	NC	NC	NC
100	100.0	18.9	29.7	NC	NC	NC

Cuadro 42: Comparación de métodos de gradiente Euclidiano y Riemanniano para matrices con elementos en $[0, 15]^n$

Caso para matrices con elementos en $[0, 10]^n$

	Gradiente Riemanniano			Gradiente Euclideano		
n	%	it	nfev	%	it	nfev
5	100.0	11.41	16.41	100.0	16.64	135.35
10	100.0	13.01	19.01	75.0	30.81	1982.62
15	100.0	5.28	11.28	21.0	98.52	4422.05
20	100.0	14.83	21.83	3.0	34.0	1311.67
25	100.0	10.28	17.28	*NC*	*NC*	*NC*
30	100.0	5.98	12.98	*NC*	*NC*	*NC*
50	100.0	11.04	19.04	*NC*	*NC*	*NC*
100	100.0	12.0	21.0	*NC*	*NC*	*NC*

Cuadro 43: Comparación de métodos de gradiente Euclidiano y Riemanniano para matrices con elementos en $[0, 10]^n$

Caso para matrices con elementos en $[0, 20]^n$

	Gradiente Riemanniano			Gradiente Euclideano		
n	%	it	nfev	%	it	nfev
5	100.0	11.99	16.99	100.0	23.01	269.27
10	100.0	13.56	219.56	19.0	59.53	2356.47
15	100.0	5.33	11.33	*NC*	*NC*	*NC*
20	100.0	15.09	22.09	*NC*	*NC*	*NC*
25	100.0	17.7	26.1	*NC*	*NC*	*NC*
30	100.0	17.4	25.7	*NC*	*NC*	*NC*
50	100.0	11.26	19.26	*NC*	*NC*	*NC*
100	100.0	12.37	21.37	*NC*	*NC*	*NC*

Cuadro 44: Comparación de métodos de gradiente Euclidiano y Riemanniano para matrices con elementos en $[0, 20]^n$

6.6. Implementación del Subgradiente de Greemberg-Pierskalla

6.6.1. Comparacion de Subgradiente en el espacio Euclidiano y espacio Riemanniano

En el siguiente caso para elementos entre $[0, 5]^n$ el método subgradiente de Greemberg-Pierskalla

Primer Caso para matrices con elementos en $[0, 5]^n$

	Subgradiente Euclidiano		Subgradiente Riemanniano	
n	%	it	%	it
10	6.00	64.17	100.00	64.41
20	NC	NC	100.00	98.83
30	NC	NC	100.00	158.42
50	NC	NC	98.00	312.47
100	NC	NC	71.00	622.87

Cuadro 45: Comparación de métodos de subgradiente Euclidiano y Riemanniano en la variedad SPD

En el siguiente caso para elementos entre $[0, 10]^n$ el método subgradiente en el espacio Euclidiano generó un índice muy bajo de convergencia en las cien ejecuciones del programa, de este modo podemos asegurar que en el espacio Euclidiano tiende a converger cuando los elementos de las matrices están entre $[0, 5]^n$ o menores a estos, por ejemplo con matrices de dimensiones 5×5 se puede obtener como resultados

- Promedio de iteraciones: 87.00
- Eficiencia: 63.00 %

pasando estos parámetros, las funciones sobrepasan el número máximo de iteraciones, lo cual generará que no converja a un punto crítico; de este modo, mostraremos cómo se comporta el método subgradiente en el espacio Riemanniano

caso para matrices simétricas definidas positivas con elementos $[0, 10]^n$

n	Subgradiente Riemanniano ($[0, 10]^n$)	
	%	it
10	100.00	161.69
20	76.00	474.12
30	33.00	590.97
50	1.00	909.00
100	NC	NC

Cuadro 46: Rendimiento del método de subgradiente Riemanniano con puntos iniciales aleatorios en $[0, 10]^n$.

Para elementos que exceden dimensiones de 30×30, el programa experimenta una disminución significativa en su velocidad de convergencia. Al alcanzar matrices con dimensiones de 50×50 o 100×100, se supera el límite de iteraciones permitido. Aunque es posible que el programa pueda converger con más de mil iteraciones, dichas iteraciones no serán consideradas al calcular el promedio de iteraciones ni al determinar el porcentaje de funciones que convergen de forma exitosa.

VII. Conclusiones

Hemos introducido el método subgradiente en Variedades Riemannianas, centrándonos en particular en la variedad del plano superior de Poincaré y el espacio de las matrices definidas positivas para funciones cuasi-convexas en este contexto. Es importante destacar que, bajo las definiciones y referencias literarias, el subdiferencial convexo puede resultar nulo para funciones cuasiconvexas.

Un aporte de este trabajo se encuentra en el capítulo II, donde desarrollamos de manera exhaustiva las propiedades geométricas en el espacio superior de Poincaré. En particular, desarrollamos la solución de la ecuación diferencial vinculada al cálculo de las geodésicas. Debido que esta aplicación a problemas computacionales se presenta altamente compleja, empleamos una aproximación de la solución del problema de Cauchy de las geodésicas utilizando el método de Runge-Kutta, apoyandonos en el trabajo de Udristre [26]. Así, esta aproximación se implementó en los algoritmos del método gradiente y subgradiente.

Se aporta también en el cálculo del subdiferencial de Gremmberg-Pierskalla para la variedad de matrices Definidas Positivas (SPD), permitiendo aplicar estos modelos en el ámbito computacional y realizar comparaciones con los mismos métodos en el espacio Euclidiano.

Con respecto a la parte computacional, un aporte adicional es comparar tanto los métodos gradiente y subgradiente en la variedad del plano superior de Poincaré como en la variedad de matrices simétricas definidas positivas, así mostramos que el método gradiente y subrgradiente tienen ventaja a comparación de los mismos métodos pero en espacios Euclidianos.

Respecto a la parte teórica, se ha presentado la convergencia que desarrolló Hansari y Uddin et.al [5], particularizando para el caso de una función objetivo cuasi-convexa.

Investigaciones futuras aún requieren examinar variedades con una curvatura seccional acotada inferiormente, lo que permitirá definir y determinar un subdiferencial adecuado para poder hallar el mínimo de funciones cuasiconvexas pero restringidas en la variedad.

Referencias

[1] S. S. Ablaev et al. “On Some Works of Boris Teodorovich Polyak on the Convergence of Gradient Methods and Their Development”. En: *Computational Mathematics and Mathematical Physics* 64.4 (abr. de 2024), págs. 635-675.

[2] Qamrul Ansari, Feeroz Babu y Mohd Zeeshan. “Incremental Quasi-Subgradient Method for Minimizing Sum of Geodesic Quasi-Convex Functions on Riemannian Manifolds with Applications”. En: *Numerical Functional Analysis and Optimization* 42 (nov. de 2021). DOI: 10.1080/01630563.2021.2001823.

[3] Qamrul Ansari y Moin Uddin. “Convergence analysis of incremental quasi-subgradient method on Riemannian manifolds with lower bounded curvature”. En: *Optimization* (oct. de 2024), págs. 1-20. DOI: 10.1080/02331934.2024.2414782.

[4] Qamrul Hasan Ansari, Feeroz Babu y Mohd Zeeshan. “Incremental Quasi-Subgradient Method for Minimizing Sum of Geodesic Quasi-Convex Functions on Riemannian Manifolds with Applications”. En: *Numerical Functional Analysis and Optimization* 42.13 (2021), págs. 1492-1521.

[5] Qamrul Hasan Ansari y Moin Uddin. “Convergence analysis of incremental quasi-subgradient method on Riemannian manifolds with lower bounded curvature”. En: *Optimization* 0.0 (2024), págs. 1-20. DOI: 10.1080/02331934.2024.2414782.

[6] Manfredo Perdigao Do Carmo y Francis J. Flaherty. *Riemannian Geometry*. Vol. 6. Progress in Mathematics. Springer, 1992.

[7] Rafael Correa y Claude Lemaréchal. “Convergence of some algorithms for convex minimization”. En: *Mathematical Programming* 62 (1993), págs. 261-275.

[8] João Xavier da Cruz Neto et al. “A Subgradient Method for Multiobjective Optimization”. En: *Computational Optimization and Applications* 54.3 (2013), págs. 461-472.

[9] Manfredo P Do Carmo. *Differential geometry of curves and surfaces: revised and updated second edition*. Courier Dover Publications, 2016.

[10] O. P. Ferreira, M. S. Louzeiro y L. F. Prudente. “Iteration-Complexity of the Subgradient Method on Riemannian Manifolds with Lower Bounded Curvature”. En: *Unknown Journal* (2020).

[11] O. P. Ferreira y P. R. Oliveira. “Subgradient Algorithm on Riemannian Manifolds”. En: *Journal of Optimization Theory and Applications* 97 (1998), págs. 93-104.

[12] Orizon P Ferreira, Maurício Silva Louzeiro y LF4018420 Prudente. “Gradient method for optimization on Riemannian manifolds with lower bounded curvature”. En: *SIAM Journal on Optimization* 29.4 (2019), págs. 2517-2541.

[13] Philipp Grohs, Martin Holler y Andreas Weinmann. *Handbook of variational methods for nonlinear geometric data*. Springer, 2020.

[14] R Herman. *Euler equation and geodesics*. 2018.

[15] Jiang Hu et al. *A Brief Introduction to Manifold Optimization*. 2020. DOI: 10.1007/s40305-020-00295-9.

[16] Jürgen Jost. *Riemannian Geometry and Geometric Analysis*. 6.ª ed. Universitext. 12 b/w illustrations, 4 illustrations in colour. Springer Berlin, Heidelberg, 2011, págs. XIII+611. ISBN: 978-3-642-21298-7. DOI: 10.1007/978-3-642-21298-7. URL: https://doi.org/10.1007/978-3-642-21298-7.

[17] Philip D. Loewen. “The Proximal Subgradient Formula in Banach Space”. En: *Canadian Mathematical Bulletin* 31.3 (1988), págs. 353-361.

[18] Martina Magliocca. *Apuntes del curso Análisis Matemático: Bloque 2: Cálculo Diferencial e Integral en varias variables*. Grado en Física, Grupo 2, Curso 2023/2024. Universidad de Sevilla, 2023.

[19] Santiago Martínez Balvanera. "De las variedades con curvatura seccional positiva". Spanish. Asesor: Palmas Velasco, Óscar Alfredo. Tesis de Licenciatura. México: Facultad de Ciencias, UNAM, 2015. URL: https://repositorio.unam.mx/contenidos/177912.

[20] Nesterov y Todd. "On the Riemannian geometry defined by self-concordant barriers and interior-point methods". En: *Foundations of Computational Mathematics* 2 (2002), págs. 333-361.

[21] Lyle Noakes y Erchuan Zhang. "Finding geodesics joining given points". En: *Advances in Computational Mathematics* 48.4 (2022), pág. 50.

[22] Erik Alex Papa Quiroz. *Un Método no Euclidiano para Problemas de Optimización en Espacios Euclidianos*. Resolución Rectorial N° 1352-2007-R. Perú: Universidad Nacional del Callao, Facultad de Ciencias Naturales y Matemática, Instituto de Investigación, 2007.

[23] T. Rapcsák. "Geodesic Convexity in Nonlinear Optimization". En: *Journal of Optimization Theory and Applications* 91 (1991), Unknown.

[24] Takashi Sakai. *Riemannian Geometry*. Vol. 149. Translations of Mathematical Monographs. American Mathematical Society, 1996.

[25] Naun Zuselevich Shor. "The Subgradient Method Minimization Methods for Non-Differentiable Functions". En: *Unknown Journal* (1985), págs. 22-47.

[26] Constantin Udriste. *Convex Functions and Optimization Methods on Riemannian Manifolds*. English. Vol. 297. Mathematics and its Applications. Kluwer Academic Publishers, 1994.

[27] X. M. Wang. "Subgradient Algorithms on Riemannian Manifolds of Lower Bounded Curvatures". En: *Optimization* 67.1 (2018), págs. 179-194.

[28] X. M. Wang, Chong Li y Jen-Chih Yao. "Subgradient Projection Algorithms for Convex Feasibility on Riemannian Manifolds with Lower Bounded Curvatures". En: *Journal of Optimization Theory and Applications* 164 (2015), págs. 202-217.

[29] Xiangmei Wang, Chong Li y Jen-Chih Yao. "On Some Basic Results Related to Affine Functions on Riemmanian Manifolds". En: *Journal of Optimization Theory and Applications* 170 (sep. de 2016). DOI: 10.1007/s10957-016-0979-x.

[30] Xiangmei Wang et al. "Linear Convergence of Subgradient Algorithm for Convex Feasibility on Riemannian Manifolds". En: *SIAM Journal on Optimization* 25.4 (2015), págs. 2334-2358.

[31] Xueyong Wang, Shengjie Li, Xipeng Kou et al. "An Extension of Subgradient Method for Variational Inequality Problems in Hilbert Space". En: *Abstract and Applied Analysis*. Vol. 2013. Hindawi. Hindawi, 2013.

[32] Peng Zhang y Gejun Bao. "An Incremental Subgradient Method on Riemannian Manifolds". En: *Journal of Optimization Theory and Applications* 176 (2018), págs. 711-727.

[33] Dennis G Zill. *Differential equations with boundary-value problems*. Cengage Learning, 2016.

VIII. Anexos

8.1. Códigos Maple

En esta sección mostraremos los códigos que se han usado para poder realizar algunos cálculos tanto de las derivadas de las componentes de las geodésicas, para poder hallar $\exp_y^{-1} x = \gamma'(0)$. También se muestra el código del método de Runge-Kutta de orden 4 para aproximar las geodésicas y, por último, el código para calcular la matriz hessiana de una función definida en el plano superior de Poincaré.

8.1.1. Código Maple para las derivadas de la primera componente de las geodésicas

```
restart;
with(Student[Calculus1]);
with(plots);
bxy := 'bxy';
rxy := 'rxy';
t1 := 't1';
s1 := 's1';
s := 's';
gamma_s := bxy - rxy*tanh((1 - s)*arctanh((bxy - t1)/rxy) + s*arctanh((bxy -
    s1)/rxy));
dgamma_s := diff(gamma_s, s);
dgamma_s0 := subs(s = 0, dgamma_s);
dgamma_s;
dgamma_s0;
```

Listing 1: Codigo Maple para las derivadas de las geodesicas

8.1.2. Cálculo de la derivada de la segunda componente de la geodésica

```
restart;
with(Student[Calculus1]);
with(plots);
bxy := 'bxy';
rxy := 'rxy';
t1 := 't1';
s1 := 's1';
s := 's';
arg := (1 - s)*arctanh((bxy - t1)/rxy) + s*arctanh((bxy - s1)/rxy);
gamma2 := rxy/cosh(arg);
dgamma2 := diff(gamma2, s);
du_ds := diff(arg, s);
d_cosh := -sinh(arg)*du_ds/cosh(arg)^2;
dgamma2_0 := simplify(subs(s = 0, d_cosh));
arg;
gamma2;
du_ds;
d_cosh;
dgamma2_0;
```

Listing 2: Codigo Maple para las derivadas de las geodesicas

8.1.3. Código de método de Runge-Kutta para aproximar geodésicas

```
1
2  with(plots):
3  # 1) Definicion de la derivada del sistema
4  f := proc(s::list)
5      local x,y,u,v;
6      x := s[1];          y := s[2];
7      u := s[3];          v := s[4];
8      if y <= 0 then
9          error "y debe ser > 0 en todo momento";
10     end if;
11     return [u, v, 2*u*v/y, (v^2 - u^2)/y];
12 end proc:
13
14 # 2) Paso de Runge-Kutta de cuarto orden
15 rk4_step := proc(s::list, h::numeric)
16     local k1,k2,k3,k4,s2,s3,s4;
17     k1 := f(s);
18     s2 := [op(s) + (h/2)*op(k1)];
19     k2 := f(s2);
20     s3 := [op(s) + (h/2)*op(k2)];
21     k3 := f(s3);
22     s4 := [op(s) + h*op(k3)];
23     k4 := f(s4);
24     return [seq(
25         s[i] + (h/6)*(k1[i] + 2*k2[i] + 2*k3[i] + k4[i]),
26         i=1..4
27     )];
28 end proc:
29
30 # 3) Procedimiento principal para integrar y graficar
31 PoincareGeodesic := proc(
32     x0::numeric, y0::numeric,
33     u0::numeric, v0::numeric,
34     tau::numeric, nsteps::integer
35 )
36     local h,i,states,new,pts_xy;
37     if y0 <= 0 then
38         error "La condicion inicial y0 debe ser > 0";
39     end if;
40     if u0 = 0 and v0 = 0 then
41         error "El vector direccion no puede ser el vector nulo";
42     end if;
43
44     h := tau/nsteps;
45     states := [[x0,y0,u0,v0]];
46
47     printf("Step    x        y        u        v\n");
48     printf("%4d %8.4f %8.4f %8.4f %8.4f\n",
49            0, x0, y0, u0, v0);
50
51     for i from 1 to nsteps do
52         new := rk4_step(states[-1], h);
53         states := [op(states), new];
54         printf("%4d %8.4f %8.4f %8.4f %8.4f\n",
55                i, new[1], new[2], new[3], new[4]);
56     end do:
57
58     pts_xy := map(p -> [p[1], p[2]], states);
59     with(plots):
60     display(
61       [ listplot(pts_xy, style=line),
62         listplot(pts_xy, style=point) ],
63       title="Trayectoria de las geodesicas de Poincare",
64       labels=["x","y"], grid=true
65     );
66 end proc:
67
68 # Ejemplo de uso:
69 PoincareGeodesic(1.0, 0.1, 0.1, 1.0, 3.0, 100);
```

Listing 3: Código Maple para la aproximación de las geodésicas por Runge-Kutta

8.1.4. Código para el calculo de la matriz hessiana en el plano superior de Poincaré en Maple

```
Hessiano := proc(f, x1, x2)
    local grad_f, hess_f, Gamma_x1, Gamma_x2, ecuacion_3, hessiana_final;

    Gamma_x1 := Matrix([[0, -1/x2],
                        [-1/x2, 0]]);
    Gamma_x2 := Matrix([[1/x2, 0],
                        [0, -1/x2]]);

    grad_f := Vector([diff(f, x1), diff(f, x2)]);

    hess_f := Matrix([[diff(f, x1, x1), diff(f, x1, x2)],
                      [diff(f, x2, x1), diff(f, x2, x2)]]);

    ecuacion_3 := grad_f[1]*Gamma_x1 + grad_f[2]*Gamma_x2;

    hessiana_final := hess_f - ecuacion_3;

    return hessiana_final;
end proc;

#Evaluacion para f(x1,x2) = (x1 - 5)^2 + x2^2:
# Resultado:
# B = Hess_g f(x1, x2) =
#      [   0            2*(x1 - 5)/x2 ]
#      [ 2*(x1 - 5)/x2       4        ]
```

Listing 4: Codigo Maple para el calculo del matriz Hessiana

8.1.5. Código para la determinante Hessiana en Maple

```
with(LinearAlgebra):

x1, x2, a := 'x1', 'x2', 'a':

C := B:

detC := Determinant(C):

simplify(detC):

# Si B = Matrix([[0, 2*(x1 - 5)/x2], [2*(x1 - 5)/x2, 4]])
# Entonces:
# detC = -4*(x1 - 5)^2 / x2^2
```

Listing 5: Codigo Maple para el calculo de la determinante de la matriz Hessiana

8.2. Códigos para la comparación del método Gradiente en la variedad SPD y el espacio Euclidiano

En esta subsección se adjuntará los códigos que se usaron para poder realizar la comparación de la eficiencia del método gradiente en el espacio Euclidiano como en la variedad SPD, para ello se modificó el código de [12].
Los códigos están organizados de la siguiente manera:

- main.m: Programa principal.

- Gradient.m:Ejecuta el método gradiente Euclidiano para matrices simétricas definidas positivas.
- GradRiemann.m: Ejecuta el método gradiente Riemanniano en la variedad de matrices simétricas definidas positivas
- evalf.m: Define la función que se va a evaluar: $f(X) = a.\ln(det(X))^2 - b.ln(det(X))$, para el caso de $a = b = 1$.
- evalg.m: Se calcula el gradiente Euclidiano de la función f.
- comp.m: Se realiza la comparación de los métodos de gradiente Euclidiano y de la variedad SPD.

8.2.1. main.m

```
    clear all
clc

global a b problem

strategy = 3;
problem = 3;
if  problem == 3

    % Set the dimension of the problem
    n = 10;

    % Set the problem data
    alpha = 0;
    beta  = 10;

    a = 1.0;
    b = 1.0;
    % Set the stating point
    l = 0;
    u = 10;

    lambda = l + rand(1,n) * ( u - l );

    xini = orth(randn(n));
    xini = xini * diag(lambda) * xini';

    if ( ~isreal(xini) )
        xini = real(xini);
    end
    fprintf('xini = [\n');
    for i = 1:n
        fprintf('  %s\n', sprintf('%12.6f ', xini(i,:)));
    end
fprintf(']\n');
    % -------- Llamada a GradRiemannian -----
[xR, solinfoR, itR, timeR, nfevR, ngevR] = GradRiemannian(xini, n, strategy);
fprintf('Riemannian gradient:\n');
fprintf('  iter = %4d,  nfev = %4d,  ngev = %4d,  time = %6.3fs\n\n', ...
        itR, nfevR, ngevR, timeR);

% ------ Llamada a Gradient (Euclidiano) --------
[xG, solinfoG, itG, timeG, nfevG, ngevG] = Gradient(xini, n);
fprintf('Euclidean gradient:\n');
fprintf('  iter = %4d,  nfev = %4d,  ngev = %4d,  time = %6.3fs\n', ...
        itG, nfevG, ngevG, timeG);

end
```

Listing 6: Código Matlab para el cálculo de la gradiente Euclidiana

8.2.2. Gradient.m

```
function [x,solinfo,it,time,nfev,ngev] = Gradient(x,n)

global  problem

% ------------------------------------------------------------------------
%   Inicializacion
% ------------------------------------------------------------------------

% Iniciar cronometro

tic;

% Establecer parametros por defecto

opttol = 10^(-4);
maxit  = 1000;
ftol   = 1/2;
eps    = 10^(-10);
%iprint = true;
iprint=false;

% Contadores

it   = 0;
nfev = 0;
ngev = 0;

% Evaluar funcion objetivo y gradiente

[f]  = evalf(x,n);
nfev = nfev + 1;

[g]  = evalg(x,n);
ngev = ngev + 1;

% Calcular la norma del gradiente

gsupn = norm(g,inf);

% ------------------------------------------------------------------------
%   Bucle principal
% ------------------------------------------------------------------------

while(1)

    % Imprimir informacion

    if ( iprint )
        if ( it == 0 )
            fprintf('-------------------------------------------------------\n')
            fprintf('                 metodo del gradiente Euclidiano \n')
            fprintf('-------------------------------------------------------\n')
        end

        if ( mod(it,10) == 0 )
            fprintf('\n')
            fprintf('  It     objective      gsupn          t            nfev \n')
        end

        if ( it == 0 )
            fprintf(' %4d    %+6.2e     %6.2e        -          %5d
                \n',it,f,gsupn,nfev)
```

```
62          else
63              fprintf(' %4d    %+6.2e     %6.2e      %6.2e      %5d
                    \n',it,f,gsupn,t,nfev)
64          end
65      end
66
67      % ----------------------------------------------------------------------
68      %   Verificar criterio de paro
69      % ----------------------------------------------------------------------
70
71      % Verificar optimalidad
72
73      if ( gsupn <= opttol )
74          solinfo = 0;
75
76          % Detener cronometro
77
78          time = toc;
79
80          if ( iprint )
81              fprintf('\n')
82              fprintf('Se encontro la solucion \n\n')
83              fprintf('Numero de evaluaciones de la funcion: %6d \n',nfev)
84                          fprintf('Tiempo total de CPU en segundos: %10.4f
                                \n\n',time)
85          end
86
87          return
88      end
89
90      % Verificar si se alcanzo el maximo de iteraciones
91
92      if ( it >= maxit )
93          solinfo = 1;
94
95          % Detener cronometro
96
97          time = toc;
98
99          if ( iprint )
100             fprintf('\n')
101             fprintf('Se alcanzo el maximo de iteraciones \n\n')
102             fprintf('Numero de evaluaciones de la funcion: %6d \n',nfev)
103                         fprintf('Tiempo total de CPU en segundos: %10.4f
                                \n\n',time)
104         end
105
106         return
107     end
108
109     % Incrementar el contador de iteraciones
110
111     it = it + 1;
112
113     % ----------------------------------------------------------------------
114     %   Busqueda lineal
115     % ----------------------------------------------------------------------
116
117     if  ( problem == 3  )
118
119         % Calcular ||g||^2
120
121         geucn2 = trace( g ^ 2 );
122
123         % Calcular el maximo valor de t tal que t <= 1 y x - t * g > 0
124
125         % Calcular el mayor autovalor de g
126
127         eigg = eigs(g,1,'LA');
128
129         if ( eigg <= 0 )
130             t = 1;
```

```
131         else
132
133             % Calcular el menor autovalor de x
134
135             eigx = eigs(x,1,'SA');
136
137             if ( eigx <= 0 )
138
139                 solinfo = 2;
140
141                 % Detener cronometro
142
143                 time = toc;
144
145                 if ( iprint )
146                     fprintf('\n')
147                     fprintf('Error: el punto actual no es definido positivo \n\n')
148                     fprintf('Numero de evaluaciones de la funcion: %6d \n',nfev)
149                     fprintf('Tiempo total de CPU en segundos: %10.4f
                            \n\n',time)
150                 end
151
152                 return
153             end
154
155             t = eigx / eigg;
156
157             if ( t <= eps )
158                 t = 0.1 * t;
159                 %disp('Advertencia: paso permitido demasiado pequeno')
160             else
161                 t = t - eps;
162             end
163
164             t = min( 1 , t );
165         end
166
167     end
168
169     % Definir el punto de prueba
170
171     xtrial    = x - t * g;
172
173     % Evaluar la funcion objetivo en el punto de prueba
174
175     [ftrial] = evalf(xtrial,n);
176     nfev = nfev + 1;
177
178     while ( ftrial > f - ftol * t * geucn2 )
179
180         % Definir el nuevo paso de prueba
181
182         t = t/2;
183
184          % Definir el nuevo punto de prueba
185
186         xtrial    = x - t * g;
187
188         % Evaluar la funcion objetivo en el nuevo punto de prueba
189
190         [ftrial] = evalf(xtrial,n);
191         nfev = nfev + 1;
192     end
193
194     % ------------------------------------------------------------------
195     %   Actualizar x
196     % ------------------------------------------------------------------
197
198     x = xtrial;
199     f = ftrial;
200
201     % Calcular el gradiente de f
```

```
202
203     [g]  = evalg(x,n);
204     ngev = ngev + 1;
205
206     % Calcular la norma  del gradiente
207
208     gsupn = norm(g,inf);
209
210     % ---------------------------------------------------------------------
211     %   Iterar
212     % ---------------------------------------------------------------------
213 end
```

Listing 7: Código Matlab para el cálculo de la gradiente Euclidiana

8.2.3. GradRiemann

```
1     function [x,solinfo,it,time,nfev,ngev] = GradRiemannian(x,n,strategy)
2
3 global a b problem
4
5 % ---------------------------------------------------------------------
6 %   Initialization
7 % ---------------------------------------------------------------------
8
9 % Start timing
10
11 tic;
12
13 % Set default parameters
14
15 opttol = 10^(-3);
16 maxit  = 1000;
17 ftol   = 1/2;
18 %iprint = true;
19 iprint=false;
20
21 % Set the initial trial value for the Lipschitz constant for strategy 3
22
23 if ( strategy == 3 )
24     L0 = 1;
25     t  = 1 / L0;
26 end
27
28 % Counters
29
30 it   = 0;
31 nfev = 0;
32 ngev = 0;
33
34 % Evaluate objective function and gradient
35
36 [f]  = evalf(x,n);
37 nfev = nfev + 1;
38
39 [g]  = evalg(x,n);
40 ngev = ngev + 1;
41
42 % Compute the sup-norm of the gradient
43
44 gsupn = norm(g,inf);
45
46 % ---------------------------------------------------------------------
47 %   Main loop
48 % ---------------------------------------------------------------------
49
50 while(1)
51
```

```
52      % Print information
53
54      if ( iprint )
55          if ( it == 0 )
56              fprintf('------------------------------------------------------------\n')
57              fprintf('                  Riemannian gradient method
                                \n')
58              fprintf('------------------------------------------------------------\n')
59          end
60
61          if ( mod(it,10) == 0 )
62              fprintf('\n')
63              fprintf('  It     objective      gsupn           t            nfev \n')
64          end
65
66          if ( it == 0 )
67              fprintf(' %4d    %+6.2e     %6.2e     - %5d\n',it,f,gsupn,nfev)
68          else
69              fprintf(' %4d    %+6.2e     %6.2e      %6.2e      %5d\n',it,f,gsupn,t,nfev)
70          end
71      end
72
73      % -----------------------------------------------------------------------
74      %   Test stopping criteria
75      % -----------------------------------------------------------------------
76
77      % Test optimality
78
79      if ( gsupn <= opttol )
80          solinfo = 0;
81
82          % Stop timing
83
84          time = toc;
85
86          if ( iprint )
87              fprintf('\n')
88              fprintf('Solution was found \n\n')
89              fprintf('Number of functions evaluations  : %6d \n',nfev)
90                          fprintf('Total CPU time in seconds: %10.4f \n\n',time)

91          end
92
93          return
94      end
95
96      % Test whether the number of iterations is exhausted
97
98      if ( it >= maxit )
99          solinfo = 1;
100
101         % Stop timing
102
103         time = toc;
104
105         if ( iprint )
106             fprintf('\n')
107             fprintf('Maximum of iterations reached \n\n')
108             fprintf('Number of functions evaluations  : %6d \n',nfev)
109                         fprintf('Total CPU time in seconds: %10.4f \n\n',time)

110         end
111
112         return
113     end
114
115     % Increment the iteration counter
116
117     it = it + 1;
118
119     % -----------------------------------------------------------------------
120     %   Use the constant stepsize for strategy 1 or perform the line search
```

```
    %   for strategys 2 and 3
    % ------------------------------------------------------------------


    % Define the trial point

    if ( problem == 3 )
        gradfeucn2 = trace( ( x * g )^2 );
        detx = det(x);
        xtrial = ( detx^( 2 * a ) * exp( - b ) )^( -t ) * x;
    end

    % Evaluate the objective function at the trial point

    [ftrial] = evalf(xtrial,n);
    nfev = nfev + 1;

    % ------------------------------------------------------------------
    %   Line search
    % ------------------------------------------------------------------

    % Perform the line search for strategys 2 and 3

    if (strategy == 3 )
        while ( ftrial > f - ftol * t * gradfeucn2 )

            % Define the new trial stepsize

            t = t/2;

            % Define the new trial point

            if  ( problem == 3 )
                xtrial = ( detx^( 2 * a ) * exp( - b ) )^( -t ) * x;
            end

            % Evaluate the objective function at the new trial point

            [ftrial] = evalf(xtrial,n);
            nfev = nfev + 1;
        end
    end

    % ------------------------------------------------------------------
    %   Update x
    % ------------------------------------------------------------------

    x = xtrial;
    f = ftrial;

    % Compute the gradient of f

    [g]  = evalg(x,n);
    ngev = ngev + 1;

    % Compute the sup-norm of the gradient

    gsupn = norm(g,inf);

    % ------------------------------------------------------------------
    %   Iterate
    % ------------------------------------------------------------------
end
```

Listing 8: Codigo Matlab para el calculo de la gradiente Riemanniana

8.2.4. evalf.m

```
    function f = evalf(x,~)

global a b problem

% ---------------  Problem 3 ---------------

if  problem == 3
    detx = det( x );

    f = a * log( detx )^2 - b * log( detx );
    return
end


end
```

Listing 9: Código Matlab para evaluar la función

8.2.5. evalg.m

```
    function g = evalg(x,~)

global a b problem
% ---------------  Problem 3 ---------------

if  problem == 3
    detx = det(x);

    g = ( 2 * a * log( detx ) - b ) * inv(x);

    if ( ~isreal(g) )
        g = real(g);
    end

    if ( ~issymmetric(g) )
        g = ( g + g' )/2;
    end
    return
end
end
```

Listing 10: Código Matlab para hallar el gradiente Euclidiano de la función

8.2.6. comp.m

```
    clear all
clc

global a b problem

%% Configuracion general
strategy = 3;       % Estrategia Armijo
problem  = 3;       % Problema 3 del articulo
numRuns  = 100;     % Numero total de ejecuciones

%% Pre-asignacion de vectores para almacenar resultados
iterR    = zeros(numRuns,1);    % Iteraciones Riemanniano
nfevR    = zeros(numRuns,1);    % Evaluaciones f Riemanniano
successR = false(numRuns,1);    % Convergencia Riemanniano

iterG    = zeros(numRuns,1);    % Iteraciones Euclidiano
nfevG    = zeros(numRuns,1);    % Evaluaciones f Euclidiano
successG = false(numRuns,1);    % Convergencia Euclidiano
```

```
19
20 %% Bucle de ejecuciones
21 for k = 1:numRuns
22     % Parametros especificos del problema
23     n = 100;        % Dimension
24     a = 1.0;        % Parametro a de la funcion
25     b = 1.0;        % Parametro b de la funcion
26     l = 0; u = 20; % Intervalo de autovalores
27
28     % Generacion de la matriz inicial semidefinida positiva X0
29     lambda = l + rand(1,n)*(u - l);
30     X0 = orth(randn(n));
31     X0 = X0 * diag(lambda) * X0';
32     if ~isreal(X0)
33         X0 = real(X0);
34     end
35
36     %-----------------------------------------
37     %Metodo de Gradiente Riemanniano
38     %-----------------------------------------
39     [~, solinfoR_k, itR_k, ~, nfevR_k, ~] = GradRiemannian(X0, n, strategy);
40     iterR(k)    = itR_k;
41     nfevR(k)    = nfevR_k;
42     successR(k) = (solinfoR_k == 0);
43
44     % ---------------------------------
45     %Metodo de Gradiente Euclidiano
46     %---------------------------------
47     [~, solinfoG_k, itG_k, ~, nfevG_k, ~] = Gradient(X0, n);
48     iterG(k)    = itG_k;
49     nfevG(k)    = nfevG_k;
50     successG(k) = (solinfoG_k == 0);
51 end
52
53 %% CAlculo de convergencias efectivas
54 NR = sum(successR);
55 NG = sum(successG);
56
57 %% Promedios solo sobre ejecuciones convergentes
58 if NR > 0
59     avgIterR = mean(iterR(successR));
60     avgNfevR = mean(nfevR(successR));
61 else
62     warning('No hubo convergencias en el metodo Riemanniano.');
63     avgIterR = NaN;
64     avgNfevR = NaN;
65 end
66
67 if NG > 0
68     avgIterG = mean(iterG(successG));
69     avgNfevG = mean(nfevG(successG));
70 else
71     warning('No hubo convergencias en el metodo Euclidiano.');
72     avgIterG = NaN;
73     avgNfevG = NaN;
74 end
75
76 pctSuccR = NR/numRuns*100;
77 pctSuccG = NG/numRuns*100;
78
79 %% PresentaciOn de resultados finales
80 fprintf('\n--- Resultados tras %d ejecuciones ---\n\n', numRuns);
81
82 fprintf('Gradiente Riemanniano:\n');
83 fprintf('  Convergencias efectivas   : %4d/%d (%.1f%%)\n', NR, numRuns, pctSuccR);
84 fprintf('  Promedio de iteraciones   : %7.2f (solo convergentes)\n', avgIterR);
85 fprintf('  Promedio de nfev          : %7.2f (solo convergentes)\n\n', avgNfevR);
86 %============================
87 %Gradiente Euclidiano
88 %============================
89 fprintf('Gradiente Euclidiano:\n');
90 fprintf('  Convergencias efectivas   : %4d/%d (%.1f%%)\n', NG, numRuns, pctSuccG);
```

```
fprintf('  Promedio de iteraciones    : %7.2f (solo convergentes)\n', avgIterG);
fprintf('  Promedio de nfev           : %7.2f (solo convergentes)\n', avgNfevG);
```

Listing 11: Código Matlab para la comparación de los métodos gradiente en el espacio Euclidiano y la variedad SPD

8.3. Subgradiente de Greember-Pierskalla para la variedad de matrices simétricas definidas positivas (SPD)

Para esta subsección se hace uso del subgradiente de Greemberg-Pierskalla en la variedad de matrices definidas positivas(SPD), que se define en la subsección 5.0.4 para el caso Euclidiano y en la subsección 5.0.5 en el caso de la variedad SPD.

8.3.1. Caso Euclidiano

8.3.2. main.m

```
    %% Subgradiente GP Riemanniano con matriz generada automaticamente
% --- Entrada de usuario ---
n = input('Ingrese la dimension de la matriz: ');
if ~isscalar(n) || n <= 0 || floor(n) ~= n
    error('La dimension debe ser un entero positivo.');
end

% --- Parametros para autovalores ---
l = 0;         % Limite inferior
u = 10;        % Limite superior

% --- Generacion automatica de matriz PSD ---
X0 = generate_psd_matrix(n, l, u);
fprintf('\nMatriz semidefinida positiva generada aleatoriamente:\n');
disp(X0);

% --- Validaciones ---
if ~isequal(X0, X0.')
    warning('La matriz no es simetrica.');
end
[~, p] = chol(X0);
if p ~= 0
    error('X0 debe ser positiva definida.');
end

% --- Parametros del algoritmo ---
tol      = 1e-6;
max_iter = 1000;

% --- Ejecucion del algoritmo ---
[X_star, log_data] = incremental_GP_subgradient_f(X0, tol, max_iter);

% --- Salida de resultados ---
fprintf('\n%10s | %15s | %15s\n','Iter','f(X)','det(X)');
fprintf('%s\n', repmat('-',1,10+3+15+3+15));
for k = 1:numel(log_data)
    d = log_data(k);
    fprintf('%10d | %15.6f | %15.6f\n', d.Iteracion, d.fX, d.detX);
end

% --- Resultado final ---
fprintf('\nResultado final:\n');
fprintf('  f(X*): %f\n', f_func(X_star));
```

```
Lf = chol(X_star,'lower');
logdet_f = 2*sum(log(diag(Lf)));
fprintf('  det(X*): %f\n', exp(logdet_f));

%% Funcion para generar matriz PSD
function X = generate_psd_matrix(n, l, u)
    % Genera una matriz simetrica semidefinida positiva:
    %  n - dimension
    %  l - minimo autovalor
    %  u - maximo autovalor

    % 1. Autovalores aleatorios en [l, u]
    lambda = l + (u - l) * rand(1, n);

    % 2. Base ortonormal
    Q = orth(randn(n));

    % 3. Construccion
    X = Q * diag(lambda) * Q';

    % 4. Correcciones numericas
    if ~isreal(X)
        X = real(X);
    end
    if ~issymmetric(X)
        X = (X + X') / 2;
    end
end
```

Listing 12: Código Matlab para el cálculo de la gradiente Riemanniana

8.3.3. f_func.m

```
    function val = f_func(X)
    % Calcula f(X) = (ln(det(X)))^2 - ln(det(X))
    L = chol(X, 'lower');
    logdet = 2 * sum(log(diag(L)));
    val = logdet^2 - logdet;
end
```

Listing 13: Código Matlab para definir la función a evaluar

8.3.4. exp_map.m

```
    function Y = exp_map(X, V)
    % exp_X(V) = X^(1/2) * expm(X^(-1/2) * V * X^(-1/2)) * X^(1/2)
    X_sqrt = sqrtm(X);
    X_inv_sqrt = inv(X_sqrt);
    Y = X_sqrt * expm(X_inv_sqrt * V * X_inv_sqrt) * X_sqrt;
end
```

Listing 14: Código Matlab para definir la geodésica en la variedad SPD

8.3.5. GP_subdiff_f.m

```
    function u = GP_subdiff_f(X, tol_sub)
    if nargin < 2
        tol_sub = 1e-4;
    end
    L = chol(X, 'lower');
```

```
    logdet = 2 * sum(log(diag(L)));
    if abs(logdet - 0.5) < tol_sub
        u = zeros(size(X));
    elseif logdet > 0.5
        u = X;
    else
        u = -X;
    end
end
```

Listing 15: Código Matlab para definir el subdiferencial de Greemberg-Pierskalla

8.3.6. metric_norm.m

```
    function norm_val = metric_norm(X, U)
    % ||U||_X = sqrt(tr(U * X^{-1} * U * X^{-1}))
    norm_val = sqrt(trace(U * (inv(X)) * U * (inv(X))));
end
```

Listing 16: Código Matlab para definir la métrica Riemanniana en la variedad SPD

8.3.7. incremental_GP_subgradient_.m

```
    function [X_star, log_data, nfev] = incremental_GP_subgradient_f(X0, tol,
        max_iter)
    % Funcion que implementa el metodo incremental de subgradiente
    % sobre el espacio de matrices simetricas semidefinidas positivas.
    %
    % Entradas:
    %   X0      : matriz inicial (PSD)
    %   tol       : tolerancia para la norma del subgradiente y cambio en f
    %   max_iter  : numero maximo de iteraciones
    %
    % Salidas:
    %   X_star    : matriz resultante tras la iteracion
    %   log_data  : estructura con informacion por iteracion
    %   nfev      : numero de evaluaciones de la funcion objetivo

    if nargin < 2, tol = 1e-6;      end
    if nargin < 3, max_iter = 1000; end

    X = X0;
    log_data = struct('Iteracion', {}, 'fX', {}, 'detX', {}, 'X', {});

    % Inicializar contador de evaluaciones
    nfev = 0;

    % --- Iteracion 0 ---
    L      = chol(X, 'lower');
    logdet = 2 * sum(log(diag(L)));
    detX   = exp(logdet);
    f_prev = f_func(X);
    nfev   = nfev + 1;

    % Guardar datos de iteracion 0
    log_data(1) = struct('Iteracion', 0, ...
                         'fX', f_prev, ...
                         'detX', detX, ...
                         'X', X);

    iter_count = 0;
    while iter_count < max_iter
        iter_count = iter_count + 1;

```

```
        % Paso t_k = 1/(k+1)
        t = 1.0 / (iter_count + 1);

        % Subgradiente de f en X
        u = GP_subdiff_f(X, tol);

        % Condicion de parada por subgradiente pequeno
        if all(abs(u(:)) < tol)
            fprintf('Iteracion %d: optimo alcanzado.\n', iter_count);
            break;
        end

        % Paso de subgradiente
        norm_u  = metric_norm(X, u);
        u_norm  = u / norm_u;
        X_next  = exp_map(X, -t * u_norm);

        % Evaluacion de f y det en nuevo punto
        Ln       = chol(X_next, 'lower');
        logdet_n = 2 * sum(log(diag(Ln)));
        detX_n   = exp(logdet_n);
        f_curr   = f_func(X_next);
        nfev     = nfev + 1;

        % Guardar datos de iteracion
        log_data(end+1) = struct('Iteracion', iter_count, ...
                                 'fX', f_curr, ...
                                 'detX', detX_n, ...
                                 'X', X_next);

        % Criterio de convergencia en funcion
        if abs(f_curr - f_prev) < tol
            X = X_next;
            fprintf('Convergencia en iteracion %d.\n', iter_count);
            break;
        end

        % Actualizar variables para siguiente iteracion
        X      = X_next;
        f_prev = f_curr;
    end

    X_star = X;
end
```

Listing 17: Código Matlab para el método de Subgradiente en la variedad SPD

8.3.8. comp.m

```
    clear all
clc

% ------------------------------
%     Configuracion general
% ------------------------------
numRuns    = 100;     % Numero total de ejecuciones (por ejemplo, 100)
n          = 50;      % Dimension de las matrices
l          = 0;       % Limite inferior de autovalores
u          = 10;       % Limite superior de autovalores
tol        = 1e-4;    % Tolerancia de parada
max_iter   = 1000;      % Max. numero de iteraciones permitidas

% ------------------------------
%   Pre-alocacion de vectores
% ------------------------------
iterR       = nan(numRuns,1);     % Iteraciones registradas (solo para convergentes)
nfevR       = nan(numRuns,1);     % Numero de evaluaciones de f (solo para
    convergentes)
```

```
convergedR  = false(numRuns,1);  % Booleano: true si convergio antes de max_iter
failedR     = false(numRuns,1);  % Booleano: true si hubo error en la ejecucion

% ------------------------------
%       Bucle de ejecuciones
% ------------------------------
for k = 1:numRuns
    try
        % --- Generar matriz PSD aleatoria ---
        X0 = generate_psd_matrix(n, l, u);

        % --- Metodo GP Subgradiente Riemanniano ---
        [~, log_data] = incremental_GP_subgradient_f(X0, tol, max_iter);

        % Extraer la ultima iteracion registrada
        lastIter = log_data(end).Iteracion;

        % Si la ultima iteracion es menor que max_iter, consideramos que convergio
        if lastIter < max_iter
            convergedR(k) = true;
            iterR(k)      = lastIter;
            nfevR(k)      = length(log_data);  % asume que cada fila de log_data
                corresponde a una evaluacion de f
        else
            % Si lastIter == max_iter, asumimos que no cumplio la tolerancia
                dentro de max_iter
            convergedR(k) = false;
            % iterR(k) y nfevR(k) quedan en NaN para no contarlos en el promedio
        end

    catch
        % Si ocurre cualquier error (numerico u otro), marcamos como fallido
        failedR(k) = true;
        % iterR(k) y nfevR(k) permanecen en NaN
    end
end

% ------------------------------
%     Procesamiento de resultados
% ------------------------------
% Numero de ejecuciones totales
totalRuns = numRuns;

% Numero de ejecuciones con error
nFailed = sum(failedR);

% Numero de ejecuciones que no alcanzaron max_iter (es decir, convergentes)
nConverged = sum(convergedR);

% Porcentaje de ejecuciones que convergieron exitosamente (respecto al total)
pctConverged = nConverged / totalRuns * 100;

% Porcentaje de ejecuciones que fallaron por error (opcional)
pctFailed = nFailed / totalRuns * 100;

% Promedio de iteraciones **solo** sobre las ejecuciones que convergieron
if nConverged > 0
    avgIterR = mean(iterR(convergedR));
    avgNfevR = mean(nfevR(convergedR));
else
    avgIterR  = NaN;
    avgNfevR  = NaN;
end

% ------------------------------
%       Imprimir resumen
% ------------------------------
fprintf('\n--- Resumen tras %d ejecuciones totales ---\n\n', totalRuns);
fprintf('Metodo Riemanniano (GP-Subgrad):\n');
fprintf('  Ejecuciones con error        : %3d/%3d (%.2f%%)\n', nFailed, totalRuns,
    pctFailed);
```

```
fprintf('  Ejecuciones convergentes     : %3d/%3d (%.2f%%)\n', nConverged,
    totalRuns, pctConverged);
if nConverged > 0
    fprintf('  Promedio de iteraciones      : %.2f (solo convergentes)\n',
        avgIterR);
    fprintf('  Promedio de nfev             : %.2f (solo convergentes)\n',
        avgNfevR);
else
    fprintf('  No hubo ejecuciones convergentes (Iter < max_iter).\n');
end

% ------------------------------
%      Evaluacion final extra
% ------------------------------
% (Opcional) Mostrar f(X*) y det(X*) para la ultima corrida convergente
lastIdx = find(convergedR, 1, 'last');
if ~isempty(lastIdx)
    % Volver a ejecutar incremental_GP_subgradient_f para obtener X_star
    X0 = generate_psd_matrix(n, l, u);
    [~, log_data] = incremental_GP_subgradient_f(X0, tol, max_iter);
    X_star = log_data(end).X;  % suponiendo que log_data(end).X guarda la matriz
        final
    fprintf('\nResultado final (ultima corrida convergente):\n');
    fprintf('  f(X*): %g\n', f_func(X_star));
    Lf = chol(X_star,'lower');
    logdet_f = 2 * sum(log(diag(Lf)));
    fprintf('  det(X*): %g\n', exp(logdet_f));
end

% ------------------------------
%      Funcion auxiliar
% ------------------------------
function X = generate_psd_matrix(n, l, u)
    % GENERATE_PSD_MATRIX genera una matriz simetrica semidefinida positiva
    %   de dimension nxn, cuyas autovalores estan uniformemente distribuidos
    %   entre l y u.
    lambda = l + (u - l) * rand(1, n);     % vector de autovalores aleatorios en
        [l, u]
    Q      = orth(randn(n));               % matriz ortogonal aleatoria
    X      = Q * diag(lambda) * Q';        % construccion de PSD
    if ~isreal(X)
        X = real(X);
    end
    if ~issymmetric(X)
        X = (X + X') / 2;
    end
end
```

Listing 18: Código Matlab para la comparación del método subgradiente en la variedad SPD y el epacio Euclidiano

8.4. Subgradiente de Greemberg-Pierskalla en el espacio Euclidiano

8.4.1. comp.m

```
    clear; clc;

% ------------------------------
%     Configuracion general
% ------------------------------
numRuns    = 10;      % Numero total de ejecuciones
n          = 5;        % Dimension de las matrices
l          = 0;        % Limite inferior de autovalores
u          = 5;        % Limite superior de autovalores
tol        = 1e-4;     % Tolerancia de parada
max_iter   = 1000;     % Max. numero de iteraciones
```

```
12
13 % ----------------------------
14 %   Pre-alocacion de vectores
15 % ----------------------------
16 iterE    = zeros(numRuns,1);    % Iteraciones Euclideo
17 nfevE    = zeros(numRuns,1);    % Evaluaciones de f
18 successE = false(numRuns,1);    % Convergencia
19
20 global a b;
21
22 % ----------------------------
23 %      Bucle de ejecuciones
24 % ----------------------------
25 for k = 1:numRuns
26     % --- Parametros de la funcion ---
27     a = 1.0;
28     b = 1.0;
29
30     % --- Generar matriz PSD aleatoria ---
31     X0 = generate_psd_matrix(n, l, u);
32
33     try
34         [X_star, log_data, nfev_k] = incremental_Euclid_subgradient_f(X0, tol,
               max_iter);
35         itE_k = log_data(end).iter;
36
37         iterE(k)    = itE_k;
38         nfevE(k)    = nfev_k;
39         successE(k) = itE_k < max_iter;
40
41         fprintf('Corrida %3d: iters = %4d, nfev = %4d\n', k, itE_k, nfev_k);
42
43     catch ME
44         warning('Corrida %d fallo: %s', k, ME.message);
45         iterE(k)    = NaN;
46         nfevE(k)    = NaN;
47         successE(k) = false;
48     end
49 end
50
51 % ----------------------------
52 %     Resultados finales
53 % ----------------------------
54 NE = sum(successE);               % Numero de ejecuciones exitosas
55 pctSuccE = NE / numRuns * 100;    % Porcentaje de exito
56
57 if NE > 0
58     avgIterE = mean(iterE(successE));
59     avgNfevE = mean(nfevE(successE));
60 else
61     warning('No hubo convergencias en el metodo Euclideo.');
62     avgIterE = NaN;
63     avgNfevE = NaN;
64 end
65
66 % ----------------------------
67 %        Imprimir resumen
68 % ----------------------------
69 fprintf('\n--- Resultados tras %d ejecuciones ---\n\n', numRuns);
70 fprintf('Metodo Euclideo (GP-Subgrad):\n');
71 fprintf('  Convergencias efectivas   : %4d/%d (%.1f%%)\n', NE, numRuns, pctSuccE);
72 fprintf('  Promedio de iteraciones   : %7.2f (solo convergentes)\n', avgIterE);
73 fprintf('  Promedio de nfev          : %7.2f (solo convergentes)\n\n', avgNfevE);
74
75 % ----------------------------
76 %     Evaluacion final extra
77 % ----------------------------
78 if successE(end)
79     X_star = log_data(end).X;
80     fprintf('Resultado final (ultima corrida):\n');
81     fprintf('  f(X*): %f\n', f_func(X_star));
82     Lf = chol(X_star,'lower');
```

```
    logdet_f = 2 * sum(log(diag(Lf)));
    fprintf('  det(X*): %f\n', exp(logdet_f));
end

% ----------------------------
%      Funcion auxiliar
% ----------------------------
function X = generate_psd_matrix(n, l, u)
    lambda = l + (u - l) * rand(1, n);
    Q = orth(randn(n));
    X = Q * diag(lambda) * Q';
    if ~isreal(X)
        X = real(X);
    end
    if ~issymmetric(X)
        X = (X + X') / 2;
    end
end
```

Listing 19: Codigo Matlab para el calculo de la gradiente Riemanniana

8.4.2. Euclid_subdiff_f.m

```
    %% Euclid_subdiff_f.m
function u = Euclid_subdiff_f(X, tol)
% Elemento del subdiferencial euclideo segun ln det X mayor 1/2
if nargin < 2, tol = 1e-4; end
L = chol(X, 'lower');
logdet = 2 * sum(log(diag(L)));
if abs(logdet - 0.5) < tol
    u = zeros(size(X));
elseif logdet > 0.5
    u = inv(X);
else
    u = -inv(X);
end
```

Listing 20: Código Matlab para el cálculo del subdiferencial de Greemberg-Pierskalla en el espacio Euclidiano.

8.4.3. f_func.m

```
    %% f_func.m
function val = f_func(X)
% f(X) = (ln det X)^2 - ln det X
L = chol(X, 'lower');
logdet = 2 * sum(log(diag(L)));
val = logdet^2 - logdet;
end
```

Listing 21: Código para definir la función objetivo

8.4.4. generate_psd_matrix.m

```
    function X = generate_psd_matrix(n, l, u)
% Genera una matriz simetrica semidefinida positiva:
%  n - dimension de la matriz
%  l - limite inferior de autovalores
%  u - limite superior de autovalores
```

```
% Ejemplo: X = generate_psd_matrix(5, 0, 20);
    lambda = l + (u - l) * rand(1, n);         % Autovalores aleatorios
    Q = orth(randn(n));                        % Base ortonormal
    X = Q * diag(lambda) * Q';                 % Matriz simetrica PSD
    if ~isreal(X), X = real(X); end
    if ~issymmetric(X), X = (X + X') / 2; end
end
```

Listing 22: Código Matlab para crear una matriz simétrica definida positiva de manera aleatoria.

8.4.5. incremental_Euclid_subgradient_f.m

```
    %% incremental_Euclid_subgradient_f.m
function [X, log_data, nfev] = incremental_Euclid_subgradient_f(X0)
% Metodo de subgradiente incremental en el espacio euclideo
% Devuelve ademas nfev = numero de evaluaciones de f

    % Parametros
    tol      = 1e-5;
    max_iter = 1000;
    X        = X0;

    % Inicializacion del contador de evaluaciones
    nfev = 0;

    % Primera evaluacion de f
    detX   = det(X);
    f_prev = f_func(X);
    nfev   = nfev + 1;

    % Registro inicial (anadimos campo nfev)
    log_data = struct('iter', 0, 'fX', f_prev, 'detX', detX, 'nfev', nfev);

    for k = 1:max_iter
        t = 1/(k + 1);           % tamano de paso
        u = Euclid_subdiff_f(X, tol);

        % Criterio optimo
        if norm(u, 'fro') < tol
            fprintf('optimo alcanzado en iter %d .\n', k);
            break;
        end

        % Paso euclideo normalizado
        U_norm = u / norm(u, 'fro');
        Xn     = (X - t * U_norm + (X - t * U_norm)') / 2;  % simetrizar

        % Verificar SPD
        [~, p] = chol(Xn);
        if p ~= 0
            error('Iter %d: X no SPD tras el paso.', k);
        end

        % Evaluacion de f en Xn y actualizacion de nfev
        detXn   = det(Xn);
        f_curr  = f_func(Xn);
        nfev    = nfev + 1;

        % Registro de esta iteracion
        log_data(end+1) = struct( ...
            'iter', k, ...
            'fX',    f_curr, ...
            'detX',  detXn, ...
            'nfev',  nfev); %#ok<AGROW>

        % Convergencia en f
        if abs(f_curr - f_prev) < tol
```

```
            fprintf('Convergencia en f en iter %d.\n', k);
            X = Xn;
            break;
        end

        % Preparar siguiente iteracion
        X      = Xn;
        f_prev = f_curr;
    end
end
```

Listing 23: Código Matlab para el subgradiente Euclidiano

8.4.6. main.m

```
    clear all
clc

% ================================
% Script principal del metodo de subgradiente euclideo sobre SPD
% ================================

% 1) Parametros de entrada para generacion de matriz
n = input('Ingrese la dimension de la matriz: ');
l = input('Ingrese el limite inferior de los autovalores: ');
u = input('Ingrese el limite superior de los autovalores: ');

if ~isscalar(n) || n <= 0 || floor(n) ~= n
    error('La dimension debe ser un entero positivo.');
end
if l < 0 || u <= l
    error('Se requiere 0 <= l < u para los autovalores.');
end

% 2) Generacion automatica de matriz simetrica semidefinida positiva
X0 = generate_psd_matrix(n, l, u);
fprintf('Matriz X0 generada automaticamente.\n');

% 3) Verificacion de simetria y positividad definida
if norm(X0 - X0','fro') > 1e-12
    warning('X0 no simetrica (corrijo a (X0+X0'')/2).');
    X0 = (X0 + X0')/2;
end
[~,p] = chol(X0);
if p ~= 0
    error('X0 no es positiva definida.');
end

% 4) Ejecucion del metodo de subgradiente euclideo
[X_star, log_data] = incremental_Euclid_subgradient_f(X0);

% 5) Mostrar resultados
fprintf('\n%10s | %15s | %15s\n','Iter','f(X)','det(X)');
fprintf('%s\n', repmat('-',1,10+3+15+3+15));
for k = 1:numel(log_data)
    d = log_data(k);
    fprintf('%10d | %15.6e | %15.6e\n', d.iter, d.fX, d.detX);
end
fprintf('\nIteraciones totales: %d\n', log_data(end).iter);
fprintf('Valor final f(X*) = %.6e\n', log_data(end).fX);
fprintf('det(X*) = %.6e\n', log_data(end).detX);

% 6) Guardar resultado si se desea
% save('X_star.mat', 'X_star');
```

Listing 24: Codigo Matlab para el calculo de la gradiente Riemanniana

8.5. Gradiente y subgradiente en el espacio superior de Poincaré

8.5.1. main.m

```
    %Programa Main
clear,clc;
% Solicitar coordenadas iniciales al usuario
x0 = input('Introduce las coordenadas iniciales [x, y] separadas por coma: ');
format long;
tol=1e-6;% tolerancia a escoger
% 1. Descenso por gradiente con Armijo
[xListGrad, resultsGrad] = riemannianGradientDescentArmijo(x0, 0.1, tol, 1000);
resultsGrad = round(resultsGrad, 8); % <--- AQUi
disp('Metodo de descenso por gradiente (Armijo):');
disp(array2table(resultsGrad, 'VariableNames', {'Iter','x','y','f','gradNorm'}));

% 2. Descenso por gradiente con tamano de paso decreciente
[~, resultsDecr] = riemannianGradientDecreasing(x0, 3.0, 1.0, tol, 1000);
resultsDecr = round(resultsDecr, 8); % <--- AQUi
disp('Descenso por gradiente (tamano decreciente 1/k):');
disp(array2table(resultsDecr, 'VariableNames', {'Iter','x','y','f','gradNorm'}));

% 3. Metodo del subgradiente simple (tamano de paso 1/k)
[~, resultsSubgSimple] = riemannianSubgradientMethod(x0, 'simple', 1000, tol, 1.0);
resultsSubgSimple = round(resultsSubgSimple, 8); % <--- AQUi
disp('Metodo del subgradiente (1/k):');
disp(array2table(resultsSubgSimple, 'VariableNames', {'Iter','x','y','f'}));

% 4. Metodo del subgradiente Polyak
[~, resultsSubgPolyak] = riemannianSubgradientMethod(x0, 'polyak', 1000, tol, 1.0);
resultsSubgPolyak = round(resultsSubgPolyak, 8); % <--- AQUi
disp('Metodo del subgradiente (Polyak):');
disp(array2table(resultsSubgPolyak, 'VariableNames', {'Iter','x','y','f'}));


% ================================================================================
% Graficos comparativos: evolucion de f(x,y) para cada metodo y tamano de
% paso correspondiente
% ================================================================================
% ================================================================================
% Graficos comparativos: evolucion de f(x,y) para cada metodo y tamano de paso
% ================================================================================
figure('Name','Evolucion de f(x,y)','NumberTitle','off',...
       'Position',[100 100 1000 600]);  % [left bottom width height]
hold on;

% Definimos colores contrastantes (puedes elegir otros)
colores = lines(4);

% Gradiente Armijo
plot(resultsGrad(:,1), resultsGrad(:,4), ...
     'LineStyle','-','Marker','o','LineWidth',2,'MarkerSize',8,...
     'Color',colores(1,:), 'DisplayName','Gradiente (Armijo)');

% Gradiente decreciente 1/k
plot(resultsDecr(:,1), resultsDecr(:,4), ...
     'LineStyle','--','Marker','s','LineWidth',2,'MarkerSize',8,...
     'Color',colores(2,:), 'DisplayName','Gradiente (1/k)');

% Subgradiente simple 1/k
plot(resultsSubgSimple(:,1), resultsSubgSimple(:,4), ...
     'LineStyle','-.','Marker','d','LineWidth',2,'MarkerSize',8,...
     'Color',colores(3,:), 'DisplayName','Subgradiente (1/k)');

% Subgradiente Polyak
plot(resultsSubgPolyak(:,1), resultsSubgPolyak(:,4), ...
     'LineStyle',':','Marker','^','LineWidth',2,'MarkerSize',8,...
     'Color',colores(4,:), 'DisplayName','Subgradiente (Polyak)');

% Leyenda y ejes
```

```
%xlabel('Iteracion','FontSize',14,'FontWeight','bold');
%ylabel('f(x,y)','FontSize',14,'FontWeight','bold');
%title('Comparacion de metodos de optimizacion','FontSize',16,'FontWeight','bold');

%set(gca, ...
    % 'FontSize',12, ...
    % 'LineWidth',1.5, ...
    % 'Box','on', ...
    % 'GridLineStyle','--', ...
    % 'XMinorGrid','on', ...
    % 'YMinorGrid','on' ...
%);

%legend('Location','northeast','Box','off','FontSize',12);
%grid on;
%hold off;
```

Listing 25: Código Matlab principal

8.5.2. riemannianGradient.m

```
    function gR = riemannianGradient(X, gE)
    % Gradiente Riemanniano: y^2 * gradE(gradiente euclideano)
    gR = (X(2)^2) * gE;
end
```

Listing 26: Código Matlab para el cálculo de la gradiente Riemanniano en el plano superior de Poincaré

8.5.3. geodesicPoints.m

```
    function pts = geodesicPoints(X, U, numPoints, tau)
    % Aproxima puntos de la geodesica con RK4
    if nargin < 3, numPoints = 30; end
    if nargin < 4, tau = 1.0; end
    h = tau / numPoints;
    state = [X(1), X(2), U(1), U(2)];
    pts   = zeros(numPoints,2);
    pts(1,:) = X;
    for i = 1:numPoints-1
        state     = rk4Step(state, h);
        pts(i+1,:) = state(1:2);
    end
end
```

Listing 27: Código Matlab para la aproximación de las geodésicas en el plano superior de Poincaré

8.5.4. hyperbolicDistance.m

```
    function d = hyperbolicDistance(p, q)
    % Calcula la distancia hiperbolica entre dos puntos p y q en H^2
    % p, q son vectores [x, y] con y > 0

    x1 = p(1); y1 = p(2);
    x2 = q(1); y2 = q(2);

    arg = 1 + ((x2 - x1)^2 + (y2 - y1)^2) / (2 * y1 * y2);
    d = acosh(arg);
```

```
end
```

Listing 28: Código Matlab para el cálculo de la distancia hiperbólica Riemanniana

8.5.5. subgradientApprox

```
function sX = subgradientApprox(X, epsilon)
% Aproxima subgradiente Ferreira-Oliveira con diferencias finitas
if nargin < 2, epsilon = 1e-5; end
f0 = f_val(X);
e1 = [1, 0]; e2 = [0, 1];%tomamos los vectores canonicos en R^2
X1 = expMap(X, epsilon*e1);
X2 = expMap(X, epsilon*e2);
d1 = (f_val(X1) - f0) / epsilon;
d2 = (f_val(X2) - f0) / epsilon;
sX = [d1 * X(2)^2, d2 * X(2)^2];
end
```

Listing 29: Código Matlab para el cálculo subgradiente clásico aproximado

8.5.6. rk4Step

```
function newState = rk4Step(state, h)
% Paso RK4 para la ecuacion geodesica
function ds = dyn(s)
    x = s(1); y = s(2); u = s(3); v = s(4);
    if y < 1e-7, y = 1e-7;
    end
    ds = [u, v, 2*u*v/y, (v^2 - u^2)/y];
end
k1 = dyn(state);
k2 = dyn(state + h*k1/2);
k3 = dyn(state + h*k2/2);
k4 = dyn(state + h*k3);
newState = state + h*(k1 + 2*k2 + 2*k3 + k4)/6;
end
```

Listing 30: Código Matlab para el método de Runge-Kutta

8.5.7. riemannianSubgradientMethod

```
function [xList, results] = riemannianSubgradientMethod(xInit, stepStrategy,
    maxIter, tol, fStar)
% Metodo del subgradiente en el plano superior de Poincare con paso de Polyak

xList    = zeros(maxIter+1, 2);
results  = zeros(maxIter+1, 4);
x        = xInit;
x0       = xInit;                  % Guardar p0
qStar    = [0, 1];                 % Punto en el conjunto solucion

% Calcular d0 = d(p0, q)
d0 = hyperbolicDistance(x0, qStar);
kappa = -1;  % Curvatura del plano superior de Poincare

% Calcular alpha_max segun formula
if kappa ~= 0
    alpha_max = 2 * tanh(sqrt(abs(kappa)) * d0) / (sqrt(abs(kappa)) * d0);
else
    alpha_max = 2;
```

```
    end

    alpha = 0.9 * alpha_max;

    % Inicializar iteracion
    xList(1,:)    = x;
    results(1,:)  = [0, x, f_val(x)];
    s = subgradientApprox(x);

    for k = 1:maxIter
        normS = norm(s) / x(2);
        if normS < tol
            xList   = xList(1:k,:);
            results = results(1:k,:);
            break;
        end

        normE = norm(s);
        v = - (x(2) * s) / normE;

        switch stepStrategy
            case 'simple'
                t = 1 / k;

            case 'polyak'
                fDiff = f_val(x) - fStar;
                if fDiff <= 0
                    t = 1e-4;
                else
                    t = alpha * x(2) * fDiff / normE;
                end

            otherwise
                error('Estrategia de paso no reconocida.');
        end

        U = t * v;
        x = expMap(x, U);
        xList(k+1,:)   = x;
        results(k+1,:) = [k, x, f_val(x)];
        s = subgradientApprox(x);
    end
end
```

Listing 31: Código Matlab para el método de subgradiente clásico Riemanniano con el tamaño de paso exógeno y de Polyak.

8.5.8. riemannianGradientDescentArmijo.m

```
function [xList, results] = riemannianGradientDescentArmijo(x0, armijoParam,
    gradTol, maxIter)
% Descenso por gradiente Riemanniano con busqueda de Armijo
xList  = zeros(maxIter+1,2);
results = zeros(maxIter+1,5);
xList(1,:)   = x0;
results(1,:) = [0, x0, f_val(x0), norm(euclideanGradient(x0))];
x = x0;
for it = 1:maxIter
    euclidGrad = euclideanGradient(x);
    gradR = riemannianGradient(x, euclidGrad);
    gradNorm = norm(gradR);
    if gradNorm < gradTol
        xList   = xList(1:it,:);
        results = results(1:it,:);
        break;
    end
    i = 0;
    while true
```

```
            alpha = 2^(-i);
            step  = -alpha * gradR;
            xNew  = expMap(x, step);
            lhs   = f_val(xNew);
            rhs   = f_val(x) + armijoParam * alpha * (euclideanGradient(x) *
                step');
            if lhs <= rhs
                break;
            else
                i = i + 1;
            end
        end
        x = xNew;
        xList(it+1,:)   = x;
        results(it+1,:) = [it, x, f_val(x), gradNorm];
    end
end
```

Listing 32: Código Matlab para el cálculo de la gradiente Riemanniano con la condición de Armijo.

8.5.9. riemannianGradientDecreasing

```
function [x, results] = riemannianGradientDecreasing(x0, initialStep, beta,
    gradTol, maxIter)
% Descenso por gradiente con tamano de paso decreciente
x       = x0;
results = zeros(maxIter+1,5);
euclid  = euclideanGradient(x);
gradR   = riemannianGradient(x, euclid);
results(1,:) = [0, x, f_val(x), norm(gradR)];
for k = 1:maxIter
    euclid  = euclideanGradient(x);
    gradR   = riemannianGradient(x, euclid);
    gradNorm = norm(gradR);
    if gradNorm < gradTol
        results = results(1:k,:);
        break;
    end
    t = initialStep / (k^beta + 5);%el tamano de paso es 1/(k^beta+5)
    step = -t * gradR;
    x = expMap(x, step);
    results(k+1,:) = [k, x, f_val(x), gradNorm];
end
end
```

Listing 33: Código Matlab para el cálculo de la gradiente Riemanniano con tamaño de paso exógeno.

8.5.10. f_val

```
function val = f_val(X)
% Funcion objetivo f(x,y) = 1 + x^2/y^2
if X(2) <= 0
    error('La coordenada y debe ser > 0.');
end
val = 1 + (X(1)^2) / (X(2)^2);
end
```

Listing 34: Codigo Matlab para el calculo de la gradiente Riemanniana

8.5.11. expMap.m

```
    function Xnew = expMap(X, U)
    % Mapeo exponencial mediante geodesicas aproximadas
    pts  = geodesicPoints(X, U);
    Xnew = pts(end,:);
end
```

Listing 35: Codigo Matlab para el calculo de la gradiente Riemanniana

8.5.12. euclideanGradient

```
    function gE = euclideanGradient(X)
    % Gradiente euclidiano
    if X(2) <= 0
        error('La coordenada y debe ser > 0.');
    end
    gE = [2*X(1)/(X(2)^2), -2*(X(1)^2)/(X(2)^3)];
end
```

Listing 36: Codigo Matlab para el calculo de la gradiente Riemanniana

8.5.13. comp.m

```
    %% ============================================================================
% A) Cuatro metodos Riemannianos (en el plano superior de Poincare, y>0):
%    1. Descenso por gradiente con Armijo
%    2. Descenso por gradiente con paso decreciente (1/k)
%    3. Subgradiente simple (1/k)
%    4. Subgradiente Polyak
% B) Cuatro metodos Euclideos:
%    5. Descenso por gradiente con Armijo
%    6. Descenso por gradiente con paso decreciente (1/k)
%    7. Subgradiente simple (1/k)
%    8. Subgradiente Polyak
%
% ============================================================================

clear; clc;

%% ----------------------------------------------------------------------------
% Parametros generales
tol       = 1e-6;        % Tolerancia de convergencia
maxIter   = 1000;        % Numero maximo de iteraciones permitido
alphaSub  = 1.0;         % Parametro alpha para los metodos de subgradiente
Nexec     = 10;         % Numero de ejecuciones aleatorias

%% ----------------------------------------------------------------------------
% Preasignacion de vectores para almacenar resultados:
%   Puntos iniciales
x0_all = zeros(Nexec,1);
y0_all = zeros(Nexec,1);

%  Riemanniano: iteraciones y convergencia
riem_itersGradArmijo   = zeros(Nexec,1);
riem_convGradArmijo    = false(Nexec,1);

riem_itersGradDecr     = zeros(Nexec,1);
riem_convGradDecr      = false(Nexec,1);

riem_itersSubgSimple   = zeros(Nexec,1);
riem_convSubgSimple    = false(Nexec,1);
```

```
39
40 riem_itersSubgPolyak    = zeros(Nexec,1);
41 riem_convSubgPolyak     = false(Nexec,1);
42
43 %  Euclideo: iteraciones y convergencia
44 euc_itersGradArmijo     = zeros(Nexec,1);
45 euc_convGradArmijo      = false(Nexec,1);
46
47 euc_itersGradDecr       = zeros(Nexec,1);
48 euc_convGradDecr        = false(Nexec,1);
49
50 euc_itersSubgSimple     = zeros(Nexec,1);
51 euc_convSubgSimple      = false(Nexec,1);
52
53 euc_itersSubgPolyak     = zeros(Nexec,1);
54 euc_convSubgPolyak      = false(Nexec,1);
55
56 %% ------------------------------------------------------------------------
57 % Bucle principal: 100 ejecuciones con valores iniciales aleatorios
58 for k = 1:Nexec
59     x0 = -1000 + 2000*rand();    % rand\in[0,1) -> x0\in [-10,10)
60     y0 =   1   +  999*rand();    % rand\in[0,1) -> y0\in [1,10)
61
62     x0_all(k) = x0;
63     y0_all(k) = y0;
64     xInit = [x0, y0];
65
66     %% ------------------------------------------------------------------------
67     % A) Metodos Riemannianos (y > 0)
68     %% ------------------------------------------------------------------------
69     % 1. Gradiente Armijo Riemanniano
70     try
71         [~, resRiemGA] = riemannianGradientDescentArmijo(xInit, 0.1, tol, maxIter);
72         lastIter = resRiemGA(end,1);
73         riem_itersGradArmijo(k) = lastIter;
74         riem_convGradArmijo(k)  = (lastIter < maxIter);
75     catch ME
76         if contains(ME.message, 'coordenada y debe ser > 0')
77             riem_itersGradArmijo(k) = maxIter;
78             riem_convGradArmijo(k)  = false;
79         else
80             rethrow(ME);
81         end
82     end
83
84     % 2. Gradiente (1/k) Riemanniano
85     try
86         [~, resRiemGD] = riemannianGradientDecreasing(xInit, 3.0, 1.0, tol,
               maxIter);
87         lastIter = resRiemGD(end,1);
88         riem_itersGradDecr(k) = lastIter;
89         riem_convGradDecr(k)  = (lastIter < maxIter);
90     catch ME
91         if contains(ME.message, 'coordenada y debe ser > 0')
92             riem_itersGradDecr(k) = maxIter;
93             riem_convGradDecr(k)  = false;
94         else
95             rethrow(ME);
96         end
97     end
98
99     % 3. Subgradiente simple (1/k) Riemanniano
100    try
101        [~, resRiemSS] = riemannianSubgradientMethod(xInit, 'simple', maxIter,
               tol, alphaSub);
102        lastIter = resRiemSS(end,1);
103        riem_itersSubgSimple(k) = lastIter;
104        riem_convSubgSimple(k)  = (lastIter < maxIter);
105    catch ME
106        if contains(ME.message, 'coordenada y debe ser > 0')
107            riem_itersSubgSimple(k) = maxIter;
108            riem_convSubgSimple(k)  = false;
```

```
109         else
110             rethrow(ME);
111         end
112     end
113 
114     % 4. Subgradiente Polyak Riemanniano
115     try
116         [~, resRiemSP] = riemannianSubgradientMethod(xInit, 'polyak', maxIter,
                tol, alphaSub);
117         lastIter = resRiemSP(end,1);
118         riem_itersSubgPolyak(k) = lastIter;
119         riem_convSubgPolyak(k)  = (lastIter < maxIter);
120     catch ME
121         if contains(ME.message, 'coordenada y debe ser > 0')
122             riem_itersSubgPolyak(k) = maxIter;
123             riem_convSubgPolyak(k)  = false;
124         else
125             rethrow(ME);
126         end
127     end
128 
129     %% ---------------------------------------------------------------------
130     % B) Metodos Euclidianos
131     %% ---------------------------------------------------------------------
132     % 5. Gradiente Armijo Euclideo
133     try
134         [~, resEucGA] = gradientDescentArmijoEuclideo(xInit, 0.1, tol, maxIter);
135         lastIter = resEucGA(end,1);
136         euc_itersGradArmijo(k) = lastIter;
137         euc_convGradArmijo(k)  = (lastIter < maxIter);
138     catch ME
139         % Se asume que no hay error de dominio en Euclideo, pero se captura
140         rethrow(ME);
141     end
142 
143     % 6. Gradiente (1/k) Euclideo
144     try
145         [~, resEucGD] = gradientDecreasingEuclideo(xInit, 3.0, 0.5, tol, maxIter);
146         lastIter = resEucGD(end,1);
147         euc_itersGradDecr(k) = lastIter;
148         euc_convGradDecr(k)  = (lastIter < maxIter);
149     catch ME
150         rethrow(ME);
151     end
152 
153     % 7. Subgradiente simple (1/k) Euclideo
154     try
155         [~, resEucSS] = subgradientMethodEuclideo(xInit, 'simple', maxIter, tol,
                alphaSub);
156         lastIter = resEucSS(end,1);
157         euc_itersSubgSimple(k) = lastIter;
158         euc_convSubgSimple(k)  = (lastIter < maxIter);
159     catch ME
160         rethrow(ME);
161     end
162 
163     % 8. Subgradiente Polyak Euclideo
164     try
165         [~, resEucSP] = subgradientMethodEuclideo(xInit, 'polyak', maxIter, tol,
                alphaSub);
166         lastIter = resEucSP(end,1);
167         euc_itersSubgPolyak(k) = lastIter;
168         euc_convSubgPolyak(k)  = (lastIter < maxIter);
169     catch ME
170         rethrow(ME);
171     end
172 end
173 
174 %% ========================================================================
175 % 1) Construccion de la tabla con resultados detallados
176 % ========================================================================
177 Run = (1:Nexec).';
```

```
178
179 T = table( ...
180     Run, ...
181     x0_all, ...
182     y0_all, ...
183     riem_itersGradArmijo, ...
184     riem_itersGradDecr, ...
185     riem_itersSubgSimple, ...
186     riem_itersSubgPolyak, ...
187     euc_itersGradArmijo, ...
188     euc_itersGradDecr, ...
189     euc_itersSubgSimple, ...
190     euc_itersSubgPolyak, ...
191     'VariableNames', { ...
192         'Run', ...
193         'X0', ...
194         'Y0', ...
195         'Riem_Iter_GradArmijo', ...
196         'Riem_Iter_GradDecr_1divk', ...
197         'Riem_Iter_SubgSimple_1divk', ...
198         'Riem_Iter_SubgPolyak', ...
199         'Euc_Iter_GradArmijo', ...
200         'Euc_Iter_GradDecr_1divk', ...
201         'Euc_Iter_SubgSimple_1divk', ...
202         'Euc_Iter_SubgPolyak' ...
203     } ...
204 );
205
206 fprintf('\n=== Tabla de resultados detallados (100 corridas) ===\n');
207 disp(T);
208
209 %% ========================================================================
210 % 2) Calculo de estadisticas resumidas para los 8 metodos
211 % ========================================================================
212 % Promedios de iteraciones (todas las corridas):
213 meanRiem_GA   = mean(riem_itersGradArmijo);
214 meanRiem_GD   = mean(riem_itersGradDecr);
215 meanRiem_SS   = mean(riem_itersSubgSimple);
216 meanRiem_SP   = mean(riem_itersSubgPolyak);
217
218 meanEuc_GA    = mean(euc_itersGradArmijo);
219 meanEuc_GD    = mean(euc_itersGradDecr);
220 meanEuc_SS    = mean(euc_itersSubgSimple);
221 meanEuc_SP    = mean(euc_itersSubgPolyak);
222
223 % Porcentajes de convergencia:
224 pctRiem_GA   = 100 * sum(riem_convGradArmijo)   / Nexec;
225 pctRiem_GD   = 100 * sum(riem_convGradDecr)     / Nexec;
226 pctRiem_SS   = 100 * sum(riem_convSubgSimple)   / Nexec;
227 pctRiem_SP   = 100 * sum(riem_convSubgPolyak)   / Nexec;
228
229 pctEuc_GA    = 100 * sum(euc_convGradArmijo)    / Nexec;
230 pctEuc_GD    = 100 * sum(euc_convGradDecr)      / Nexec;
231 pctEuc_SS    = 100 * sum(euc_convSubgSimple)    / Nexec;
232 pctEuc_SP    = 100 * sum(euc_convSubgPolyak)    / Nexec;
233
234 %% ========================================================================
235 % 3) Impresion de estadisticas resumidas
236 % ========================================================================
237 fprintf('\n==== Estadisticas resumidas tras %d ejecuciones ====\n\n', Nexec);
238
239 fprintf('--- Metodos Riemannianos ---\n');
240 fprintf('1) Gradiente Armijo Riemanniano:\n');
241 fprintf('    Promedio de iteraciones: %.2f\n', meanRiem_GA);
242 fprintf('    %% de convergencia: %.2f%%  (%d de %d)\n\n', ...
243         pctRiem_GA, sum(riem_convGradArmijo), Nexec);
244
245 fprintf('2) Gradiente (1/k) Riemanniano:\n');
246 fprintf('    Promedio de iteraciones: %.2f\n', meanRiem_GD);
247 fprintf('    %% de convergencia: %.2f%%  (%d de %d)\n\n', ...
248         pctRiem_GD, sum(riem_convGradDecr), Nexec);
249
```

```
fprintf('3) Subgradiente simple (1/k) Riemanniano:\n');
fprintf('   Promedio de iteraciones: %.2f\n', meanRiem_SS);
fprintf('   %% de convergencia: %.2f%%  (%d de %d)\n\n', ...
        pctRiem_SS, sum(riem_convSubgSimple), Nexec);

fprintf('4) Subgradiente Polyak Riemanniano:\n');
fprintf('   Promedio de iteraciones: %.2f\n', meanRiem_SP);
fprintf('    %% de convergencia: %.2f%%  (%d de %d)\n\n', ...
        pctRiem_SP, sum(riem_convSubgPolyak), Nexec);

fprintf('--- Metodos Euclideos ---\n');
fprintf('5) Gradiente Armijo Euclideo:\n');
fprintf('    Promedio de iteraciones: %.2f\n', meanEuc_GA);
fprintf('    %% de convergencia: %.2f%%  (%d de %d)\n\n', ...
        pctEuc_GA, sum(euc_convGradArmijo), Nexec);

fprintf('6) Gradiente (1/k) Euclideo:\n');
fprintf('    Promedio de iteraciones: %.2f\n', meanEuc_GD);
fprintf('    %% de convergencia: %.2f%%  (%d de %d)\n\n', ...
        pctEuc_GD, sum(euc_convGradDecr), Nexec);

fprintf('7) Subgradiente simple (1/k) Euclideo:\n');
fprintf('    Promedio de iteraciones: %.2f\n', meanEuc_SS);
fprintf('    %% de convergencia: %.2f%%  (%d de %d)\n\n', ...
        pctEuc_SS, sum(euc_convSubgSimple), Nexec);

fprintf('8) Subgradiente Polyak Euclideo:\n');
fprintf('    Promedio de iteraciones: %.2f\n', meanEuc_SP);
fprintf('    %% de convergencia: %.2f%%  (%d de %d)\n\n', ...
        pctEuc_SP, sum(euc_convSubgPolyak), Nexec);

% ------------------------------
% Fin del script
% ------------------------------
```

Listing 37: Codigo Matlab para el calculo de la gradiente Riemanniana