UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA MATEMÁTICA

IMPLEMENTACIÓN Y EXPERIMENTACIÓN CON MÉTODOS NUMÉRICOS DE OPTIMIZACIÓN DE PRIMER ORDEN EN SISTEMAS INERCIALES CON FACTOR DE AMORTIGUAMIENTO HESSIANO.

TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MATEMÁTICAS APLICADAS

JORGE ANTONIO SEPÚLVEDA ORTEGA

PROFESOR GUÍA:
JUAN PEYPOUQUET URBANEJA

MIEMBROS DE LA COMISIÓN:
ARIS DANIILIDIS
PEDRO GAJARDO ADARO
HÉCTOR RAMÍREZ CABRERA

Este trabajo ha sido parcialmente financiado por:
CMM ANID PIA AFB170001

SANTIAGO DE CHILE
2021

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE MAGÍSTER EN CIENCIAS
DE LA INGENIERÍA,
MENCIÓN MATEMÁTICAS APLICADAS
POR: **JORGE ANTONIO SEPÚLVEDA ORTEGA**
FECHA: 2021
PROF. GUÍA: JUAN PEYPOUQUET URBANEJA

IMPLEMENTACIÓN Y EXPERIMENTACIÓN CON MÉTODOS NUMÉRICOS DE OPTIMIZACIÓN DE PRIMER ORDEN EN SISTEMAS INERCIALES CON FACTOR DE AMORTIGUAMIENTO HESSIANO.

Este informe describe la aplicación y el desempeño de un grupo métodos numéricos de primer orden tomando como problema de ejemplo el proceso de reconstrucción del original de una imagen borrosa. La recuperación de la imagen se propone como un problema de optimización convexa usando una función que modela la diferencia de esta imagen borrosa (generada artificialmente) con la imagen original. Se plantea un problema de optimización convexa tipo *LASSO* Hastie et al. [13] (*Least Absolute Shrinkage and Selection Operator*) que se conforma mediante una función de mínimos cuadrados y una función de regularización de norma ℓ_1. La función objetivo, de este modo, se descompone en una función suave, convexa, diferenciable y una función también convexa pero no diferenciable.

Saliendo del mínimo local...

Agradecimientos

Me gustaría iniciar agradeciendo a mi familia, por la paciencia y comprensión durante todo el periodo de estudios. En especial, a mi esposa por su confianza y constante apoyo, sobre todo, en los tiempos de inquietud y desaliento. No podría haber seguido sino fuera por su respaldo incondicional.
Agradecer también Juan Gabriel Peypouquet mi profesor de tesis por su buena disposición para aceptarme como su alumno memorista, por sus sugerencias para abordar los problemas y por su enorme paciencia, el proyecto ha tomado más tiempo del estimado, pero han sido tiempos excepcionales. Agradecer igualmente, al profesor Alejandro Maass que me aceptó en el programa de Magister, siendo coordinador, sin yo poseer quizás las habilidades necesarias para enfrentar el programa de estudios. De igual modo, me gustaria agradecer también, a los distintos profesores del departamento de Ingeneria Matemática (DIM), que me aceptaron en sus cursos con mis limitaciones de tiempo, y también agradecer a los funcionarios, y a las asistentes, por su invaluable ayuda sobre todo al comienzo, cuando nada entendia como funcionaba el DIM.

Muchas Gracias

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

Este trabajo de tesis se formula como un problema de optimización convexa aplicado a un problema de procesamiento de imágenes. En particular, se presenta la experimentación realizada con un grupo de algoritmos numéricos para mejorar la calidad de una imagen borrosa. La recuperación de la imagen se propone, como un problema de optimización convexa sin restricciones modelado por la suma de dos funciones. Una de estas funciones, en el avance iterativo de los métodos, evalúa la distancia de aproximación a la imagen verdadera, mientras que la otra función, realiza una regularización de las soluciones encontradas en el avance. La combinación de estas dos funciones, establece las condiciones que aseguran la existencia de un conjunto de solución para este tipo de problemas de optimización.

Las etapas de optimización para la recuperación de la imagen se pueden entender, por una parte, como un proceso de minimización de energía, esto es, una minimización de la medida de diferencia cuadrática entre la imagen real y un estimado. Mientras que, por otro lado, la minimización de un término de penalización que promueve dispersión (*sparsity*) en la solución final. La compilación de estas dos etapas resulta en un esquema iterativo, que hace un balance entre el aporte del componente cuadrático y el grado de dispersión generado por el componente de regularización.

El conjunto de solución óptima, se encuentra mediante la ejecución de este grupo de algoritmos, denominados de primer orden, los cuales se basan en las propiedades de un operador de proximidad. Este operador, en particular, utilizado en la recuperación de imagen, funciona adecuadamente en problemas de optimización donde las funciones son no-suaves. Asegurando además, las condiciones de unicidad y consistencia de la solución.

El trabajo, describe los resultados experimentales obtenidos con dos algoritmos de primer orden, que presentan características inerciales y de fricción, basados en iteraciones proximales Attouch et al. [4]. Especificamente con los métodos denominados: *IPAHD-NS* (*Inertial Proximal Algorithm with Hessian Damping Non-Smooth*), *IGAHD* (*Inertial Gradient Algorithm with Hessian Damping.*). Adicionalmente, se experimenta con los algoritmos *ISTA* y *FISTA* (fast iterative shrinkage-thresholding algorithm), este último propuesto por Beck and Teboulle [6], que es una versión proximal del algoritmo de gradiente acelerado de Nesterov [16].

1.1. Métodos de Optimización

En líneas muy generales, optimización es un conjunto de herramientas matemáticas y algorítmicas que posibilita encontrar el óptimo de un problema de minimización (o maximización), condicionado a un conjunto de restricciones impuestas por el entorno del problema. Habitualmente un problema de optimización (P) se expresa en el siguiente formato:

$$\text{(P) Minimizar } f \text{ sobre todos los } x \in \mathcal{D} \text{ que satisfacen las restricciones } x \in \mathcal{C}$$

El dominio del problema $\mathcal{D} \subset \mathbb{R}^n$ corresponde a todos los puntos donde la función objetivo está definida y el subconjunto $\mathcal{C} \subset \mathbb{R}^n$ se denomina el conjunto factible, o conjunto de restricciones de todos los puntos $x \in \mathcal{D}$ que satisfacen la función objetivo y sus restricciones. De igual manera, la variable x es el vector de variables de decisión, y, la función $f : \mathbb{R}^n \to \mathbb{R}$ representa el criterio de minimización que modela el costo de escoger x como variable de decisión. La solución al problema (P) se denomina óptimo o mínimo y corresponde a un punto x^* que verifica la condición

$$f(x^*) \leq f(x) \text{ para todo } x \in \mathcal{D} \text{ tal que } x \in \mathcal{C}$$

Por otra parte, el rol de convexidad en optimización, garantiza que la solución óptima encontrada sea la más adecuada conforme a las restricciones establecidas. Se hace uso de una de las propiedades elementales que señala en condiciones de convexidad cualquier mínimo local es también el mínimo global y además, se establece que las funciones convexas son continuas con buenas propiedades de diferenciación. En este sentido, optimización convexa se puede definir como una combinación de las disciplinas de optimización, análisis convexo y computación numérica, que proporciona una serie de métodos adicionales para abordar problemas más complejos (con miles de variables) que los métodos convencionales formulados en programación lineal y cuadrados mínimos. La importancia de optimización convexa se basa en que muchos de los problemas prácticos en optimización son convexos, o pueden ser transformados en convexos, y podrían ser representados por algunos de los múltiples modelos existentes de optimización convexa (Lasso, regresión logística, etc.). Cabe considerar, por otro lado, que existe un permanente desarrollo de algoritmos de optimización que permite manejar grandes volúmenes de datos y resolver eficientemente varios de los modelos mencionados. Un problema de optimización convexa es definido, cuando el dominio es convexo y las funciones del problema (objetivo y restricciones) cumplen con la siguiente desigualdad

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y), \quad \forall x, y \in \mathbb{R}^n, \quad \theta \in [0,1].$$

La teoría de análisis convexo proporciona, por su parte, los principios que determinan las condiciones necesarias y suficientes de optimalidad, para asegurar que un subconjunto de mínimos exista, y el problema de optimización pueda resolverse. En particular, la información del gradiente y Hessiano (primera y segunda derivada) de la función f (asumiendo diferenciable) juega un rol importante, que permite caracterizar las soluciones óptimas a través de las denominadas condiciones necesarias y suficientes de optimalidad. Una forma de usar estas condiciones, consiste primeramente, en encontrar todos los puntos estacionarios que satisfacen la condición necesaria de primer orden $\nabla f(x) = 0$. Posteriormente, asumiendo que la función f es no convexa, pero dos veces diferenciable, se emplea la condición necesaria de segundo orden, $\nabla^2 f(x) \succeq 0$ para asegurar que los puntos estacionarios son mínimos locales.

En caso de que la función sea convexa, el mínimo local es global y la condición $\nabla f(x) = 0$ es suficiente para optimalidad.

No existe una formula general para resolver analíticamente los problemas de optimización convexa. Es por esto que, se recurre a métodos de optimización para estimar una solución y aproximar numéricamente el valor mínimo de una función. Estos algoritmos en general, son procesos iterativos que parten desde un punto inicial arbitrario y producen una secuencia de estimaciones, que bajo ciertas condiciones, converge al conjunto de valores óptimos.

El grupo de algoritmos estudiados en este trabajo, se basan en la idea de generación de una secuencia descendente, que en cada paso de iteración, reduce el valor de la función que mide el progreso del método en dirección de un óptimo. El esquema iterativo en términos generales, define un operador $G : \mathbb{R}^n \to \mathbb{R}^n$, que determina el siguiente punto por la iteración $x_{k+1} = G(x_k)$, resultando en una secuencia de valores $\{x_k\}_{k=0}^{\infty}$ que al ser evaluados en la función $f : \mathbb{R}^n \to \mathbb{R}$, del modelo de optimización cumple $f(x_{k+1}) \leq f(x_k)$. Se asegura convergencia, en varios de estos métodos, comprobando la existencia de un punto fijo de modo que $x^* = G(x^*)$.

El paso de iteración de descenso en los algoritmos más clásicos se divide en dos partes. Una primera etapa, corresponde al procedimiento de búsqueda de una dirección válida $d_k \in \mathbb{R}^n$, que entrega el rumbo de descenso de la próxima iteración, y su cómputo se basa en la información local del punto en curso. La segunda parte, consiste en una exploración en la línea de dirección (*line search*) que determina el valor del punto x_{k+1} de la siguiente iteración. Esta etapa, estima el valor de un parámetro de paso t_k que pondera el grado de avance en la línea de dirección,

$$t_k = \arg\min_{t>0} f(x_k + t d_k).$$

Encontrar un valor óptimo del parámetro t_k, en cada iteración se formula también como un problema de optimización independiente. Se puede observar que, escogiendo un valor pequeño, podría resultar en un gran número de iteraciones para lograr el destino final, y al contrario, un valor grande la sucesión podría nunca converger. Se proponen varias reglas en la literatura para escoger este parámetro, entre ellas por ejemplo, dejar un valor constante o que disminuya con el avance, también está el procedimiento de *backtracking*. Con el vector de dirección d_k y el parámetro de paso t_k se estima el próximo punto mediante el esquema iterativo siguiente:

$$x_{k+1} := x_k - t_k x_k.$$

Esta iteración se puede detener, ya sea, si cumple condición de optimalidad $d_k = 0$, algún otro criterio, o alcanza una cierta tolerancia $\epsilon > 0$,

$$||d_k|| \leq \epsilon.$$

Los tipos de métodos numéricos, en un problema de optimización convexa sin restricciones, se pueden separar en dos enfoques, dependiendo de la clase de función que modela el problema; el caso diferenciable, y no diferenciable. En el primer caso el método más conocido, es la iteración de gradiente. El cual emplea la información de gradiente como dirección de descenso,

$$x_{k+1} = x_k - t_k \nabla f(x_k).$$

La finalidad de este método es buscar en cada iteración la condición de optimalidad, un punto donde el gradiente desaparece $\nabla f(x) = 0$. El parámetro $t_k > 0$ es un valor que siempre asegura el avance de la iteración hacia el conjunto de soluciones del problema. Asumiendo el gradiente de la función f es Lipschitz continuo de constante L y además $t_k L < 2$, se demuestra que la sucesión de valores $\{x_k\}$ generada por el método converge a un minimizador de f.

Para el caso no-diferenciable, los métodos de primer orden (observar que existen también métodos de orden superior, u orden cero, pero esos quedan fuera del estudio de este trabajo), se puede también dividir en dos aproximaciones, mediante el método de subgradientes, o algoritmos proximales. El método de subgradientes hace uso del concepto de diferencial de una función (generalización del gradiente) y proporciona dirección de descenso en puntos donde la función no es diferenciable. Los algoritmos proximales, por otra parte, convierten la función convexa no diferenciable en una función equivalente convexa y diferenciable con la propiedad de preservar el mínimo de la función original. En términos más simples, el procedimiento de minimización de una función propia convexa semicontinua inferior se convierte en la minimización de la misma función convexa con la inclusión de un término cuadrático, tal como se indica a continuación,

$$x_{k+1} \in \arg\min_{x \in \mathbb{R}^n} \{f(x) + \frac{1}{2\lambda}||x - x_k||^2\}.$$

La combinación del término cuadrático se denomina regularización de Moureau-Yosida, y entre sus propiedades se establece que para cada combinación de valores de λ y $x_k \in \mathbb{R}^n$ se tiene un único minimizador $\bar{x}$. Adicionalmente, la condición de optimalidad para este tipo de algoritmos queda caracterizado por la siguiente inclusión diferencial

$$0 \in \partial f(x_{k+1}) + \frac{x_{k+1} - x_k}{\lambda}.$$

Este algoritmo se denomina también de punto proximal, entre sus propiedades, se determina que para las sucesiones $\{x_k\}$ generadas por este algoritmo, se tiene que la sucesión $f(x_k)$ es decreciente. Y en caso la función f alcance el mínimo, la sucesión de valores x_k es convergente.

1.2. Problemas de procesamiento de imágenes

Los problemas de procesamiento de imágenes en optimización, se modelan con una función objetivo de costo o energía, en el cual, las etapas de optimización se entienden como una minimización de costo de aproximación, del modelo propuesto para la recuperación de la imagen verdadera. También se puede explicar como un proceso minimización de energía.

Entre los temas estudiados en procesamiento de imágenes, se pueden mencionar, el problema de compresión de imágenes para transmisión y almacenamiento, reducción de ruido generado durante la captura, extracción de características, y reconocimiento de objetos para distinguir entre algunos.

Un problema característico en procesamiento de imágenes es el relacionado a la recons-

trucción de imágenes. Donde se intenta obtener una versión mejorada de una imagen original disminuyendo los efectos de degradación producidos por el proceso de formación (difuminado o *blurring*) y registro (introducción de ruido) de la imagen. El efecto de difuminado es más bien, un proceso determinístico y se puede describir por modelos matemáticos con cierta precisión. Por otra parte, el ruido se asocia con un proceso aleatorio donde se asume conocida ciertas propiedades estadísticas del proceso.

El problema puede ser abordado por diferentes enfoques. Una forma directa, es tratando una imagen en el dominio espacial o en el de frecuencias (o cualquier otra transformación), aplicando todas las técnicas utilizadas en procesamiento clásico digital con filtros espaciales de mejora o realce (suavizado, ecualización, etc.) o también de frecuencia. Otra perspectiva, es considerar el procesamiento de imágenes como un problema inverso, estimando la imagen original, a partir de los datos o parámetros obtenidos por las observaciones, con un conjunto de herramientas matemáticas o métodos de optimización.

La recuperación de una imagen en optimización convexa se puede interpretar como un problema de búsqueda de dispersión (*sparsity*) en la matriz del problema. En este caso, el desafío en el problema de optimización, se convierte en tratar de encontrar una combinación apropiada de vectores con la mayor cantidad de ceros, que represente adecuadamente la imagen recuperada del problema. En ese marco, el trabajo presentado por Chen, Donoho, and Saunders [11] de un principio de búsqueda de bases de representación (*basis pursuit*) para lograr una descomposición por *átomos*, plantea una forma alternativa de representar una señal (imagen) más allá de la superposición de señales sinusoidales (*Fourier*). Ellos presentan una colección de formas de ondas con parámetros que denominan *diccionarios*, y donde cada una de estas formas de onda contenidas en el diccionario, constituye un elemento que denominan *átomo* (forma de onda discretizada de largo n). De este modo, se propone descomponer una señal, con lo que denominan una superposición "óptima" de elementos tomados de un diccionario que está sobre representado, y encontrar una representación de la señal mediante optimización convexa.

El principio de descomposición de señales, permite plantear la reconstrucción de una imagen con un sistema de ecuaciones lineales, de la forma $\mathbf{A}x = b$ con la matriz $\mathbf{A} \in \mathbb{R}^{n\times m}$ un *diccionario* redundante (matriz de rango completo) y $x \in \mathbb{R}^n$ vector disperso. Entonces, el problema de descomposición por *átomos* se basa en encontrar una representación dispersa (*sparse*) de b y se puede expresar de manera equivalente de la siguiente forma,

$$\min_x \left\{ \|x\|_0 \right\} \quad \text{s.a} \quad \|b - \mathbf{A}x\|_2^2 \leq \epsilon,$$

donde el factor de ϵ, es una tolerancia entre el nivel de representación de x con el original b.

Este problema entonces, se plantea como una minimización de una medida de dispersión, encontrar la mayor cantidad de ceros en el vector $x \in \mathbb{R}^n$. Sin embargo, el problema en norma ℓ_0 es un problema tipo **NP**-completo (problema difícil en NP) [9], que se convierte en una búsqueda combinatoria de una composición apropiada de componentes. Una forma de resolver esta dificultad es cambiando el término minimización a uno de norma ℓ_1. Esta modificación convierte el problema original en un problema de optimización convexa (no suave), donde las soluciones son una aproximación al problema real y se pueden lograr con diferentes

herramientas de optimización.

Se presenta entonces un modelo de optimización general que combina la minimización de dos elementos, un componente energía y una medida de dispersión representados por las normas ℓ_2 y ℓ_1 respectivamente,

$$\frac{1}{2}\|\mathbf{A}x - b\|_2^2 + \lambda\|x\|_1.$$

La expresión anterior, es entonces la representación general del problema de optimización con norma ℓ_0, y dependiendo del tipo de problema que aborda tiene algunas variaciones; mover, por ejemplo, el operador $\mathbf{A}$ hacia el término de regularización. El parámetro λ, que reemplaza el factor de tolerancia ϵ, se encarga de ponderar el error de la representación y la dispersión. El trabajo de *basis pursuit* [11], emplea norma ℓ_1 como criterio de minimización de coeficientes de los *átomos* en su problema de descomposición. También, el modelo *LASSO* [13], se puede comprender como una reconstrucción de la imagen b usando sólo un pequeño número de *átomos*, y el factor de norma ℓ_1 (que reemplaza ℓ_0) se utiliza como un término de penalización o recorte de los coeficientes menor a cierto rango.

En compresión de señales, la idea es reemplazar la imagen real con una aproximación, se usa por ejemplo un diccionario con señales *wavelets* para hacer un cambio de base de la imagen, y obtener una representación de la imagen en el dominio de *wavelets*, transformación que resulta conveniente porque favorece dispersión en los coeficientes. Se usa la función de recorte para obtener una aproximación comprimida de coeficientes *wavelets* que corresponde una imagen comprimida del original. Entonces, para comprimir una señal $x \in \mathbb{R}^n$ se define un cierto valor ℓ_2 de error ϵ, de manera que cumpla con la siguiente expresión

$$\hat{x} = \underset{x\in\mathbb{R}^n}{\arg\min} \|x\|_1 \quad \text{s.a} \quad \|b - \mathbf{A}x\|_2^2 \leq \epsilon^2.$$

Este problema se puede representar de manera similar al modelo general señalado anteriormente, salvo en este caso, se hace una búsqueda de λ que cumpla con la restricción próximo a la igualdad.

En un contexto similar de compresión de datos, el trabajo Donoho [12] plantea que gran parte de los datos adquiridos (muestrados) de una señal, son descartados luego en el proceso de compresión. En ese sentido, expresa la idea de medir directamente los datos que no serán descartados, proponiendo el protocolo de *Compressed Sensing*, con el objetivo de obtener la información relevante de las imágenes y/o señales, prescindiendo de la parte que será descartada por las pérdidas en compresión. El objetivo, es adquirir directamente la data comprimida, reemplazando las muestras del proceso de compresión por un conjunto de mediciones directas de las señales. El número de mediciones aproxima al número de muestras. Asumiendo la señal original de entrada es dispersa (*sparse*) $\Psi \in \mathbb{R}^m$, en caso de no ser dispersa se hace un cambio de base (*wavelets*, *Fourier*, *DCT* etc) para lograr dispersión. Posteriormente mediante una reducción de dimensionalidad, se genera una matriz, $\Phi \in \mathbb{R}^{n\times m}$ con $n << m$, de mediciones lineales con toda la información importante de la señal. Se genera un vector $y = \Phi\Psi$ que es la combinación de las mediciones lineales y la señal *sparse* de entrada. El proceso de recuperación de la señal se obtiene resolviendo

$$\hat{x} = \underset{\Phi\Psi=y}{\arg\min} \quad \|x\|_1.$$

El modelo ROF [22] de manera similar estimula dispersión, pero en este caso, con la información de gradiente de una imagen. El modelo ROF propone utilizar variación total como término de regularización. El modelo de ROF discreto [10] establece un operador de gradiente en el término de norma ℓ_1.

$$\min_x \quad \lambda\|Dx\|_1 + \frac{1}{2}\|\mathbf{A}x - b\|_2^2.$$

La idea de usar variación total para remover el ruido, se basa en la norma ℓ_1 para inducir dispersión en los gradientes de la imagen, de esta manera favorecer zonas de imagen constante con bordes también dispersos.

1.3. Antecedentes Preliminares

Este trabajo de tesis introduce el estudio de un grupo de métodos numéricos [4], basados en la discretización de una ecuación diferencial ordinaria de segundo orden, aplicados sobre un problema optimización convexa en el dominio del procesamiento de imágenes. Estos métodos se basan en la noción de aceleración en el tiempo, propio de una dinámica de segundo orden, para mejorar la tasa de convergencia de minimización. No obstante se denominan métodos descendentes de primer orden debido al uso del gradiente para generar una secuencia de valores.

En su esquema iterativo más simple, esto métodos, construyen una secuencia descendente de valores valiéndose esencialmente de la información de gradiente. Sin embargo, en problemas de optimización con componentes no diferenciables (no suaves), se recurre a un operador de proximidad para convertir el problema a uno fuertemente convexo. De este modo, el esquema iterativo de primer orden genera la secuencia de valores con una combinación de un operador de proximidad y gradiente (subgradiente). Estos métodos bajo condiciones muy generales aseguran diferentes tasas de convergencia de acuerdo a las restricciones y presentación del problema de optimización.

Los métodos de primer orden experimentados en este trabajo se apoyan fuertemente en el operador proximidad. Este operador en conjunción con el operador de gradiente (o subgradiente) evidencian una mejora en las tasas de convergencia, y constituyen una importante herramienta para problemas de optimización con gran volumen de datos. Los operadores proximales operan adecuadamente en condiciones de funciones convexa y no-suaves (diferenciables). Asimismo, existe un grupo de operadores proximales que se pueden obtener de manera explícita que facilita la aproximación a funciones más complejas, resultando en elaboraciones más simples, y fácil de implementar en algoritmos de optimización. Igualmente, son muy favorables para problemas de optimización distribuida lo que permite una mejora en términos de complejidad computacional, en general son métodos con procesos iterativos de bajo costo y uso en recursos memoria.

El desarrollo del trabajo se apoya en optimización convexa para la recuperación de una

imagen borrosa. Una imagen de dos dimensiones se convierte en un vector de columnas del tamaño de la imagen. El problema de optimización general se define como la estimación de una imagen $x \in \mathbb{R}^n$ desconocida cumpliendo la siguiente relación.

$$\mathbf{A}x = b + w$$

Donde $\mathbf{A} \in \mathbb{R}^{n \times n}$ es un operador de difuminación que produce borrosidad en la imagen de verificación, $b \in \mathbb{R}^n$ es la imagen borrosa generada artificialmente, $x \in \mathbb{R}^n$ es el vector estimado que representa la imagen recuperada, y $w \in \mathbb{R}^n$ un vector de ruido. Se propone entonces recuperar la imagen x desde una medición ruidosa b, como un problema de minimización de cuadrados mínimos regularizado.

$$\min_{x \in \mathbb{R}^n} \Big\{ \frac{1}{2} \|\mathbf{A}x - b\|_2^2 + \lambda \|x\|_1 \Big\}.$$

El término $\frac{1}{2}\|\mathbf{A}x - b\|_2^2$ de mínimos cuadrados es una medida de distancia entre b y Ax en norma ℓ_2. El elemento $\|x\|_1$ es un regularizador convexo usado para estabilizar la solución. Este término además se usa para reemplazar el problema original, que podría estar mal condicionado, por uno mejor condicionado que aproxima de buena manera a la solución del problema original. El factor de regularización λ proporciona el grado de compromiso entre sensitividad del ruido y fidelidad de las mediciones.

1.4. Organización del documento

Este trabajo de tesis ha sido estructurado en seis capítulos. El Capítulo 2 hace una revisión de los métodos tradicionales de primer orden. Se detalla el método de gradiente de descenso y sus derivados con aceleración. Se hace la conexión entre el método de gradiente acelerado de Nesterov y una ecuación diferencial ordinaria. El Capítulo 3 describe los algoritmos que abordan problemas convexos modelados por funciones no diferenciables, se detalla la función regularizada de Moreau y la conexión con el operador proximal. Se exponen los algoritmos de punto y gradiente proximal. El Capítulo 4, explica el algoritmo FISTA que hace uso del operador proximal, y es una versión proximal acelerada del método de gradiente. El Capítulo 5 describe los métodos basados en una ecuación de segundo orden con coeficientes de amortiguamiento de viscosidad y Hessiano. El Capítulo 6, hace una descripción del problema y la forma de generar la imagen borrosa de verificación de los métodos de primer orden. Expone los resultados experimentales obtenidos y el Capítulo 7 son las principales conclusiones de este trabajo.

Capítulo 2

Métodos tradicionales de primer orden en optimización convexa

Para elaborar el estudio de los métodos de primer orden basados en discretización de modelos dinámicos inerciales, con factores de amortiguamiento y fricción, se requiere iniciar con una descripción de los métodos clásicos utilizados en optimización convexa, en particular, los métodos de gradiente y sus derivados con aceleración. Estos son métodos numéricos iterativos, los cuales se apoyan principalmente en la información de gradiente, para generar una sucesión de valores que bajo ciertas condiciones garantiza la convergencia a un mínimo, local o global, dependiendo del tipo de condiciones del problema.

En líneas generales, los algoritmos de descenso son procesos iterativos que se determinan por una dirección de descenso, y un parámetro de paso que define el grado de avance de cada iteración. Un paso de iteración en particular, identifica por un lado, el vector de dirección $d_k \in \mathbb{R}^n$ que señala el camino de descenso de $f(x_k)$ y por otro, fija mediante alguna regla el paso de avance $t_k > 0$ de la iteración. El algoritmo se detiene cuando el vector de dirección se hace cero ($d_k = 0$) o alcanza algún criterio de parada (tolerancia de optimalidad preestablecida) $\|d_k\| < \varepsilon$.

$$x_{k+1} = x_k + t_k d_k. \tag{2.1}$$

Para los propósitos de este trabajo, se asume un tipo de función compuesta que permite construir un modelo de optimización más general. Una función global conformada por dos tipos de funciones, una función suave (diferenciable) convexa y una función convexa pero no diferenciable.

$$\min_{x \in \mathbb{R}^n} \Big\{ f(x) + h(x) \Big\} \tag{2.2}$$

- $f : \mathbb{R}^n \to \mathbb{R}$ es una función suave, convexa.
- $h : \mathbb{R}^n \to \mathbb{R}$ es una función, convexa, propia y no diferenciable.

Además la función f tiene gradiente *Lipschitz* continuo dado para alguna constante $L > 0$.

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \forall x, y \in \mathbb{R}^n$$

Se puede observar que dejando en cero la función $h(x) \equiv 0$, el modelo se convierte en un problema de minimización convexa sin restricciones.

$$\min_{x \in \mathbb{R}^n} f(x). \tag{2.3}$$

2.1. Método del gradiente o de máximo descenso

El método de gradiente forma parte de un grupo de algoritmos numéricos que se basan en un principio denominado iteración de descenso (*relaxation sequence*, Nesterov [15]). La idea de este tipo de métodos es generar una secuencia de valores $\{x_k\}$, de modo que, la función objetivo disminuya en cada avance de una iteración $f(x_{k+1}) < f(x_k)$, hasta alcanzar eventualmente un mínimo. Las propiedades de convexidad de la función objetivo f, y el dominio del problema $\mathbb{R}^n$, asegura las condiciones necesarias para la convergencia de la sucesión de valores $\{x_k\}$ generadas por el algoritmo de gradiente descendente.

En general, los algoritmos de descenso escogen una dirección $d_k \in \mathbb{R}^n$ que satisface la regla $\langle \nabla f(x), d_k \rangle < 0$. Esta regla se puede interpretar geométricamente, como la condición donde el vector d_k forma un ángulo de menos de 90 grados con la dirección de máximo descenso. En el caso del método de gradiente, la dirección corresponde al vector de gradiente negativo $d_k = -\nabla f(x_k)$. Por consiguiente, el vector de gradiente proporciona la dirección de máximo descenso o mayor tasa de descenso en la zona de evaluación local [15, p.17]. No obstante, dependiendo del tipo de problema y función, existen otras opciones para el vector de dirección, por ejemplo, subgradientes (cuando la función es no diferenciable) o métodos que utilizan información de segundo orden como el método de Newton que hace uso del Hessiano $(\nabla^2 f(x_k))^{-1}$ y en cada iteración minimiza una aproximación cuadrática de la función.

El método de gradiente de máximo descenso genera una secuencia de puntos $\{x_k\} \in \mathbb{R}^n$ mediante la siguiente expresión de recurrencia

$$x_{k+1} = x_k - t_k \nabla f(x_k),\ k = 0, 1, 2, ... \tag{2.4}$$

El esquema iterativo se inicia en algún punto $x_0 \in \mathbb{R}^n$ arbitrario y un nuevo punto se determina siguiendo la dirección negativa del gradiente, limitado por el valor del factor de paso t_k. Las repeticiones acaban cuando el gradiente es cero $\nabla f(x_k) = 0$. Pero, en términos prácticos, se detienen cuando la norma del gradiente esta dentro de algún rango de parada preestablecido ε, de manera que cumpla $\|\nabla f(x_k)\| \leq \varepsilon$, o también la distancia entre dos puntos sea $\|x_{k+1} - x_k\| \leq \varepsilon$.

2.1.1. Aproximación cuadrática

En un problema de optimización convexa sin restricciones, el método de gradiente puede ser representado por la optimización de una aproximación cuadrática de la función objetivo.

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^n} \Big\{ f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{L}{2} \|x - x_k\|^2 \Big\}. \tag{2.5}$$

Este modelo de optimización se puede derivar a partir de una aproximación de Taylor de segundo orden. Suponiendo, en un inicio, que $f \in C^2$ es parte del conjunto de funciones con-

tinuas dos veces diferenciable. El esquema de funcionamiento de una iteración en particular, se representa con la siguiente expansión de Taylor

$$f(y) \approx f(x) + \nabla f(x)^T(y-x) + \frac{1}{2}(y-x)^T\nabla^2 f(x)(y-x),\ \forall\ x, y \in \mathbb{R}^n.$$

Asumiendo que la función f es convexa, diferenciable con gradiente Lipschitz-continuo de constante L. La condición Lipschitz significa que $\nabla^2 f(x) \preccurlyeq LI$, el mayor valor propio del Hessiano no es más grande que la constante L. Utilizando este límite superior, el término de segundo orden de esta expansión puede ser reemplazado por $L\|y-x\|^2$

$$f(y) \leq f(x) + \nabla f(x)^T(y-x) + \frac{L}{2}\|y-x\|^2 \text{ para todo } x, y \in \mathbb{R}^n. \tag{2.6}$$

Esta desigualdad se define como el Lema de máximo descenso ([15, Teorema 2.1.5], [7, Proposición 6.1.2], [5, Lemma 5.7]), ver demostración en A.2. El cual afirma, que para una función convexa f, evaluada en un punto (local) $x \in \mathbb{R}^n$, queda delimitada superiormente por una función de aproximación cuadrática. La Figura 2.1 muestra un ejemplo del proceso de aproximación para una iteración de gradiente. El punto x_{k+1} de la iteración es el punto (color rojo) que minimiza esta aproximación cuadrática.

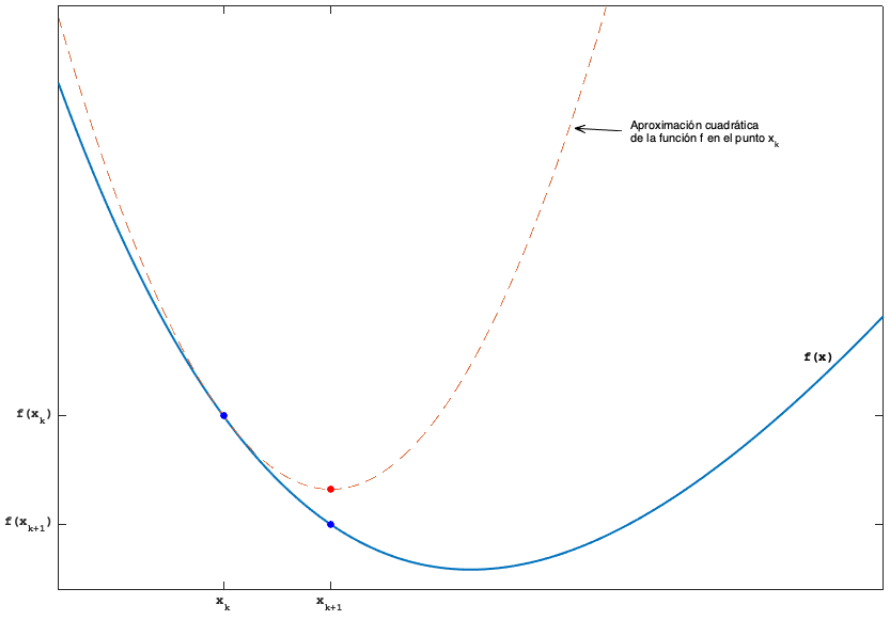


Figura 2.1: Ejemplo de aproximación cuadrática del método de gradiente. El paso de iteración, se puede interpretar, como un desplazamiento de la función cuadrática (linea segmentada roja) hasta que la pendiente de esta función coincida con la pendiente de la función convexa en el punto x_k, el siguiente punto x_{k+1} es el mínimo de esta función cuadrática.

Reemplazando un paso de iteración de gradiente $y = x - t\nabla f(x)$ (2.4), en el lado derecho de la desigualdad del Lema de máximo descenso (2.6) se tiene lo siguiente

$$\begin{aligned} f(y) &\leq f(x) - t\|\nabla f(x)\|^2 + \frac{t^2}{2}L\|\nabla f(x)\|^2 \\ &= f(x) - t(1 - \frac{t}{2}L)\|\nabla f(x)\|^2. \end{aligned} \tag{2.7}$$

Usando $t \leq \frac{1}{L}$ se obtiene

$$f(y) \leq f(x) - \frac{1}{2L}\|\nabla f(x)\|^2. \tag{2.8}$$

Esta última desigualdad muestra que el valor de la función objetivo disminuye regularmente, en cada iteración del gradiente, hasta que alcanza el valor óptimo $f(x) = f(x^*)$. Esto debido, principalmente, que el término $\|\nabla f(x)\|^2$ es siempre positivo o cero cuando $\nabla f(x^*) = 0$. Se puede observar, que la convergencia de la sucesión $\{x_k\}$ se consigue con un valor de paso $t \leq \frac{1}{L}$, para valores superiores a $\frac{1}{L}$ el método de gradiente podría no converger.

2.1.2. Elección del tamaño de paso

Una adecuada elección del tamaño del paso t_k asegura que la evaluación de la función $f(x_k)$ sea decreciente en cada nueva iteración. La selección del paso, es un fino ajuste entre un valor que no reduzca o aumente demasiado la dirección de descenso. Para un valor muy reducido del paso, el algoritmo podría nunca alcanzar el mínimo, o al contrario, con un valor muy grande la función podría incrementar (no decrecer) y nunca lograr la convergencia. Se proponen variadas reglas para la elección del paso (*step size*), algunas de las cuales se describen a continuación.

i) Elegir una sucesión de valores de paso $\{t_k\}_{k=0}^{\infty}$ con antelación. Donde, los valores podrían ser una secuencia constante $t_k = t,\ t > 0$ (opción utilizada en este trabajo), o una sucesión monótona estrictamente decreciente dada por $t_k = \frac{t}{(k+1)^n}$, donde $n \in \{\frac{1}{2}, 1, 2, ...\}$.

ii) Buscar t_k como problema de optimización independiente. Esto es, hacer una búsqueda sobre una recta, seleccionando un t_k que minimice f en la recta dada por $x_k + t_k\nabla f(x_k)$.

$$t_k = \arg\min_{t>0} f(x_k - t\nabla f(x_k)).$$

iii) Otra opción es la regla de Armijo o *backtracking*. Este procedimiento comienza con el valor de paso $t = 1$ y un parámetro $0 < \beta < 1$, luego por cada iteración que verifica la desigualdad

$$f(x - t\nabla f(x)) > f(x) - \frac{t}{2}\|\nabla f(x)\|^2$$

se actualiza $t = \beta t$.

La búsqueda se inicia con un tamaño de paso unitario y que se va reduciendo por el factor β, hasta lograr la condición de parada $f(x - t\nabla f(x)) \leq f(x) - \frac{t}{2}\|\nabla f(x)\|^2$.

2.1.3. Tasa de convergencia

La tasa de convergencia proporciona los márgenes teóricos de complejidad donde los algoritmos se desempeñan. Estos bordes se modifican de acuerdo a la estructura del problema donde son aplicados. El peor desempeño (límite superior) queda delimitado por la condición

Lipschitz del gradiente, mientras que el mejor desempeño se obtiene, suponiendo, el algoritmo evalúa una función fuertemente convexa con parámetro $\mu > 0$. Ambos limites, pueden ser determinados mediante una aproximación cuadrática de la función objetivo f, empleando las constantes L y μ como parámetros del factor cuadrático (como se observa en las desigualdades (2.9), (2.10) de abajo). El desempeño de la tasa de convergencia para una función convexa, entonces, se caracteriza por situarse entre ambas condiciones de aproximación cuadrática,

$$f(y) \leq f(x) + \nabla f(x)^T(y-x) + \frac{L}{2}\|y-x\|_2^2 \tag{2.9}$$

$$f(y) \geq f(x) + \nabla f(x)^T(y-x) + \frac{\mu}{2}\|y-x\|_2^2. \tag{2.10}$$

De manera equivalente usando el Hessiano para todo x en $\mathbb{R}^n$,

$$\mu I \preccurlyeq \nabla^2 f(x) \preccurlyeq LI. \tag{2.11}$$

En este sentido, para una función f convexa, suave con gradiente *Lipschitz* continuo de constante L, se demuestra que la tasa de convergencia del método de gradiente de máximo descenso es lineal o logarítmica [15], [8].

La demostración se inicia con la desigualdad (2.7), que asegura decrecimiento de la función en cada iteración

$$\begin{aligned} f(y) &\leq f(x) - (1 - \frac{1}{2}tL)t\|\nabla f(x)\|^2 \\ &= f(x) - \frac{(2-tL)t}{2}\|\nabla f(x)\|^2 \end{aligned}$$

usando α como variable auxiliar para el término $(2-tL)t$ y reemplazando se obtiene,

$$f(y) \leq f(x) - \frac{\alpha}{2}\|\nabla f(x)\|^2. \tag{2.12}$$

Interesa incluir $f(x_*)$ en esta desigualdad (2.12), para esto, se puede acotar el valor de $f(x)$ en torno al valor óptimo de la función $f(x_*)$, usando la propiedad de convexidad de la función, se tiene la siguiente aproximación lineal en torno al valor óptimo

$$\begin{aligned} f(x_*) &\geq f(x) + \nabla f(x)^T(x_* - x) \\ f(x) &\leq f(x_*) + \nabla f(x)^T(x - x_*) \end{aligned}$$

reemplazando en (2.12)

$$\begin{aligned} f(y) &\leq f(x_*) + \nabla f(x)^T(x - x_*) - \frac{\alpha}{2}\|\nabla f(x)\|^2 \\ f(y) - f(x_*) &\leq \frac{1}{2\alpha}\Big(2\alpha\nabla f(x)^T(x-x_*) - \alpha^2\|\nabla f(x)\|^2\Big) \\ &= \frac{1}{2\alpha}\Big(2\alpha\nabla f(x)^T(x-x_*) - \alpha^2\|\nabla f(x)\|^2 + \|x-x_*\|^2 - \|x-x_*\|^2\Big) \\ &= \frac{1}{2\alpha}\Big(\|x-x_*\|^2 - \|x-\alpha\nabla f(x) - x_*\|^2\Big). \end{aligned}$$

Se puede observar que $y = x - \alpha\nabla f(x)$ es la definición de un paso de gradiente

$$f(y) - f(x_*) \leq \frac{1}{2\alpha}\Big(\|x - x_*\|^2 - \|y - x_*\|^2\Big)$$

esta última desigualdad ocurre para cada paso de iteración, sumando todos los pasos

$$\begin{aligned}\sum_{i=1}^{k}(f(x_i) - f(x_*)) &\leq \sum_{i=1}^{k}\frac{1}{2\alpha}\Big(\|x_{i-1} - x_*\|^2 - \|x_i - x_*\|^2\Big)\\ &\leq \frac{1}{2\alpha}\Big(\|x_0 - x_*\|^2 - \|x_k - x_*\|^2\Big).\end{aligned}$$

La sumatoria del lado derecho se cancela haciendo uso de la suma telescópica. Eliminando el término positivo $\|x_k - x_*\|^2$ permite dejar la diferencia $\|x_0 - x_*\|^2$ como límite superior de la desigualdad.

$$\sum_{i=1}^{k}(f(x_i) - f(x_*)) \leq \frac{1}{2\alpha}\|x_0 - x_*\|^2$$

sabiendo que la evaluación de la función $f(x_k)$ disminuye en cada iteración

$$f(x_k) - f(x_*) \leq \sum_{i=1}^{k}(f(x_i) - f(x_*)) \leq \frac{1}{2\alpha}\|x_0 - x_*\|^2$$

obtenemos la tasa de convergencia del método de gradiente

$$f(x_k) - f(x_*) \leq \frac{\|x_0 - x_*\|^2}{2t(2 - tL)k} \quad \square \tag{2.13}$$

Se verifica entonces que para una función convexa, bajo la condición de Lipschitz continuo el método de gradiente presenta una tasa de convergencia lineal del $\mathcal{O}(\frac{1}{k})$, donde k es el número de iteraciones. Usando un valor de paso $t \leq \frac{1}{L}$ se obtiene una estimación

$$f(x_k) - f(x_*) \leq \frac{L}{2k}\|x_0 - x_*\|^2,\ \forall\ x_k \in \mathbb{R}^n. \tag{2.14}$$

Si se quiere asegurar que $f(x_k) - f(x_*) < \varepsilon$ se requiere un número de iteraciones k sea del orden de $\mathcal{O}(\frac{1}{\varepsilon})$

Una función fuertemente convexa está acotada inferiormente por una constante $\mu > 0$ como se indica en la desigualdad (2.10). Entonces, para el caso de una función fuertemente convexa, gradiente Lipschitz continua, se verifica que la tasa de convergencia es del orden $\mathcal{O}(c^k)$ con $0 < c < 1$ [15, p.70], [8, p.467]. Con un tamaño de paso $t = \frac{2}{\mu+L}$ [19], se evidencia que $c = \frac{\kappa-1}{\kappa+1}$ es mínimo, donde $\kappa = \frac{L}{\mu}$ se denomina número de condición. El método de gradiente de descenso presenta la siguiente tasa de convergencia

$$\begin{aligned}f(x_k) - f(x_*) &\leq \frac{L}{2}\Big(\frac{\kappa - 1}{\kappa + 1}\Big)^{2k}\|x_0 - x_*\|^2\\ \|x_k - x_*\| &\leq \Big(\frac{\kappa - 1}{\kappa + 1}\Big)^{k}\|x_0 - x_*\|.\end{aligned} \tag{2.15}$$

El número de iteraciones para $f(x_k) - f(x_*) \leq \varepsilon$ es del orden $\mathcal{O}(\log(\frac{1}{\varepsilon}))$

$$\frac{\log\left(\frac{f(x_0)-f(x_*)}{\varepsilon}\right)}{\log\left(\frac{2}{L}\left(\frac{\kappa+1}{\kappa-1}\right)\right)}.$$

El número de condición κ presenta influencia en la tasa de convergencia. Un valor de $\kappa > 1$ muestra que $f(x_k)$ converge a $f(x_*)$ cuando $k \to \infty$, pero un valor muy grande de este número incrementa enormemente el número de iteraciones lo que podría provocar nunca lograr una convergencia.

2.2. Método de subgradientes

El subgradiente de una función convexa $f : \mathbb{R}^n \to \mathbb{R}$, evaluado en $\bar{x} \in \mathbb{R}^n$, es algún vector $v \in \mathbb{R}^n$ que proporciona un límite inferior similar a un plano tangente a f en $\bar{x}$, y se representa por la desigualdad siguiente

$$f(x) \geq f(\bar{x}) + \langle v, x - \bar{x} \rangle, \text{ para todos los } x \in \mathbb{R}^n.$$

Por su parte, el subdiferencial $\partial f(\bar{x})$ de la función f en $\bar{x}$ se define como la colección de todos los subgradientes de f en $\bar{x}$. La idea del subdiferencial, es agrupar las pendientes de las funciones afines continuas que tocan el punto $(\bar{x}, f(\bar{x}))$, y están por debajo de la función convexa.

Existe además, una relación entre el subdiferencial $\partial f(\bar{x})$ y las derivadas direccionales de la función f en un punto $\bar{x}$. La derivada direccional de una función convexa, propia, no necesariamente diferenciable, puede ser representada mediante la colección vectores de subgradientes que sustentan los hiperplanos al epígrafo de la función. En este sentido, la derivada direccional de la función f en el punto $\bar{x}$ se puede obtener a través del subdiferencial $\partial f(\bar{x})$ [21, Teorema 23.2]

$$f'(\bar{x}; d) = \max\{\langle v, d \rangle | v \in \partial f(\bar{x})\} \text{ para todo } d \in \mathbb{R}^n,$$

donde $f'(\bar{x}; d)$ es la derivada direccional de la función f evaluada en $\bar{x}$ en la dirección $d \in \mathbb{R}^n$.

La Figura 2.2 muestra un ejemplo de dos subgradientes en el punto $\bar{x}$ que constituyen parte del subdiferencial $\partial f(\bar{x})$. La interpretación geométrica de la desigualdad de subgradiente, tiene sentido cuando la función f es finita en $\bar{x}$, esto es, la representación gráfica de la función afín $h(x) = f(\bar{x}) + \langle v, \bar{x} - x \rangle$ no es un hiperplano vertical al epígrafo de la función en el punto $(\bar{x}, f(\bar{x}))$ [21].

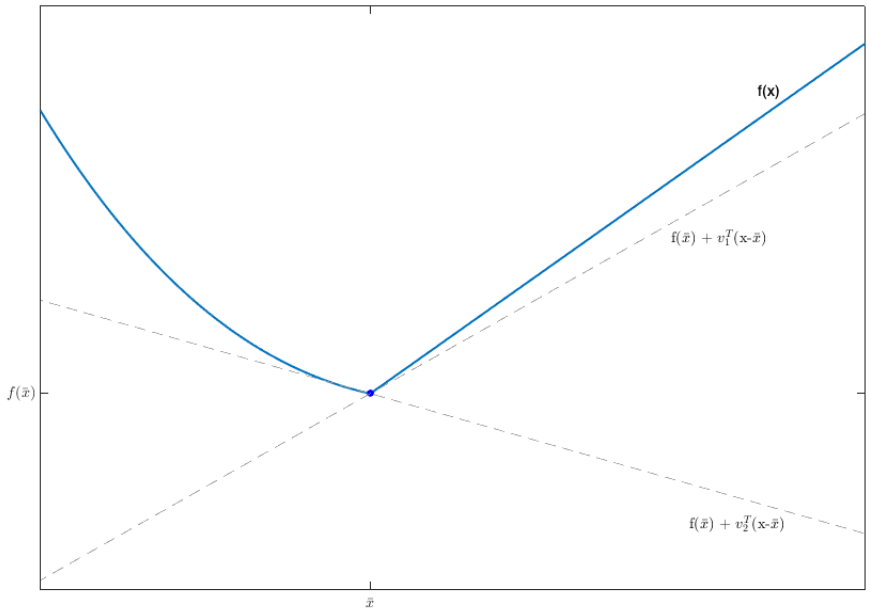


Figura 2.2: El *subgradiente* se puede interpretar geométricamente como la pendiente de la recta (hiperplano) que soporta el epígrafo de la función en un punto $(\bar{x}, f(\bar{x}))$, y aproxima inferiormente a la función f.

El método de subgradiente, es una generalización del método de gradiente, que reemplaza la dirección de gradiente del paso de iteración por un vector de dirección $d_k \in \partial f(x_k)$. De este modo, la búsqueda de una dirección de máximo descenso d_k para una función f, convexa, Lipschitz continua en el punto $\bar{x}$ se deriva de la siguiente regla de minimización.

$$\frac{d_k}{\|d_k\|} \in \arg\min_{\|d\|\leq 1} f'(x_k; d).$$

Se puede observar que si la función f es convexa y diferenciable en el punto $x_k \in \mathbb{R}^n$, entonces el subdiferencial queda definido por $\partial f(x_k) = \{\nabla f(x_k)\}$. La dirección de descenso, en este caso, corresponde al vector de gradiente $d_k = -\nabla f(x_k)$. De esta forma, el algoritmo escoge la dirección d_k que minimiza la derivada direccional de f en el punto $\bar{x}$ (no diferenciable) para generar una secuencia $\{x_k\}$ mediante un esquema iterativo

$$x_{k+1} = x_k - t_k d_k, \quad d_k \in \partial f(x_k).$$

Se advierte que la sucesión $\{f(x_k)\}$ no es necesariamente monótona decreciente, debido que la dirección calculada en cada iteración podría, eventualmente, no ser una dirección de descenso. Para estimar convergencia, se define entonces una nueva sucesión de valores donde cada elemento corresponde a

$$f_k = \min\{f(x_1), f(x_2), ..., f(x_k)\}$$

y ayuda a aproximar al valor óptimo.

El análisis de convergencia del algoritmo de subgradiente, asume, por una parte que la función f es Lipschitz-continua, y por otra asegura existe un conjunto de soluciones óptimas $\mathcal{S} \subset \mathbb{R}^n$.

La estimación de convergencia de la sucesión $\{x_k\}$ obtenida mediante el método de subgradientes, se apoya en el supuesto que el subgradiente está acotado por la constante Lipschitz ℓ [7, p.153]

$$\|v\| \leq \ell,\ v \in \partial f(x_k)$$

y, por una proposición [7, 3.2.2] que expone la reducción de distancia de las iteraciones

$$\|x_{k+1} - x\|^2 \leq \|x_k - x\|^2 - 2\alpha_k(f(x_k) - f(x))^2 + \alpha_k^2 \ell^2, \quad \text{para todo } x \in \mathbb{R}^n.$$

El límite de convergencia por consiguiente, con un tamaño de paso t_k (variable), se expresa de la siguiente manera

$$\min_{k=1,..,n}\{f(x_k)\} - \min(f) \leq \frac{d(x_0;\mathcal{S})^2 + \ell^2 \sum_{i=0}^{k} t_i^2}{2\sum_{i=0}^{k} t_i}. \tag{2.16}$$

Para el caso del valor de paso constante, $t_k = t$

$$\begin{aligned} 0 \leq \min_{k=1,..,n}\{f(x_k)\} - \min(f) &\leq \frac{d(x_0;\mathcal{S})^2 + k\ell^2 t^2}{2kt} \\ &= \frac{d(x_0;\mathcal{S})^2}{2kt} + \frac{\ell^2 t}{2}. \end{aligned} \tag{2.17}$$

Para lograr que $\min_{k=1,..,n}\{f(x_k)\} - \min(f) \leq \varepsilon$, cada término se limita a $\leq \frac{\varepsilon}{2}$. De esta manera, escogiendo $t = \frac{\varepsilon}{\ell^2}$, se tiene $k = \frac{\ell^2 d(x_0;\mathcal{S})^2}{\varepsilon^2}$. Esta última expresión, demuestra que la tasa de convergencia del método de subgradiente es del orden $\mathcal{O}(\frac{2}{\varepsilon^2})$, más lenta que el método de gradiente clásico.

2.3. Método gradiente acelerado

Con el propósito de mejorar la tasa de convergencia del método de gradiente, se incorpora una variante de aceleración en el proceso iterativo. A este nuevo esquema de iteración se denomina algoritmos de gradiente acelerado. Este tipo de métodos se puede representar por la idea de bola pesada con fricción (*Heavy Ball Method*) de Polyak (1964, 1987), donde se vale de la idea de una bola ubicada en cierto plano que sigue la dirección de gradiente, y al conseguir determinada velocidad deja de seguir la regla del máximo descenso. El momento (inercia) acumulado condiciona para que esta bola continúe en la dirección previa. Las posibles oscilaciones se reducen en las zonas muy curvas sólo combinando el gradiente de signo opuesto. La aceleración es permanente en las zonas suaves donde el gradiente es persistente. El esquema de iteración se construye agregando un nuevo término de momento (inercia) $\beta(x_k - x_{k-1})$, al método de gradiente de máximo descenso, señalado en la ecuación (2.4).

$$x_{k+1} = x_k - \alpha\nabla f(x_k) + \beta(x_k - x_{k-1}) \tag{2.18}$$

Este método, evidencia una tasa de convergencia lineal (ver [19, Teorema 1, sección 3.2.1]). Se demuestra que para un punto mínimo x^* de $f(x), x^* \in \mathbb{R}^n$, con los parámetros,

$$0 \leq \beta < 1, \;\; 0 < \alpha < \frac{2(1+\beta)}{L}, \;\; \ell I \leq \nabla^2 f(x^*) \leq LI$$

donde, ℓ y L son respectivamente el mínimo y máximo valor propio de $\nabla^2 f(x^*)$. La secuencia generada $\{x_k\}$ converge al mínimo x^* con una tasa de progresión geométrica (lineal),

$$\|x_k - x^*\| \leq c(\delta)(q+\delta)^k, \;\; 0 \leq q < 1, \;\; 0 < \delta < 1 - q.$$

El mecanismo de incorporar uno o más puntos previos en el esquema iterativo general, Polyak los denomina métodos de múltiples pasos, y en particular, el esquema que considera sólo un par de puntos (el actual y previo), se refiere a un esquema de dos pasos. Para encontrar la tasa de convergencia del algoritmo con aceleración de dos pasos, se propone aumentar la dimensión del espacio para reducir el proceso de dos pasos a otro de un paso. Para lograr esto, se define entonces, un nuevo esquema iterativo modificado de un paso,

$$z_{k+1} = Az_k + o(z_k),$$

estableciendo el vector z_k de dimensión $2n$,

$$z_k = \begin{bmatrix} x_k - x^* \\ x_{k-1} - x^* \end{bmatrix},$$

y la matriz

$$A \coloneqq \begin{bmatrix} (1+\beta)I - \alpha\nabla^2 f(x^*) & -\beta I \\ I & 0 \end{bmatrix}.$$

Se observa que la matriz A tiene los mismos valores propios que $\nabla^2 f(x^*)$, entonces los valores de α y β se escogen de manera de minimizar el máximo valor propio de A, obteniendo los siguientes valores óptimos para α, β, y q,

$$q^* = \frac{\sqrt{L} - \sqrt{\ell}}{\sqrt{L} + \sqrt{\ell}} \quad , \quad \alpha = \frac{4}{(\sqrt{L} + \sqrt{\ell})^2} \quad , \quad \beta = \left(\frac{\sqrt{L} - \sqrt{\ell}}{\sqrt{L} + \sqrt{\ell}}\right)^2.$$

El método demuestra que para los valores de α y β óptimos (señalados arriba), la distancia $\|x_k - x_*\|_2$, proporciona una tasa aproximada de convergencia de $(1 - \frac{2}{\sqrt{\kappa}+1})$, con $\kappa = \frac{L}{\ell}$ un número de condición.

Comparando la tasa de convergencia del método de gradiente de máximo descenso (un paso) para el caso ideal de una función fuertemente convexa (2.15), con este método de gradiente acelerado de dos pasos, definido con un conjunto de parámetros óptimos (α, y β). Se obtiene respectivamente, en ambos casos, una tasa de convergencia lineal,

$$q_{\text{un paso}} = 1 - \frac{2}{\kappa + 1} \quad , \quad q_{\text{dos pasos}} = 1 - \frac{2}{\sqrt{\kappa} + 1}.$$

Sin embargo, el método de gradiente requiere más iteraciones que la versión con acelera-

ción. Por ejemplo lograr un valor tolerancia similar $\|x_k - x^*\| < \varepsilon\|x_0 - x^*\|$, se requieren

$$k \geq \frac{\kappa}{2} \log \frac{1}{\varepsilon}$$

iteraciones para el método de gradiente, comparado con

$$k \geq \frac{\sqrt{\kappa}}{2} \log \frac{1}{\varepsilon}$$

iteraciones para el gradiente acelerado. Un factor de diferencia de $\sqrt{\kappa}$.

De igual modo, el trabajo de Nesterov (1983, 2013) presenta un esquema general que denomina método óptimo para lograr una tasa de convergencia óptima. Señala que la tasa de convergencia del método de gradiente se puede aumentar, incorporando la información producida por la evolución del proceso. El método de gradiente acelerado se independiza del principio de iteración descendente (el cual garantiza convergencia en el proceso de minimización), y propone la necesidad de introducir aceleración en el avance de las iteraciones, lo que resulta en una mejora de la tasa global de convergencia. Este método plantea aprovechar el momento que generan los puntos previos, en la secuencia de valores producidas, y de esta forma incrementar (o disminuir) el paso de la dirección de descenso en las siguientes iteraciones. Este método incorpora el término de momento como un sumando adicional dentro del paso de gradiente en la iteración

$$\begin{aligned} x_{k+1} &= x_k + p_k \\ p_{k+1} &= \beta_k(x_k - x_{k-1}) - \alpha_k \nabla f(x_k + \beta_k(x_k - x_{k-1})). \end{aligned} \tag{2.19}$$

El tamaño de paso α_k es una secuencia de valores que se ajusta en cada iteración, de manera que sea el menor valor recíproco del coeficiente Lipschitz observado del gradiente de f en la trayectoria de optimización ([16, eq. 4]). Sin embargo, se puede escoger también con algunas de las formas señaladas anteriormente en la sección 2.1.2, o simplemente si el valor de la constante de gradiente Lipschitz es conocida L, se emplea $\alpha_k = \frac{1}{L}$.

Por otra parte, el factor β_k es una sucesión monótona creciente de valores, que aproxima por debajo al valor 1 para un k grande, definido [16, eq. 5] de la siguiente manera,

$$\begin{aligned} a_{k+1} &= \frac{1 + \sqrt{4a_k^2 + 1}}{2} \\ \beta_k &= \frac{a_k - 1}{a_{k+1}}. \end{aligned} \tag{2.20}$$

Una variación del coeficiente β_k es el factor $\frac{k-1}{k+2}$ presentada en [23], el cual genera una secuencia de valores similar a la representación original. El proceso iterativo general del gradiente acelerado, también puede ser interpretado, como el resultado de aproximar una ecuación diferencial de segundo orden [23], que describe una dinámica de descenso acelerado, conjuntamente con el coeficiente de amortiguamiento $\beta > 0$ que contribuye a disminuir las oscilaciones. Esta variante del coeficiente β_k sirve de base para derivar el coeficiente de valor constante 3 en la ecuación diferencial que modela el esquema de Nesterov, en el que se afirma que este valor asegura tasa de convergencias mayores. Se puede advertir que dejando $\beta = 0$

en (2.19) se recupera el método de gradiente.

Independiente de la forma de incluir el término de momento $\beta(x_k - x_{k-1})$ en los procesos de iteración, la tasa de convergencia de ambos esquemas es el del orden $\mathcal{O}(\frac{1}{k^2})$. El método de Nesterov, en condiciones que el gradiente de la función objetivo f es Lipschitz de constante L, con un tamaño de paso definido $\alpha = \frac{1}{L}$, la tasa de convergencia (lineal) es la siguiente, [15, Teorema 2.2.2].

$$f(x_k) - f(x_*) \leq \frac{4L}{(k+2)^2}\|x_0 - x_*\|^2. \tag{2.21}$$

Se puede observar que, el método con aceleración no requiere que la función f sea estrictamente convexa, para lograr esta tasa de convergencia lineal del orden de $\mathcal{O}(\frac{1}{k^2})$, tasa de convergencia similar a la obtenida en convexidad fuerte (2.15). Esta mejora de la tasa de convergencia se obtiene abandonando una sucesión estrictamente decreciente y se permite ciertas oscilaciones controladas por el parámetro β.

2.4. Ecuación Diferencial que modela el método de gradiente acelerado

El estudio de Su, Boyd, and Candes [23] establece que la trayectoria de convergencia de un proceso iterativo en optimización numérica, puede ser modelada por una ecuación diferencial ordinaria, mediante la reducción de los pasos discretos del proceso de optimización. De este modo, presentan una conexión entre el esquema iterativo de Nesterov [15] con una ecuación diferencial de segundo orden, donde esta ecuación resulta de reducir al límite el tamaño del paso de este método. Demostrando, además, que un esquema de primer orden puede ser modelado por una ecuación diferencial ordinaria de segundo orden (ODE).

La derivación de esta ecuación diferencial fue realizada tomando el esquema iterativo del método de gradiente acelerado de Nesterov [16], que se muestra a continuación, y asumiendo que el algoritmo se inicia con $x_0 = y_0$.

$$\begin{aligned} x_k &= y_{k-1} - s\nabla f(y_{k-1}) \\ y_k &= x_k + \frac{k-1}{k+2}(x_k - x_{k-1}). \end{aligned} \tag{2.22}$$

Luego, mediante una combinación de ambas expresiones de este proceso (2.22). Formulando además, la discretización de una curva $X(t)$ para $t > 0$ con la aproximación $x_k \approx X(k\sqrt{s})$ (*Ansatz*). Consiguen la equivalencia entre un punto de valor contínuo y discreto $X(t) \approx x_{t/\sqrt{s}} = x_k$, tomando el límite del tamaño de paso s en el término definido $k = t/\sqrt{s}$, y finalmente mediante un manejo de la aproximación de Taylor se obtiene la ecuación diferencial de segundo orden.

$$\ddot{X} + \frac{3}{t}\dot{X} + \nabla f(x) = 0, \tag{2.23}$$

con $\ddot{X} = \frac{d^2X}{dt^2}$ el factor aceleración y $\dot{X} = \frac{dX}{dt}$ la velocidad o la derivada de X con respecto al tiempo. Las condiciones iniciales $X(0) = x_0$ y $\dot{X} = 0$ es el punto de inicio del algoritmo de

Nesterov.

Se puede advertir que el valor constante 3 es un factor importante en esta ecuación diferencial. El componente $3/t$, por una parte, resulta de la aproximación al factor $(k-1)/(k+2) \approx 1-3/k$, coeficiente de fricción β del esquema iterativo de Nesterov. Además, se afirma que el valor constante 3 es el menor valor que garantiza la tasa de convergencia $\mathcal{O}(\frac{1}{t^2})$.

Asimismo, el factor de $3/t$ es interpretado como un factor de amortiguamiento de la dinámica de segundo orden. Es el elemento que permite relacionar la dinámica de un sistema amortiguado con el comportamiento iterativo del esquema de Nesterov. Se puede deducir que la parte inicial de la dinámica, el factor de amortiguamiento $\frac{3}{t}$ es grande y ayuda que la respuesta de la ecuación actúe como un sistema sobreamortiguado, para lograr parte del equilibrio sin oscilaciones. Sin embargo, a medida que el valor de t aumenta, esta ecuación responde como un sistema sub-amortiguado, produciendo oscilaciones en la zona de equilibrio con valores de amplitud que progresivamente llegan a cero. Finalmente, se demuestra que la tasa de convergencia de la ecuación diferencial es similar al caso discreto.

$$f(X(t)) - f^* \leq \mathcal{O}(\frac{\|x_0 - x^*\|^2}{t^2}). \tag{2.24}$$

Capítulo 3

Algoritmos Proximales

Los algoritmos de tipo de proximal forman parte de un grupo de métodos de primer orden, que permiten resolver problemas convexos no diferenciables. Esto es, el vector de gradiente, empleado en el esquema de minimización del algoritmo de máximo descenso no es *Lipschitz* (función no suave), o no existe en algunos puntos. Para abordar esta dificultad, el problema original se transforma en uno fuertemente convexo, valiéndose de una serie de funciones convexas que aproximan por debajo a la función no-diferenciable. Esta aproximación se basa en las propiedades de la regularización de *Moreau-Yosida* que ayuda a conseguir una función equivalente suave y diferenciable.

Se define la regularización de *Moreau-Yosida* para una función $f : X \to \mathbb{R} \cup \{+\infty\}$, propia, convexa semicontinua inferior de parámetros (λ, x), $X \subset \mathbf{R}^n$, de la siguiente manera

$$f_{(\lambda,x)}(z) = f(z) + \frac{1}{2\lambda}||z - x||_2^2,\ \lambda > 0 \text{ para todo } x \in X. \tag{3.1}$$

Una de las propiedades importantes de esta función $f_{(\lambda,x)}$, es que asegura la existencia de un mínimo $\bar{x} \in X$ y es único [18, Teorem 2.19]. La existencia se puede obtener al observar que la función $y \mapsto f(y) + \frac{1}{2\lambda}||x - y||^2$ es estrictamente convexa, por consiguiente, garantiza al menos un mínimo. Y el mínimo es único, suponiendo que la función es propia, convexa, semicontinua inferior y coerciva. Por su parte, tomando como base la regla generalizada de Fermat [18, Teorem 3.24],

$$\bar{x} \in \arg\min_X(f) \iff 0 \in \partial f(\bar{x})$$

y la regla de suma para subdiferenciales (Teorema de Moreau-Rockafellar),

$$\partial f(x) + \partial g(x) \subset \partial(f + g)(x)$$

el mínimo $\bar{x} \in X$ queda caracterizado por la siguiente relación de inclusión,

$$0 \in \partial f_{(\lambda,x)}(\bar{x}) = \partial f(\bar{x}) + \frac{\bar{x} - x}{\lambda}\ ,\ \forall x \in X \tag{3.2}$$

despejando $\bar{x}$ de esta inclusión diferencial (3.2)

$$\bar{x} = (I + \lambda\partial f)^{-1}(x). \tag{3.3}$$

La expresión $(I+\lambda\partial f)^{-1}$ es una función que se denomina operador de proximidad de la función f con parámetro λ, donde $I : X \to X$ es la una función de identidad, y se representa por la identidad $\texttt{prox}_{\lambda \texttt{f}} = (I+\lambda\partial f)^{-1}$. Este resultado se denomina también *resolvente* del operador subdiferencial ∂f con parámetro $\lambda > 0$. Se demuestra, que para una función convexa, propia, semicontinua inferior, el operador de proximidad es no-expansivo, $\bar{x} = (I + \lambda\partial f)^{-1}(x)$, $\bar{y} = (I + \lambda\partial f)^{-1}(y)$

$$||\texttt{prox}_{\lambda f}(x) - \texttt{prox}_{\lambda f}(y)|| \leq ||x - y||$$

el operador proximal es Lipschitz continuo de constante 1.

Se puede advertir, que el resolvente $(I + \lambda\partial f)^{-1}$ es un operador que relaciona el subdiferencial ∂f (conjunto de subgradientes en un punto donde la función no es diferenciable) con el operador proximal $\texttt{prox}_{\lambda f}$. Este operador, toma un punto (vector) del dom(f) y lo asocia con otro punto del dom(f). Esta relación se verifica de la siguiente manera

$$\begin{aligned}
&z \in (I + \lambda\partial f)^{-1}(x) \\
&x \in (I + \lambda\partial f)(z) = z + \lambda\partial f(z) \\
&0 \in \partial f(z) + \frac{1}{\lambda}(z - x) \\
&0 \in \partial_z(f(z) + \frac{1}{2\lambda}(z - x)^2),
\end{aligned}$$

donde, en la última expresión, el subdiferencial es con respecto a z, y se observa que la condición necesaria y suficiente para que z sea mínimo, de la función fuertemente convexa dentro de los paréntesis, tiene que verificar lo siguiente

$$z = \arg\min_{w \in \mathbb{R}^n} \left\{ f(w) + \frac{1}{2\lambda}\|w - x\|_2^2 \right\} \tag{3.4}$$

demostrando que $z \in (I + \lambda\partial f)^{-1}(x)$ si, y solo si $z = \texttt{prox}_{\lambda f}(x)$.

3.1. Función regularizada de Moreau

El propósito de la función regularizada de *Moreau-Yosida* $f_{(\lambda,x)}$ es usarla como base para obtener una representación suavizada de la función original f. Se define, entonces, la función de aproximación (o recubrimiento) de *Moreau* $f_\lambda(x)$, con parámetro λ, asumiendo que la función f, es convexa, propia, semicontinua inferior, y no suave,

$$f_\lambda(x) = \inf_{z \in X} \left\{ f(z) + \frac{1}{2\lambda}||z - x||_2^2 \right\}. \tag{3.5}$$

Las ventajas de la aproximación de *Moreau* es que transforma cualquier función f (generalmente no suave) en una función diferenciable que preserva el mínimo y los minimizadores de f. Igualmente, para todo $x \in \text{dom}(f)$ se tiene también que $f_\lambda(x) \leq f(x)$. De esta manera, cualquier función f se puede aproximar a una función suave por debajo, con exactitud proporcionada por el parámetro λ.

Se define también, el operador proximal $\texttt{prox}_f : \mathbb{R}^n \to \mathbb{R}^n$, con parámetro λ, mediante la

siguiente expresión

$$\texttt{prox}_{\lambda f}(x) = \arg\min_{z\in\mathbb{R}^n}\left\{f(z) + \frac{1}{2\lambda}\|z-x\|_2^2\right\}. \tag{3.6}$$

El operador proximal, en línea muy general se puede interpretar como una operación de aproximación, donde la entrada es el valor x y su salida es un punto $z \in \mathbb{R}^n$ cercano a este valor x, que además minimiza la función f. La constante λ controla el grado de avance hacia el mínimo, similar al factor α de tamaño de paso en los métodos de gradiente. En resumen, el operador proximal devuelve un mínimo cercano al argumento de entrada, y dependiendo del valor de λ es el avance hacia ese mínimo.

Asimismo, el operador proximal $\texttt{prox}_{\lambda \texttt{f}}(x)$ y la función de aproximación de *Moreau* $f_\lambda(x)$ se relacionan mediante la siguiente expresión

$$f_\lambda(x) = f(\texttt{prox}_{\lambda \texttt{f}}(x)) + \frac{1}{2\lambda}||x - \texttt{prox}_{\lambda \texttt{f}}(x)||^2. \tag{3.7}$$

La función $f_\lambda(x)$ es convexa, propia, gradiente Lipschitz continua de constante $\frac{1}{\lambda}$

$$\nabla f_\lambda(z) = \frac{1}{\lambda}(I - \texttt{prox}_{\lambda \texttt{f}}(z)), \quad \text{para cada } z \in X, \tag{3.8}$$

reordenando esta expresión, se obtiene el operador proximal en términos del gradiente de la aproximación de Moreau

$$\texttt{prox}_{\lambda \texttt{f}}(x) = x - \lambda\nabla f_\lambda(x). \tag{3.9}$$

La evaluación del operador proximal se puede interpretar como un paso de gradiente, que minimiza una versión regularizada de la función original con un parámetro de paso λ.

La función f_λ es un ejemplo particular, de una operación más general de convolución entre dos funciones, denominado convolución de ínfimos. En el cual, la operación realiza una convolución entre ambas funciones, y proporciona el ínfimo de cada valor x para todos los valores de z, y es definido de la siguiente forma [17].

$$(f\square g)(x) = \inf_x(f(z) + g(x-z)).$$

El resultado de la convolución, en una descripción muy simple, permite construir un recubrimiento de la función f, con los ínfimos que resultan, al convolucionar la función g para cada valor de x. Del mismo modo, la aproximación de *Moreau* produce una envolvente, que es una forma suave y diferenciable de la función f. Sin embargo, el aspecto más importante de esta aproximación es la existencia y unicidad de un minimizador único. Esta característica se hereda de las propiedades de la función regularizada de *Moreau-Yosida*, y en consecuencia debido a la definición se tiene.

$$\inf_{z\in X} f(z) \leq f_\lambda(x) \leq f(x),\ \lambda > 0\ \forall x \in X$$
$$\inf_{z\in X} f(z) = \inf_{z\in X} f_\lambda(z)\square$$

De esta manera, sí el valor $\bar{x} \in X$ minimiza la función objetivo f, entonces, también minimiza la envolvente suave de *Moreau*, y en el caso reverso, sí minimiza la función suave también la no suave. Finalmente, el problema de minimización es equivalente para ambas

funciones, debido que el mínimo del problema es el mismo para la función suave generada por la envolvente de *Moreau* y la función no-diferenciable del problema.

La figura 3.1 presenta un ejemplo del operador proximal y la envolvente de Moreau, para una función convexa no diferenciable definida en partes, representada en azul

$$f(x) = \begin{cases} -\frac{1}{2}x + 2 & x < 4 \\ \frac{1}{2}x - 2 & 4 \leq x \leq 6 \\ x - 5 & x > 6. \end{cases}$$

La línea en rojo corresponde a la envolvente de Moreau $f_\lambda(x)$ con un parámetro $\lambda = 1$. La línea gris muestra la función $f(x) + {}^1/_2(x - x_0)^2$ para $x_0 = 5$, donde el mínimo (cruz en rojo) define el operador de proximidad. Se verifica que $\texttt{prox}_{\lambda f}(x_0) = 4.5$ se encuentra más cerca del mínimo que x_0.

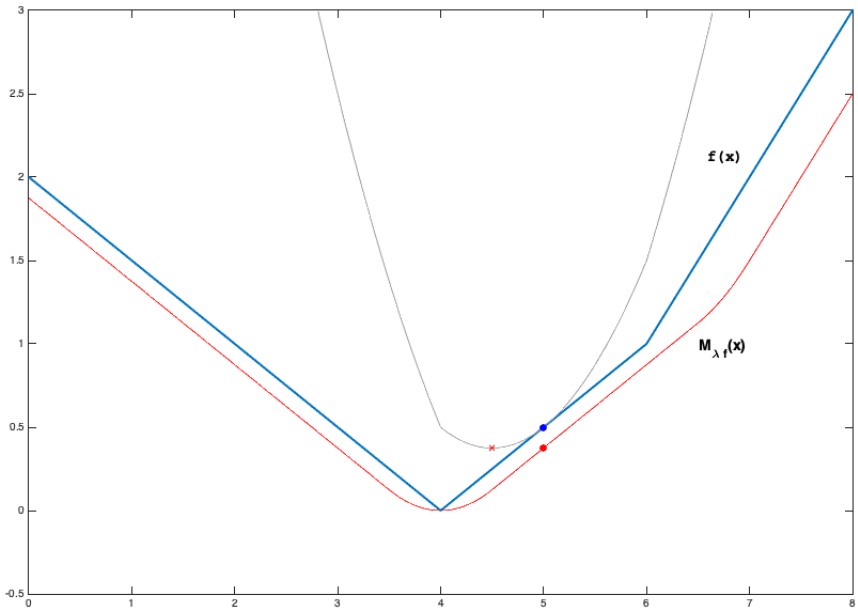


Figura 3.1: Ejemplo de operador proximal y envolvente de Moreau para una función convexa no diferenciable definida por partes. La función convexa $f(x)$ no-diferenciable está representada por la línea azul, la función suavizada de Moreau se muestra por la curva en color rojo. La línea gris presenta un ejemplo de evaluación del operador proximal en el punto $x_0 = 5$.

3.2. Función proximal para la norma ℓ_1

Se puede derivar de manera muy simple la función *proximal* de algunas funciones convexas conocidas, sólo incorporando el término cuadrático a la función y manipulando algebraica-

mente para obtener los puntos críticos de la función proximal. Una de estas funciones es la función de norma ℓ_1 empleada como función de regularización en el problema tipo *LASSO* y tiene especial relevancia en el desarrollo de este trabajo. Esta función en particular fomenta la generación de matrices dispersas o *sparse*.

A continuación, se muestra la derivación de la función *proximal* de la función de norma ℓ_1 para el caso en $\mathbb{R}$, posteriormente se puede expandir fácilmente a $\mathbb{R}^n$. Sea la función $f : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$, $f(x) = \lambda|x|$. Entonces, la función regularizada de *Moreau-Yosida* $f_{(\lambda,x)}(z)$, para todo $x, z \in \mathbb{R}$ y $\lambda > 0$, se define de la siguiente forma:

$$f_{(\lambda,x)}(z) = \begin{cases} f_1 \equiv \lambda z + \frac{1}{2}(z-x)^2 & z > 0 \\ f_2 \equiv -\lambda z + \frac{1}{2}(z-x)^2 & z \leq 0 \end{cases} \tag{3.10}$$

i) $f_1'(z) = \lambda + z - x = 0$ el mínimo se obtiene en $z = x - \lambda$. Entonces si $x > \lambda$ el $\mathbf{prox}_{\lambda f}(x) = x - \lambda$.

ii) $f_2'(z) = -\lambda + z - x = 0$ el mínimo se obtiene en $z = x + \lambda$. Entonces si $x < -\lambda$ el $\mathbf{prox}_{\lambda f}(x) = x + \lambda$.

iii) Finalmente si $|x| \leq \lambda$ entonces $\mathbf{prox}_{\lambda f}(x) = 0$ el punto no diferenciable.

A continuación se muestra la definición de la función proximal de $f(x) = \lambda|x|$, y la figura (3.2) señala un ejemplo de esta función proximal para un $\lambda = 1$.

$$\mathbf{prox}_{\lambda f}(x) = \begin{cases} x - \lambda & , x > \lambda \\ 0 & , -\lambda \leq x \leq \lambda \\ x + \lambda & , x < -\lambda \end{cases}$$

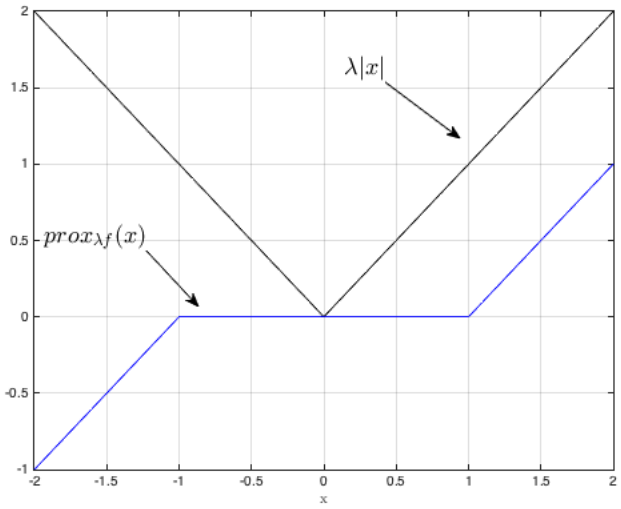


Figura 3.2: Operador proximal para la función $f(x) = \lambda|x|$.

Esta función se denomina en inglés operador de *Soft-Thresholding* (límite suave) o recorte, tomando ahora para $f : \mathbb{R}^n \to \mathbb{R}$ la función $f(x) = \lambda||x||_1$, con $\lambda > 1$, entonces para $\mathbb{R}^n$ se define el operador *Soft-Thresholding* $\mathcal{T}_\lambda$

$$[\mathcal{T}_\lambda(x_k)]_i = \begin{cases} [x_k]_i + \lambda & , [x_k]_i < -\lambda \\ 0 & , -\lambda \leq [x_k]_i \leq \lambda \\ [x_k]_i - \lambda & , [x_k]_i > \lambda. \end{cases} \tag{3.11}$$

3.3. Métodos de Discretización Implícito/Explícito

El proceso iterativo de minimización basado en operación proximal, se puede interpretar como un método de discretización que resuelve una ecuación diferencial, donde los puntos críticos ($\nabla f(x) = 0$) corresponden a los mínimos de una función f convexa diferenciable. Por medio de una estrategia de discretizar y luego optimizar [18] se puede aproximar un problema infinito-dimensional, a uno finito-dimensional. Una técnica de aproximación es utilizar diferencias finitas para discretizar la ecuación diferencial, en el que el dominio se divide (particiona) en bloques muy pequeños de dimensión n, que corresponde a la dimensión $\mathbb{R}^n$ de la aproximación. De este tipo de aproximación surgen dos clases de métodos iterativos, uno de denominado *explícito* y otro *implícito*. Los cuales tienen directa relación con el método de gradiente y proximal respectivamente.

Asumiendo que la función $f : X \subset \mathcal{H} \to \mathbb{R} \cup \{+\infty\}$ es continua y diferenciable, se puede representar una ecuación diferencial de esa función f a través de la siguiente expresión,

$$\dot{x}(t) = -\nabla f(x), \quad x(0) = x_0$$

la cual tiene solución única para $x_0 \in X$. Los puntos estacionarios de este sistema son los puntos mínimos de la función f. Esta ecuación resuelve el problema de minimización de la función f, debido la trayectoria de x se moverá en el sentido de $f(x(t)) \to f(x^*)$, donde $f(x^*)$ es el mínimo de f.

La primera forma de discretizar la ecuación diferencial es el método de Euler, para un paso de discretización h, se aproxima el gradiente mediante el siguiente esquema

$$\frac{x_{k+1} - x_k}{h} = -\nabla f(x_k)$$

en este caso, el gradiente del punto x_k se estima tomando la diferencia entre el siguiente punto x_{x+1} y el punto actual x_k, luego resolviendo para x_{k+1} se obtiene en forma *explícita* el método del gradiente

$$x_{k+1} = x_k - h\nabla f(x_k).$$

La segunda forma de discretizar, el gradiente se estima en x_{k+1}

$$\frac{x_{k+1} - x_k}{h} = -\nabla f(x_{k+1}).$$

Resolviendo nuevamente para x_{k+1} se obtiene una regla actualización *ímplicita*

$$x_{k+1} + h\nabla f(x_{k+1}) = x_k.$$

Generalizando a casos donde la función f no es diferenciable, la ecuación diferencial

asociada con el gradiente de f, se puede extender utilizando la inclusión diferencial del subgradiente

$$\dot{x}(t) \in -\partial f(x(t)).$$

Asimismo, esta última inclusión, se puede generalizar la regla de discretización ímplicita.

$$\frac{x_{k+1} - x_k}{h} \in -\partial f(x_{k+1})$$

La inclusión subdiferencial de esta discretización se puede escribir de similar forma a la ecuación (3.3)

$$x_{k+1} = (I + h\partial f)^{-1}(k_n)$$

La ecuación de este esquema es equivalente al método de minimización proximal

$$\mathbf{prox}_{hf} = (I + h\partial f)^{-1}$$

$$x_{k+1} = \mathbf{prox}_{hf}(x_k)$$

El método de discretización *implícita* presenta mejores propiedades de estabilidad que el método de Euler, y se puede aplicar a funciones no diferenciables. El parámetro h utilizado en la discretización es el parámetro λ empleado en el término cuadrático del operador proximal.

3.4. Algoritmo punto Proximal

El método de punto proximal, de forma similar al método de gradiente, genera a partir de un punto de inicial $x_0 \in \mathbb{R}^n$, una secuencia de puntos realizando evaluaciones del operador proximal. Para una función $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ convexa, propia, cerrada, el operador proximal proporciona un punto mínimo "próximo" al punto de evaluación, o el mismo punto en caso de estar en el conjunto de mínimos del problema.

$$x_{k+1} := \mathbf{prox}_{\lambda_k f}(x_k). \tag{3.12}$$

El nivel de avance se define por el valor de λ que controla el grado de regularización de la función proximal. Pequeños valores de λ_k causa el siguiente punto x_{k+1} se mantenga cerca de x_k que significa muchos pasos (lenta convergencia) del proceso para llegar al mínimo. Se demuestra que los puntos estacionarios de la secuencia generada por las evaluaciones del operador proximal son los minimizadores de la función objetivo.

Las principales características de las secuencias $\{x_k\}$ generadas con evaluaciones del método de punto proximal, asumiendo existe un conjunto de solución no vacío, son las siguientes:

i) la sucesión de $f(x_k)$ es decreciente, disminuye estrictamente mientras $x_{k+1} \neq x_k$.

ii) La función f, bajo supuestos generales, converge débilmente a un punto de este conjunto de solución no vacío. Lo relevante es que la estimación de un punto x_{k+1} requiere tener información de ese punto, advirtiendo que la información futura también se considera en la estimación del siguiente punto x_{k+1}, asegurando sucesiones más estables.

iii) El vector de desplazamiento forma un ángulo agudo entre los desplazamientos $x_{k+1} - x_k$ y $x_k - x_{k-1}$, en dirección del conjunto de mínimos. Permitiendo un avance más directo, y no zigzagueante, hacia el conjunto solución.

3.5. Algoritmo gradiente proximal

El método de *gradiente proximal* es un esquema más elaborado, que se aplica a un problema de optimización general, modelado por una composición de funciones, similar a la definición dada en (2.2). Para la elaboración de este método se asume que la función $f : \mathbb{R}^n \to \mathbb{R}$ es convexa, continua, diferenciable de constante Lipschitz $L > 0$ y una función $h : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ convexa, propia, no diferenciable

$$\min_{x \in \mathbb{R}^n} \Big\{ F(x) = f(x) + h(x) \Big\}. \tag{3.13}$$

El método de *gradiente proximal* se construye a partir de una aproximación lineal de la función f en el punto x_k, regularizado por un término *proximal* cuadrático que proporciona una medida local de error

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^n} \Big\{ f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{2\lambda_k} ||x - x_k||^2 + h(x) \Big\}.$$

Esta aproximación resulta en una expresión de aproximación cuadrática fuertemente convexa, la cual se puede expandir algebráicamente de la siguiente forma

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^n} \Big\{ f(x_k) + \frac{1}{2\lambda_k} ||x - (x_k - \lambda_k \nabla f(x_k))||^2 - \frac{\lambda_k}{2} ||\nabla f(x_k)||^2 + h(x) \Big\}.$$

Eliminando el término constante que depende de x_k

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^n} \Big\{ h(x) + \frac{1}{2\lambda_k} \|x - (x_k - \lambda_k \nabla f(x_k))\|^2 \Big\}.$$

Comparando esta expresión con la definición del operador proximal (3.6), se obtiene la relación para el método de *gradiente proximal*

$$x_{k+1} = \text{prox}_{\lambda_k h}\Big(x_k - \lambda_k \nabla f(x_k)\Big). \tag{3.14}$$

Se puede notar que cada iteración de este algoritmo se compone de dos pasos:

1. Un primer paso de gradiente el cual define un punto intermedio w_k, tomando el gradiente de la función diferenciable $f(x)$

$$w_k = x_k - \lambda \nabla f(x_k).$$

2. Un segundo paso, donde el operador proximal evalúa de la parte no diferenciable $h(x)$, esto es el punto intermedio obtenido en el paso previo

$$x_{k+1} = \texttt{prox}_{\lambda_k \texttt{h}}(w_k) = \texttt{prox}_{\lambda_k \texttt{h}}(x_k - \lambda_k \nabla f(x_k)).$$

El tamaño del paso de avance λ_k se delimita por el valor de la constante Lipschitz, $\lambda \in (0, \frac{1}{L})$. Cuando es dificil estimar la constante L se usa la regla *backtracking* para obtener un paso de avance que se ajusta en cada iteración.

Dependiendo del tipo de problema de optimización (por ejemplo con/sin restricciones, o el modelo compuesto), el paso de actualización de una iteración puede variar, derivando en diferentes versiones de algoritmos de primer orden.

i) Optimización sin restricciones: $\min_{x\in\mathbb{R}^n}\{f(x)\}$
método de gradiente: $x_{k+1} = x_k - t_k\nabla\{f(x_k)\}$

ii) Optimización con restricciones: $\min_{x\in C\subset\mathbb{R}^n} f(x)$
método de gradiente proyectado: $x_{k+1} = P_C(x_k - t_k\nabla f(x_k))$

iii) Optimización de composición de funciones (3.13): $\min_{x\in\mathbb{R}^n}\{f(x) + h(x)\}$
método ISTA: $x_{k+1} = \mathcal{T}_{\lambda t_k}(x_k - t_k\nabla f(x_k))$. Cuando $h(x) \equiv \|x\|_1$

El método de gradiente proyectado [7], mencionado en el punto (ii), P_C es la proyección del paso de gradiente en el conjunto de restricciones $C \subset \mathbb{R}^n$ (se asume la proyección está bien definida, debido el conjunto C es cerrado y convexo). El método ISTA del punto (iii) será descrito en el siguiente capítulo.

3.5.1. Tasa de convergencia

Una secuencia de puntos $\{x_k\}$ generada por el método de gradiente proximal, con un paso de avance constante (o variable usando la técnica de *backtracking*), evidencia una tasa de convergencia del orden de $\mathcal{O}(\frac{1}{k})$. Se observa que la tasa de convergencia del método de gradiente proximal es del mismo orden que la tasa convergencia del método clásico de gradiente señalado en (2.14)

$$F(x_k) - F(x^*) \leq \frac{L\|x_0 - x^*\|^2}{2k}.$$

Una manera de comprobar la tasa de convergencia, es definiendo un gradiente generalizado de f

$$G_\lambda(x) = \frac{x - \texttt{prox}_{\lambda f}(x - \lambda\nabla f(x))}{\lambda}. \tag{3.15}$$

Asumiendo que el tamaño de paso es fijo $\lambda_k = \lambda$ con $\lambda < \frac{1}{L}$, y la función (3.13) compuesta F tiene un mínimo $F(x^*)$. Un paso de iteración del algoritmo de gradiente proximal (3.14) se puede representar en forma equivalente

$$x_{k+1} = x_k - \lambda \cdot G_\lambda(x_k). \tag{3.16}$$

Desde la desigualdad del Lema de descenso (A.2) y aplicando la dirección del gradiente generalizado (3.16) en la función F, se obtiene una cota superior de la función F en el punto x_{k+1}

$$\begin{aligned} F(x_{k+1}) &= f(x_{k+1}) + h(x_{k+1}) \\ &\leq f(x_k) - \lambda\nabla f(x_k)^T \underbrace{G_\lambda(x_k)}_{\frac{x_{k+1}-x_k}{\lambda}} + \frac{L\lambda^2}{2}\|G_\lambda(x_k)\|^2 + h(x_{k+1}). \end{aligned} \tag{3.17}$$

Interesa obtener un límite superior de esta última desigualdad (3.17) para cualquier punto arbitrario $z \in \mathbb{R}^n$. Utilizando para esto, la propiedad de convexidad de la función f en el punto x_k,

$$f(z) \geq f(x_k) + \nabla f(x_k)^T (z - x_k) \;\text{ para todo } z \in \mathbb{R}^n \tag{3.18}$$

y la definición del proximal,

$$\begin{aligned} x_{k+1} &= \texttt{prox}_{\lambda h}(x_k - \lambda \nabla f(x_k)) \\ &= \arg\min_z \{h(z) + \frac{1}{2\lambda}\|z - (x_k - \lambda \nabla f(x_k))\|^2\} \end{aligned}$$

$$\begin{aligned} &0 \in \partial\Big\{h(z) + \tfrac{1}{2\lambda}\|z - (x_k - \lambda \nabla f(x_k))\|^2\Big\} \\ &0 \in \partial h(x_{k+1}) + \partial\Big\{\frac{1}{2\lambda}\|x_{k+1} - x_k + \lambda \nabla f(x_k)\|^2\Big\} \\ &0 \in \partial h(x_{k+1}) - G_\lambda(x_k) + \nabla f(x_k) \Rightarrow G_\lambda(x_k) - \nabla f(x_k) \in \partial h(x_{k+1}) \end{aligned}$$

se obtiene, de este modo, una cota superior para la función h

$$h(z) \geq h(x_{k+1}) + (G_\lambda(x_k) - \nabla f(x_k))^T (z - x_{k+1}) \text{ para todo } z \in \mathbb{R}^n \tag{3.19}$$

juntando (3.18), (3.19), en (3.17)

$$\begin{aligned} F(x_{k+1}) &\leq \underbrace{f(z) - \nabla f(x_k)^T (z - x_k)}_{(3.18)} - \lambda \nabla f(x_k)^T G_\lambda(x_k) + \frac{\lambda}{2}\|G_\lambda(x_k)\|^2 \\ &\quad + \underbrace{h(z) - (G_\lambda(x_k) - \nabla f(x_k))^T (z - x_k + \lambda G_\lambda(x_k)))}_{(3.19)} \\ &= f(z) - G_\lambda(x_k)^T (z - x_k) + \lambda(\frac{L\lambda}{2} - 1)\|G_\lambda(x_k)\|^2 \\ &\leq f(z) - G_\lambda(x_k)^T (z - x_k) - \frac{\lambda}{2}\|G_\lambda(x_k)\|^2 \text{ para } \lambda \leq \frac{1}{L}. \end{aligned}$$

Se puede observar que evaluando $z = x_k$ la función no es creciente en cada paso iterativo

$$F(x_{k+1}) \leq F(x_k) - \frac{\lambda}{2}\|G_\lambda(x_k)\|^2 \leq F(x_k)$$

y para el punto óptimo $z = x_*$,

$$\begin{aligned} F(x_{k+1}) - F(x^*) &\leq G_\lambda(x_k)^T (x_k - x^*) - \frac{\lambda}{2}\|G_\lambda(x_k)\|^2 \text{ para } \lambda \leq \frac{1}{L} \\ &= \frac{1}{2\lambda}\Big(2\lambda G_\lambda(x_k)^T (x_k - x^*) - \lambda^2\|G_\lambda(x_k)\|^2 + \|x_k - x^*\|^2 - \|x_k - x^*\|^2\Big) \\ &= \frac{1}{2\lambda}\Big(\|x_k - x^*\|^2 - \|x_k - \lambda G_\lambda(x_k) - x^*\|^2\Big) \\ &= \frac{1}{2\lambda}\Big(\|x_k - x^*\|^2 - \|x_{k+1} - x^*\|^2\Big). \end{aligned}$$

Sumando todos los pasos se obtiene la tasa de convergencia

$$F(x_k) - F(x^*) \leq \frac{1}{k}\sum_{i=1}^{k} F(x_{i+1}) - F(x^*) \leq \frac{\|x_0 - x^*\|^2}{2\lambda k}. \tag{3.20}$$

Se puede ver que, para que un punto $\hat{x}$ arbitrario de la secuencia, se encuentre dentro de un rango de solución $F(\hat{x}) - F(x^*) > \epsilon$, con una tolerancia $\epsilon > 0$ dada. Se requiere al menos $\lceil \frac{C}{\epsilon} \rceil$ iteraciones, donde $C = \frac{L\|x_0 - x^*\|^2}{2}$.

Capítulo 4

Algoritmos Iterativos ISTA/FISTA

Esta sección conecta dos de los algoritmos utilizados en problemas de procesamiento de imágenes, que toman como fundamento el operador proximal descrito en la sección anterior. Se hace una breve descripción del algoritmo denominado ISTA, el cual es una especialización del algoritmo de gradiente proximal (3.14). Utiliza como función de regularización no-suave la norma ℓ_1 de los puntos de entrada. El segundo algoritmo es FISTA, que de manera similar a los algoritmos con aceleración, incorpora la información de momento en su esquema iterativo. Este esquema, al agregar aceleración logra mejorar la tasa de convergencia con respecto a ISTA.

4.1. Método ISTA

El método **ISTA** (*Iterative Shrinkage-Thresholding Algorithm*), es una instancia particular del algoritmo de gradiente proximal referido en (3.14). Se obtiene sustituyendo la función no-diferenciable $h(x)$, del problema de optimización modelado por la función compuesta F (señalada en (3.13)), por la función $\lambda\|x\|_1$. Donde, la función f es ahora regularizada por un factor de norma ℓ_1, y el parámetro de regularización ($\lambda > 0$) nivela, en el caso de procesamiento de imágenes, el grado de dispersión (*sparsity*) inducido por esta función no-suave. De igual manera, que el método de gradiente y la versión gradiente proximal, este método presenta una tasa de convergencia de orden de $\mathcal{O}(\frac{1}{k})$.

El problema regularizado con la norma ℓ_1, expresado por,

$$\min_{x\in\mathbb{R}^n}\Big\{F(x) = f(x) + \lambda\|x\|_1\Big\}, \tag{4.1}$$

presenta un esquema iterativo similar al derivado en (2.5). Es una formación, conformada por la aproximación cuadrática en el punto particular x_k, y la función dada por la suma de los valores absolutos de los componentes de x, regularizado por el parámetro λ,

$$x_{k+1} = \arg\min_x\Big\{f(x_k) + \langle x - x_k, \nabla f(x_k)\rangle + \frac{1}{2t_k}\|x - x_k\|^2 + \lambda\|x\|_1\Big\}. \tag{4.2}$$

De manera análoga, a la operación realizada para obtener el algoritmo de gradiente proximal (utilizando una aproximación lineal de la función en un punto particular), este esquema se puede reordenar, desestimando los términos constantes,

$$x_{k+1} = \arg\min_x \Big\{ \frac{1}{2t_k} \|x - (x_k - t_k \nabla f(x_k))\|^2 + \lambda \|x\|_1 \Big\}. \tag{4.3}$$

El cual, consigue un esquema de iteración combinado por el paso de gradiente $x_k - t_k \nabla f(x_k)$ y la operación de recorte o *soft-thresholding* (3.11) (ver 3.2)

$$x_{k+1} = \mathcal{T}_{\lambda t_k}(x_k - t_k \nabla f(x)), \tag{4.4}$$

donde t_k es el tamaño para el paso de gradiente de la iteración k, y λt_k es el factor utilizado por este operador de recorte suave $\mathcal{T}_\lambda$ definido en la sección 3.2.

4.2. Método FISTA

Una propuesta de mejora para la tasa de convergencia de *ISTA* es agregar aceleración, incorporando información previa para corregir el avance del algoritmo de gradiente proximal. El método **FISTA** (*Fast Iterative Shrinkage-Thresholding Algorithm*) propuesto por Beck and Teboulle [6] genera los pasos iterativos tomando en cuenta la información de dos pasos anteriores

$$\begin{aligned} x_k &= \texttt{prox}_{\lambda \frac{1}{L} h}(y_k - \frac{1}{L} \nabla f(y_k)) \\ t_{k+1} &= \frac{1 + \sqrt{1 + 4t_k^2}}{2} \\ y_{k+1} &= x_k + \frac{t_k - 1}{t_{k+1}}(x_k - x_{k-1}). \end{aligned} \tag{4.5}$$

La principal diferencia de este método con ISTA, reside en que el operador de recorte suave (*soft-thresholding*) no se aplica en el punto anterior, sino en y_k que es una combinación lineal de los puntos $\{x_{k-1}, x_k\}$, ponderados por el factor $\frac{t_k - 1}{t_{k+1}}$ que aumenta en cada iteración, con límite 1.

De igual manera, se puede también puntualizar la diferencia de FISTA con el método de gradiente acelerado de Nesterov [16]. Los dos métodos son muy similares desde el punto de vista de la implementación algorítmica. En el sentido, que ambos precisan de la misma sucesión de valores, en el paso intermedio, para obtener el valor que pondera el término de momento $\beta_k(x_k - x_{k-1})$. Pero, mientras el gradiente acelerado aplica el operador de gradiente en el punto intermedio y_k, FISTA utiliza el operador proximal sobre el gradiente de ese punto intermedio. La tabla 4.1 muestra la diferencia de ambos métodos.

Tabla 4.1: Comparación entres FISTA y el método de gradiente acelerado, ambos aprovechan el término de momento, pero el gradiente acelerado se basa en el operador de gradiente, y FISTA aplica el operador proximal sobre la operación de gradiente descendente.

	Gradiente acelerado	FISTA
(i)	$x_k = y_k - s\nabla f(y_k)$	$x_k = \texttt{prox}_{\lambda sh}(y_k - s\nabla f(y_k))$
(ii)	$y_{k+1} = x_k + \beta_k(x_k - x_{k-1})$	

De esta forma, el algoritmo FISTA se puede también interpretar como una extensión del método de gradiente acelerado de Nesterov [16]. En este aspecto, el algoritmo de Nesterov [16] procesa el avance sólo con la información de gradiente (información de momento) obtenida mediante la combinación de dos puntos previos consecutivos. Adicionalmente, FISTA aplica el operador proximal sobre la información de gradiente de los dos puntos anteriores. En este sentido, el esquema de FISTA se puede comprender como la versión proximal del método de gradiente acelerado de Nesterov [16]. El método de gradiente acelerado, sin embargo, está orientado a problemas de optimización convexa donde la función es suave. FISTA emplea el operador proximal para problemas donde la función convexa no es suave.

La Figura 4.1 señala un desempeño comparativo de los algoritmos de gradiente descendente clásico, gradiente con aceleración de Nesterov, y el algoritmo FISTA, para un tamaño de paso fijo, aplicados a una función convexa $f(x_1, x_2) = x_1^2 + 2x_2^2 + 3$. Se puede notar que FISTA emplea menos pasos iterativos, que el gradiente acelerado y gradiente clásico para alcanzar el mínimo. También se puede ver que el método de gradiente clásico experimenta muchos pasos para llegar al mínimo.

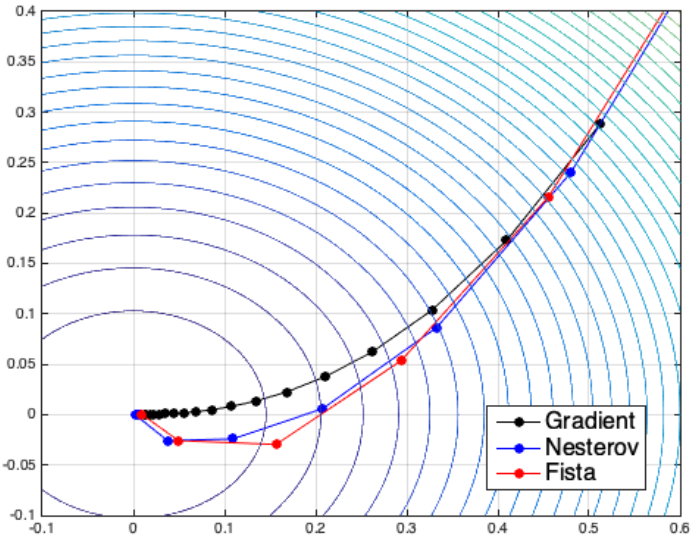


Figura 4.1: Ejemplo de convergencia del método de gradiente descendente clásico y los métodos de gradiente acelerado [16] y *FISTA* [6], método de gradiente proximal acelerado. Se puede observar que los métodos con aceleración toman menos pasos para llegar el mínimo de la función.

Por su parte, la Figura 4.2 muestra un acercamiento de los pasos de avance de FISTA [6] y el gradiente acelerado [16], para el mismo ejemplo previo de función convexa. Se puede observar el efecto de la aceleración en el paso intermedio de operación proximal o gradiente para ambos algoritmos.

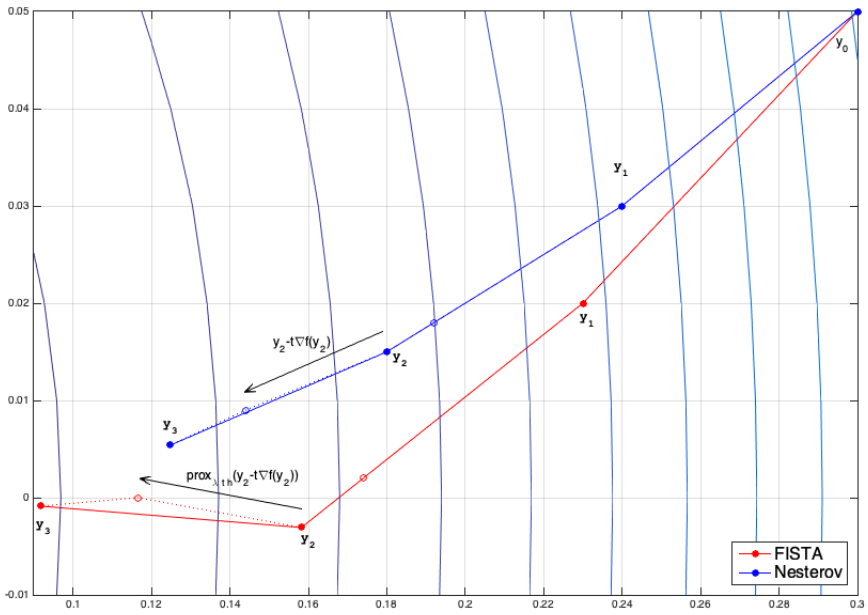


Figura 4.2: Ejemplo que muestra tres iteraciones de los algoritmos FISTA [6], y gradiente acelerado [16] aplicados a una función convexa $f(x, y) = x^2 + 2y^2 + 3$. Se puede observar en el paso de iteración 3, de ambos algoritmos, el efecto de aceleración aplicado a los pasos intermedios del gradiente y operador proximal para alcanzar el punto y_3. Asimismo, se advierte también que FISTA en la misma cantidad iteraciones ha avanzado más que el algoritmo de gradiente acelerado.

4.2.1. Tasa de Convergencia

Para un problema de optimización modelado por una función compuesta $F(x) = f(x) + h(x)$, en condiciones similares a las asumidas en (2.2), (3.13),

i) $f : \mathbb{R}^n \to \mathbb{R}$: convexa, propia, semicontinua inferior, diferenciable, con gradiente continuo constante L.

ii) $h : \mathbb{R}^n \to \mathbb{R}$: convexa y, propia.

iii) El conjunto óptimo del problema existe, y es no-vacío.

La tasa de convergencia para el método FISTA se demuestra [6, Teorema 4.4] es del orden $\mathcal{O}(\frac{1}{k^2})$

$$F(x_k) - F(x^*) \leq \frac{2\alpha L_f \|x_0 - x^*\|^2}{(k+1)^2}. \tag{4.6}$$

Los parámetros α y L_f vienen de la estrategia de *backtracking* (ver 2.1.2) para estimar el tamaño de paso óptimo en cada iteración. La idea de *backtracking*, señalada en [6] [5], busca encontrar en cada iteración un valor L_k, de manera que cumpla la siguiente desigualdad,

$$f(x_{k+1}) \leq f(x_k) + \langle \nabla f(x_k), x_{k+1} - x_k \rangle + \frac{L_k}{2}\|x_{k+1} - x_k\|^2.$$

Esta desigualdad asegura una condición suficiente de decrecimiento, ya sea para un tamaño de paso constante o variable por *backtracking*. Sólo, si cumplen las condiciones de convexidad de las funciones (mencionadas arriba), y el Lema de descenso (A.2). Los bordes inferior y superior de la variable L_k requerido en *backtracking* (donde L_f es la constante Lipschitz del gradiente de la función f) queda acotado por los siguientes límites [5, 10.4.2],

$$\beta L_f \leq L_k \leq \alpha L_f$$

donde,

$$\alpha = \begin{cases} 1, & \text{constante} \\ \max\{\eta, \frac{s}{L_f}\}, & \textit{backtracking} \end{cases} \qquad \beta = \begin{cases} 1, & \text{constante} \\ \frac{s}{L_f}, & \textit{backtracking} \end{cases}$$

con $\eta > 1$ y $s > 0$.

La comprobación de la tasa de convergencia se basa una desigualdad denominada *proximal-gradiente* [5, Teorema 10.16]

$$\begin{gathered} F(x) - F(T_L(y)) \leq \frac{L}{2}\|x - T_L(y)\|^2 - \frac{L}{2}\|x - y\|^2 + \ell_f(x, y), \\ \ell_f = f(x) - f(y) - \langle \nabla f(y), x - y \rangle \end{gathered} \tag{4.7}$$

donde, $T_L : dom\{f\} \to \mathbb{R}$ $(L > 0)$ es un operador proximal-gradiente definido:

$$T_L \equiv \texttt{prox}_{\frac{1}{L}h}(x - \frac{1}{L}\nabla f(x)). \tag{4.8}$$

Demostración. Sea $k \geq 1$, definiendo $x = t_k^{-1}x^* + (1 - t_k^{-1})x_k$, $y = y_k$, $L = L_k$, y sustituyendo esas definiciones en la desigualdad (4.7)

$$\begin{aligned} & F\Big(\underbrace{t_k^{-1}x^* + (1 - t_k^{-1})x_k}_{x}\Big) - F(\underbrace{x_{k+1}}_{T_L(y_k)}) \\ & \geq \frac{L_k}{2}\Big\|x_{k+1} - (t_k^{-1}x^* + (1 - t_k^{-1})x_k)\Big\|^2 - \frac{L_k}{2}\Big\|y_k - (t_k^{-1}x^* + (1 - t_k^{-1})x_k)\Big\|^2 \\ & + \ell_f(t_k^{-1}x^* + (1 - t_k^{-1})x_k, y_k) \\ & = \frac{L_k}{2t_k^2}\|t_k x_{k+1} - (x^* + (t_k - 1)x_k)\|^2 - \frac{L_k}{2t_k^2}\|t_k y_k - (x^* + (t_k - 1)x_k)\|^2 \end{aligned} \tag{4.9}$$

Observar que en la última desigualdad, usando la convexidad de f se puede dejar fuera ℓ_f.

Aplicando la convexidad de la función F,

$$F(t_k^{-1}x^* + (1-t_k^{-1})x_k) \leq t_k^{-1}F(x^*) + (1-t_k^{-1})F(x_k)$$

Definiendo, $v_n \equiv F(x_n) - F(x^*)$, para $n > 0$,

$$\begin{aligned} F(t_k^{-1}x^* + (1-t_k^{-1})x_k) - F(x_{k+1}) &\leq t_k^{-1}F(x^*) + (1-t_k^{-1})F(x_k) - F(x_{k+1}) + F(x^*) - F(x^*) \\ &= (1-t_k^{-1})(F(x_k) - F(x^*)) - (F(x_{k+1}) - F(x^*)) \\ &= (1-t_k^{-1})v_k - v_{k+1} \end{aligned} \tag{4.10}$$

usando la relación $y_k = x_k + \frac{t_{k-1}-1}{t_k}(x_k - x_{k-1})$ del algoritmo FISTA (4.5), en el último término de la desigualdad (4.9),

$$\begin{aligned} \|t_k y_k - (x^* + (t_k - 1)x_k)\|^2 &= \|t_k x_k + (t_{k-1} - 1)(x_k - x_{k-1}) - (x^* + (t_k - 1)x_k)\|^2 \\ &= \|t_{k-1}x_k - (x^* + (t_{k-1} - 1)x_{k-1})\|^2 \end{aligned} \tag{4.11}$$

Combinando (4.9), (4.10), y (4.11)

$$(t_k^2 - t_k)v_k - t_k^2 v_{k+1} \geq \frac{L_k}{2}\|\mathbf{u}_{k+1}\|^2 - \frac{L_k}{2}\|\mathbf{u}_k\|^2$$

donde, $\mathbf{u}_n = t_{n-1}x_n - (x^* + (t_{n-1} - 1)x_{n-1})$, para todo $n > 1$. Además se tiene $t_k^2 - t_k = t_{k-1}^2$ (usando la regla de actualización t_k de FISTA)

$$\frac{2}{L_k}t_{k-1}^2 v_k - \frac{2}{L_k}t_k^2 v_{k+1} \geq \|\mathbf{u}_{k+1}\|^2 - \|\mathbf{u}_k\|^2$$

$$\|\mathbf{u}_{k+1}\|^2 + \frac{2}{L_k}t_k^2 v_{k+1} \leq \|\mathbf{u}_k\|^2 + \frac{2}{L_{k-1}}t_{k-1}^2 v_k \ ,$$

entonces, para todo $k \geq 1$,

$$\|\mathbf{u}_k\|^2 + \frac{2}{L_{k-1}}t_{k-1}^2 v_k \leq \|\mathbf{u}_1\|^2 + \frac{2}{L_0}t_0^2 v_1 = \|x_1 - x^*\|^2 + \frac{2}{L_0}(F(x_1) - F(x^*)). \tag{4.12}$$

□

Reemplazando, $x = x^*$, $y = y_0$, y $L = L_0$, en la desigualdad de proximal-gradiente (4.7), y sacando el término ℓ_f por la convexidad de f, se tiene lo siguiente,

$$\frac{2}{L_0}(F(x^*) - F(x_1)) \geq \|x_1 - x^*\|^2 - \|y_0 - x^*\|^2$$

se sabe que al inicio de las iteraciones $y_0 = x_0$,

$$\|x_1 - x\|^2 + \frac{2}{L_0}(F(x_1) - F(x^*)) \leq \|x_0 - x^*\|^2.$$

Combinando esta última desigualdad con (4.12),

$$\frac{2}{L_{k-1}}t_{k-1}^2 v_k \leq \|\mathbf{u}_k\|^2 + \frac{2}{L_{k-1}}t_{k-1}^2 v_k \leq \|x_0 - x^*\|^2 \ .$$

Usando, el límite superior $L_{k-1} \leq \alpha L_f$, la definición de v_k y el Lema [5, Lema 10.33] que garantiza $t_k \geq \frac{k+2}{}2$,

$$F(x_k) - F^* \leq \frac{L_{k-1}\|x_0 - x^*\|^2}{2t_{k-1}^2} \leq \frac{2\alpha L_f \|x_0 - x^*\|^2}{(k+1)^2} \tag{4.13}$$

El número de iteraciones necesarias para lograr $F(\hat{x}) - F(x^*) \leq \epsilon$ con una tolerancia ϵ es al menos $\lceil \frac{C}{\sqrt{\epsilon}-1} \rceil$, con $C = \sqrt{2\alpha L(f)\|x_0 - x^*\|^2}$.

Capítulo 5

Métodos primer orden que incorporan coeficiente amortiguamiento Hessiano

Este trabajo explora algoritmos que derivan sus pasos de actualización iterativo, desde la discretización de dinámicas modeladas por una ecuación diferencial de segundo orden. La convergencia de los métodos discretizados, se basan en el comportamiento asintótico de las trayectorias de un modelo inercial de segundo orden, con coeficientes de amortiguamiento de viscosidad y fricción

$$\ddot{x}(t) + \gamma\dot{x}(t) + \beta\nabla^2 f(x(t))\dot{x}(t) + b(t)\nabla f(x(t)) = 0. \tag{5.1}$$

Se desprende de los trabajos [4] [2] que un adecuado ajuste de los parámetros de amortiguamiento γ y β de este modelo, proporciona métodos de primer orden con tasas de convergencia más rápidas. Estos algoritmos se denominan de primer orden debido que el término $\nabla^2 f(x(t))\dot{x}(t)$ es la derivada con respecto al tiempo de $\nabla f(x(t))$.

La discretización en el tiempo de este sistema dinámico proporciona un esquema general de actualización iterativo, que comparten todos los algoritmos de primer orden derivados de esta dinámica inercial con coeficientes de amortiguamiento

$$\text{Esquema-General} \begin{cases} y_k = x_k + \alpha_k(x_k + x_{k-1}) - \beta_k(\nabla f(x_k) - \nabla f(x_{k-1})) \\ x_{k+1} = T(y_k). \end{cases}$$

Los algoritmos de primer orden se basan en este esquema general y su grado de convergencia se determina por diferentes combinaciones de los parámetros de amortiguamiento α, β y los pasos de discretización. Asimismo la intervención de T en este esquema involucra operaciones de gradiente o de un operador proximal.

El coeficiente de amortiguamiento Hessiano $\beta\nabla^2 f(x(t))\dot{x}(t)$ que presenta este sistema dinámico general (5.1) proviene de un modelo inercial $(DIN)_{\gamma,\beta}$ (*Dinamical Inertial Newton*) propuesto en [1]

$$(\text{DIN})_{\gamma,\beta} \qquad \ddot{x} + \gamma\dot{x} + \beta\nabla^2 f(x(t))\dot{x}(t) + \nabla f(x(t)) = 0.$$

El coeficiente Hessiano en este sistema, se propone como un factor de amortiguamiento para las oscilaciones transversales producidas en el modelo de bola pesada (*Heavy Ball*) de

Polyak [19]. De igual modo, el trabajo de Su, Boyd, and Candes [23] (mencionado en el punto 2.4) propone el sistema $(\text{AVD})_\alpha$ (*Asymptotic Vanishing Damping*), donde el coeficiente de viscosidad γ fijo en el sistema DIN, se modifica por un coeficiente $\gamma = \frac{3}{t}$ que desvanece en el tiempo

$$(\text{AVD})_\alpha \quad \ddot{x} + \frac{\alpha}{t}\dot{x}(t) + \nabla f(x(t)) = 0.$$

Como se menciona en la sección 2.4, este modelo dinámico se entiende como la versión continua del modelo acelerado de Nesterov [16]. Se evidencia además que las trayectorias del modelo AVD convergen para cualquier $\alpha \geq 3$, y su tasa de convergencia es similar al algoritmo discreto $\mathcal{O}(\frac{1}{t^2})$.

El coeficiente α del modelo $(\text{AVD})_\alpha$ juega un rol importante en las tasas de convergencia de los métodos derivados de este sistema. En [3] se demuestra que para un valor de $\alpha > 3$ las trayectorias convergen débilmente a un mínimo de la función objetivo ($\arg\min f$). Asimismo, en Attouch et al. [4] se señala que para un valor de $\alpha > 3$ la tasa de convergencia óptima es $\mathcal{O}(\frac{1}{t^2})$. Igualmente, mencionan $\alpha \leq 3$ como un caso sub-crítico, con un valor óptimo de $\mathcal{O}(\frac{1}{t^{\frac{2\alpha}{3}}})$ en la tasa de convergencia. Finalmente, el caso para $\alpha = 3$ la convergencia de las trayectorias es un trabajo de investigación abierto.

El trabajo de Attouch, Peypouquet, and Redont [2] combina los dos coeficientes de amortiguamiento en el modelo $(\text{AVD-DIN})_{\alpha,\beta}$, una ecuación diferencial de segundo orden

$$(\text{DIN-AVD})_{\alpha,\beta} \quad \ddot{x} + \frac{\alpha}{t}\dot{x}(t) + \beta\nabla^2 f(x(t))\dot{x}(t) + \nabla f(x(t)) = 0.$$

Este trabajo menciona el efecto de ambos coeficientes en las trayectorias de convergencia de este modelo. Primeramente, el término $\frac{\alpha}{t}\dot{x}(t)$ proporciona un amortiguamiento lineal isotrópico con un parámetro de viscosidad $\frac{\alpha}{t}$ que desaparece asintóticamente. Observan que bajos condiciones moderadas de disipación de este coeficiente, cada solución de x la función f tiende a un mínimo, $f(x(t)) \to \min_{\mathcal{H}} f$ (con $\mathcal{H}$ un espacio de Hilbert y f función convexa dos veces diferenciable). Por otro lado, relacionan de forma natural el amortiguamiento geométrico del término $\beta\nabla^2 f(x(t))\dot{x}(t)$ con el sistema dinámico inercial de Newton (DIN). Se menciona que el sistema DIN (con coeficiente $\gamma > 0$) se puede presentar como un sistema de primer orden empleando sólo el gradiente de f. Esto además facilita una extensión natural para funciones propias, semicontínuas inferior, y convexas reemplazando el gradiente por el subdiferencial.

Se establece que el sistema (DIN-AVD) hereda las propiedades de convergencia de AVD y DIN. Pero además el modelo muestra similar propiedad de convergencia a [23] $\mathcal{O}(\frac{1}{t^2})$ en condiciones de $\alpha \geq 3$, $\beta > 0$ y $\arg\min f \neq \emptyset$. También se demuestra que para $\alpha > 3$ cada trayectoria converge débilmente y el límite está en $\arg\min f$, con orden de convergencia $f(x(t)) - \min_{\mathcal{H}} f = o(\frac{1}{t^2})$. Una propiedad significativa, el sistema (DIN-AVD) se puede ser generalizar a un caso convexo no-suave.

5.1. Algoritmos Proximales

Los algoritmos proximales introducidos en Attouch et al. [4] se derivan del sistema $(\text{DIN-AVD})_{\alpha,\beta,b}$

$$\ddot{x}(t) + \frac{\alpha}{t}\dot{x}(t) + \beta\nabla^2 f(x(t))\dot{x}(t) + b(t)\nabla f(x(t)) = 0. \tag{5.2}$$

i) **caso suave** La discretización implícita en tiempo de este modelo para un caso convexo, con tamaño de paso $h > 0$, produce la siguiente expresión discreta.

$$\frac{x_{k+1} - 2x_k + xk - 1}{h^2} + \frac{\alpha}{kh}\frac{x_{k+1} - x_k}{h} + \frac{\beta}{h}(\nabla f(x_{k+1}) - \nabla f(x_k)) + b_k\nabla f(x_{k+1}) = 0.$$

De esta ecuación discreta se elabora el algoritmo **IPAHD** (*Inertial Proximal Algorithm with Hessian Damping*). Con $s = h^2$, β_k y b_k variando con el valor de la iteración k.

$$\text{paso k: } \mu_k := \frac{k}{k+\alpha}(\beta_k\sqrt{s} + sb_k)$$

$$\text{IPAHD}\begin{cases} y_k &= x_k + (1 - \frac{\alpha}{k+\alpha})(x_k + x_{k-1}) - \beta_k\sqrt{s}(1 - \frac{\alpha}{k+\alpha})\nabla f(x_k) \\ x_{k+1} &= \text{prox}_{\mu_k f}(y_k). \end{cases}$$

ii) **caso no-suave** El caso no-suave se apoya en las propiedades de regularización *Moreau-Yosida* (3.5). Definiendo la función f_λ envolvente de *Moreau* con el índice $\lambda > 0$, se sabe que es una función convexa con una constante Lipschitz λ continua y que $\arg\min f_\lambda = \arg\min f$. Además como los mínimos se mantienen, se reemplaza f con f_λ en el algoritmo anterior del caso convexo.

$$\ddot{x}(t) + \frac{\alpha}{t}\dot{x}(t) + \beta\nabla^2 f_\lambda(x(t))\dot{x}(t) + b(t)\nabla f_\lambda(x(t)) = 0$$

Reemplazando f_λ en el algoritmo IPAHD

$$\begin{cases} y_k &= x_k + (1 - \frac{\alpha}{k+\alpha})(x_k + x_{k-1}) - \beta_k\sqrt{s}(1 - \frac{\alpha}{k+\alpha})\nabla f_\lambda(x_k) \\ x_{k+1} &= \text{prox}_{\frac{k}{k+\alpha}(\beta\sqrt{s}+sb_k)f_\lambda}(y_k) \end{cases}$$

Formulando f_λ en términos de f y su operador proximal

$$\begin{aligned} f_\lambda &= f(\text{prox}_{\lambda f}(x)) + \frac{1}{2\lambda}\left\|x - \text{prox}_{\lambda f}(x)\right\|^2 \\ \nabla f_\lambda(x) &= \frac{1}{\lambda}(x - \text{prox}_{\lambda f}(x)) \\ \text{prox}_{\theta f_\lambda}(x) &= \frac{\lambda}{\lambda + k}x + \frac{\theta}{\lambda + \theta}\text{prox}_{(\lambda+\theta)f}(x) \end{aligned}$$

Se obtiene el algoritmo proximal inercial para el caso no-suave

$$\text{paso k:}\ \ \mu_k := \frac{\lambda(k+\alpha)}{\lambda(k+\alpha)+k(\beta\sqrt{s}+sb_k)}$$

$$\text{IPAHD-NS}\begin{cases} y_k &= x_k + (1-\frac{\alpha}{k+\alpha})(x_k - x_{k-1}) + \frac{\beta\sqrt{s}}{\lambda}(1-\frac{\alpha}{k+\alpha})(x_k - \text{prox}_{\lambda f}(x_k)) \\ x_{k+1} &= \mu_k y_k + (1-\mu_k)\text{prox}_{\frac{\lambda}{\mu_k}}(y_k) \end{cases}$$

iii) **caso gradiente-inercial** La discretización en tiempo de $(\text{DIN-AVD})_{\alpha,\beta,1+\frac{\beta}{t}}$ para el caso de una función convexa con constante L Lipschitz. Con coeficientes de amortiguamiento $\alpha \geq 3$, $\beta > 0$.

$$\frac{1}{s}(x_{k+1} - 2x_k + x_{k-1}) + \frac{\alpha}{ks}(x_k - x_{k-1}) + \frac{\beta}{\sqrt{s}}(\nabla f(x_k) - \nabla f(x_{k-1})) + \frac{\beta}{k\sqrt{s}}\nabla f(x_{k-1}) + \nabla f(y_k) = 0$$

Para este caso de gradiente inercial se obtiene el siguiente esquema iterativo, con y_k similar al esquema acelerado de *Nesterov*

$$\text{paso k:}\ \ \alpha_k = 1 - \frac{\alpha}{k}$$

$$\text{IGAHD}\begin{cases} y_k &= x_k + \alpha_k(x_k - x_{k-1}) - \beta\sqrt{s}(\nabla f(x_k) - \nabla f(x_{k-1})) - \frac{\beta\sqrt{s}}{k}\nabla f(x_{k-1}) \\ x_{k+1} &= y_k - s\nabla f(y_k). \end{cases}$$

Capítulo 6

Resultados Experimentales

En los capítulos anteriores, se ha hecho una descripción abreviada de algunos métodos de primer orden. Los cuales se han clasificado, desde una perspectiva muy general, en algoritmos orientados a problemas convexos diferenciables y no diferenciables. Asimismo, se ha descrito el esquema de iteración utilizado, al igual que la convergencia al mínimo del problema. El propósito de este capítulo, es exponer los resultados numéricos para algunos de estos algoritmos, y comparar, de manera experimental la convergencia de cada uno de ellos con el análisis teórico. El análisis se apoya en un gráfico que ilustran las tasas de convergencias obtenidas, para cada algoritmo, cotejando las diferencias producidas durante los experimentos.

Para realizar la evaluación de desempeño se ha utilizado una imagen de verificación en blanco y negro, a la cual, se le ha aplicado un efecto de borrosidad (Figura 6.1). Cada algoritmo, aplica su esquema iterativo sobre la imagen borrosa y calcula la distancia de la imagen verdadera con la imagen producida en cada iteración. Idealmente, el algoritmo se detiene cuando el valor de error (o la distancia) en la imagen real y la recuperada es menor que cierto rango, en este caso 10^{-6}.

En la primera parte, se experimenta con una imagen borrosa generada artificialmente (procedimiento indicado en la sección B), y en la segunda etapa, se repite el mismo conjunto de experimentos incorporando ruido gaussiano a la imagen borrosa.

(a) imagen original. (b) imagen borrosa (c) imagen borrosa con ruido

Figura 6.1: Ejemplo de imagen utilizada en los experimentos

6.1. Presentación del problema

El proceso de recuperación de la imagen borrosa a su representación más natural se modela como un problema de optimización convexa en $\mathbb{R}^n$, donde n (ancho multiplicado por alto) es el tamaño de un vector que representa de la imagen a restaurar. La imagen borrosa se asume es producto de un operador lineal de difuminación aplicado sobre la imagen natural, asimismo el problema de reconstruir esta imagen se representa mediante un sistema lineal discreto.

$$\mathbf{A}x = b + w, \tag{6.1}$$

con $\mathbf{A} \in \mathbb{R}^{n\times n}$ el operador lineal, conocido, de difuminación que produce (o simula) el efecto de borrosidad en la imagen original no conocida, representada por el vector $x \in \mathbb{R}^n$, $b \in \mathbb{R}^n$ es la imagen borrosa generada artificialmente en este estudio, y w es el ruido o perturbación, que simula los efectos de contaminación durante el proceso de adqusición de la imagen.

A partir de este sistema lineal se construye la función objetivo como un problema de aproximación, minimizando una expresión de cuadrados mínimos, esto es, la medición de distancia producida en la norma ℓ_2 entre los vectores $\mathbf{Ax}$ y $\mathbf{b}$. Se combina además con un término de regularización en norma ℓ_1 usado para estabilizar la solución, y λ parámetro de regularización

$$\min_{x\in\mathbb{R}^n} \Big\{F(x) = \frac{1}{2}||\mathbf{A}x - b||_2^2 + \lambda|x|_1\Big\}. \tag{6.2}$$

De esta manera, la función objetivo que se intenta minimizar para recuperar la imagen original se presenta como el problema de tipo *LASSO* [13] y el paso iterativo de actualización para el método general de gradiente *proximal* (*ISTA*) definido en (3.11), para este problema en específico queda establecido de la siguiente forma

$$x_{k+1} = \mathcal{T}_{\lambda t_k}\Big(x_k - t_k\mathbf{A}^T(\mathbf{A}x_k - b)\Big), \tag{6.3}$$

involucrando la multiplicación de las matriz $\mathbf{A}$ con su transpuesta en cada paso de iteración.

6.2. Resultados Numéricos

Esta parte ilustra el desempeño obtenido, en relación con su tasa de convergencia, de los algoritmos **ISTA**, **FISTA** [6], **IGAHD** [4], **IPAHD-NS** [4]. En términos globales, los resultados logrados muestran que los algoritmos **IPAHD-NS**, **IGAHD** tienen un desempeño levemente mejor que **FISTA** y muy superior a **ISTA**.

La tabla 6.1 muestra un resumen de los valores de parámetros de configuración, para los distintos algoritmos empleados en los experimentos numéricos. La selección de estos parámetros, es consecuencia de varios ensayos con valores de configuración independientes que presentaran una mejora en el desempeño de los algoritmos. El procedimiento de selección se describe en la siguiente sección.

Tabla 6.1: Parámetros de configuración de los algoritmos.

Nombre	λ	h (tamaño de paso)	α	β	b
ISTA	1e−4	1	–	–	–
FISTA	1e−4	1	–	–	–
IGAHD	1e−4	1	3.01	0.5	1
IPAHD-NS	1e−4	$(\frac{1}{150})^2 = (0.0066)^2$	3.01	$0.8(2\sqrt{h}) = 0.0107$	1

6.2.1. Selección de parámetros

En una etapa previa se manipularon los píxeles de la imagen borrosa para reducirlos al rango de valores $[0, 1]$. Luego, usando de base los experimentos realizados en [6], se definió inicialmente el factor de regularización λ en el valor 2e−5. Sin embargo, en etapas posteriores este valor se fue modificando a $\lambda = 1\text{e}{-4}$, debido que el método **IPAHD-NS** presentó problemas de estabilidad en la convergencia. De este modo, se ajustó y escogió $\lambda = 1\text{e}{-4}$ como un parámetro regularización constante para la evaluación de todos los métodos, proporcionando además, una base de comparación común del desempeño de todos los algoritmos, al evaluar la función objetivo con el mismo término de regularización λ.

FISTA: El mecanismo de selección del tamaño de paso h en este método, se llevo a cabo, realizando ejecuciones del algoritmo con distintos valores del paso, y luego observando el mejor rendimiento de la tasa de convergencia. En particular $h = \{\frac{1}{L}, \frac{1}{2L}, \frac{1}{10L}\}$, donde $L = \lambda_{\text{máx}}(\mathbf{A}^T\mathbf{A})$ es la constante Lipschitz del gradiente de la función $\frac{1}{2}\|\mathbf{A}x - b\|^2$. La figura 6.2 muestra las curvas de tasa de convergencia de FISTA, para tres valores de tamaño de paso constante. Se puede notar que para un tamaño de paso entre $\frac{1}{L}$ y $\frac{1}{2L}$, el resultado es similar. De esta manera, para el propósito de los experimentos se escogió un tamaño de paso $h = \frac{1}{L}$ (es decir, para $L = 1$ se tiene que $h = 1$). Este valor de paso fue empleado en los algoritmos ISTA, FISTA, IGAHD. En el caso de IPAHD-NS, los valores de $\lambda = 2\text{e}{-5}$ y tamaño $h = 1$ resultaron en serios problemas de convergencia, por este motivo se disminuyo el valor de λ y se escogieron otros valores del tamaño de paso en este algoritmo (procedimiento descrito a continuación).

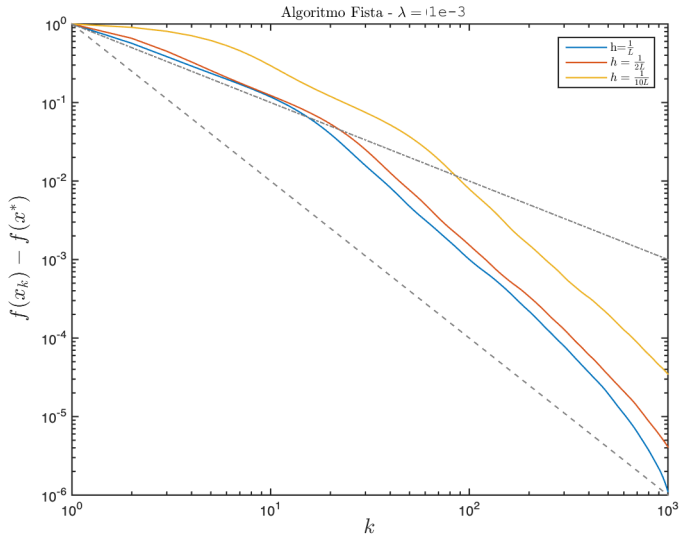
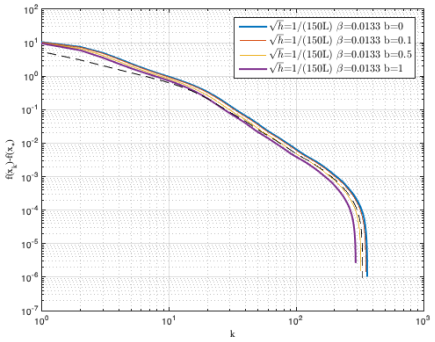
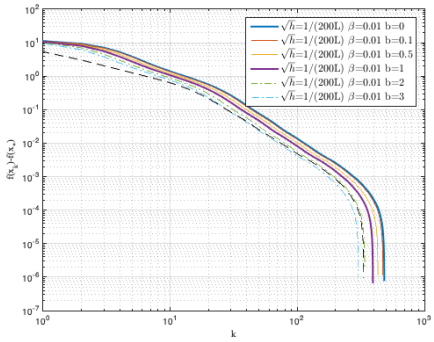


Figura 6.2: Resultado algoritmo FISTA con parámetro $\lambda = 1e-3$ para tres diferentes tamaños de paso $h = \{\frac{1}{L}, \frac{1}{2L}, \frac{1}{10L}\}$.

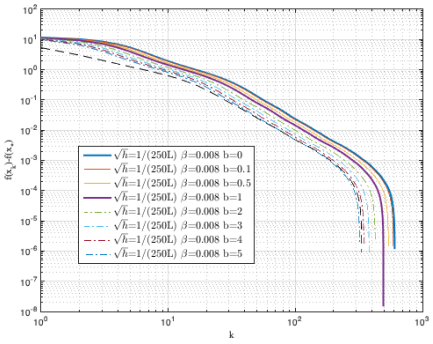
IPAHD-HD: La metodología empleada, para encontrar los parámetros que evidenciaran un mejor rendimiento de este algoritmo, consistió en realizar múltiples repeticiones, de diferentes combinaciones de el tamaño de paso h y los parámetros α, β, b, del modelo $(\text{DIN-AVD})_{\alpha,\beta,b}$ señalado en la ecuación (5.1). El valor α se mantuvo constante en 3.01. Por su parte, β fue modificado de acuerdo a la condición $\beta < 2\sqrt{h}$, y el parámetro b fue configurado con un valor constante b_0 y un término variable, basado en β, que tiende a cero con el avance de las iteraciones, $b = b_0 + \frac{\beta}{k}$. En una primera instancia, el procedimiento realizado fue para cada tamaño de paso h tomado desde el subconjunto de valores $\sqrt{h} = \{\frac{1}{50L}, \frac{1}{100L}, \frac{1}{150L}, \frac{1}{200L}, \frac{1}{250L}, \frac{1}{300L}, \frac{1}{350L}, \frac{1}{400L}\}$, se hicieron repeticiones con los valores de $b_0 = \{0, 0.1, 0.5, 1, 2, 3, 4, 5, 7, 9, 10\}$. La figura 6.3 presenta los resultados obtenidos para estas diferentes combinaciones de tamaño de paso $\sqrt{h}$ y la variable b_0. Para los tamaños de paso $\sqrt{h} = \{\frac{1}{150L}, \frac{1}{200L}, \frac{1}{250L}\}$ el máximo valor de la constante b_0 fue 1, 3 y 5 respectivamente, valores mayores el método resultaba inestable y no se lograba convergencia. Para tasas de convergencia menores ($\sqrt{h} = \{\frac{1}{300L}, \frac{1}{350L}, \frac{1}{400L}\}$) el algoritmo logró convergencias para todos los valores de b_0 explorados.



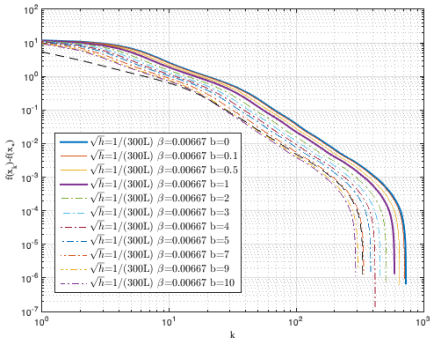
(a) Tamaño de paso $\sqrt{h} = \frac{1}{150L} = 0.0067$.



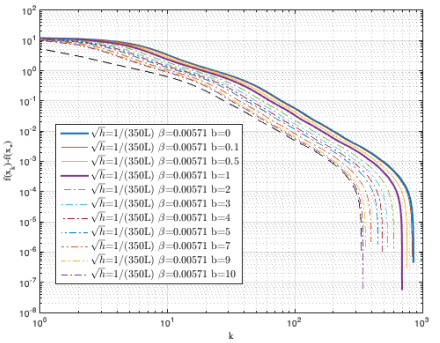
(b) Tamaño de paso $\sqrt{h} = \frac{1}{200L} = 0.005$.



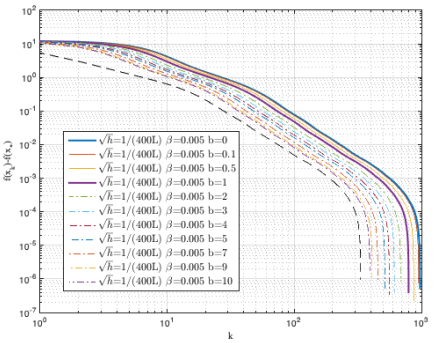
(c) Tamaño de paso $\sqrt{h} = \frac{1}{250L} = 0.004$.



(d) Tamaño de paso $\sqrt{h} = \frac{1}{300L} = 0.0033$.



(e) Tamaño de paso $\sqrt{h} = \frac{1}{350L} = 0.0029$.



(f) Tamaño de paso $\sqrt{h} = \frac{1}{400} = 0.0025$.

Figura 6.3: Resultados obtenidos con diferentes combinaciones de tamaño de paso $\sqrt{h}$ y constante b_0. Los gráficos indican las tasas de convergencia obtenidas para un valor fijo de tamaño de paso ($\sqrt{h}$) y distintos valores de la constante b_0.

Las múltiples tasas de convergencias mostradas en la Figura 6.3, indican que para un tamaño de paso $\sqrt{h} = 6.7\text{e}{-3}$ se obtiene una tasa de convergencia del orden de 10^{-6}, cerca de la iteración 300. Por otra parte, un tamaño de paso menor $\sqrt{h} = 2.5\text{e}{-3}$ consigue convergencia entre el rango de iteración $[400, 1000]$, dependiendo del valor de la constante b_0. En este caso, si el valor de $b_0 = 10$ la convergencia del método ocurre en la iteración 400 y para $b_0 = 0$ en la iteración 1000. A partir de la observación, de estas múltiples repeticiones, el valor escogido de tamaño de paso corresponde a $\sqrt{h} = \frac{1}{150L} = 6.7\text{e}{-3}$, y la constante $b_0 = 1$, donde consigue convergencia en la iteración 300.

Con ambos valores de $\sqrt{h}$ y b establecidos, se realizó un segundo ensayo modificando sólo el valor de β en cada repetición, respetando la condición $\beta < 2\sqrt{h}$. Se utilizó un conjunto de 10 valores de $\beta = \{0.1(2\sqrt{h}), ..., 1(2\sqrt{h})\}$. La Figura 6.4 presenta el resultado obtenido, y se observa que para $\beta = 0.8(2\sqrt{h}) = 0.0107$ se consigue un convergencia de 10^{-8}.

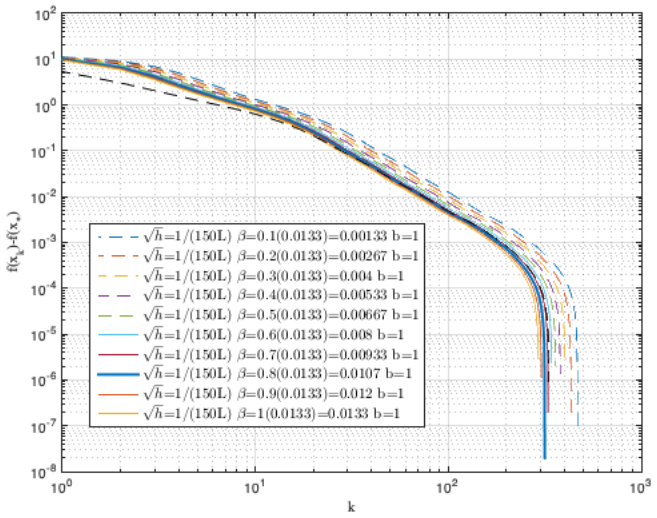


Figura 6.4: Cuadro comparativo de las tasas de convergencia obtenidas para el algoritmo IPAHD-NS con una tamaño de paso $\sqrt{h} = 6.7\text{e}{-3}$, constante $b_0 = 1$ y diferentes variaciones de β.

IGAHD: Para este algoritmo, se mantuvieron los mismos valores α y el tamaño de paso de FISTA, esto es $\sqrt{h} = \frac{1}{L}$. La variable b, por definición, en este método es 1. El parámetro β fue seleccionado utilizando el mismo procedimiento anterior. Mediante una serie de ejecuciones con distintos valores de β, pero en este caso, se generó un conjunto de valores de $\beta = \{0, 0.1, 0.5, 1.5, 2\}$. La Figura 6.5 muestra que configurando $\beta = 0.5$ se logra un error de $10^{-}8$ en la tasa de convergencia.

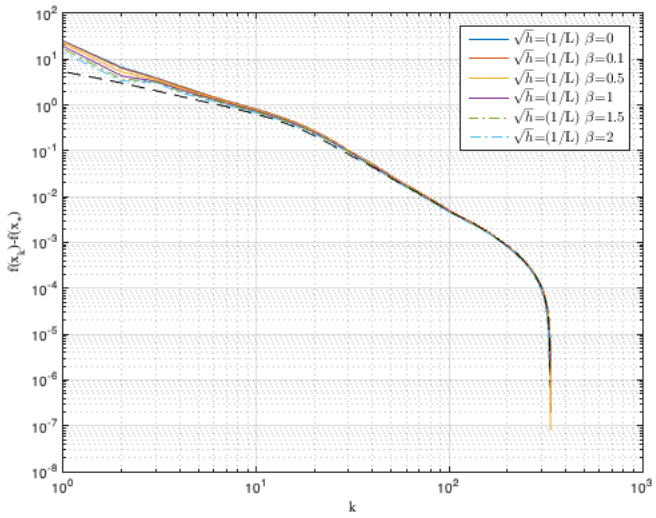


Figura 6.5: Tasas de convergencia obtenidas con el algoritmo IGAHD con un valor 1 de tamaño de paso, y diferentes valores de β. Se puede notar que con $\beta = 0.5$ se consigue un error del orden de 10^{-8}.

Con respecto a los tiempos de ejecución, cada algoritmo realizó una corrida de 1000 iteraciones, y se registro su duración en la iteración 100 y final. La siguiente tabla, indica los tiempos promedios de ejecución de los algoritmos.

Tabla 6.2: Registros de los tiempos de ejecución ([s]) para la iteración 100 y 1000.

Iteración	ISTA	FISTA	IGAHD	IPAHD
100	1.85	1.86	4.28	3.02
1000	18.3	18.5	43.7	30.9

6.2.2. Imagen borrosa sin ruido

La Figura 6.6 muestra las curvas de tasa de convergencia, obtenidas por cada algoritmo, después de 1000 iteraciones. En primer lugar, se puede observar que el método **ISTA** presenta el peor desempeño, comparado con los otros algoritmos. Esto se evidencia, puesto que la curva de convergencia está por sobre las curvas de los otros algoritmos. También, se puede comprobar que este método presenta una tasa de convergencia del orden $\mathcal{O}(\frac{1}{k})$, al tener la curva de convergencia una pendiente similar a la recta $\frac{1}{k}$, representada por la linea superior segmentada de color gris. En contraposición, los métodos **FISTA**, **IGAHD**, **IPAHD-NS** presentan una tasa de convergencia del orden de $\mathcal{O}(\frac{1}{k^2})$. Esto se constata, al observar que las curvas a partir de la iteración 10, exhiben una de tasa de convergencia similar a la recta $\frac{1}{k^2}$ (linea segmentada gris inferior), hasta un poco después de la iteración 300 estos métodos tienen una respuesta muy abrupta y consiguen valores mayores de 10^{-6} en convergencia.

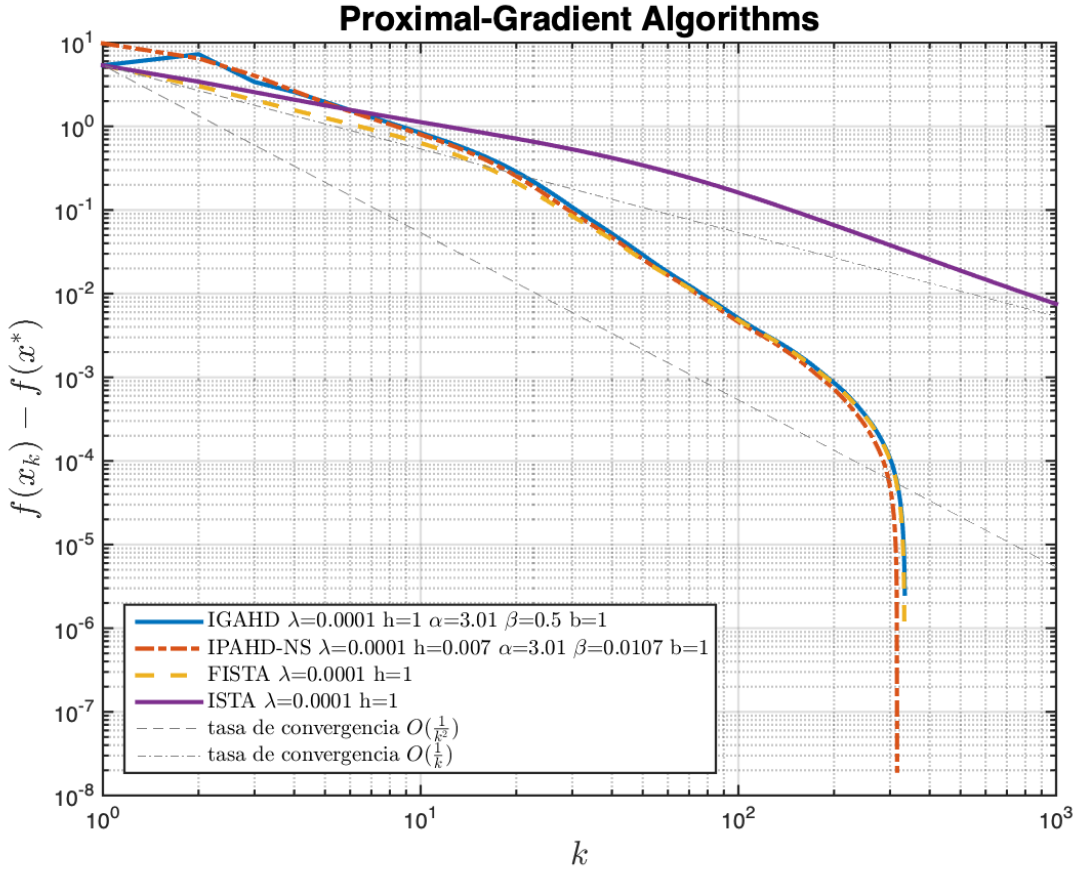


Figura 6.6: Curvas tasa de convergencias obtenidas para los métodos **ISTA**, **FISTA**, **IGAHD**, **IPAHD-NS**. Configurados con los parámetros mencionados en la tabla 6.1.

Con respecto al desempeño individual, se puede notar, al observar la Figura 6.7 normalizada, que el método **IPAHD-NS** presenta un mejor rendimiento, debido su curva de convergencia se localiza permanente por debajo de las curvas de los otros métodos. El siguiente algoritmo con mejor desempeño es **IGAHD** seguido por **FISTA**, ambas curvas presentan un desempeño muy parecido. Se podría mencionar que **FISTA** tiene un desempeño un poco inferior debido que su curva está por sobre los otros dos métodos. No obstante, independiente del rendimiento de las tasas de convergencia, se observa que el método **IPAHD-NS** consigue un mejor error de convergencia, aproximadamente 1×10^{-9} en la iteración 316, asimismo el método **IGAHD** logra en la iteración 334 un error de 1×10^{-7}, y muy similar **FISTA** pero con un error de 3×10^{-7} en la iteración 334.

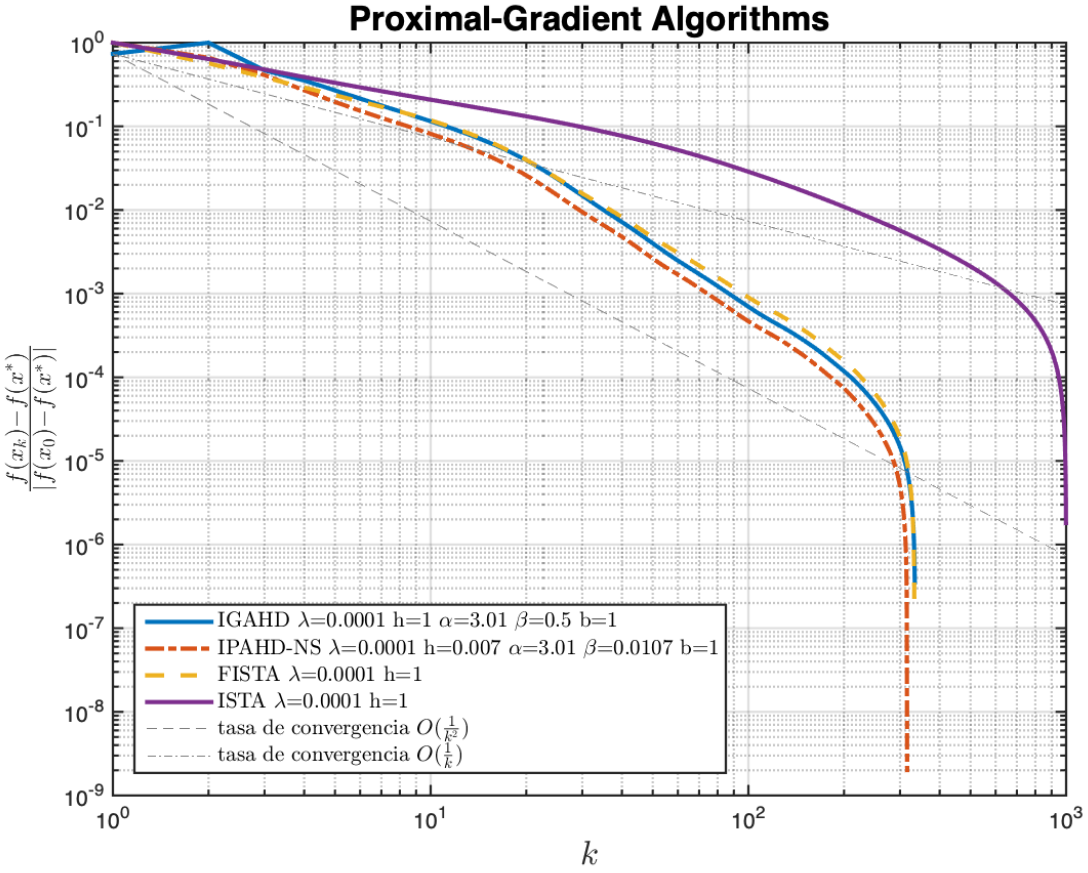


Figura 6.7: Curvas tasa de convergencias normalizado. El valor de cada punto se divide para la diferencia entre el máximo y mínimo de cada serie. Después de efectuada la normalización, el máximo valor de cada curva es 1.

La Figura (6.8) muestra la respuesta de los algoritmos, empleando un punto de inicio aleatorio (los puntos del vector de inicio se toman de una distribución uniforme $[0, 1]$). Se observa en este gráfico, que el algoritmo **IPAHD-NS** tiene un mejor desempeño en la recta $\frac{1}{k^2}$, y los métodos **IGAHD**, **FISTA** presentan similar respuesta pero más cercano a la recta $\frac{1}{k}$.

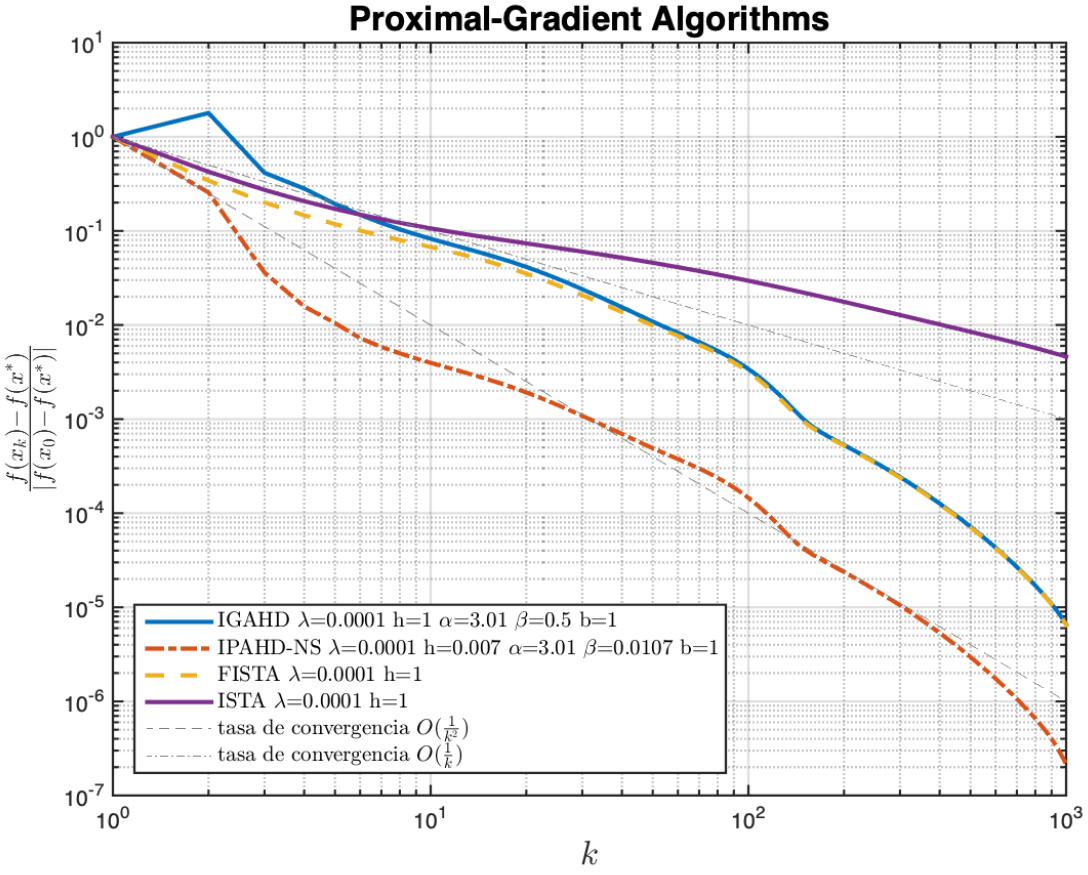


Figura 6.8: Curvas de respuestas de tasa de convergencias normalizado, utilizando un punto de inicio aleatorio.

La Figura (6.10) presenta un cuadro consolidado con las imágenes recuperadas, en la iteración 100 y cerca de 330 (1000 para ISTA), logradas por cada algoritmo. Las imágenes incluyen también el valor de la función en ambas iteraciones, el valor óptimo es $F(x^*) = 3.05127960$. Los valores de la función F de ISTA, FISTA, IGAHD, IPAHD-NS en la iteración donde se obtiene la menor diferencia (en torno a 330 para todos excepto ISTA) fue de 3.05881726, 3.05128053, 3.05127968, 3.05127962 respectivamente. Se puede notar que el valor de la función F para **IPAHD-NS** es igual al óptimo hasta el decimal 7.

6.2.3. Imagen borrosa con ruido

Esta parte, agrega ruido esparciendo puntos en forma aleatoria sobre toda la imagen. Este es un tipo de ruido de tipo impulsivo denominado 'Sal y Piminenta' (del inglés de *salt-and-pepper*), el cual puede ser provocado durante la transmisión de datos, o errores en la conversión análoga a digital. Se presenta en forma de puntos negros y blancos esparcidos de manera aleatoria, deteriorando la calidad de la imagen. Este tipo de ruido se simula en *Matlab* con el comando `imnoise`, como se indica a continuación.

```
img = im2double(imread(filename));
I = imnoise(img,'salt & pepper', 0.02);
```

La Figura 6.9 muestra las curvas de convergencias resultantes aplicando los diferentes algoritmos sobre esta imagen borrosa y con ruido. Se puede observar, que el efecto del ruido modifica la convergencia de los métodos, se necesitan, en este caso, un número cercano a 600 iteraciones para lograr un error similar al conseguido con la imagen sin ruido (300 iteraciones). Esto es, un error de 1×10^{-6} para **ISTA** in la iteración 620, un valor 4×10^{-4} para el método **IGAHD** en la iteración 623, y un error 1×10^{-7} para **IPAHD-NS** en la iteración 583. Todos los métodos presentan una tasa de convergencia levemente más lenta, que el ejemplo anterior. Esto es, los métodos toman más tiempo en recuperar la imagen verdadera, se puede notar que las curvas están más cerca de la línea gris segmentada superior (la línea $\mathcal{O}(\frac{1}{k})$) que la linea gris inferior. El efecto del ruido se puede también apreciar en las imágenes recuperadas señaladas en el Figura 6.11, donde se puede ver que ISTA sigue siendo el método con desempeño inferior comparado con los otros algoritmos.

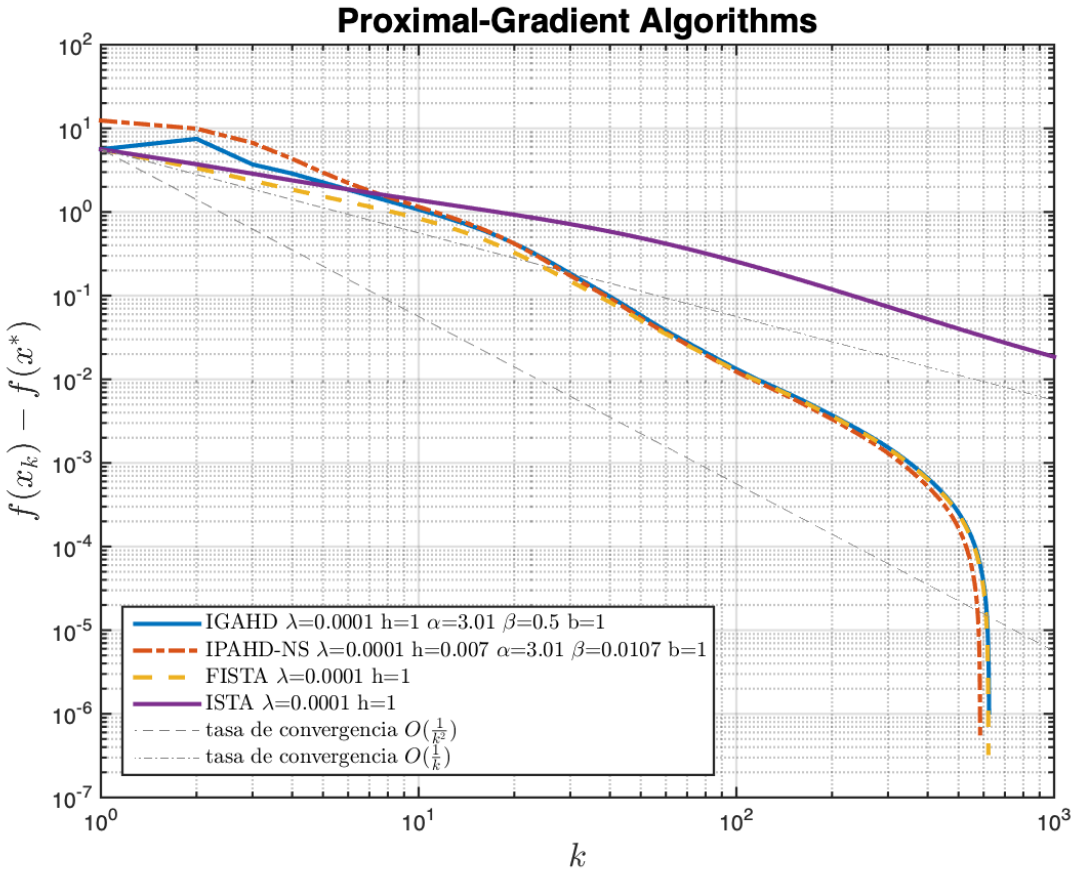


Figura 6.9: Curvas tasa de convergencias obtenidas para los métodos **ISTA**, **FISTA**, **IGAHD**, **IPAHD-NS** aplicados sobre la imagen borrosa con ruido gaussiano.

(a) **ISTA**: $F_{100} = 3.21255415$.

(b) **ISTA**: $F_{1000} = 3.05881726$

(c) **FISTA**: $F_{100} = 3.05611428$

(d) **FISTA**: $F_{333} = 3.05128053$

(e) **IGAHD**: $F_{100} = 3.05624258$

(f) **IGAHD**: $F_{334} = 3.05127968$

(g) **IPAHD-NS**: $F_{100} = 3.05587034$

(h) **IPAHD-NS**: $F_{316} = 3.05127962$

Figura 6.10: Imágenes resultantes de los algoritmos **ISTA**, **FISTA**, **IGAHD**, **IPADH** aplicados sobre la imagen borrosa sin ruido. Para cada algoritmo se muestra el resultado obtenido en la repetición 100 y un valor poco más allá de la iteración 300 (cerca de la diferencia mínima), y la iteración 1000 para ISTA. También se entrega el valor de la función F en esas dos iteraciones.

(a) **ISTA**: $F_{100} = 3.0666536$.

(b) **ISTA**: $F_{1000} = 3.0688317$

(c) **FISTA**: $F_{100} = 5.41103$.

(d) **FISTA**: $F_{1000} = 3.0567177$

(e) **IGAHD**: $F_{100} = 3.0618960$.

(f) **IGAHD**: $F_{1000} = 3.0514366$

(g) **IPAHD-NS**: $F_{100} = 3.0665649$.

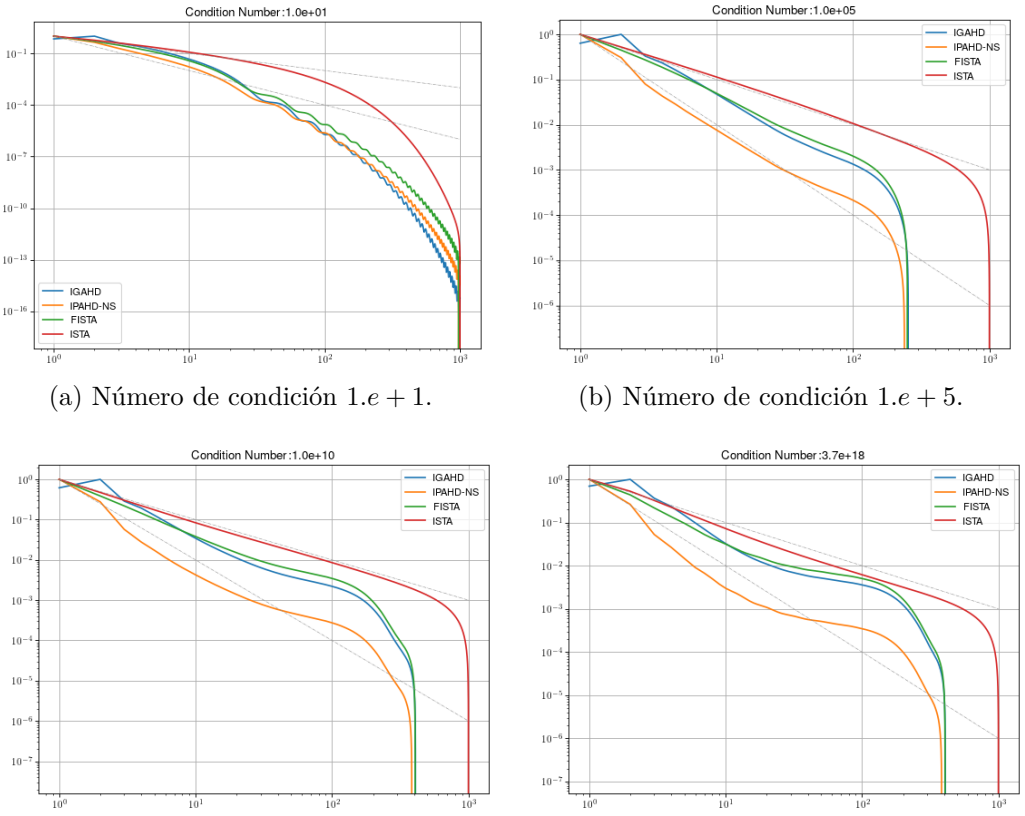
(h) **IPAHD-NS**: $F_{1000} = 3.0566963$

Figura 6.11: Resultado obtenido los algoritmos **ISTA**, **FISTA**, **IGAHD**, **IPADH** después de procesar una imagen borrosa con ruido gaussiano. Se muestra resultado para el valor de iteración 100 y 1000.

6.2.4. Experimentación con un tamaño reducido Matriz A

En la sección anterior se puede observar que el comportamiento de las curvas de convergencia de los algoritmos estudiados, excluyendo ISTA, tienen un rendimiento comparable. Aparentemente el número de condición de la matriz A, tiene influencia en el rendimiento general de las tasas de convergencia. Una estimación del número de condición de la matriz A (comando Matlab `cond(A)`) proporciona un valor de 1.3e+9. Este apartado describe el ensayo realizado con una matriz de tamaño reducido, 100×100, para observar el efecto del número de condición en el comportamiento de las curvas de rendimiento de estos algoritmos.

La matriz A de los ensayos fue generada aleatoriamente con valores uniformes entre $[0, 1]$, y los valores diagonales se manipularon para conseguir diferentes valores de número de condición. También el vector $\mathbf{x}$ (tamaño n) se generó aletoriamente de acuerdo a una normal $\mathcal{N}(0, 1)$. Se realizaron 100 repeticiones para cada uno de los siguientes valores del número de condición $\{1.\text{e+}1, 1.\text{e+}5, 1.\text{e+}10, 1.\text{e+}20\}$. La Figura (6.12) muestra las curvas obtenidas de cada uno de los números de condición señalados, y cada curva corresponde a la media de las 100 repeticiones. Se puede notar que, todos los algoritmos tienen un comportamiento muy similar para un valor de número de condición menor cercano a 1, y que el algoritmo IPAHD presenta una mejor tasa de convergencia para un número de condición mayor a 5.



(a) Número de condición $1.e+1$.

(b) Número de condición $1.e+5$.

(c) Número de condición $1.e+10$.

(d) Número de condición $3.7e+18$.

Figura 6.12: Curvas de rendimiento para diferentes número de condición de la matriz A.

Capítulo 7

Conclusiones

Este trabajo de tesis, se ha basado en experimentar con métodos numéricos denominados de primer orden, que pueden ser aplicados a problemas de optimización definidos (o compuestos) por dos funciones convexas; una suave y la otra no-suave. Estos métodos se fundamentan en un modelo de gradiente proximal, combinando un esquema de paso directo que utiliza la información de gradiente de la función suave, y un esquema indirecto, asociado a un operador de proximidad que se vincula con la función no-suave de esta función global compuesta. Las propiedades del operador de proximidad, en conjunto con las propiedades de convexidad de las funciones, sirven de base para asegurar la convergencia de las secuencia, de valores obtenido mediante este tipo de esquemas iterativos. En términos prácticos, se ha podido comprobar en la etapa de los experimentos, que la sucesión de valores generada por estos métodos de primer orden, convergen hacia al mínimo de la función que modela el problema. Esto ocurre siempre, independiente del punto inicial del proceso iterativo, y con una adecuada configuración de parámetros de los algoritmos.

Los ensayos también demostraron que los algoritmos, en general, consiguen una tasa de convergencia del orden $\mathcal{O}(\frac{1}{k^2})$ después de las 300 repeticiones, y el doble, en el caso de una imagen borrosa y afectada con ruido aleatorio. Sin embargo, durante los primeros ejercicios las tasas de convergencia, de estos métodos de primer orden (con amortiguamiento Hessiano), no se acercaban al resultado esperado y su rendimiento era inferior, o al menos similar, al algoritmo con aceleración como FISTA. Se esperaba, en un principio, obtener una sucesión de valores con una tasa de convergencia muy superior a los métodos clásicos con aceleración, comparables a las secuencias óptimas generadas por el método acelerado de *Nesterov*, en la versión que se aplica a problemas convexos y diferenciable. Pero, los coeficientes de amortiguamiento, al parecer, no producían el efecto de mejora en el rendimiento general, que se proponía en el enunciado original para este tipo de métodos. Se necesitaron finalmente varios ensayos de prueba y error, en la configuración de parámetros de los algoritmos (especialmente en IPAHD-NS), para lograr mejoras en las tasas de convergencia, o al menos se acercaran al orden $\mathcal{O}(\frac{1}{k^2})$ como suponía el análisis teórico de estos métodos.

Escoger un adecuado valor de tamaño de paso, en particular para el método IPAHD-NS, precisó experimentar con diferentes valores proporcionales a la constante de Lipschitz de la matriz A de borrosidad. Una primera observación muestra, que en la medida que el tamaño de paso es menor, más tiempo demoran los algoritmos en lograr convergencia, debido que se requiere una mayor cantidad de pasos para llegar al mínimo. Esto se advierte en el com-

portamiento de las múltiples curvas de la tasa de convergencia, generadas durante el proceso de selección de parámetros. Además, durante el procedimiento de selección de parámetros, se pudo observar que una adecuada combinación de los parámetros tamaño de paso h, β y la constante b, contribuyen a mejorar la convergencia de la secuencia generada al óptimo del problema. Por otra parte, la introducción de ruido en la imagen de verificación, como era de esperarse, contribuyó a degradar la tasa de convergencia de igual manera sobre los tres métodos. Estos preservaron el mismo comportamiento (o la misma tendencia) que el exhibido en el proceso de recuperación de la imagen sin ruido. Es decir, la incorporación de ruido afecto de igual manera a los tres métodos, la disminución del rendimiento se refleja en una extensión de las curvas de convergencia.

En relación al resultado de las tasas de convergencias, que presentan las curvas normalizadas (en particular con el ejemplo de la imagen sin ruido), se observa que el método IGAHD queda permanentemente debajo de los otros dos algoritmos, sugiriendo un mejor rendimiento que los otros métodos. Pero, se como se menciona en el párrafo anterior, todos los métodos obtienen su mejor valor de convergencia (en términos de la diferencia entre la evaluación de la función objetivo con algún punto y el valor óptimo), poco después de la repetición 300. Del mismo modo, se puede advertir desde esta figura con las curvas de convergencias logarítmicas y normalizadas, que IGAHD presenta un avance importante en las primeras iteraciones, pero muy pronto su cadencia de avance es equivalente a los demás, es decir, las pendientes de las tasas de convergencia siguen un patrón de descenso muy similar, a partir de la iteración 20. De lo cual se puede extraer, que en general, los métodos tienen un patrón de desempeño parecido y el elemento de diferenciación entre uno y otro, es el aporte de los factores de amortiguamiento que incorporan estos algoritmos.

Asimismo, se observa que IGAHD tiene un funcionamiento muy parecido a FISTA. La principal diferencia de este algoritmo, es que incluye en su esquema iterativo el factor de amortiguamiento Hessiano que pondera una combinación de gradientes del punto en curso con el anterior, y podría emular FISTA sólo configurando el coeficiente β en cero. Esta característica permite hacer una comparación entre ambos, y se puede destacar a partir de las curvas, que ambos métodos logran convergencia prácticamente en la mismo número de iteración y se evidencia que el aporte del coeficiente Hessiano es más significativo en esta etapa final del avance. Produciendo una mejora de la tasa de convergencia, en este punto de iteración, en aproximadamente un orden de magnitud con respecto a FISTA (8×10^{-8} versus 9×10^{-7}). Demostrando de esta manera, el aporte del factor de amortiguamiento Hessiano en la convergencia de este método. Como contrapartida, el tiempo de ejecución de este algoritmo es mayor que todos los otros métodos, esto se explica por las operaciones de gradiente sobre el punto actual y anterior en la etapa intermedia, así como la operación de gradiente sobre el punto final obtenido.

Por su parte, el desempeño individual del método IPAHD-NS (la versión no-suave del esquema proximal con amortiguamiento Hessiano), aplicado en un problema simple, como la recuperación de la imagen borrosa sin ruido, verifica que consigue el menor valor de error de convergencia ($F(x_k) - F(x_*)$), comparado con los otros algoritmos de primer orden sometidos a evaluación. Aún cuando, la curva de convergencia registrada en ambos experimentos es levemente superior a IGAHD y FISTA (revelando una tasa inferior) en las primeras repeticiones, este método aproxima al óptimo antes que los otros métodos y con una buena cifra de

error (10^{-8} y más próximo a 10^{-9}). No obstante, el tiempo de ejecución promedio también es mayor, 1/3 superior que FISTA, y aproximadamente 1/3 inferior a IGAHD. El mayor tiempo de procesamiento, es producto del cómputo de los términos adicionales, del factor de amortiguamiento Hessiano y las dos operaciones proximales empleadas en los pasos del algoritmo. Esto último, demuestra que el tiempo promedio de procesamiento de la operación proximal es menor que la operación de gradiente, debido que el método IPAHD-NS se apoya en operaciones proximales a diferencia de IGAHD que se basa principalmente en el operador de gradiente.

Por otro lado, este método requirió de muchos ensayos preliminares para conseguir un adecuado ajuste, de los parámetros de amortiguamiento y de esta manera mejorar el rendimiento de la tasa de convergencia. De la misma forma, se pudo observar que la estabilidad del método se relaciona con una precisa selección del tamaño de paso y la combinación de los parámetros β y b. El valor del tamaño de paso utilizado fue inferior en dos ordenes de magnitud comparado con IGAHD y FISTA. Cuando se usaron tamaños de tasa del mismo orden que FISTA ($\sim \frac{1}{L}$), la secuencia de valores generadas abandonaba el régimen decreciente, y cada nuevo punto obtenido aumentaba el valor de la función objetivo, resultando en que el algoritmo no consiguiera la convergencia al óptimo. Se logró establecer en los ensayos, que valores del orden de 10^{-3} del tamaño de paso (o menores), el algoritmo generaba una secuencia decreciente de puntos y por consiguiente aseguraba convergencia al óptimo. Un tamaño de paso reducido, resulta como consecuencia en β también pequeño (por la condición $2\sqrt{h}$), y aparentemente, el parámetro β tendría poca incidencia en el desempeño global del algoritmo. Osea, el impacto del Hessiano en el rendimiento del algoritmo no sería muy significativo. Se debe considerar que tamaño de pasos muy menores, requiere más repeticiones del algoritmo para lograr algún error de convergencia previamente definido.

En igual forma, e independiente de los resultados experimentales, el algoritmo FISTA presenta una buena estabilidad y respuesta en la obtención del mínimo de la función. Logra obtener una buena reconstrucción de la imagen borrosa, y además el tiempo de ejecución es relativamente menor comparado con los otros métodos inerciales, con factor de coeficiente Hessiano. La implementación del método no es muy compleja y se puede comprender como la versión proximal del gradiente acelerado de *Nesterov*. La influencia del término de momento (la diferencia del punto actual con el anterior) en este método, inicialmente no es importante, pero cobra relevancia en el avance de la evolución del algoritmo.

En resumen, se puede afirmar que los algoritmos estudiados y aplicados en la recuperación de la imagen borrosa se aproximan al óptimo del problema y en general, presentan entre ellos una tasa de convergencia muy similar. La inclusión de los coeficientes de amortiguamiento Hessiano, no producen (al menos en esta clase de problemas) una mejora sustancial en las tasas de convergencia, o un cambio evidente en las pendientes (comparado con los métodos con aceleración). Pero, sí refuerzan la convergencia hacia el óptimo en la etapa final de las repeticiones de estos algoritmos. Sin embargo, esta mejora en la convergencia final se contrarresta con el incremento en los tiempos de ejecución. En un problema de optimización relativamente de baja dimensión, los tiempos de ejecución tienen poca influencia en la elección del algoritmo más adecuado, y se puede aplicar el algoritmo con mejor relación de convergencia. Pero, en problemas de dimensión muy alta el tiempo de ejecución es un factor que cobra más relevancia, lo que hace que FISTA sea un algoritmo con mejores características para ser usado en este tipo de problemas.

Finalmente, como trabajo futuro, se puede implementar el esquema de reinicio propuesto en el trabajo [23], con la idea de mantener una alta velocidad en el proceso de descenso, esto significa que después de un cierto número de iteraciones, y condiciones que la velocidad disminuye, los parámetros del algoritmo se reinician con los valores de arranque. Por otro lado, se puede abordar el problema de configuración de parámetros óptimos, que mejoran las tasas de convergencia, y en particular para IPAHD-NS, con alguna una metodología que facilite la búsqueda de valores apropiados de los coeficientes, de modo que se ajusten a los distintos tipos de problemas. Es importante mencionar, que se necesitó de bastante ensayos (fuerza bruta) para lograr unos parámetros adecuados que mejoraran los rendimientos. Se debe considerar, que esta búsqueda empírica de ajustar los parámetros, puede funcionar apropiadamente para el tipo de problemas de baja dimensión utilizado en este trabajo. Pero, en problemas de mayor dimensión o complejidad, este tipo de ajuste puede resultar en una operación poco práctica y más compleja que la evaluación propia de los algoritmos. Del mismo modo, la experimentación realizada fue basada sólo en valores constantes de los parámetros, excepto b que disminuía con el avance de las iteraciones. Se podría explorar, el uso de funciones a valores que se adapten con el avance de las repeticiones de los algoritmos, de modo que los coeficientes de amortiguamiento tenga un mayor efecto en el desempeño global de estos algoritmos. Por otro lado, se debe asegurar también, que el tamaño de paso encontrado sea el más adecuado, un tamaño de paso mas grande claramente disminuiría el numero de iteraciones para lograr el óptimo, pero no es evidente que el tamaño de paso encontrado en los experimentos fuera el más óptimo para este tipo de problemas. Se puede analizar también, mejorar el número de condición de la matriz de borrosidad y experimentar con una matriz equivalente que mejore la condición del problema. Un aspecto interesante a explorar en estos métodos de primer orden, desde el punto de vista de la implementación computacional, consiste en la posibilidad de paralelizar los cómputos del operador proximal. Para el caso, donde el problema está regularizado por el norma ℓ_1 el operador proximal resulta en una operación de recorte denominada *Soft-Thresholding*. Una de las propiedades de esta operación, es la posibilidad de aplicar el operador a nivel de componente de un vector. Esto, posibilita en problemas de procesamiento de imágenes con un número muy grande de pixeles, aplicar técnicas de procesamiento paralelo, lo que podría reducir la complejidad computacional del problema.

Anexo A

Lemas y Proposiciones usadas en optimización convexa

Lema A.1 *Sea una función $f : \mathbb{R}^n \to \mathbb{R}$ continua y diferenciable, suponiendo que f es convexa, entonces*

$$f(\bar{x}) + \langle \nabla f(\bar{x}), x - \bar{x} \rangle \leq f(x) \;\; \textit{para todo} \;\; x, \bar{x} \in \mathbb{R}^n \tag{A.1}$$

Demostración. Fijando $x, \bar{x} \in \mathbb{R}^n$ y para cualquier $t \in (0,1)$, se tiene que

$$\begin{array}{rcl} f(\bar{x} + t(x - \bar{x})) & = & f(tx + (1-t)\bar{x}) \\ & \leq & tf(x) + (1-t)f(\bar{x}) \\ & = & f(\bar{x}) + t(f(x) - f(\bar{x})) \\ f(\bar{x} + t(x - \bar{x})) - f(\bar{x}) & \leq & t(f(x) - f(\bar{x})) \;\; \text{para todo } t \in (0,1) \\ \dfrac{f(\bar{x} + t(x - \bar{x})) - f(\bar{x})}{t} & \leq & f(x) - f(\bar{x}) \end{array}$$

Como $f(x) \in C^1$ y haciendo que $t \to 0^+$

$$\langle \nabla f(\bar{x}), x - \bar{x} \rangle \;\; \leq \;\; f(x) - f(\bar{x})$$

□

Definición A.1 *Una función $f : \mathbb{R}^n \to \mathbb{R}$ de denomina Lipschitz continua con constante $L > 0$ si*

$$||f(x) - f(y)|| \leq L||x - y|| \;\; \textit{para todo} \;\; x, y \in \mathbb{R}^n$$

Además, si la función $f \in C^1$ es continua diferenciable, se dice que el gradiente ∇f es Lipschitz continuo de constante $L > 0$, si

$$||\nabla f(x) - \nabla f(y)|| \leq L||x - y|| \;\; \textit{para todo} \;\; x, y \in \mathbb{R}^n$$

Lema A.2 (Lema de Descenso) *Sea una función $f : \mathbb{R}^n \to \mathbb{R}$ continua y diferenciable con gradiente Lipschitz continuo de constante L, entonces.*

$$f(x) \leq f(\bar{x}) + \langle \nabla f(\bar{x}, y - \bar{x}) + \frac{L}{2}||x - \bar{x}||^2 \rangle \;\; \textit{para cada} \;\; x, y \in \mathbb{R}^n \tag{A.2}$$

Demostración. Definiendo $\varphi : \mathbb{R} \to \mathbb{R}$

$$\begin{aligned}
\varphi(t) &= f(\bar{x} + t(x - \bar{x})),\ t \in \mathbb{R} \\
\varphi(0) &= f(\bar{x}),\ \ \varphi(1) = f(x) \\
\varphi'(t) &= \langle \nabla f(\bar{x} + t(x - \bar{x})), x - \bar{x} \rangle
\end{aligned}$$

Entonces, por Teorema fundamental del Cálculo

$$\begin{aligned}
\varphi(1) - \varphi(0) &= \int_0^1 \varphi'(t)\, dt \\
f(x) - f(\bar{x}) &= \int_0^1 \langle \nabla f(\bar{x} + t(x - \bar{x})), x - \bar{x} \rangle\, dt \\
f(x) - f(\bar{x}) - \langle \nabla f(\bar{x}), x - \bar{x} \rangle &= \int_0^1 \Big[\langle \nabla f(\bar{x} + t(x - \bar{x})), x - \bar{x} \rangle - \langle \nabla f(\bar{x}), x - \bar{x} \rangle \Big]\, dt \\
&= \int_0^1 \Big[\langle \nabla f\Big(\bar{x} + t(x - \bar{x})\Big) - \nabla f(\bar{x}), x - \bar{x} \rangle \Big]\, dt
\end{aligned}$$

utilizando la desigualdad de Cauchy-Schwarz

$$\begin{aligned}
&\leq \int_0^1 \underbrace{||\nabla f\Big(\bar{x} + t(x - \bar{x})\Big) - \nabla f(\bar{x})||}_{\text{Lipschitz-continuo}} \cdot ||x - \bar{x}||\, dt \\
&\leq \int_0^1 L||t(x - \bar{x})|| \cdot ||x - \bar{x}||\, dt \\
&= \frac{L}{2}||x - \bar{x}||^2
\end{aligned}$$

□

Anexo B

Construcción Matriz difuminado

La imagen de verificación borrosa (256×256 píxeles) se ha creado con un filtro (*kernel*) gaussiano mediante una ventana deslizante de ancho/alto variable como se muestra en la figura B.2. Cada píxel borroso es una combinación lineal de este *kernel* gaussiano aplicado en una zona particular de la imagen. De este modo, cada píxel difuminado en la imagen borrosa es el resultado de la suma ponderada del valor de los píxeles vecinos y el valor de la ventana (de color rojo en el ejemplo de la fig. B.1) del filtro gaussiano.

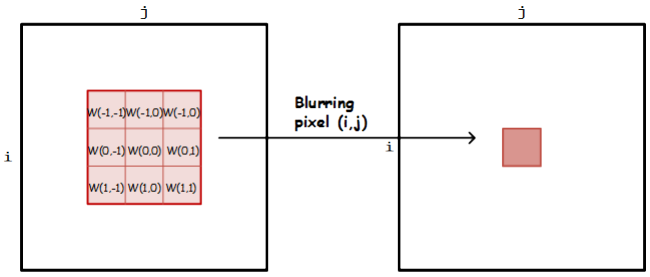


Figura B.1: Filtro de difuminado.

La generación de la matriz de difuminado (**A**) resulta de aplicar y deslizar consecutivamente el *kernel* gaussiano en una matriz auxiliar nula (ceros) de similar tamaño a la imagen original. Cada deslizamiento agrupa todas las columnas de la matriz auxiliar nula en una sola fila, de largo dado por el tamaño de la matriz auxiliar, y luego copia esta extensa fila única en una fila de la matriz A (direccionado por un índice de deslizamiento). La figura B.2 muestra un ejemplo de creación de la matriz A con el kernel gaussiano deslizante, donde se observa que la primera fila de la matriz A corresponde al filtro gaussiano posicionado en la esquina superior derecha de la matriz auxiliar. Este proceso se repite para cada deslizamiento de la ventana hasta cubrir todos los píxeles de la imagen auxiliar nula.

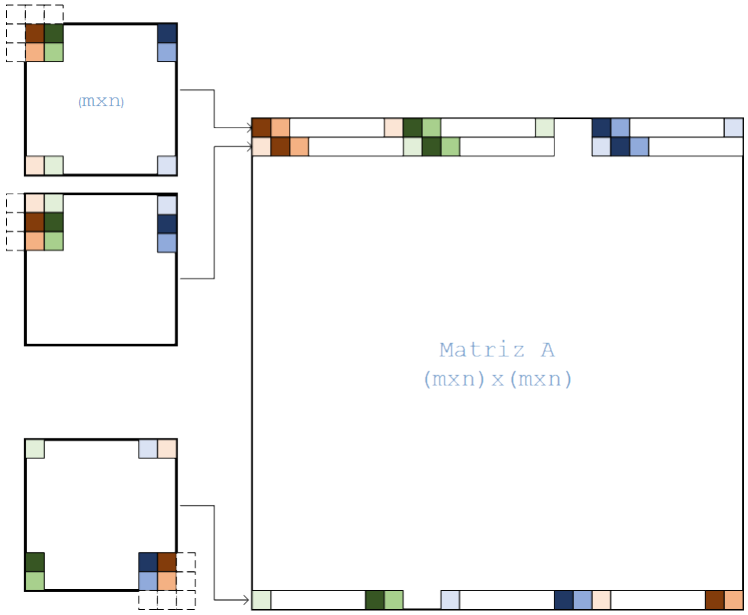


Figura B.2: Ejemplo deslizamiento filtro gaussiano para generar matrix de difuminación A.

El tamaño de la matriz A para el caso de una imagen 256×256 es de 65536×65536 pixeles. Esta matriz resulta ser una matriz tipo *sparse* con valores del filtro gaussiano localizados principalmente en la diagonal y esquina superior derecha, y inferior izquierda como se muestra en la figura B.3.

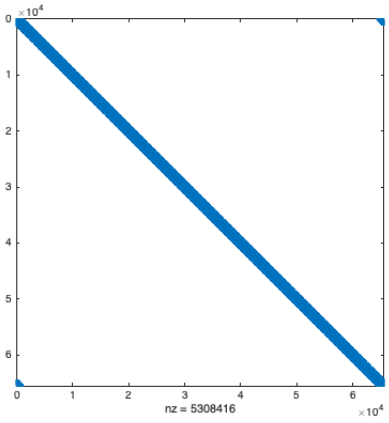


Figura B.3: La matriz de difuminado A resulta dispersa (*sparse*).

Anexo C

Algoritmos Matlab

Esta parte hace una descripción general de los algoritmos en *Matlab* [14] experimentados. La parte principal del código se encarga de las configuraciones iniciales, necesarias para el funcionamiento de los algoritmos. Entre las configuraciones principales, se define dos funciones anónimas para la función objetivo y su gradiente C. Estas funciones anónimas son usadas por los algoritmos como argumentos de entrada. El propósito es manejar eficientemente el tamaño en memoria de la matriz de difuminado $\mathbf{A}$, por los diferentes algoritmos. Se debe considerar esta matriz es de gran tamaño comparado con el tamaño de la imagen procesada. Para una imagen 256×256 la matriz $\mathbf{A}$ es de 65536×65536.

Código C.1: Función objetivo y Gradiente.

```
% gradient
grad_fx = @(x) A_transpose*(A*x - b);
% cost function
cost_fx = @(x,lambda) 0.5 * norm(A*x-b)^2 + lambda*sum(abs(x));
```

La parte principal del programa funciona como un coordinador de llamadas a los algoritmos. Cada llamada está estructurada con los mismos argumentos de entrada, y recibe los mismos resultados. Los argumentos de entradas son, las dos funciones anonimas definidas primeramente (función de optimización y su gradiente), más un vector de inicialización, con las principales parámetros particulares de manejo de los algoritmos.

Código C.2: Parte principal.

```
for i=1:length(names)
    switch names{i}
        case 'ista'
            [fx(i,:), x_mid, x_k, resp(i,:)]  = ista(grad_fx, cost_fx, x_0, opts);
        case 'fista'
            [fx(i,:), x_mid, x_k, resp(i,:)]  = fista(grad_fx, cost_fx, x_0, opts);
        case 'ipahd_ns'
            opts.lambda = 1e-4;    % ipahd_ns
            opts.step = 1/(150*L);  %6.6e-3
            [fx(i,:), x_mid, x_k, resp(i,:)]  = ipahd_ns(grad_fx, cost_fx, x_0, opts);
        case 'igahd'
            [fx(i,:), x_mid, x_k, resp(i,:)]  = igahd(grad_fx, cost_fx, x_0, opts);
    end
```

```
end
```

El algoritmo ISTA es el más sencillo de implementar. Los pasos de actualización se ejecutan en las primeras dos líneas del bucle iterativo.

Código C.3: Algoritmo ISTA.

```
for k=1:maxiter
    % gradient step
    x_grad = x_k - step .* grad_fx(x_k);

    % soft thresholding
    x_next = prox_l1(x_grad,lambda * step);

    % get criteria  error  for  stoppping algorithm
    [ etol, errors([1  2 3  4], k) ] = stop_criteria( 'norm2', x_next, x_k );

    % update
    x_k = x_next ;

    % function value for current  iteration
    fx_k(k) = cost_fx(x_next,lambda);
    % check tolerance
    if  etol<tol, break; end
end
```

El algoritmo FISTA los pasos intermedios de aceleración.

Código C.4: FISTA

```
for k=1:maxiter

    % gradient step
    x_grad = y_k - step .* grad_fx(y_k);

    % soft thresholding
    x_next = prox_l1(x_grad,lambda * step);

    % intermediate t step
    t_next = 0.5 * ( 1 + sqrt(1 + 4*t_k^2) );

    % FISTA update
    y_next = x_next + ((t_k - 1)/t_next) * (x_next - x_k);
    % check tolerance
    if  etol<=tol, break; end
end
```

Código C.5: Algoritmo IPAHD-NS.

```
% preparing mu_k
mu = zeros(1,maxiter);
```

```
H  = zeros(1,maxiter);

lambda_k_alpha = lambda*((1:maxiter)+alpha);
alpha_k = 1 - (alpha./((1:maxiter)+alpha));
for k=1:maxiter
    mu(k) = lambda_k_alpha(k) /...
          ( lambda_k_alpha(k) + k*(beta_k*sqrt(h) + h*b_k(k)) );
    H(k) = beta_k * sqrt(h) * alpha_k(k) * (1/lambda);
end

for k=1:maxiter

    % viscous-acceleration part
    M_k = alpha_k(k) .* (x_k - x_prev);

    % friction-hessian part
    Gfx_k =  H(k) .* ( x_k - ...
             prox_l1( x_k - sqrt(h)*grad_fx(x_k),lambda*sqrt(h) ) );

    % intermediate step
    y_k = x_k + M_k + Gfx_k ;

    % final step
    x_next = mu(k)*(y_k) + (1-mu(k)) * prox_l1(y_k-(1/mu(k))*grad_fx(y_k), lambda/
    ↪ mu(k) ) ;
    %       ------------------------------------------------
    % end algorithm update
    %       ------------------------------------------------
    % function value for current  iteration
    fx_k(k) = cost_fx(x_next,lambda);
    % update
    x_prev = x_k; x_k = x_next;
end
```

Código C.6: IGAHD

```
% prox and gradient in the metric M (eq (22) p.24 Attouch-Fadili-Riah)
proxF_M = @(x) prox_l1( x - h * grad_fx(x),lambda * h);
gradF_M = @(x) x - proxF_M(x);

for k=1:maxiter
    %x_k = (y_k-h*grad_fx(y_k)) - h * gradF_M(y_k);
    x_k = (1-h)*y_k + h*prox_l1( y_k - h * grad_fx(y_k), lambda*h );

    % momentum
    M_k = alpha_k(k) .* (x_k - x_p);

    % Hessian part. See Attouch-Fadili-Riah p.10
    H_k = beta * sqrt(h) * (gradF_M(x_k) - (1-1/k) * gradF_M(x_p));

```

```
15      % intermediate step
16      y_k = x_k + M_k - H_k;
17
18      % final step
19      x_next = y_k - step * grad_fx(y_k);
20      % update
21      y_p = y_k; x_p = x_k;
22
23      % function value for current  iteration
24      fx_k(k) = cost_fx(y_k,lambda);
25  end
```

Bibliografía

[1] Felipe Alvarez, Hedy Attouch, Jérôme Bolte, and Patrick Redont. A second-order gradient-like dissipative dynamical system with hessian-driven damping.: Application to optimization and mechanics. *Journal de mathématiques pures et appliquées*, 81(8): 747–779, 2002.

[2] Hedy Attouch, Juan Peypouquet, and Patrick Redont. Fast convex optimization via inertial dynamics with hessian driven damping. *Journal of Differential Equations*, 261 (10):5734–5783, 2016.

[3] Hedy Attouch, Zaki Chbani, Juan Peypouquet, and Patrick Redont. Fast convergence of inertial dynamics and algorithms with asymptotic vanishing viscosity. *Mathematical Programming*, 168(1-2):123–175, 2018.

[4] Hedy Attouch, Zaki Chbani, Jalal Fadili, and Hassan Riahi. First-order optimization algorithms via inertial systems with hessian driven damping. *arXiv preprint arXiv:1907.10536*, 2019.

[5] Amir Beck. *First-order methods in optimization / Amir Beck, Tel-Aviv University, Tel-Aviv, Israel.* MOS-SIAM series on optimization. Society for Industrial and Applied Mathematics, Mathematical Optimization Society, Philadelphia. ISBN 9781611974980.

[6] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[7] D.P. Bertsekas and Massachusetts Institute of Technology. *Convex Optimization Algorithms*. Athena Scientific, 2015. ISBN 9781886529281. URL https://books.google.cl/books?id=AfB5rgEACAAJ.

[8] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, USA, 2004. ISBN 0521833787.

[9] Alfred M Bruckstein, David L Donoho, and Michael Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM review*, 51(1): 34–81, 2009.

[10] Antonin Chambolle and Thomas Pock. An introduction to continuous optimization for imaging. *Acta Numerica*, 25:161–319, 2016.

[11] Scott Shaobing Chen, David L Donoho, and Michael A Saunders. Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159, 2001.

[12] David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52 (4):1289–1306, 2006.

[13] Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical learning with sparsity: the lasso and generalizations*. Chapman and Hall/CRC, 2015.

[14] MATLAB Optimization Toolbox. Matlab optimization toolbox, 2015.

[15] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[16] Yurii Evgen'evich Nesterov. A method of solving a convex programming problem with convergence rate o(kˆ2). In *Doklady Akademii Nauk*, volume 269, pages 543–547. Russian Academy of Sciences, 1983.

[17] Neal Parikh, Stephen Boyd, et al. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, 2014.

[18] Juan Peypouquet. *Convex optimization in normed spaces: theory, methods and examples*. Springer, 2015.

[19] Boris T Polyak. Introduction to optimization. optimization software. *Inc., Publications Division, New York*, 1, 1987.

[20] B.T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1 – 17, 1964. ISSN 0041-5553. doi: https://doi.org/10.1016/0041-5553(64)90137-5. URL http://www.sciencedirect.com/science/article/pii/0041555364901375.

[21] R. Tyrrell Rockafellar. *Convex analysis*. Princeton Mathematical Series. Princeton University Press, Princeton, N. J., 1970.

[22] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

[23] Weijie Su, Stephen Boyd, and Emmanuel Candes. A differential equation for modeling nesterov's accelerated gradient method: Theory and insights. In *Advances in neural information processing systems*, pages 2510–2518, 2014.