UNIVERSIDAD DE BUENOS AIRES

Facultad de Ciencias Exactas y Naturales

Departamento de Matemática

Tesis de Licenciatura

Teoría de Regularización para Problemas Mal Puestos y Super-Resolución Basada en Información: Aplicaciones en Microscopía.

Franco Alessio Bongiovanni

Directora: Dra. Sandra Martínez
Director: Dr. Axel Lacapmesure

Fecha de Presentación: 19/09/2025

Resumen

En este trabajo ahondaremos en el problema de la deconvolución para imágenes ópticas. Presentaremos el problema y por qué resulta un problema "mal puesto" que al intentar invertirlo directamente lleva a una imagen con baja resolución o artificios. Nos adentraremos en la teoría de operadores (en particular, los operadores compactos). Presentaremos una serie de métodos (regularización) que lo convierten en un problema "bien puesto", y los probaremos con un caso de estudio particular. Posteriormente, presentaremos un método de súper-resolución para microscopías de fluorescencia basado en aproximar la muestra como una superposición de fuentes puntuales, llamado SUPPOSe. En esta tesis realizamos la implementación en lenguaje Python de una variante del mismo, llamada DSUPPOSe. Explicaremos este método y su diferencia con el método original. Finalmente, buscaremos aplicar nuestra implementación a un caso de estudio particular (el mismo utilizado para los métodos de regularización). Con esta información, mostraremos las soluciones obtenidas por ambos métodos y haremos comparaciones.

Agradecimientos

En primer lugar, quiero agradecer a mi mamá Sandra, por apoyarme, aconsejarme y siempre brindarme un panorama amplio de opciones, para ayudarme a tomar las mejores decisiones. Por siempre proveerme todas las herramientas posibles para desenvolverme de la mejor manera en la actividad que me propusiera. Por haberme llevado a conocer el mundo y los lugares hermosos que hay en él.

A la Dra. Sandra Martínez y al Dr. Axel Lacapmesure, por haber sido mis directores en esta tesis. Por haberme guiado, orientado y respondido las infinitas consultas que iban surgiendo, tanto con la teoría matemática como con la implementación en Python, por estar siempre a disposición ante cualquier necesidad que pudiera tener y por la infinita paciencia.

Al Lic. Guillermo Enrique Crotti, por todos los consejos sobre temas de tesis a elegir y sobre la futura vida profesional, además de los muchos partidos de tejo y truco compartidos en los veranos.

A la Olimpiada Matemática Argentina (OMA), por haberme cambiado la vida y haberme hecho parte de su excelente comunidad.

A mis entrenadores (y amigos) de OMA, Nicolás Cogorno y Ian Fleschler, por haberme ayudado a mejorar día a día en esa etapa, por todo lo aprendido y por haberme hecho lograr mis mejores resultados.

A los grandes amigos que me dejó la OMA, que son muchos y espero no olvidarme de ninguno. Agus, Bat, Cami, Chino, Facu, Joa, Julián, Lolo, Lucía, Marcos, Martu, Nico, Pedro, y el Ruso. Especialmente a Santi, por haber sido el mejor compañero de entrenamientos que pudiera haber tenido, y a Marchi, por tantos partidos de Racing compartidos en el Cilindro de Avellaneda.

A Damián y Martín, por haber transitado tantos exámenes y materias juntos. Dos ex-

celentes amigos que me dejó este camino.

A Tefi y Mati, por ser los mejores compañeros que me dejó el colegio y por ayudar a organizarme siempre que tenía que ausentarme por exámenes, viajes y competencias. Fueron una gran parte de mi desempeño académico en esos tiempos y hoy, dos grandes amigos.

A los profesores que tuve en el camino. Todos dejaron un granito de arena para convertirme en la persona que soy hoy.

A todos mis alumnos de OMA, por permitirme transmitirles la pasión que siento por esta disciplina, y por enseñarme día a día.

Índice general

1. Introducción **5**

1.1. Fundamentos de microscopía 5

1.2. Formulación del problema y métodos de deconvolución 7

1.3. Deconvolución con súper-resolución 9

1.4. Objetivos 11

2. Regularización de problemas inversos: métodos de deconvolución **13**

2.1. El problema de la deconvolución 13

2.2. Operadores compactos, descomposición en valores singulares e inversa generalizada 19

2.3. Regularización de la inversa generalizada 29

2.4. Calificación de un Método de Regularización 33

3. Los métodos SUPPOSe y DSUPPOSe **42**

3.1. El método SUPPOSe 42

3.2. Algoritmo genético 46

3.3. La Transformada de Fourier Discreta (DFT) 51

3.4. El método SUPPOSe Discreto 55

4. Análisis **59**

4.1. Cuantificación de Resultados 64

4.2. Aplicación del Caso de Estudio a otros métodos de deconvolución. 68

5. Conclusiones **84**

5.1. Trabajo futuro 85

Capítulo 1

Introducción

1.1. Fundamentos de microscopía

La microscopía óptica es una herramienta esencial para la exploración de estructuras microscópicas en campos como la biología, la física y la ciencia de materiales. Sin embargo, las cualidades de los microscopios ópticos están limitadas por ciertos principios físicos fundamentales, preponderando entre ellos el fenómeno de la difracción de la luz. Esta propiedad indica que cuando un haz de luz pasa por una abertura o interactúa con un objeto, la trayectoria de dicho haz cambia de dirección, formando una serie de patrones muy característicos. Al tomar una foto de una fuente puntual, en lugar de obtenerse un punto bien definido, se obtiene un patrón de difracción llamado Disco de Airy, compuesto por un máximo central rodeado de anillos.

En general, la forma de este patrón de difracción está determinada por la propagación del haz de luz emitido desde la muestra al detector. Cuando la muestra está conformada por un emisor puntual, el patrón de difracción proyectado sobre el detector obtiene una expresión paramétrica denominada "Función de Dispersión de Punto" (la PSF, por las siglas en inglés: *point-spread function*), que representa la función de Green del operador que propaga el campo electromagnético de la luz a lo largo del experimento. Dado que las ecuaciones de Maxwell son lineales, vale el principio de superposición, de modo que dos emisores puntuales se observarán en la imagen como dos PSF superpuestas, separadas una cierta distancia. Y una distribución arbitraria de emisores se observará como el resultado de una integral de convolución. En este sentido, la microscopía óptica define la

"resolución" de una imagen como la mínima distancia de separación entre dos emisores puntuales tal que ambos sean distinguibles en la imagen adquirida. De modo que la resolución es una medida de nitidez en la escala física del objeto, y no debe confundirse con el uso de la palabra resolución en fotografía, donde la resolución es una medida de tamaño de la imagen en cantidad de píxeles.

En la microscopía convencional, los instrumentos aspiran a tener una PSF degradada únicamente por la propia difracción. Este es el llamado "límite clásico", y en esos casos se dice que la PSF está "limitada por difracción". En condiciones más realistas, además del fenómeno de difracción, existen otros fenómenos que afectan a la PSF que podemos llamar "aberraciones ópticas": singularidades debido a imperfecciones en los instrumentos que distorsionan la forma y posición de los puntos en la imagen, afectando drásticamente la calidad de la imagen. En el fondo, la PSF modela todos los efectos ópticos que han "degradado" una imagen ideal. En general, podremos aproximar la intensidad de la PSF por una distribución de tipo gaussiana de la siguiente forma en $\mathbb{R}^2$,

$$PSF(x_1, x_2) = e^{-\frac{x_1^2+x_2^2}{2\sigma^2}}. \tag{1.1}$$

Donde x_1 y x_2 representan las coordenadas en el detector, presuponiendo que el emisor puntual se halla en $x_1 = 0, x_2 = 0$. A partir de esta expresión, existen varios criterios para definir la resolución de un sistema.

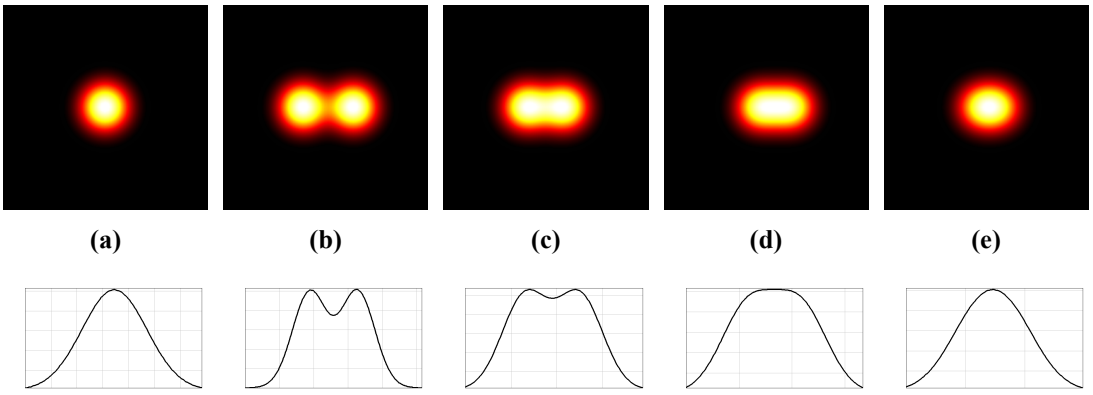


Figura 1.1: Análisis cualitativo de la PSF de un microscopio óptico (corte transversal) bajo la aproximación de PSF gaussiana. Tenemos en (a), el caso de una sola partícula, mientras que en los demás colocamos dos partículas a distancias entre ellas que van entre $2,8\sigma$ y $0,8\sigma$. Para cada caso, tenemos (arriba) la imagen adquirida y (abajo) un perfil horizontal tomado a lo largo del centro de cada imagen.

Como mencionamos recién, ciertos criterios permiten de algún modo cuantificar este concepto de resolución, como por ejemplo, el criterio de Rayleigh, que determina que el esquema para dos partículas está "resuelto" cuando el máximo del disco de Airy coincide con el primer valor mínimo del disco correspondiente al emisor vecino, que podemos visualizar en (b) de la Figura 1.1. Al aproximar por una PSF gaussiana, este criterio coincide con tener una distancia de $2,8\sigma$. De acuerdo a la teoría de microscopía clásica [1], σ depende de la longitud de onda λ de la luz utilizada (que en microscopía óptica va desde ~ 300 a ~ 800 nanómetros) y de la apertura numérica AN del microscopio (un número adimensional que caracteriza el tamaño del cono angular para el cual el sistema óptico acepta luz). En sí, el límite de resolución viene dado por la siguiente fórmula, tomando $C \sim 1$ como una constante a fijar, que depende del criterio utilizado,

$$C\frac{\lambda}{AN}.$$

Teniendo longitudes de onda cortas y apertura numérica grande, mejoramos la resolución; aunque, en condiciones óptimas, igualmente la difracción establece una barrera física limitante.

1.2. Formulación del problema y métodos de deconvolución

Ahora estamos en condiciones de modelar la muestra obtenida. Esto se sustenta en el principio de que una imagen observada S se obtiene como el resultado de la convolución del objeto real (al que llamamos R) y la PSF, que llamaremos I, sumado a un término η que modela el ruido. Es decir,

$$S(x) = R(x) * I(x) + \eta(x) \tag{1.2}$$

η, como dijimos, representa el ruido en la imagen, que puede ser aleatorio de distintos tipos (Gaussiano, Poisson u otro tipo), dependiendo de cómo fue adquirida la imagen. La idea para obtener R con una mejor resolución que el método original es invertir la influencia

de la distorsión con procesos deconvolutivos (conociendo, claro, la PSF) y resolver este problema inverso.

Una vez planteadas estas condiciones y limitaciones, surge un fuerte desarrollo de nuevas perspectivas y técnicas -llamadas de súper-resolución- para mejorar la resolución por debajo del límite clásico y obtener así una imagen más "nítida" del objeto. En este tipo de problemas, pequeñas perturbaciones en las variables de entrada pueden producir cambios significativos en la solución. Las características del operador de convolución, entonces, hacen que, según la definición dada en [2], en general el problema esté *mal condicionado* o mal puesto (del inglés: *ill-posed*). Al hacer la operación de la convolución con la PSF, esta última también está filtrando los aportes de frecuencias altas a la imagen original, y dichos aportes guardan información crucial en cuanto a los detalles finos de la imagen. Aparte de esto último, el inconveniente mayor radica entonces en que el problema inverso no resulta continuo.

A mediados de la década del 70, comenzaron a surgir técnicas denominadas métodos de regularización, cuya idea radica en transformar el problema "mal puesto" de la deconvolución en uno bien condicionado a partir de incorporar un cierto parámetro de regularización. En [3] podemos encontrar estos y otros métodos implementados como herramientas plug-ins para realizar análisis de imágenes. El funcionamiento de estas técnicas será mostrada con ejemplos de uso a lo largo del siguiente capítulo. A continuación, describiremos algunas técnicas de relevancia.

Algunos de los métodos que toman preponderancia dentro de los métodos de regularización incluyen al Método de Tikhonov [4]. Junto con otras variantes, como el Método de Tikhonov-Miller [5], integran un grupo de métodos de tipo penalización. La idea es encontrar una forma de penalizar la función objetivo utilizada. Para ello, se modifica la función objetivo agregando un término adicional junto con un parámetro regularizador como constante multiplicativa. Es decir, se cambia la función objetivo de modo que ese término adicional deje al problema bien condicionado.

La elección de los parámetros resulta crucial. Si los elegimos de manera inadecuada, damos lugar a la aparición de artificios. En cuanto a las variantes de Tikhonov, si bien el objetivo es mejorar la resolución sobre el objeto real, por ejemplo, en la variante Tikhonov-Miller, se incorpora información previa sobre la solución que esperamos. En todas estas técnicas, si se elige de forma adecuada el parámetro de regularización, se logra reducir el

ruido de la imagen. En cambio, la mejora en la resolución resulta marginal.

Como parte de las técnicas de regularización, surgen métodos de deconvolución que son más robustos al ruido utilizando algoritmos iterativos. De los más utilizados en microscopía óptica podemos mencionar el método de Richardson-Lucy [6] y [7], un algoritmo que mejora la calidad de la imagen (en el sentido del problema que buscamos resolver) a partir de plantear cada iteración como un problema de Máxima Verosimilitud para una distribución de ruido de tipo Poisson. Este método resulta ser más robusto que la inversión directa cuando la imagen adquirida está limitada por ruido de disparo (en inglés: *shot-noise*), pero cuando estamos en presencia de otros tipos de ruido (como ruido electrónico de las cámaras S-CMOS y CCD), no obtendremos una buena solución. Además, a medida que crece el número de iteraciones podemos observar grandes fluctuaciones de la solución (podríamos estar aumentando el ruido). En este caso, el parámetro de regularización viene dado por el número de iteraciones del método (técnica *early-stopping*).

En la década del 90, David Donoho propone en [8] el método de "compressed sensing", ampliamente utilizado en procesamiento de señales. La idea radica en intentar reconstruir una señal completa a partir de un número significativamente menor de mediciones de aquellas dadas por el teorema de muestreo (Nyquist-Shannon). En general, el método busca aprovechar la redundancia presente en las señales de interés. En particular, muchas señales son ralas (sparse), es decir, contienen muchos coeficientes cercanos o iguales a cero cuando se representan en algún dominio. Entonces el CS generalmente comienza tomando una combinación lineal ponderada de muestras en una base diferente de aquella en la que la señal es conocida por ser dispersa. La clave está en que la mayoría de las señales son ralas, lo que significa que, aunque estas últimas pueden ser complejas, la mayor parte de su información se encuentra condensada en unos pocos coeficientes no nulos. Por esta razón, CS en vez de medir la señal completa, utiliza un número reducido de mediciones aleatorias que contienen suficiente información para recuperar la señal original.

1.3. Deconvolución con súper-resolución

A mediados de la década del 2000, Emmanuel Candès incorpora la idea planteada en el párrafo anterior en resolver el problema de superresolución (obtener factores de mejora en la resolución mayores a 2) [9, 10, 11]. Y para ello, busca mediante técnicas de opti-

mización, implementar la minimización de la norma L^1. Dicha minimización impone la condición de raleza sobre la muestra. A diferencia de los métodos clásicos de regularización, en este caso, se imponen condiciones sobre la muestra para conseguir convertir el problema mal puesto en uno bien puesto. Ahora, si bien estamos utilizando una menor cantidad de mediciones, este método consigue recuperar con superresolución imágenes ralas, cuantificándolo, por ejemplo, en los casos en que hay dos o tres fuentes dentro de una PSF. Ver [11].

En los últimos años se ha desarrollado un método de súper-resolución llamado SUPPOSe (Superposition of virtual point sources) [12, 13] el cual obtiene, utilizando una sola imagen (al igual que compressed sensing), una reconstrucción del objeto original con mejor resolución que la determinada por el tamaño de la PSF. El método SUPPOSe se basa en suponer que la estructura real del objeto se puede aproximar como una superposición de fuentes puntuales de igual intensidad, denominadas fuentes virtuales. Este enfoque convierte el problema mal condicionado de hallar la intensidad para cada posición en un problema bien planteado de encontrar las posiciones de las fuentes virtuales. En [14] se demostró que aunque el método no requiere asumir que la imagen es rala como sí ocurre en los métodos de compressed sensing (CS). No obstante, cuanto menor es la densidad, mejor es la solución obtenida. Se obtuvieron las cotas teóricas de las incertezas en función de la densidad del objeto, el ruido y el error en la PSF. En dicho trabajo, también se demuestra, a partir de las cotas teóricas obtenidas allí, cuál es el número de fuentes virtuales óptimas, en función de los distintos parámetros de la muestra. Las distintas variaciones de intensidad se van a construir acumulando distintas cantidades de fuentes cercanas. El problema a resolver involucra nuevamente una instancia de minimización, donde ahora la única incógnita que se busca hallar son las posiciones de las fuentes virtuales, dejando que estas tomen valores en todo el espacio de la imagen como dominio.

Ahora, hagamos foco en la calidad de la solución. En principio, dependerá de algunos factores, como cuán bien hemos estimado la PSF y cuántas fuentes virtuales se utilizan. La manera de hallar las posiciones de las fuentes virtuales que mejor aproximan a la estructura real en SUPPOSe, es a través de un algoritmo iterativo de minimización denominado genético [15, 16] que va a requerir de múltiples evaluaciones de una función objetivo determinada. Al ser la convolución una operación costosa en términos de complejidad computacional, los tiempos de cálculo pueden mejorarse calculando la convolución en

placas GPU para paralelizar la operación [17], pues debido a su alto costo computacional, la mayor parte del tiempo en SUPPOSe se gasta en el cálculo de estas evaluaciones. Sin embargo, a pesar de esta mejora en los tiempos, estos terminan aumentando significativamente cuando se aplican a muestras 2D más grandes, muchas muestras, o muestras 3D.

La solución de SUPPOSe tiene fuerte dependencia en el número de fuentes virtuales utilizadas (aunque el rango de valores resulta amplio), la calidad de la estimación de la PSF y principalmente de la relación señal/ruido de la imagen adquirida (análisis planteado en [14]). Si bien, como mencionamos antes, se tenían cotas teóricas que dependían de la densidad del objeto, del ruido y del error en la PSF [14], tenía entonces sentido plantear la pregunta de cuál resulta el desempeño del método para cada caso. Por ello, en [18] se realiza un análisis de cómo es la precisión, exactitud y resolución del método dependiendo de la relación señal/ruido. Dicho análisis cuantitativo del desempeño se realizó utilizando la pseudométrica definida en [19].

Para mejorar la velocidad de convergencia de SUPPOSe, se propuso en [20] una variante del método, denominada DSUPPOSe, en el cual se discretiza el espacio donde las fuentes estarían localizadas. Para esta nueva formulación se utiliza una función objetivo distinta que ya no requiere evaluar la PSF a cada paso del proceso iterativo, pero por sobre todas las cosas, donde la convolución pasa a ser reemplazada por una multiplicación de matrices. Si se comparan ambos métodos, obtendremos el mismo desempeño de resolución, exactitud y precisión, pero con una mejora en el tiempo de ejecución para DSUPPOSe. Dicha situación fue implementada en código Matlab con una CPU clásica, para luego compararse con la versión SUPPOSe, implementada en una GPU, logrando una sustancial mejora en los tiempos de ejecución de DSUPPOSe (un factor de mejora entre 5 y 10, en función de la cantidad de fuentes virtuales utilizadas), con datos que pueden visualizarse en [20].

1.4. Objetivos

En esta tesis buscaremos hacer una implementación integral del método DSUPPOSe en lenguaje Python, aprovechando el buen comportamiento de la Transformada de Fourier con el operador de convolución, que es la función definida en DSUPPOSe [20]. Luego,

para llevar a cabo la minimización de dicha función objetivo, implementaremos el algoritmo de tipo genético utilizado en SUPPOSe [12]. A su vez, a lo largo del trabajo realizado, se buscó lograr una implementación que sea práctica para quien la utilice con guardados cómodos de las eventuales corridas del método y modularización de scripts tanto para la implementación como para análisis y gráficos futuros sobre datos de la salida.

Además, en el objetivo de esta tesis buscaremos describir la teoría de regularización de los problemas inversos, presentando el problema, y avanzando sobre la teoría sobre operadores compactos y SVD, regularidad y ejemplos de métodos de regularización. Utilizaremos resultados de los libros [21, 22]. La idea es poder distinguir el concepto de regularización (donde se adiciona un término a la función objetivo) de lo que es la súper-resolución (en donde se agrega información sobre el objeto). Posteriormente, buscaremos presentar el método SUPPOSe y describir el algoritmo genético como método de minimización utilizado, para derivar en la variante DSUPPOSe y la teoría matemática que lo motiva. Finalmente, nos dedicaremos a definir un caso de estudio, aplicarle el método DSUPPOSe implementado en Python, y compararlo con los métodos de deconvolución (presentados en el capítulo 2).

Esta tesis estará estructurada en 5 capítulos. El Capítulo 2 incluirá la teoría sobre regularización. En el Capítulo 3, presentamos SUPPOSe, el algoritmo genético y DSUPPOSe. Posteriormente, en el Capítulo 4 nos dedicaremos a definir el caso de estudio y a hacer comparaciones entre métodos. Para finalizar, en el Capítulo 5 realizamos una conclusión del trabajo realizado.

Capítulo 2

Regularización de problemas inversos: métodos de deconvolución

2.1. El problema de la deconvolución

La descripción matemática estándar de una imagen bidimensional (en blanco y negro) ubicada dentro de un dominio $D \subset \mathbb{R}^2$ suele darse mediante una función acotada y no negativa.

$$R : D \to \mathbb{R}_0^+,$$

que representa la intensidad del "objeto". La visualización S que obtendremos, que en adelante llamaremos imagen, suele describirse mediante la siguiente integral:

$$S(y) = \int_D I(x, y) R(x)\, dx,$$

donde el núcleo I, que también se asume no negativo, se conoce como la función de dispersión de punto o PSF (Point Spread Function). La función $I(x, \cdot)$ describe cuánto se "distribuye" (o desenfoca) la intensidad de la escena en un punto $x \in D$ dentro de la imagen.

El dominio de y (que representa los píxeles de la imagen) en principio no tendría por qué coincidir con D; de hecho, en aplicaciones reales, y suele estar restringido a un sub-

conjunto propio de D. En este trabajo nos enfocaremos en problemas de imagen donde la PSF es invariante en el espacio, es decir, cada punto $x \in D$ se desenfoca de la misma manera. En este caso, I solo depende de la diferencia $y - x$, lo que permite reescribir el proceso de formación de imagen como una convolución:

$$S(y) = \int_{\mathbb{R}^2} I(y - x)R(x)\, dx.$$

Además, supondremos que D es todo el espacio bidimensional, lo cual no implica pérdida de generalidad si extendemos f como cero fuera de D.

En la mayoría de los casos, las funciones de dispersión, en la práctica, tienen soporte compacto o son despreciables fuera de una cierta región. Por esta razón, generalmente se asume que pertenecen al espacio $L^1(\mathbb{R}^2)$. Nos concentraremos, como mencionamos en la introducción, en una PSF modelada con una función gaussiana.

Ahora, estamos en condiciones de definir el operador de convolución $A : L^2(\mathbb{R}^2) \to L^2(\mathbb{R}^2)$ como

$$(AR)(y) = \int_{\mathbb{R}^2} I(y - x)R(x)\, dx, \quad y \in \mathbb{R}^2, \tag{2.1}$$

el cual resulta continuo para $I \in L^1(\mathbb{R}^2)$; (puede hallarse la demostración de este hecho en el Apéndice E de [21]).

El operador A también puede expresarse en términos de la transformada de Fourier, según el Teorema de Convolución: si $S = AR$, entonces

$$\hat{S} = \hat{I}\hat{R},$$

donde la transformada de Fourier de una función $f \in L^2(\mathbb{R}^2)$ viene definida como

$$\hat{f}(\omega) = \int_{\mathbb{R}^2} e^{-2\pi i\omega\cdot x} f(x)\, dx, \quad \omega \in \mathbb{R}^2.$$

En principio, la ecuación anterior pareciera ofrecer un método simple para resolver el problema inverso, es decir, recuperar el objeto R a partir de la imagen S, siguiendo estos pasos:

1. Calcular la transformada de Fourier $\hat{S}$ de la imagen S.

2. Resolver $\hat{R} = \frac{\hat{S}}{\hat{I}}$.

3. Obtener la R original aplicando la transformada inversa de Fourier a $\hat{R}$.

Gracias a que el Teorema de Plancherel nos garantiza que la transformada de Fourier es un isomorfismo, los pasos 1 y 3 son estables. Sin embargo, en el caso restante nos encontraremos con un claro problema, que podríamos llamarlo *inestabilidad*. Este último término refiere a que, si nuestro operador no resulta continuamente invertible (como demostraremos aquí abajo), pequeñas variaciones en el lado derecho de nuestra ecuación repercuten en grandes fluctuaciones en la solución del problema. En general, en la práctica tendremos una ecuación

$$S^\delta(y) = \int_{\mathbb{R}^2} I(y-x)R^\delta(x)\,dx \tag{2.2}$$

donde vamos a tener controlada la diferencia entre S y S^δ por este parámetro δ.

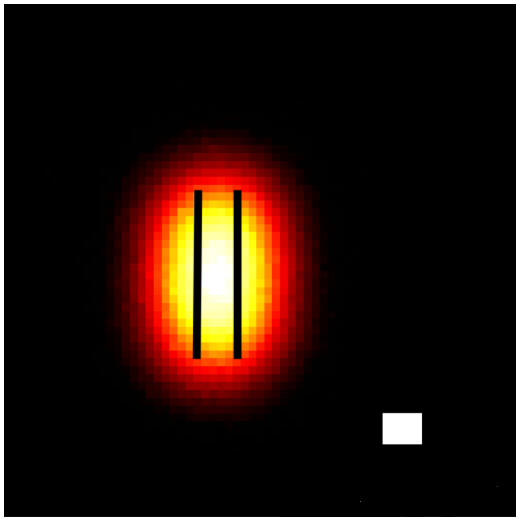
Sin embargo, la inestabilidad de nuestro problema inverso se manifestará en el paso 2, como veremos enseguida.

A priori, podríamos utilizar la siguiente idea para resolver el problema, que se conoce como el Naive Inverse Filter Method. La gracia del método radica en su sencillez, pues la idea es considerar como solución

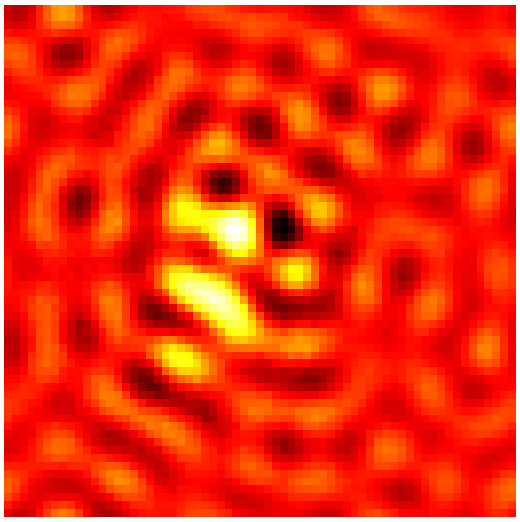
$$\hat{R} = \frac{\hat{S}}{\text{máx}(\varepsilon, \hat{I})}$$

donde ε es una constante pequeña que evita las divisiones por cero. Luego, la solución se obtiene tomando la transformada inversa.
La gran ventaja del método radica en que es libre de parámetros (salvando la constante ε) y al ser inversión directa (en el dominio transformado) es fácil de computar. Sin embargo, amplía mucho el ruido y lleva a que la imagen obtenida tenga artificios, como veremos en la Figura 2.1. Le aplicaremos el método a una imagen que utilizaremos a lo largo de la tesis, y trazaremos un perfil, para ver, por ejemplo, que toma valores negativos en ciertos lugares. El objeto constará de dos segmentos paralelos, con una separación entre ellos de un σ. Se genera con una sucesión de puntos que modelan esta estructura, convolucionados con una PSF de tipo Gaussiana, de amplitud σ. En ambos casos, σ toma el valor de 5 pí-

xeles. Además, se le suma un término de ruido modelado con una distribución de Poisson. Describiremos con más detalle este objeto y su construcción en el cuarto capítulo de este trabajo. El perfil trazado se realiza seleccionando la sección transversal en el centro de la imagen. La resolución, dependiendo del criterio que seleccionemos, será del orden del ancho a media altura de la gaussiana (lo llamaremos FWHM), que resulta $\approx 2,35\sigma$.



(a) Imagen Sintética de dos líneas verticales paralelas separadas a un $\sigma = 5$ píxeles de distancia.



(b) Solución obtenida luego de aplicar el método Naive.

Figura 2.1: Antes y después de una imagen aplicando el método Naive.

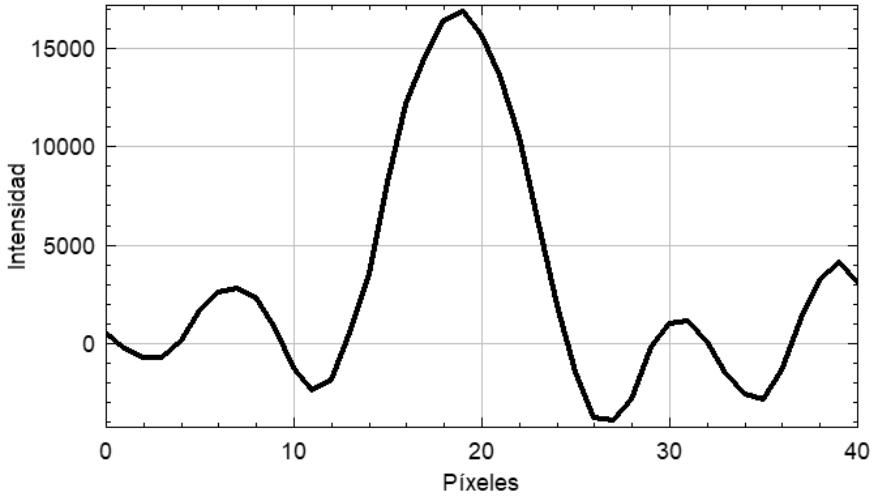


Figura 2.2: Perfil trazado en la Figura 2.1b donde se puede apreciar los valores que toma la imagen.

Podemos ver que este método trajo artificios mirando la Figura 2.2, pues la imagen pasó a tomar valores negativos de casi $\frac{1}{3}$ de la intensidad máxima. Ahora, retornando a la cuestión de la inestabilidad, enunciamos el siguiente teorema.

Teorema 2.1.1. *Para $I \in L^1(\mathbb{R}^2)$, el operador de convolución $A : L^2(\mathbb{R}^2) \to L^2(\mathbb{R}^2)$ dado en (2.1) no es continuamente invertible. Si $I \not\equiv 0$, entonces su espectro está dado por el conjunto conexo acotado (incontablemente infinito) $\hat{I}(\mathbb{R}^2) \cup \{0\}$, es decir, la unión del origen y todos los valores de la función $\hat{I}$.*

Para la demostración, haremos uso de un lema previo:

Lema 2.1 (Riemann-Lebesgue)**.** *Dada $f \in L^1(\mathbb{R}^2)$, entonces la transformada de Fourier $\hat{f}$ es continua y tiende a 0 cuando $|\omega| \longrightarrow \infty$.*

Demostración. Para ver la continuidad, miremos:

$$|\hat{f}(\omega) - \hat{f}(\omega')| \leq \int_{\mathbb{R}^d} |e^{-2\pi i\omega x} - e^{-2\pi i\omega' x}||f(x)|\, dx$$

Ahora podemos dividir el dominio de integración en dos partes. La integral sobre $|x| > R$ se hace pequeña cuando R toma valores grandes, pues $f \in L^1(\mathbb{R}^d)$. A su vez, la integral sobre el complemento $|x| \leq R$ se hace pequeña eligiendo ω' cerca de ω, debido a que la función exponencial compleja es una función uniformemente continua sobre compactos. Para la segunda parte, tomamos, dado $x \in \mathbb{R}^d$ la función característica χ_A de un cubo $A \subset \mathbb{R}^d$ centrado en x, con lados de longitud h. Ahora, se puede calcular manualmente que

$$\hat{\chi_A} = e^{-2\pi ix\omega} \prod_{k=1}^{d} \frac{sin(h\pi\omega_k)}{\pi\omega_k}$$

donde $\omega = (\omega_1, ..., \omega_d)$ y donde consideramos la extensión continua por h en $\omega_k = 0$. Cuando $|\omega| \longrightarrow \infty$, $\hat{\chi_A} \longrightarrow 0$. Luego, utilizando que las combinaciones lineales de funciones características son densas en $L^1(\mathbb{R}^d)$ y como el subespacio $C_0(\mathbb{R}^d)$ es cerrado en $C(\mathbb{R}^d)$, se sigue el lema. □

Demostración Teorema. Utilizando el lema de Riemann-Lebesgue, podemos notar que $\hat{I}(\mathbb{R}^2) \in \mathbb{C}$ es un conjunto acotado, conexo y con $\lambda = 0$ punto de acumulación. Entonces, $\hat{I}(\mathbb{R}^2) \cup \{0\}$ es un conjunto de puntos no numerable, a menos que $\hat{I} \equiv 0$, y por ende $I \equiv 0$. Entonces, dado $\omega_0 \in \mathbb{R}^2$, sea $\lambda = \hat{I}(\omega_0)$. Dado $\varepsilon > 0, \exists \delta = \delta(\epsilon)$ tal que:

$$|\hat{I}(\omega) - \hat{I}(\omega_0)| \leq \varepsilon \quad si \quad |\omega - \omega_0| \leq \delta$$

Consideremos $R_\delta \in L^2(\mathbb{R}^d)$ como la transformada de Fourier inversa de la función característica de un disco de radio δ centrado en ω_0, y definimos $S_\delta = (A - \lambda I_d)R_\delta$. Si tomamos la transformada de S_δ, por el Teorema de Convolución, está será:

$$\hat{S_\delta} = (\hat{I} - \hat{I}(\omega_0))\hat{R_\delta}$$

Además, si tomamos norma, utilizando Plancherel obtenemos

$$\frac{||S_\delta||_{L^2(\mathbb{R}^2)}}{||R_\delta||_{L^2(\mathbb{R}^2)}} = \frac{||\hat{S_\delta}||_{L^2(\mathbb{R}^2)}}{||\hat{R_\delta}||_{L^2(\mathbb{R}^2)}} \leq \varepsilon$$

(vale aclarar que la integral de $\hat{S_\delta}^2$ resulta soportada en el disco de radio δ).
Esto nos dice que

$$\frac{||(A - \lambda I_d)R_\delta)||_{L^2(\mathbb{R}^2)}}{||R_\delta||_{L^2(\mathbb{R}^2)}} \longrightarrow 0 \quad si \quad \delta \longrightarrow 0$$

por lo cual, $A - \lambda I_d$ no puede tener una inversa acotada y entonces, λ pertenece al espectro de A. Como vimos que el origen era punto de acumulación de $\hat{I}(\mathbb{R}^2)$, $\lambda = 0$ también pertenece al espectro.
Ahora tomemos un λ que no pertenezca a la clausura del rango de $\hat{I}$, es decir,

$$|\hat{I}(\omega) - \lambda| \geq \varepsilon \quad \text{para algún} \quad \varepsilon > 0 \quad \text{y cualquier} \quad \omega \in \mathbb{R}^2.$$

Así, dada cualquier $S \in L^2(\mathbb{R}^2)$ se tiene,

$$\hat{R} = \frac{\hat{S}}{\hat{I} - \lambda} \in L^2(\mathbb{R}^2) \quad \text{verifica} \quad ||\hat{R}||_{L^2(\mathbb{R}^2)} \leq \frac{1}{\varepsilon}||\hat{S}||_{L^2(\mathbb{R}^2)},$$

y luego, la transformada inversa de Fourier R de $\hat{R}$ es solución de $(A - \lambda I_d)R = S$ gracias al Teorema de Convolución. Entonces, $A - \lambda I_d$ tiene una inversa acotada, y λ no está en el espectro de A. □

Retornando a los pasos enumerados en líneas anteriores sobre nuestro problema inverso, supongamos que nuestro dato S^δ viene equipado con ruido de la forma $||S^\delta - S||_{L^2(\mathbb{R}^2)} \leq \delta$. Cuando hagamos la división propuesta en el paso 2 por pequeños valores

$\hat{I}(\omega)$, estaremos agrandando el ruido de las componentes de frecuencia del dato. En particular, componentes ruidosas de altas frecuencias (que se dan con $|\omega|$ grande), tomarán valores más grandes sin una cota que las controle (debido a que $\hat{I}(\omega) \longrightarrow 0 \quad \text{si} \quad |\omega| \longrightarrow \infty$, por el Lema de Riemann-Lebesgue), y nuestro resultado va a estar influenciado fuertemente por grandes errores. A su vez, los ceros que pueda tener $\hat{I}$ para frecuencias finitas, suman más dificultades a la cuestión.

En las secciones que siguen nos adentraremos en la teoría general, donde ahora A resulta un operador compacto. Para ello será necesario hacer el desarrollo de la teoría de operadores compactos, así como la definición de problema mal puesto en este contexto y en donde se vea dónde se encuentra el problema intrínseco. Finalmente definiremos los métodos de regularización.

2.2. Operadores compactos, descomposición en valores singulares e inversa generalizada

En esta sección vamos a desarrollar la teoría general para comprender cuál es la problemática detrás del problema mencionado en la sección anterior. Para eso vamos a desarrollar algunos resultados sobre operadores compactos así como la definición de lo que es un problema mal puesto.

Definición 2.1 (Producto Interno)**.** *Decimos que una aplicación en un espacio X a un cuerpo $\mathbb{K}$, $\langle .,. \rangle : X \times X \to \mathbb{K}$ es un producto interno si verifica:*

1. $\langle \lambda x + z, y \rangle = \lambda \langle x, y \rangle + \langle z, y \rangle \quad \forall x, y, z \in X, \ \forall \lambda \in \mathbb{K}$
2. $\langle y, x \rangle = \overline{\langle x, y \rangle} \quad \forall x, y \in X$
3. $\langle x, x \rangle \geq 0, \quad \forall x \in X \setminus \{0\}$

Definición 2.2 (Espacio de Hilbert)**.** *Decimos que X es un espacio de Hilbert si es un espacio de Banach (normado y completo), cuya norma proviene de un producto interno.*

En general, dados X, Y espacios de Hilbert, $A \in \mathcal{L}(X, Y)$, si tenemos $Ax = y, \quad x \in X, \quad y \in Y$, decimos que el problema está *"mal puesto"* si la inversa A^{-1} no existe, o resulta no continua. En principio, habría tres causas que podrían llevarnos a tener un problema mal puesto.

1. $Ker(A) = \{v : v \in X, \quad Av = 0\} \neq \{0\}$

2. $Rango(A) = \{u : \exists v \in X : Av = u\} \neq \overline{Rango(A)}$ (no es cerrado en Y)

3. $Rango(A) = \overline{Rango(A)} \neq Y$

A grandes rasgos, si el problema viene dado por la primera o tercera causa, podríamos esquivarlo cambiando los espacios y manteniendo la estructura de espacio de Hilbert. Sin embargo, si el inconveniente viene inducido por la segunda causa, no podemos operar como antes y decimos que tenemos un problema *esencialmente* mal puesto.

Enunciamos ahora una propiedad que dará una caracterización a la segunda causa y contextualizará los próximos resultados.

Proposición 2.1. *$Rango(A) = \overline{Rango(A)}$ si y solo si*

$$Ker(A)^{\perp} \neq \{0\} \, y \qquad \inf_{\substack{v \in Ker(A)^{\perp} \\ \|v\|_X = 1}} \|Av\|_Y > 0.$$

Un ejemplo importante de operador cuyo rango no es cerrado es un operador compacto. Ahora definiremos esta noción y probaremos equivalencias y propiedades útiles.

Definición 2.3. *Dados espacios E y F, un operador lineal $A : E \longrightarrow F$ se dice compacto si $\forall (x_n)_n \subset E$ sucesión acotada, entonces Ax_n tiene una subsucesión convergente.*

Proposición 2.2. *A es compacto si y sólo si $\forall D \subset E$ acotado, $A(D)$ resulta relativamente compacto*

Demostración. Sea D un conjunto acotado, A operador compacto y $(y_n)_n \subset \overline{A(D)}$. Considero, para cada n,$Ax_n \in A(D)$ tal que $||y_n - Ax_n|| < \frac{1}{n}$. Como $(x_n)_n \subset D$ y D acotado, entonces $(x_n)_n \subset D$ resulta una sucesión acotada. Como A es compacto, Ax_n tiene una subsucesión $(Ax_{n_k})_k$ convergente a un elemento y. Veamos que este es el límite de y_{n_k}.

$$\begin{aligned} ||y - y_{n_k}|| &\leq ||y - Ax_{n_k}|| + ||Ax_{n_k} - y_{n_k}|| \\ &\leq ||y - Ax_{n_k}|| + \frac{1}{n_k} \longrightarrow 0 \end{aligned}$$

Entonces $(y_{n_k})_k$ converge a y, por lo que $\overline{A(D)}$ resulta compacto.
La recíproca resulta inmediata. □

Ahora enunciamos algunas propiedades:

Definición 2.4 (Operador Adjunto)**.** *Sea un operador lineal $A : D \subset X \to X$ en un espacio de Hilbert X y $x \in X$. Si para cada $y \in D$ se tiene que $\langle x, Ay\rangle = \langle z, y\rangle$, para algún $z \in X$, entonces decimos que x está en el dominio del operador adjunto de A (que denotaremos A^*), que z es la imagen de x por dicho operador ($x \in D,\ z = A^*x$).*

Definición 2.5 (Operador Autoadjunto)**.** *Un operador $A \in \mathcal{L}(X, X)$ es autoadjunto si $A = A^*$*

Observación 2.1. *A^*A y AA^* son operadores compactos, auto-adjuntos y no negativos. Pues*

$$(A^*A)^* = A^*(A^*)^* = A^*A, \quad y \quad \forall u \in X, \langle A^*Au, u\rangle = \langle Au, Au\rangle = ||Au||^2.$$

Además, en [23] puede verse que si A compacto, entonces A^* es compacto, con lo que podemos deducir la compacidad.

Proposición 2.3. *Si X espacio de Hilbert, y $A \in \mathcal{L}(X, X)$ es un operador autoadjunto, entonces*

$$\alpha(A) := \sup_{||x||_X=1} |\langle x, Ax\rangle| = ||A||$$

Demostración. Acotando con Cauchy-Schwarz tenemos:

$$\sup_{||x||_X=1} |\langle x, Ax\rangle| \leq \sup_{||x||_X=1} ||A||||x||_X^2 = ||A||.$$

Para la otra desigualdad:

$$\langle x_1 \pm x_2, A(x_1 \pm x_2)\rangle_X = \langle x_1, Ax_1\rangle_X \pm 2Re\langle x_2, Ax_1\rangle_X + \langle x_2, Ax_2\rangle_X$$

Lo que implica

$$\langle x_1 + x_2, A(x_1 + x_2)\rangle_X - \langle x_1 - x_2, A(x_1 - x_2)\rangle_X = 4Re\langle x_2, Ax_1\rangle_X.$$

y utilizando la definición de α

$$4Re\langle x_2, Ax_1\rangle_X \leq \alpha(A)(||x_1 + x_2||_X^2 + ||x_1 - x_2||_X^2) = 2\alpha(A)(||x_1||_X^2 + ||x_2||_X^2)$$

para cualquier $x_1, x_2 \in X$. Si elegimos $x_1 \notin Ker(A)$ con $||x_1||_X = 1$ y $x_2 = \frac{Ax_1}{||Ax_1||_X}$, nuestra desigualdad se convierte en

$$4||Ax_1||_X \leq 4\alpha(A)$$

que tomando supremo con $||x_1||_X = 1$, nos da $||A|| \leq \alpha(A)$ □

Enunciemos un resultado sobre el espectro de los operadores autoadjuntos y compactos.

Teorema 2.2.1. *Sea $A : X \to X$ un operador compacto y auto-adjunto. Entonces, A tiene a lo sumo contables autovalores no nulos $\lambda_n, n \in \mathbb{N}$, que son reales, y esta sucesión no tiene punto de acumulación no nulo. Más aún, hay una base ortonormal $\{v_n : n \in \mathbb{N}\}$ de $Ker(A)^{\perp}$, para la cual $Av_n = \lambda v_n, n \in \mathbb{N}$.*

Demostración. Si $A = 0$, entonces no hay nada que demostrar. Por lo tanto, podemos suponer que $\|A\| \neq 0$. Usando la notación de la proposición anterior, sea $(x_k)_k \subset X$ una sucesión con $\|x_k\|_X = 1$ para todo $k \in \mathbb{N}$, tal que

$$|\langle x_k, Ax_k \rangle_X| \longrightarrow \alpha(A) = \|A\|, \quad k \to \infty.$$

Dado que A es autoadjunto, estos productos escalares resultan todos reales y, por lo tanto, sus únicos posibles puntos de acumulación son $\pm\|A\|$. Tomando una subsucesión (si es necesario), podemos suponer sin pérdida de generalidad que

$$\langle x_k, Ax_k \rangle_X \to \lambda_1, \quad k \to \infty, \tag{2.3}$$

donde $\lambda_1 \in \{\pm\|A\|\}$. Como la sucesión $(x_k)_k$ está acotada, tiene una subsucesión (que, sin pérdida de generalidad, denotaremos con la misma notación) tal que existe $v_1 \in X$ con

$$x_k \rightharpoonup v_1 \quad \text{y} \quad Ax_k \to Av_1, \quad k \to \infty,$$

notando que $\rightharpoonup$ indica convergencia de tipo débil.

Aquí hemos utilizado la compacidad de A. En vista de (2.3), v_1 debe ser distinto de cero, y

$$\|Ax_k - \lambda_1 x_k\|_X^2 = \|Ax_k\|_X^2 - 2\lambda_1 \langle x_k, Ax_k\rangle_X + \lambda_1^2 \|x_k\|_X^2$$
$$\leq \|A\|^2 - 2\lambda_1 \langle x_k, Ax_k\rangle_X + \lambda_1^2 = 2\lambda_1 (\lambda_1 - \langle x_k, Ax_k\rangle_X),$$

lo cual tiende a cero cuando $k \to \infty$. Se sigue que

$$\lambda_1 x_k = (\lambda_1 x_k - Ax_k) + (Ax_k - Av_1) + Av_1 \longrightarrow Av_1,$$

cuando $k \to \infty$. Pero dado que $x_k \rightharpoonup v_1$, hemos establecido que v_1 es una "autofunción" de norma unitaria de A asociada al autovalor λ_1.

A continuación, introducimos $X_2 = \text{span}\{v_1\}^\perp \subset X$ y observamos que

$$\langle Ax, v_1\rangle_X = \langle x, Av_1\rangle_X = \lambda_1 \langle x, v_1\rangle_X = 0 \quad \forall x \in X_2,$$

es decir, X_2 es un subespacio invariante para A. Podemos definir entonces

$$A_2 = A|_{X_2} \in \mathcal{L}(X_2) := \mathcal{L}(X_2, X_2),$$

y dado que $X_2 \subset X$, la norma correspondiente de A_2 está acotada por $|\lambda_1| = \|A\|$. A menos que $A_2 = 0$, podemos proceder como antes y encontrar $\lambda_2 \in \mathbb{R}$ y un correspondiente $v_2 \in X_2 \subset X$, tal que

$$Av_2 = A_2 v_2 = \lambda_2 v_2.$$

Por construcción, tenemos $v_2 \perp v_1$ y $0 < |\lambda_2| \leq |\lambda_1|$.

Repitiendo este procedimiento inductivamente, obtenemos una sucesión decreciente de subespacios

$$X = X_1 \supset X_2 \supset X_3 \supset \cdots, \quad X_n = \text{span}\{v_1, \ldots, v_{n-1}\}^\perp,$$

con restricciones $A_n = A|_{X_n} \in \mathcal{L}(X_n)$ y una sucesión de valores propios reales $(\lambda_n)_n$ con $|\lambda_n| = \|A_n\|$ y

$$|\lambda_1| \geq |\lambda_2| \geq \cdots,$$

con un sistema ortonormal correspondiente $\{v_n\}_n$ de "autofunciones" de A.

Este procedimiento termina si la restricción A_n de A a algún X_n se vuelve cero; en este caso, el operador tiene rango finito con un número finito de valores propios distintos de cero.

En caso contrario, los autovalores no pueden acumularse en algún $\lambda \neq 0$, es decir, no existe $\delta > 0$ tal que $|\lambda_n| \geq \delta$ para todo $n \in \mathbb{N}$. De otro modo, $Av_n \to 0$ cuando $n \to \infty$, pues todo sistema ortogonal $\{v_n : n \in \mathbb{N}\}$ converge débilmente a cero (como consecuencia de la Desigualdad de Bessel). Esto produce una contradicción, ya que

$$\|Av_n\|_X = |\lambda_n| \, \|v_n\|_X = |\lambda_n| \geq \delta > 0.$$

Finalmente, queda por demostrar que $\{v_n : n \in \mathbb{N}\}$ es una base ortonormal de $Ker(A)^\perp$. Esto es evidente cuando A tiene rango finito. En el caso no degenerado, si $x \in Ker(A)^\perp$, consideramos

$$x' = \sum_{n=1}^{\infty} \langle v_n, x \rangle_X \, v_n \in Ker(A)^\perp.$$

Dado que $x - x'$ es ortogonal a todos los v_n, se sigue que $x - x' \in Ker(A)$, lo que implica $x' = x$, mostrando que $\{v_n\}_n$ es una base ortonormal de $Ker(A)^\perp$. □

Podemos deducir la descomposición en valores singulares utilizando el Teorema 2.2.1.

Teorema 2.2.2 (SVD)**.** *Sea A un operador compacto entre espacios de Hilbert X e Y, entonces puede escribirse de la forma*

$$Ax = \sum_{n=1}^{\infty} \sigma_n \langle v_n, x \rangle_X \, u_n, \quad x \in X$$

donde $||A|| = \sigma_1 \geq \sigma_2 \geq ...$ son sus valores singulares no negativos, $\{u_n\}_n, \{v_n\}_n$ son las bases ortonormales de $\overline{Rango(A)}$ y $Ker(A)^\perp$ respectivamente, y los vectores de las bases se relacionan con

$$Av_n = \sigma_n u_n \quad y \quad A^* u_n = \sigma_n^{-1} v_n$$

Esta se llama la expansión en valores singulares del operador A.

Demostración. Sabemos que A^*A es un operador compacto y auto-adjunto. Luego, podemos aplicarle el Teorema 2.2.1. Como A^*A es semi-definido positivo, sus autovalores no nulos $(\lambda_n)_n$ son positivos y $\lambda_1 = ||A^*A|| = ||A||^2$. Además los valores $\sigma_n = \sqrt{\lambda_n}, \quad n = 1, 2, ...,$ con $\sigma_1 = ||A||$ son los que llamaremos *valores singulares* de A. El sistema de "autofunciones" $\{v_n\}_n \subset X$ de A^*A construido en 2.2.1 es una base ortonormal de $Ker(A^*A)^\perp = Ker(A)^\perp$ y son las *funciones singulares* a derecha de A. Las *funciones singulares* a izquierda se definen como $u_n = \frac{1}{\sigma_n} A v_n, \quad n \in \mathbb{N}$. Y cumplen

$$\langle u_n, u_m \rangle_Y = \frac{1}{\sigma_n \sigma_m} \langle v_n, A^*A v_m \rangle_X = \frac{\lambda_m}{\sigma_n \sigma_m} \delta_{nm} = \delta_{nm}$$

con $n, m \in \mathbb{N}$, donde δ_{nm} es la delta de Kronecker (forman un sistema ortonormal). Luego,

$$A^* u_n = \frac{1}{\sigma_n} A^*A v_n = \sigma_n v_n, \quad n \in \mathbb{N}$$

y como cada $x \in X$ admite una expansión

$$x = x_0 + \sum_{n=1}^{\infty} \langle v_n, x \rangle_X v_n$$

con $x_0 \in Ker(A)$, cualquier $y \in Rango(A)$ puede expandirse como

$$y = Ax = \sum_{n=1}^{\infty} \langle v_n, x \rangle_X A v_n = \sum_{n=1}^{\infty} \sigma_n \langle v_n, x \rangle_X u_n,$$

donde $x \in X$ es cualquier preimagen de y □

Sean $r = dim(Rango(A))$ y notando como s_k a las raíces cuadradas de los autovalores no nulos de los operadores $A^*A \in \mathcal{L}(X, X), AA^* \in \mathcal{L}(Y, Y)$, en orden decreciente con multiplicidad. A^* es el operador adjunto de A y $\{v_k\}, \{u_k\}$ son sistemas ortonormales de autovectores de A^*A y AA^* respectivamente. Así,

$$Ax = \sum_{k=1}^{r} s_k u_k \langle x, v_k \rangle$$

$$A^*A v_k = s_k^2 v_k, \quad AA^* u_k = s_k^2 u_k, \quad ||u_k||_Y = ||v_k||_X = 1, \quad k = 1, 2, ..., r.$$

Además, como A^* es compacto, vale que:

$$A^*y = \sum_{k=1}^{r} s_k v_k \langle y, u_k \rangle$$

y

$$Av_k = s_k u_k, \quad A^*u_k = s_k v_k.$$

Si tenemos una cantidad infinita de valores singulares s_k, entonces $r = \infty$. La compacidad, el Teorema 2.2.1 y el hecho de que los autovectores sean ortogonales, nos indica que

$$\lim_{k\to\infty} s_k = 0.$$

Y esto nos lleva a que el rango de A no es cerrado, pues, para cada $k = 1, 2, ...$

$$0 \leq \inf_{\substack{v\in Ker(A)^\perp \\ \|v\|_X=1}} \|Av\|_Y \leq ||Av_k||_Y = s_k||u_k||_Y = s_k \longrightarrow 0.$$

Por lo cual,

$$\inf_{\substack{v\in N(A)^\perp \\ \|v\|_X=1}} \|Av\|_Y = 0.$$

Lo cual nos indica que en el caso de tener un operador compacto en dimensión infinita, utilizando la propiedad enunciada antes, $Rango(A)$ no resulta cerrado. Esto lleva a tener inestabilidad en el problema $Ax = y$, ya que pequeñas perturbaciones del lado derecho de la ecuación pueden llevarnos a tener una ecuación sin solución.

Por otro lado en la práctica, en general, nos es dada una ecuación $Ax = y$, y aunque $y \notin Rango(A)$, debemos "asignar" un elemento x a cada par (A, y) que sea una aproximación a la solución ideal. Por ello, para tener una buena relación entre la realidad y el modelo, invocamos a la solución por *cuadrados mínimos* $\overline{x}$, que verificará.

$$||A\overline{x} - y|| = \text{ínf}\{||Ax - y||, x \in X\}.$$

Si $Ker(A) \neq \{0\}$, entonces la solución por cuadrados mínimos no es única, ya que $\forall v \in Ker(A), v \neq 0, \quad A(\overline{x} + v) = A\overline{x}$. En este caso, podemos buscar una que minimice la

norma por mínimos cuadrados $x^\dagger$, tal que $x^\dagger \in Ker(A)^\perp$ y

$$||Ax^\dagger - y|| = \text{ínf}\{||Ax = y||, x \in X\}.$$

Sea Q el proyector ortogonal en $\overline{Rango(A)}$. Entonces

$$||Ax^\dagger - y||^2 = ||Ax^\dagger - Qy + (I-Q)y||^2 = ||Ax^\dagger - Qy||^2 + ||(I-Q)y||^2,$$

pues $Ax^\dagger \in Rango(A) \implies Ax^\dagger \in \overline{Rango(A)} \implies Ax^\dagger - Qy \in \overline{Rango(A)}$, y $(I-Q)y \in \overline{Rango(A)}^\perp$. Teniendo en cuenta que la segunda parte de la suma no depende de $x^\dagger$, se puede ver que si $Qy \in Rango(A)$ entonces $x^\dagger$ es solución de $Ax = Qy$. En la práctica, no siempre es posible construir Q, pero la siguiente proposición nos garantizará que la ecuación puede ser transformada a otra equivalente que no dependa de Q.

Proposición 2.4. *Las ecuaciones $Ax = Qy$ y $A^*Ax = A^*y$ tienen el mismo conjunto solución.*

Demostración. Sea x_0 solución de $A^*Ax = A^*y$. Entonces, para cualquier $v \in X$, tenemos que $\langle A^*Ax_0 - A^*y, v\rangle = 0$, donde $\langle\rangle$ es el producto interno en X. Por definición de operador adjunto, tenemos que

$$\langle A^*Ax_0 - A^*Y, v\rangle = \langle Ax_0 - y, Av\rangle = 0,$$

lo cual implica que $Ax_0 - y \in Rango(A)^\perp \subset \overline{Rango(A)}^\perp$. Entonces

$$Q(Ax_0 - y) = QAx_0 - Qy = Ax_0 - Qy = 0.$$

Ya que $Ax_0 \in Rango(A)$ y entonces $QAx_0 = Ax_0$. Luego, x_0 es solución de $Ax = Qy$.

Ahora recíprocamente supongamos que x_0 es solución de $Ax = Qy$. Primero, veamos que Q es auto-adjunto. Para cualquier x, $x - Px$ pertenece al núcleo, pues $Q(x - Qx) = Qx - Qx = 0$. Así, para cualesquiera u, v vale que $\langle Qu, v - Qv\rangle = \langle u - Qu, Qv\rangle = 0$, y se deduce que es auto-adjunto. Utilizando que $Q = Q^*$ por ser proyector ortogonal y que

$QA = A$, tenemos que

$$A^* = (QA)^* = A^*Q^* = A^*Q,$$

y luego aplicando $A^* = A^*Q$ a ambos lados de la ecuación $Ax_0 = Qy$ obtenemos que es solución de $A^*Ax = A^*y$.

$$A^*Ax_0 = A^*Qy = A^*y$$

□

Ahora intentemos formular una condición necesaria y suficiente para que haya solución de $A^*Ax = A^*y$ en el espacio X.

Proposición 2.5 (Criterio de Picard). *Sea $A \in \mathcal{L}(X, Y)$ un operador compacto con $dim(Rango(A)) = \infty$ y con la expansión en valores singulares dada por*

$$Ax = \sum_{k=1}^{\infty} s_k u_k \langle v_k, x \rangle.$$

Entonces la ecuación $A^*Ax = A^*y$ tiene solución en X (o similarmente $Qy \in Rango(A)$, donde Q es el proyector ortogonal en $\overline{Rango(A)}$) si y solo si

$$\sum_{k=1}^{\infty} s_k^{-2} \langle u_k, y \rangle^2 < \infty.$$

Bajo esta condición, la solución de $A^*Ax = A^*y$ que minimiza la norma tiene la forma

$$x^\dagger = \sum_{k=1}^{\infty} s_k^{-1} v_k \langle u_k, y \rangle.$$

Demostración. Utilizando que $A^*u_k = s_k v_k$ tenemos que

$$A^*Ax = \sum_{k=1}^{\infty} s_k^2 v_k \langle v_k, x \rangle, \quad A^*y = \sum_{k=1}^{\infty} s_k v_k \langle u_k, y \rangle$$

Por simplicidad, utilizamos el abuso de notación en el producto interno con $\langle\rangle$ para ambos espacios de Hilbert. Los $\{v_k\}$ son un sistema ortonormal, por lo que son linealmente independientes. Así, la ecuación $A^*Ax = A^*y$ resulta equivalente a

$$\langle v_k, x \rangle = s_k^{-1} \langle u_k, y \rangle, \quad k = 1, 2, \dots .$$

Como los coeficientes de Fourier $\langle v_k, x\rangle$ definen un elemento de un espacio de Hilbert si y solo si $\sum_{k=1}^{\infty} \langle v_k, x\rangle^2 < \infty$ (conocida propiedad que se desprende de la Desigualdad de Bessel, y puede hallarse en muchos libros de análisis funcional, como por ejemplo [23]), entonces,

$$\sum_{k=1}^{\infty} \langle v_k, x\rangle^2 = \sum_{k=1}^{\infty} s_k^{-2} \langle u_k, y\rangle^2 < \infty.$$

Y llegamos a lo pedido. □

Podemos notar que la proposición previa, en el fondo, define subyacentemente un operador $A^\dagger$ que toma elementos en el espacio

$$Dom(A^\dagger) = \{y : y \in Y; Qy \in Rango(A)\}$$

y va al espacio X. Este operador algunas veces es nombrado como la *inversa generalizada de Moore-Penrose*. Esta asigna a cada elemento $y \in Dom(A^\dagger)$, el elemento

$$x^\dagger = A^\dagger y = \sum_{k=1}^{\infty} v_k s_k^{-1} \langle u_k, y\rangle,$$

que resulta ser la solución de norma mínima de $A^*Ax = A^*y$. Así, para $y \in Dom(A^\dagger)$, la inversa generalizada de Moore-Penrose puede representarse como:

$$A^\dagger = (A^*A)^{-1}A^*.$$

2.3. Regularización de la inversa generalizada

Ahora, si tenemos un operador compacto A con rango de dimensión infinita, $A^\dagger$ resulta un operador lineal pero no acotado de $Dom(A^\dagger)$ a X ($s_k \longrightarrow 0$). Esto nos indica que no estamos en condiciones de resolver la ecuación $A^*Ax = A^*y$ de manera estable. Cualquier perturbación pequeña que pudiésemos realizar en y, por ejemplo, podría cambiar de manera drástica la solución. En ese contexto, lo máximo que podemos buscar es obtener una manera estable de aproximar la solución de $A^*Ax = A^*y$.

Podríamos pensar que la no acotación dada por el factor $(A^*A)^{-1}$ puede ser considerada como el valor por el operador t^{-1} en el "punto" A^*A, pero que podría aproximarse por una

función acotada $g(t)$ tal que $g(A^*A)A^*y$ esté lo suficientemente cerca de $(A^*A)^{-1}A^*y$.
Necesitaremos algunos resultados de Análisis Funcional
Como mostramos en la sección anterior, cualquier operador auto-adjunto B admite una descomposición de valores singulares de la forma (si $r_B = dim(Rango(B))$).

$$B = \sum_{k=1}^{r_B} \lambda_k \langle u_k, . \rangle u_k,$$

con $||B|| = \lambda_1 \geq \lambda_2 \geq ...\lambda_k \geq ... > 0$.
Dado cualquier operador B compacto, auto-adjunto, y no negativo, si consideramos cualquier $f : [0, ||B||] \longrightarrow \mathbb{R}$ acotada (tal que $||B|| \leq a$), podemos considerar el operador $f(B) = \sum_{k=1}^{r_B} f(\lambda_k)\langle u_k, . \rangle u_k$, que determinará un operador lineal acotado de X en X, y que además cumple (lo demostraremos en las próximas líneas)

$$\left\| \sum_{k=1}^{r_B} f(\lambda_k)\langle u_k, . \rangle u_k \right\|_{X \to X} \leq \sup_{\lambda \in [0, ||B||]} |f(\lambda)|.$$

Haremos uso de las siguientes propiedades

Proposición 2.6. *Con las condiciones anteriores:*

1. $||f(B)||_{X \to X} \leq \sup_{\lambda \in [0, ||B||]} |f(\lambda)|$
2. $||f(A^*A)A^*||_{Y \to X} \leq \sup_{\lambda \in [0, ||A||^2]} \sqrt{\lambda}|f(\lambda)|$

Demostración. Para la primera parte podemos utilizar la desigualdad de Bessel y la definición de la norma

$$\begin{aligned}
||f(B)|| &= \left\| \sum_{k=1}^{r_B} f(\lambda_k)\langle u_k, \cdot \rangle u_k \right\|_{X \to X} \\
&= \sup_{||v|| \leq 1} \left\| \sum_{k=1}^{r_B} f(\lambda_k) u_k \langle u_k, v \rangle \right\|_X \\
&= \sup_{||v|| \leq 1} \left(\sum_{k=1}^{r_B} [f(\lambda_k)]^2 \langle u_k, v \rangle^2 \right)^{1/2} \\
&\leq \sup_{\lambda \in [0, ||B||]} |f(\lambda)| \sup_{||v|| \leq 1} \left(\sum_{k=1}^{r_B} |\langle u_k, v \rangle|^2 \right)^{1/2} \\
&\leq \sup_{\lambda \in [0, ||B||]} |f(\lambda)|.
\end{aligned}$$

Para la segunda parte sea $A = \sum_{k=1}^{r_A} s_k u_k \langle v_k, . \rangle$ la expansión en valores singulares de A ($r_A = dim(Rango(A))$). Entonces, $A^* = \sum_{k=1}^{r_A} s_k v_k \langle u_k, . \rangle$ y

$$f(A^*A) = \sum_{k=1}^{r_A} f(s_k^2) \langle v_k, . \rangle v_k$$

$$f(A^*A)A^* = \sum_{k=1}^{r_A} f(s_k^2) s_k \langle u_k, . \rangle v_k$$

Luego

$$\begin{aligned} ||f(A^*A)A^*||_{Y \to X} &= \sup_{||v|| \leq 1} \left\| \sum_{k=1}^{r_A} f(s_k^2) s_k \langle u_k, v \rangle v_k \right\|_Y \\ &= \sup_{||v|| \leq 1} \left(\sum_{k=1}^{r_A} |f(s_k^2) s_k|^2 \langle u_k, v \rangle^2 \right)^{1/2} \\ &\leq \sup_k s_k |f(s_k^2)| \sup_{||v|| \leq 1} \left(\sum_{k=1}^{r_A} \langle u_k, v \rangle^2 \right)^{1/2} \\ &\leq \sup_{\lambda \in [0, ||A||^2]} \sqrt{\lambda} |f(\lambda)|. \end{aligned}$$

Donde el último paso se justifica notando que $s_k^2 \in [0, ||A^*A||] \subset [0, ||A||^2]$

□

Volviendo al plan de elegir una función g, la idea sería tomarla tal que para cualquier y para el cual se cumpla el Criterio de Picard enunciado anteriormente, $A^\dagger y = (A^*A)^{-1}A^*y$ esté suficientemente cerca de $g(A^*A)A^*y$.

A primera vista, pareciera que bastaría elegir $g(\lambda)$ como una buena aproximación de λ^{-1}, para cualquier $\lambda \in (0, ||A||^2]$. Aunque, dicha aproximación no podría darse con una única función, pues, para cada g acotada $|\lambda^{-1} - g(\lambda)| \xrightarrow{\lambda \to 0} \infty$. Por ello, surge la idea de utilizar una familia $\{g_\alpha(\lambda)\}$ de funciones acotadas, indexadas por el parámetro α (parámetro regularizador), tal que

$$\lim_{\alpha \to 0} |\lambda^{-1} - g_\alpha(\lambda)| = 0, \quad \forall \lambda \in (0, ||A||^2).$$

También podríamos considerar similarmente una familia indexada por un parámetro r de manera que se cumpla el límite anterior con $r \to \infty$, ya que podemos reducirla al caso

anterior considerando el cambio de variable $\alpha = \frac{1}{r}$.

Ahora vemos qué condiciones debemos imponerle a esta familia de funciones.

Proposición 2.7. *Sean* $g_\alpha(\lambda) : [0, ||A||^2] \to \mathbb{R}, \quad \alpha > 0$ *funciones que cumplen lo siguiente:*

1. $\exists c : |\lambda g_\alpha(\lambda)| \leq c$,
2. $\forall \lambda \in (0, ||A||^2], \quad \lim_{\alpha \to 0} g_\alpha(\lambda) = \frac{1}{\lambda}$.

Entonces para cualquier $y \in Y$ *(lado derecho de nuestra ecuación) que cumple el Criterio de Picard, vale*

$$\lim_{\alpha \to 0} ||A^\dagger y - g_\alpha(A^*A)A^*y||_X = 0.$$

Demostración. Sea $A = \sum_{k=1}^{\infty} s_k u_k \langle v_k, . \rangle$ la expansión en valores singulares del operador A. Asumimos sin pérdida de generalidad que $Rango(A) = \infty$. El Criterio de Picard nos afirma que

$$\sum_{k=1}^{\infty} s_k^{-2} \langle u_k, y \rangle^2 < \infty.$$

Por lo tanto, para cualquier $\varepsilon > 0, \quad \exists k_0 = k_0(\varepsilon)$, tal que

$$\sum_{k=k_0+1}^{\infty} s_k^{-2} \langle u_k, y \rangle^2 < \frac{\varepsilon^2}{2(1+c)^2}.$$

Así,

$$\begin{aligned} ||A^\dagger y - g_\alpha(A^*A)A^*y||^2 &= \sum_{k=1}^{\infty} [s_k^{-1} - g_\alpha(s_k^2)s_k]^2 \langle u_k, y \rangle^2 \\ &= \sum_{k=1}^{k_0} [s_k^{-1} - g_\alpha(s_k^2)s_k]^2 \langle u_k, y \rangle^2 + \sum_{k=k_0+1}^{\infty} [s_k^{-1} - g_\alpha(s_k^2)s_k]^2 \langle u_k, y \rangle^2 \\ &= T_1 + T_2. \end{aligned}$$

Acotemos T_1 y T_2 por separado.

$$T_2 = \sum_{k=k_0+1}^{\infty} s_k^2[s_k^{-1} - g_\alpha(s_k^2)s_k]^2 \frac{\langle u_k, y\rangle^2}{s_k^2} = \sum_{k=k_0+1}^{\infty} [1 - g_\alpha(s_k^2)s_k^2]^2 \frac{\langle u_k, y\rangle^2}{s_k^2}$$
$$\leq \sup_{\lambda\in[0,||A||^2]} (1 + |g(\lambda)\lambda|)^2 \sum_{k=k_0+1}^{\infty} s_k^{-2}\langle u_k, y\rangle^2$$
$$\leq (1+c)^2 \frac{\varepsilon^2}{2(1+c)^2} \leq \frac{\varepsilon^2}{2}.$$

Para T_1 utilizaremos la segunda hipótesis de la proposición. $\forall\varepsilon > 0, \exists\alpha_k = \alpha_k(\varepsilon)$, tal que para $\alpha \leq \alpha_k$ y $\lambda = s_k^2$

$$\left|\frac{1}{s_k^2} - g_\alpha(s_k^2)\right|^2 \leq \frac{\varepsilon^2}{2||A^*y||^2}.$$

Si tomamos $\alpha \leq \text{mín}\{\alpha_k\}_{k=1}^{k_0}$, la cota anterior vale para cualquier k. Luego

$$T_1 = \sum_{k=1}^{k_0} \left[\frac{1}{s_k^2} - g_\alpha(s_k^2)\right]^2 s_k^2\langle u_k, y\rangle^2 \leq \frac{\varepsilon^2}{2||A^*y||^2} \sum_{k=1}^{k_0} s_k^2\langle u_k, y\rangle^2$$
$$\leq \frac{\varepsilon^2}{2||A^*y||^2}||A^*y||^2 \leq \frac{\varepsilon^2}{2}.$$

Así, $\forall\varepsilon > 0$ podemos hallar un $\alpha = \alpha(\varepsilon, y)$, tal que

$$||A^\dagger y - g_\alpha(A^*A)A^*y||_X \leq \varepsilon.$$

Y probamos el límite buscado □

2.4. Calificación de un Método de Regularización

En la sección anterior vimos que las técnicas de regularización convierten el método mal puesto en uno bien condicionado utilizando un parámetro de regularización. Además cuando ese parámetro tiene a cero converge a la solución del problema original. También en la práctica existe un error en el dato y, por lo tanto es importante elegir el parámetro óptimo que minimice todos los errores. Dentro del campo de métodos de regularización, existen diversas técnicas para encontrar ese parámetro óptimo como el "Principio de dis-

crepancia" [24], el "Cross validation" [25] o el "Criterio de la L-curva" [26].

En esta tesis vamos a definir un concepto desarrollado recientemente denominado "calificación" y la "one half condition" que nos permitirá obtener cotas de los errores para toda esta familia de métodos de regularización.

Para que $g_\alpha(A^*A)A^*y$ converja a $A^\dagger y$ cuando $\alpha \to 0$, es suficiente que se cumpla la siguiente condición:

$$\lim_{\alpha\to 0} \sup_{\lambda\in[0,\|A\|^2]} |1-\lambda g_\alpha(\lambda)| = 0.$$

Esto equivale a decir que $g_\alpha(\lambda)$ converge uniformemente a $\frac{1}{\lambda}$ en la norma suprema ponderada con el peso λ, definida como

$$\left\| \frac{1}{\lambda} - g_\alpha(\lambda) \right\|_\lambda = \sup_{\lambda\in[0,a]} \lambda \left| \frac{1}{\lambda} - g_\alpha(\lambda) \right| = \sup_\lambda |1-\lambda g_\alpha(\lambda)|\,.$$

Al considerar la convergencia en dicha norma, podemos hablar sobre la "tasa de convergencia" de manera uniforme con respecto a λ. Cabe destacar que el uso de normas ponderadas es común en la teoría de aproximación cuando se estudia la aproximación de funciones con singularidades. En tal caso, generalmente se utilizan pesos que se anulan en los puntos de singularidad.
En la teoría de problemas mal puestos, se busca la aproximación de la función λ^{-1} que tiene una singularidad en cero. Por lo tanto, resultará natural considerar las normas supremo ponderadas con pesos de la forma λ^p. La tasa de aproximación de $\frac{1}{\lambda}$ por $g_\alpha(\lambda)$ en la norma supremo ponderada con peso λ^p se mide frente a la tasa α^p cuando $\alpha \to 0, p \geq 1$.

Recordando la ecuación (2.2) de la segunda sección, observamos que en la práctica lo que tenemos en realidad es un error en el dato. En general, lo que vamos a tener es un y^δ donde $\|y - y^\delta\| \leq \delta$ para algún $\delta > 0$. Por ello, nuestro siguiente objetivo será estimar el error entre $A^\dagger y$ y la solución obtenida regularizando el operador $A^\dagger$ usando como dato y^δ. Para obtener una estimación razonable para el error, pediremos una condición a cumplir y calcularemos un parámetro que actuará como un identificador de mejora en cuanto a orden de convergencia y error cometido. Estas cuestiones resultarán naturales cuando aparezcan en la siguiente cuenta, donde buscamos dar con una estimación del "error". Buscamos

acotar

$$||A^{\dagger}y - g_{\alpha}(A^*A)A^*y^{\delta}|| \leq ||A^{\dagger}y - g_{\alpha}(A^*A)A^*y|| + ||g_{\alpha}(A^*A)A^*y - g_{\alpha}(A^*A)Ay^{\delta}|| \quad (2.4)$$

Ahora utilizamos que

$$||g_{\alpha}(A^*A)A^*y - g_{\alpha}(A^*A)Ay^{\delta}|| \leq \delta||g_{\alpha}(A^*A)A^*|| \leq \delta \sup_{\lambda\in[0,||A||^2]} \sqrt{\lambda}|g_{\alpha}(\lambda)|,$$

donde la última desigualdad viene dada por la segunda propiedad de la Proposición 2.6. Por otro lado, aplicando el Teorema 2, en la página 798 de [27], se obtiene que

$$||A^{\dagger}y - g_{\alpha}(A^*A)A^*y|| \leq c \cdot \sup_{\lambda\in[0,||A||^2]} |1 - \lambda g_{\alpha}(\lambda)|\lambda^p$$

con c una constante.

Estas conclusiones nos motivan a definir *calificación* y a pedir que valga la *one-half condition*.

Definición 2.6. *La calificación de la regularización generada por la familia* $\{g_{\alpha}(\lambda)\}$ *es el máximo* p*, tal que*

$$\sup_{\lambda\in[0,\|A\|^2]} \lambda^p \left|1 - \lambda g_{\alpha}(\lambda)\right| \leq \gamma_p \alpha^p,$$

donde γ_p *depende solo de* p *y* g*.*

En general, vamos a pedir que nuestros métodos de regularización generados por familias $\{g_{\alpha}\}$ cumplan la *"one-half condition"*.

Definición 2.7 (One-half condition)**.**

$$\exists \gamma_{-1/2} : \sup_{\lambda\in(0,||A||^2]} \left|\lambda^{1/2} g_{\alpha}(\lambda)\right| \leq \frac{\gamma_{-1/2}}{\sqrt{\alpha}}.$$

Observar que si volvemos a la cuenta (2.4) y a las últimas cotas averiguadas, ahora obtenemos la cota

$$c\gamma_p \alpha^p + \delta\gamma_{-1/2}\frac{1}{\sqrt{\alpha}},$$

que se resume en

$$C\left(\alpha^p + \frac{\delta}{\sqrt{\alpha}}\right).$$

Por lo tanto, cuanto mayor es el p menor va a ser el error debido a la aproximación del operador.

Por otro lado, la one-half condition nos agrega un término que depende del error del dato. En los casos en donde ese dato es conocido se podría estimar cual es el mejor α calculando el valor mínimo de ese termino. En este caso, ese mínimo se realiza en $\alpha = C\delta^{1/(p+1/2)}$.

Habiendo dado todas estas nociones previas, en lo que sigue veremos algunos ejemplos clásicos de métodos de regularización conocidos para resolver el problema de la deconvolución, junto con las familias respectivas que los generan. Probaremos para cada uno de estos ejemplos que se cumple la one-half y calcularemos su calificación.

Ejemplo 2.1 (Spectral cut-off method)**.** *Este método de regularización es generado por la familia* $\{g_\alpha(\lambda)\}$

$$g_\alpha(\lambda) = \begin{cases} \frac{1}{\lambda}, & \alpha \leq \lambda < \infty \\ 0, & 0 \leq \lambda < \alpha \end{cases}.$$

Es fácil ver que

$$\sup_{\lambda \in [0,\infty)} |\lambda g_\alpha(\lambda)| = 1,$$

y que $\forall \lambda_0 \in (0,\infty), \varepsilon > 0, \exists \alpha_0 = \lambda_0$ *tal que para cualquier* $\alpha < \alpha_0$

$$\left|\frac{1}{\lambda_0} - g_\alpha(\lambda_0)\right| = 0 \leq \varepsilon$$

Lo cual implica que para cualquier $\lambda \in (0,\infty)$*,*

$$\lim_{\alpha \to 0} \left|\frac{1}{\lambda} - g_\alpha(\lambda)\right| = 0$$

Por lo tanto, todas las condiciones de la Proposición 2.7 se cumplen. Así, para cualquier y *que cumpla el criterio de Picard, tenemos*

$$\lim_{\alpha\to 0} \left\| A^{\dagger} y - g_{\alpha}\left(A^{*}A\right) A^{*} y \right\|_{Y} = 0.$$

A su vez, veamos que se cumple la one-half condition con $\gamma_{-1/2} = 1$

$$\sup_{\lambda\in(0,\infty)} \lambda^{1/2} g_{\alpha}(\lambda) = \sup_{\lambda\in[\alpha,\infty)} \frac{1}{\lambda^{1/2}} \leq \frac{1}{\sqrt{\alpha}}.$$

Para estimar la calificación del método, notemos que para cualquier $p \geq 0$

$$\sup_{\lambda\in[0,\infty)} \lambda^{p} |1 - \lambda g_{\alpha}(\lambda)| = \sup_{\lambda\in[0,\alpha)} \lambda^{p} = \alpha^{p},$$

Por ende, no hay valor máximo de p *para el cual vale. Entonces, la regularización por corte espectral tiene calificación infinita. Sin embargo, este método solo es viable cuando se conoce el sistema* $\{s_k, u_k, v_k\}$ *del operador* A*, ya que en este caso* $g_{\alpha}\left(A^{*}A\right)$ *no es más que una porción de la expansión formal en valores singulares de* $\left(A^{*}A\right)^{-1} y$

$$g_{\alpha}\left(A^{*}A\right) A^{*} y = \sum_{k:s_k^2\geq\alpha} s_k^{-1} v_k \left\langle u_k, y \right\rangle$$

Ejemplo 2.2 (Tikhonov-Phillips)**.** *Este método viene generado por la familia* $\left\{ g_{\alpha}(\lambda) = \frac{1}{\alpha+\lambda} \right\}$*. A primera vista, podríamos señalar que, a diferencia del método previo, para implementar este método no necesitamos conocer el sistema que viene inducido en la expansión por valores singulares de* A*, pues*

$$x_{\alpha} = g_{\alpha}(A^{*}A)A^{*}y = (\alpha I + A^{*}A)^{-1}A^{*}y$$

no es otra cosa que la solución de

$$\alpha x + A^{*}Ax = A^{*}y$$

que puede resolverse al menos numéricamente sin necesidad de conocer el sistema involucrado en la expansión SVD. Más aún, podemos notar que lo anterior resulta en el problema variacional

$$\min_{x\in X}\{||Ax - y||_{Y}^{2} + \alpha||x||_{X}^{2}\},$$

Vamos a verificar que se cumplen las condiciones de la Proposición 2.7.

$$\sup_{\lambda\in[0,\infty)} |\lambda g_\alpha(\lambda)| = \sup_{\lambda\in[0,\infty)} \frac{\lambda}{\alpha+\lambda} = 1;$$
$$\forall \lambda \in (0,\infty), \quad \lim_{\alpha\to 0} \left|\frac{1}{\lambda} - g_\alpha(\lambda)\right| = \lim_{\alpha\to 0} \frac{\alpha}{\lambda(\alpha+\lambda)} = 0.$$

Verifiquemos ahora que $g_\alpha(\lambda) = \frac{1}{\alpha+\lambda}$ cumple con la condición "one-half" con la constante $\gamma_{-1/2} = \frac{1}{2}$ y hallemos la calificación del método. Para ello, hacemos uso de la siguiente observación:

Observación 2.2. *Para $p > 0$*

$$\frac{d\left(\frac{\lambda^p}{\alpha+\lambda}\right)}{d\lambda} = \frac{p\lambda^{p-1}(\alpha+\lambda) - \lambda^p}{(\alpha+\lambda)^2} = \frac{\lambda^{p-1}}{(\alpha+\lambda)^2}(p\alpha - \lambda(1-p))$$

Esto significa que para $p \in (0,1)$

$$\sup_{\lambda\in[0,\|A\|^2]} \frac{\lambda^p}{\alpha+\lambda} = \left(\frac{p}{1-p}\right)^p \frac{\alpha^p}{\frac{\alpha}{1-p}} = \alpha^{p-1}p^p(1-p)^{1-p},$$

mientras que para $p \geq 1$ y $\alpha < \|A\|^2$ suficientemente pequeño

$$\left(\frac{\lambda^p}{\lambda+\alpha}\right)' > 0 \Rightarrow \frac{1}{2}\|A\|^{2(p-1)} \leq \sup_{\lambda\in[0,\|A\|^2]} \frac{\lambda^p}{\alpha+\lambda} \leq \|A\|^{2(p-1)}.$$

Ahora, tenemos que se cumple la one-half condition

$$\sup_{\lambda\in[0,\|A\|^2]} \left|\lambda^{1/2} g_\alpha(\lambda)\right| = \sup_{\lambda\in[0,\|A\|^2]} \frac{\lambda^{1/2}}{\lambda+\alpha} = \frac{1}{2\sqrt{\alpha}}$$

y la calificación resulta igual a 1, pues para $p \geq 1$

$$\frac{1}{2}\alpha\|A\|^{2(p-1)} \leq \sup_{\lambda\in[0,\|A\|^2]} \lambda^p\,|1-\lambda g_\alpha(\lambda)| = \sup_{\lambda\in[0,\|A\|^2]} \alpha\frac{\lambda^p}{\lambda+\alpha} \leq \alpha\|A\|^{2(p-1)}$$

mientras que para $p \in (0,1)$

$$\sup_{\lambda\in[0,\|A\|^2]} \lambda^p\,|1-\lambda g_\alpha(\lambda)| = \sup_{\lambda\in[0,\|A\|^2]} \frac{\lambda^p\alpha}{\lambda+\alpha} = \alpha \sup_\lambda \frac{\lambda^p}{\lambda+\alpha} = \alpha^p p^p (1-p)^{1-p}$$

Ejemplo 2.3 (Tikhonov Iterativo)**.** *Pensemos por un instante que tenemos alguna información previa sobre el elemento $A^\dagger y$. Por ejemplo, supongamos que está "cerca" de un*

elemento $x_0 \in X$. Este x_0 será nuestra primera aproximación a $A^\dagger y$. En esta situación, resulta razonable cambiar el problema de minimización motivado en el Ejemplo 2.2 a

$$\min_{x \in X} \{||Ax - y||^2 + \alpha||x - x_0||^2\}.$$

Cambiando variables $x_1 = x - x_0$, llegamos al siguiente problema de minimización

$$\min_{x_1 \in X} \{||Ax_1 - (y - Ax_0)||^2 + \alpha||x_1||^2\},$$

que resulta el funcional original de Tikhonov, donde se reemplaza y por $y - Ax_0$. Del Ejemplo 2.2 nuevamente, tenemos que el mínimo del funcional anterior es solución de

$$\alpha x_1 + A^*Ax_1 = A^*(y - Ax_0).$$

Recordando quién era x_1 tenemos

$$\alpha x + A^*Ax = \alpha x_0 + A^*y.$$

En principio, x_0 puede tomar el papel del $x_\alpha = (\alpha I + A^*A)^{-1}A^*y$, que se correspondería con un inicial $x_0 = 0$. Al repetir este proceso iterativamente de manera que el paso previo tome el papel de elemento inicial para la siguiente aproximación, podríamos armar la iteración de la siguiente forma:

$$\begin{aligned}
x_{0,\alpha} &= 0 \\
x_{1,\alpha} &= x_\alpha \to \alpha x + A^*Ax = A^*y \\
x_{2,\alpha} &= \ldots \to \alpha x + A^*Ax = \alpha x_{1,\alpha} + A^*y \\
&\vdots \\
x_{m,\alpha} &= \ldots \to \alpha x + A^*Ax = \alpha x_{m-1,\alpha} + A^*y.
\end{aligned}$$

Resolviendo en cada paso y utilizando la representación dada por el método de Tikhonov-

Phillips en términos del operador $g_\alpha(A^*A)$, queda

$$
\begin{aligned}
x_{1,\alpha} &= g_\alpha\left(A^*A\right)A^*y \\
x_{2,\alpha} &= g_\alpha\left(A^*A\right)\left(\alpha x_{1,\alpha} + A^*y\right) = g_\alpha\left(A^*A\right)A^*y + \alpha g_\alpha^2\left(A^*A\right)A^*y \\
&\vdots \\
x_{m,\alpha} &= \sum_{k=1}^{m} \alpha^{k-1} g_\alpha^k\left(A^*A\right)A^*y = g_{m,\alpha}\left(A^*A\right)A^*y
\end{aligned}
$$

donde

$$
\begin{aligned}
g_{m,\alpha}(\lambda) &= \sum_{k=1}^{m} \alpha^{k-1} g_\alpha^k(\lambda) = g_\alpha(\lambda)\sum_{k=0}^{m-1} \alpha^k g_\alpha^k(\lambda) \\
&= g_\alpha(\lambda)\frac{1-\alpha^m g_\alpha^m(\lambda)}{1-\alpha g_\alpha(\lambda)}.
\end{aligned}
$$

Teniendo en cuenta que $g_\alpha(\lambda) = \frac{1}{\lambda+\alpha}, 1-\alpha g_\alpha(\lambda) = \frac{\lambda}{\lambda+\alpha}$ nos queda

$$
g_{m,\alpha}(\lambda) = \frac{1}{\lambda}\left(1 - \frac{\alpha^m}{(\lambda+\alpha)^m}\right).
$$

Estas serán las integrantes de la familia de funciones regularizadoras. La gran ventaja de este método es que tiene una calificación arbitrariamente grande. En efecto, si $p < m$, podemos utilizar el mismo argumento que en el Ejemplo 2.2 para ver que

$$
\begin{aligned}
\sup_{\lambda\in[0,\infty)} \lambda^p \left|1-\lambda g_{m,\alpha}(\lambda)\right| &= \sup_{\lambda\in[0,\infty)} \lambda^p \frac{\alpha^m}{(\lambda+\alpha)^m} = \sup_{\lambda} \lambda^p \left(1-\lambda(\lambda+\alpha)^{-1}\right)^m \\
&= \sup_{\lambda\in(0,\infty)} \left[\lambda^{\frac{p}{m}}\left(1-g_\alpha(\lambda)\lambda\right)\right]^m \\
&\leq \left[\left(\frac{p}{m}\right)^{\frac{p}{m}}\left(1-\frac{p}{m}\right)^{1-\frac{p}{m}}\alpha^{\frac{p}{m}}\right]^m \\
&= \left(\frac{p}{m}\right)^p \left(1-\frac{p}{m}\right)^{m-p}\alpha^p.
\end{aligned}
$$

Lo cual nos indica que después de la iteración número m, obtenemos un método con calificación m. Finalmente, verifiquemos que se cumple la one-half condition. Notemos que

$$
\left|g_{m,\alpha}(\lambda)\right| \leq \left|g_\alpha(\lambda)\right| \sum_{k=0}^{m-1} \alpha^k \left|g_\alpha(\lambda)\right|^k = \frac{1}{\lambda+\alpha}\sum_{k=0}^{m-1}\left(\frac{\alpha}{\alpha+\lambda}\right)^k \leq \frac{m}{\alpha}.
$$

Luego

$$\begin{aligned}\sup_{\lambda\in[0,\infty)} \lambda^{1/2}\left|g_{\alpha,m}(\lambda)\right| &= \sup_{\lambda} \frac{1}{\lambda^{1/2}}\left(1-\frac{\alpha^m}{(\lambda+\alpha)^m}\right)\\ &= \sup_{\lambda}\left[\frac{1}{\lambda}\left(1-\frac{\alpha^m}{(\alpha+\lambda)^m}\right)\right]^{1/2}\left[1-\frac{\alpha^m}{(\lambda+\alpha)^m}\right]^{1/2}\\ &\leq \sup_{\lambda}\left[\frac{1}{\lambda}\left(1-\frac{\alpha^m}{(\alpha+\lambda)^m}\right)\right]^{1/2}\\ &= \sup_{\lambda}\left[g_{\alpha,m}(\lambda)\right]^{1/2} \leq \frac{m^{1/2}}{\sqrt{\alpha}},\end{aligned}$$

y verificamos la one-half condition.

Ejemplo 2.4 (Landweber)**.** *Este método es otro ejemplo de método iterativo. La diferencia entre ambos radica en que, en 2.3 el número de iteraciones del método resultaba fijo y la regularización venía guiada por el parámetro α. En cambio, en este método, buscamos que el número de iteraciones a efectuar m sea utilizado como parámetro de regularización, como $\alpha = \frac{1}{m}$, o $m = \left\lfloor \frac{1}{\alpha} \right\rfloor$.*
La fórmula para las iteraciones de Landweber es

$$x_n = x_{n-1} - \mu A^*\left(Ax_{n-1} - y\right), \quad n = 1, 2, \ldots$$

donde μ es un número fijo en $(0, ||A||^{-2})$. Es fácil verificar que $x_m = g_\alpha\left(A^*A\right)A^*y$, con $\alpha = \frac{1}{m}$, y

$$g_\alpha(\lambda) = \sum_{k=0}^{m-1}(1-\mu\lambda)^k\mu = \frac{1}{\lambda}\left[1-(1-\mu\lambda)^m\right]$$

Y la one-half condition se verifica como antes

$$\sup_{\lambda\in[0,\|A\|^2]} \lambda^{1/2}\left|g_\alpha(\lambda)\right| \leq (\mu m)^{1/2} = \frac{\sqrt{\mu}}{\sqrt{\alpha}}, \quad \alpha = \frac{1}{m}.$$

Mejor aún, sin mucho esfuerzo puede verse que para cada $0 < p < \infty$,

$$\sup_{\lambda\in[0,||A||^2]} \lambda^p|1-\lambda g_\alpha(\lambda)| \leq \gamma_p m^{-p} = \gamma_p\alpha^p,$$

con $\gamma_p = (p/\mu e)^p$. Así, puede considerarse que el Landweber es un método con calificación arbitrariamente grande, pero con la salvedad de que $\gamma_p \to \infty$ si $p \to \infty$.

Capítulo 3

Los métodos SUPPOSe y DSUPPOSe

3.1. El método SUPPOSe

Como se motivó en la introducción con una descripción similar, la imagen adquirida S puede escribirse como,

$$S(\mathbf{x}) = R(\mathbf{x}) * I(\mathbf{x}) + \eta(\mathbf{x}) + B, \tag{3.1}$$

donde R es la estructura real que se desea observar, I es la PSF del sistema óptico, η un término de ruido de valor medio cero y B es un nivel de fondo constante. Aquí, $\mathbf{x} \in \mathbb{R}^2$ representa la coordenada en la cual se desarrolla la medición.

El algoritmo SUPPOSe [12] utiliza un enfoque alternativo a la deconvolución. En este método, en vez de introducir un término de regularización sobre la función a minimizar, se asume cierta estructura sobre el objeto. Más precisamente,se asume que la estructura real R puede aproximarse como una superposición de fuentes puntuales de igual intensidad α, a las que llama *fuentes virtuales*. Es decir que supone que la estructura real puede modelarse como un conjunto de puntos dispersos distribuidos en el espacio. De esta manera, estima la estructura real R como,

$$\tilde{R}(\mathbf{x}) = \alpha \sum_{k=1}^{N} \delta(\mathbf{x} - \boldsymbol{a}_k), \tag{3.2}$$

La matriz $\boldsymbol{A} = \{\boldsymbol{a}_k\}_{k=1}^{N} \in \mathbb{R}^{N\times 2}$ representa la posición de las N fuentes virtuales utilizadas. Vamos a darle la notación con tilde a variables o cantidades para indicar que son una aproximación de la misma variable sin tilde. A partir de dicho planteamiento con

la aproximación SUPPOSe, el problema se transformó en un problema de minimización irrestricto en $\mathbb{R}^N \times \mathbb{R}^2$, donde ahora las únicas incógnitas a hallar resultan ser las posiciones $\boldsymbol{a}_k$ de las N fuentes virtuales que minimizan la diferencia entre la imagen observada inicialmente S y la imagen reconstruida a través de estas fuentes.

Estas posiciones podrían tomar cualquier valor en $\mathbb{R}^2$, y también muchas fuentes virtuales podrían coincidir en una misma ubicación. Pero como las fuentes vienen dotadas con una intensidad fija e igual para todas, podemos reconstruir la intensidad en un sector específico de la imagen superponiendo una cantidad más grande o más pequeña de fuentes virtuales acumuladas en ese lugar.

A continuación describimos el método SUPPOSe para el caso donde el fondo B es despreciable, o conocido y restado. En este caso, aproximamos la imagen S por

$$\tilde{S}(\mathbf{x}) = \tilde{R}(\mathbf{x}) * \tilde{I}(\mathbf{x}) = \alpha \sum_{k=1}^{N} \tilde{I}(\mathbf{x} - \boldsymbol{a}_k), \tag{3.3}$$

que se deriva de (3.1) tomando $B = 0$ y aproximando la PSF del sistema óptico I por una estimación $\tilde{I}$, la cual puede ser de origen teórico o experimental.

Para hallar las posiciones de las fuentes virtuales que mejor aproximan al objeto, el método busca minimizar una función objetivo determinada. En este caso se toma la norma L^2 de la diferencia entre la imagen observada y la aproximación obtenida por SUPPOSe,

$$\chi^2 = ||S - \tilde{S}||^2 = \sum_{i=1}^{n} (S(x_i) - \tilde{S}(x_i))^2, \tag{3.4}$$

donde la sumatoria se realiza sobre todos los píxeles de la imagen (decimos n), y utilizamos la notación $x_i \in \mathbb{R}^2$, para representar un píxel. Fijado un valor para N y α, el problema de SUPPOSe consistirá en hallar el vector de posiciones $A = \{a_k\}_{k=1}^{N} \in \mathbb{R}^{N \times 2}$ que minimiza (3.4).

Podemos contemplar el caso donde la imagen que adquirimos tenga un fondo de señal B constante y de valor no conocido simplemente restando el valor medio de la imagen y

de la PSF,

$$S_{dev}(x) = S - \frac{1}{n}\sum_{i=1}^{n} S(x_i), \tag{3.5}$$

$$\tilde{I}_{dev}(x) = \tilde{I} - \frac{1}{n}\sum_{i=1}^{n} \tilde{I}(x_i). \tag{3.6}$$

Así, la aproximación (3.3) se reescribe como

$$S_{dev}(x) = S - \frac{1}{n}\sum_{i=1}^{n} S(x_i), \tag{3.7}$$

$$\tilde{I}_{dev}(x) = \tilde{I} - \frac{1}{n}\sum_{i=1}^{n} \tilde{I}(x_i), \tag{3.8}$$

$$\tilde{S}(\mathbf{x}) = \tilde{R}(\mathbf{x}) * \tilde{I}_{dev}(\mathbf{x}) = \alpha \sum_{k=1}^{N} \tilde{I}_{dev}(\mathbf{x} - \boldsymbol{a}_k). \tag{3.9}$$

Para que (3.9) sea una buena aproximación de (3.7), el fondo B que se resta debe ser constante, de modo que se compense al restar valores medios. De esta manera, el problema de minimización sigue siendo en esencia el mismo, solo que la función a minimizar se redefine como

$$\chi^2 = \|S_{dev} - \tilde{S}\| = \sum_{i=1}^{n} (S_{dev}(x_i) - \tilde{S}(x_i))^2. \tag{3.10}$$

Las técnicas de optimización clásicas [28, 29], como los métodos de Newton, Quasi-Newton, gradiente conjugado, descenso del gradiente, Lagrangiano Aumentado, son métodos iterativos que suelen converger de forma veloz a un óptimo local, lo que puede ser un inconveniente en problemas donde nos encontremos en presencia de un gran número de extremos locales. Estos métodos podrían quedar atrapados en una región específica y requerir de manipulaciones manuales y ad-hoc para explorar otras partes del espacio de soluciones.

Es por este motivo que, para atacar el problema (3.4), en [12] se optó por utilizar un algoritmo genético. Ideados por el Dr. John Henry Holland [29] en los años 70, este tipo de algoritmos se inspiran en la heurística de la evolución biológica y su base genética como mecanismo de optimización. Es un algoritmo que será explicado con detalle en la Sección 3.1.

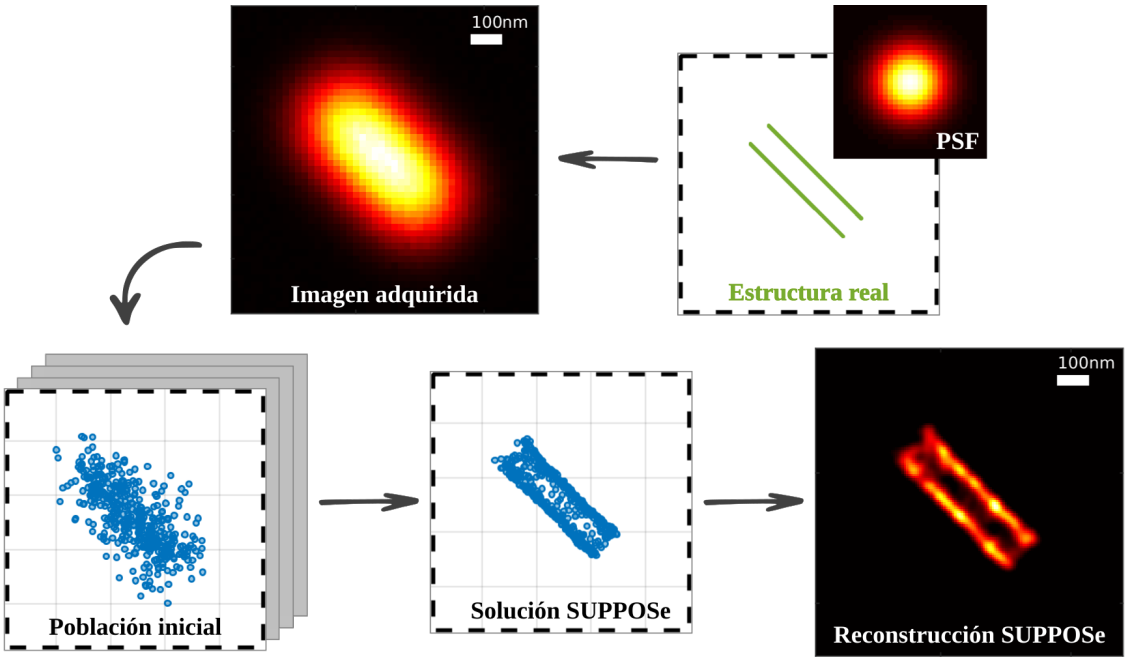


Figura 3.1: Caso de aplicación SUPPOSe en una imagen sintética, cuya estructura real son dos segmentos de rectas paralelas separadas una distancia $d = 0{,}42FWHM$.

En la Figura 3.1 se puede observar un esquema de aplicación del algoritmo SUPPOSe sobre una imagen simulada. Aquí, la estructura real R a la cual quisiéramos aproximarnos consta de dos segmentos paralelos separados a una cierta distancia d, y se convoluciona con una PSF de tipo gaussiana de tamaño $\sigma_{PSF} = 4,72$ píxeles. La separación entre segmentos es de $d = 0,42FWHM$, y cada segmento se construyó con 72 fuentes puntuales equiespaciadas a lo largo. Luego de aplicar la operación de la convolución, a la imagen se le agregó un valor de fondo constante y un ruido de tipo Poisson. En este caso es una imagen con un fondo bajo y un nivel de ruido medio. Los parámetros de síntesis de la imagen fueron seleccionados para simular las condiciones experimentales de un microscopio de fluorescencia de campo amplio utilizado en [18].

En la imagen se observa que los segmentos son prácticamente indistinguibles, conformando una única estructura elíptica y suave. Para aproximarnos lo mejor posible al objeto original y obtener una distribución de fuentes virtuales que represente dicha aproximación, SUPPOSe parte de una población inicial de p posibles soluciones, que llamaremos *individuos*, donde cada uno consiste en un conjunto de coordenadas de las N fuentes virtuales elegidas para el procesamiento de la imagen (en el esquema $N \sim 400$). El algoritmo irá evolucionando y obteniendo familias de individuos distintas mediante una mecánica que explicaremos a continuación, de manera que mejora el valor de la función objetivo χ^2. Al

alcanzar la condición de corte del algoritmo, el individuo del conjunto con mejor valor en la función objetivo es elegido como solución, y el conjunto de coordenadas resulta ser la posición de las fuentes virtuales.

Finalmente, a efectos de generar una imagen de superresolución, se convoluciona con una PSF con un valor de σ mucho más pequeño que el original del sistema óptico. Y como se observa en la última imagen de la Figura 3.1, se obtiene una reconstrucción que permite identificar la presencia de dos segmentos paralelos.

La cantidad N de fuentes virtuales es uno de los parámetros centrales de SUPPOSe, de hecho es el que actúa como parámetro de regularización para convertir el problema mal puesto en uno bien puesto. La elección de un valor adecuado resulta fundamental para lograr una buena reconstrucción del objeto: por un lado, una poca cantidad de fuentes virtuales puede ser insuficiente para reconstruir un objeto grande o complejo, resultando en reconstrucciones poco representativas; por el otro, una gran cantidad de fuentes virtuales incrementa el tiempo de cómputo y puede introducir artefactos en la reconstrucción debido al ruido. En [12] se deriva una cota para las incertezas en la reconstrucción de SUPPOSe de la forma

$$\frac{1}{N}\sum_{i=1}^{N} d(a_i|R)^2 \leq \frac{E_{tr}Q}{N} + N\left(E_{PSF} + \frac{1}{SNR}\right) \tag{3.11}$$

E_{tr} depende del error de truncado, Q representa la cantidad de fuentes reales en el objeto, E_{PSF} es un valor que depende del error en la PSF, y SNR resulta la relación señal-ruido de la imagen. Además, $d(a_i|R)$ representa la distancia de a_i a la fuente más cercana de la estructura real R. En [12] se encontró a partir de (3.11) un valor óptimo de N, para cada imagen. Además de estas cotas teóricas, se mostró en [30] utilizando imágenes sintéticas que el rango para el que se obtienen valores similares de exactitud es bastante amplio.

Nos ocuparemos en la próxima sección de explicar en detalle el algoritmo genético.

3.2. Algoritmo genético

Como introdujimos en la sección anterior, el gran problema es que los métodos clásicos de optimización son algoritmos de minimización local. Esto puede traer consecuencias en problemas con muchos mínimos locales.

Por otro lado, los algoritmos genéticos como los introducidos en [15, 16, 31, 32], son algoritmos iterativos de minimización de tipo global. En ellos, cada solución (los *individuos*) se expresa en términos de sus coordenadas (los *genes*). En una iteración, un conjunto de soluciones (una *población*) se modifica de acuerdo a ciertas operaciones inspiradas en la biología, para dar lugar a nuevas soluciones más aptas (con menor función objetivo). Esto va a permitir una mejor exploración del espacio en simultáneo: aunque ciertos individuos queden "atrapados" en un mínimo local, los demás pueden seguir avanzando por otros sectores y hallar mejores soluciones.

Otra gran ventaja de este tipo de métodos es que no requieren conocer características propias de la función objetivo, como las derivadas o el gradiente, evitando cálculos que en espacios grandes pueden aumentar el costo computacional en términos de tiempos y memoria. Este tipo de métodos tiene una cierta cuota estadística, lo que los hace más tolerantes a problemas donde la función objetivo incluye ruido. Más aún, la función puede ser no continua o no diferenciable, ya que solo necesitamos poder evaluarla en las coordenadas de los individuos.

Para el problema de minimización de SUPPOSe se utiliza una variante del algoritmo genético definido en [33]. El algoritmo utilizado constará de dos etapas. La primera es una fase de inicialización para dar con una primera población y que sirva como punto de partida para nuestro proceso. La segunda es un bucle iterativo que constará de puntuar a cada individuo de la población (y ordenarlos en función de dicha puntuación), seleccionar algunos de ellos, cruzarlos genéticamente (operación explicada en próximos párrafos), aplicarles una mutación y volver a comenzar. Expliquemos el funcionamiento de cada etapa.

Algoritmo 3.1 (Algoritmo Genético)**.** *Dividamoslo en dos fases.*

Fase I. *En esta fase como comentamos brevemente antes, buscamos dar con una población inicial que sirva como punto de partida a nuestro algoritmo. A priori el algoritmo genético podría comenzar con cualquier población y distribución de individuos. Pero vamos a tomarnos el trabajo trabajar un poco sobre ella para comenzar un poco más cerca de la solución. La clave estará en la generación de un individuo inicial. Una vez obtenido este, se genera una población de p individuos realizando perturbaciones a este individuo inicial (lo que en la práctica denominamos "patear" el*

individuo). Vamos entonces con el cálculo del individuo inicial.

Paso 1. *Se define una variable* $T := S_{dev} = S - \frac{1}{n}\sum_{i=1}^{n} S(x_i)$. *Se le resta dicho promedio si tenemos imágenes con fondo, sino es* S.

Paso 2. *Se confecciona un ciclo donde se hallan las posiciones de las fuentes virtuales que conforman el individuo inicial. En cada iteración*

1) *Buscamos el valor máximo de* T *y hallamos la posición* b_k *que la realiza. La guardamos.*
2) *Calculamos el factor* $m_k = \frac{1}{n}\sum_{i=1}^{n} \tilde{I}(x_i - b_k)$.
3) *Guardamos el individuo* b_k.
4) *Redefinimos* T *como*

$$T(x) = T(x) - \alpha_0 \sum_{k=1}^{i} (\tilde{I}(x - b_k) - m_k).$$

5) *Calculamos* $l(k) = ||T||$.

Paso 3. *El ciclo culmina cuando se llega al número de particulas prefijada por el usuario o cuando se alcanza un valor mínimo de* l.
Cuando ocurre esto último, la cantidad de fuentes utilizadas para llegar hasta dicho valor mínimo se selecciona como valor para N, *y será la cantidad de fuentes virtuales a ser utilizadas en el procesamiento de la imagen. El conjunto de las posiciones* b_k *(que resultan ser* N*) es nuestro individuo inicial.*

Paso 4. *Generaremos* p *individuos a partir del individuo inicial obtenido. Realizamos pequeñas perturbaciones normales en las coordenadas de las fuentes virtuales (patadas). En general, dicha "patada" vendrá gobernada por un parámetro de perturbación, que se elige como* $\sigma_r = 0,5\sigma$, *donde* σ *refiere al parámetro de la PSF. Podemos notar que el individuo inicial es puramente determinístico utilizando información de la imagen. Y por otro lado, la población inicial se genera con cierta aleatoriedad, lo que brinda una buena diversidad.*

Dada o generada la población, estamos en condiciones de ingresar en el bucle donde propiamente comienza el algoritmo genético. La población se irá transformando mediante varios pasos que modifican aleatoriamente las posiciones de las fuentes virtuales.

Fase II. *En este paso asumimos que tenemos una población inicial generada. Vamos a confeccionar un bucle que transformará de manera aleatoria mediante ciertas operaciones la posición de las fuentes virtuales. Recordemos que en este tipo de algoritmo trabajamos con un conjunto de soluciones y cada individuo resultaría ser una solución factible.*

El proceso constará de dotar de una puntuación a cada individuo, ordenarlos en función de dicha puntuación, fijar a los mejores de la lista (élite), y a los restantes aplicarles los operadores genéticos, que podríamos condensar en selección, cruza y mutación. Luego de esto, obtenemos una nueva población del mismo tamaño que la anterior (al aplicar los tres operadores genéticos obtendremos una población del mismo tamaño que la que teníamos antes de aplicarlos). Finalmente, añadimos los individuos fijados al inicio, para obtener una nueva población de igual tamaño que la que existía al comienzo del bucle, y continuamos iterando. En esta fase, debemos fijar un número N_{iter} que determinará cuántas iteraciones hará nuestro algoritmo (generaciones). A posteriori, se consta que el valor de la la función objetivo a lo largo de las generaciones haya dejado de evolucionar de manera significativa.

Expliquemos cada paso en detalle.

Puntuación: *Simplemente evaluamos nuestra función objetivo en cada individuo de la población.*

Ordenar: *Se ordena la población en función del puntaje obtenido antes de menor a mayor. Mientras más pequeño dicho valor, mejor candidato a solución es dicho individuo.*

Élite: *Fijamos a los mejores N_{elite} individuos de la lista (los que menor puntaje tienen). Estos individuos se mantienen invariantes en la iteración. Notar que el valor de N_{elite} es un parámetro del algoritmo. Llamamos a esta mini-familia de individuos élite.*

Selección: *Si teníamos una colección de p individuos, ahora tomaremos la población de $p - N_{elite}$ individuos que viene de quitar la élite de la población inicial recibida en la iteración. Ahora buscamos hacer réplicas de los individuos más aptos (dependiendo de un factor de escala que puede fijarse como parámetro del algoritmo). Buscaremos descartar a aquellos con peor puntuación y*

reemplazarlos por copias de los mejores (sin contar la élite). Para este paso pueden existir variantes del algoritmo como hacer la copia teniendo en cuenta a la élite o no. Observaremos que, de manera indirecta, se replicarán los individuos con menor valor en la función objetivo.

Cruza: *A la población de tamaño $p - N_{elite}$ que sale del paso de selección se le aplica este operador. Actúa sobre dos individuos generando dos descendientes que combinan características de los individuos originales (padres). Formalmente, se intercambian las coordenadas de algunas de las fuentes virtuales que conforman los individuos. La elección para dicha cruza se realiza de forma aleatoria, en función de un parámetro de cruza (una fracción del total de la población). Los dos individuos nuevos reemplazan a los padres en la nueva población.*

Mutación: *En este paso se generan modificaciones aleatorias a la población que viene de aplicar el operador de cruza. Esto va a permitir alcanzar nuevas soluciones. A este operador debemos darle como parámetro la fracción de individuos a ser mutados. Cada mutación constará de mover una fuente virtual una distancia aleatoria.*

Agregado de la élite: *Finalmente a la población ya mutada de tamaño $p - N_{elite}$ se le añade la élite fijada al inicio de la iteración, para obtener una nueva población de tamaño p.*

Algo interesante de notar es que en cada paso al fijar una élite, estamos garantizando que nuestro algoritmo mejora la solución (o al menos no la empeora). Además, en cada iteración, estamos evaluando toda la población en la función objetivo (χ^2). Esto implica que para cada individuo debemos reconstruir la imagen convolucionando las posiciones de las fuentes virtuales de SUPPOSe con la PSF estimada. A su vez, repetimos esto a lo largo de todas las generaciones, por lo que, en términos computacionales, resulta difícil realizar todas estas operaciones. En [30] se probó que para una familia de imágenes, las soluciones no se veían modificadas entre distintas corridas del algoritmo genético. Se observó allí también que, al igual que en las cotas teóricas, el número de fuentes virtuales no es crítico para la exactitud del método, permitiéndose un rango amplio de valores para N.

Finalmente, definiremos dos conceptos que nos serán de utilidad para evaluar la calidad

de la solución obtenida, que son exactitud y precisión. En el caso de la exactitud, buscamos cuantificar cuán cerca se encuentra una solución SUPPOSe de la estructura real (ground-truth) que buscamos reconstruir. En cambio, para la precisión, necesitamos cuantificar la dispersión entre dos soluciones SUPPOSe cualquiera. Si ambas soluciones provienen de procesar una imagen donde la estructura real es la misma, es deseable que la dispersión entre ambas sea lo más pequeña posible. Daremos más detalle de cómo cuantificar estos conceptos en la Sección 4.1, cuando presentemos una herramienta para medir distancias entre soluciones, que se definió en [19].

Implementando el algoritmo en una CPU de forma convencional puede verse que el cálculo de la convolución representa más del 95 porciento del tiempo de cómputo del algoritmo. Incluso el tiempo de cómputo crecerá a medida que aumentemos el número de fuentes virtuales y el tamaño de la población p. Entonces, se formuló otra función objetivo tal que permita reducir tiempos en el cálculo de la convolución. Es lo que plasmamos en la sección sobre el método SUPPOSe discreto (DSUPPOSe), pero para ello necesitamos ciertos preliminares, conceptos y teoría sobre la Transformada de Fourier discreta (DFT), que es lo que veremos en la próxima sección.

3.3. La Transformada de Fourier Discreta (DFT)

Como mencionamos anteriormente, una de las partes más costosas del método radica en el cálculo de la convolución. Más adelante, vamos a reformular el método utilizando la transformada de Fourier discreta (DFT), para mejorar este cómputo. Para ello, a modo de preliminares, en esta sección definiremos dicha transformada, así como algunas propiedades fundamentales necesarias.

Empezaremos por la definición en el caso en que nos encontramos en una dimensión.

Definición 3.1 (DFT)**.** *Dada una secuencia finita de números reales o complejos* $\{x_m\}_{m=0}^{n-1}$*, se define la transformada de Fourier Discreta de la forma,*

$$X_k = \sum_{m=0}^{n-1} x_m e^{-\frac{2\pi i}{n} km} \quad k = 0, \ldots, n-1 \tag{3.12}$$

donde $e^{\frac{2\pi i}{n}}$ *es la* n*-ésima raíz de la unidad.*

Los vectores $\left(e^{\frac{2\pi i}{n}km}\right)_m$ resultan una secuencia de exponenciales complejas que forman una base ortogonal en el espacio de los vectores complejos de dimensión n. Podríamos representarlo formalmente de la siguiente forma,

$$\sum_{m=0}^{n-1}\left(e^{\frac{2\pi i}{N}tm}\right)\left(e^{-\frac{2\pi i}{N}t'm}\right) = n\delta_{tt'}$$

donde $\delta_{tt'}$ es la delta de Kronecker, que vale 1 si $t = t'$ y 0 si $t \neq t'$. Vamos a notar la transformada de x con, $X,\ \mathcal{F}(x)$ o $\hat{x}$.

Observación 3.1. *La DFT es un operador lineal.*

Habiendo hecho estos comentarios, podemos ahora definir la fórmula inversa para la Transformada de Fourier Discreta.

Definición 3.2 (Inversa DFT)**.**

$$x_m = \frac{1}{n}\sum_{k=0}^{n-1} X_k e^{\frac{2\pi i}{n}km} \quad m = 0, \ldots, n-1.$$

Donde utilizamos el hecho de que X_k son las coordenadas del vector **x** en esa base. En general, notaremos a la transformada inversa como $\mathcal{F}^{-1}(\mathbf{X})$ o $\check{\mathbf{X}}$.

Enunciemos algunos teoremas útiles, como el Teorema de Plancherel y Parseval (donde el segundo resulta ser un caso específico del primero)

Teorema 3.3.1 (Plancherel)**.** *Si* **X** *e* **Y** *son las DFTs de* $\mathbf{x} = \{x_n\}$ *e* $\mathbf{y} = \{y_n\}$*, entonces*

$$\sum_{m=0}^{n-1} x_m y_m^* = \frac{1}{n}\sum_{k=0}^{n-1} X_k Y_k^*$$

(el supraíndice $*$ *indica conjugación compleja)*

Teorema 3.3.2 (Parseval)**.** *Si* **X** *resulta la DFT de* **x***, entonces*

$$\sum_{m=0}^{n-1} |x_m|^2 = \frac{1}{n}\sum_{k=0}^{n-1} |X_k|^2$$

Sin embargo, el teorema más útil para nuestro método resulta el Teorema de Convolución circular para la Transformada de Fourier discreta. Para ello, definamos la *convolución circular* y la extensión periódica.

Definición 3.3. *La convolución circular de* **x** *con* **y** *se define como*

$$(x * y)_n := \sum_{l=0}^{n-1} x_l \left(\sum_{r=-\infty}^{\infty} \tilde{y}_{m-l} \right),$$

donde˜denota la extensión periódica de y*, que cumple que* $\forall m \in \mathbb{Z}$*,*

$$\tilde{y}_m := y_{(m-rn)} = y_{m|n|}$$

y donde r *es el entero que verifica* $0 \leq m - rn \leq n$*.*

Observación 3.2. *La convolución resulta conmutativa.*

El siguiente teorema ahora nos plantea que la convolución es una operación que se relaciona bien con la Transformada de Fourier.

Teorema 3.3.3 (Convolución)**.** *Si* $\mathbf{X} = DFT(x)$ *y* $\mathbf{Y} = DFT(y)$*, entonces*

$$DFT^{-1}(\mathbf{X} \cdot \mathbf{Y}) = x * \tilde{y}$$

donde $\tilde{y}$ *es la extensión periódica de* y*.*

Demostración. Sean X_k e Y_k las transformadas de vectores de longitud n. La inversa del producto de las transformadas de Fourier resulta

$$\begin{aligned} DFT^{-1}(\mathbf{X} \cdot \mathbf{Y})_m &= \frac{1}{n} \sum_{k=0}^{n-1} X_k \cdot Y_k \cdot e^{\frac{2\pi i}{n} km} \\ &= \frac{1}{n} \sum_{k=0}^{n-1} \left(\sum_{l=0}^{n-1} x_l e^{-\frac{2\pi i}{n} kl} \right) \cdot \left(\sum_{r=0}^{n-1} y_r e^{-\frac{2\pi i}{n} kr} \right) \cdot e^{\frac{2\pi i}{n} km} \\ &= \sum_{l=0}^{n-1} x_l \sum_{r=0}^{n-1} y_r \left(\frac{1}{n} \sum_{k=0}^{n-1} e^{\frac{2\pi i}{n} k(m-l-r)} \right) \end{aligned}$$

Notar que hay dos posibilidades. Si $m-l-r$ resulta múltiplo de n (i.e. $r = m-l-zn$ para algún z entero), el término entre paréntesis da 1. Si no, dado que es una suma geométrica, da 0. Luego sobreviven solo

$$DFT^{-1}(\mathbf{X} \cdot \mathbf{Y})_m = \sum_{l=0}^{n-1} x_l \left(\sum_{z=-\infty}^{\infty} y_{m-l-zn} \right),$$

y la última línea resulta la definición de la convolución circular. □

Tanto la definición de DFT como el Teorema de Convolución pueden extenderse a más dimensiones denotando, para alguna matriz x en $\mathbb{C}^{\mathbf{n}}$, la transformada de Fourier Discreta como,

Definición 3.4.

$$X(\mathbf{k}) = \sum_{\mathbf{m=1}}^{\mathbf{n}} e^{-i2\pi \mathbf{k}\cdot(\mathbf{m}/\mathbf{n})} x(\mathbf{m}),$$

donde

$$\sum_{\mathbf{m=1}}^{\mathbf{n}} X(\mathbf{m}) = \sum_{m_1=1}^{n_1} \sum_{m_2=1}^{n_2} \cdots \sum_{m_D=1}^{n_D} X\left(m_1, m_2, \ldots, m_D\right).$$

$$\mathbf{n} = (n_1, n_2, \ldots, n_D), \quad \mathbf{k} = (k_1, k_2, \ldots, k_D), \quad \mathbf{m} = (m_1, m_2, \ldots, m_D),$$
$$\mathbf{m}/\mathbf{n} = (m_1/n_1, m_2/n_2, \ldots, m_D/n_D).$$

Ahora podemos obtener de manera análoga la fórmula inversa de la Transformada de Fourier Discreta ($\mathcal{F}^{-1}(X)$), y los análogos correspondientes a los teoremas previos, Plancherel, Parseval y Convolución.

Observación 3.3. *Sean $\{a_k\}_{k=1}^N \subseteq \mathbb{Z}$. Si cada a_k está soportado en el intervalo $[D+1, M-D]$, e I es una función con $sop(I) \subseteq [-c_0, c_0]$, y $D \geq c_0$, entonces la convolución circular entre $x_i = \sum_{k=1}^{N} \delta_{a_k}(i)$ e $y_i = I(i)$ queda dada por*

$$S(i) = \sum_{k=1}^{N} I(i - a_k) \quad \forall 1 \leq i \leq M.$$

Demostración. Sea $\{\tilde{a_k}\}_{k=1}^N$ la extensión periódica de a_k. La convolución circular entre x_i e y_i será

$$S(i) = \sum_{k=1}^{N} I(i - a_k) + \sum_{k=1}^{N} I(i - \tilde{a_k}), \quad \tilde{a_k} \in [1, M]^C.$$

Si $\tilde{a_k} < -D + 1$, entonces $i - \tilde{a_k} > i + D - 1 \geq D \geq c_0$. Por lo que, $I(i - \tilde{a_k}) = 0$.

Si $\tilde{a_k} > M + D$, entonces $i - \tilde{a_k} < i - M - D \leq M - M - D = -D \leq -c_0$. Por lo que, $I(i - \tilde{a_k}) = 0$, y se tiene lo pedido.

□

3.4. El método SUPPOSe Discreto

En la sección anterior a DFT vimos el método SUPPOSe y concluimos que uno de sus principales problemas cuando se lo aplica en imágenes 2D muy grandes es el tiempo de cómputo. Para mejorar esto se propone una variante que involucra discretizar el espacio donde las fuentes están localizadas. Para esta nueva formulación, utilizaremos una nueva función objetivo que no requerirá de evaluar la PSF en cada iteración, evitando el alto costo de las convoluciones ya que estas se transforman en un producto matricial. Además, esta formulación nos permitirá hacer una modificación en la instancia de mutación del algoritmo genético. Esta nueva variante fue formulada en [20] e implementada allí en Matlab. En esta tesis implementamos esta nueva versión utilizando Python [34]. En esta sección describiremos esta función objetivo y algunos resultados sobre su desempeño.

En primer lugar, para liberarnos del parámetro de la intensidad α cambiamos la función objetivo por la siguiente:

$$cov(\mathbf{A}) = 1 - \frac{\sum_{i=1}^{n}\sum_{k=1}^{N} I(\mathbf{x}_i - \mathbf{a}_k)S(\mathbf{x}_i)}{\sqrt{\sum_{i=1}^{n}\left(\sum_{k=1}^{N} I(\mathbf{x}_i - \mathbf{a}_k)\right)^2}\sqrt{\sum_{i=1}^{n}\left(S(\mathbf{x}_i)\right)^2}}. \tag{3.13}$$

Esta función se construye a partir del coeficiente de correlación (no centrado) entre la imagen original $S(\mathbf{x_i})$ y la convolución $\sum_{k=1}^{N} \alpha I(\mathbf{x}_i - \mathbf{a}_k)$, su mínimo vale 0 y, al estar normalizada, la dependencia con α se cancela. En [35, 36] se realizó una comparación en términos numéricos y teóricos para SUPPOSe utilizando χ^2 y cov como función objetivo del método. Se demostró que ambas funciones comparten los mismos mínimos y se concluyó que ambos algoritmos convergen a la misma solución.

Si asumimos que R (la estructura original) tiene dominio dentro de un conjunto compacto $\Omega = [1, n_1] \times [1, n_2]$ y aprovechamos la buena relación entre la convolución y la transformada de Fourier, aplicando DFT en (3.13) obtenemos por el Teorema de Plancherel 3.3.1, el Teorema de Convolución 3.3.3 y la Observación 3.3, que

$$Cov(\mathbf{A}) \sim 1 - \frac{\sum_{\mathbf{l}=1}^{\mathbf{n}} \overline{\hat{S}(\mathbf{l})}\hat{I}(\mathbf{l})\hat{R}(\mathbf{l})}{\sqrt{\sum_{\mathbf{l}=1}^{\mathbf{n}} \left|\hat{R}(\mathbf{l})\right|^2}\sqrt{\sum_{\mathbf{l}=1}^{\mathbf{n}} \left|\hat{S}(\mathbf{l})\right|^2}}. \tag{3.14}$$

Aquí utilizamos la notación $\hat{}$ para la Transformada de Fourier discreta (DFT), donde si las fuentes virtuales están en un espacio continuo $\mathbb{R}^2$ se tiene,

$$\hat{R}(\mathbf{l}) = \alpha \sum_{k=1}^{N} e^{-2\pi i \mathbf{a_k} \mathbf{l}}$$

Para acelerar el cálculo de la función objetivo, reemplazamos (3.14) por una nueva función objetivo (que llamaremos cov_{FT}) que consistirá en realizar una discretización $\{\mathbf{a_k}\}_{k=1}^{N}$ acorde al pixelado de la estructura de datos. Dados $\{\mathbf{a_k}\}_{k=1}^{N}$, S e I (o en su defecto $\hat{S}$ o $\hat{I}$), calculamos la discretización de la siguiente manera:

1. Inicializamos una matriz de ceros $Q_A \in \mathbb{Z}^{n_1 \times n_2}$ que acumulará la cantidad de fuentes en cada píxel.

2. Para cada fuente en $\{\mathbf{a_k}\}_{k=1}^{N}$, redondeamos su posición acorde al pixelado de la estructura de datos (se busca el píxel más cercano) y se suma 1 a Q_A en la posición correspondiente a ese píxel.

3. Se calcula la transformada de Fourier discreta de Q_A, y se la denota como H. En particular para $D = 2$ si se define F_{n_j} como $F_{n_j}(k, m) = e^{-2\pi i/n_j mk}$ (las matrices DFT en cada dimensión), se tiene que

$$H = F_{n_1} Q_A F_{n_2},$$

4. Calculamos

$$\begin{aligned} M &= \hat{I} \odot \overline{\hat{S}}, \\ J &= M \odot H, \\ B &= \hat{I} \odot H, \end{aligned} \tag{3.15}$$

   donde $\odot$ denota el producto punto a punto. Además es importante notar que el cálculo de M se realiza una sola vez y que B representa la DFT de la aproximación SUPPOSe.

5. Finalmente definimos la función objetivo como

$$cov_{FT} = 1 - \frac{\sum_{\mathbf{l}=1}^{\mathbf{n}} J(\mathbf{l})}{||B||_F ||\hat{S}||_F} \tag{3.16}$$

donde $\hat{I}, ||\hat{S}||_2, M, F_{n_j}$ se calculan solo una vez y $||\cdot||_F$ denota la norma de Frobenius.

Observación 3.4. *En este trabajo, cuando se realiza la implementación de DSUPPOSe en Python, particularmente al haber implementado la función* (3.16)*, verificamos que la DFT se hacía en menor tiempo utilizando la función fft.fft2 del módulo SciPy en vez de multiplicando con las matrices DFT.*

Ahora que tenemos nuestra función objetivo en mente, podemos modificar cierta parte del algoritmo genético con esta versión, puntualmente en la parte de las mutaciones. Dado que el primer paso en el cálculo de la función objetivo (o de puntuación) requiere redondear al píxel más cercano, solo habría una modificación en la fitness cuando la coordenada del individuo salte de un píxel a otro. Por esa razón, para evitar pasos innecesarios, modificamos el paso de mutación del algoritmo. Es decir, ahora las mutaciones consistirán en mover aleatoriamente la fuente un píxel (a la izquierda, derecha, arriba, abajo o en diagonal). Entonces, ahora las mutaciones consisten en movimientos de un píxel completo. Además, como estamos en el caso en que el objeto original tiene soporte compacto en Ω, agregamos un paso para imponer que las fuentes no se salgan del dominio. Fijamos los porcentajes de individuos y fuentes a mutar en el próximo capítulo. En el trabajo [20] se desarrolla la siguiente modificación.

Algoritmo 3.2 (Mutaciones Discretas)**.** *La entrada requerirá de una población y dos parámetros, que son una fracción de individuos a mutar N_{mut} (igual que en SUPPOSe) y una proporción de fuentes virtuales a ser mutadas $N_{sources_{mut}}$ (en cada individuo). Con esto en mente, seleccionamos al azar una cantidad de individuos a mutar.*

1. *Inicializamos una matriz con valores 1 y -1 de tamaño igual a $N_{sources\_mut} \times 2$.*

2. *Para cada individuo se seleccionan al azar $N_{sources_{mut}}$ de sus fuentes para ser mutadas.*

3. *Se suma la matriz creada en el paso 1 a las fuentes del individuo seleccionadas en el paso 2, salvo para las fuentes que están en el borde (en el algoritmo, ponemos una excepción para este caso y las direccionamos hacia dentro del dominio). Guardamos el nuevo individuo.*

4. *Finalizamos el ciclo cuando mutaron todos los individuos seleccionados.*

En [20] y tal como mencionamos en la introducción, se muestra que el desempeño de DSUPPOSe, en exactitud, precisión y resolución es igual al de SUPPOSe pero logrando una mejora de tiempo entre 5 y 10 veces. El método recientemente enunciado funciona para imágenes que no poseen fondo. En el caso de que lo tuviéramos, deberíamos reemplazar S por $S- < S >$ (restamos el valor medio) y en el paso (3.15), deberíamos agregar la condición $B(\mathbf{k} = 0) = 0$ como en [20] (donde la B se refiere al término de fondo y no a la matriz del cómputo en DSUPPOSe). No obstante, en esta tesis vamos a trabajar con imágenes sin fondo. En particular, en el próximo capítulo aplicaremos la implementación de DSUPPOSe realizada en Python en esta tesis sobre imágenes sin fondo, definiendo el caso de estudio y mostrando resultados y comparaciones con otros métodos.

Capítulo 4

Análisis

En este capítulo hablaremos directamente de la implementación del Método SUPPOSe discreto (DSUPPOSe) para ciertas imágenes modelo. Describiremos nuestro caso de estudio, que involucra cuál será nuestra imagen sintética a analizar y cómo la construimos. Explicaremos también cómo generamos el eventual ruido en la imagen y el procesado de la imagen sintética por este método, haciendo foco en los pasos intermedios del algoritmo, por ejemplo, en cómo obtenemos el individuo inicial, etc.

Luego de eso, haremos un análisis de los resultados obtenidos por este método. Para ello, es menester definir una suerte de cuasi-métrica para tener una noción concreta sobre el error cometido y/o obtener gráficos que nos indiquen si logramos mejorar la resolución.

Finalmente aplicaremos a la misma imagen sintética distintos métodos de deconvolución (por ejemplo, aquellos especificados en la Sección 2.4 en el Capítulo 2). Veremos qué resultados obtenemos y haremos un análisis de resolución para los diferentes casos. Para esto último será necesaria la utilización de la cuasi-métrica motivada arriba.

Vamos a empezar por la imagen sintética a estudiar y cómo la generamos. Para simular la PSF utilizaremos una función gaussiana como la definida en (1.1). Nuestro objeto a estudiar serán en esencia dos líneas verticales paralelas, con una separación determinada. Se elige este tipo de estructuras, ya que en muchos campos de aplicación lo que se busca es resolver estructuras filamentosas muy cercanas. Por simplicidad de la estructura elegimos el caso de rectas paralelas. El lienzo o dominio que contendrá a este objeto será de 65×65 (en píxeles). Para simular la estructura de dos líneas paralelas, construimos un arreglo de puntos (vectores de tamaño 2). Llamamos *groundtruth* a la distribución de puntos del obje-

to real (en este caso las dos líneas) y *sub-estructura* a cada una de las dos líneas verticales del groundtruth. Estos son 500 puntos, de los cuales la mitad pertenece a cada una de las subestructuras; están separados equidistantemente, por lo que la distancia entre puntos en cada segmento es de $0,08$ veces el tamaño del píxel. A su vez, las dos subestructuras están separadas por un σ. Es importante destacar que se elige esa separación ya que corresponde a una distancia $2{,}8$ veces menor que la resolución de nuestras muestras. El objetivo acá, al igual que en [18], es ver cuál es el alcance de cada método.

Fijaremos como valor de $\sigma = 5$ píxeles de ahora en adelante. La longitud de cada línea de puntos será de 20 píxeles. Si numeramos de izquierda a derecha y de arriba a abajo de forma creciente, la primera línea se construye sobre el píxel 24 (y la otra naturalmente sobre el píxel 29), y están contenidas del píxel 24 al píxel 44 (mirando la numeración vertical).

Con esto estamos en condiciones de hacer la convolución y obtener la imagen "distorsionada". Falta agregar el ruido a nuestro problema.

Para simular el ruido contextualicemos la situación en un detector que realiza conteo de fotones. En la realidad, cuando detectamos luz (fotones), el número de fotones en un tiempo dado sigue una distribución de Poisson. El nivel de ruido es mayor cuando hay pocos fotones y se vuelve relativamente menor cuando hay muchos (mejora la relación señal/ruido). Por eso, luego de reescalar la imagen, simulamos un ruido de tipo Poisson. De alguna forma, si esperamos, por ejemplo, 1000 fotones en un píxel, podemos recibir un valor ligeramente mayor o menor. Utilizamos como intensidad máxima para reescalar 5000. Podemos ver la construcción final de nuestra muestra en la Figura 4.1.

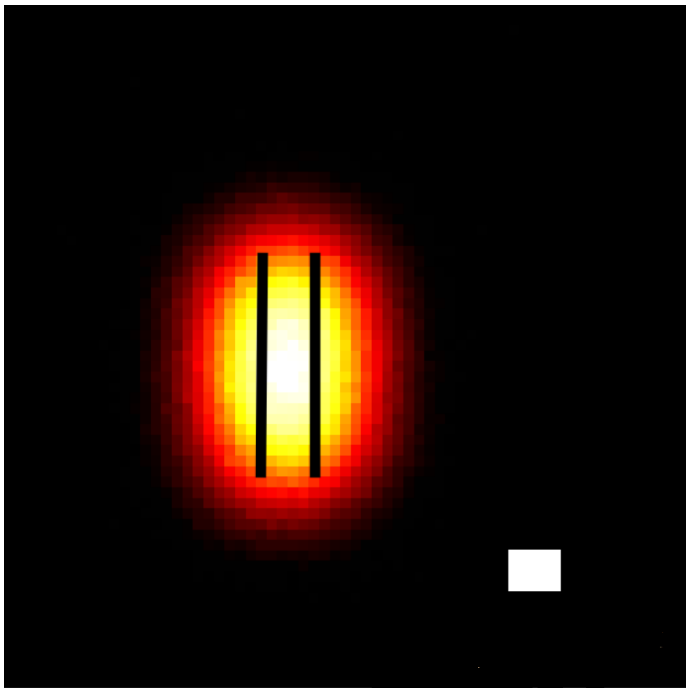


Figura 4.1: Imagen Sintética de las dos líneas verticales paralelas separadas a un σ de distancia. Superponemos el groundtruth. Barra de escala=σ. Intensidad máxima: 5000.

Una vez generada la imagen sintética, estamos en condiciones de procesarla y aplicarle DSUPPOSe.

Teniendo eso, estamos en condiciones de aplicar nuestro algoritmo genético, para lo que necesitamos de un individuo inicial. Dicho cómputo se realizó como detalla la Fase I del Algoritmo 3.1. Se toma como condición de corte cuando el valor de l resulta mayor al obtenido en el paso anterior (y tomamos como N el correspondiente al paso anterior). Dicho proceso nos arrojó un valor de $N = 1511$ utilizando como parámetro $\alpha = 6{,}3$ (este valor de α se elige como una fracción de la intensidad máxima de la imagen). A su vez, fijamos un número máximo de iteraciones para esta rutina en 2000. Generalmente intentaremos que nuestro N no venga dado por esta cantidad. La importancia del N se trató en el capítulo anterior, cuando hablamos sobre el método de individuos iniciales.

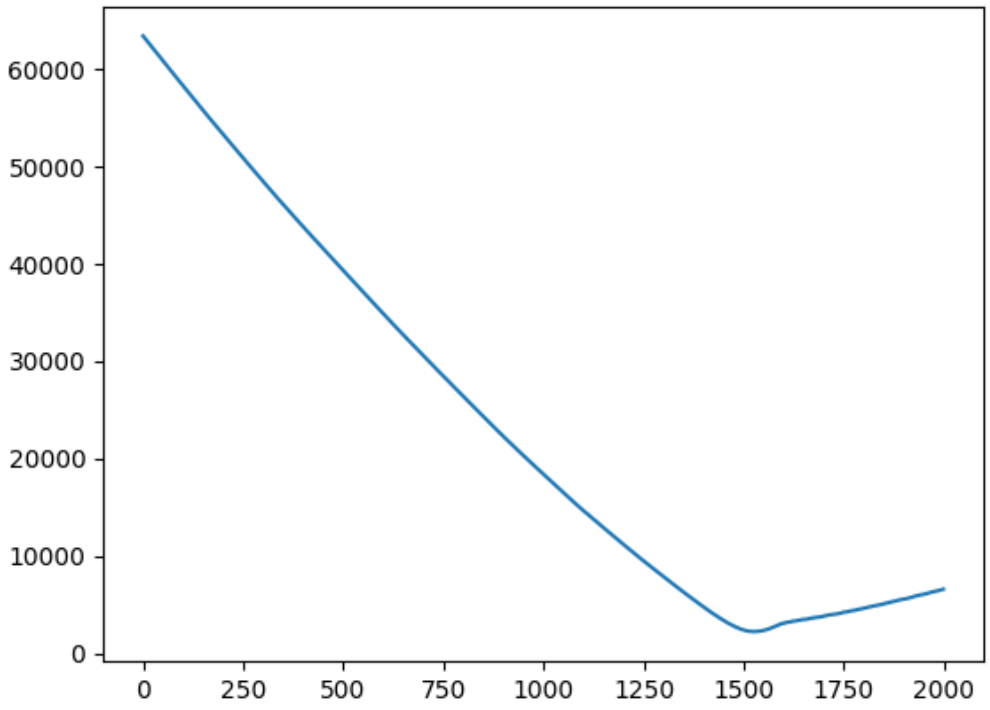
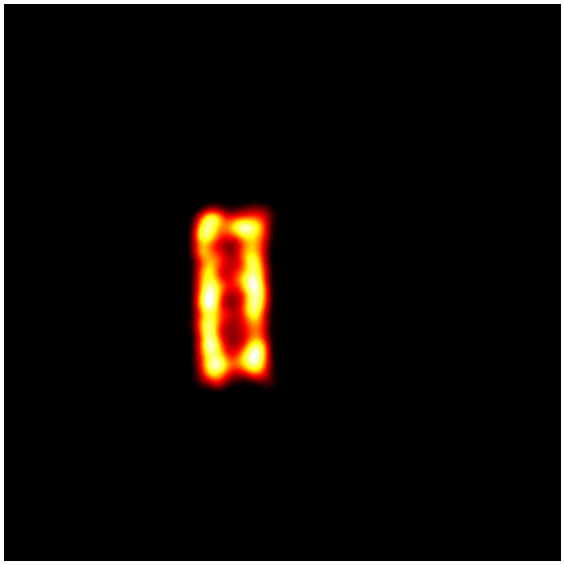


Figura 4.2: Curva que muestra el valor de l (definida en el Algoritmo 3.1) a lo largo de las iteraciones de la rutina de individuos iniciales, que indica el punto de corte del algoritmo.

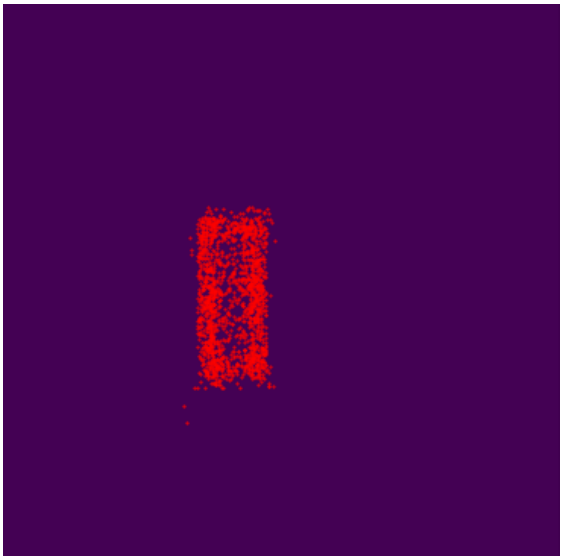
Luego, a partir de este individuo inicial se genera una población de 100 individuos. Dijimos que estos se generan a partir del inicial realizando "patadas" aleatorias, que dependen de un parámetro. Este se toma como $\sigma_r = 0,5\sigma$ (tal como comentamos en el Algoritmo 3.1).

Construida la población inicial, estamos en condiciones de aplicar la Fase II del Algoritmo 3.1 (bucle). Para esto, hacemos uso de las funciones genéticas, que requieren de fijar ciertos parámetros. El tamaño de élite fijado es de $N_{elite} = 4$, el factor que determina la tasa para realizar las cruzas es de $p_{cross} = 0,2$, la cantidad de individuos a mutar (como porcentaje del total) viene dada por $p_{mut} = 0,2$, mientras que el factor de fuentes a mutar (en cada individuo) se fijó en $pormut = 0,01$. Fijamos la cantidad de iteraciones que realizará el algoritmo genético en $N_{iter} = 40000$. Estos parámetros son los mismos utilizados para las distintas variantes del método en trabajos previos [12, 17, 18, 35].
El algoritmo hasta ahora implementado, nos devuelve una población final. A esta última, se le aplica por última vez la operación de puntuar y ordenar, para quedarnos con el mejor

individuo (nuestra solución del método). Podemos ver una visualización de dicha salida, que viene de convolucionar la solución con una PSF con un tamaño de $\frac{\sigma}{7}$, en la Figura 4.3a.



(a) Solución obtenida luego de aplicar 40000 iteraciones del método DSUPPOSe.



(b) Scatter plot de las posiciones de las fuentes que conforman la solución.

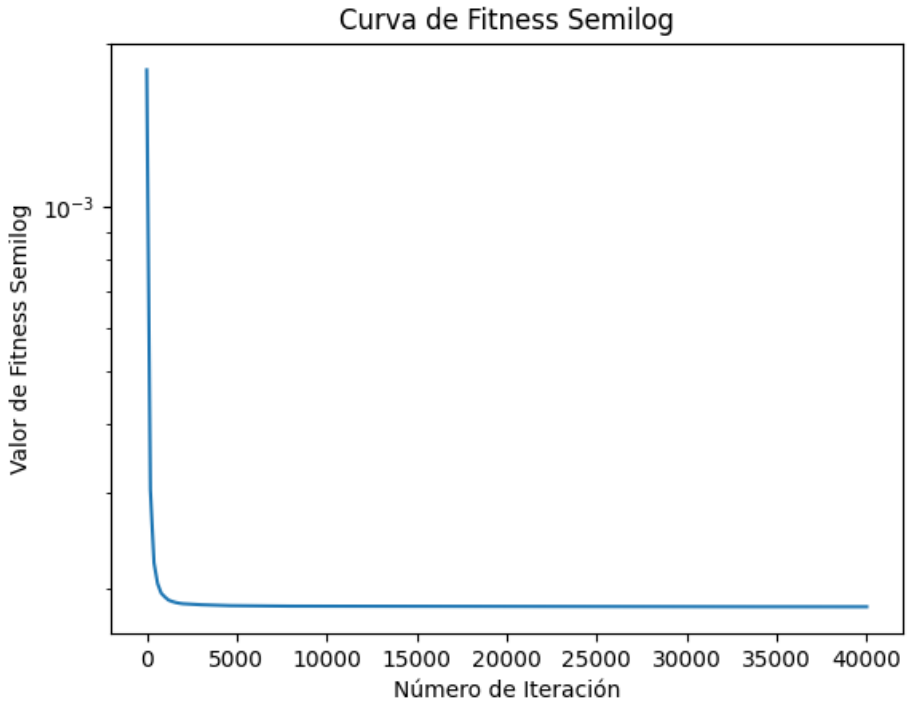


Figura 4.4: Curva que muestra el valor de la función objetivo (fitness) del mejor individuo de la población tomada en iteraciones intermedias del algoritmo. Se puede observar que la función objetivo decae muy rápidamente.

La implementación constó esencialmente de 5 módulos. Uno que genera la imagen sintética, otro que computa la función objetivo cov_{FT}, otro que contiene los operadores genéticos, otro que genera el individuo y la población inicial, y finalmente el quinto corre el método llamando a las funciones necesarias de los módulos previos. Mostramos también la evolución del mejor individuo a lo largo de algunas iteraciones.

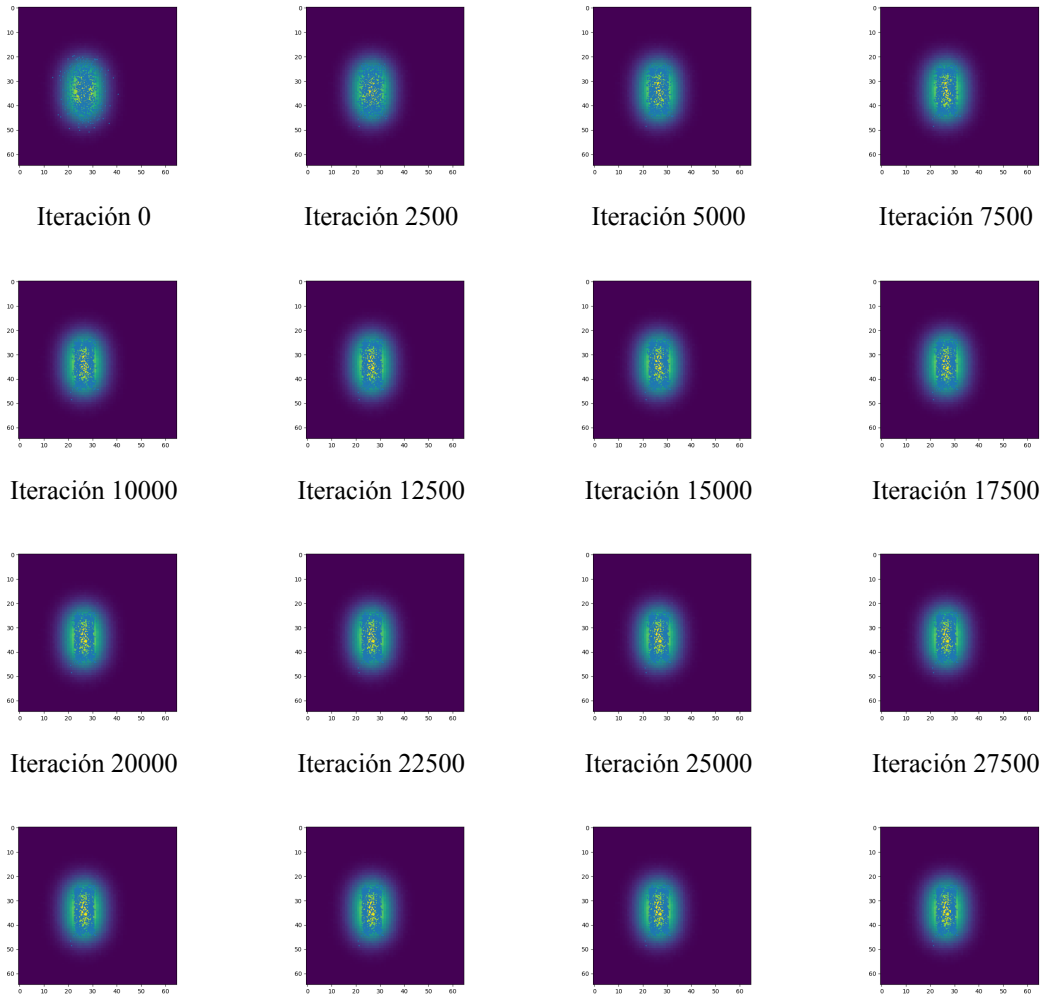


Figura 4.5: Evolución del mejor individuo a lo largo de 16 iteraciones intermedias del algoritmo, superpuestas a la muestra

4.1. Cuantificación de Resultados

Como introdujimos al inicio del capítulo, resulta menester utilizar alguna herramienta para determinar de manera cuantitativa la similitud entre la solución SUPPOSe y el groundtruth (exactitud) . En definitiva, cualquier caso se trata de dos conjuntos de coor-

denadas $a_k \in \mathbb{R}^2$ (donde la cantidad total de coordenadas en ambos conjuntos puede ser distinta). Para esto, utilizamos una cuantificación presentada en [19] , que busca comparar la ubicación relativa entre las coordenadas de las fuentes de ambos conjuntos.

Definición 4.1. *Sean dos conjuntos de coordenadas* $A = \{a_k\}_{k=1}^{K}$ *y* $B = \{b_l\}_{l=1}^{L}$ *donde* K *y* L *representan la cantidad total de coordenadas en los respectivos conjuntos. Para cada posición de una fuente en* A *definimos la distancia a* B *como*

$$d_k^{AB} = \min_{1 \leq l \leq L} ||a_k - b_l||, \quad \forall 1 \leq k \leq K \tag{4.1}$$

Es decir, para cada fuente en A buscamos la fuente en B que se encuentra más cercana y registramos ese valor de distancia. Repetimos esta operación para todas las fuentes que conforman el conjunto A. Al comparar el conjunto A con el B y viceversa obtendremos como resultado dos vectores d_k^{AB} y d_l^{BA} que contendrán los valores de distancias mínimas hallados. Las distribuciones de los datos que contienen estos vectores nos darán una idea de la similitud entre ambos conjuntos A, B. Una manera en la que podemos visualizar esta información es generando histogramas de estos vectores y finalmente registrando el valor medio de ellos. Mostramos este parámetro para nuestra solución obtenida en la Figura 4.6.

Estos valores fueron utilizados en [18] para cuantificar tanto la exactitud, la precisión como la resolución. Esta medida de distancia nos da una herramienta cuantitativa para comparar dos conjuntos de coordenadas entre sí. En el caso de imágenes sintéticas (estructuras de dos segmentos paralelos), utilizamos esta medida de distancia para obtener información sobre la exactitud del algoritmo si comparamos una solución SUPPOSe con la estructura real (groundtruth). En un caso similar, también se puede obtener la precisión comparando dos soluciones SUPPOSe entre sí. Es importante destacar que para evaluar el desempeño en [18] se sintetiza un conjunto de datos, donde para cada escenario se simular varias realizaciones. En nuestro caso, al estar tratando con un caso con bajo ruido, vamos a cuantificar estos valores utilizando una sola realización.

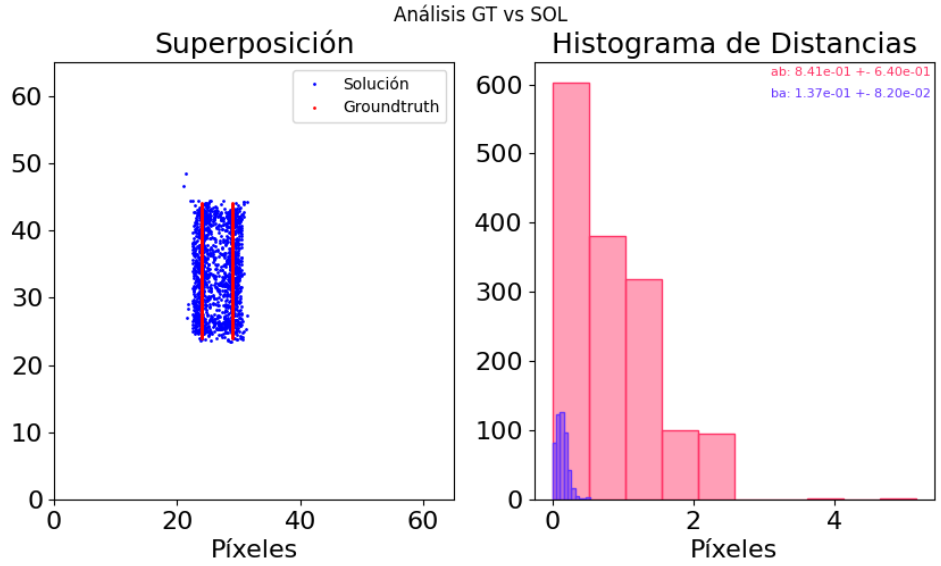


Figura 4.6: Superposición de groundtruth sobre la solución e histogramas solución vs groundtruth (ab) y groundtruth vs solución (ba). Obtenemos valores medios de $8,41 \cdot 10^{-1}$ y $1,37 \cdot 10^{-1}$ respectivamente para ab y ba.

Describimos ahora otro tipo de parámetro para cuantificar la similitud que podríamos aplicar a esta imagen sintética, presentada en [35, 36].

Buscaremos ver cómo ajustar una suma de dos gaussianas sobre las posiciones de las partículas luego de una corrida de SUPPOSe sobre la muestra de dos rectas paralelas. Dado que la distribución real es de dos líneas verticales, tiene sentido que una buena solución de SUPPOSe, si consideramos las coordenadas horizontales de dichas partículas, cumpla que ellas vivan en un entorno pequeño de la coordenada horizontal real de las dos líneas del groundtruth. Luego, tomando dichas coordenadas recuperamos un arreglo $A^1 = \{a_k^1\}_{k=1}^N$. Entonces se propone que cada partícula resulta, en realidad, la realización de una variable estocástica con densidad de probabilidad bimodal igual a una mixtura de dos gaussianas, de modo que la probabilidad de haber obtenido cada a_k^1 es

$$P(a_k^1|\theta) = \frac{\pi_1}{\sqrt{2\pi}\sigma_1} exp\left\{-\frac{(a_k^1-\mu_1)^2}{2\sigma_1^2}\right\} + \frac{\pi_2}{\sqrt{2\pi}\sigma_2} exp\left\{-\frac{(a_k^1-\mu_2)^2}{2\sigma_2^2}\right\},$$

donde $\theta = (\pi_1, \mu_1, \sigma_1, \pi_2, \mu_2, \sigma_2)$ representa el vector de parámetros de las rectas, μ_1, μ_2 son el centro de las rectas, σ_1, σ_2 son el ancho de las rectas y π_1, π_2 (sujeto a $\pi_1 + \pi_2 = 1$) representan el peso de cada recta y están asociados al número de partículas que les corresponden. Si las partículas son independientes, la probabilidad de obtener el arreglo

completo A^1 a partir de esta probabilidad es

$$P(A^1|\theta) = \prod_{k=1}^{N} P(a_k^1|\theta)$$

La idea es elegir θ tal que maximice la verosimilitud $\mathcal{L}(\theta) = P(A^1|\theta)$. Como siempre en estos casos, consideramos la log-verosimilitud.

$$l(\theta) = \log P(A^1|\theta) = \sum_{k=1}^{N} \log P(a_k^1|\theta)$$

y pedimos,

$$\frac{dl}{d\theta} = 0.$$

La minimización se realizó con una función del módulo SciPy de Python, basada en el algoritmo "Nelder-Mead". y los parámetros necesarios serían los dos centros, anchos y pesos, definidos en el párrafo previo. Realizamos este proceso con los siguientes parámetros iniciales:

$$\theta_{\mathbf{0}} = \begin{cases} \pi_1 = 0,5 \\ \mu_1 = 24 \quad \text{coordenada vertical real} \\ \sigma_1 = \sigma/5 \\ \pi_2 = 0,5 \\ \mu_2 = 29 \quad \text{coordenada vertical real} \\ \sigma_2 = \sigma/5 \end{cases}$$

recordando que estamos utilizando $\sigma = 5$.

Definimos el *sesgo* de la solución como

$$E = \frac{|\mu_1 - x_1'| + |\mu_2 - x_2'|}{2\sigma},$$

donde x_1' y x_2' indican la posición horizontal real de cada recta (groundtruth) y los μ_1, μ_2 son los que vienen del ajuste mencionado en líneas anteriores.

Definimos también la *mejora de resolución* como

$$M = \frac{2\sigma}{\sigma_1 + \sigma_2}$$

Aplicando esta métrica a nuestra solución SUPPOSe (4.3b), obtenemos la Figura

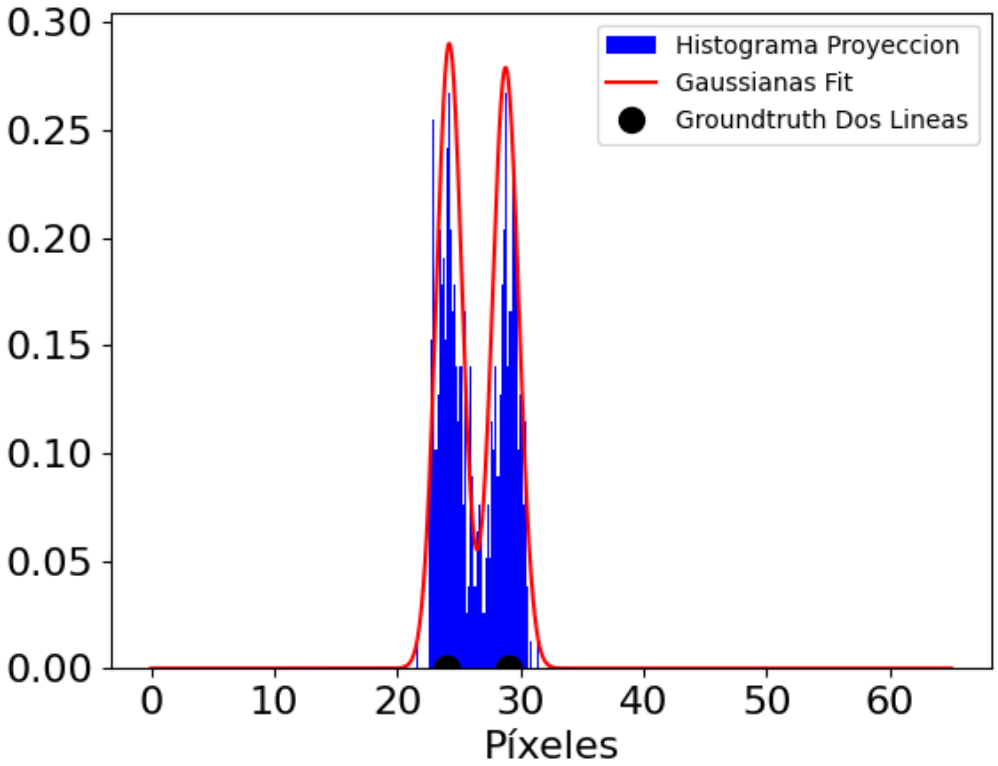


Figura 4.7: Superposición del histograma dado por la proyección y las dos gaussianas obtenidas por el algoritmo. Agregamos la posición horizontal del groundtruth.

Con los valores obtenidos del ajuste propuesto por esta herramienta, obtuvimos un sesgo de $E = 0,0424$ y una mejora en la resolución de $M = 4,725$, para este caso donde la separación de las estructuras es casi 3 veces menor que la resolución original.

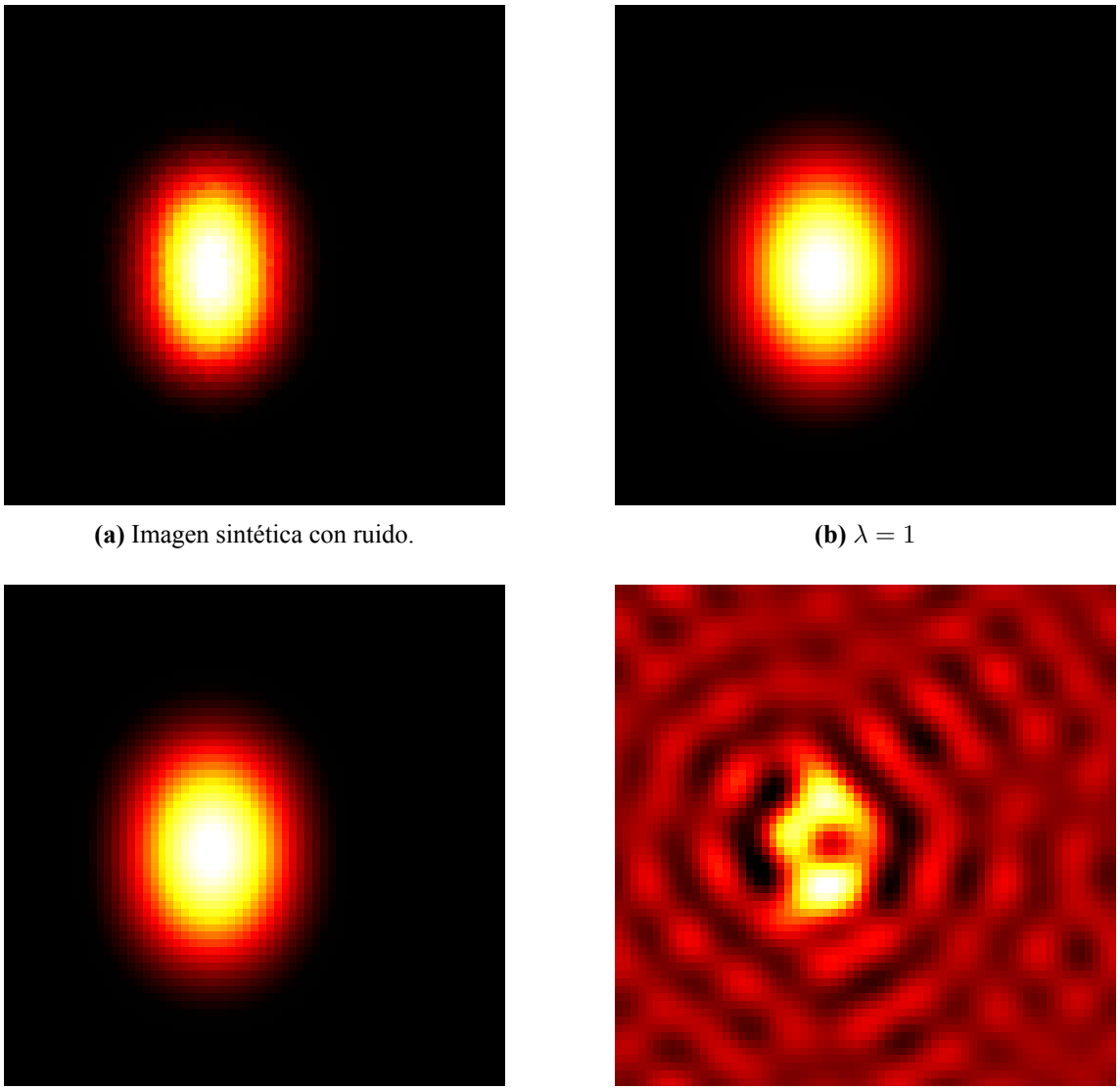
4.2. Aplicación del Caso de Estudio a otros métodos de deconvolución.

En la Sección 2.4, mencionamos algunos ejemplos de métodos de deconvolución (Tikhonov-Phillips, Tikhonov-Phillips Iterativo, Landweber). En esta sección buscamos aplicar di-

chos métodos de deconvolución a nuestro caso de estudio definido en 4. Mostraremos las soluciones obtenidas, y en algunas, trazaremos un perfil (de la misma forma y por el mismo segmento definido para la Figura 2.1b).

Queremos mostrar que toda esta familia de métodos es adecuada para eliminar ruido, pero que a pesar de elegir adecuadamente los parámetros la mejora en la resolución, que es lo que nos interesa en estas aplicaciones, es marginal. Si bien en la Sección 2.4 describimos cotas de los parámetros óptimos como queremos obtener resultados para ejemplos concretos optamos por hacer un barrido de parámetros en todos estos casos.

Empezamos por el método de Tikhonov-Phillips. Recordamos que este método requiere de un parámetro λ. Como se mencionó previamente, hicimos un barrido amplio de parámetros. Damos soluciones para tres de estos valores de λ.



(a) Imagen sintética con ruido.

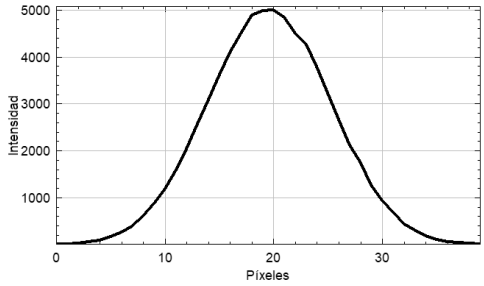
(b) $\lambda = 1$

(c) $\lambda = 10^4$

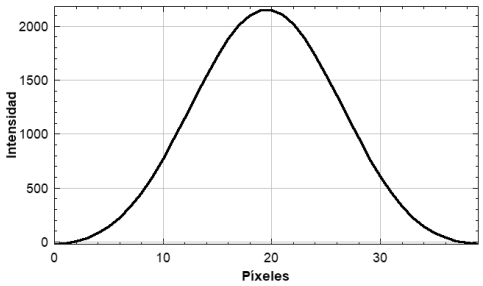
(d) $\lambda = 0,02$

Figura 4.8: Soluciones para el método de Tikhonov-Phillips con tres valores de λ, junto con la imagen sintética a deconvolucionar.

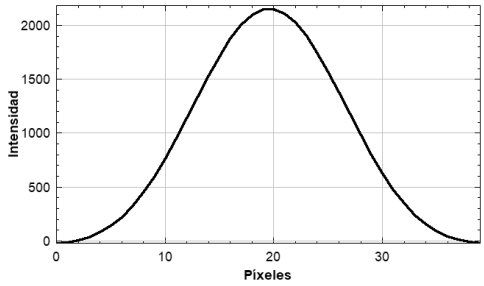
Notamos que con valores de $\lambda = 0,02$ o menores, obtenemos artificios. La forma de la solución no se asemeja en nada a la estructura real. En rangos de valores similares a $\lambda = 1$ obtenemos una versión más suave de la imagen, pero no tenemos resolución. Y si tomamos valores en rangos similares a $\lambda = 10^4$, obtenemos un poco más de suavizado y distorsión. Agregamos perfiles de estas imágenes, por la sección transversal en el centro de la imagen, en la Figura 4.9



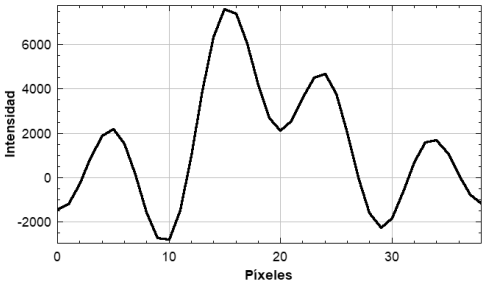
(a) Perfil de la imagen sintética con ruido.



(b) Perfil para $\lambda = 1$



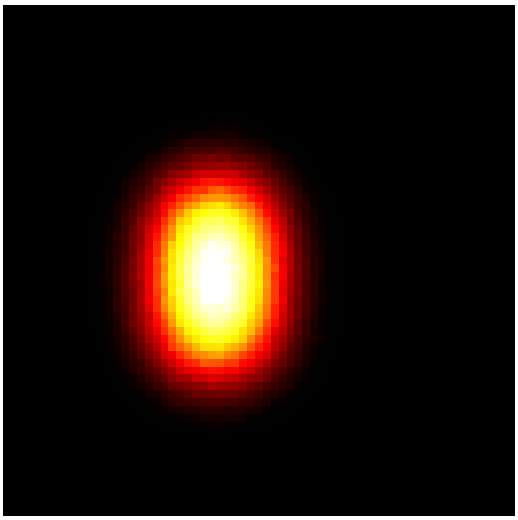
(c) Perfil para $\lambda = 10^4$



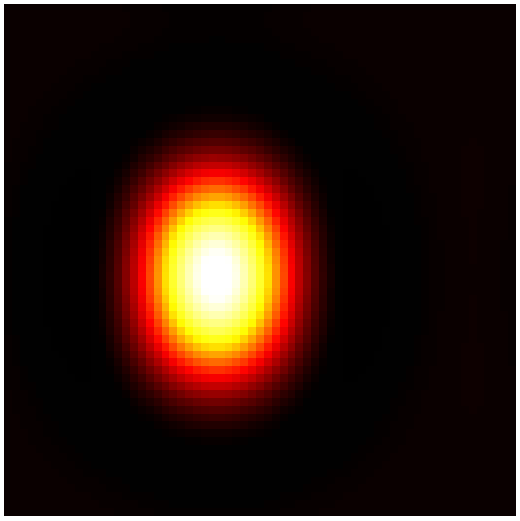
(d) Perfil para $\lambda = 0,02$

Figura 4.9: Perfiles de soluciones para el método de Tikhonov-Phillips.

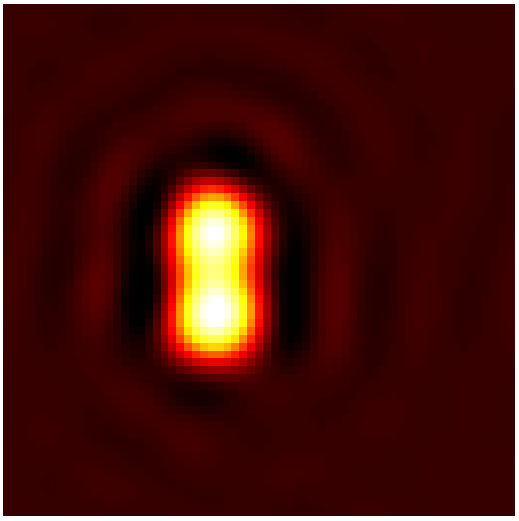
Continuamos con el método Landweber. Si recordamos su formulación, este método era puramente iterativo y solamente debemos darle como parámetro la cantidad de iteraciones a realizar. Nuevamente barrimos en un número amplio de parámetros, mostraremos dos grandes casos. Uno en el que le damos poca cantidad de iteraciones, y otro en el que le damos una buena cantidad de ellas. Vemos los resultados en la Figura 4.10



(a) Imagen sintética con ruido.



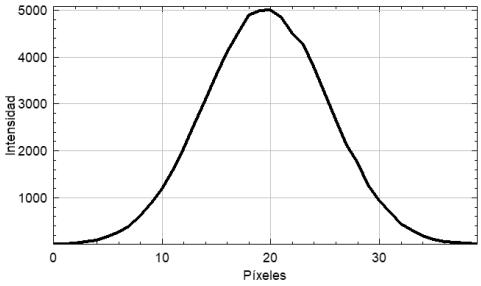
(b) Solución con $N = 2$ iteraciones.



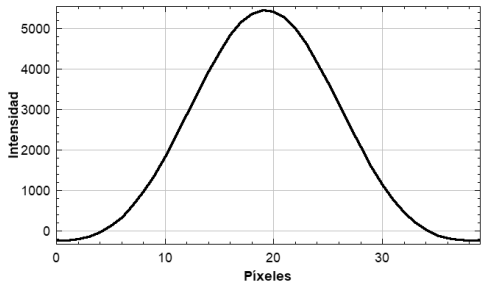
(c) Solución con $N = 10000$ iteraciones.

Figura 4.10: Soluciones para el método de Landweber con dos valores de N (cantidad de iteraciones), junto con la imagen sintética a deconvolucionar.

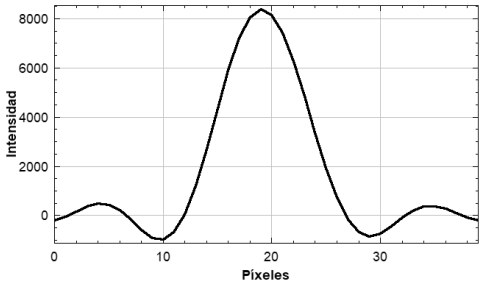
Obtenemos que si realizamos una cantidad de iteraciones del orden de $N = 10000$ obtenemos artificios, ya que la solución no se asemeja al groundtruth (estaríamos tentados a conjeturar que la estructura real es de dos bolas). En cambio, para pocas iteraciones (como $N = 2$) obtenemos una versión más suavizada y distorsionada, en la que claramente no obtenemos resolución. Nuevamente, agregamos perfiles de dichas imágenes en la Figura 4.11a.



(a) Perfil de imagen sintética con ruido.



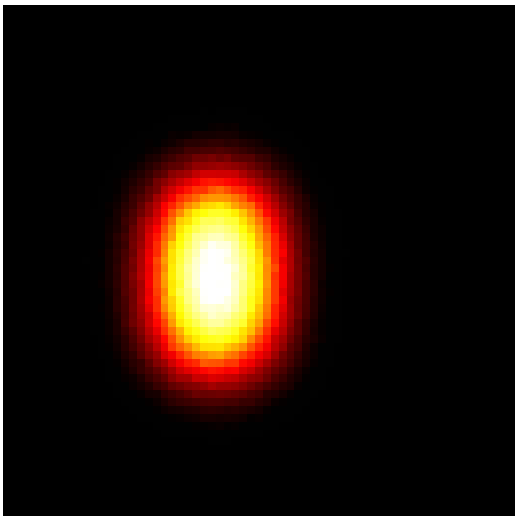
(b) Perfil de solución con $N = 2$ iteraciones.



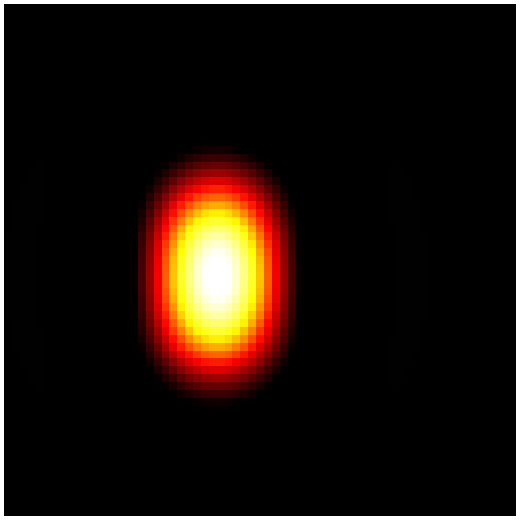
(c) Perfil de solución con $N = 10000$ iteraciones.

Figura 4.11: Perfiles de soluciones fabricadas con el método de Landweber.

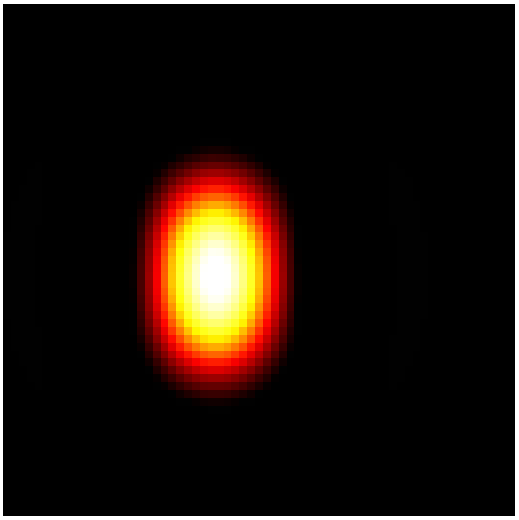
Consideramos ahora el tercer método de los mencionados en la introducción de esta sección (presentado también en el Ejemplo 2.3), que es el método iterativo de Tikhonov-Phillips. En este método, al ser iterativo, tendremos que fijar una cantidad N de iteraciones, además de un parámetro λ (al igual que en Tikhonov-Phillips clásico). Nuevamente, barrimos ambos parámetros pero mostraremos dos valores N y para cada uno de ellos, mostraremos dos valores para λ. Para $N = 2$, consideramos valores $\lambda = 1$ y $\lambda = 0{,}02$. Aplicamos el método con estos parámetros y obtenemos las soluciones que mostramos en la Figura 4.12



(a) Imagen sintética con ruido.



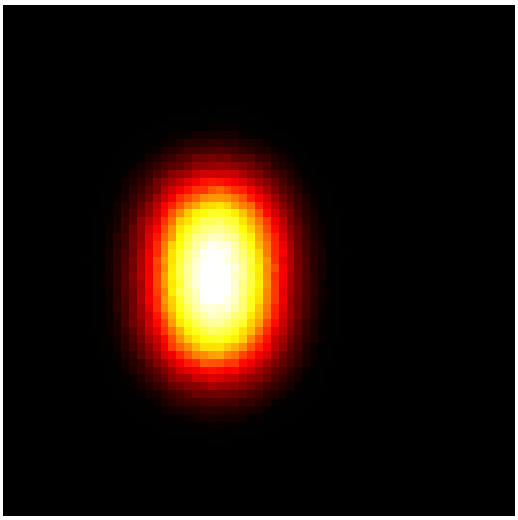
(b) Solución con $N = 2$ iteraciones y $\lambda = 1$.



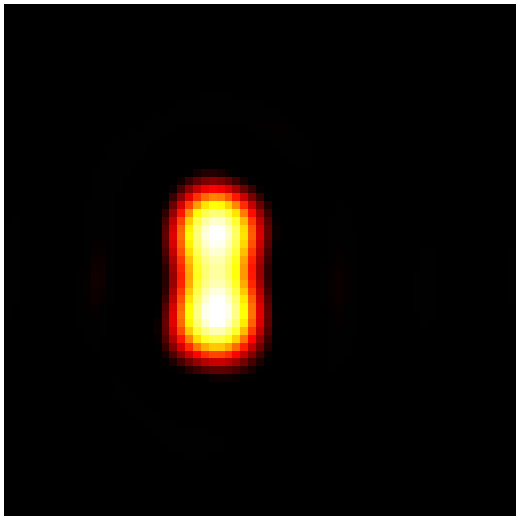
(c) Solución con $N = 2$ iteraciones y $\lambda = 0,02$.

Figura 4.12: Soluciones para el método iterativo de Tikhonov con $N = 2$ iteraciones, y con dos valores de λ, junto con la imagen sintética a deconvolucionar.

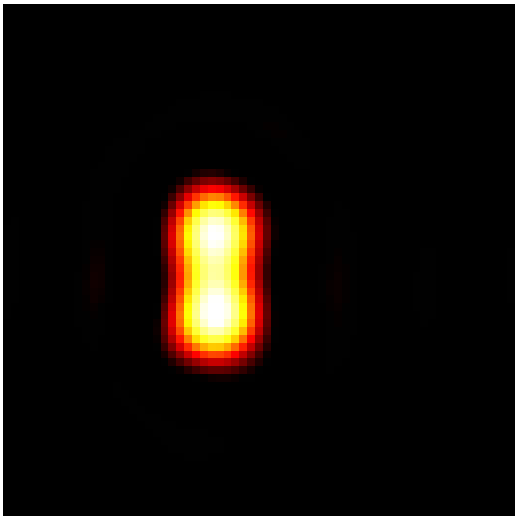
Operamos análogamente, ahora con cantidad de iteraciones $N = 5000$ y $\lambda = 1, \lambda = 0,02$. Vemos las soluciones en la Figura 4.13.



(a) Imagen sintética con ruido.



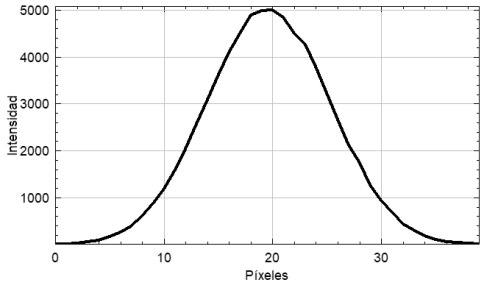
(b) Solución con $N = 5000$ y $\lambda = 1$.



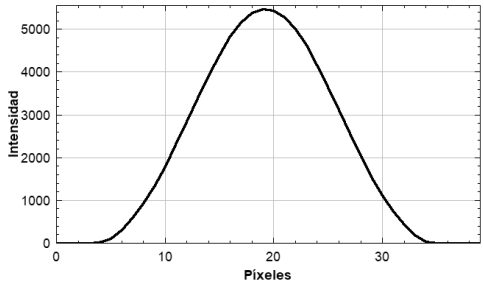
(c) Solución con $N = 5000$ y $\lambda = 0,02$.

Figura 4.13: Soluciones para el método iterativo de Tikhonov con $N = 5000$ iteraciones, y con dos valores de λ, junto con la imagen sintética a deconvolucionar.

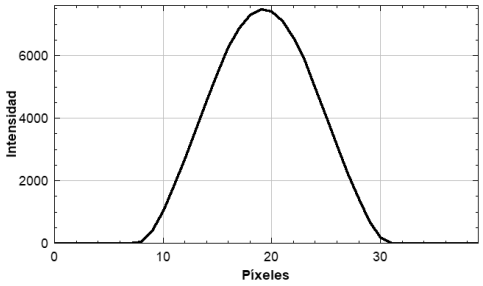
Se ve que el parámetro dominante en estos casos es la cantidad de iteraciones N. Para cada uno de los valores de N no observamos diferencias sustanciales en las imágenes obtenidas. Para $N = 2$ obtenemos un suavizado de la imagen, pero al igual que antes, no obtenemos resolución. En cambio, con $N = 5000$ iteraciones, vemos que la solución tiene artificios (nuevamente no asemeja al groundtruth que queremos visualizar). Obtenemos una disposición que invita a asumir que la estructura real son dos bolas (además de una suave suerte de aureola alrededor del objeto). Al igual que antes, mostramos en las Figuras 4.14 y 4.15 los perfiles obtenidos en cada solución en los que se logra ver que en ningún caso es posible resolver las dos líneas.



(a) Perfil de imagen sintética con ruido.

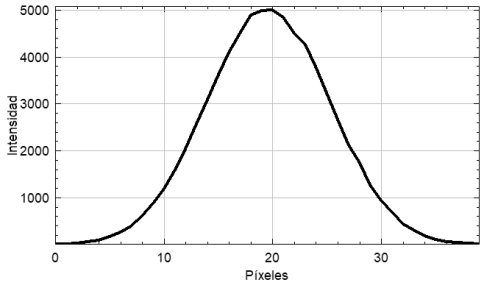


(b) Perfil de solución con $N = 2$ iteraciones y $\lambda = 1$.

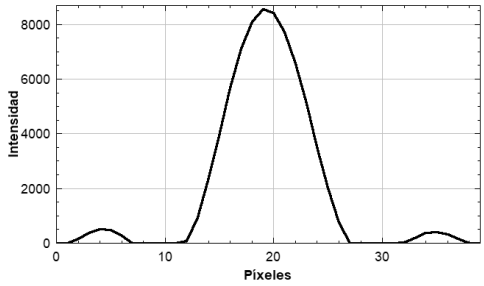


(c) Perfil de solución con $N = 2$ iteraciones y $\lambda = 0,02$.

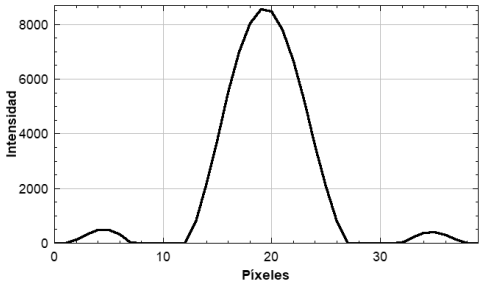
Figura 4.14: Perfiles de soluciones para el método iterativo de Tikhonov con $N = 2$ iteraciones.



(a) Perfil de imagen sintética con ruido.



(b) Perfil de solución con $N = 5000$ iteraciones y $\lambda = 1$.



(c) Perfil de solución con $N = 5000$ iteraciones y $\lambda = 0,02$.

Figura 4.15: Perfiles de soluciones para el método iterativo de Tikhonov con $N = 5000$ iteraciones.

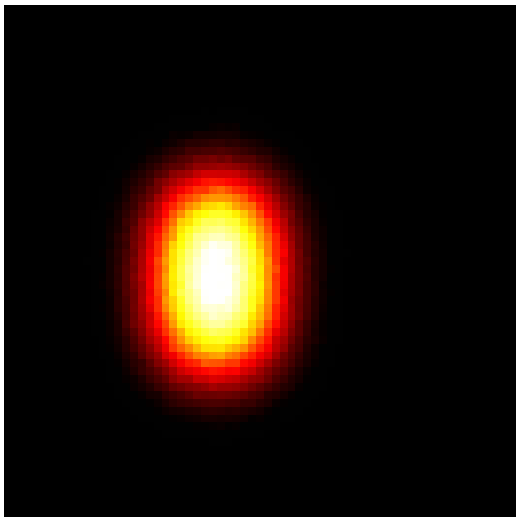
En la introducción de esta tesis hicimos mención de otro método de deconvolución, el

método de Richardson-Lucy. No es tema relevante para esta tesis la descripción exacta de este método iterativo, pero mostramos la forma de la recursión que se plantea (basada en un enfoque probabilístico). Utilizando la notación del problema descrito en (3.1), queda

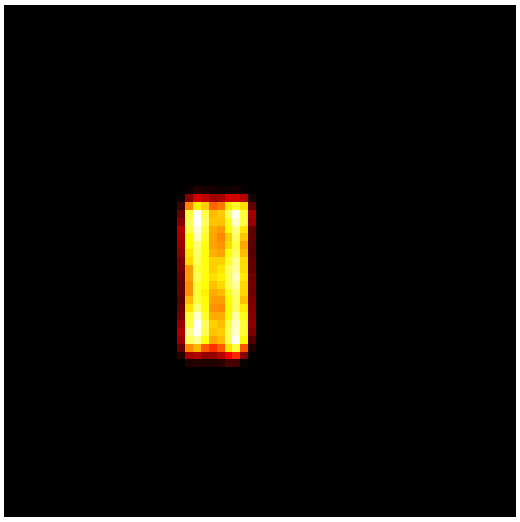
$$\hat{R}^{(t+1)} = \hat{R}^{(t)} \cdot \left(\frac{S}{\hat{R}^{(t)} * I} * I^* \right)$$

donde $\hat{R}^{(t)}$ es la estimación para R en la iteración t, y $*$ es la convolución en dos dimensiones. En el método, se construye una función de verosimilitud a minimizar (likelihood) asumiendo que el dato observado sigue una distribución de Poisson; por ende, al aplicar la minimización y obtener la iteración, estamos armando un método que tendría que funcionar bien en contextos de ruido de tipo Poisson (como el que tenemos en nuestro caso de estudio).

En efecto, aplicando 5000 iteraciones del método de Richardson-Lucy a nuestra imagen sintética, vemos el resultado en la Figura 4.16



(a) Imagen Sintética de dos líneas verticales paralelas separadas a un σ de distancia.



(b) Solución obtenida luego de aplicar el método Richardson-Lucy con 5000 iteraciones.

Figura 4.16: Antes y después de una imagen aplicando el método Richardson-Lucy.

En este caso, podemos notar a simple vista que el método obtiene una solución prácticamente sin artificios. Trazando un perfil, como se ve en la figura, podemos ver dos picos, lo que indica que sobre ese segmento, las intensidades se acumulan sobre dos sectores. Esto último y la vista natural de la solución muestran una distribución de fuentes que invita a sugerir que la distribución real es de dos segmentos paralelos.

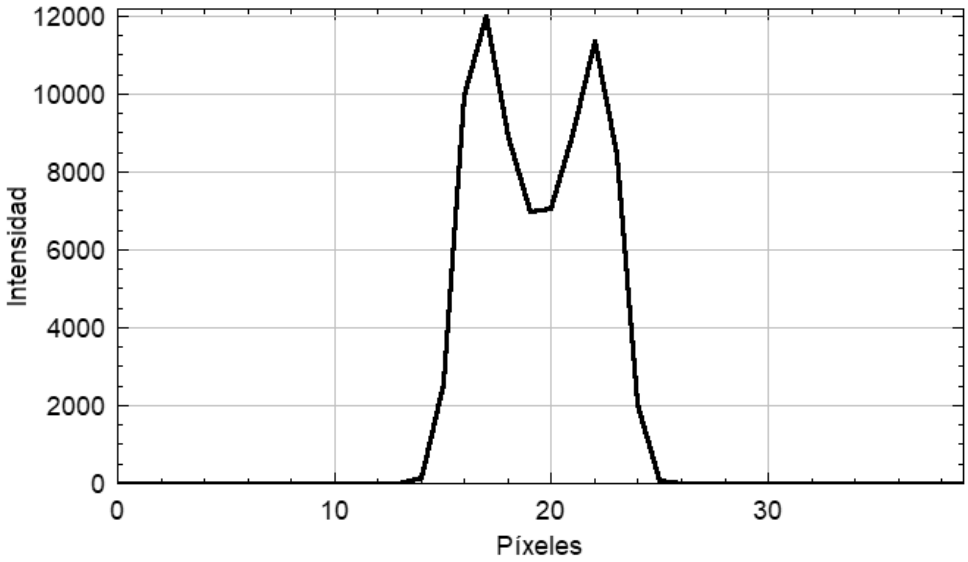


Figura 4.17: Perfil de solución obtenida luego de aplicar el método Richardson-Lucy.

Al tener una solución aceptable, realizamos un ajuste que nos permita cuantificar la exactitud para este caso. Para ello, sumamos todos los valores de intensidad por cada columna de píxeles, y cada uno de esos valores totales lo guardamos en un arreglo unidimensional. Luego, buscamos ajustar dichos valores por una suma de dos gaussianas. Realizamos esta última operación en Python invocando a la función *curve_fit* del módulo *scipy.optimize*. Para este caso, observamos los resultados obtenidos en la Figura 4.18.

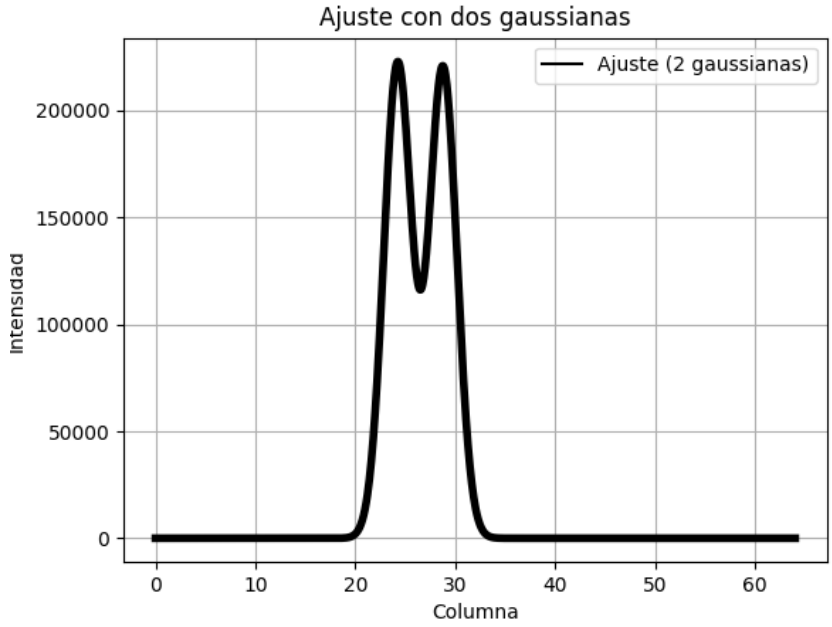
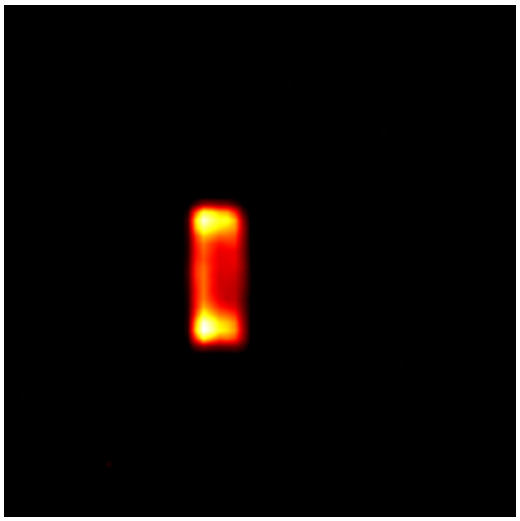


Figura 4.18: Ajuste de una suma de dos gaussianas para el arreglo obtenido de proyectar las intensidades de la solución del método de Richardson-Lucy.

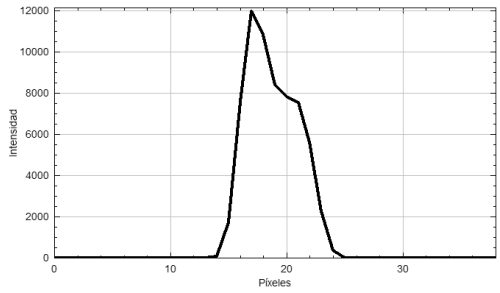
La pregunta natural ahora resulta, ¿Por qué no analizamos todos nuestros casos con este último método? En principio, podemos mencionar dos grandes motivos. El primero de ellos radica en que el método de Richardson-Lucy resulta sensible a variaciones en la estimación de la PSF. Por ejemplo, tenemos un mecanismo óptico que obtiene una imagen sintética a deconvolucionar. En la práctica se suele usar la PSF teórica (sabiendo la longitud de onda y la apertura numérica), que suele estar subestimada. En muchos casos, la PSF se puede medir experimentalmente, pero hay casos en los que eso no es posible. Por eso tiene sentido simular una situación así, en donde la PSF utilizada en el método es distinta a la real. En la tesis [30], se hizo un análisis de cuánto afecta el desempeño este error en la PSF para SUPPOSe. En este caso, la PSF para simular la imagen sintética tendrá un valor de $\sigma = 5$, aunque trabajaremos brindándole al algoritmo una PSF con un valor de $\sigma = 5,25$. Buscamos obtener una solución aceptable en un caso como este. El segundo motivo, refiere a que normalmente al obtener una imagen real, no solo obtenemos ruido por conteo de fotones (Poisson), sino que también se obtiene ruido por la electrónica (sensor y procesos internos del aparato de medición). Este tipo de ruido típicamente se modela con una distribución normal (suele llamarse ruido gaussiano). Richardson-Lucy asume solo ruido de tipo Poisson, por ende, si el ruido electrónico resulta significativo, el

resultado obtenido puede ser poco robusto.

Llevamos estos ejemplos a nuestro caso de estudio (confeccionamos la imagen sintética con un $\sigma = 5$ y variamos la PSF a un $\sigma = 5,25$). Además, agregamos un término de ruido proveniente de una distribución normal con $\mu = 0$ y desvío de 5 (además del ruido modelado en formato Poisson, con el mismo número de cuentas que en la Figura 4.1). Procesamos usando el método de Richardson-Lucy variando los parámetros y en donde la mejor de todas las soluciones se puede visualizar en la Figura 4.19, con 5000 iteraciones. Podemos observar que no obtenemos una visualización clara que invite a conjeturar que la distribución real es de dos líneas paralelas. Incluso, trazando el perfil en 4.19b se ve que obtenemos un solo gran pico (diferente a antes, que obteníamos dos).



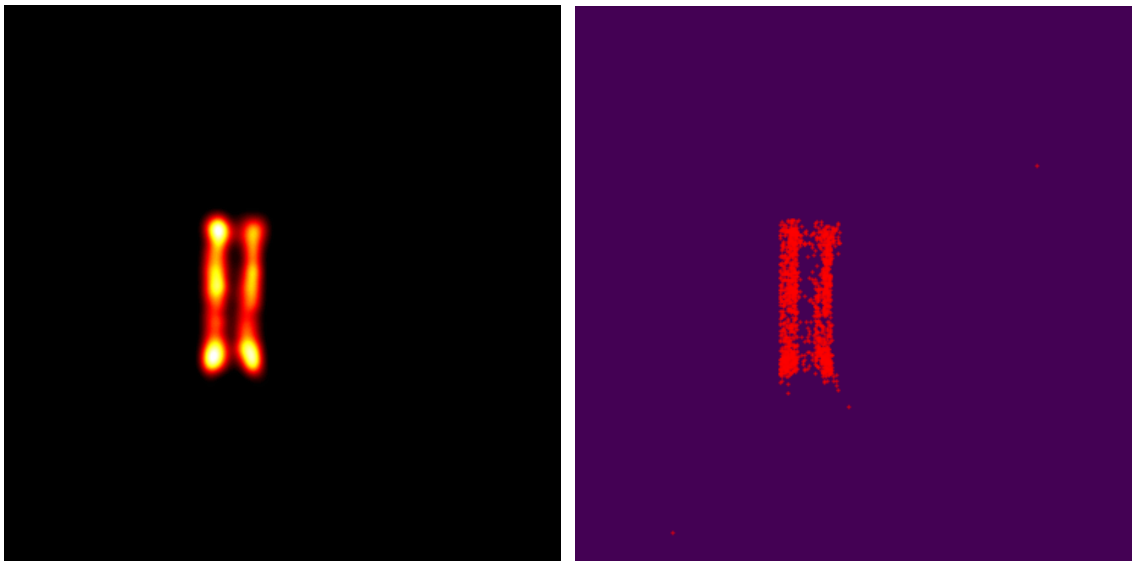
(a) Solución obtenida con el método de Richardson-Lucy con una PSF de $\sigma = 5,25$.



(b) Perfil de solución obtenida con el método de Richardson-Lucy con una PSF de $\sigma = 5,25$.

Figura 4.19: Análisis en caso de error en la estimación de la PSF para el método de Richardson-Lucy.

En cambio, si procesamos el mismo caso de estudio con nuestra implementación en Python de DSUPPOSe, obtenemos la siguiente solución en la Figura 4.20a, que claramente permite distinguir la configuración de los dos segmentos paralelos, en contraposición con la obtenida por el método R-L.



(a) Solución DSUPPOSe para el caso de estudio con modelados de ruido Gaussiano y (intensidad máxima 5000), y variación de PSF con $\sigma = 5, 25$.

(b) Scatter plot de las posiciones de las fuentes que conforman la solución.

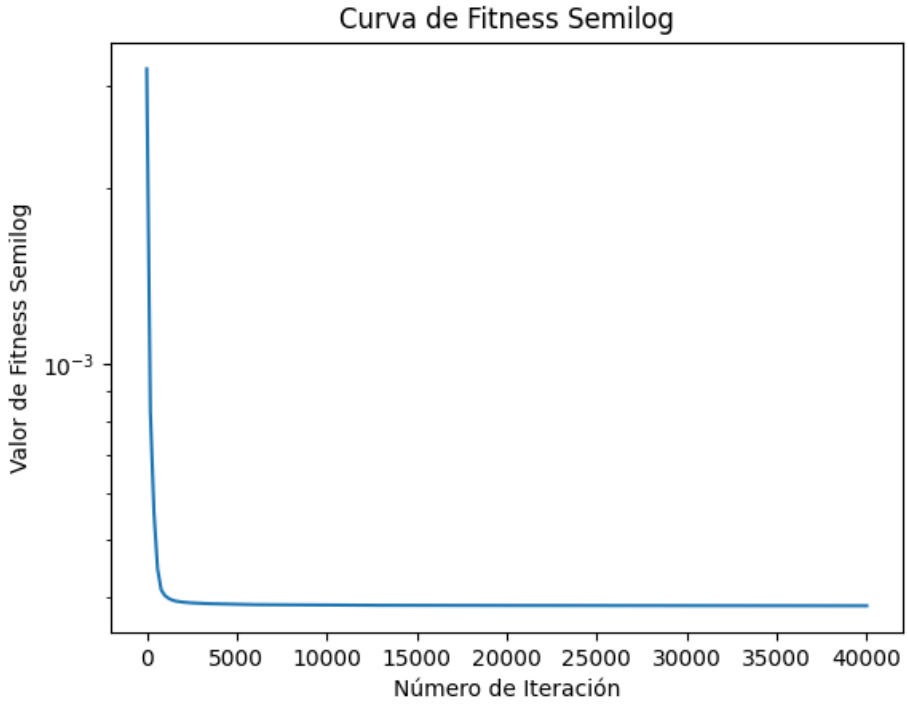


Figura 4.21: Curva que muestra el valor de la función objetivo (fitness) del mejor individuo de la población tomada en iteraciones intermedias del algoritmo. Se puede observar que la función objetivo decae muy rápidamente.

Si utilizamos las herramientas presentadas en la Sección 4.1, obtenemos,

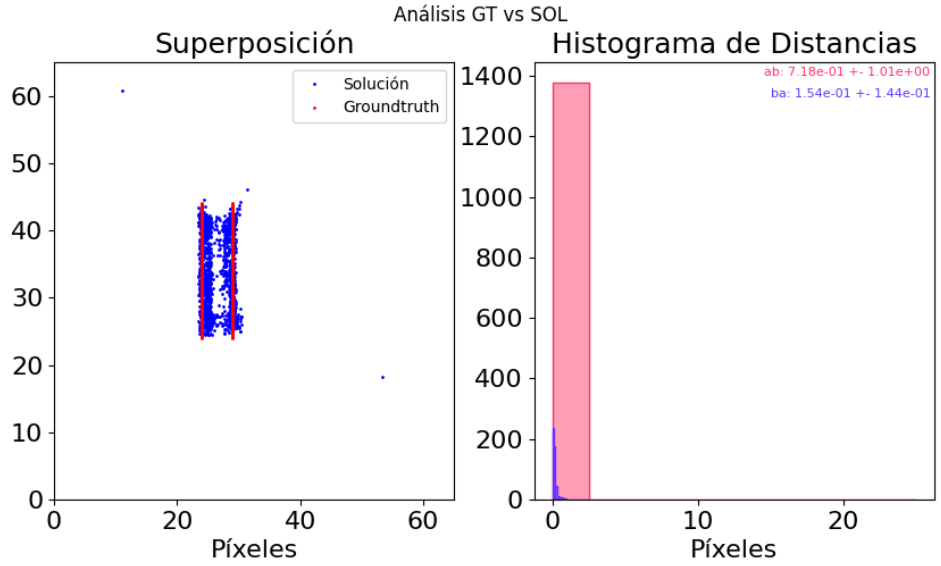


Figura 4.22: Superposición de groundtruth sobre la solución e histogramas solución vs groundtruth (ab) y groundtruth vs solución (ba). Obtenemos valores medios de $7,18 \cdot 10^{-1}$ y $1,54 \cdot 10^{-1}$ respectivamente para ab y ba.

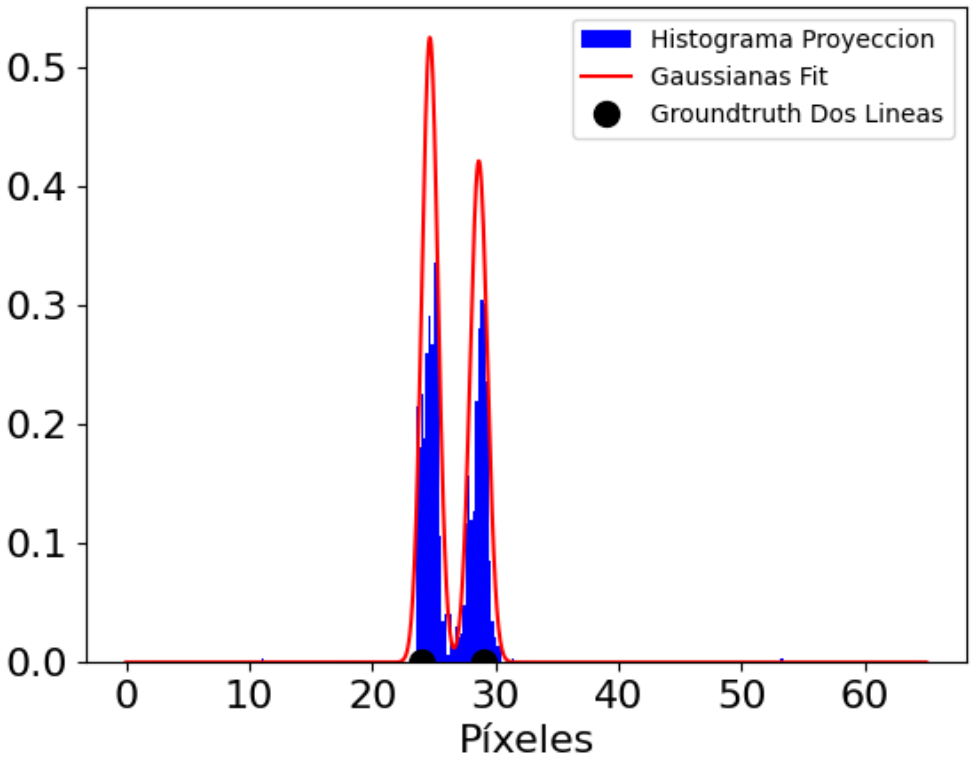


Figura 4.23: Superposición del histograma dado por la proyección y las dos gaussianas obtenidas por el algoritmo. Agregamos la posición horizontal del groundtruth.

Esta última herramienta, aplicando las definiciones de sesgo y mejora de resolución

definidas cuando fue presentada, nos devuelve un sesgo de $E = 0,098$ y una mejora de resolución de $M = 7,8$ en este caso, donde la separación de las líneas es casi 3 veces menor que la resolución original.

Capítulo 5

Conclusiones

En este trabajo hemos hecho un estudio exhaustivo del problema de la deconvolución, describiéndolo matemáticamente, mostrando por qué este problema resulta "mal puesto" y mostramos que la pseudoinversa no resulta continua. Luego, dimos lugar al estudio de los operadores compactos para poder desembocar en los métodos de regularización. Definimos la "one-half condition" y la calificación de dichos métodos, además de describir los principales métodos de este tipo (Tikhonov, Landweber, etc.).

Posteriormente, tomamos un enfoque alternativo para abordar el hecho de que este problema está "mal puesto". En vez de agregar términos de regularización a la función objetivo, se adiciona información sobre el objeto. En el caso de SUPPOSe esto se hace, aproximando el objeto real por una superposición de fuentes virtuales de igual intensidad. Acá también tenemos un problema de optimización, donde se quiere minimizar el error entre la muestra dada por SUPPOSe y la muestra real. Para ello, hacemos uso de un algoritmo de tipo genético que definimos en la Sección 3.2.

Definimos una variante del método SUPPOSe, llamada DSUPPOSe, cuyo objetivo es mejorar la velocidad de convergencia, dado que la operación computacional de realizar una convolución en cada paso del algoritmo resulta en un alto porcentaje del tiempo de cómputo. Para ello, se discretiza el espacio donde las fuentes virtuales se localizan. Este método utiliza otra función objetivo, que ahora no requiere de hacer el cálculo de la convolución en cada evaluación, ya que se reemplaza por el producto matricial de Transformadas de Fourier discreta.

Más adelante, hemos descrito nuestro caso de estudio. Definimos la imagen sintética a

procesar y cómo la hemos construido. Una vez que contamos con todas estas cuestiones, procesamos nuestro caso de estudio con la implementación Python de DSUPPOSe realizada para este trabajo. Obtuvimos la solución y definimos nociones para cuantificar nuestros resultados. Mostrando que el método podía distinguir objetos a distancias alrededor de 3 veces menor que la resolución original, con un factor de mejora en la exactitud de 5.

También, realizamos comparaciones entre la implementación realizada para DSUPPOSe y los resultados obtenidos con el mismo caso de estudio, para todos los métodos de regularización definidos en la Sección 2.4. Realizamos una comparación integral, barriendo los parámetros de cada método de regularización, para obtener soluciones que en algunos casos nos neutralizaban el ruido y "suavizaban" la imagen, mientras que en otros obtuvimos artificios y configuraciones que nada tenían que ver con el objeto original. Para todos los métodos obtuvimos resultados de este estilo, salvo con el método de Richardson-Lucy. Describimos dicho método y mencionamos que este algoritmo funcionaba muy bien con imágenes que fueron generadas con ruido de tipo Poisson (que fue el caso de nuestra imagen sintética a estudiar). Por ello, definimos un nuevo caso de estudio *ad-hoc* que simulaba también el ruido de la electrónica, el cual fue modelado con una distribución normal (gaussiana), y además simulamos un error en la adquisición de la PSF. Con el método de Richardson-Lucy obtuvimos una solución realmente mala, que no logra resolver la configuración de las dos líneas paralelas. Sin embargo, procesando el mismo caso de estudio para nuestra implementación de DSUPPOSe, obtenemos una solución en la que sí podemos distinguir el esquema de las dos líneas, con un factor de mejora de 7.

5.1. Trabajo futuro

Como parte de la conclusión, es menester agregar algunos comentarios respecto a tareas pendientes que se desprenden de este trabajo:

1. Uno de los puntos principales de esta tesis fue la implementación en Python del método DSUPPOSe para el caso en que la imagen no tiene fondo. Como en casi todas las mediciones es natural que aparezca un fondo, sería de interés implementar este caso, al igual que en [20]. Así como hacer un análisis de tiempos de computo.

2. Optimizar los parámetros utilizados en el algoritmo genético para mejorar la veloci-

dad. Si bien los parámetros utilizados aquí son los mismos que en trabajos previos, sería de interés diseñar algun método estadístico que elija la mejor elección de estos parámetros.

3. Para acelerar la convergencia, utilizar o combinar el algoritmo genético con algún otro tipo de método de optimización, como el método ADAM utilizado en [37].

4. Hacer un análisis estadístico del desempeño variando distintos parámetros de la muestra: separación de los objetos, ruido de la imagen, tipo de ruido, etc. Así como fabricar un conjunto de datos con distintas realizaciones de la misma muestra, similar a [18]. También sería de interés analizar cómo es la resolución del método con respecto a la topología del objeto. En muchos casos de interés se busca distinguir radios de circunferencias, circunferencias concéntricas, contiguas y otro tipo de geometrías no resueltas.

Bibliografía

[1] F. Pedrotti, L. M. Pedrotti, and L. S. Pedrotti, *Introduction to Optics*. Cambridge University Press, 2007.

[2] J. Hadamard, "Sur les problèmes aux dérivées partielles et leur signification physique.," *Princeton University Bulletin*, 1902.

[3] D. Sage, L. Donati, F. Soulez, D. Fortun, G. Schmit, A. Seitz, R. Guiet, C. Vonesch, and M. Unser, "Deconvolutionlab2: An open-source software for deconvolution microscopy," *Methods*, vol. 115, pp. 28–41, 2017.

[4] A. N. Tikhonov, "On the solution of ill-posed problems and the method of regularization," in *Doklady Akademii Nauk*, vol. 151, pp. 501–504, Russian Academy of Sciences, 1963.

[5] F. Lenzen and O. Scherzer, "Tikhonov regularization for nonlinear ill-posed problems," *EC-COMAS*, 2004.

[6] N. Dey, L. Blanc-Feraud, C. Zimmer, P. Roux, Z. Kam, J.-C. Olivo-Marin, and J. Zerubia, "Richardson-lucy algorithm with total variation regularization for 3d confocal microscope deconvolution," *Microscopy research and technique*, vol. 69, no. 4, pp. 260–266, 2006.

[7] L. B. Lucy, "An iterative technique for the rectification of observed distributions," *The astronomical journal*, vol. 79, p. 745, 1974.

[8] D. L. Donoho, "Superresolution via sparsity constraints," *SIAM journal on mathematical analysis*, vol. 23, no. 5, pp. 1309–1331, 1992.

[9] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on information theory*, vol. 52, no. 2, pp. 489–509, 2006.

[10] E. J. Candes, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, vol. 59, no. 8, pp. 1207–1223, 2006.

[11] V. I. Morgenshtern and E. J. Candes, "Super-resolution of positive sources: The discrete setup," *SIAM Journal on Imaging Sciences*, vol. 9, no. 1, pp. 412–444, 2016.

[12] S. Martínez, M. Toscani, and O. E. Martínez, "Superresolution method for a single wide-field image deconvolution by superposition of point sources," *Journal of Microscopy*, vol. 275, no. 1, pp. 51–65, 2019.

[13] M. Toscani and S. Martínez, "Solving the boundary artifact for the enhanced deconvolution algorithm suppose applied to fluorescence microscopy," *Computer Optics*, vol. 45, no. 3, pp. 418–426, 2021.

[14] S. Martínez, M. Toscani, and O. E. Martínez, "Supplementary material of the article superresolution method for a single wide-field image deconvolution by superposition of point sources," *Journal of Microscopy*, 2019.

[15] D. E. Goldberg, "Optimization, and machine learning," *Genetic algorithms in Search*, 1989.

[16] A. H. Wright, "Genetic algorithms for real parameter optimization," in *Foundations of genetic algorithms*, vol. 1, pp. 205–218, Elsevier, 1991.

[17] M. Toscani, A. Mazzeo, S. Martínez, O. Martínez, A. Lacapmesure, and G. B. Vazquez, "Fuentes de error, artificios, aceleración y validación del algoritmo de deconvolución con super-resolución para imágenes de microscopía," in *2020 IEEE Congreso Bienal de Argentina (ARGENCON)*, pp. 1–7, IEEE, 2020.

[18] M. Toscani, O. E. Martínez, and S. Martínez, "Resolution, accuracy and precision in super-resolved microscopy images using suppose," *Optics and Lasers in Engineering*, vol. 161, p. 107337, 2023.

[19] G. D. B. Vazquez, S. Martínez, and O. E. Martínez, "Super-resolved edge detection in optical microscopy images by superposition of virtual point sources," *Optics Express*, vol. 28, no. 17, pp. 25319–25334, 2020.

[20] S. Martínez and O. E. Martínez, "Discrete suppose: A new, faster and accurate superresolution method for appications to fluorescence microscopy images," *ScienceDirect*, vol. 9, no. 7, pp. 4–7, 2023.

[21] M. Hanke, *A Taste of Inverse Problems: Basic Theory and Examples*. SIAM, 2017.

[22] S. Lu and S. V. Pereverzev, *Regularization theory for ill-posed problems: selected topics*, vol. 58. Walter de Gruyter, 2013.

[23] H. Brézis, *Functional Analysis, Sobolev Spaces and Partial Differential Equations*. Springer, 2010.

[24] V. A. Morozov, "On restoring functions by the regularization method," *USSR Computational Mathematics and Mathematical Physics*, vol. 7, no. 4, pp. 208–219, 1967.

[25] G. H. Golub, M. Heath, and G. Wahba, "Generalized cross-validation as a method for choosing a good ridge parameter," *Technometrics*, vol. 21, no. 2, pp. 215–223, 1979.

[26] P. C. Hansen, "Analysis of discrete ill-posed problems by means of the l-curve," *SIAM review*, vol. 34, no. 4, pp. 561–580, 1992.

[27] M. P. and P. S. V., "Geometry of linear ill-posed problems in variable hilbert scales. citation peter mathé and sergei v pereverzev 2003 inverse problems 19 789-803," 2003.

[28] L. Grippo and M. Sciandrone, *Introduction to Methods for Nonlinear Optimization*, vol. 152. Springer Nature, 2023.

[29] J. Holland, "Adaptation in natural and artificial systems, university of michigan press, ann arbor,"," *Cité page*, vol. 100, 1975.

[30] M. Toscani, *Deconvolución con super-resolución en imágenes de microscopía por superposición de fuentes virtuales.* Tesis doctoral, Universidad de Buenos Aires, 2021.

[31] L. Davis, "Adapting operator probabilities in genetic algorithms," in *Proceedings of the 3rd International Conference on Genetic Algorithms*, pp. 61–69, 1989.

[32] F. Herrera, M. Lozano, and J. L. Verdegay, "Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis," *Artificial intelligence review*, vol. 12, no. 4, pp. 265–319, 1998.

[33] D. E. Goldberg *et al.*, *Real-coded genetic algorithms, virtual alphabets and blocking*. Citeseer, 1990.

[34] F. A. Bongiovanni, "Repositorio DSUPPOSe Python." https://gitlab.com/bongiovannifranco2b/codigo-tesis, 2025.

[35] A. M. Lacapmesure, S. Martínez, and O. E. Martínez, "A new objective function for super-resolution deconvolution of microscopy images by means of a genetic algorithm," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, pp. 271–272, 2020.

[36] A. M. Lacapmesure, *Detección con súper-resolución de estructuras ralas en imágenes ópticas mediante nuevas variantes del algoritmo SUPPOSe*. Tesis doctoral, Universidad de Buenos Aires, 2024.

[37] A. M. Lacapmesure, G. D. B. Vazquez, A. Mazzeo, S. Martínez, and O. E. Martínez, "Combining deep learning with suppose and compressed sensing for snr-enhanced localization of overlapping emitters," *Applied Optics*, vol. 61, no. 7, pp. D39–D49, 2022.