Facultad

de

Ciencias

Tratamiento de imágenes mediante ecuaciones en derivadas parciales

(Image processing using partial differential equations)

Trabajo de Fin de Grado

Para acceder al

GRADO EN MATEMÁTICAS

Autora: María Díaz Alzórriz

Director: Rafael Granero Belinchón

Fecha: Febrero - 2022

Agradecimientos

A mi director del TFG, Rafa, por haber hecho posible este trabajo.

A Javi, por haberse convertido en un pilar fundamental en mi vida. Aunque ya me gustaban las matemáticas, el poder compartirlas contigo ha hecho que me gusten todavía más. Gracias por tu ayuda y apoyo en todo momento.

A mis amigas, tanto las de siempre como las que he hecho en estos años de carrera. En especial, a mis compañeras de piso por haber sido mi segunda familia en Santander.

Por último y lo más importante, a mi familia, por apoyarme incondicionalmente en todas mis decisiones, por su cariño y por haber confiado siempre en mí.

Tengo mucha suerte de teneros a todos.

Resumen

Las ecuaciones en derivadas parciales tienen multitud de aplicaciones. Por ejemplo, son una de las bases de la descripción de los fenómenos físicos y químicos. Además de este papel en las ciencias naturales, las ecuaciones en derivadas parciales también están en el tratamiento de imágenes moderno.

En este proyecto se propone estudiar diversas ecuaciones importantes en el tratamiento de imágenes. En particular, además del estudio analítico de las ecuaciones en derivadas parciales, se harán aplicaciones a imágenes reales.

Palabras clave: Siluetas, ecuación de Poisson, Lema de Lax-Milgram, métodos iterativos.

Abstract

Partial differential equations have plenty of applications. For instance, they are one of the foundations of physical and chemical phenomena. Besides its role in natural sciences, partial differential equations appear in modern image processing.

In this project we propose to study diverse important equations in image processing. In particular, apart from the analytic study of partial differential equations, applications to real images will be made.

Key words: Silhouettes, Poisson equation, Lax-Milgram Lemma, iterative methods.

Índice general

Capítulo 1

Introducción

1.1. Motivación / Objetivo

Si nos detenemos a observar por ejemplo un edificio, nos damos cuenta de sus características: si su tejado es plano o picudo, si sus ventanas son amplias, pequeñas, rectangulares o redondas, o de cualquier matiz que podamos examinar. De la misma forma ocurre cuando observamos la naturaleza, donde se pueden apreciar todas las clases de árboles y plantas que hay, más altos, frondosos, con hojas o sin ellas, etc. Lo mismo ocurre con los animales y las personas: la forma de su cabeza, sus ojos, su nariz o su cuerpo. O incluso de este mismo texto en el que cada letra tiene su propia forma que la diferencia de las demás.

Cuando ves una imagen, es muy fácil para tu cerebro reconocer qué estás viendo y diferenciarlo de otra imagen. Sin embargo, ¿cómo hace este proceso un ordenador? Una de las aplicaciones del análisis matemático consiste en el estudio y clasificación de imágenes. Matemáticamente, hay muchas propiedades que nos indican cómo es la forma de cada imagen y que ayudan a distinguir unas figuras de otras y sobre esto trata esta memoria. Nuestro objetivo es entender las imágenes desde un punto de vista matemático, más allá de lo que vemos con los ojos simplemente.

En este trabajo estudiaremos analíticamente la existencia de solución débil para problemas elípticos usando el conocido lema de Lax-Milgram. Además, introduciremos brevemente varios algoritmos iterativos para la resolución de la ecuación de Poisson en un dominio dos-dimensional. Finalmente, utilizaremos estas ideas y técnicas para desarrollar un método que extrae información de una determinada silueta dada. Dicho

método podría integrarse en códigos más complejos de forma que, una vez que el programa estuviese adecuadamente afinado, el ordenador pudiese distinguir unas siluetas de otras.

Para ello, en el Capítulo 2 resolveremos un problema formado por la ecuación de Poisson junto con una condición de contorno, obteniendo así una representación matemática de la imagen mediante una función solución del problema. En dicho capítulo se va a integrar toda la parte teórica, donde explicaremos con mayor profundidad el problema a resolver, así como la existencia y unicidad de su solución, previamente a su aproximación numérica.

En el Capítulo 3 se implementarán algunos métodos en Matlab con el objetivo de resolver un caso particular del problema que estudiemos en el Capítulo 2. A su vez, se desarrollarán varios ejemplos que muestren visualmente las principales diferencias entre las imágenes.

Es interesante ver cómo a lo largo de la memoria se complementarán distintas ramas del análisis, tanto diferencial como numérico. Así como se observará que las matemáticas, además de ser teóricas y rigurosas, también presentan numerosas aplicaciones prácticas, y en este trabajo se pone de manifiesto una de ellas.

Capítulo 2

Formulación matemática del problema

Comencemos notando que una imagen digital está formada por un conjunto definido de puntos llamados píxeles, formando una matriz. Cada píxel de una imagen almacena la información de su tono o luminosidad, donde el tono negro es el valor 0 y el blanco es el valor 255. En concreto, lo que nosotros entenderemos por silueta, será un dominio acotado del plano que encierra un área en color blanco sobre un fondo en negro, es decir, una imagen que solo tiene dos colores. Debemos notar que estamos acostumbrados a ver siluetas sobre fondos blancos, justo al revés. Sin embargo, a lo largo de esta memoria siempre nos referiremos a las siluetas como se acaba de explicar. Para describirlas, será necesario extraer propiedades que nos ayuden a distinguir unas siluetas de otras. Para ello, nos vamos a servir de las ecuaciones diferenciales. En particular, utilizaremos la llamada ecuación de Poisson. La ecuación de Poisson es una ecuación en derivadas parciales elíptica de segundo orden cuya expresión en el caso bidimensional para una función $u(x, y) \in C^2(\Omega)$ está definida como:

$$\Delta u(x, y) = -f(x, y),$$

donde Δ es el operador laplaciano y $f(x, y)$ una función conocida.

El problema matemático consistirá en resolver un problema de contorno. Dicho problema estará formado por la ecuación de Poisson en el dominio Ω y unas ciertas condiciones de frontera definidas sobre la frontera $\partial\Omega$:

$$\begin{cases} \Delta u(x,y) = -f(x,y) & \text{si } (x,y) \in \Omega, \quad \text{Ecuación de Poisson,} \\ u(x,y) = 0 & \text{si } (x,y) \in \partial\Omega, \quad \text{Condición de Contorno.} \end{cases} \tag{2.1}$$

Es decir, tenemos una condición de contorno Dirichlet -al aplicarse sobre la función $u(x,y)$- y homogénea, al ser esta condición nula. Si bien el objetivo es encontrar una función u definida sobre un dominio $\Omega \subset \mathbb{R}^2$ que satisfaga la ecuación de Poisson con condición de contorno Dirichlet homogénea en la frontera, lo cierto es que encontrar soluciones que verifiquen de forma clásica una ecuación en derivadas parciales es un problema generalmente complicado. Para acercarnos a su resolución, primero vamos a relajar el concepto de solución y vamos a hablar de las llamadas soluciones débiles, que satisfacen la ecuación en un sentido integral.

En este capítulo nos centraremos en demostrar la existencia y unicidad de solución de dicho problema. Para ello, será necesario definir primero una serie de conceptos y resultados teóricos previos.

2.1. Marco teórico

Antes de introducir el concepto de solución débil y los resultados que garantizan su existencia, vamos a definir unos espacios de clases de funciones $u : \Omega \longrightarrow \mathbb{R}$, donde $\Omega \subset \mathbb{R}^2$ es un dominio acotado.

Por un lado, se define $\mathcal{L}^2(\Omega)$ como el espacio de funciones $u : \Omega \longrightarrow \mathbb{R}$ de cuadrado integrable en Ω, es decir:

$$\mathcal{L}^2(\Omega) := \left\{ u : \Omega \longrightarrow \mathbb{R} \text{ integrable y } \int_\Omega u^2 dx < +\infty \right\}.$$

Tomando clases de equivalencia, donde se identifican dos funciones de $\mathcal{L}^2(\Omega)$ mediante la relación de equivalencia $\sim$ si coinciden en casi todo Ω excepto en un conjunto de medida Lebesgue nula, se define el espacio $L^2(\Omega)$ como:

$$L^2(\Omega) := \mathcal{L}^2(\Omega)/\sim .$$

El producto escalar de $L^2(\Omega)$ es:

$$(u,v)_{L^2(\Omega)} = \int_\Omega uv\,dx \qquad \forall u,v \in L^2(\Omega)$$

cuya norma asociada es:

$$\|u\|_{L^2(\Omega)} = \left(\int_\Omega u^2 dx.\right)^{1/2} \tag{2.2}$$

Por otro lado, definimos el espacio de Sobolev $H^1(\Omega) \subset L^2(\Omega)$ como el espacio de clases de funciones de $L^2(\Omega)$ las cuales todas sus derivadas parciales de primer orden son elementos de $L^2(\Omega)$. Matemáticamente:

$$H^1(\Omega) := \left\{ u \in L^2(\Omega) : \frac{\partial u}{\partial x_i} \in L^2(\Omega), \text{ para } i = 1,\ldots,N \right\}.$$

Así mismo, se define el producto escalar de $H^1(\Omega)$, el cual se puede expresar en función del producto de $L^2(\Omega)$, como:

$$(u,v)_{H^1(\Omega)} = (u,v)_{L^2(\Omega)} + (\nabla u, \nabla v)_{L^2(\Omega)} = \int_\Omega uv\,dx + \int_\Omega \nabla u \cdot \nabla v\,dx \qquad \forall u,v \in H^1(\Omega),$$

con la norma asociada:

$$\|u\|_{H^1(\Omega)} = \left(\|u\|^2_{L^2(\Omega)} + \|\nabla u\|^2_{L^2(\Omega)}\right)^{1/2} = \left(\int_\Omega u^2 dx + \int_\Omega |\nabla u|^2 dx\right)^{1/2} \tag{2.3}$$

En tercer lugar, introducimos el espacio $H^1_0(\Omega) \subset H^1(\Omega)$ como las funciones de $H^1(\Omega)$ que se anulan en la frontera de Ω, esto es:

$$H^1_0(\Omega) := \left\{ u \in H^1(\Omega) : u = 0 \text{ si } x \in \partial\Omega \right\}.$$

con el producto escalar y norma inducidas.

A lo largo de la sección, H será un espacio de Hilbert separable y H^* su espacio dual, definido como el conjunto de todos los funcionales lineales y continuos en H y cuya

norma es:

$$\|\varphi\|_{H^*} = \sup_{\|x\|\leq 1} |\varphi(x)| = \sup_{\|x\|\leq 1} \varphi(x),$$

con $x \in H$ y $\varphi \in H^*$. Denotaremos por $\langle \varphi, x\rangle = \varphi(x)$ a la dualidad cuando $\varphi \in H^*$ y $x \in H$.

Por otro lado, definimos $a(\cdot,\cdot) : H \times H \longrightarrow \mathbb{R}$, una forma bilineal simétrica continua y coerciva, esto es:

- Continua: existe una constante $C > 0$ tal que $|a(u,v)| \leq C\|u\|_H\|v\|_H$ para todo $u, v \in H$.
- Coerciva: existe una constante $\alpha > 0$ tal que $a(u,u) \geq \alpha\|u\|_H^2$ para todo $u \in H$.

Una vez introducida toda la notación necesaria, nos centraremos por último lugar en los resultados teóricos necesarios para enunciar el resultado de existencia y unicidad de solución del problema (2.1). Comenzaremos por el Teorema de Representación de Riesz:

Teorema 2.1.1 (Representación de Riesz) *Dada $\varphi \in H^*$, existe un único $f \in H$ tal que:*

$$\langle \varphi, u\rangle = (f,u)_H \qquad \forall u \in H \tag{2.4}$$

siendo $(f,u)_H$ el producto escalar definido en H. Además,

$$\|f\|_H = \|\varphi\|_{H^*}.$$

Notemos que escribimos $\langle \varphi, u\rangle$ debido a que $\varphi \in H^*$ y $u \in H$. La demostración de este Teorema se puede consultar en una amplia bibliografía relativa a análisis funcional, de la cual se ha consultado [1].

Por último ya podemos enunciar y demostrar el resultado que utilizaremos para asegurar la existencia y unicidad de solución débil del problema de contorno, el lema de Lax-Milgram.

Lema 2.1.2 (Lax-Milgram) *Sea $a(u,v)$ una forma bilineal, continua y coerciva sobre*

H. Entonces, para todo $\varphi \in H^*$, existe un único elemento $u \in H$ tal que:

$$a(u,v) = \langle \varphi, v \rangle \qquad \forall v \in H. \tag{2.5}$$

Su demostración se recoge a continuación y ha sido adaptada de [1] junto con [3]. En ella se utilizará el Teorema de Representación de Riesz introducido previamente.

Demostración. Fijamos $u \in H$ y definimos el operador:

$$\langle \varphi_u, v \rangle = a(u,v) : H \longrightarrow \mathbb{R}. \tag{2.6}$$

Se observa que φ_u es un funcional lineal continuo y por tanto, $\varphi_u \in H^*$. Con lo cual, el Teorema de Representación de Riesz afirma la existencia de un único elemento $f_u \in H$ que satisface:

$$\langle \varphi_u, v \rangle = (f_u, v) \tag{2.7}$$

Como podemos hacer esto para todo $u \in H$, se define la aplicación:

$$T : H \longrightarrow H$$

tal que

$$T(u) = f_u. \tag{2.8}$$

Veamos primero que T es un operador lineal. En efecto, si $\lambda, \beta \in \mathbb{R}$ y $u, w \in H$, tenemos que para cada $v \in H$ dada arbitrariamente:

$$\begin{aligned}
(T(\alpha u + \beta w), v)_H &= (f_{\alpha u + \beta w}, v)_H && \text{por (2.8)} \\
&= \langle \varphi_{\alpha u + \beta w}, v \rangle && \text{por (2.7)} \\
&= a(\alpha u + \beta w, v) && \text{por (2.6)} \\
&= (\alpha u + \beta w, v)_H \\
&= \alpha (u, v)_H + \beta (w, v)_H \\
&= \alpha a(u,v) + \beta a(w,v) \\
&= \langle \alpha \varphi_u, v \rangle + \beta \langle \varphi_w, v \rangle \\
&= (\alpha T(u), v)_H + (\beta T(w), v)_H \\
&= (\alpha T(u) + \beta T(w), v)_H
\end{aligned}$$

de donde se tiene:

$$T(\alpha u + \beta w) = \alpha T(u) + \beta T(w).$$

Asímismo, veamos que T es continuo:

$$\begin{aligned}
\|T(u)\|_H^2 &= \|f_u\|_H^2 && \text{por (2.8)} \\
&= (f_u, f_u)_H \\
&= \langle \varphi, f_u \rangle && \text{por (2.7) con } v = f_u \\
&= a(u, f_u) && \text{por (2.6)} \\
&\leq \|u\|_H \|f_u\|_H && \text{por la continuidad de } a
\end{aligned}$$

de donde llegamos a que:

$$\|T(u)\|_H \leq \|u\|_H \qquad \forall u \in H.$$

A continuación, veremos que el operador T es invertible. Para ello empezamos viendo que T está acotado inferiormente:

$$\begin{aligned}
\|u\|_H^2 &\leq a(u, u) && \text{por la coercividad de a} \\
&= \langle \varphi_u, u \rangle && \text{por (2.6)} \\
&= (f_u, u)_H && \text{por (2.7) con } v = u \\
&\leq \|f_u\|_H \|u\|_H \\
&\leq \|T(u)\|_H \|u\|_H
\end{aligned}$$

de donde obtenemos:

$$\|u\|_H \leq \|T(u)\|_H \qquad \forall u \in H. \tag{2.9}$$

Gracias a la linealidad de T podemos observar que:

$$\|u_1 - u_2\|_H \leq \|T(u_1 - u_2)\|_H \leq \|T(u_1) - T(u_2)\|_H,$$

de donde se sigue directamente la inyectividad de T ya que si $T(u_1) = T(u_2)$, implicaría $u_1 = u_2$. Ahora bien, al hablar de inyectividad, se puede afirmar que T posee inversa. Por tanto, podemos definir el operador inverso T^{-1} como:

$$T^{-1} : R \subset H \longrightarrow H,$$

donde R es el rango de T. Además, observamos que T^{-1} es continuo:

$$\|T^{-1}(f_u)\|_H = \|u\|_H \leq \|T(u)\|_H \leq \|f_u\|_H,$$

donde hemos utilizado la desigualdad (2.9).

Hasta aquí lo que hemos probado es que, dada $f \in R$, entonces existe $u \in H$ solución de:

$$a(u, v) = \langle \varphi, v \rangle \qquad \forall v \in H.$$

Falta demostrar que $R = H$. Vamos a ver primero que R es cerrado. Para ello, tomamos una sucesión $f_n \in R \longrightarrow f \in H$ y vamos a demostrar que $f \in R$. Como T es lineal y esta acotado inferiormente, se tiene:

$$\|f_n - f_m\|_H = \|T(u_n) - T(u_m)\|_H = \|T(u_n - u_m)\|_H \geq \|u_n - u_m\|_H.$$

Entonces, al ser f_n una sucesión convergente, u_n una sucesión de Cauchy y H un espacio de Hilbert, podemos afirmar que u_n es una sucesión convergente. Sea u el límite de esta sucesión u_n y dado que T es continua, se tiene:

$$T(u) = \lim_{n\to\infty} T(u_n) = \lim_{n\to\infty} f_u = f.$$

Por tanto, $f \in R$, luego R es cerrado. Ahora procedemos por reducción al absurdo, suponiendo que $R \neq H$. Al ser R cerrado, definimos el conjunto $R^\perp$ el cual es no vacío, es decir, existe $0 \neq g \in R^\perp \subset H$ tal que:

$$(f, g)_H = 0 \qquad \forall f \in R.$$

Entonces:

$$\begin{aligned}
\|g\|_H^2 &\leq a(g, g) && \text{por la coercividad de } a \\
&= \langle \varphi_g, g \rangle && \text{por (2.6)} \\
&= (f_g, g)_H && \text{por (2.7)} \\
&= (T(g), g)_H && \text{por (2.8)}
\end{aligned}$$

Y como $T(g) \in R$, entonces $(T(g), g)_H = 0$, lo cual contradice que $g \neq 0$, por lo que debe ser $R = H$. Por último, falta ver que la solución u es única. Para ello, supongamos

que existe otra solución distinta $\tilde{u} \in H$ tal que

$$a(\tilde{u}, v) = \langle \varphi, v \rangle \quad \forall v \in H.$$

Usando la bilinealidad de a tenemos:

$$a(u - \tilde{u}, v) = a(u, v) - a(\tilde{u}, v) = \langle \varphi, v \rangle - \langle \varphi, v \rangle = 0.$$

Y gracias a la coercividad de a:

$$\|v\|_H^2 \leq a(v, v) = a(u - \tilde{u}, v) = 0,$$

donde hemos tomado $v = u - \tilde{u}$. De esta manera, obtenemos:

$$\|u - \tilde{u}\|_H = 0,$$

de donde debe ser $u = \tilde{u}$. Con lo cual hemos probado la unicidad de solución y con ello concluimos la demostración.

■

2.2. Existencia y unicidad

En esta sección vamos a comenzar por reformular el problema (2.1) para llevarlo a su forma débil. Posteriormente comprobaremos que dicha reformulación se encuentra bajo las hipótesis del lema de Lax-Milgram, gracias al cual podremos garantizar la existencia y unicidad de solución débil.

Se ha trabajado con los espacios de funciones $L^2(\Omega)$ y como espacio H presente en la Sección 2.1 se ha tomado $H = H_0^1(\Omega)$.

En primer lugar, se precisa lo que se entiende por solución clásica y solución débil. Una solución clásica de (2.1) es una función $u \in C^2(\Omega)$ que satisface dicho problema en el sentido usual. Sin embargo, una solución débil de (2.1) es una función $u \in H_0^1(\Omega)$ que verifica:

$$\int_\Omega \nabla u \cdot \nabla v \, dx = \int_\Omega f v \, dx, \qquad \forall v \in H_0^1(\Omega). \tag{2.10}$$

A continuación vamos a deducir que toda solución clásica es débil. Sea u una solución clásica de (2.1). Se multiplica la ecuación de Poisson por una función de prueba $v \in H_0^1(\Omega)$ y posteriormente se integra en el dominio Ω, llegando a:

$$\int_\Omega \Delta u \cdot v \; dx = \int_\Omega -fv \; dx.$$

En este momento aplicamos a la parte izquierda la primera identidad de Green sobre Ω, la cual es:

$$\int_\Omega \Delta u \cdot v \, dx = \int_{\partial\Omega} (\nabla u \cdot \mathbf{n}) v \, dS - \int_\Omega \nabla u \cdot \nabla v \, dx$$

siendo $\mathbf{n}$ el vector normal exterior a Ω. Como $v \in H_0^1(\Omega)$, v se anula en $\partial\Omega$ y en consecuencia la integral sobre la frontera $\partial\Omega$ es nula. Así, se llega al resultado (2.10).

De este modo estamos ampliando la búsqueda de solución, lo cual nos interesa puesto que es mucho más complicado encontrar soluciones clásicas que soluciones débiles. Dicho de otra manera, al contar con más soluciones débiles que clásicas y al tener que toda solución clásica es solución débil, entonces reformulando el problema a su forma débil nos resultará más sencillo encontrar alguna solución.

Partiendo de dicha formulación débil (2.10), definimos como forma bilineal $a(\cdot,\cdot)$ la siguiente:

$$a(u,v) = \int_\Omega \nabla u \cdot \nabla v \, dx = (\nabla u, \nabla v)_{L^2(\Omega)} \qquad \forall u, v \in H_0^1(\Omega). \tag{2.11}$$

Esta forma bilineal es simétrica, y veremos a continuación que también es continua y coerciva para funciones en el espacio $H_0^1(\Omega)$.

En primer lugar vamos a demostrar la continuidad. Partiendo de la definición de la forma bilineal $a(u,v) = (\nabla u, \nabla v)_{L^2(\Omega)}$ y utilizando la desigualdad de Cauchy-Schwarz, se tiene:

$$(\nabla u, \nabla v)_{L^2(\Omega)} \leq \|\nabla u\|_{L^2(\Omega)} \, \|\nabla v\|_{L^2(\Omega)}.$$

Por otro lado, a partir de las normas asociadas en $L^2(\Omega)$ y $H_0^1(\Omega)$ descritas en (2.2) y

(2.3), respectivamente, se deduce que:

$$\|\nabla u\|_{L^2(\Omega)} \leq \|u\|_{H_0^1(\Omega)} \qquad \forall\, u \in H_o^1(\Omega)$$

y finalmente se llega a:

$$a(u,v) \leq \|u\|_{H_0^1(\Omega)} \|v\|_{H_0^1(\Omega)},$$

como queríamos demostrar.

A continuación se demuestra la coercividad, para lo cual hacemos uso de la desigualdad de Poincaré. Su demostración se puede encontrar en [1] y enuncia que:

$$\|u\|_{L^2(\Omega)} \leq C\|\nabla u\|_{L^2(\Omega)} \qquad \forall u \in H_0^1(\Omega)$$

con $C > 0$ una constante. Partiendo de la definición de la norma en $H_0^1(\Omega)$ en (2.3), tenemos:

$$\begin{aligned} \|u\|^2_{H_0^1(\Omega)} &= \|u\|^2_{L^2(\Omega)} + \|\nabla u\|^2_{L^2(\Omega)} \\ &\leq (C^2+1)\|\nabla u\|^2_{L^2(\Omega)}, \end{aligned}$$

donde hemos utilizado la desigualdad de Poincaré. Así, se llega a:

$$a(u,u) = \|\nabla u\|^2_{L^2(\Omega)} \geq (C^2+1)^{-1}\|u\|^2_{H_0^1(\Omega)}.$$

En consecuencia, la forma $a(\cdot,\cdot)$ definida en (2.11) es coerciva para funciones en el espacio $H_0^1(\Omega)$ con $\alpha = (C^2+1)^{-1}$.

Así, ya hemos probado que la forma bilineal $a(u,v) : H_0^1 \times H_0^1 \longrightarrow \mathbb{R}$ es continua y coerciva. Por otra parte, el funcional $f : H_0^1 \longrightarrow \mathbb{R}$ viene dado por:

$$\langle f, v\rangle = \int_\Omega fv \; dx,$$

el cual veremos que es un elemento de H^*. En efecto, es un funcional lineal y a continuación se demostrará su continuidad. Para ello utilizamos en primer lugar la desigualdad de Cauchy-Schwarz:

$$\langle f, v\rangle \leq \|f\|_{L^2(\Omega)}\|v\|_{L^2(\Omega)}.$$

Por otro lado, como $f \in L^2(\Omega)$, su norma es acotada y junto a la desigualdad de Poincaré se llega a:

$$\|f\|_{L^2(\Omega)}\|v\|_{L^2(\Omega)} \leq C\|\nabla v\|_{L^2(\Omega)} \leq C\|v\|_{H_0^1(\Omega)}.$$

Como conclusión, se cumplen las hipótesis del lema de Lax-Milgram, con lo que podemos asegurar la existencia de una u que satisface (2.10) y por lo tanto existe una única solución débil del problema (2.1).

Capítulo 3

Aplicación a las siluetas

Habiendo introducido en el capítulo anterior el problema tratado, en este último capítulo se implementrá un algoritmo que resuelva este problema tomando como dominio una silueta y se realizarán varios ejemplos.

3.1. Planteamiento del problema

En primer lugar, se presentan algunas imágenes de siluetas que ayuden a visualizar lo que se va a realizar. Como explicamos al principio del Capítulo 2, matemáticamente entendemos las siluetas como un dominio acotado del plano que encierra un área en color blanco sobre un fondo negro. Sin embargo, para poder representarlas de manera estética y práctica a lo largo del trabajo, lo haremos de manera contraria, como se muestra a continuación:

Figura 3.1: Siluetas variadas.

Así mismo, presentamos el problema a resolver en esta sección:

$$\begin{cases} \Delta u(x,y) = -1 & \text{si } (x,y) \in \Omega, \\ u(x,y) = 0 & \text{si } (x,y) \in \partial\Omega. \end{cases} \tag{3.1}$$

Podemos afirmar la existencia y unicidad de solución de este problema, pues ya ha sido demostrado de manera general para el problema (2.1) en el capítulo anterior. En particular, se toma $f(x,y) = 1$ y para esta función se cumplen todas las hipótesis necesarias.

En concreto, para la resolución de este problema procederemos por el método de diferencias finitas, el cual consiste en realizar una discretización del dominio formando un mallado y de las derivadas en los puntos de dicho mallado. Como hemos visto, tenemos condición de contorno Dirichlet en la frontera y nos interesa calcular los puntos interiores. Para ello, queremos encontrar una función $U(x,y)$ que aproxime $u(x,y)$ en los puntos del mallado.

Consideremos una silueta incrustada en una cuadrícula $n \times n$ con un tamaño de malla $h = 1/(n-1)$ rodeada por un contorno cerrado simple $\partial\Omega$. Vamos a calcular $\Delta U(x,y)$ mediante el método de diferencias finitas, en concreto diferencias finitas centradas, cuya expresión es:

$$\Delta U(x,y) \approx \frac{U(x+h,y) + U(x-h,y) + U(x,y+h) + U(x,y-h) - 4U(x,y)}{h^2}. \tag{3.2}$$

Para deducirla, se trabajará solamente con una de las dos coordenadas, en este caso la coordenada x, al ser su obtención análoga en ambas coordenadas. De este modo la componente y será constante en este proceso. Si bien se puede obtener de distintas formas, para llegar a la segunda derivada parcial en la coordenada x utilizaremos el desarrollo de Taylor de segundo orden de la función $U(x,y)$. A partir del desarrollo de $U(x,y)$ en el punto (x,y) evaluado en el punto del mallado $(x+h,y)$, se obtiene:

$$U(x+h,y) - U(x,y) = h\,\frac{\partial U(x,y)}{\partial x} + \frac{h^2}{2}\,\frac{\partial^2 U(x,y)}{\partial x^2} \tag{3.3}$$

ya que $(x+h,y) - (x,y) = (h,0)$. De manera similar, evaluando el desarrollo de Taylor en el punto $(x-h,y)$ se tiene:

$$U(x-h,y) - U(x,y) = -h\,\frac{\partial U(x,y)}{\partial x} + \frac{h^2}{2}\,\frac{\partial^2 U(x,y)}{\partial x^2} \tag{3.4}$$

donde en este caso $(x-h,y)-(x,y) = (-h,0)$. Operando (3.3) + (3.4):

$$U(x+h,y) - 2U(x,y) + U(x-h,y) = h^2\,\frac{\partial^2 U(x,y)}{\partial x^2}$$

y llegamos a:

$$\frac{\partial^2 U(x,y)}{\partial x^2} = \frac{U(x+h,y) - 2U(x,y) + U(x-h,y)}{h^2}.$$

Análogamente para la coordenada y se obtiene:

$$\frac{\partial^2 U(x,y)}{\partial y^2} = \frac{U(x,y+h) - 2U(x,y) + U(x,y-h)}{h^2}$$

y sumando ambas derivadas de segundo orden se llega a la expresión del laplaciano (3.2).

Puesto que $\Delta u(x,y) = -1$, podemos expresar (3.2) como:

$$-1 \approx \frac{U(x+h,y) + U(x-h,y) + U(x,y+h) + U(x,y-h) - 4U(x,y)}{h^2}. \tag{3.5}$$

Y por lo tanto, se obtiene:

$$U(x,y) \approx \frac{1}{4}\Big(h^2 + U(x+h,y) + U(x-h,y) + U(x,y+h) + U(x,y-h)\Big) \tag{3.6}$$

Utilizaremos esta expresión más adelante para implementar un método que resuelva el problema (3.1).

3.2. Métodos iterativos

En numerosas ocasiones los métodos de resolución directos de una ecuación no son efectivos, bien por la dimensión del problema o por su complejidad. Una alternativa a estos últimos son los llamados métodos iterativos. El concepto de los métodos iterativos consiste en la construcción de una sucesión de soluciones $x^{(k)}$ de tal forma que $\lim_{k\to\infty} x^{(k)} = \bar{x}$, donde $\bar{x}$ es la solución exacta. Esta sucesión se obtiene a partir de una solución inicial y la expresión:

$$x^{(k+1)} = F(x^{(k)})$$

donde F es una función que depende del método. En esta sección la bibliografía consultada ha sido [4] y [5].

A continuación vamos a introducir los principales métodos iterativos clásicos que sirven para resolver sistemas lineales $Ax = b$, siendo en nuestro caso $A \in \mathbb{R}^{N\times N}$ y b, $x \in \mathbb{R}^N$, con $N \in \mathbb{N}$.

Notar que en nuestro problema en particular, fijándonos en (3.5), tenemos que los vectores x y b son:

$$x = (U_{1,1}, \ldots, U_{1,n-1}, U_{2,1}, \ldots, U_{n-2,n-1}, U_{n-1,1}, \ldots, U_{n-1,n-1})^T$$

$$b = (-h^2, \ldots, -h^2)^T$$

ambos de dimensión $(n-1)^2$. Y la matriz A:

$$A = \begin{pmatrix} C & I & 0 & \cdots & \cdots & 0 \\ I & C & I & \ddots & & \vdots \\ 0 & I & C & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & I & C & I \\ 0 & \cdots & \cdots & 0 & I & C \end{pmatrix}$$

donde I es la matriz identidad de dimensión $(n-1)\times(n-1)$ y C es la siguiente matriz de la misma dimensión:

$$C = \begin{pmatrix} -4 & 1 & 0 & \cdots & \cdots & 0 \\ 1 & -4 & 1 & \ddots & & \vdots \\ 0 & 1 & -4 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & 1 & -4 & 1 \\ 0 & \cdots & \cdots & 0 & 1 & -4 \end{pmatrix}$$

Con ello, la matriz A es una matriz de dimensión $(n-1)^2 \times (n-1)^2$.

Los métodos iterativos clásicos son los métodos de Jacobi, Gauss-Seidel y SOR (relajación). Estos métodos presentan el siguiente esquema:

$$Mx^{(k+1)} = Nx^{(k)} + b \tag{3.7}$$

siendo M y N una descomposición de la matriz del sistema A de la forma $A = M - N$. Dependiendo de la elección de las matrices M y N se tiene un determinado método u otro.

Por ejemplo, si D es la matriz diagonal formada por la diagonal de A, entonces tomando $M = D$ y $N = -(A - D)$ se obtiene el método de Jacobi, el cual se va a realizar en una primera instancia. La implementación de este método utiliza toda la solución anterior $x^{(k)}$ para hallar un elemento de la nueva solución $x^{(k+1)}$, por lo que requiere el almacenamiento de ambas soluciones simultáneamente.

En nuestro caso concreto, implementando (3.6) con el método de Jacobi, se tiene:

$$U_{i,j}^{(k+1)} = \frac{h^2 + U_{i+1,j}^{(k)} + U_{i-1,j}^{(k)} + U_{i,j+1}^{(k)} + U_{i,j-1}^{(k)}}{4},$$

siendo k el orden de la iteración e i, j las filas y columnas de la matriz respectivamente.

Una de las maneras de optimizar el método anterior consiste en utilizar los elementos de la nueva solución $x^{(k+1)}$ ya calculados anteriormente para hallar el resto de elementos de la nueva solución, utilizando así la información más reciente y adicionalmente, almacenando solamente una solución. Esta mejora se corresponde al método de Gauss-

Seidel, el cual se obtiene descomponiendo la matriz $A = D + L + \tilde{U}$, siendo $\tilde{U}$ los elementos de A correspondientes a la parte triangular superior y L la matriz triangular inferior formada por los elementos de A de su parte triangular inferior. Así, con la elección de $M = D + L$ y $N = -\tilde{U}$ se llega a este método.

Adicionalmente y para acelerar la convergencia del método anterior, una vez hallada $x^{(k+1)}$, se realiza un promedio ponderado de ambas soluciones $x^{(k+1)}$ y $x^{(k)}$. La solución obtenida $\hat{x}^{(k+1)}$ es:

$$\hat{x}^{(k+1)} = \omega\hat{x}^{(k+1)} + (1-\omega)\hat{x}^{(k)} \quad \text{con } \omega \in (0,2). \tag{3.8}$$

Esta modificación se conoce como Gauss-Seidel con relajación (SOR).

3.3. Método de Jacobi en un cuadrado

Antes de presentar el código que resuelve el problema (3.1) en una silueta, vamos a comenzar implementado un algoritmo con el método de Jacobi que resuelve dicho problema en un cuadrado de dimensión $n \times n$ con condición de frontera Dirichlet homogénea. Esto nos servirá como base para, posteriormente, implementar el método al que se va a llegar. Como hemos comentado anteriormente, el algoritmo de Jacobi particularizado a nuestro problema es:

$$U_{i,j}^{(k+1)} = \frac{h^2 + U_{i+1,j}^{(k)} + U_{i-1,j}^{(k)} + U_{i,j+1}^{(k)} + U_{i,j-1}^{(k)}}{4}. \tag{3.9}$$

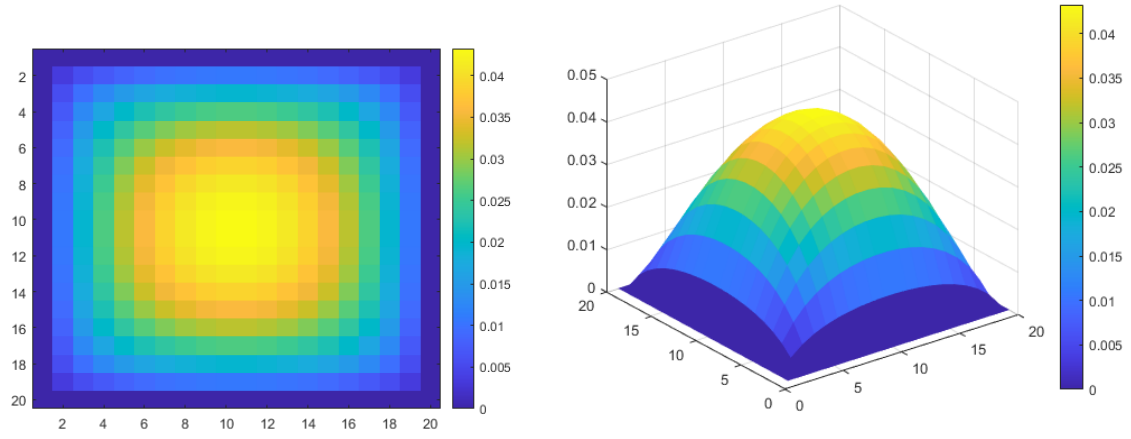
Este algoritmo se muestra a continuación:

```
function [u,iter,err]=jacobi_poisson(tol,itermax,n)
% Codigo que aproxima la solucion de la ecuacion de Poisson
% en un cuadrado utilizando el metodo de Jacobi.
% argumentos de entrada:
% tol es la tolerancia.
% itermax es el numero de iteraciones permitidas.
% n es la medida de los lados del cuadrado.
```

```
% argumentos de salida:
% u es la solucion.
% iter es el numero de iteraciones que se han producido.
% err indica el error entre la solucion exacta y la obtenida.
u=zeros(n,n);
h=1/(n+1);
err=1;
iter=0;
u_aprox=u;
while((err>tol)&(iter<=itermax))
for i=2:n-1
for j=2:n-1
    u_aprox(i,j)=(h^2 + u(i+1,j) + u(i-1,j) + u(i,j+1)
    + u(i,j-1))/4;
end
end
err=norm((u_aprox-u),inf);
iter=iter+1;
u=u_aprox;
end
% Mostramos graficamente la solucion.
figure;
imagesc(u);
colorbar()
figure;
surf(u)
colorbar()
shading flat
end
```

A continuación mostraremos un ejemplo de este método.

Ejemplo 3.3.1 *Sea un cuadrado con mallado de tamaño* $\mathrm{n} = 20$. *Tomando una tolerancia* $\mathrm{tol} = 10^{-5}$ *y un número máximo de iteraciones* $\mathrm{itermax} = 100$, *el resultado se muestra en las Figuras 3.2a y 3.2b.*



(a) Representación bidimensional de la solución. (b) Representación tridimensional de la solución.

Figura 3.2: Representación gráfica de la solución del Ejemplo 3.3.1, tanto mediante un mapa de calor en el mallado como a través de la gráfica tridimensional.

3.4. Código para la silueta

Ahora bien, tomando como dominio una silueta de tamaño $n \times m$, vamos a resolver el problema (3.1) en el interior de dicha silueta, utilizando la condición de contorno Dirichlet en la frontera.

Recordemos que la silueta a tratar queremos que sea un dominio acotado del plano que encierre un área en color blanco sobre un fondo negro. Normalmente suele ser al revés, como podemos observar en 3.1. Es decir, la imagen de color negro y el fondo blanco, y en consecuencia, nuestro código deberá permutar estos dos colores.

Como ya se comentó en el Capítulo 2, el color negro se representa mediante el valor 0 y el color blanco mediante el valor 255. Por ello, nos interesa que el código intercambie ambos colores de manera que el color negro predomine sobre el blanco y podamos trabajar con matrices sparse, es decir, matrices con una cantidad notable de ceros, ya que cuantos más ceros haya en la matriz, más sencillas resultaran las operaciones necesarias. Además, de este modo, ya está implícita la condición de frontera nula en la silueta.

Como hemos comentado anteriormente, se va a mejorar el código anterior utilizando

el método SOR para implementar este nuevo algoritmo, ahora sí, proporcionándole la silueta a resolver y añadiendo los cambios pertinentes. De nuevo, basándonos en (3.8) y particularizando en nuestro problema se tiene:

$$U_{i,j}^{(k+1)} = w \cdot \frac{h_x \cdot h_y + U_{i+1,j}^{(k)} + U_{i-1,j}^{(k+1)} + U_{i,j+1}^{(k)} + U_{i,j-1}^{(k+1)}}{4} + (1-w) \cdot U_{i,j}^{(k)}, \qquad (3.10)$$

siendo w el parámetro de relajación (SOR), h_x y h_y el tamaño del mallado en los lados x e y respectivamente.

A continuación se muestra el método resultante:

```
function [u,iter,err]=silueta_SOR(tol,itermax)
% Codigo que aproxima la solucion de la ecuacion de Poisson
% en una silueta utilizando el metodo SOR.
% argumentos de entrada:
% tol es la tolerancia.
% itermax es el numero de iteraciones permitidas.
% argumentos de salida:
% u es la imagen resultante
% iter es el numero de iteraciones que se han producido.
% err indica el error entre la imagen exacta y la obtenida.
u=imread('silueta.png');
u=double(u);
[n,m]=size(u);
hx=1/(n+1);
hy=1/(m+1);
w=1.5; % Parametro de relajacion(SOR).
% Queremos una silueta en blanco sobre un fondo negro.
% Por lo tanto, intercambiamos el blanco y el negro. Asi tambien
% aseguramos la condicion de contorno nula en la frontera.
for i=1:n
    for j=1:m
        u(i,j)=abs(u(i,j)-255);
    end
end
```

```
% Adaptamos el algoritmo realizado anteriormente.
err=1;
iter=0;
u_aprox=u;
while((err>tol)&&(iter<=itermax))
for i=2:n-1
for j=2:m-1
    if (u(i,j)==0)
    % El fondo negro se deja igual.
    % Trabajamos solo sobre el area blanca.
    else
    u_aprox(i,j)= w*(hx*hy+u(i+1,j)+u_aprox(i-1,j)+u(i,j+1)
                  +u_aprox(i,j-1))/4 + (1-w)*u(i,j);
    end
end
end
err=norm(u_aprox-u,inf);
iter=iter+1;
u=u_aprox;
end
% Mostramos la silueta resultante y la solucion graficamente.
figure;
imagesc(u);
colorbar()
figure;
surf(u)
colorbar()
shading flat
end
```

Ahora mostramos algunos ejemplos de este método aplicados a diferentes siluetas.

Ejemplo 3.4.1 *Sea la imagen* 'silueta.png' *la que se muestra en la Figura 3.3. Tomando una tolerancia* tol $= 10^{-5}$ *y un número máximo de iteraciones* itermax $= 100$*, el resultado se muestra en las Figuras 3.4a y 3.4b.*

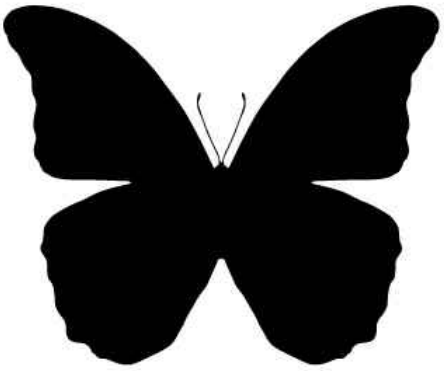
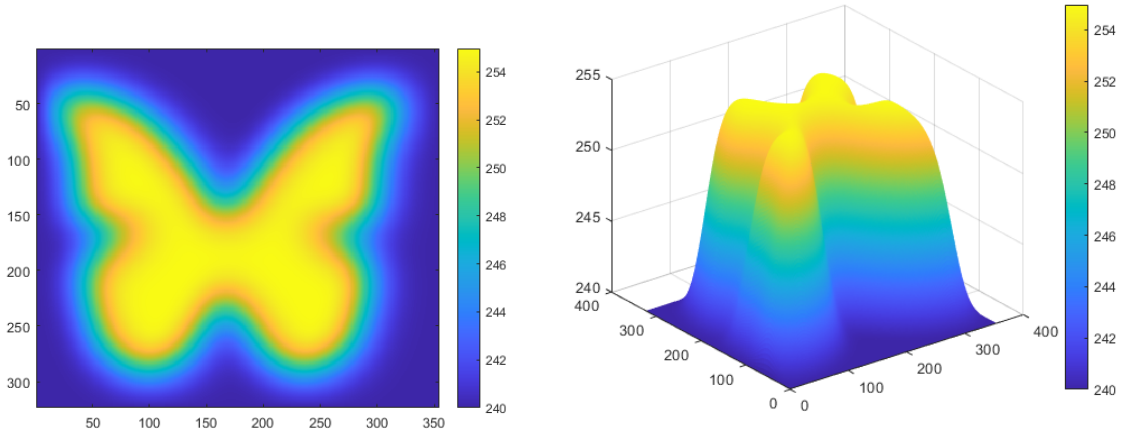


Figura 3.3: Silueta de una mariposa.

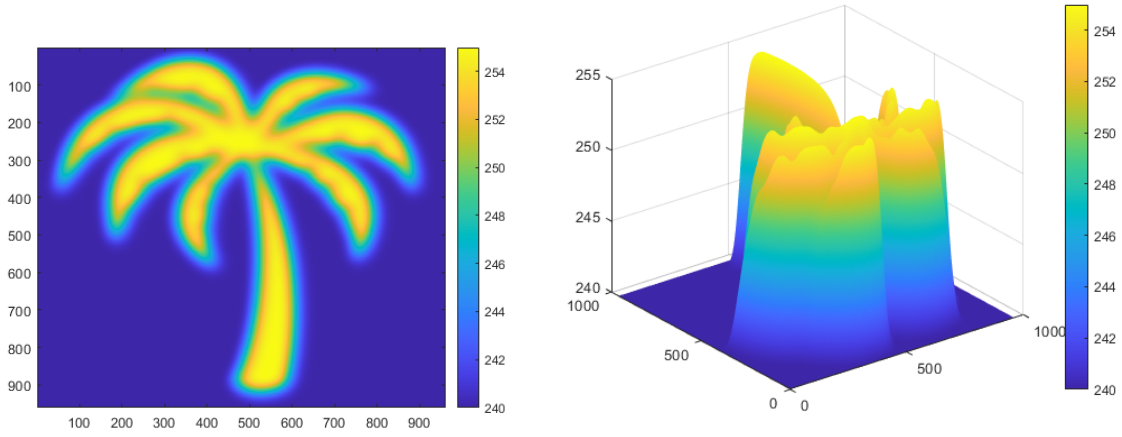


(a) Representación bidimensional de la solución. (b) Representación tridimensional de la solución.

Figura 3.4: Representación gráfica de la solución del Ejemplo 3.4.1, tanto mediante un mapa de calor en el mallado como a través de la gráfica tridimensional.

Ejemplo 3.4.2 *Sea la imagen* 'silueta.png' *la que se muestra en la Figura 3.5. Tomando una tolerancia* tol $= 10^{-5}$ *y un número máximo de iteraciones* itermax $= 100$, *el resultado se muestra en las Figuras 3.6a y 3.6b.*

Figura 3.5: Silueta de una palmera.



(a) Representación bidimensional de la solución. (b) Representación tridimensional de la solución.

Figura 3.6: Representación gráfica de la solución del Ejemplo 3.4.2, tanto mediante un mapa de calor en el mallado como a través de la gráfica tridimensional.

3.5. Recopilación de información sobre las siluetas

Hasta aquí hemos visto cómo el ordenador lee una imagen y traduce las propiedades de la silueta a un determinado problema matemático. En esta sección vamos a ver cómo puede un ordenador extraer información de dicho problema matemático. En particular, nos vamos a centrar en el caso de las concavidades y las convexidades. Para ello, introducimos la siguiente función:

$$\phi(x, y) = U(x, y) + |\nabla U(x, y)|^2, \tag{3.11}$$

donde $U(x, y)$ es la solución calculada en el código anterior.

Los valores más altos de $\phi(x, y)$ se obtienen cerca de las concavidades. Esto es debido a que u crece más rápido con la distancia de las concavidades y, por lo tanto, la magnitud del gradiente es mayor.

Además, podemos describir otra función más completa, que detecta tanto concavidades como convexidades. Esta es la siguiente:

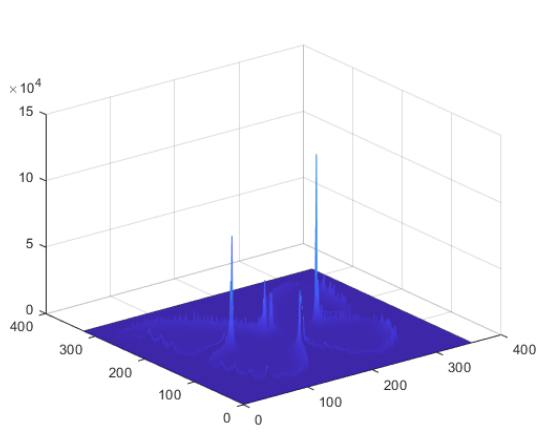
$$\psi(x, y) = -\nabla \cdot \left(\frac{\nabla U(x, y)}{|\nabla U(x, y)|} \right). \tag{3.12}$$

Los valores negativos de ψ indican concavidades. Un valor negativo alto significa una concavidad más aguda. De la misma manera, los valores positivos altos indican convexidades.
A continuación, presentamos un método que calcula las partes cóncavas y convexas de una silueta utilizando las funciones (3.11) y (3.12).

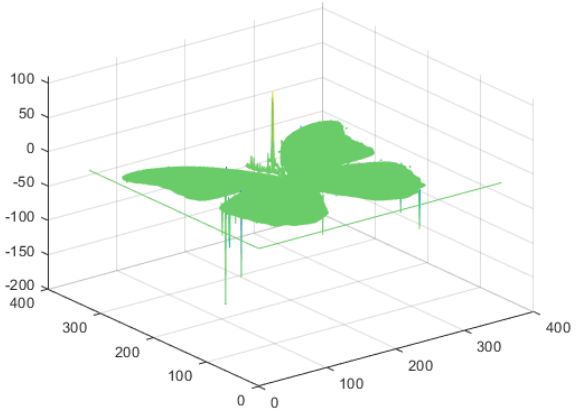
```
function [phi,psi]=concavidades(u)
% u es la solucion del metodo silueta_SOR.
[n,m]=size(u);
for i=2:n
    for j=2:m
        %Primero calculamos el gradiente y su norma.
        gradx(i,j)=u(i,j)-u(i-1,j);
        grady(i,j)=u(i,j)-u(i,j-1);
        normagrad(i,j)=(gradx(i,j)^2+grady(i,j)^2)^0.5;
        %Calculamos phi.
        phi(i,j)=u(i,j)+normagrad(i,j)^2;
        %Calculamos psi.
        psi(i,j)=-((gradx(i,j)-gradx(i-1,j))*normagrad(i,j)
        -gradx(i,j)*(normagrad(i,j)-normagrad(i-1,j))
        +(grady(i,j)-grady(i,j-1))*normagrad(i,j)
        -grady(i,j)*(normagrad(i,j)-normagrad(i,j-1)))
        /normagrad(i,j)^2;
    end
end
% Mostramos las soluciones graficamente.
figure;
```

```
mesh(phi)
figure;
mesh(psi)
end
```

Ejemplo 3.5.1 *Se van a utilizar los ejemplos 3.4.1 y 3.4.2 para ilustrar las diferencias entre las concavidades y convexidades de cada silueta.*

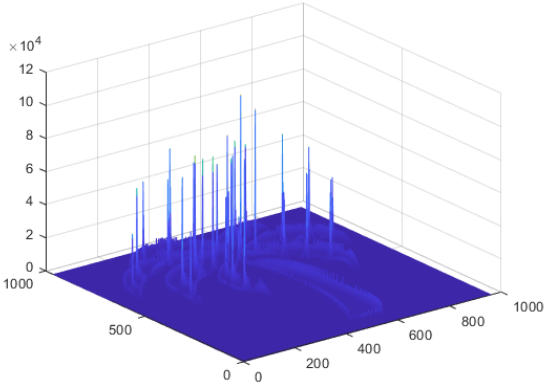


(a) Representación de $\phi(x, y)$.

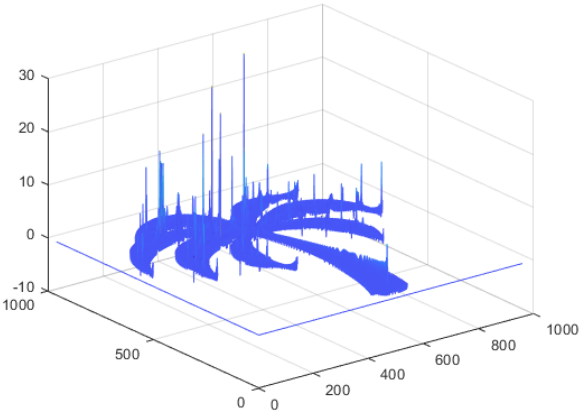


(b) Representación de $\psi(x, y)$.

Figura 3.7: Representación gráfica de las funciones $\phi(x, y)$ y $\psi(x, y)$ del Ejemplo 3.4.1.



(a) Representación de $\phi(x, y)$.



(b) Representación de $\psi(x, y)$.

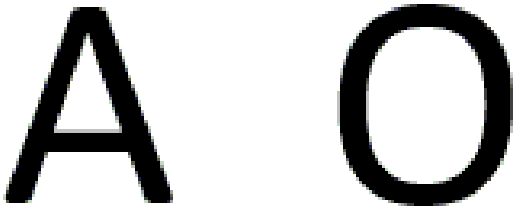
Figura 3.8: Representación gráfica de las funciones $\phi(x, y)$ y $\psi(x, y)$ del Ejemplo 3.4.2.

Si nos fijamos en las siluetas estudiadas de la mariposa y la palmera, recogidas en 3.3

y 3.5 respectivamente, por una parte tenemos una silueta sencilla de una mariposa, que se divide simplemente en cuatro partes que corresponden a las alas. Por otro lado, se observa que la silueta de la palmera es más compleja en ese sentido. Se puede dividir en el tronco y en cada una de sus hojas, donde además las curvas son bastante más agudas. Esto es lo que se demuestra matemáticamente con este ejemplo, donde las líneas verticales nos indican gráficamente que la palmera cuenta con muchas más concavidades y convexidades que la mariposa. Además esta última está dibujada con un contorno muy suave, por lo que apenas se ven líneas verticales quitando las que hay justo en las bifurcaciones de las alas.

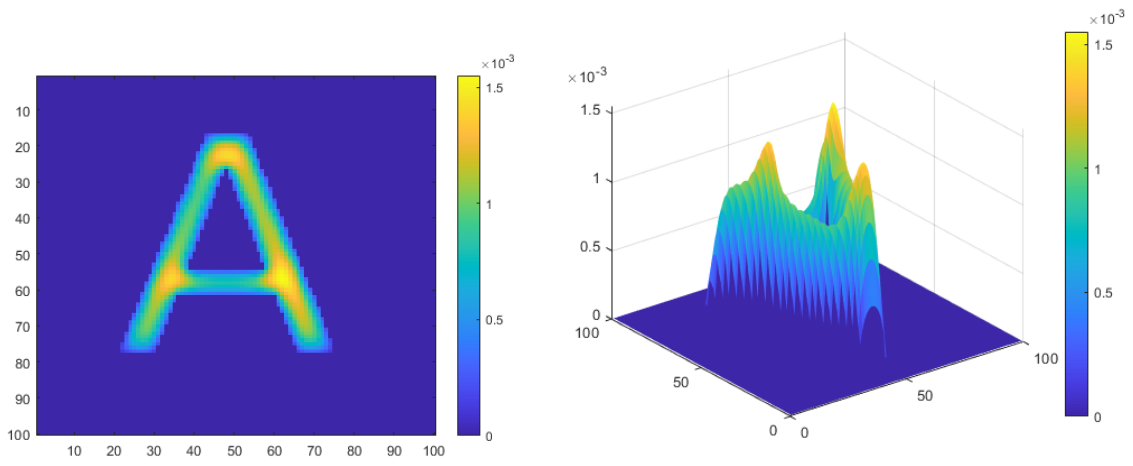
Para finalizar, notemos que estos métodos también son muy útiles para describir signos, números o letras. Por ejemplo, nosotros al leer un texto sabemos distinguir automáticamente una letra de otra, pero si queremos que un ordenador lea ese mismo texto, ¿cómo sabe que letra esta leyendo? Para ilustrar esto, se ha realizado un último ejemplo combinando los dos últimos algoritmos presentados anteriormente.

Ejemplo 3.5.2 *Sea la imagen* 'letraA.png' *la que se muestra en la Figura 3.9a y sea la imagen* 'letraO.png' *la que se muestra en la Figura 3.9b. Tomando en ambas una tolerancia* $\text{tol} = 10^{-12}$ *y un número máximo de iteraciones* $\text{itermax} = 1000$*, los resultados se muestran en las Figuras 3.10a, 3.10b, 3.12a, 3.12b y 3.11a, 3.11b, 3.13a, 3.13b respectivamente.*



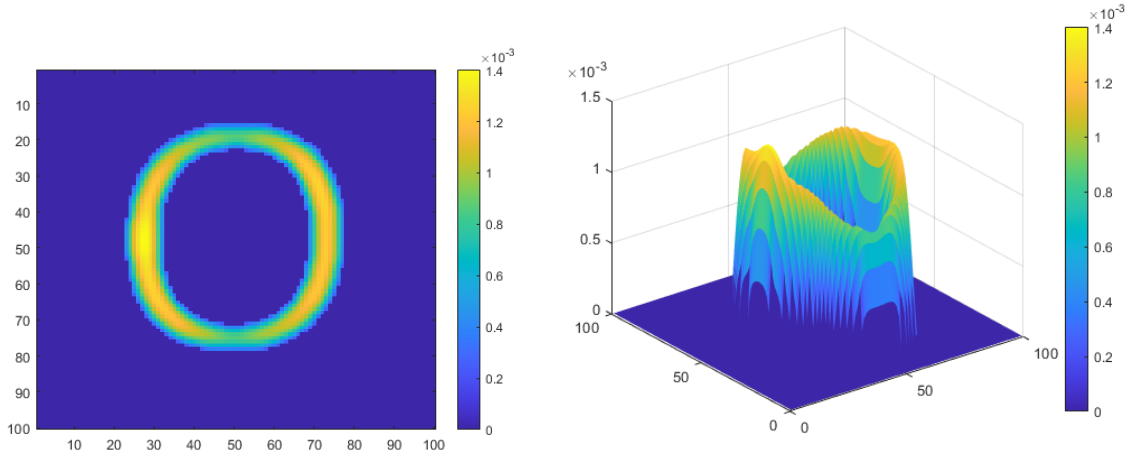
(a) Silueta de la letra A. (b) Silueta de la letra O.

Figura 3.9



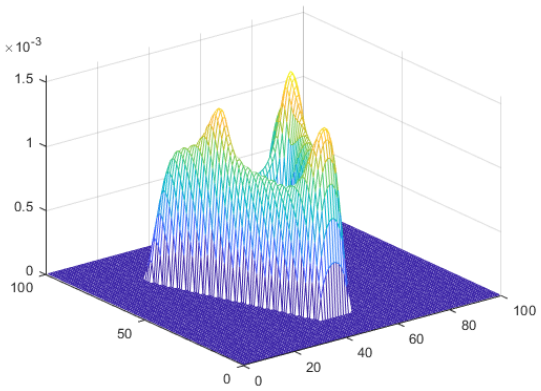
(a) Representación bidimensional de la solución. (b) Representación tridimensional de la solución.

Figura 3.10: Representación gráfica de la solución del Ejemplo 3.5.2 para la Figura 3.9a, tanto mediante un mapa de calor en el mallado como a través de la gráfica tridimensional.

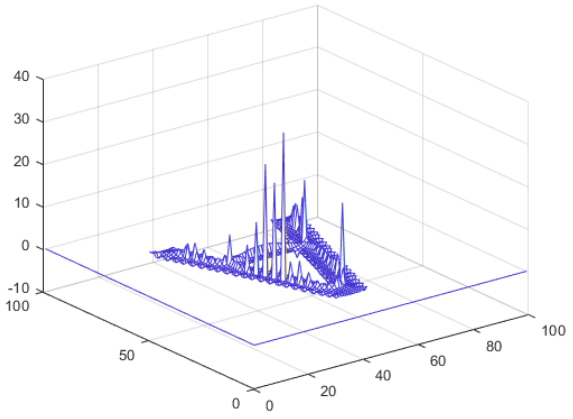


(a) Representación bidimensional de la solución. (b) Representación tridimensional de la solución.

Figura 3.11: Representación gráfica de la solución del Ejemplo 3.5.2 para la figura 3.9b, tanto mediante un mapa de calor en el mallado como a través de la gráfica tridimensional.

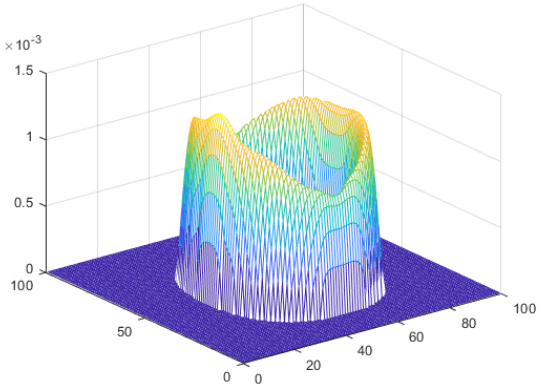


(a) Representación de $\phi(x, y)$.

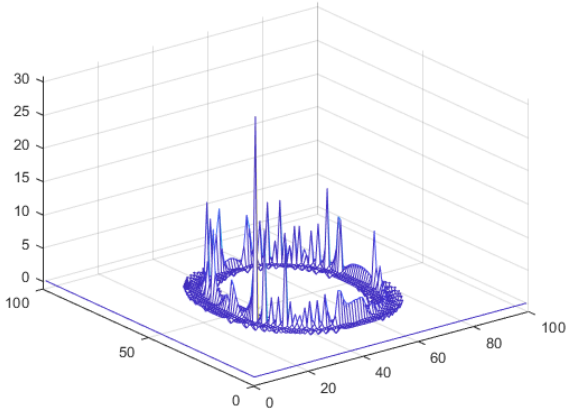


(b) Representación de $\psi(x, y)$.

Figura 3.12: Representación gráfica de las funciones $\phi(x, y)$ y $\psi(x, y)$ del Ejemplo 3.5.2 para la Figura 3.9a.



(a) Representación de $\phi(x, y)$.



(b) Representación de $\psi(x, y)$.

Figura 3.13: Representación gráfica de las funciones $\phi(x, y)$ y $\psi(x, y)$ del Ejemplo 3.5.2 para la Figura 3.9b.

Podemos observar la diferencia entre ambas letras por su curvatura. La **A** está formada por líneas rectas que se intersecan formando esquinas, donde destacan mayoritariamente los picos, mientras que la **O** presenta curvatura en todos sus puntos, y por tanto presenta picos en toda la zona.

Bibliografía

[1] H. Brezis. *Functional Analysis, Sobolev Spaces and Partial Differential Equations*, editorial Springer, 2011.

[2] L. Gorelick. et al. *Shape Representation and Classification Using the Poisson Equation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 28, Nº. 12, 2006.

[3] R. Granero, *Ampliación de análisis*, Universidad de Cantabria, 2021.

[4] M. C. Pola, *Algunos temas de Cálculo Numérico*, Universidad de Cantabria, 2021.

[5] Y. A. Qahraman. *Basic Iterative Methods for Solving Elliptic Partial Differential Equation*, Eastern Mediterranean University, 2014.