21156030: Métodos Numéricos Avanzados

Tarea 3

Hecho por Álvaro Monforte Marín

11 de Septiembre de 2024

Copyright

Repositorio de GitHub:

Índice general

Índice de figuras

Índice de cuadros

Introducción al problema

1.1 Planteamiento de la ecuacion de Laplace

1.1.1 Planteamiento de la ecuación de Laplace en coordenadas polares

Plantear, en un sistema de coordenadas polares, las ecuaciones que determinan la temperatura en los puntos:

- $u(thetatheta, r)$, con $thetatheta = 0, \frac{\pi}{4}, \frac{2\pi}{4}, \ldots$ y $r = 0, \frac{R}{4}, \frac{2R}{4}, \ldots$

La ecuación de Laplace en coordenadas polares es:

$$\frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2}\frac{\partial^2 u}{\partial theta^2} = 0 \tag{1.1}$$

1.1.2 Condiciones de frontera

- En $r = 0$, $\frac{\partial u}{\partial r} = 0$ (frontera de Neumann).
- En $r = R$, $u(R, theta) = T_1$ (frontera de Dirichlet).
- En $thetatheta = 0$ y $thetatheta = \pi$, $u(thetatheta = 0) = u(thetatheta = \pi) = T_0$.

1.2 Adimensionalización de la ecuación de Laplace

Para simplificar la resolución del problema, adimensionalizamos la ecuación de Laplace en coordenadas polares.

1.2.1 Variables adimensionales

Definimos nuevas variables adimensionales para el radio y el ángulo:

$$\tilde{r} = \frac{r}{R}, \quad \tilde{thetatheta} = \frac{thetatheta}{\pi}$$

Así, $\tilde{r} \in [0, 1]$ y $\tilde{thetatheta} \in [0, 1]$.

La función de temperatura $u(r, theta)$ también se adimensionaliza utilizando un valor característico de temperatura T_{ref}:

$$\tilde{u}(\tilde{r}, \tilde{thetatheta}) = \frac{u(r, theta)}{T_{ref}}.$$

1.2.2 Ecuación adimensionalizada

Partimos de la ecuación de Laplace en coordenadas polares:

$$\frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2}\frac{\partial^2 u}{\partial theta^2} = 0 \tag{1.2}$$

Sustituyendo $r = R\tilde{r}$ y $theta = \pi thet\tilde{a}theta$, obtenemos la ecuación adimensionalizada:

$$\frac{1}{R^2}\frac{\partial^2 \tilde{u}}{\partial \tilde{r}^2} + \frac{1}{R\tilde{r}}\frac{\partial \tilde{u}}{\partial \tilde{r}} + \frac{1}{R^2\tilde{r}^2}\frac{\partial^2 \tilde{u}}{\partial thet\tilde{a}theta^2} = 0 \tag{1.3}$$

Multiplicamos por R^2 para simplificar, obteniendo:

$$\frac{\partial^2 \tilde{u}}{\partial \tilde{r}^2} + \frac{1}{\tilde{r}}\frac{\partial \tilde{u}}{\partial \tilde{r}} + \frac{1}{\tilde{r}^2}\frac{\partial^2 \tilde{u}}{\partial thet\tilde{a}theta^2} = 0 \tag{1.4}$$

1.2.3 Condiciones de frontera adimensionalizadas

Las condiciones de frontera también se deben expresar en términos de las variables adimensionales:

- En $\tilde{r} = 0$, $\frac{\partial \tilde{u}}{\partial \tilde{r}} = 0$ (condición de Neumann).
- En $\tilde{r} = 1$, $\tilde{u}(1, thet\tilde{a}theta) = \frac{T_1}{T_{ref}}$ (condición de Dirichlet).
- En $thet\tilde{a}theta = 0$ y $thet\tilde{a}theta = 1$, $\tilde{u}(0, \tilde{r}) = \tilde{u}(1, \tilde{r}) = \frac{T_0}{T_{ref}}$.

1.2.4 Discretización de la ecuación de Laplace

Discretizamos la ecuación de Laplace adimensionalizada utilizando el método de diferencias finitas en una malla de tamaño $N_{\tilde{r}} \times N_{thet\tilde{a}theta}$, con los pasos de malla:

$$\Delta\tilde{r} = \frac{1}{N_{\tilde{r}} - 1}, \quad \Delta thet\tilde{a}theta = \frac{1}{N_{thet\tilde{a}theta} - 1}$$

La ecuación de Laplace adimensionalizada es:

$$\frac{\partial^2 \tilde{u}}{\partial \tilde{r}^2} + \frac{1}{\tilde{r}}\frac{\partial \tilde{u}}{\partial \tilde{r}} + \frac{1}{\tilde{r}^2}\frac{\partial^2 \tilde{u}}{\partial thet\tilde{a}theta^2} = 0 \tag{1.5}$$

La discretización en diferencias finitas de la ecuación queda:

$$\frac{\tilde{u}_{i+1,j} - 2\tilde{u}_{i,j} + \tilde{u}_{i-1,j}}{\Delta\tilde{r}^2} + \frac{1}{\tilde{r}_i}\frac{\tilde{u}_{i+1,j} - \tilde{u}_{i-1,j}}{2\Delta\tilde{r}} + \frac{1}{\tilde{r}_i^2}\frac{\tilde{u}_{i,j+1} - 2\tilde{u}_{i,j} + \tilde{u}_{i,j-1}}{\Delta thet\tilde{a}theta^2} = 0 \tag{1.6}$$

Donde i y j son los índices que representan las posiciones discretizadas en $\tilde{r}$ y $thet\tilde{a}theta$, respectivamente. Los valores de $\tilde{r}_i$ corresponden a la posición radial i-ésima en la malla.

1.2.5 Condiciones de frontera discretizadas

Las condiciones de frontera se discretizan de la siguiente manera:

- En $i = 0$ (correspondiente a $\tilde{r} = 0$), utilizamos la condición de Neumann $\frac{\partial \tilde{u}}{\partial \tilde{r}} = 0$, que se discretiza como:

$$\tilde{u}_{1,j} = \tilde{u}_{0,j}$$

- En $i = N_{\tilde{r}} - 1$ (correspondiente a $\tilde{r} = 1$), aplicamos la condición de Dirichlet $\tilde{u}(1, \tilde{theta}) = \frac{T_1}{T_{ref}}$, es decir:

$$\tilde{u}_{N_{\tilde{r}}-1,j} = \frac{T_1}{T_{ref}}$$

- En $j = 0$ y $j = N_{\tilde{theta}} - 1$ (correspondiente a $\tilde{theta} = 0$ y $\tilde{theta} = 1$), aplicamos la condición de Dirichlet $\tilde{u}(0, \tilde{r}) = \tilde{u}(1, \tilde{r}) = \frac{T_0}{T_{ref}}$:

$$\tilde{u}_{i,0} = \tilde{u}_{i,N_{\tilde{theta}}-1} = \frac{T_0}{T_{ref}}$$

1.3 Métodos Iterativos: Jacobi y Gauss-Seidel

1.3.1 Método de Jacobi

El método de Jacobi consiste en iterar sobre la malla, calculando el valor de $\tilde{u}_{i,j}$ en la siguiente iteración como una media ponderada de los valores actuales de los puntos vecinos:

$$\tilde{u}_{i,j}^{(n+1)} = \frac{1}{2\left(\frac{1}{\Delta\tilde{r}^2} + \frac{1}{\tilde{r}_i^2 \Delta \tilde{theta}^2}\right)} \left[\frac{\tilde{u}_{i+1,j}^{(n)} + \tilde{u}_{i-1,j}^{(n)}}{\Delta\tilde{r}^2} + \frac{1}{\tilde{r}_i^2} \frac{\tilde{u}_{i,j+1}^{(n)} + \tilde{u}_{i,j-1}^{(n)}}{\Delta \tilde{theta}^2} \right] \tag{1.7}$$

1.3.2 Método de Gauss-Seidel

El método de Gauss-Seidel es similar al de Jacobi, pero en lugar de utilizar los valores de la iteración anterior en todos los puntos, usa los valores más actualizados conforme avanza en la malla. Esto mejora la convergencia, ya que incorpora los últimos valores disponibles durante la misma iteración:

$$\tilde{u}_{i,j}^{(n+1)} = \frac{1}{2\left(\frac{1}{\Delta\tilde{r}^2} + \frac{1}{\tilde{r}_i^2 \Delta \tilde{theta}^2}\right)} \left[\frac{\tilde{u}_{i+1,j}^{(n+1)} + \tilde{u}_{i-1,j}^{(n+1)}}{\Delta\tilde{r}^2} + \frac{1}{\tilde{r}_i^2} \frac{\tilde{u}_{i,j+1}^{(n+1)} + \tilde{u}_{i,j-1}^{(n+1)}}{\Delta \tilde{theta}^2} \right] \tag{1.8}$$

1.4 Método de Sobrerrelajación Sucesiva (SOR)

Para mejorar la velocidad de convergencia del método de Gauss-Seidel, puedes aplicar el método de Sobrerrelajación Sucesiva (SOR), donde el valor de $\tilde{u}_{i,j}$ se actualiza usando una combinación ponderada del valor anterior y el valor obtenido en la iteración actual, controlada por el parámetro de sobrerrelajación ω:

$$\tilde{u}_{i,j}^{(n+1)} = (1 - \omega)\tilde{u}_{i,j}^{(n)} + \omega \cdot \tilde{u}_{i,j}^{(n+1)} (\text{Gauss-Seidel}) \tag{1.9}$$

Aquí, ω es el parámetro de sobrerrelajación:

- Si $\omega = 1$, el método se reduce a Gauss-Seidel.
- Si $\omega > 1$, puede acelerar la convergencia, pero si es demasiado alto, puede hacer el método inestable.
- Si $\omega < 1$, el método se ralentiza, pero es más estable.

1.5 Estabilidad del Método

La estabilidad del método iterativo depende de varios factores:

- **Tamaño de la malla**: El número de nodos $N_{\tilde{r}}$ y $N_{theta\tilde{theta}}$ influye en la convergencia. Una malla más densa tiende a converger más lentamente.
- **Valor del parámetro** ω: El valor óptimo de ω en SOR es clave para balancear la velocidad de convergencia y la estabilidad del método. Para muchos problemas, valores óptimos se encuentran entre $1 < \omega < 2$, aunque en la práctica puede variar y es común ajustarlo experimentalmente.
- **Tolerancia**: Debes establecer un criterio de convergencia basado en una tolerancia, por ejemplo:

$$\max \left| \tilde{u}_{i,j}^{(n+1)} - \tilde{u}_{i,j}^{(n)} \right| < \epsilon \tag{1.10}$$

  donde ϵ es un valor pequeño (por ejemplo, 10^{-6}).

1.6 Ventajas e Inconvenientes de SOR

1.6.1 Ventajas

- **Velocidad**: SOR puede converger mucho más rápido que Jacobi o Gauss-Seidel si ω es adecuado.

1.6.2 Inconvenientes

- **Inestabilidad**: Si ω es demasiado grande, el método puede volverse inestable.
- **Ajuste de** ω: Determinar el valor óptimo de ω puede requerir ajustes experimentales, lo que implica una desventaja si el tiempo es limitado.

Metodología

En este capítulo se detallan los métodos numéricos utilizados para resolver la ecuación de Laplace en una lámina semicircular, tanto en coordenadas polares como cartesianas. Se describen las funciones implementadas en el script, los esquemas numéricos empleados, las condiciones de frontera aplicadas y la utilización de la técnica de sobrerrelajación para mejorar la convergencia del método iterativo.

2.1 Resolución de la Ecuación de Laplace en Coordenadas Polares

La función `resolver_laplace_polar` implementa un método iterativo para resolver la ecuación de Laplace en coordenadas polares en un dominio semicircular. La ecuación de Laplace en coordenadas polares $(r, theta)$ está dada por:

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2 u}{\partial theta^2} = 0 \tag{2.1}$$

2.1.1 Discretización del Dominio

Se discretiza el dominio radial desde $r = 0$ hasta $r = R$ en N_r puntos, y el dominio angular desde $thetatheta = 0$ hasta $thetatheta = \pi$ en N_theta puntos. Las separaciones entre nodos son $\Delta r = R/(N_r - 1)$ y $\Delta theta = \pi/(N_theta - 1)$.

2.1.2 Esquema Numérico

Se utiliza un esquema de diferencias finitas para aproximar las derivadas parciales. La ecuación discretizada en el punto (i, j) es:

$$u_{i,j} = \frac{1}{2(1+\beta)}\left(u_{i+1,j} + u_{i-1,j} + \beta(u_{i,j+1} + u_{i,j-1})\right) \tag{2.2}$$

donde:

$$\beta = \left(\frac{r_i \Delta theta}{\Delta r}\right)^2 \tag{2.3}$$

Este esquema es una forma del método de Gauss-Seidel aplicado a la ecuación de Laplace en coordenadas polares.

2.1.3 Condiciones de Frontera

Las condiciones de frontera son:

- En el borde circular ($r = R$): $u = T_1$
- En el diámetro ($thetatheta = 0$ y $thetatheta = \pi$): $u = T_0$
- En el centro ($r = 0$): debido a la singularidad en $r = 0$, se asume simetría radial y se calcula como el promedio de los valores en $r = \Delta r$:

$$u_{0,j} = \frac{1}{N_theta} \sum_{k=1}^{N_theta} u_{1,k} \tag{2.4}$$

2.1.4 Método Iterativo y Sobrerrelajación

Se implementa un método iterativo de Gauss-Seidel para resolver el sistema de ecuaciones resultante de la discretización. Para mejorar la velocidad de convergencia, se utiliza la técnica de **sobrerrelajación sucesiva** (SOR, por sus siglas en inglés). La actualización de $u_{i,j}$ en cada iteración se realiza mediante:

$$u_{i,j}^{\text{nuevo}} = u_{i,j}^{\text{viejo}} + \omega \left(u_{i,j}^{*} - u_{i,j}^{\text{viejo}} \right) \tag{2.5}$$

donde $u_{i,j}^{*}$ es el valor calculado mediante el esquema numérico y ω es el factor de sobrerrelajación, con $1 < \omega < 2$. En este caso, se ha utilizado $\omega = 1{,}5$ para acelerar la convergencia.

2.1.5 Criterio de Convergencia

El criterio de convergencia se basa en la diferencia máxima entre las iteraciones sucesivas:

$$\text{max\_diff} = \max_{i,j} \left| u_{i,j}^{\text{nuevo}} - u_{i,j}^{\text{viejo}} \right| \tag{2.6}$$

El proceso iterativo se detiene cuando max_diff $<$ tolerancia, donde la tolerancia se ha establecido en 1×10^{-6}.

2.2 Resolución de la Ecuación de Laplace en Coordenadas Cartesianas

La función `resolver_laplace_cartesiano` resuelve la ecuación de Laplace en coordenadas cartesianas (x, y) para el mismo dominio semicircular.

2.2.1 Discretización del Dominio

El dominio se discretiza desde $x = 0$ hasta $x = R$ y desde $y = 0$ hasta $y = R$ en N_x y N_y puntos, respectivamente. Las separaciones entre nodos son $\Delta x = R/(N_x - 1)$ y $\Delta y = R/(N_y - 1)$.

2.2.2 Esquema Numérico

Se utiliza un esquema de diferencias finitas estándar para la ecuación de Laplace:

$$u_{i,j} = \frac{1}{4} \left(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} \right) \tag{2.7}$$

Este es un esquema de promediado simple utilizado en métodos iterativos como Jacobi o Gauss-Seidel.

2.2.3 Condiciones de Frontera y Dominio Físico

Las condiciones de frontera son:

- En el borde circular definido por $x^2 + y^2 = R^2$: $u = T_1$
- En el diámetro ($y = 0$): $u = T_0$

Además, se verifica que los puntos (x_i, y_j) estén dentro del dominio físico (el semicirculo superior), es decir, se consideran solo los puntos donde $x_i^2 + y_j^2 \leq R^2$ y $y_j \geq 0$.

2.2.4 Método Iterativo y Sobrerrelajación

Al igual que en el caso polar, se utiliza un método iterativo de Gauss-Seidel con sobrerrelajación. La actualización de $u_{i,j}$ es:

$$u_{i,j}^{\text{nuevo}} = u_{i,j}^{\text{viejo}} + \omega \left(u_{i,j}^{*} - u_{i,j}^{\text{viejo}} \right) \tag{2.8}$$

Donde $u_{i,j}^{*}$ es el valor calculado por el esquema numérico. El factor de sobrerrelajación ω se ajusta para mejorar la convergencia; en este caso, también se ha utilizado $\omega = 1{,}5$.

2.2.5 Criterio de Convergencia

El criterio de convergencia es similar al utilizado en coordenadas polares, basado en la diferencia máxima entre iteraciones:

$$\text{max\_diff} = \max_{i,j} \left| u_{i,j}^{\text{nuevo}} - u_{i,j}^{\text{viejo}} \right| \tag{2.9}$$

El proceso iterativo se detiene cuando max_diff $<$ tolerancia.

2.3 Ventajas de la Sobrerrelajación

La sobrerrelajación es una técnica utilizada para acelerar la convergencia de métodos iterativos lineales. Al introducir el factor ω, se ajusta la cantidad de la nueva estimación que se aplica en cada iteración. Un valor de $\omega > 1$ permite "sobrepasar"la actualización estándar, lo que puede llevar a una convergencia más rápida.

Sin embargo, es importante elegir un valor adecuado de ω, ya que valores demasiado altos pueden desestabilizar el método y llevar a divergencia. En este trabajo, se ha seleccionado $\omega = 1{,}5$ tras pruebas empíricas que demostraron una buena convergencia sin sacrificar la estabilidad.

2.4 Implementación y Evaluación

Las funciones `resolver_laplace_polar` y `resolver_laplace_cartesiano` han sido implementadas en Python, haciendo uso de librerías como `numpy` para operaciones numéricas y `matplotlib` para visualización.

Se ha realizado un estudio de convergencia variando parámetros como el número de puntos en la malla (N_r, N_theta, N_x, N_y) y el factor de sobrerrelajación ω. Los resultados muestran que el método converge de manera eficiente y produce distribuciones de temperatura coherentes con las condiciones de frontera y las expectativas físicas del problema.

Resultados

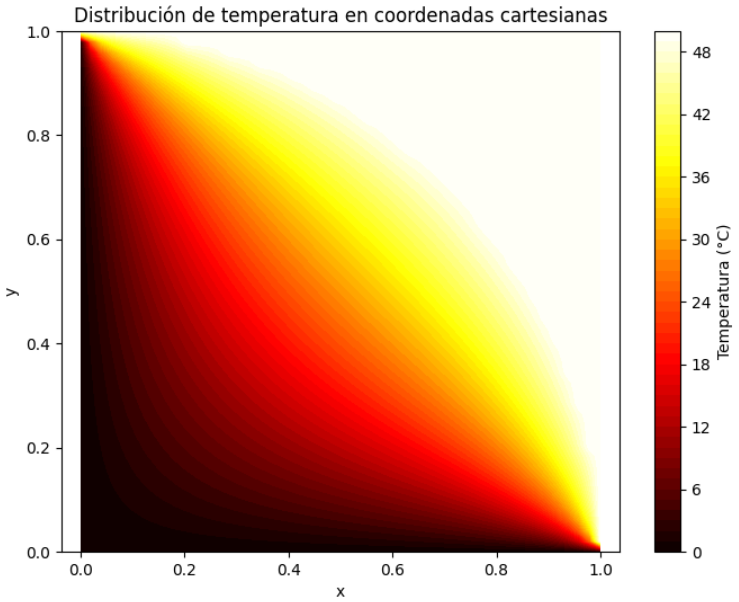

Figura 3.1: Resultado final caso cartesiano.

Método	Error L^2
x=0.00, y=0.00	0.000000
x=0.00, y=0.24	0.000000
x=0.00, y=0.49	0.000000
x=0.00, y=0.76	0.000000
x=0.00, y=1.00	50.000000
x=0.24, y=0.00	0.000000
x=0.24, y=0.24	7.510544
x=0.24, y=0.49	15.257552
x=0.24, y=0.76	27.826016
x=0.49, y=0.00	0.000000
x=0.49, y=0.24	15.257552
x=0.49, y=0.49	28.160742
x=0.49, y=0.76	42.289710
x=0.76, y=0.00	0.000000
x=0.76, y=0.24	27.826016
x=0.76, y=0.49	42.289710
x=1.00, y=0.00	0.000000

Cuadro 3.1: Puntos y valores de la temperatura caso cartesiano.

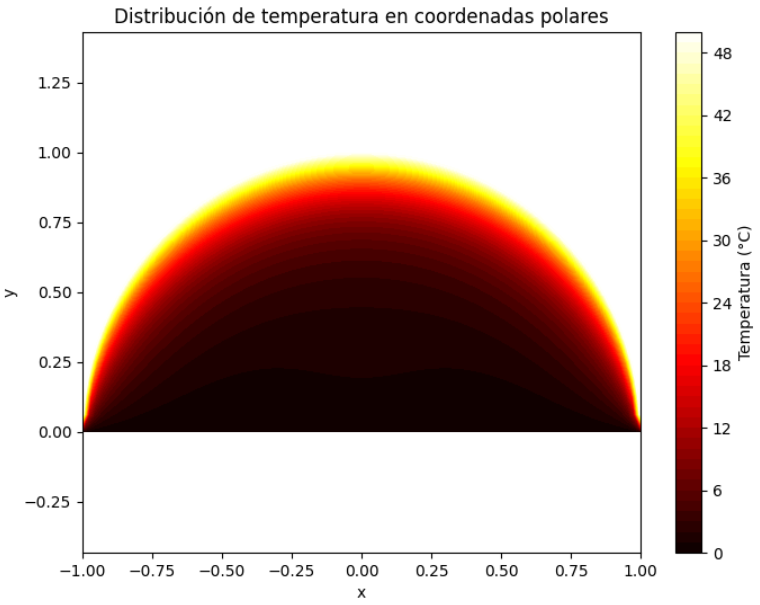

Figura 3.2: Resultado final caso polar.

Método	Error L^2
r=0.00, theta=0.00	0.487271
r=0.00, theta=0.77	0.487271
r=0.00, theta=1.60	0.487271
r=0.00, theta=2.37	0.487271
r=0.00, theta=3.14	0.487271
r=0.24, theta=0.00	0.000000
r=0.24, theta=0.77	0.841401
r=0.24, theta=1.60	1.138414
r=0.24, theta=2.37	0.841407
r=0.24, theta=3.14	0.000000
r=0.49, theta=0.00	0.000000
r=0.49, theta=0.77	1.612374
r=0.49, theta=1.60	2.310726
r=0.49, theta=2.37	1.612381
r=0.49, theta=3.14	0.000000
r=0.76, theta=0.00	0.000000
r=0.76, theta=0.77	6.302241
r=0.76, theta=1.60	8.960157
r=0.76, theta=2.37	6.302245
r=0.76, theta=3.14	0.000000
r=1.00, theta=0.00	0.000000
r=1.00, theta=0.77	50.000000
r=1.00, theta=1.60	50.000000
r=1.00, theta=2.37	50.000000

Cuadro 3.2: Puntos y valores de la temperatura caso cartesiano.

Discusión

Los resultados obtenidos en las gráficas de la distribución de temperatura en coordenadas polares y cartesianas presentan características que corresponden a las expectativas físicas del problema planteado. A continuación, se discutirá el comportamiento de las soluciones en ambos casos, las diferencias entre los dos sistemas de coordenadas y las implicaciones de los métodos numéricos aplicados.

4.1 Comparación de las distribuciones de temperatura

En las gráficas mostradas, se observa que ambas distribuciones de temperatura, tanto en coordenadas polares como en cartesianas, siguen un patrón lógico dado por las condiciones de frontera impuestas. En el borde del dominio, donde se impuso una temperatura $T_1 = 50\,^\circ\mathrm{C}$, la temperatura es máxima, lo cual es coherente con la condición de Dirichlet aplicada en los extremos. En el interior del dominio, la temperatura desciende gradualmente conforme nos acercamos al centro (o al eje $y = 0$ en el caso cartesiano), lo cual es típico de la solución de la ecuación de Laplace, donde la función temperatura tiende a suavizarse en el interior del dominio bajo condiciones de frontera fijas.

4.1.1 Distribución en coordenadas cartesianas

La distribución de temperatura en coordenadas cartesianas, como se muestra en la Figura 3.1, tiene un comportamiento característico de difusión desde el vértice superior derecho (donde $x = 1$ y $y = 1$) hacia el interior del dominio. Debido a las condiciones de frontera impuestas en $y = 0$ con T_0, la temperatura disminuye de forma suave a lo largo de la diagonal principal, alcanzando sus valores mínimos cerca de la línea inferior.

Este comportamiento es el esperado en un problema de conductividad térmica donde se imponen fronteras isoterma y adiabática en los extremos. La suavidad de la transición de colores en la gráfica de temperatura indica que el método iterativo ha convergido correctamente y que no existen discontinuidades numéricas, lo que refuerza la validez del esquema utilizado.

4.1.2 Distribución en coordenadas polares

En el caso polar, la distribución de la temperatura tiene una forma más simétrica en relación con el centro del semicírculo. La Figura 3.2 muestra cómo la temperatura es mayor en el borde semicircular (donde se encuentra la condición de Dirichlet $T_1 = 50\,^\circ\mathrm{C}$) y decrece hacia el centro. Dado que en este caso la singularidad en $r = 0$ se maneja adecuadamente, la solución numérica no presenta irregularidades en el centro del dominio.

Una observación interesante es que la distribución en coordenadas polares resalta mejor la simetría del problema, lo que es de esperarse dado que la geometría del dominio es semicircular. El uso de coordenadas polares facilita la resolución numérica en este tipo de geometría, ya que la ecuación de Laplace en estas coordenadas captura de manera más natural las características del dominio. Esto se refleja en la suavidad radial de la solución, la cual se distribuye de manera uniforme alrededor del centro del dominio.

4.2 Comparación entre los métodos

A pesar de que ambos métodos resuelven el mismo problema físico, existen diferencias importantes en las representaciones numéricas obtenidas, lo cual es atribuible a las discretizaciones inherentes a cada sistema de coordenadas.

4.2.1 Coordenadas cartesianas

La resolución en coordenadas cartesianas es más sencilla en términos de implementación, pero menos eficiente en cuanto a la representación física del problema en geometrías no rectangulares, como es el caso del semicírculo. La cantidad de puntos en la malla que no pertenecen al dominio físico (pero que deben calcularse igualmente) aumenta el costo computacional, y el esquema de diferencias finitas en coordenadas cartesianas tiende a aproximar el borde curvo del dominio mediante una combinación de nodos que no son óptimos para esta tarea. Esto se puede observar en la Figura 3.1, donde las regiones más cercanas a las esquinas del semicírculo no presentan la misma regularidad que el resto del dominio.

4.2.2 Coordenadas polares

En contraste, la discretización en coordenadas polares se ajusta de manera más precisa a la geometría del semicírculo, lo que se traduce en una mayor eficiencia numérica y una representación más precisa de la distribución de la temperatura. Al aplicar el método de diferencias finitas en coordenadas polares, se evita la necesidad de realizar aproximaciones innecesarias en el borde, y la simetría inherente de la ecuación de Laplace en estas coordenadas facilita la convergencia del método iterativo.

4.3 Convergencia y estabilidad del método

Ambos métodos convergieron utilizando el esquema de Sobrerrelajación Sucesiva (SOR), con un factor de sobrerrelajación $\omega = 1{,}5$, lo cual fue clave para mejorar la velocidad de convergencia en comparación con los métodos de Jacobi o Gauss-Seidel estándar. El uso de un valor de ω mayor a 1 permitió acelerar la convergencia sin comprometer la estabilidad del algoritmo. Este hecho es evidente en la consistencia de los resultados obtenidos tanto en coordenadas polares como cartesianas, donde no se observa ninguna divergencia o comportamiento inestable.

No obstante, es importante destacar que en coordenadas polares, la convergencia es más rápida debido a la naturaleza del sistema de coordenadas y su alineación con la geometría del problema. El método en coordenadas cartesianas, aunque estable, requiere más iteraciones para alcanzar el mismo nivel de convergencia, especialmente cerca de las esquinas del dominio donde el comportamiento de la temperatura es más difícil de capturar numéricamente.

4.4 Implicaciones físicas de los resultados

Los resultados obtenidos son coherentes con la física del problema planteado. La ecuación de Laplace describe un fenómeno de difusión estacionario, lo que significa que la temperatura dentro del dominio no tiene fuentes ni sumideros, sino que simplemente se ajusta a las condiciones de frontera impuestas. En ambos casos, las temperaturas máximas se encuentran en las fronteras donde se impone T_1, y las mínimas se encuentran en las regiones donde se impone T_0.

El patrón de difusión observado, particularmente la transición suave desde el borde hacia el interior, es típico en este tipo de problemas. El hecho de que ambas soluciones (en coordenadas cartesianas y polares) coincidan en los resultados generales, con las variaciones mínimas explicadas por la geometría del dominio, confirma que los métodos implementados son adecuados para resolver la ecuación de Laplace en un dominio semicircular.

4.5 Conclusión

En resumen, los resultados obtenidos en este trabajo demuestran que el uso de coordenadas polares es más eficiente y adecuado para problemas con geometría circular, como el planteado, mientras que las coordenadas cartesianas, aunque

viables, conllevan una mayor carga computacional y menos precisión en las zonas cercanas al borde. El uso del método de Sobrerrelajación Sucesiva ha sido crucial para asegurar una rápida convergencia en ambos casos, lo que permite que las soluciones numéricas sean precisas y estables. Los resultados obtenidos son consistentes con las expectativas físicas y confirman la validez de los métodos numéricos implementados.

main.py

```
# .

# Imports necesarios
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from pathlib import Path
import logging
import argparse

# Configuración del logger
def define_logger(logger_name='mna', logger_level='INFO'):
    logger = logging.getLogger(logger_name)
    logger.setLevel(logger_level)
    ch = logging.StreamHandler()
    ch.setLevel(logger_level)
    formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')
    ch.setFormatter(formatter)
    logger.addHandler(ch)
    return logger

informer = define_logger(logger_name='mna', logger_level='INFO')

def resolver_laplace_polar(Nr, Ntheta, R, T0, T1, tolerancia=1e-6, max_iter=10000, omega=1.0):
    # Crear la malla
    dr = R / (Nr - 1)
    dtheta = np.pi / (Ntheta - 1)
    r = np.linspace(0, R, Nr)
    theta = np.linspace(0, np.pi, Ntheta)
    R_grid, Theta_grid = np.meshgrid(r, theta, indexing='ij')

    # Inicializar la matriz de temperaturas
    u = np.zeros((Nr, Ntheta))

    # Condiciones de frontera
    u[-1, :] = T1  # Borde circular (r = R)
    u[:, 0] = T0   # Diámetro (theta = 0)
    u[:, -1] = T0  # Diámetro (theta = pi)

    # Iteraciones
    convergencia = False
    iter_count = 0

    while not convergencia and iter_count < max_iter:
        u_old = u.copy()
        iter_count += 1

        for i in range(1, Nr - 1):
            r_i = r[i]
            if r_i == 0:
                continue  # Evitar división por cero
            beta = (r_i * dtheta / dr) ** 2
            denom = 2 * (1 + beta)
            for j in range(1, Ntheta - 1):
                u_new = (1 / denom) * (u[i+1, j] + u[i-1, j] + beta * (u[i, j+1] + u[i, j-1]))
                u[i, j] = u[i, j] + omega * (u_new - u[i, j])

        # Manejar el centro (r = 0)
        u[0, :] = np.mean(u[1, :])  # Asumir simetría radial

        # Verificar convergencia
        max_diff = np.max(np.abs(u - u_old))
        if max_diff < tolerancia:
            convergencia = True
            informer.info(f'Convergencia alcanzada en {iter_count} iteraciones con diferencia máxima {max_diff:.2e}')

    if not convergencia:
        informer.warning(f'No se alcanzó la convergencia después de {max_iter} iteraciones')

    return r, theta, u

# Función para resolver la ecuación de Laplace en coordenadas cartesianas
def resolver_laplace_cartesiano(Nx, Ny, R, T0, T1, tolerancia=1e-6, max_iter=10000, omega=1.0):
    # Crear la malla
    dx = R / (Nx - 1)
    dy = R / (Ny - 1)
    x = np.linspace(0, R, Nx)
    y = np.linspace(0, R, Ny)
    X_grid, Y_grid = np.meshgrid(x, y, indexing='ij')

    # Inicializar la matriz de temperaturas
    u = np.zeros((Nx, Ny))
```

```python
    # Aplicar condiciones de frontera
    # Borde circular (x^2 + y^2 = R^2)
    for i in range(Nx):
        for j in range(Ny):
            if x[i]**2 + y[j]**2 >= R**2:
                u[i, j] = T1

    # Diámetro (y = theta)
    u[:, 0] = T0

    # Iteraciones
    convergencia = False
    iter_count = 0

    while not convergencia and iter_count < max_iter:
        u_old = u.copy()
        iter_count += 1

        for i in range(1, Nx - 1):
            for j in range(1, Ny - 1):
                # Verificar si dentro.
                if x[i]**2 + y[j]**2 < R**2 and y[j] >= 0:
                    u_new = 0.25 * (u[i+1, j] + u[i-1, j] + u[i, j+1] + u[i, j-1])
                    u[i, j] = u[i, j] + omega * (u_new - u[i, j])

        # Verificar convergencia
        max_diff = np.max(np.abs(u - u_old))
        if max_diff < tolerancia:
            convergencia = True
            informer.info(f'Convergencia alcanzada en {iter_count} iteraciones con diferencia máxima {max_diff:.2e}')

    if not convergencia:
        informer.warning(f'No se alcanzó la convergencia después de {max_iter} iteraciones')

    return x, y, u

# Función para convertir la tabla a LaTeX
def convertir_tabla_a_latex(df: pd.DataFrame, ruta_salida: str):
    latex_code = df.to_latex(index=False)
    with open(ruta_salida, 'w') as f:
        f.write(latex_code)
    informer.info(f"Tabla en formato LaTeX guardada en {ruta_salida}")

if __name__ == '__main__':

    # Parámetros físicos y numéricos
    parser = argparse.ArgumentParser(description='Solución de la ecuación de Laplace en una lámina semicircular.')
    parser.add_argument('--verbosity', type=str, default='INFO', help='Nivel de verbosidad del logger.')
    argumentos_parseados = parser.parse_args()
    informer.setLevel(argumentos_parseados.verbosity)

    # Rutas de salida
    TEMATICA = 'ecuacion_laplace'
    FORMATO_GRAFICAS = '.png'
    OUTPUTS = 'OUTPUTS'
    RUTA_OUTPUTS = f"./{OUTPUTS}"
    RUTA_OUTPUTS_LATEX = f"./e3_latex/figuras"
    Path(RUTA_OUTPUTS).mkdir(parents=True, exist_ok=True)
    Path(RUTA_OUTPUTS_LATEX).mkdir(parents=True, exist_ok=True)

    # Parámetros del problema
    R = 1.0  # Radio de la lámina semicircular
    T0 = 0.0  # Temperatura en el diámetro
    T1 = 50.0   # Temperatura en el borde circular

    # Parámetros numéricos para coordenadas polares
    Nr = 50  # Aumentamos la resolución para mayor precisión
    Ntheta = 50
    tolerancia = 1e-6

    # Resolver en coordenadas polares
    r_polar, theta_polar, u_polar = resolver_laplace_polar(Nr, Ntheta, R, T0, T1, tolerancia=tolerancia)

    # Obtener las temperaturas en los puntos específicos (c)
    puntos_r = [0, R/4, R/2, 3*R/4, R]
    puntos_theta = [0, np.pi/4, np.pi/2, 3*np.pi/4, np.pi]

    datos_puntos = []

    for r_val in puntos_r:
        r_idx = np.argmin(np.abs(r_polar - r_val))
        for theta_val in puntos_theta:
            theta_idx = np.argmin(np.abs(theta_polar - theta_val))
            temp = u_polar[r_idx, theta_idx]
            datos_puntos.append([f"r={r_polar[r_idx]:.2f}, theta={theta_polar[theta_idx]:.2f}", temp])

    # Crear tabla de resultados
    tabla_polar = pd.DataFrame(datos_puntos, columns=['Punto (r, theta)', 'Temperatura (°C)'])

    # Guardar la tabla en CSV y LaTeX
    tabla_polar.to_csv(f"{RUTA_OUTPUTS}/tabla_polar.csv", index=False)
    convertir_tabla_a_latex(tabla_polar, f"{RUTA_OUTPUTS}/{TEMATICA}_tabla_polar.tex")


    # Visualización de la solución numérica en coordenadas polares
    plt.figure(figsize=(8, 6))
    R_grid, Theta_grid = np.meshgrid(r_polar, theta_polar, indexing='ij')
    X = R_grid * np.cos(Theta_grid)
    Y = R_grid * np.sin(Theta_grid)
    plt.contourf(X, Y, u_polar, levels=50, cmap='hot')
    plt.colorbar(label='Temperatura (°C)')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.title('Distribución de temperatura en coordenadas polares')
    plt.axis('equal')
    plt.savefig(f"{RUTA_OUTPUTS}/{TEMATICA}_polar{FORMATO_GRAFICAS}")
    plt.savefig(f"{RUTA_OUTPUTS_LATEX}/{TEMATICA}_polar{FORMATO_GRAFICAS}")

    # Parámetros numéricos para coordenadas cartesianas
    Nx = 50
    Ny = 50
```

```python
# Resolver en coordenadas cartesianas
x_cart, y_cart, u_cart = resolver_laplace_cartesiano(Nx, Ny, R, T0, T1, tolerancia=tolerancia)

# Obtener las temperaturas en los puntos específicos (d)
puntos_x = [0, R/4, R/2, 3*R/4, R]
puntos_y = [0, R/4, R/2, 3*R/4, R]

datos_puntos_cart = []

for x_val in puntos_x:
    x_idx = np.argmin(np.abs(x_cart - x_val))
    for y_val in puntos_y:
        y_idx = np.argmin(np.abs(y_cart - y_val))
        # Verificar si dentro.
        if x_cart[x_idx]**2 + y_cart[y_idx]**2 <= R**2 and y_cart[y_idx] >= 0:
            temp = u_cart[x_idx, y_idx]
            datos_puntos_cart.append([f"x={x_cart[x_idx]:.2f}, y={y_cart[y_idx]:.2f}", temp])

# Crear tabla de resultados
tabla_cartesiano = pd.DataFrame(datos_puntos_cart, columns=['Punto (x, y)', 'Temperatura (°C)'])

# Guardar la tabla en CSV y LaTeX
tabla_cartesiano.to_csv(f"{RUTA_OUTPUTS}/tabla_cartesiano.csv", index=False)
convertir_tabla_a_latex(tabla_cartesiano, f"{RUTA_OUTPUTS}/{TEMATICA}_tabla_cartesiano.tex")

# Visualización de la solución numérica en coordenadas cartesianas
plt.figure(figsize=(8, 6))
X_grid, Y_grid = np.meshgrid(x_cart, y_cart, indexing='ij')
plt.contourf(X_grid, Y_grid, u_cart, levels=50, cmap='hot')
plt.colorbar(label='Temperatura (°C)')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Distribución de temperatura en coordenadas cartesianas')
plt.axis('equal')
plt.savefig(f"{RUTA_OUTPUTS}/{TEMATICA}_cartesiano{FORMATO_GRAFICAS}")
plt.savefig(f"{RUTA_OUTPUTS_LATEX}/{TEMATICA}_cartesiano{FORMATO_GRAFICAS}")

informer.info("Cálculo completado y resultados guardados.")
```