Laboratorio de Métodos Numéricos

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Trabajo Práctico Número 1

Con 15 θ's discretizo alto horno

Integrante	LU	Correo electrónico
Ciruelos Rodríguez, Gonzalo	063/14	gonzalo.ciruelos@gmail.com
Costa, Manuel José Joaquín	035/14	manuc94@hotmail.com
Gatti, Mathias Nicolás	477/14	mathigatti@gmail.com

En el presente trabajo práctico nos proponemos modelar un sistema complejo: la pared de un alto horno de fundición, para poder encontrar isotermas en la misma a partir de conocer las temperaturas interiores y exteriores. A este fin haremos uso de la ecuación diferencial del calor de Laplace, discretizando las derivadas a partir de diferencias finitas atrasadas y adelantadas, al tiempo que particionaremos la pared con ángulos y radios. Luego, podremos plantear un sistema de ecuaciones que nos permita obtener la temperatura en los puntos de la discretización, lo cual haremos usando los métodos de eliminación gaussiana y factorización LU. Notaremos que la matriz de este sistema posee características especiales que aseguran la existencia de factorización LU y permiten optimizar los métodos. Realizaremos experimentos para ver la relación entre la granularidad de la discretización y el tiempo de cómputo de los métodos, así como la proximidad de la isoterma buscada a la pared exterior. También contrastaremos la efectividad de los métodos usados entre si, en distintos casos. Finalmente concluiremos que se verifican las complejidades teóricas de los métodos númericos, además de que a medida que se aumenta la granularidad la posición de la isoterma tiende a estabilizarse en una posición como sería deseable.

Eliminación Gaussiana **Factorización LU** **Discretización** **Diferencia Finita**

1. Introducción teórica

El objetivo del presente informe es resolver un problema práctico mediante el modelado matemático del mismo. Este problema consiste en considerar la sección horizontal de un horno de acero cilíndrico, y dadas las temperaturas en el interior y en el exterior de este, analizar si se encuentra en peligro o no.

Para ello, se debe encontrar una cierta isoterma, que si se encuentra muy cerca (para algún significado de la palabra muy) de la pared del horno, consideraremos que el sistema se encuentra en peligro.

Para modelar la difusión de la temperatura, utilizaremos la ecuación de Laplace.

$$\frac{\partial^2 T(r,\theta)}{\partial r^2} + \frac{1}{r}\frac{\partial^2 T(r,\theta)}{\partial r} + \frac{1}{r^2}\frac{\partial T(r,\theta)}{\partial \theta^2} = 0. \tag{1}$$

Como puede verse en la ecuación (1), esta ecuación depende de variables que son continuas, lo que es matemáticamente válido, pero computacionalmente imposible (a menos que se haga simbólicamente) de calcular.

Para resolver este problema computacionalmente, debemos discretizar el dominio del problema en coordenadas polares. Por eso consideramos una particion $0 < \theta_0 < \theta_1 < ... < \theta_n = 2\pi$ en n ángulos discretos, con $\theta_i - \theta_{i-1} = \Delta\theta$ constante, y una partición $r_i = r_0 < r_1 < ... < r_m = r_e$ en $m+1$ radios discretos con $r_j - r_{j-1} = \Delta r$ para $j = 1, ..., m$.

Entonces ahora, aproximando las derivadas numéricamente utilizando idea del cociente incremental con diferencias finitas podemos obtener un sistema de ecuaciones lineales que describe a el sistema. La formulación detallada del sistema se encuentra en el desarrollo. Para una exposición más completa del tema de resolución de ecuaciones diferenciales con derivadas parciales mediante diferencias finitas se puede consultar [BF11, Cap. 11].

Además, como explicaremos más adelante, la matriz que determina el sistema del problema tiene una forma muy especial, denominada "banda". Esto nos permite asegurar propiedades de la matriz, que serán útiles a la hora de implementar los métodos o intentar optimizarlos.

Para resolver estos sistemas, utilizaremos los dos métodos vistos en clase, eliminación Gaussiana y factorización LU. Por otra parte, como explicaremos mejor y también demostramos en el anexo, podemos realizar estos métodos sin utilizar pivoteo, dado que nunca aparecerá ningun 0 en la diagonal cuando triangulemos la matriz.

2. Desarrollo

2.1. Convenciones

De aquí en adelante, si no se aclara otra cosa, se asumen las siguientes convenciones:

- r_i es el radio que va del centro del horno al borde interno de la pared, mientras que r_e es el radio considerando el borde externo;
- n es la cantidad de ángulos discretos ($0 = \theta_0 < \ldots < \theta_{n-1} = 2\pi - \Delta\theta$) en los que se particiona la pared;
- $m + 1$ es el total de radios discretos ($r_i = r_0 < \ldots < r_m = r_e$);
- $t_{k,j} = T(r_k, \theta_j)$, donde T es la función de temperatura de la pared (que desconocemos);
- $b \in \mathbb{R}^{n\times(m+1)} : b = (b_0, b_1, \ldots, b_{(n\times(m+1))-1})$ será el vector de términos independientes del sistema que plantearemos luego (veremos que la dimensión escogida es correcta).

2.2. Métodos numéricos usados

A partir de la ecuación del calor de Laplace y las discretizaciones de las derivadas parciales dadas en el enunciado del presente trabajo práctico, se puede obtener un sistema de ecuaciones donde las soluciones son las temperaturas en los puntos de la discretización. Es decir que el problema de hallar la isoterma se reduce a dos sub-problemas: el primero es resolver efectivamente el sistema planteado, mientras que el segundo consiste en usar las temperaturas halladas para estimar la posición de la isoterma.

Para resolver el sistema haremos uso de los métodos de eliminación gaussiana sin pivoteo y la factorización LU (además de los algoritmos conocidos como *backward substitution* y *forward substitution*, que son utilizados por estos). Como estos métodos son ampliamente usados y conocidos, no creemos necesario reexplicarlos aquí. Sin embargo, para una explicación en detalle sobre estos métodos sugerimos consultar [DB74, Cap 5.3].

Además, contrastaremos la eficiencia de estos métodos ante distintas situaciones. El hecho de que siempre se puedan aplicar estos métodos a la matriz del sistema se prueba en el apéndice de este informe.

2.2.1. Isoterma

Para hallar la posición de la isoterma (de temperatura t_{iso}), dado un ángulo θ_j de la discretización, lo que haremos es buscar dos radios, r_k y r_{k+1}, tales que $t_{k+1,j} \leq t_{iso} \leq t_{k,j}$. A partir de estos dos valores, fácilmente podemos realizar un ajuste lineal considerando la recta de pendiente $m = \dfrac{t_{k,j} - t_{k+1,j}}{\Delta r}$ que pasa por $(r_k, t_{k,j})$ y $(r_{k+1}, t_{k+1,j})$. Es decir la recta

$$y = m \times (x - r_{k+1}) + t_{k,j}$$

nos da una aproximación del valor de la temperatura en el punto que está a distancia x del centro. Como a nosotros lo que nos interesa es la inversa de esta función (dada la temperatura queremos saber a que distancia del centro se encuentra) despejamos y obtenemos

$$x = \frac{y - t_{k,j}}{m} + r_{k+1} \tag{2}$$

Luego, si queremos obtener la posición de la isoterma solo debemos reemplazar y por el valor de la misma.

2.2.2. Algoritmos optimizados

Adicionalmente, realizaremos una optimización de los algoritmos de eliminación gaussiana y factorización LU, aprovechando una propiedad importante que cumplirá la matriz: ser banda (veremos la validez de esta afirmación en el apartado siguiente).

Básicamente como conoceremos el ancho de la banda inferior de la matriz, al realizar un paso de la eliminación gaussiana no será necesario recorrer todas las filas para colocar 0's debajo del elemento de la diagonal, sino que bastará con poner 0's hasta alcanzar la banda (luego, por definición sabemos que debajo de esta son todos 0's).

Además, cuando querramos hacer el paso de $F_i \leftarrow F_i - cF_k \ \forall i > k$ y c el coeficiente de siempre (para poner 0's debajo del elemento k-ésimo de la diagonal), es notorio que F_k, es un vector que en un momento tiene todos coeficientes iguales a 0, dado que la matriz es banda. Por esa razón podemos terminar el paso de resta de filas en ese momento.

Esto nos permitirá que cada paso de la eliminación gaussiana (es decir, poner 0's en toda una columna) solo nos cueste $O(n^2)$ flops[1]. Esto será ampliado en la parte de resultados y discusión.

2.3. Armado del sistema de ecuaciones y su matriz asociada

Queremos armar un sistema de ecuaciones que nos permita hallar los valores de la temperatura en los puntos de nuestra discretización.

Primero contamos con n variables, los $t_{0,j}$ con $j = 0, 1, \ldots, n-1$ (es decir, las temperaturas interiores), cuyos valores conocemos *a priori* pues nos son dados como inputs. Lo mismo sucede con $t_{m,j}$ para $j = 0, 1, ..., n-1$ (temperaturas externas). Luego, para las temperaturas interiores sabemos gracias a la función de Laplace y a la discretización de las derivadas que vale

$$\frac{t_{k-1,j} - 2t_{k,j} + t_{k+1,j}}{(\Delta r)^2} + \frac{1}{r_k} \times \frac{t_{k,j} - t_{k-1,j}}{\Delta r} + \frac{1}{r_k^2} \times \frac{t_{k,j-1} - 2t_{k,j} + t_{k,j+1}}{(\Delta\theta)^2} = 0$$

Reescribiendo convenientemente la ecuación de arriba nos queda que

$$\frac{r_k - \Delta r}{r_k(\Delta r)^2} t_{k-1,j} + \frac{1}{r_k^2(\Delta\theta)^2} t_{k,j-1} + \frac{r_k\Delta r(\Delta\theta)^2 - 2(\Delta r)^2}{r_k^2(\Delta r)^2(\Delta\theta)^2} t_{k,j} + \frac{1}{r_k^2(\Delta\theta)^2} t_{k,j+1} + \frac{1}{(\Delta r)^2} t_{k+1,j} = 0 \qquad (3)$$

Hay que distinguir dos casos bordes que deben tratarse un poco diferente: si $j = 0$ entonces en lugar de $t_{k,j-1}$ se usa $t_{k,n-1}$, mientras que si $j = n-1$ en lugar de $t_{k,j+1}$ va $t_{k,0}$.

Vale destacar que si bien todos los coeficientes dependen de la distancia al centro del horno (r_k), ninguno lo hace respecto del ángulo concreto en que se encuentra el punto. Esto resulta muy razonable si tenemos en cuenta que el valor del ángulo depende exclusivamente del sistema de referencia escogido (dónde ubicamos el ángulo 0), mientras que dado un punto de la pared es lógico que su temperatura no dependa del sistema de referencia escogido para su medición.

Para facilitar la lectura, de ahora en adelante llamaremos a los coeficientes de la ecuación (3):

- $a_k = \dfrac{r_k - \Delta r}{r_k(\Delta r)^2}$
- $b_k = \dfrac{1}{r_k^2(\Delta\theta)^2}$
- $c_k = \dfrac{r_k\Delta r(\Delta\theta)^2 - 2(\Delta r)^2}{r_k^2(\Delta r)^2(\Delta\theta)^2}$
- $d_k = \dfrac{1}{(\Delta r)^2}$

Pensamos ahora el orden que le daremos a los puntos de la discretización. Este será el orden en el que escribiremos las ecuaciones. Sin demasiadas complicaciones, un orden razonable es primero por ángulo (desde θ_0 y avanzando en sentido antihorario) y luego por radio (de menor a mayor) respetando el orden relativo previo. Luego, dado un $t_{k,j}$ para obtener su posición (contada desde 0) en este orden debemos realizar la siguiente cuenta

$$posicion(t_{k,j}) = k \times n + j \qquad (4)$$

[1] Operaciones de punto flotante, de aquí en adelante flops.

En este sentido, la ecuación (3) tiene sus variables ordenadas.

La cantidad total de variables es $n \times (m+1)$ (cantidad de ángulos por cantidad de radios), y como para cada una podemos o bien plantear la ecuación (3), o bien ya conocemos su valor, podemos armar un sistema de $n \times (m+1)$ ecuaciones lineales.

A continuación presentamos el sistema de ecuaciones:

$$\begin{cases} t_{0,0} = b_0 \\ t_{0,1} = b_1 \\ \vdots \\ t_{0,n-1} = b_{n-1} \\ a_1 t_{0,0} + c_1 t_{1,0} + b_1 t_{1,1} + b_1 t_{1,n-1} + d_1 t_{2,0} = 0 = b_n \\ a_1 t_{0,1} + b_1 t_{1,0} + c_1 t_{1,1} + b_1 t_{1,2} + d_1 t_{2,1} = 0 = b_{n+1} \\ \vdots \\ a_1 t_{0,n-2} + b_1 t_{1,n-3} + c_1 t_{1,n-2} + b_1 t_{1,n-1} + d_1 t_{2,n-1} = 0 = b_{2(n-1)} \\ a_1 t_{0,n-1} + b_1 t_{1,0} + b_1 t_{1,n-2} + c_1 t_{1,n-1} + d_1 t_{2,n-1} = 0 = b_{2n-1} \\ a_2 t_{1,0} + c_2 t_{2,0} + b_2 t_{2,1} + b_2 t_{2,n-1} + d_2 t_{3,0} = 0 = b_{2n} \\ a_{m-1} t_{m-2,n-1} + b_{m-1} t_{m-1,n-1} + c_{m-1} t_{m,n-1} + d_{m-1} t_{m-1,n-2} + d_{m-1} t_{m-1,0} = 0 = b_{n \times m-1} \\ t_{m,0} = b_{n \times m} \\ \vdots \\ t_{m,n-1} = b_{n \times (m+1)-1} \end{cases} \tag{5}$$

b tiene valores pasados por el usuario en las primeras n posiciones y en las n últimas, siendo el resto todos ceros.

Como primera observación importante notemos que el valor de cada $t_{k,j}$ $(1 \leq k < m)$ depende exclusivamente del valor de sus cuatro vecinos, lo que permite prever que cada fila de la matriz asociada tendrá sólo cinco elementos distintos de 0.

Para realizar la siguiente observación primero veamos un ejemplo concreto. Consideremos la siguiente discretización con $n = 4$ (particionamos la pared en cuatro ángulos iguales) y $m = 3$ (4 radios en total). El sistema tendrá entonces $4 \times (3+1) = 16$ variables, con 16 ecuaciones. En este caso, el sistema (5) queda así

$$\begin{cases} t_{0,0} = b_0 \\ t_{0,1} = b_1 \\ t_{0,2} = b_2 \\ t_{0,3} = b_3 \\ a_1 t_{0,0} + c_1 t_{1,0} + b_1 t_{1,1} + b_1 t_{1,3} + d_1 t_{2,0} = 0 = b_4 \\ a_1 t_{0,1} + b_1 t_{1,0} + c_1 t_{1,1} + b_1 t_{1,2} + d_1 t_{2,1} = 0 = b_5 \\ a_1 t_{0,2} + b_1 t_{1,1} + c_1 t_{1,2} + b_1 t_{1,3} + d_1 t_{2,2} = 0 = b_6 \\ a_1 t_{0,3} + b_1 t_{1,0} + b_1 t_{1,2} + c_1 t_{1,3} + d_1 t_{2,3} = 0 = b_7 \\ a_2 t_{1,0} + c_2 t_{2,0} + b_2 t_{2,1} + b_2 t_{2,3} + d_2 t_{3,0} = 0 = b_8 \\ a_2 t_{1,1} + b_2 t_{2,0} + c_2 t_{2,1} + b_2 t_{2,2} + d_2 t_{3,1} = 0 = b_9 \\ a_2 t_{1,2} + b_2 t_{2,1} + c_2 t_{2,2} + b_2 t_{2,3} + d_2 t_{3,2} = 0 = b_{10} \\ a_2 t_{1,3} + b_2 t_{2,0} + b_2 t_{2,2} + c_2 t_{2,3} + d_2 t_{3,3} = 0 = b_{11} \\ t_{3,0} = b_{12} \\ t_{3,1} = b_{13} \\ t_{3,2} = b_{14} \\ t_{3,3} = b_{15} \end{cases} \tag{6}$$

Ahora escribamos la matriz asociada a (6)

$$A = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a_1 & 0 & 0 & 0 & c_1 & b_1 & 0 & b_1 & d_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & a_1 & 0 & 0 & b_1 & c_1 & b_1 & 0 & 0 & d_1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & a_1 & 0 & 0 & b_1 & c_1 & b_1 & 0 & 0 & d_1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & a_1 & b_1 & 0 & b_1 & c_1 & 0 & 0 & 0 & d_1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & a_2 & 0 & 0 & 0 & c_2 & b_2 & 0 & b_2 & d_2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & a_2 & 0 & 0 & b_2 & c_2 & b_2 & 0 & 0 & d_2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & a_2 & 0 & 0 & b_2 & c_2 & b_2 & 0 & 0 & d_2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & a_2 & b_2 & 0 & b_2 & c_2 & 0 & 0 & 0 & d_2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

Rápidamente notamos que A cumple la propiedad de ser una matriz banda. No solo eso, sino que además el ancho de ambas bandas (superior e inferior) es igual a 4, lo que coincide con n.

Tratemos ahora de convencernos que esto no se debe a una característica particular del ejemplo y que vale en el caso general. Por un lado está claro que tanto las primeras como las últimas n filas van a tener 1's en la diagonal y 0's en el resto de las posiciones, pues esas son las temperaturas que conocemos *a priori*. Por lo tanto, con estas filas no hay problema.

Para las filas de en medio analicemos la distancia (entiéndase *cantidad de variables de distancia*) desde el elemento de la diagonal hasta el último elemento no nulo de la fila, tanto a izquierda como a derecha. Viendo la ecuación (3) es fácil darse cuenta que la variable más alejada (con coeficiente no nulo) a izquierda de $t_{k,j}$ es $t_{k-1,j}$, mientras que a derecha es $t_{k+1,j}$. Ambas variables están a distancia n de $t_{k,j}$, pues para llegar desde el punto (r_k, θ_j) a $(r_k - 1, \theta_j)$ o $(r_k + 1, \theta_j)$ es necesario dar una vuelta completa (n ángulos discretos) en sentido horario o antihorario respectivamente. Por lo tanto cualquier coeficiente de la matriz asociada a más de distancia n de la diagonal es nulo. Con esto ya nos aseguramos que la matriz es banda, cuyo ancho es como mucho n. Pero además como dijimos que los coeficientes que acompañan a $t_{k-1,j}$ y $t_{k+1,j}$ son no nulos (lo cual vale siempre pues efectivamente a_k y d_k nunca pueden ser 0[2]), es cierto que el ancho de ambas bandas es exactamente n.

2.4. Estructuración del código

Para el modelado del problema diseñamos dos módulos: Matriz y Sistema.

2.4.1. Matriz

Representación interna La representación interna es realmente simple: un vector de vectores fila, y dos enteros, n_ para la cantidad de filas y m_ para la cantidad de columnas.

Interfaz La interfaz de Matriz provee las siguientes operaciones:[3]

- Matriz(**int** n, **int** m, **double** init): constructor de la matriz. n es la cantidad de filas y m la de columnas, mientras que init será el valor que tendrán inicialmente todos los elementos de la matriz.
- Otras operaciones usuales: un constructor por copia, una función para imprimir la matriz, funciones que devuelven la cantidad de filas y de columnas, y un operador que permite acceder a cada elemento de la matriz (se numera a partir de 0).

[2] Efectivamente $a_k = 0 \Leftrightarrow r_k - \Delta r = 0$, pero como $r_k = r_i + k \times \Delta r$ la única forma de que esa resta sea 0 es que r_i sea 0, lo cual no tiene sentido en el contexto del problema (sería pura pared).

[3] Cuando se escribe la aridad de la función la misma puede no coincidir con la notación usada en C++. Esto está bien pues lo único que se busca aquí es dar una orientación de lo que hace cada función y no código preciso.

- **vector<double>** backward_subst(**Matriz** A, **vector<double>** b): función que dada una matriz cuadrada triangular superior sin ceros en la diagonal y un vector de tamaño igual a la cantidad de filas de la matriz, resuelve el sistema planteado mediante el algoritmo de *backward substitution*.
- **vector<double>** forward_subst(**Matriz** A, **vector<double>** b): análogo al anterior pero para matrices triangulares inferiores, resuelve el sistema usando *forward substitution*.
- **vector<double>** gaussian_elim(**Matriz** A, **vector<double>** b): resuelve el sistema Ax=b usando el método de eliminación gaussiana. Primero triangula A, y luego aplica backward_subst.
- **pair<Matriz *, Matriz *>** LU_fact(**Matriz** A): si A tiene factorización LU, la función resuelve el sistema Ax=b usando este método.
- **vector<double>** gaussian_elim_banda(**Matriz** A, **vector<double>** b, **int** w): versión optimizada de gaussian_elim que aprovecha el hecho de que A sea banda de ancho superior e inferior w.
- **pair<Matriz *, Matriz *>** LU_fact_banda(**Matriz** A, **int** ancho): versión optimizada de LU_fact que aprovecha el hecho de que A sea banda de ancho superior e inferior w.

2.4.2. Sistema

Sistema es un módulo que engloba todo lo relacionado al modelado.

Representación interna Se almacenan varios valores: la cantidad de radios y ángulos de la discretización ($m\_mas\_uno\_$ y $n\_$), y el radio interno, el externo, Δr y $\Delta\theta$. Se guarda un puntero a la matriz asociada al sistema de ecuaciones, A. bs es un vector cuyos elementos a su vez son vectores, uno por cada instancia del problema que el usuario haya pasado. En *soluciones* se guardarán los vectores solución del sistema para cada instancia. Adicionalmente, tenemos algunas funciones auxiliares:

- *col_matriz* que permite obtener dados k, j la posición de la variable $t_{k,j}$ en el orden que les dimos (realizando la operación (1));
- **double** *resolver_isoterma*(**vector<double>** radios, **double** isoterma, **double** eps = 0.0001): *radios* tiene la temperatura en cada radio sobre un ángulo fijo, *isoterma* es el valor de la isoterma buscada, y *eps* es un parámetro *hardcodeado* que define el error que se tiene en cuenta para realizar una comparación en el caso extremo en que la isoterma buscada no se encuentre entre ningún par de radios, es decir, que el radio buscado es menor que r_i o mayor que r_e). Por decisión, en estos casos devolvemos r_i o r_e según la temperatura interior sea mayor que la isoterma buscada o no respectivamente. Sino, lo que devolvemos es la distancia desde el centro del horno hasta el punto donde estimamos se encuentra la isoterma (usando el ajuste lineal explicado en sección (2.2)).
- Tres funciones que nos dan 3 criterios para decidir si el sistema está en peligro o no, dado un vector que tiene la posición de la isoterma en cada ángulo: mediana, promedio y máximo.

Interfaz La interfaz de Sistema provee las siguientes operaciones:[4]

- Sistema(**double** r_i, **double** r_e, **int** m_mas_uno, **int** n, **vector<vector<double>>** interiores, **vector<vector<double>>** exteriores) : constructor del sistema. Cada vector de interiores es una medición de las temperaturas interiores en una determinada instancia (la cual está dada por la posición de la medición en interiores). exteriores es análogo a interiores para las mediciones de temperaturas exteriores. En una primera etapa inicializa los atributos de tipo double. Luego, arma A y bs.
- **void** solve(**ofstream&** f_soluciones, **metodo** met): toma como parámetros un archivo de salida donde imprimirá las soluciones del sistema con el formato solicitado por la cátedra, y un metodo, que es un tipo numerado definido por nosotros que puede tomar cuatro valores: ELIM_GAUSSIANA, FACTORIZACION_LU, ELIM_GAUSSIANA_BANDA, FACTORIZACION_LU_BANDA. El método escogido será el que se use para resolver el sistema.

[4] Cuando se escribe la aridad de la función la misma puede no coincidir con la notación usada en C++. Esto está bien pues lo único que se busca aquí es dar una orientación de lo que hace cada función y no código preciso.

- **void** isotermas(**ofstream&** f_isotermas, **double** isoterma): toma un archivo de salida en el que imprimirá la lista de las posiciones de la isoterma buscada para ángulo. La forma en la que trabaja consiste en iterar sobre todos los ángulos y para cada uno llamar a la auxiliar resolver_isoterma, y posteriormente imprimir el resultado.

2.5. Experimentación

Pasemos a detallar los métodos que usamos para la experimentación.

En cuanto a la experimentación en lo que concierne al comportamiento de los resultados del programa cuando se cambian las discretizaciones, podemos decir que tuvimos muchos intentos hasta obtener el resultado deseado. De hecho, la experimentación extensiva que realizamos nos permitió detectar un pequeño bug del código que no se hacía notar en los tests de la cátedra.

Para evaluar el comportamiento de los resultados, diseñamos configuraciones de sistemas que varían lo más posible cuando se cambia la discretización, como se verá en los resultados y la discusión.

Para evaluar como cambian los resultados cuando se aumenta la granularidad con respecto a los ángulos, lo que hicimos fue simplemente hacer un sistema cuyas temperaturas exteriores sean todas iguales, excepto 3 o 4 que son mucho más altas y están juntas. En consecuencia, cuando se achique la granularidad, estos detalles se perderán y el sistema parecerá más estable de lo que en realidad es.

Para evaluar como varían los resultados cuando se aumenta la granularidad con respecto a los radios, lo que hicimos fue similar, solo que no hace falta tomar un sistema muy especial, dado que las diferencias se notan fácilmente.

En cuanto a la experimentación en lo que concierne a la comparación del tiempo que le toma resolver el problema a cada uno de los métodos, lo que hicimos fue crear sistemas de variada granularidad y calcular cuanto tiempo le tomaba al programa resolverlo (desde que la matriz del sistema se terminó de construir, hasta que se resolvió el problema para todos los b's).

Los tests fueron planteados por separado para sistemas en los que se debe resolver un único b y otros en los que se debe resolver múltiples bs. Además, comparamos las implementaciones de eliminación Gaussiana, factorización LU y sus respectivas implementaciones optimizadas.

Al principio realizamos únicos tests que no distinguían granularidad de ángulos y de radios, pero luego notamos que en nuestras implementaciones optimizadas, teóricamente, podían surgir diferencias entre un cambio en la granularidad de ángulos y otro en la de radios, cosa que luego verificamos empíricamente. Por esta razón, decidimos separar los tests de tiempos de único b en 2, uno que varíe cantidad de radios y otro cantidad de ángulos.

Para esto, repetimos 50 veces cada corrida, 30 veces las que tardaban mas de 30 segundos y 15 las que tomaban más de un minuto, dado que la desviación standard es mucho menor (dado que la desviación se debe generalmente a cambios de contexto durante el runtime del programa, que para programas que corren poco tiempo varía mucho).

Elegimos este número dado que la desviación standard era lo suficientemente chica como para que el experimento fuera confiable, y al mismo tiempo que tardaran un tiempo razonable para permitirnos realizar muchos experimentos.

Los tiempos fueron tomados en una computadora que contaba con un procesador Intel Core i5-2450M @ 2.50GHz, con 8GB de memoria RAM.

3. Resultados y discusión

3.1. Criterios de análisis para la isoterma

Para tener una referencia a partir de la cual decidir si el horno podía llegar a estar en peligro o no, decidimos utilizar tres criterios clásicos: el promedio, la mediana y el máximo, ya que cada uno presenta ciertos aspectos útiles.

La media o promedio permite tener una idea general de los valores que tiene la isoterma, permitiendo darnos una idea básica de que tantos ángulos o con que tanta intensidad están superando el umbral, como desventaja presenta problemas al haber outliers en las mediciones de temperatura lo cual puede distorsionar la medición.

La mediana permite solucionar el problema previo por su mayor robustez; medidas que estén fuera de lugar respecto de las que aparezcan por mayoría no afectarán el resultado final y se obtendrá una idea más clara del valor que se está teniendo mayormente mientras que los valores estén dentro de cierto rango acotado.

El máximo permite ver picos que podrían pasar desapercibidos viendo únicamente la media o mediana pero como desventaja se pierden tendencias generales y los outliers lo afectan en gran medida.

Una vez escogidos estos criterios de análisis lo que hicimos fue, para una instancia de isoterma dada, ver si al calcular el promedio, mediana o máximo alguno supera cierto umbral escogido. De suceder esto entonces se dirá que el horno esta en peligro. Para elegir el umbral se aconseja basarse en casos previos de hornos de características similares que sufrieron daños, a partir de esto estudiar la isoterma en esos casos con los criterios establecidos previamente y deducir valores adecuados dentro de los cuales sea recomendable trabajar.

3.2. Relación entre granularidad y precisión en el calculo de la isoterma

A través de una serie de experimentos buscamos estudiar que factores contribuyen a una mayor precisión en el cálculo de la posición de la isoterma. Logrando así una predicción más fiable del peligro en el que puede llegar a estar el horno y sin perder el tiempo con cálculos innecesarios.

Al realizar los experimentos nos interesamos en estudiar como la granularidad afectaba la precisión de la estimación de la isoterma, para así poder estar seguros si el horno estaba o no en peligro. Para hacer esto separamos los experimentos en dos, primero estudiamos qué pasaba cuando utilizábamos una mayor cantidad de ángulos y luego lo mismo para los radios.

En este documento presentamos las imágenes mas representativas de nuestra investigación pero para una mayor profundización se puede visitar el link de 5.2, la segunda sección del apéndice, donde están todos nuestros experimentos sobre isotermas.

3.2.1. Granularidad de los ángulos

Antes de realizar los experimentos, nos planteamos nuestras hipótesis. Como al aumentar la cantidad de ángulos se podrán detectar mejor los cambios bruscos en la isoterma, esto permitirá ver con mayor claridad donde comienzan y donde acaban los picos mientras mejor sea la granularidad. En caso de utilizar una granularidad pobre se verán picos poco precisos o incluso gráficos de isotermas erróneos que fallan en detectar el pico, lo cual podría conllevar consecuencias muy graves al no advertir un posible peligro en el horno.

En este primer experimento se utiliza una instancia en la cual la temperatura en todos los ángulos externos es igual excepto en una pequeña zona donde aumenta. Si se piensa a r_e y r_i como listas, lo que hicimos para disminuir la granularidad fue empezar con dos listas de largo 50 y luego ir tomando primero los índices pares de cada lista, luego los índices múltiplos de 3, y así sucesivamente.

Este experimento puede pensarse como que disminuimos la cantidad de mediciones que tomamos de la temperatura dentro y fuera del horno, pero sin embargo manteniendo su equidistribución.

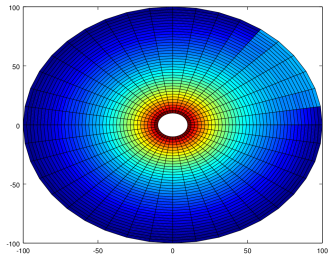

Figura 1

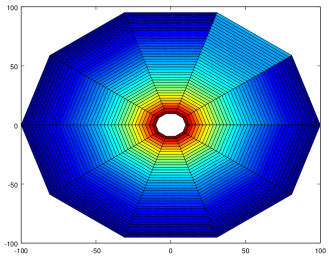

Figura 2

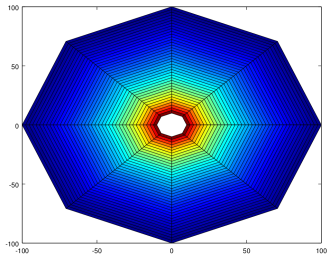

Figura 3

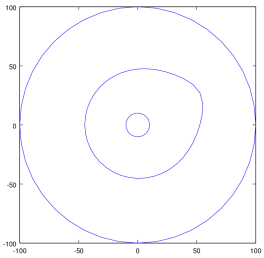

Figura 4

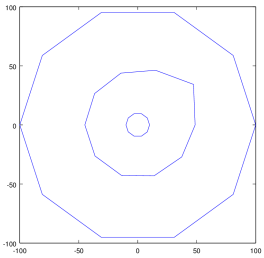

Figura 5

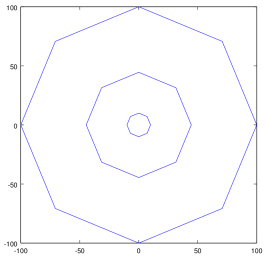

Figura 6

En las Figuras 1 y 4, con 30 ángulos, se puede detectara la perfección la existencia y ubicación de un pico en la isoterma. Luego, en las Figuras 2 y 5, con 10 ángulos si bien se detecta un pico, la forma que se muestra está muy alejada de la real. Nótese por ejemplo que la dirección en la que se encuentra el pico no es la misma que la anterior, además de que el tamaño es distinto. Esto podría causar falsas alarmas o la ausencia de las mismas. Finalmente, en las Figuras 3 y 6, en la que se discretizó con 8 ángulos se puede observar como el pico desaparece fallando dramáticamente la predicción en la forma de la isoterma.

Para mas detalles de la evolución de la isoterma y las temperaturas calculadas en el sistema se recomienda recurrir al link de la sección 2 del apéndice (5.2), en la animación "comp_angs_iso" se podrá notar con mayor claridad lo nombrado previamente, la isoterma ira marcando el pico con mayor precisión a medida que se utilicen mas ángulos, en los casos en que la cantidad de ángulos sea insuficiente el pico y las temperaturas en esa zona estarán muy distorsionadas, esto se puede observar claramente en "comp_angs_temp", donde las temperaturas en la zona del pico se calculan de forma imprecisa debido a la utilización de una cantidad de ángulos muy pobre.

3.2.2. Granularidad de los radios

Al estudiar las posibilidades en la cantidad de radios a utilizar creemos que observaremos como a mayor granularidad mejora la posición de la isoterma, convergiendo a la posición real. A diferencia de los ángulos, con los radios los picos serán detectados sin tomar demasiadas precauciones pero para asegurar que el tamaño de los picos, y de la isoterma en general, sean predichos de forma confiable se recomendará utilizar una cantidad de radios adecuadamente alta según la necesidad que se tenga.

Para verificar un poco esto experimentamos con muchas instancias de prueba, y llegamos a dos instancias que nos parecieron superadoras, ya que representan la idea principal que queremos dar del comportamiento del sistema cuando se varía la granularidad de los radios.

La primera forma una isoterma con forma de óvalo debido a un aumento en la temperatura externa en dos zonas opuestas. La segunda es un círculo perfecto producido por una temperatura constante a lo largo de todo el interior y exterior del horno. En ambos casos veremos como la figura va cambiando su tamaño al tender a la isoterma real.

En la instancia 1 empezamos con 10 radios y fuimos aumentando de a 10, mientras que en la instancia 2 empezamos con 5 y fuimos aumentando de a 5. Ponemos solo algunos pasos significativos de todo ese camino. Para ver todos los pasos (animados), se puede visitar el link del apéndice, 5.2.

Instancia 1

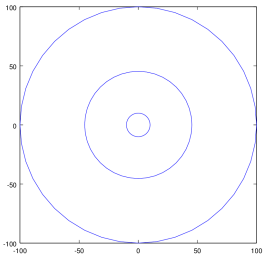

Figura 7: 60 radios

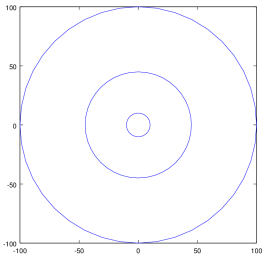

Figura 8: 40 radios

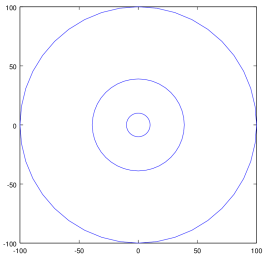

Figura 9: 10 radios

Sucede lo mismo que en en el experimento anterior, al tomar una cantidad relativamente alta de radios se logra un error muy pequeño del tamaño de la isoterma, en este caso se utilizaron 60. La diferencia entre utilizar 40, 50 y 60 radios no fue muy grande, al menos para nuestros estandáres, por lo cual se podrían utilizar simplemente 40 consiguiendo un buena relación entre tiempo de cálculo y aproximación al tamaño de la isoterma.

Para apreciar mejor esto pueden acceder al link de la segunda sección del apéndice (5.2) y ver las animaciones "comp_rads_iso" y "comp_rads_iso".

Instancia 2

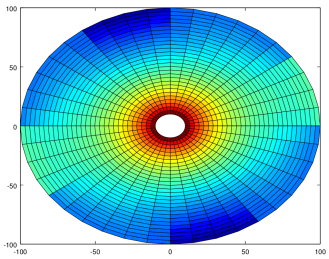

Figura 10

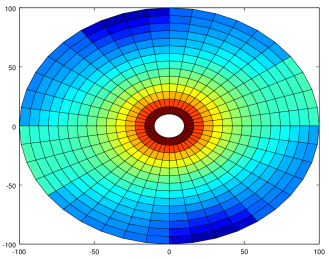

Figura 11

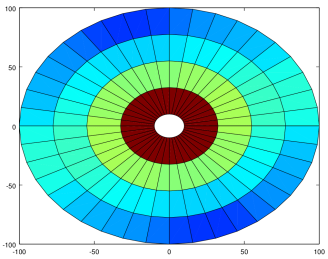

Figura 12

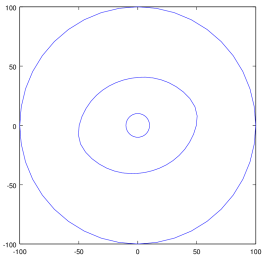

Figura 13

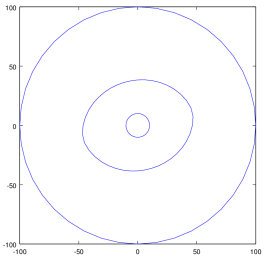

Figura 14

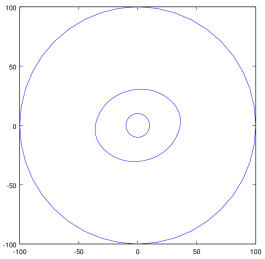

Figura 15

Se puede observar como, a pesar de tener el mismo conjunto de temperaturas exteriores e interiores, y la misma discretización con respecto a los ángulos, hay una gran diferencia entre las Figuras 13 y 10 , con 30 radios, que consigue una aproximación muy buena del tamaño de la isoterma y las Figuras 15 y 12 que con sus 5 radios consigue una isoterma muy alejada de la real.

Además, se pueden apreciar los valores intermedios en las Figuras 14 y 11 (en las que se usan 15 radios) para ver como evolucionan las soluciones frente a estos cambios.

Para apreciar mejor esto pueden acceder al link de la segunda sección del apéndice (5.2) y ver las animaciones “comp_rads_iso2” y “comp_rads_temp2”.

3.3. Evaluación de los métodos

3.3.1. Único b

Nuestro primer análisis de los métodos utilizados para la resolución del problema va a consistir en uno teórico.

La resolución mediante el método de eliminación Gaussiana tiene dos partes, la primera es la eliminación gaussiana propiamente dicha, que tiene un costo de $O(k^3)$ flops, donde k es la cantidad de filas (y columnas) de la matriz y la segunda es el algoritmo llamado *backwards substitution*, que tiene un costo de $O(k^2)$ flops.

La resolución mediante el método de factorización LU tiene tres partes, la primera es obtener la factorización LU de la matriz en cuestión, que tiene un costo de $O(k^3)$, y luego aplicar *forward substitution* y *backward substitution*, cada uno con un costo de $O(n^2)$ flops.

Para medir los tiempos de ambas implementaciones decidimos hacer dos experimentos separados, sin embargo similares. En uno fijamos una cantidad de radios y movemos la cantidad de ángulos, y en el otro al revés. Lo hicimos así para que las dimensiones de la matriz (ancho o alto, dado que son iguales porque es cuadrada) crezca linealmente, ya que si hacemos variar ángulos y radios al mismo tiempo, las dimensiones dejan de crecer linealmente. Queremos que las dimensiones crezcan linealmente para que los resultados se entiendan mejor, dado que una escala lineal es, generalmente, más natural a la vista.

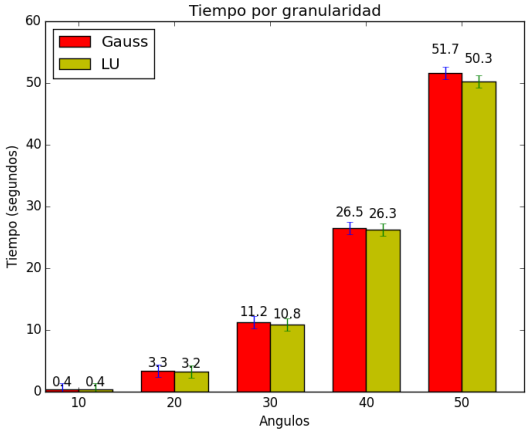

Figura 16: Tiempo tomado por la nuestra, implementación de eliminación gaussiana y de factorización LU para resolver el problema. La granularidad de radios está fija en 40 y la de ángulos se indica en el eje x. La barra principal indica el promedio, y el segmento indica la desviación standard.

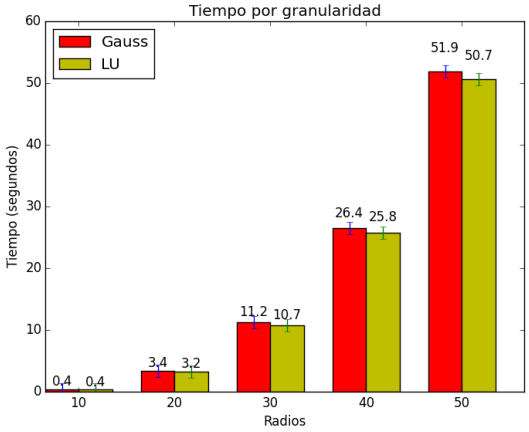

Figura 17: Tiempo tomado por nuestra implementación de eliminación gaussiana y de factorización LU para resolver el problema. La granularidad de ángulos está fija en 40 y la de radios se indica en el eje x. La barra principal indica el promedio, y el segmento indica la desviación standard.

Como se ve en las figuras 16 y 17, las performances entre las implementaciones son realmente similares. También puede observarse algo que nos llamó la atención, que es que el método por eliminación gaussiana tuvo consistentemente peor performance que LU, contradiciendo parcialmente el análisis a priori. Sin embargo, la diferencia es mínima: siempre menor al 3 %.

Por esta razón no creemos que sea algo importante a tener en cuenta, y podríamos atribuirlo a optimizaciones del compilador, dado que los códigos son realmente parecidos (de hecho hicimos nuestra implementación de la factorización LU sobre nuestra implementación de eliminación gaussiana), y las partes que son diferentes le agregan complejidad a la factorización LU.

Por eso lo atribuimos a cuestiones relacionadas con la optimización en tiempo de compilación y no a errores en las mediciones, dado que los tests fueron corridos varias veces por el hecho de que estos resultados eran llamativos.

Por otro lado, vemos que dividir entre ángulos y radios no hace diferencia en la performance, dado que como esperábamos, el runtime solamente depende del tamaño de la matriz, es decir $n(m+1)$, que en ambos gráficos es igual columna a columna.

Ahora es el turno de las implementaciones que aprovechan que la matriz es banda. La diferencia es

enorme. Para comparar rápidamente, podemos usar una tabla:

Implementación	Gauss	Gauss Banda	LU	LU Banda
Tiempo (segundos)	26.24	0.98	25.53	1.04

Figura 18: Tiempo promedio tomado por las implementaciones para resolver un sistema con $n = m + 1 = 40$.

Con la Figura 18 se observa como la diferencia entre las implementaciones *vanilla* y las optimizadas es abismal, superando el 2600 %. Esta tabla no pretende analizar las implementaciones optimizadas, simplemente probar la diferencia de rendimiento que se obtiene, dado que poner ambas implementaciones en un mismo gráfico no nos permitiría apreciar la diferencia.

A continuación analizaremos las implementaciones optimizadas.

Como puede verse en el código, la complejidad de los algoritmos optimizados para realizar la eliminación Gaussiana (y por lo tanto la factorización LU) es $O(n^3(m+1))$. Esto se debe a que, para cada fila (hay $n(m+1)$ filas) debemos realizar un trabajo que cuesta $O(n^2)$ flops.

Lo anterior es fácilmente justificable, porque en cada paso de la eliminación gaussiana el objetivo es poner ceros debajo de la diagonal, entonces solo debemos modificar tantas filas como cantidad de ángulos hay, dado que la matriz es banda y tiene la forma que fue explicada anteriormente.

Además de esas filas, sólo se deben modificar tantos coeficientes como ángulos, dado que sabemos que para la derecha de la fila en cuestión hay solamente ceros. Nuevamente, esto se sigue de la explicación anterior sobre la construcción de la matriz, que fue explicada anteriormente.

Debido a esto, esperamos que cueste mucho mas aumentar la granularidad con respecto a los ángulos que con respecto a los radios, como se vió reflejado en los experimentos que siguen.

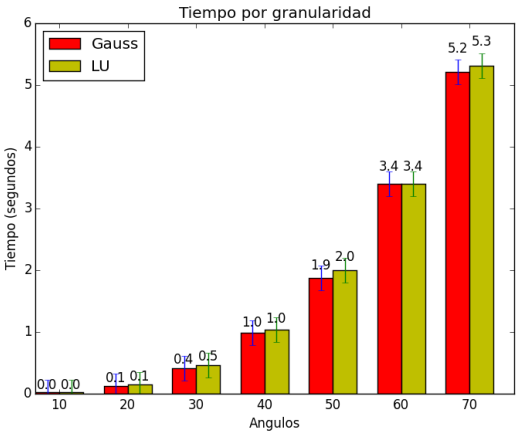

Figura 19: Tiempo tomado por la nuestra implementación optimizada de eliminación gaussiana y de factorización LU para resolver el problema. La granularidad de radios está fija en 40 y la de ángulos se indica en el eje x. La barra principal indica el promedio, y el segmento indica la desviación standard.

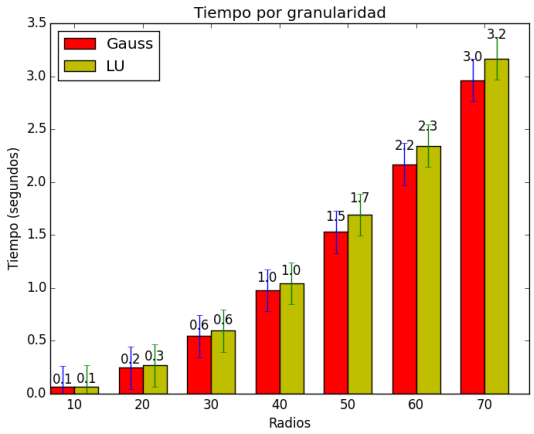

Figura 20: Tiempo tomado por nuestra implementación optimizada de eliminación gaussiana y de factorización LU para resolver el problema. La granularidad de ángulos está fija en 40 y la de radios se indica en el eje x. La barra principal indica el promedio, y el segmento indica la desviación standard.

Los resultados que se ven en las Figuras 19 y 20 son muy interesantes. Primero notemos que como los procesos de eliminación gaussiana y factorización LU ambos consumen menos tiempo, se nota más la ventaja que le saca gauss a LU con un solo b, dado que los procesos de substitución empiezan a pesar asintóticamente.

Como la complejidad de los algoritmos optimizados es $O(n^3(m+1))$, en los experimentos se refleja que es mucho más caro aumentar la granularidad con respecto a los radios que con respecto a los ángulos, como sostuvimos anteriormente.

3.3.2. Múltiples b's

Cuando buscamos simular un escenario similar al experimento anterior, pero donde las condiciones de borde (temperaturas interiores y exteriores) cambian en distintos instantes de tiempo, la situación es muy distinta. Aquí esperaríamos ver el verdadero poder de la factorización LU.

Por esta razón, esperamos que la implementación de la factorización LU supere ampliamente a la de la eliminación Gaussiana, dado que en la Gaussiana se paga un costo cúbico cada vez que se quiere resolver el sistema, mientras que en con la factorización LU el costo cúbico se paga solo una vez.

Por esta razón se puede decir que, para muchas instancias, el costo de resolver $Ax = b$ es cúbico para la eliminación gaussiana y cuadrático amortizado para la factorización LU.

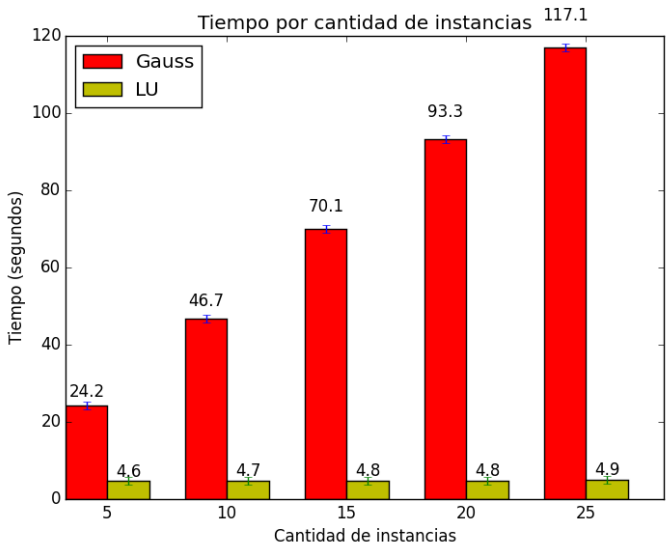

Figura 21: Tiempo tomado por nuestra implementación de eliminación gaussiana y de factorización LU para resolver el problema para varias instancias de $n = m + 1 = 30$. La barra principal indica el promedio, y el segmento indica la desviación standard.

En los resultados se refleja lo que esperábamos. Como se ve en la figura 21, luego de aplicar la factorización LU, el costo que hay que pagar para resolver cada sistema es muy poco, lo cual permite una excelente performance.

Por otra parte, se observa que el tiempo que le toma a la eliminación gaussiana resolver n instancias, es también es lineal en la cantidad de instancias, pero con una pendiente mucho mayor, lo cual confirma más aún nuestras expectativas derivadas de la teoría.

En cuanto a las implementaciones optimizadas, esperamos que su comportamiento sea similar a los resultados anteriores.

Con estas instancias, además de simplemente tomar tiempos, también guardamos los resultados y obtuvimos imágenes, a partir de las cuales armamos animaciones. Estas animaciones pueden verse en los links de YouTube de 5.2.

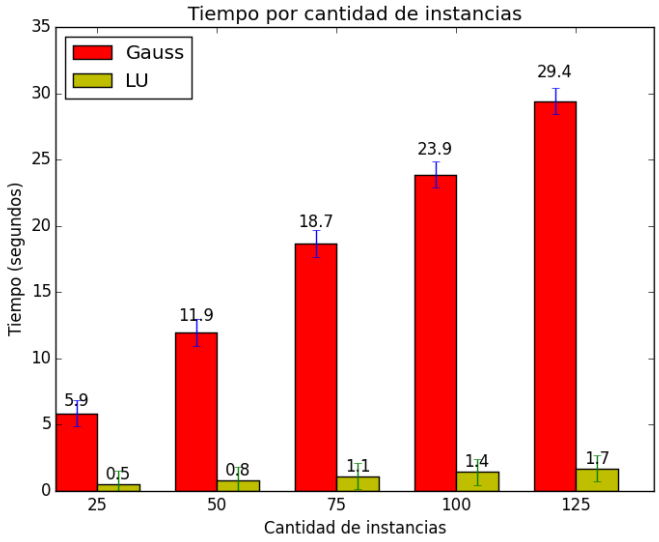

Figura 22: Tiempo tomado por la nuestra implementación optimizada de eliminación gaussiana y de factorización LU para resolver el problema para varias instancias de $n = m + 1 = 40$. La barra principal indica el promedio, y el segmento indica la desviación standard.

Como se observa en la Figura 22, confirmamos nuestras expectativas. Se observa, al igual que antes, que la implementación optimizada de Gauss crece linealmente con respecto a la cantidad de instancias del problema entradas, al igual que la implementación de la factorización LU, sólo que esta ultima lo hace con una pendiente mucho menor.

4. Conclusiones

Para empezar, queremos decir que creemos que este trabajo práctico fue fructífero. Pudimos implementar correctamente los métodos numéricos de resolución de sistemas lineales vistos en clase, lo cual nos ayudó mucho a entenderlos e internalizarlos mucho más.

Además fue muy positiva la experiencia de utilizar estos métodos para resolver un problema de la vida real. Este TP, por otro lado, nos permitió darnos un poco de idea de cuáles son las aplicaciones más usuales de los algoritmos vistos en la materia. Otra cosa notable que se desprende de este TP es lo importante que es abarcar un problema interdisciplinariamente: en este caso necesitamos conocer la ecuación de Laplace (física), saber que una derivada se puede aproximar con un cociente incremental con diferencias finitas (matemática) y utilizar métodos numéricos para resolver sistemas de ecuaciones lineales (computación).

En la misma linea, tomar tiempos de los algoritmos y comparar sus performances nos permitió corroborar lo que ya sabíamos en la teoría de manera práctica, además de ganar intuición sobre el comportamiento de los algoritmos.

Una cosa que nos gustaría decir, es que los experimentos que llegaron al TP sólo una pequeña proporción de todos los que realizamos. Muchos quedaron afuera porque estaban mal hechos (pocas instancias de prueba, bugs) o simplemente porque no mostraban exactamente lo que queríamos. De manera iterativa, llegamos a los experimentos que presentamos que, desde nuestro punto de vista, expresan perfectamente lo que queríamos que expresaran.

Para finalizar, nos gustaría plantear trabajo a futuro, y atar algunos cabos sueltos. Nuestra implementación *optimizada* de los algoritmos de eliminación gaussiana y factorización LU que aprovechaban el hecho de que la matriz es banda, solamente tenía una ganancia de performance con respecto al tiempo, dado que seguimos almacenando la matriz en un vector de vectores.

Por esa razón, es interesante plantearse el problema de como llevar a cabo esta optimización de la complejidad espacial de nuestros métodos. Algunas ideas en lo que respecta a la solución de este problema pueden hallarse en [GVL96, Cap. 4.3].

5. Apéndices

5.1. Demostración de la Proposición 1 del enunciado

Primero observemos que A es una matriz diagonal dominante no estricta. Para eso tenemos que ver que para cada fila el valor absoluto de la diagonal es mayor o igual que la norma-1 del resto de los elementos de esa fila. En nuestro caso puntual, esto significa ver que $|c_k| \geq |a_k| + 2|b_k| + |d_k|$[5].

Primero calculemos el lado derecho de la desigualdad:

$$\left|\frac{r_k - \Delta r}{r_k(\Delta r)^2}\right| + 2 \times \left|\frac{1}{r_k^2(\Delta\theta)^2}\right| + \left|\frac{1}{(\Delta r)^2}\right| = \frac{r_k - \Delta r}{r_k(\Delta r)^2} + 2 \times \frac{1}{r_k^2(\Delta\theta)^2} + \frac{1}{(\Delta r)^2} \quad (7)$$

$$= \frac{2r_k^2(\Delta\theta)^2 - (\Delta r)r_k(\Delta\theta)^2 + 2(\Delta r)^2}{r_k^2(\Delta r)^2(\Delta\theta)^2} \quad (8)$$

En (7) efectivamente podemos quitar los módulos pues todos los términos son positivos. Esto es evidente para b_k y c_k, aunque un poco menos para a_k. En efecto, $r_k - \Delta r$ podría llegar a ser negativo sólo en el caso que $k = 0$, pues $r_k = r_i + k \times \Delta r$. Pero para todos los $t_{0,j}$ no planteamos la ecuación 3 pues ya conocemos su valor, así que no afecta.

Entonces, queremos ver que $|b_k|$ es mayor o igual que (8), es decir

$$\frac{\left|-2r_k^2(\Delta\theta)^2 + (\Delta r)r_k(\Delta\theta)^2 - 2(\Delta r)^2\right|}{|r_k^2(\Delta r)^2(\Delta\theta)^2|} \geq \frac{2r_k^2(\Delta\theta)^2 - (\Delta r)r_k(\Delta\theta)^2 + 2(\Delta r)^2}{r_k^2(\Delta r)^2(\Delta\theta)^2}$$

Que es equivalente a

$$\left|-2r_k^2(\Delta\theta)^2 + (\Delta r)r_k(\Delta\theta)^2 - 2(\Delta r)^2\right| \geq 2r_k^2(\Delta\theta)^2 - (\Delta r)r_k(\Delta\theta)^2 + 2(\Delta r)^2$$

Supongamos que lo que está dentro del módulo es positivo, entonces tenemos

$$\begin{gathered} -2r_k^2(\Delta\theta)^2 + (\Delta r)r_k(\Delta\theta)^2 - 2(\Delta r)^2 \geq 2r_k^2(\Delta\theta)^2 - (\Delta r)r_k(\Delta\theta)^2 + 2(\Delta r)^2 \\ \Updownarrow \\ 2 \times (-2r_k^2(\Delta\theta)^2 + (\Delta r)r_k(\Delta\theta)^2 - 2(\Delta r)^2) \geq 0 \\ \Updownarrow \\ -2r_k^2(\Delta\theta)^2 + (\Delta r)r_k(\Delta\theta)^2 - 2(\Delta r)^2 \geq 0 \end{gathered}$$

Pero esta última desigualdad vale pues habíamos supuesto que efectivamente eso era positivo.

Ahora veamos que pasa si lo de adentro del módulo es negativo. Tenemos que

$$\begin{gathered} -2r_k^2(\Delta\theta)^2 + (\Delta r)r_k(\Delta\theta)^2 - 2(\Delta r)^2 \leq -2r_k^2(\Delta\theta)^2 + (\Delta r)r_k(\Delta\theta)^2 - 2(\Delta r)^2 \\ \Updownarrow \\ 0 \leq 0 \end{gathered}$$

Que vale trivialmente. Luego probamos que la matriz del sistema es diagonal dominante no estricta, si $|r_k - \Delta r| \geq 0$. Notar además que por la última cuenta no es cierto que sea diagonal dominante estricta.

Ahora, la demostración que sigue es similar a la demostración de la proposición que demuestra que diagonal dominante estricta implica que no aparecen ceros en la diagonal al hacer eliminación gaussiana. En esta demostración, simplemente se ve que aplicarle un paso de eliminación gaussiana a la matriz la deja diagonal superior.

Con una lógica muy parecida, podemos ver que la matriz de nuestro problema, luego de aplicarle un paso de eliminación gaussiana, queda diagonal dominante (no estricta). Además, la matriz queda banda, con las bandas exteriores siendo las mismas que antes. Esto se ve claramente, dado que cuando se modifica fila por fila,

[5] Los coeficientes están definidos en la sección 2.3

solo hay que modificar tantas filas como el ancho de la banda, y todas estas filas siguen teniendo la propiedad de tener todos valores iguales a 0 fuera de la banda, dado que restarle la fila a la que le estamos aplicando el paso de factorización gaussiana tiene todos ceros a partir de un coeficiente.

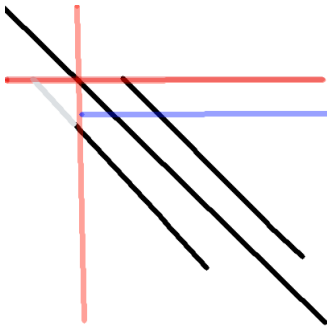

Figura 23: Representación de la matriz, la columna roja indica la columna en la que voy a poner ceros (a la izquierda hay todos ceros, ya que aplicamos eliminación gaussiana), y tengo que restarle la fila roja a la fila azul.

Luego, vimos que la matriz, al aplicarle un paso de eliminación gaussiana, queda diagonal dominante (no estricta) y banda, con las mismas bandas de siempre. Veamos entonces que la matriz tiene un elemento distinto de 0 en la diagonal, antes de aplicar el siguiente paso de eliminación gaussiana.

Como la fila tiene elementos distintos de 0 (en particular, el coeficiente determinado por la banda superior es siempre distinto de 0), y además la matriz es diagonal superior (no estricta), tenemos que

$$|a_{ii}^{(i)}| \geq \sum_{j \neq i} |a_{ij}^{(i)}| > 0$$

Entonces, $|a_{ii}^{(i)}| > 0$, que es lo que queríamos ver.

5.2. Links de animaciones

5.2.1. Comportamiento

Las animaciones generadas a partir de los experimentos se pueden encontrar en el siguiente link: http://bit.ly/1Uqt90x[6].

Estas animaciones fueron explicadas en sus respectivas secciones.

5.2.2. Performance

Para medir la performance de nuestros algoritmos sobre n instancias, lo que hicimos fue armar sistemas similares a los del video, es decir, que tengan una zona mas caliente que vaya rotando por todas los radios. Todos los experimentos de n instancias son variaciones de el de los videos.

- Temperaturas: https://www.youtube.com/watch?v=Bh3Yz_ZIcCs
- Isotermas: https://www.youtube.com/watch?v=BOs84T_7QY0

[6] Este link es un acortamiento de https://drive.google.com/folderview?id=0B5hNK09AHoDcfjZBSUZhNjhLRzBaQi1ESXY2WVduSVFES2FCMUF4OWVNTGZycWhUV1NtY0U&usp=drive_web, por comodidad

Referencias

[BF11] R. Burden y D. Faires. *Numerical Analysis*. Brooks/Cole, 2011.

[DB74] G. Dahlquist y A. Bjork. *Numerical Methods*. Prentice-Hall, 1974.

[GVL96] G. Golub y C. Van Loan. *Matrix Computations*. The John Hopkins University Press, 1996.