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

Descripcion del Tp

Eliminación gaussiana **factorización LU** **discretización** **key4**

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, aproximadoo las derivadas numéricamente utilizando idea del cociente incremental, podemos obtener un sistema de ecuaciones linales que describa el sistema. La formulación detallada de la formulación del sistema se encuentra en el desarrollo.

Para resolver estos sitemas, utilizaremos los dos métodos vistos en clase, eliminación Gaussiana y factorización LU. Además, 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 y la factorización LU, para luego poder contrastar su eficiencia ante distintas situaciones.
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, facilmente 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 $(r_{k+1}, t_{k+1,j})$. Para una explicación más detallada ver el apéndice ?????.

2.3. Armado del sistema de ecuaciones y su matriz asociada

Tenemos un sistema con $n \times (m + 1)$ incógnitas.

Primero contamos con n variables, los $t_{0,j}$ con $j = 0, 1, \ldots, n-1$ (es decir, las temperaturas interiores), cuyos valores conocemos 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

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

Vale destacar que 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 $t_{k,0}$.

De esto se desprende inmediatamente que el valor de cada $t_{k,j}$ ($1 \leq k < m$) depende exclusivamente de sus cuatro vecinos. Cabe mencionar también 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 (2) (respetando el orden en que aparecen): a_k (el coeficiente que acompaña a $t_{k-1,j}$), b_k (el coeficiente que acompaña a $t_{k,j-1}$ y $t_{k,j+1}$), c_k, d_k . Notar que por la observación anterior estos coeficientes sólo dependen del radio.

Pensamos ahora el orden que le daremos a los puntos de la discretización. Este será el orden en el que escribiremos 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.

A continuación presentamos el sistema de ecuaciones formado

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

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

2.4. Estructuración del código

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

El primero provee una representación para matrices con una interfaz conveniente. Utilizamos como representación interna un vector de vectores, junto con la cantidad de filas y columnas de la matriz. Las operaciones de Matriz permiten, dado un vector de términos independientes, obtener la solución de un sistema triangular superior sin ceros en la diagonal (o triangular inferior sin ceros en la diagonal) mediante *backward substitution* (respectivamente *forward substitution*), realizar eliminación gaussiana sin pivoteo (en caso de que sea posible), y, de existir, obtener la factorización LU de una matriz. Adicionalmente, hicimos otras versiones de eliminación gaussiana y factorización LU que requieren que la matriz en la que se aplican sea banda, y aprovechan este hecho para reducir la cantidad de operaciones básicas a realizar.

2.5. Experimentación

Pasemos a detallar los metodos 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 perimitió detectar un pequeño bug del código que no se hacia notar en los tests de la cátedra.

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

Para evaluar como varían 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 metodos, 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 distiniguían granularidad de ángulos y de radios, pero luego notamos que en nuestras implementaciones optimizadas, teóricamente, iba a ser importante diferenciar entre granularidad de ángulos y de radios, lo que finalmente sucedió. 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 permitiernos realizar muchos experimentos.

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

3. Resultados y discusión

3.1. Evaluación de los métodos

3.1.1. Único b

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

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 angulos, 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 naturales a la vista.

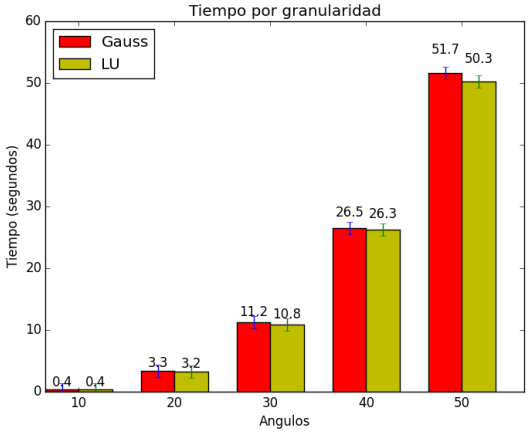

Figura 1: Tiempo tomado por la nuestra implementación de eliminación gaussiana y de factorización LU para resolver el problema. La granularidad de radios está fija en 40 y la de ángulos se indica en el eje x. La barra principal indica el promedio, y el segmento indica la desviación standard.

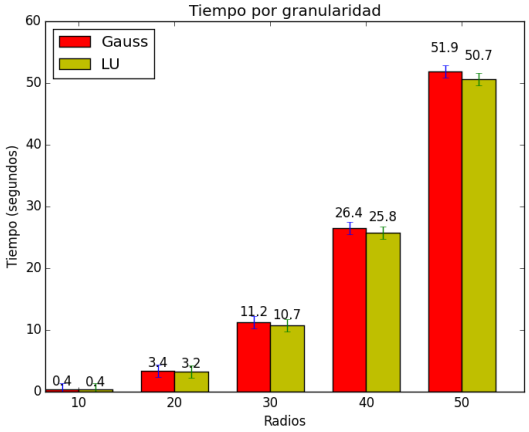

Figura 2: Tiempo tomado por la nuestra implementación de eliminación gaussiana y de factorización LU para resolver el problema. La granularidad de ángulos está fija en 40 y la de radios se indica en el eje x. La barra principal indica el promedio, y el segmento indica la desviación standard.

Como se ve en las figuras 1 y 2, 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 esperabamos, 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 3: Tiempo promedio tomado por las implementaciones para resolver un sistema con $n = m + 1 = 40$.

Con la Figura 3 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.

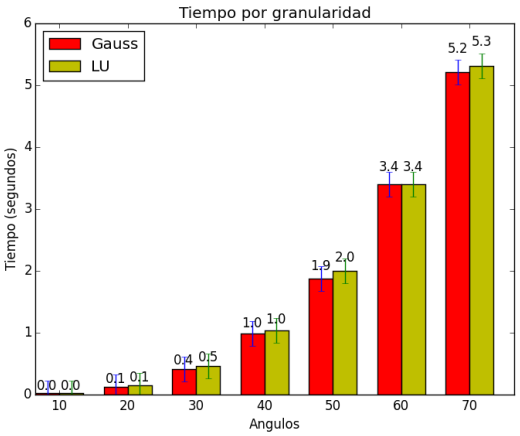

Figura 4: Tiempo tomado por la nuestra implementación optimizada de eliminación gaussiana y de factorización LU para resolver el problema. La granularidad de radios está fija en 40 y la de ángulos se indica en el eje x. La barra principal indica el promedio, y el segmento indica la desviación standard.

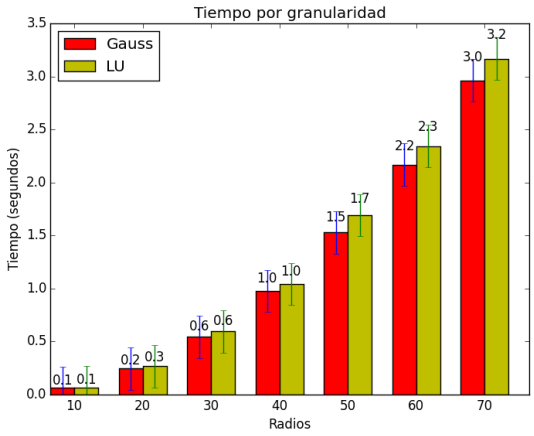

Figura 5: Tiempo tomado por la nuestra implementación optimizada de eliminación gaussiana y de factorización LU para resolver el problema. La granularidad de ángulos está fija en 40 y la de radios se indica en el eje x. La barra principal indica el promedio, y el segmento indica la desviación standard.

Los resultados que se ven en las Figuras 4 y 5 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 $O(\text{radios}^3\text{angulos})$, 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.

3.1.2. 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.

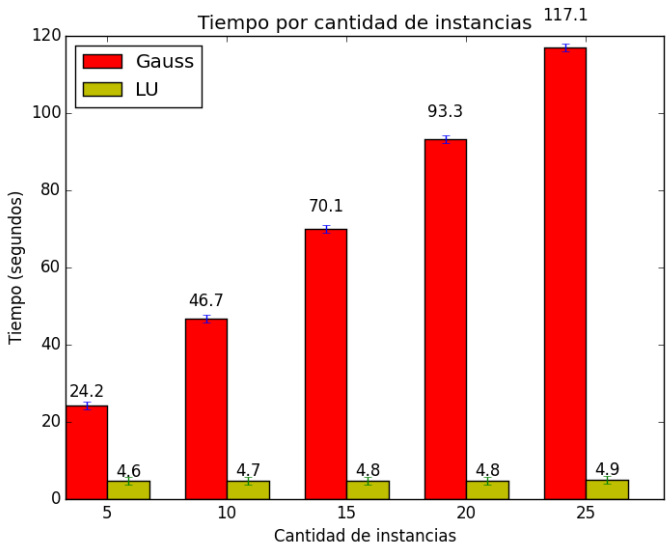

Figura 6: Tiempo tomado por la nuestra implementación de eliminación gaussiana y de factorización LU para resolver el problema para varias instancias de $n = m + 1 = 30$. La barra principal indica el promedio, y el segmento indica la desviación standard.

En los resultados se refleja lo que esperabamos. Como se ve en la figura 6, 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.

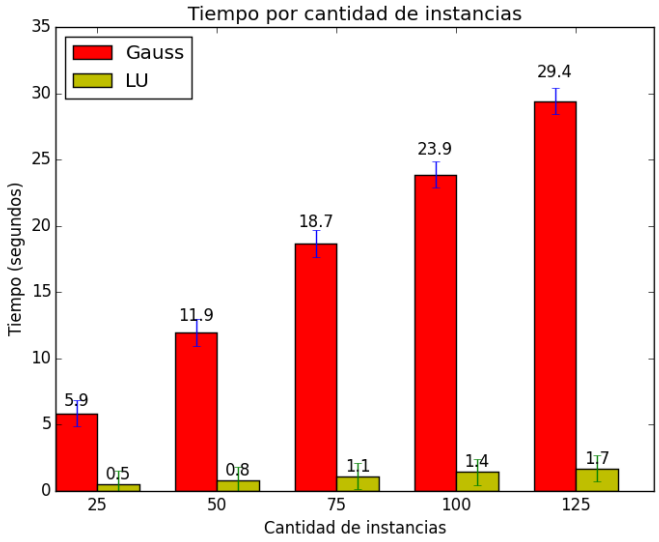

Figura 7: Tiempo tomado por la nuestra implementación optimizada de eliminación gaussiana y de factorización LU para resolver el problema para varias instancias de $n = m + 1 = 30$. La barra principal indica el promedio, y el segmento indica la desviación standard.

Como se observa en la Figura 7, 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

5. Apéndices

5.1. Demo

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 $|b_k| \geq |a_k| + 2|c_k| + |d_k|$.

Primero calculemos el lado derecho de la desigualdad:

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

Asumamos primero que $|r_k - \Delta r| \geq 0$. En ese caso la ecuación anterior es igual a

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

Entonces, queremos ver que $|b_k|$ es mayor o igual que eso, 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.