

Trabajo Práctico II

Matías Pérez (2/05)
Rodrigo Campos Catelin (561/06)

October 22, 2010

Abstract

El siguiente trabajo analiza las ventajas y desventajas de elegir un método iterativo y un método directo al resolver un sistema de ecuaciones lineales. Se usó como método directo la resolución mediante factorización LU y como método iterativo el método de Jacobis.

Como problema a resolver se escogió el cálculo del ranking de page sobre distintos dominios. Dicho problema tiene la característica de que la matriz que representa al sistema de ecuaciones suele ser rara.

Los aspectos tomados en cuenta para el análisis son: el uso de memoria, eficiencia en tiempo, aproximación del resultado y error de cálculo.

En este trabajo se muestra que el consumo de memoria y tiempo de ejecución del método iterativo es ordenes de magnitud menor al del método directo. Por esto, los métodos iterativos hacen posible la resolución de sistemas grandes que con un método directo resultan intratables.

Palabras clave: resolución de sistemas lineales, factorización LU, método de Jacobis, matriz rara.

1 Introducción teórica

Sea $A \in \mathbb{R}^{n \times n}$, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^n$. Consideremos el sistema: $Ax = b$.

1.1 Resolución mediante factorización LU

El método de resolución mediante factorización LU, consta en encontrar dos matrices L y U tales que:

- L sea Triangular Inferior con unos en la diagonal (invertible)
- U sea Triangular Superior
- $A = LU$

De esta forma se reescribe el sistema como: $Ax = LUx = b$, al ser L invertible se procede a resolver el sistema: $Ux = L^{-1}b$. Como U es Triangular Superior la resolución de este sistema es sencilla utilizando la técnica de *back substitution*.

Un método para lograr la factorización LU es el método de Gauss. Dicho método, si existe una factorización LU, la encuentra siempre.

Prop: si la matriz A es estrictamente diagonal dominante, se puede asegurar la existencia de L y U y su invertibilidad.

1.2 Resolución mediante el método de Jacobi

El método de Jacobi pertenece a la familia de métodos iterativos que descomponen a A en dos matrices M , K tal que $A = M - K$ con M invertible. Luego:

$$Ax = b$$

$$(M - K)x = b$$

$$Mx - Kx = b$$

$$x = M^{-1}Kx + M^{-1}b$$

Sea $T = M^{-1}K$, $c = M^{-1}b$. Entonces,

$$x = Tx + c$$

En base a esto se plantea el siguiente esquema recursivo:

$$x_{n+1} = Tx_n + c$$

Prop: si la sucesión $\{x_n\}$ converge $\Rightarrow \lim_{n \rightarrow \infty} x_n = x$.

Prop: si $\rho(T) < 1$, dicha sucesión converge.

Prop: sea $\|\cdot\|$ una norma matricial inducida $\Rightarrow \rho(T) < \|T\|$.

Corolario: sea $\|\cdot\|$ una norma matricial inducida, si $\|T\| < 1$, dicha sucesión converge.

En el método de jacobi se considera $M = D$ y $K = L + U$, donde:

$$\bullet d_{ij} = \begin{cases} a_{ii} & i = j \\ 0 & i \neq j \end{cases}$$

$$\bullet l_{ij} = \begin{cases} a_{ij} & i > j \\ 0 & i \leq j \end{cases}$$

$$\bullet u_{ij} = \begin{cases} a_{ij} & i < j \\ 0 & i \geq j \end{cases}$$

2 Desarrollo

2.1 Problema

Sea $W \in \mathbb{R}^{n \times n}$ una matriz rala donde $w_{ii} = 0$ y $w_{ij} \in \{0, 1\}, i \neq j$. Sea $p = 0.85$ una probabilidad. Sea $c_j = \sum_i w_{ij}$.

Se define $A \in \mathbb{R}^{n \times n}$ como:

$$a_{ij} = \begin{cases} \frac{1-p}{n} + \frac{pw_{ij}}{c_j} & c_j \neq 0 \\ \frac{1}{n} & c_j = 0 \end{cases}$$

El problema consiste en resolver el siguiente sistema:

$$Ax = x$$

que cumple que $x_i \geq 0$ y $\sum_i x_i = 1$

Es importante notar que la matriz A puede reescribirse como:

$$A = pWD + ez^\top$$

donde D es una matriz diagonal de la forma:

$$d_{jj} = \begin{cases} \frac{1}{c_j} & c_j \neq 0 \\ 0 & \text{sino} \end{cases}$$

e es un vector columna de unos de dimensión n y $z \in \mathbb{R}^n$

$$z_j = \begin{cases} \frac{1-p}{n} & c_j \neq 0 \\ \frac{1}{n} & \text{sino} \end{cases}$$

Luego, el sistema:

$$Ax = x$$

$$(pWD + ez^\top)x = x$$

$$pWDx + ez^\top x = x$$

$$pWDx - x = -ez^\top x$$

$$x - pWDx = ez^\top x$$

$$(I - pWD)x = ez^\top x$$

Se define $\gamma = z^\top x \in \mathbb{R}$. De esta manera, el procedimiento sugerido por la cátedra para resolver el sistema es:

1. Suponer $\gamma = 1$
2. Resolver la ecuación $(I - pWD)x = e\gamma$
3. Normalizar el vector x de manera que $\sum_{i=1}^n x_i = 1$

Propiedad: $\forall j \in \mathbb{N}, 1 \leq j \leq n, \sum_{i=1}^n (WD)_{ij} = 1$

Demostración: Sea $j \in \mathbb{N}, 1 \leq j \leq n$

$$\sum_{i=1}^n (WD)_{ij} = \sum_{i=1}^n \frac{w_{ij}}{c_j} = \frac{1}{c_j} \sum_{i=1}^n w_{ij} = \frac{1}{\sum_{i=1}^n w_{ij}} \sum_{i=1}^n w_{ij} = \frac{\sum_{i=1}^n w_{ij}}{\sum_{i=1}^n w_{ij}} = 1$$

Corolario: $\forall j \in \mathbb{N}, 1 \leq j \leq n, \sum_{i=1}^n (pWD)_{ij} = 0.85$

Corolario: $\forall j \in \mathbb{N}, 1 \leq j \leq n, \sum_{i=1}^n |(pWD)_{ij}| = 0.85$. Pues $(pWD)_{ij} \geq 0$.

2.2 Resolución mediante factorización LU

Existencia de factorización LU

Mostraremos que la matriz $(I - pWD)$ es estrictamente diagonal dominante. Es decir que $\forall j \in \mathbb{N}, 1 \leq j \leq n$, vale:

$$|(I - pWD)_{jj}| > \sum_{i=1, i \neq j}^n |(I - pWD)_{ij}|$$

Primero veamos que como $p(WD)_{jj} = 0$

$$|(I - pWD)_{jj}| = |I_{jj}| = 1$$

Por otro lado, dado que $i \neq j \Rightarrow I_{ij} = 0$

$$\sum_{i=1, i \neq j}^n |(I - pWD)_{ij}| = \sum_{i=1, i \neq j}^n |I_{ij} - (pWD)_{ij}| = \sum_{i=1, i \neq j}^n |(pWD)_{ij}| = 0.85$$

En consecuencia

$$|(I - pWD)_{jj}| > \sum_{i=1, i \neq j}^n |(pWD)_{ij}|$$

Luego, la matriz $(I - pWD)$ es estrictamente diagonal dominante. Lo que implica, por la propiedad antes expuesta, que existe una factorización LU.

Implementación

Se utilizó el método de eliminación Gaussiana para hallar U y $M^{-1}b$. Mediante el mismo se logró conseguir un sistema de ecuaciones equivalente cuya matriz asociada es triangular superior. Finalmente, para hallar el vector solución de este sistema equivalente, se utilizó la técnica de backsubstitution.

Para resolver el sistema, se agregó a la matriz el vector “b” como última columna. De esta forma, al triangular la matriz, en cada paso se tiene un sistema equivalente a anterior. Obteniendo al finalizar la matriz U y el vector $M^{-1}b$ buscados. El resto de la implementación no trajo mayores complicaciones.

El método de eliminación Gaussiana tiene la particularidad de que si la matriz es rala no conserva esta propiedad. Por este motivo, se decidió representar la matriz como filas contiguas en memoria, guardando todos sus elementos. Esto resulta en un consumo de memoria $O(n^2)$ para la matriz. Vale recordar que el orden del método es $O(n^3)$.

También se decidió utilizar pivoteo parcial para reducir el error numérico, aunque, como ya se mostró en secciones anteriores, esto no era necesario dado que la matriz tiene factorización LU .

2.3 Resolución mediante el método de Jacobi

Convergencia del método

Para ver la convergencia del método es suficiente ver que $\|T\| < 1$, para $\|\cdot\|$ norma matricial inducida y $T = D^{-1}(L + U)$. Por lo visto en la introducción teórica.

Tomaremos $\|\cdot\|_1$. En resumen, queremos ver que:

$$\|I^{-1}(-pWD)\|_1 < 1$$

$$\|I^{-1}(-pWD)\|_1 = |-1| \|I^{-1}pWD\|_1 = \|pIWD\|_1$$

$$\|pWD\|_1 = \max_j \sum_{i=1}^n |(pWD)_{ij}| = 0.85 < 1$$

De esta manera queda asegurada la convergencia del método.

Implementación

En contraposición con el método de eliminación Gaussiana, este método no modifica la matriz. Como consecuencia inmediata, si la matriz es rala, lo será en toda la ejecución del algoritmo. Para aprovechar esto y disminuir el requerimiento de memoria, a la vez que se reducen la cantidad de operaciones, se decidió representar solo los elementos no nulos de la matriz. A su vez, dado que la diagonal de la matriz consta de unos, ésta tampoco se incluyó.

Para lograr dicha representación se decidió usar 2 diccionarios anidados. El exterior es un diccionario cuyas claves son las columnas no nulas, y sus

valores son la representación de la columna correspondiente. Dichas columnas se representan mediante el diccionario interior. Éste tiene como claves las filas no nulas de esta columna y como valor, el elemento correspondiente a esa fila y columna.

El algoritmo, por su naturaleza iterativa, en cada iteración genera una solución que eventualmente será la solución del sistema. Cada iteración del algoritmo consta de, a partir de la matriz y la solución anterior, generar la siguiente de manera tal que el elemento i del vector generado es el que resuelve la siguiente ecuación:

$$a_{ii}X_i + \sum_{j=1, j \neq i}^n a_{ij}Y_j = b_i$$

siendo X_i la incógnita y Y la solución generada en la iteración anterior. Quedando finalmente,

$$X_i = \frac{b_i - \sum_{j=1, j \neq i}^n a_{ij}Y_j}{a_{ii}}$$

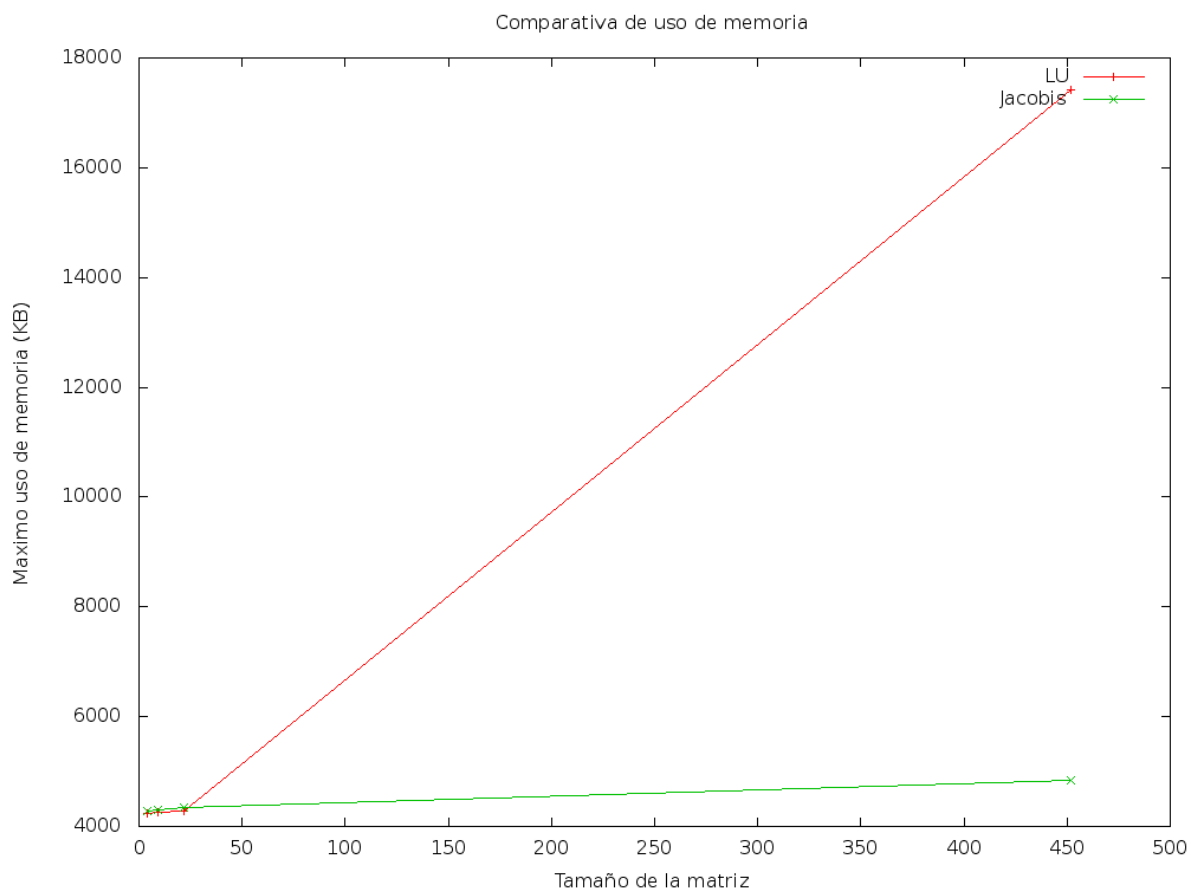
Se sabe que b es un vector de unos y $a_{ii} = 1$ para todo i . Entonces queda:

$$X_i = 1 - \sum_{j=1, j \neq i}^n a_{ij}Y_j$$

Para que el algoritmo sea eficiente, se recorre el vector Y una sola vez, acumulando en cada paso, en la correspondiente coordenada de X_i , el valor de $-a_{ij}Y_j$, solo para los elementos a_{ij} no nulos.

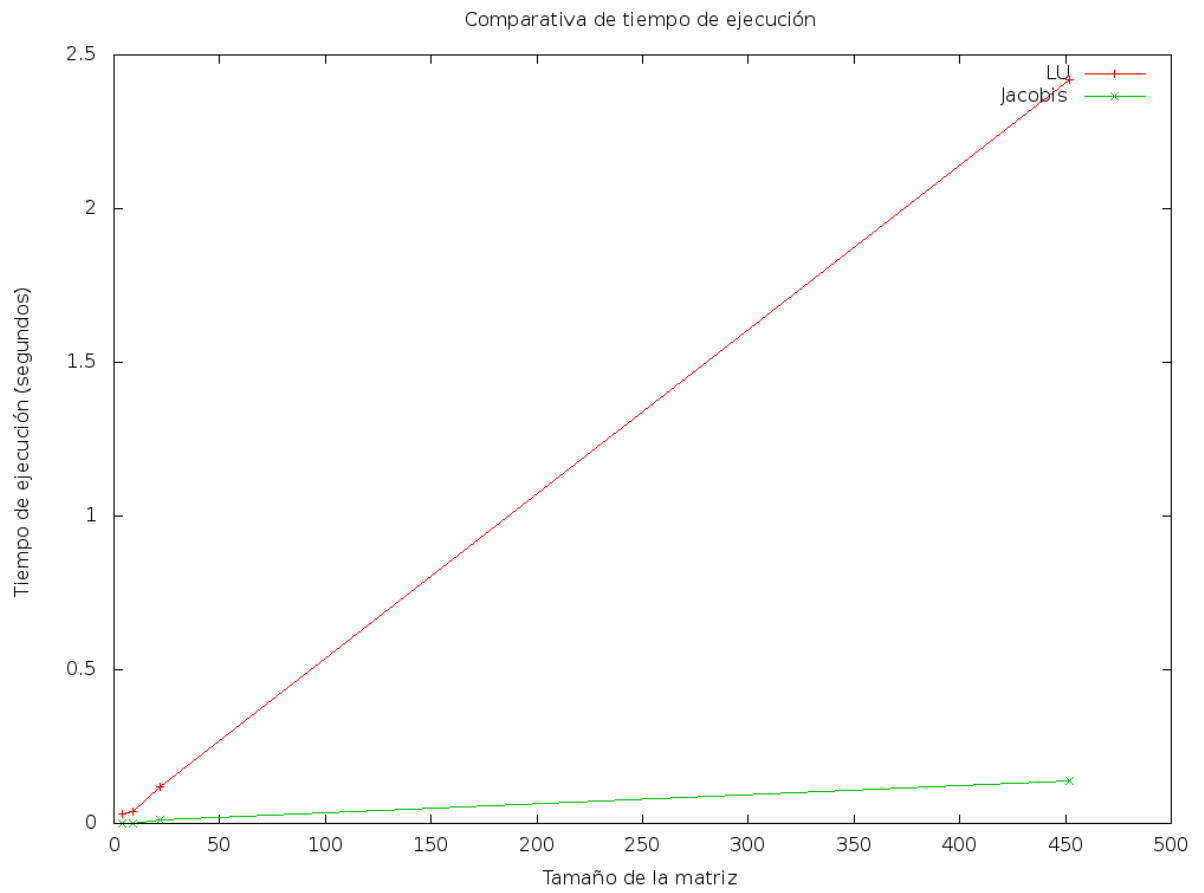
3 Resultados

Para analizar el comportamiento de los algoritmos y los metodos implementados, se realizaron distintos experimentos. A continuacion se incluyen gráficos sobre los comportamientos más relevantes.



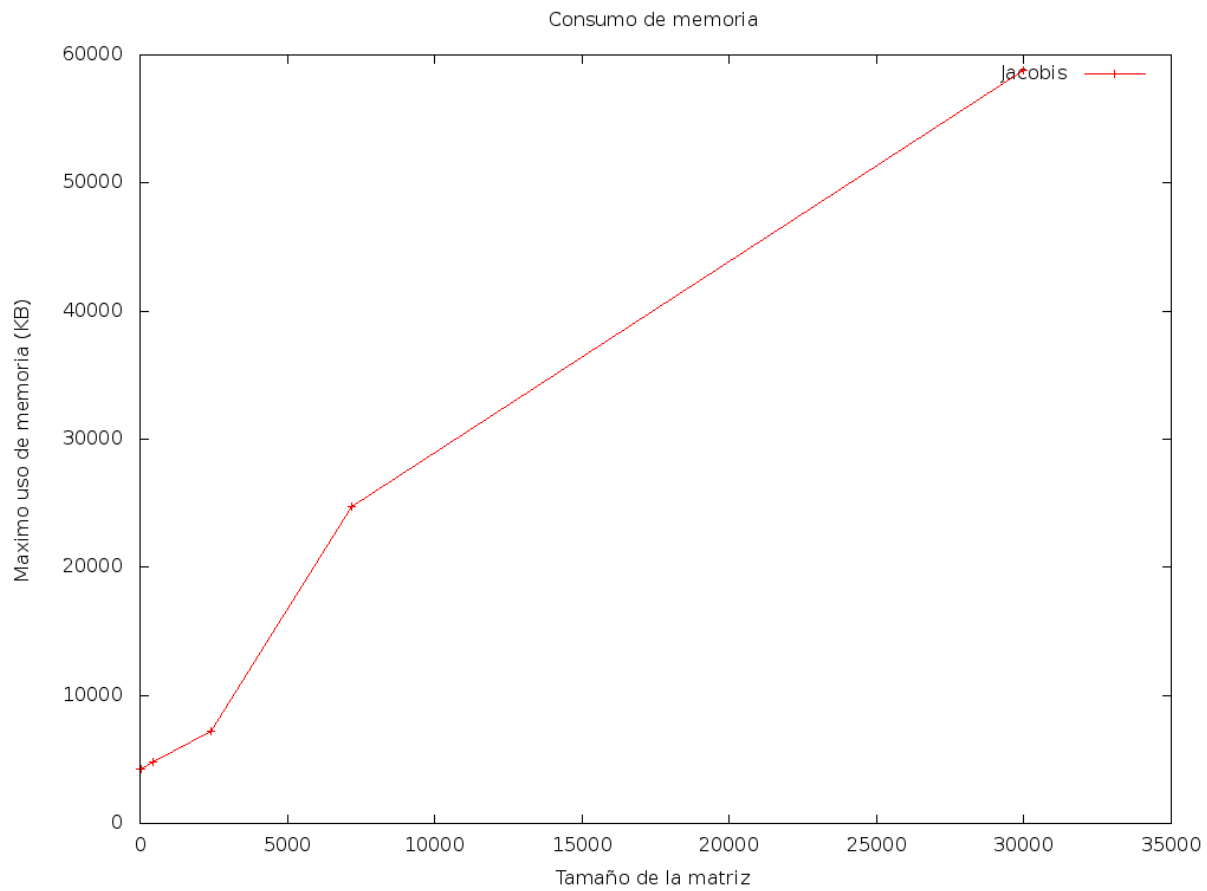
En este gráfico se puede observar el consumo máximo de memoria de ambos algoritmos al variar el tamaño de la matriz de entrada.

Para tomar las mediciones se utilizó el comando *time* de GNU con el parámetro *-f "%M"* que devuelve el consumo máximo de memoria utilizada por el programa en toda su ejecución.



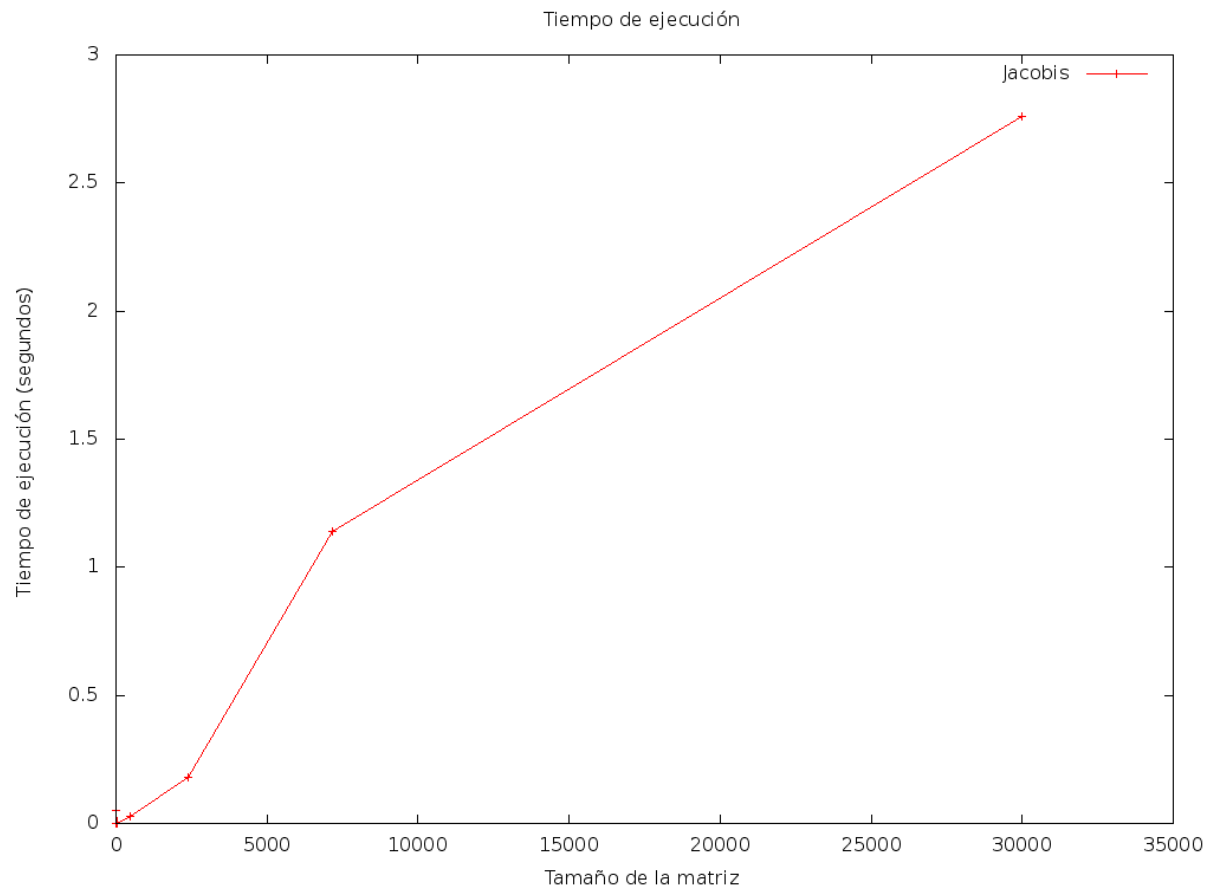
En este gráfico se puede observar cómo varía el tiempo de ejecución al cambiar el tamaño de la matriz de entrada.

Para tomar las mediciones se utilizó el comando *time* de GNU con el parámetro *-f "%e"* que devuelve el tiempo total de ejecución.



En este gráfico se puede observar cómo varía el consumo máximo de memoria del algoritmo de Jacobis en función del tamaño de la matriz para matrices grandes.

En este gráfico no se muestra el consumo de memoria del algoritmo LU ya que dicho consumo excede la cantidad de memoria disponible en las máquinas de prueba.



En este gráfico se muestra el tiempo de ejecución del algoritmo de Jacobis para matrices grandes. Se excluye el algoritmo de LU por lo mencionado en el caso anterior

4 Discusión

Luego de obtener los resultados presentados en la sección anterior, se pueden observar distintos aspectos tanto de los algoritmos como de los métodos. Los mismos serán discutidos a continuación.

4.1 Consumo de memoria

En el primer gráfico expuesto se puede ver como el consumo de memoria del método LU es ordenes de magnitud superior al del método de Jacobi. También se puede ver que para matrices pequeñas el consumo resulta similar, pero al aumentar el tamaño de la matriz, la diferencia aumenta en forma notable, debido a que el orden de consumo de memoria del algoritmo LU es cuadrático.

El orden del consumo de memoria del algoritmo de Jacobi también es cuadrático en peor caso. Pero este caso es muy difícil de encontrar en el dominio del problema ya que las matrices son típicamente ralas. Entonces, el consumo esperado de memoria se acercaría a un consumo de orden lineal en el tamaño de la matriz.

En el tercer gráfico expuesto se puede analizar el comportamiento del algoritmo de Jacobis para matrices más grandes que las soportadas por el algoritmo de LU . Se ve como si bien crecen los requerimientos de memoria, este crecimiento es aceptable en función del tamaño de la matriz.

4.2 Tiempo de ejecución

En el segundo gráfico expuesto se puede observar claramente que el tiempo de ejecución del método LU es, también, ordenes de magnitud superior al del método de Jacobi. A diferencia del anterior, para matrices pequeñas, ya se nota una diferencia no despreciable en el tiempo de ejecución. Siendo aún mayor para tamaños considerables de matrices.

En el quinto gráfico se puede ver como el algoritmo de Jacobi tiene una performance superior al la del algoritmo LU . Se puede ver que el crecimiento no es excesivo en función del tamaño de la matriz, permitiendo así trabajar con matrices más grandes que LU . En particular es interesante notar que el algoritmo de Jacobi tarda poco más de 2.5 segundos para procesar una matriz de 300000×30000 , mientras que el algoritmo LU tarda un tiempo similar (apenas menos de 2.5 segundos) para una matriz de 450×450 . Es decir, el algoritmo de Jacobis tarda un tiempo similar al algoritmo de LU para procesar una matriz dos ordenes de magnitud mas grande.

5 Conclusiones

Luego de analizar todos los resultados se llegó a la conclusión de que para matrices suficientemente grandes, los algoritmos exactos como el caso de LU resultan ser prohibitivos, ya sea tanto por el consumo de memoria que el mismo requiere, como por su performance temporal. Es por esto que los métodos iterativos resultan de gran importancia. Cuando se puede asegurar su convergencia, como es el caso de nuestro domino, se puede lograr resolver un sistema de ecuaciones que de otra forma sería imposible de resolver.

También vale decir que aunque se esté en un dominio en el que no sea posible asegurar su convergencia, un método iterativo puede ser una buena opción, ya que un método exacto puede resultar ser inutilizable para contextos con matrices suficientemente grandes, que suelen ser los casos interesantes de resolver. En estos casos un método iterativo puede llegar a ser a lo mejor que se pueda acceder. Y si bien quizá no se puede asegurar la convergencia del método, sí se puede conocer la norma del residuo $(b - Ax)$. Entonces pueden ser una excelente alternativa aún cuando no se puede asegurar su convergencia si, para ese caso, alcanza con acotar el residuo.

Los algoritmos iterativos resultan una alternativa performante, tanto en memoria como en tiempo, a los algoritmos exactos tradicionales y hacen posible trabajar con matrices que son intratables con un método directo.