DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

TP I - Métodos Numéricos

Métodos Numéricos

Grupo 2

Integrante	LU	Correo electrónico
Alejandro Danós Franco	381/10	adp007@msn.com
Fernando Ana	56/09	yolibertino@gmail.com

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

Índice

1. Introducción Teórica

El presente informe se enfoca la resolución de sistemas matriciales del tipo $Ax = b$, donde A es una matriz inversible con coeficientes reales. El método clásico por excelencia para este tipo de problemas es la eliminación gaussiana que basicamente consiste en aplicar operaciones elementales de fila sobre la matriz A y el vector b para poder simplificar el sistema de ecuaciones original, obténiendose un sistema triangulado que puede ser fácilmente resuelto sustituyendo de manera correcta las incógnitas del sistema. La eliminación gaussiana original puede (y en algunas ocasiones debe) realizar permutaciones de filas, pero como vamos a restringir nuestro estudio a matrices diagonal dominantes (para más información, ver demostración en el apéndice), se omitirá el pivoteo, y cada vez que se haga mención a dicho algoritmo se entenderá que es sin pivoteo.

El otro método estudiado en este informe es la factorización LU, que básicamente consiste en hallar una forma de A que sea igual a $L \cdot U$, donde U es triangular superior y L triangular inferior, de esta manera $Ax = b$ se convierte en $LUx = b$, y si consideramos $y = Ux$, la solución al sistema puede hallarse resolviendo primero $Ly = b$, y luego $Ux = y$. Esta factorización permite evitar tener que triangular la matriz A, cada vez que b es modificado. La desventaja de este método es que la matriz debe poder triangularse usando eliminación gaussiana sin pivoteo, pero afortunadamente en este informe trabajaremos con dichas matrices.

2. Desarrollo

Decidimos pensar al problema como un sistema lineal de ecuaciones o, equivalentemente, buscar el vector x que cumpla $Ax = b$, siendo éstas las siguientes:

- Matriz A: es una matriz cuadrada con cantidad de filas y de columnas igual a $n \times (m+1)$ está dividida en 3 partes según las filas. Sean i,j tal que $1 \leq i, j \leq (n \times (m+1))$.

  - **Caso** $i \leq n$ **ó Caso** $(n \times (m+1)) - n < i$:

$$A_{ij} = \begin{cases} 1 & \text{si } i = j; \\ 0 & \text{si } i \neq j. \end{cases}$$

  - **Caso** $n < i \leq (n \times (m+1)) - n$:

$$A_{ij} = \begin{cases} \frac{-2}{(\Delta r)^2} + \frac{1}{r \times \Delta r} - \frac{2}{r^2 \times (\Delta\theta)^2} & \text{si } i = j; \\ \frac{1}{(\Delta r)^2} - \frac{1}{r \times (\Delta r)} & \text{si } j = i - n; \\ \frac{1}{(\Delta r)^2} & \text{si } j = i + n; \\ \frac{1}{r^2 \times (\Delta\theta)^2} & \text{si } j = i - 1; \\ \frac{1}{r^2 \times (\Delta\theta)^2} & \text{si } j = i + 1; \\ 0 & \text{en otro caso.} \end{cases}$$

- Vector x: es un vector con $n \times (m+1)$ incógnitas que representarían las temperaturas de los puntos en nuestra pared. Para que sea más fácil el cálculo y que sea consistente con lo propuesto en la matriz A, están ordenados de forma *alfabética* primero según el radio (r) y después según el ángulo (θ). Es decir, X_1 representa a T(1,1), X_p representa a T(p/n,p %n), X_{n+1} a T((p+1)/n,(p+1) %n), etc.

- Vector b: es un vector con $n \times (m+1)$ valores que representan lo que sabemos sobre las temperaturas, es decir, las temperaturas internas y externas y el resultado de las ecuaciones de calor. Los primeros y últimos n valores son las temperaturas internas y externas respectivamente. Los puntos intermedios entre ellos son todos 0s, el resultado de la ecuación de calor para ese punto. De esta forma, queda subdividido en las siguientes partes:

- submatrices inducidas de A en las que hay una matriz **Identidad**, los 2 casos en la primera definición de A_ij arriba, se igualaría el respectivo X_i con su temperatura fija. En los puntos de la *submatriz inducida de A* en los que no hay una Matriz **Identidad**, están igualados a 0 para aplicar la ecuación con derivadas con los multiplicadores de las incógnitas debidamente indicados por cada fila.

Menos formalmente, sean $M_{i,j}, M_{i,i-n}, M_{i,i+n}, M_{i,i-1}$ y $M_{i,i+1}$ los multiplicadores en las filas de A "del medio" respectivamente según los enunciamos.

$$\left(\begin{array}{cccccccc|c}
\mathrm{I} & \cdots & 0 & \cdots & 0 & \cdots & 0 & \cdots \\
\vdots & \ddots & & \cdots & 0 & \cdots & 0 & \cdots \\
\hline
\vdots & \vdots & & \vdots & & \vdots & & \vdots \\
\vdots & M_{i,j-n} & \cdots & M_{i,j-1} & M_{i,j} & M_{i,j+1} & \cdots & M_{i,j+n} & \cdots \\
\vdots & \vdots & & \vdots & & \vdots & & \vdots \\
\hline
\cdots & \cdots & 0 & \cdots & 0 & \cdots & \mathrm{I} & \cdots \\
\cdots & \cdots & 0 & \cdots & & \cdots & \vdots & \ddots
\end{array}\right)$$

2.1. Algoritmos

El objetivo principal del presente trabajo es resolver sistemas matriciales de la forma $Ax = b$, para el caso en que A sea una matriz inversible y diagonal dominante. Para poder resolver un sistema de ecuaciones en forma matricial, lo esencial es triangular la matriz para transformar el sistema, en principio complejo, en uno más simple que pueda ser resuelto mediante algún algoritmo sencillo. Los métodos elegidos y estudiados para la triangulación del sistema son el algoritmo de eliminación de Gauss-Jordan sin pivoteo y la factorización LU, mientras que para resolver el sistema triangulado se usaron *backward* y *forward substitution*. A continuación se muestran los pseudocódigos de los algoritmos implementados y la resolución de los sistemas de ecuaciones.

Algoritmo 2.1 gauss(Matriz A, vector b)

```
for i = 1 hasta n do
  for j = i + 1 hasta n − 1 do
    if A_ji ! = 0 then
      m = A_ji / A_ii
      for k = i hasta n do
        A_jk = A_jk − m · A_ik
      end for
      b_j = b_j − m · b_i
    end if
  end for
end for
```

Algoritmo 2.2 LU(Matriz A)

```
for i = 1 hasta n do
  for j = i + 1 hasta n − 1 do
    if A_ji ! = 0 then
      m = A_ji / A_ii
      for k = i hasta n do
        A_jk = A_jk − m · A_ik
      end for
      A_ji = m
    end if
  end for
end for
```

Algoritmo 2.3 forwSubst(Matriz A, vector b, vector res, bool lu)

```
if lu then
  for i = 1 hasta n do
    auxVector = A_ii
    A_ii = 1
  end for
end if
for i = 1 hasta n do
  acum = 0
  for j = 1 hasta j < i do
    acum+ = res_j · A_ij
  end for
  res_i = (b_i − acum)/A_ii
end for
if lu then
  for i = 1 hasta n do
    A_ii = auxVector
  end for
end if
```

Algoritmo 2.4 backSubst(Matriz A, vector b, vector res, bool lu)

```
if lu then
  for i = 1 hasta n do
    auxVector = A_ii
    A_ii = 1
  end for
end if
for i = n hasta 1 do
  acum = 0
  for j = n hasta j > i do
    acum+ = res_j · A_ij
  end for
  res_i = (b_i − acum)/A_ii
end for
if lu then
  for i = 1 hasta n do
    A_ii = auxVector
  end for
end if
```

Algoritmo 2.5 resolverConGauss(Matriz A, vectores bes, vectores $reses$)

```
for i = 1 hasta #(bes) do
  gauss(A, bes_i)
  backSubst(A, bes_i, reses_i, false)
end for
```

Algoritmo 2.6 resolverConLU(Matriz A, vectores bes, vectores $reses$)

```
LU(A)
for i = 1 hasta #(bes) do
  backSubst(A, bes_i, aux, true)
  forwSubst(A, aux, reses_i, true)
end for
```

Aclaraciones:

- Como se puede observar en el pseudocódigo hay ciertas optimizaciones que no afectan a la correctitud de los algoritmos.
- El código implementado permite usar pivoteo parcial, pero no será utilizado ni detallado en el informe.

- La igualdad por cero está definida por la cercanía al cero de dicho número. Se usa una constante pequeña para decidir la igualdad.
- El pseudocódigo presnta abusos de notación y es una mezcla de varios lenguajes de programación y lenguaje natural.

DESCRIPCION TESTS Una vez detallado los algoritmos se pasó a la etapa de experimentación, realizandose los siguientes tests: ********** TEST UNO **********

2.2. Test de comparación Gauss vs LU

La siguiente experimentación tiene la intención de determinar las presuntas ventajas en determinadas condiciones de utilizar factorización LU en lugar del algoritmo de eliminación de Gauss-Jordan para hallar la (o las) solución (es) a un sistema de ecuaciones lineales. Para realizar dicha experimentación se generaron instancias aleatorias con las mismas semillas, utilizando los archivos (o modificaciones de los mismos) *genTest.py* y *test.sh*, y se compararon los tiempos de cómputo en función de la cantidad de puntos del sistema y la cantidad de instancias a resolver.

El tiempo de cómputo para cada algoritmo fue medido con los métodos provistos por la cátedra (ubicados en *time.h*). El tiempo de cómputo total para cada tamaño fue calculado considerando solo los métodos de los algoritmos en cuestión más los necesarios para la resolución del sistema (*backward* y *forward substitution*). Se omitió el tiempo de los métodos que plantean al sistema por no ser considerados parte de los algoritmos de resolución de sistemas matriciales. Todo esto puede apreciarse en *main.cpp*.

Se realizaron varias ejecuciones de la experimentación considerando como valor valor final el promedio de dichas ejecuciones. Cabe aclarar que si bien las instancias son "aleatorias", se usan las mismas tanto para LU como para Gauss porque se usa la misma semilla. La idea de esta experimentación es determinar si al cambiar las condiciones del entorno (o en términos más el teóricos el vector b del sistema $Ax = b$) en forma "continua", la factorización LU ahorra cálculos frente al método de eliminación de Gauss. Los tamaños de las matrices fueron fijados de manera que cubran el mayor espacio posible de instancias sin tener que caer en ejecuciones "eternas". Por otro lado, en todas las matrices la cantidad de radios y ángulos es la misma, para que los tiempos de cómputo sean lo más equilibrados posibles.

3. Resultados

A continuación mostramos los resultados obtenidos para el test de comparación entre factorización LU y eliminación gaussiana, los tiempos de cómputo se muestran en segundos. Se muestran los resultados de matrices de 2500x2500, 900x900 y 100x100.

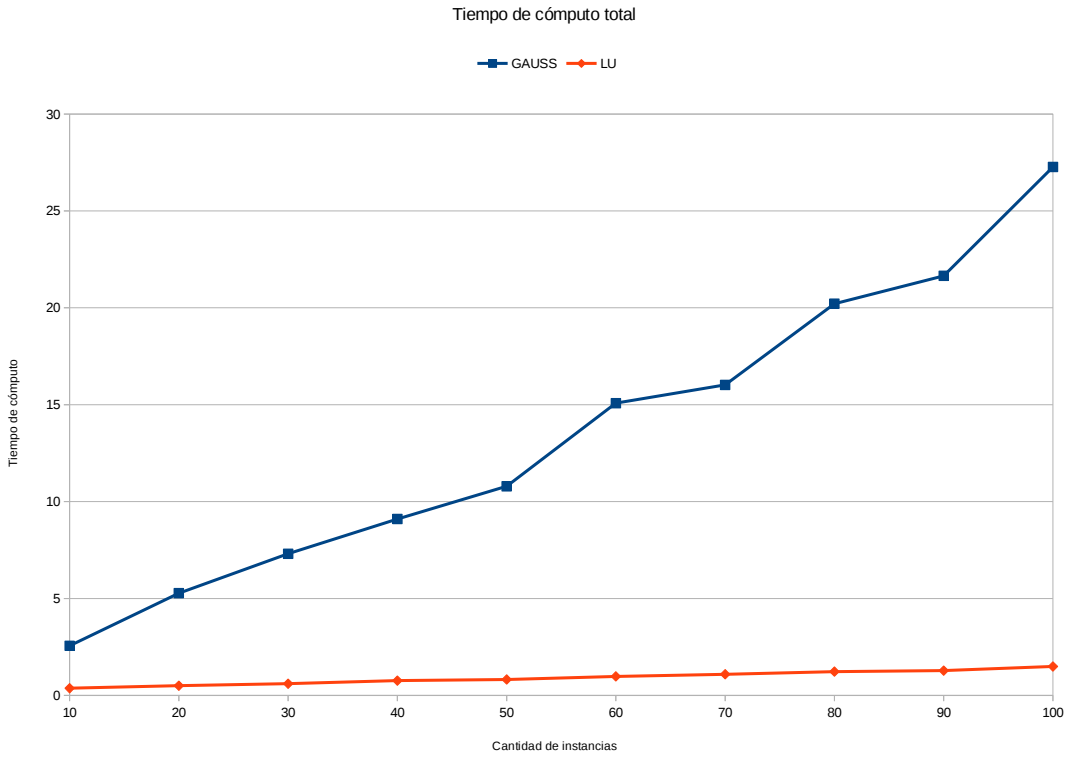

Figura 1: Resultados obtenidos usando matrices de 50 ángulos y 50 radios.

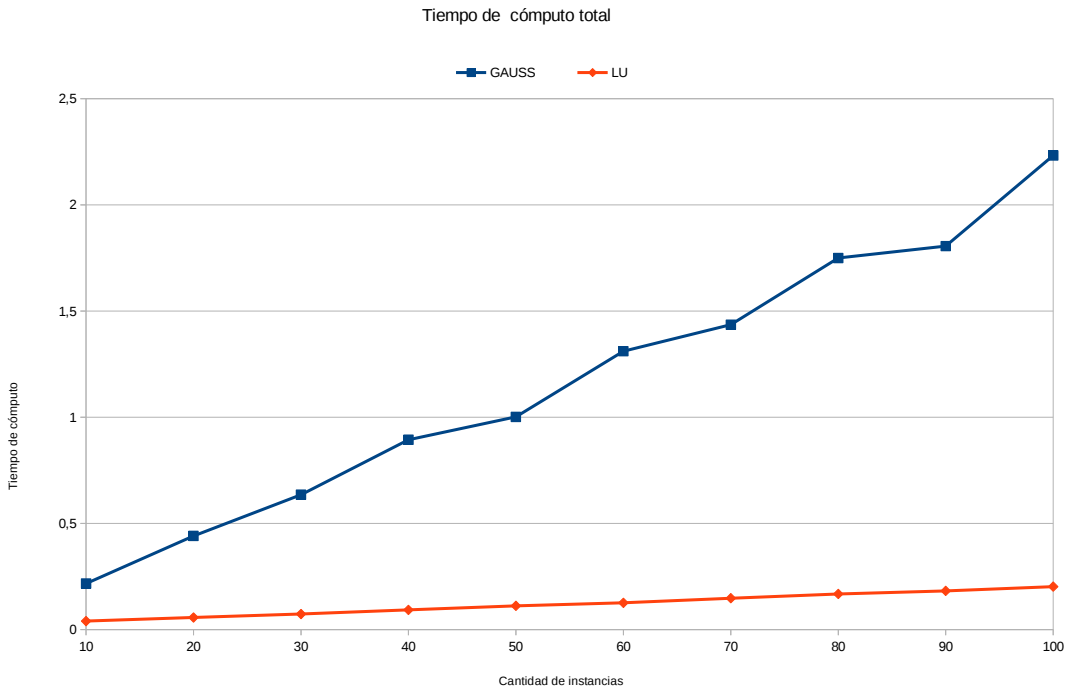

Figura 2: Resultados obtenidos usando matrices de 30 ángulos y 30 radios.

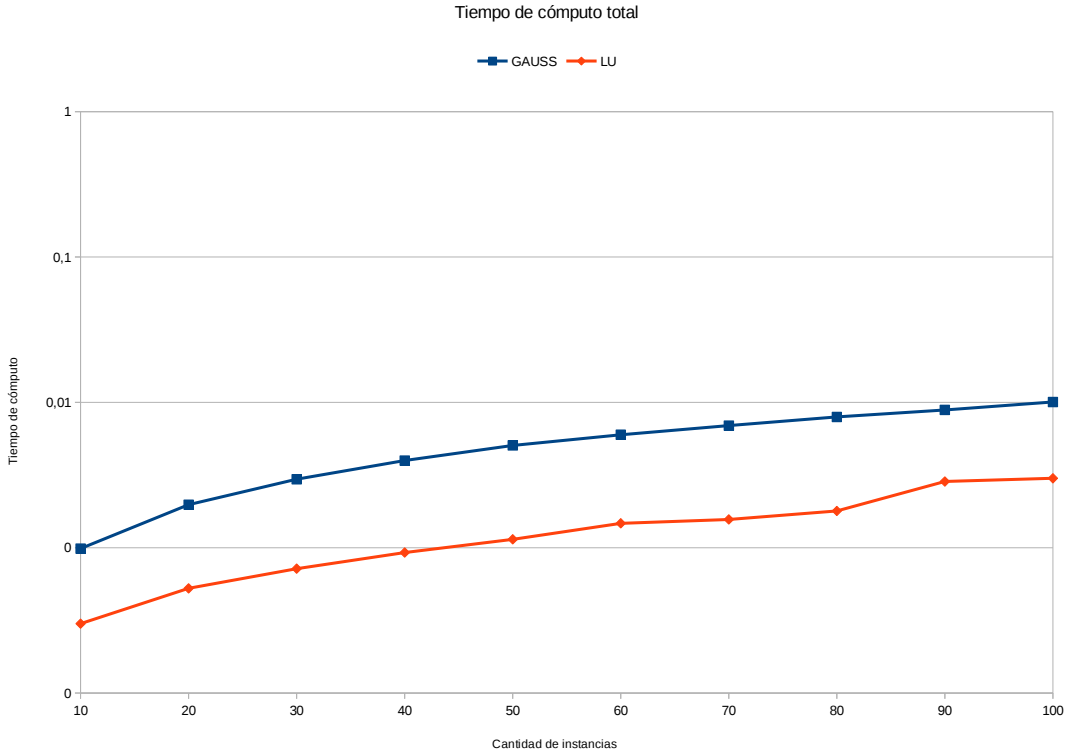

Figura 3: Resultados obtenidos usando matrices de 10 ángulos y 10 radios.

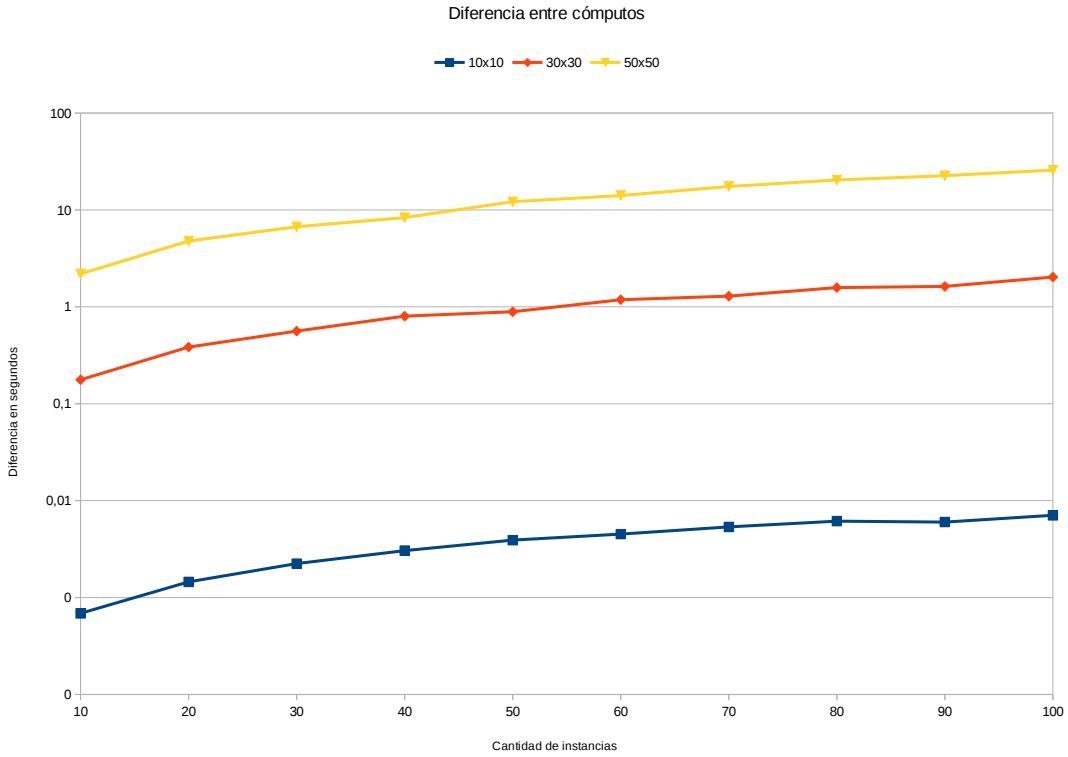

Figura 4: Diferencia entre el tiempo consumido por LU y Gauss de los gráficos anteriores. Uso de escala logarítmica.

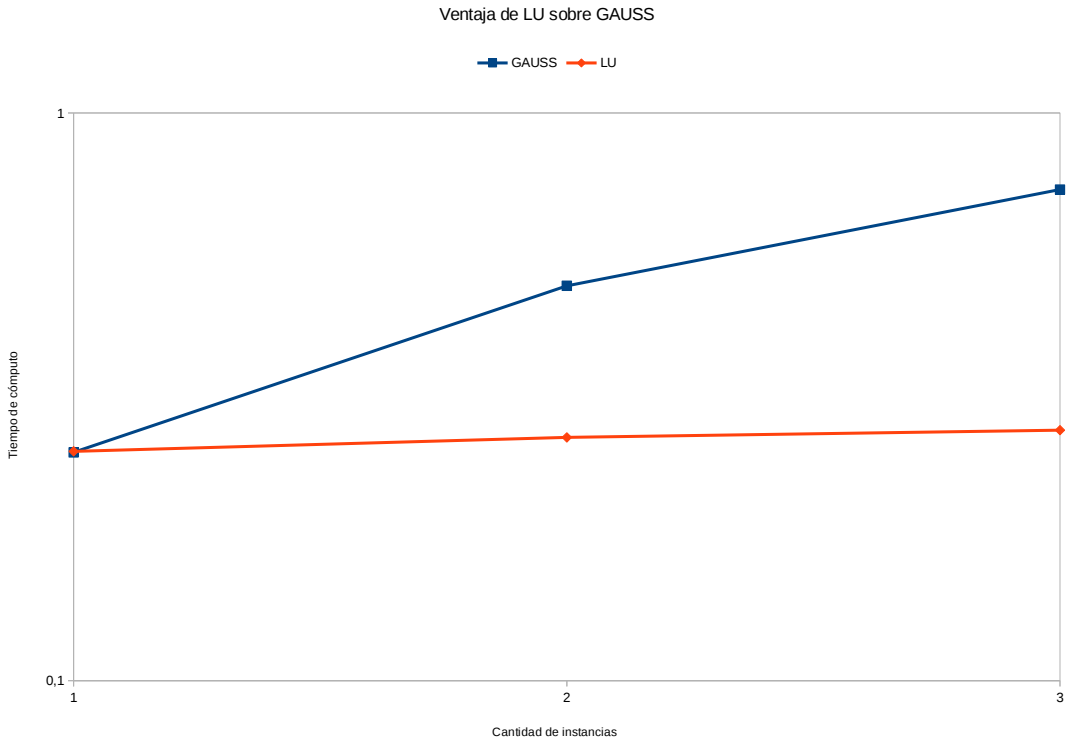

Figura 5: Tiempo consumido por Gauss y LU para instancias pequeñas.

4. Discusión

4.1. Análisis de comparación Gauss contra LU

Como se puede apreciar en los gráficos 1, 2 y 3, al variar la cantidad de instancias el tiempo de cómputo de la factorización LU fue muy inferior al tiempo de la eliminación gaussiana, y esta diferencia se hace sumamente notoria a medida que la cantidad de vectores b a calcular aumenta. En el gráfico 4 se puede observar como la diferencia es

proporcional tanto a la cantidad de instancias como al tamaño de las matrices. Por otro lado en 5 se puede determinar que basta con resolver el sistema de ecuaciones con sólo dos vectores b distintos para ya apreciar una ventaja efectiva sobre la eliminación gaussiana y que para el caso en que solo haya un sólo vector b, los tiempos de cómputo son prácticamente idénticos.

5. Conclusiones

6. Apéndices

6.0.1. Demostración de la proposición

Proposición:

Sea $A \in K^{n\times n}$ con $K = \mathbb{R}$ o $\mathbb{C}$ una matriz diagonal dominante e inversible entonces es posible aplicar eliminación gaussiana sin pivoteo.

Demostración:

Como A es diagonal dominante, se tiene que $|A_{ii}| \geq \sum_{j=1,i\neq j}^{n} |A_{ij}|$ para todo $j \neq i$. Supongamos que no se puede realizar eliminación gaussiana sin pivoteo, o lo que es lo mismo, existe un i tal que en un determinado paso de la eliminación gaussiana sucede que $A_{ii} = 0$, pero entonces por lo visto anteriormente se tiene que $0 = |A_{ii}| \geq |A_{ij}| \geq 0$ para todo $j \neq i$, por lo que $|A_{ij}| = 0$ para todo j de 1 a n y esto sucede si y solo si $A_{ij} = 0$, es decir, la fila i es nula y por lo tanto A no es inversible, lo que es un absurdo ya que A es inversible por hipótesis. El absurdo provino de suponer que no es posible utilizar eliminación gaussiana sin pivoteo sobre A, quedando demostrada la proposición.

7. Referencias