DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

# Métodos Numéricos

## Trabajo práctico 2

Tu cara me suena


***Resumen***

*En este trabajo estudiaremos algoritmos para resolver sistemas de ecuaciones para problemas reales. Se expondrán técnicas para obtener alguna isoterma buscada en un muro de concreto de un horno usando eliminación Gaussiana y Factorización LU. Se estudiará el rendimiento de ambos y las ventajas de cada uno. Se analizarán los efectos en el cambio de granularización del horno. Al final del trabajo, se llegarán a conclusiones sobre lo descubierto.*



| Integrante | LU | Correo electrónico |
|---|---|---|
| Danós, Alejandro Franco | 381/10 | adp007@gmail.com |
| Fernando | 56/09 | yolibertino@gmail.com |
| Ana Sarriés | 144/02 | abarloventos@gmail.com |



Palabras claves:

Eliminación Gaussiana. Factorización LU. Iosterma. Horno. Matrices. Sistemas de ecuaciones. Estudio temporal.

# Índice

# 1. Introducción Teórica

Dado un horno industrial de gran tamaño, se estudia en este trabajo la búsqueda de una isoterma dada en las paredes de éste. Dado que computacionalmente es imposible analizar el problema para dimensiones continuas para la pared, se lo discretiza según información dada. Esta información requerida del horno sería el radio en la parte interna de la pared, la cantidad de ángulos en el cual discretizarlo, el radio en la parte externa de la pared, la temperatura interna, la temperatura externa y la isoterma buscada. Se adjunta un gráfico para verlo con más claridad.

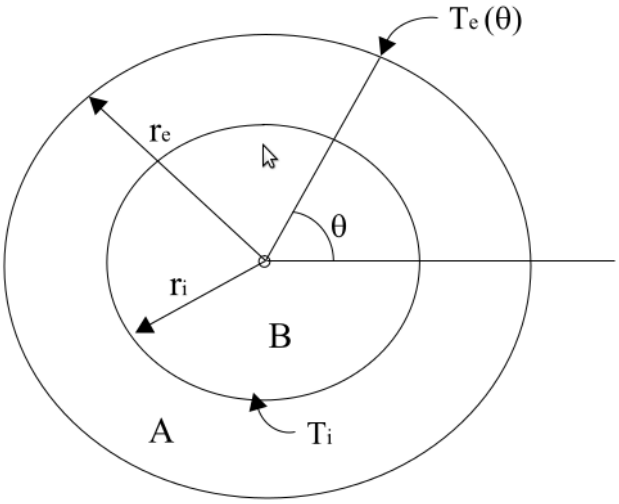


Figura 1: Sección circular del horno

Para lograr lo pedido, en nos enfocamos en 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 básicamente consiste en aplicar operaciones elementales de fila sobre la matriz $A$ y el vector $b$ para poder simplificar el sistema de ecuaciones original, obteniéndose un sistema triangulado que puede ser fácilmente resuelto aplicando un algoritmo de sustitución hacia atrás [1, 6.1]. 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 [1, 6.5].

# 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 ($\theta$). 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.

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.

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

## 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.

**Algorithm 1** gauss(Matriz $A$, vector $b$)

```
for i = 1 hasta n do
  if A_ji != 0 then
    for j = i + 1 hasta n do
      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 for
  end if
end for
```

**Algorithm 2** LU(Matriz $A$)

```
for i = 1 hasta n do
  for j = i + 1 hasta n 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
```

**Algorithm 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
```

**Algorithm 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**
    $acum+ = res_j \cdot A_{ij}$
  **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**

**Algorithm 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**

**Algorithm 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**

**Algorithm 7** calcularIsotermas(vector $temps$, int $isoterma$)

$radiosMasCerca$ = obetenerRadiosMasCercas($temps$, $isoterma$)
$isotermas$ ($\#angulos$)
**for** $i = 1$ hasta $\#angulos$ **do**
  $isotermas_i$ = promedio($radiosMasCerca_i$)
**end for**

Aclaraciones:

- El código implementado permite usar pivoteo parcial, pero no será utilizado ni detallado en el informe.
- La igualdad por cero está definida por tolerancia.
- El pseudocódigo presenta abusos de notación y es una mezcla de varios lenguajes de programación y lenguaje natural.
- Algunos algoritmos pueden estar implementados en un mismo método.

- La clase matriz es básicamente un vector de vectores, más un vector que va guardando rastro de las permutaciones. Como la eliminación gaussiana con pivoteo parcial no va a ser estudiada en el informe, ya que se privilegió la performance de los algoritmos, esta último vector puede ser omitido.

## 2.2. Tests

En esta sección haremos una introducción a los diferentes tests que decidimos hacer en nuestro trabajo.

En los tessts que respecta a calcular el tiempo de ejecución de algoritmos, éste 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.

### 2.2.1. Test de isoterma en función de la granularidad de la discretización

En nuestra primera experimentación, analizaremos cómo iría cambiando la posición de la isoterma en función de la granularidad de la discretización. Para ello, fijaremos a $r_i$, $r_e$, $T_i(\theta_j)$, $T_e(\theta_j)$ e *iso* para poder concentrarnos sólo en la isoterma[1] DEEEEEEEEEEEEEEEEEEECIRRRRRRRR CUÁAAAAALES SOOOOOOOON ESTOOOOS VALOOOOOORES Y POR QUÉEEEEEEEEEE. Otra decisión fue tomar a la cantidad de ángulos y la cantidad de radios iguales para cada instancia, es decir, $n = m$, para facilitar el test. Esto significaría que nuestra matriz será cuadrada con dimensiones $\mathbb{R}^{n*(n+1)\times n*(n+1)}$. Se podría testear en un futuro las implicancias de variar ambos al mismo tiempo. También para tener en cuenta es el hecho que la matriz es una *matriz banda* dado que en cada fila $i$ se tienen sólo 5 datos no nulos, y todos ellos entre las columnas $j - n$ y $j + n$. Esto último significaría que los datos a guardados por cada fila son mucho mayores a los realmente usados, por lo que en alguna futura expansión de este trabajo se podría adaptar el algoritmo a este hecho y disminuir su complejidad espacial rotundamente.

Finalmente, como las temperaturas externas e internas están fijas, no es necesario calcular la isoterma por cada ángulo dado que estarán siempre en el mismo radio para cada uno de ellos. Esto se debe a que entre los ángulos no hay diferencia, y según las propiedades del calor enunciadas por la cátedra éste se comportaría igual en cada uno de los ángulos.

ACÁ PONER LOS VALORES ELEGIDOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO

### 2.2.2. 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 principalmente, la cantidad de instancias a resolver.

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

[1] Los valores están especificados abajo y fueron elegidos luego de varios tests

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 ya que en este experimentación no interesa mostrar comoa afecta la discretización al tiempo de cómputo.

### 2.2.3. Test de isoterma en función de las condiciones de borde

La siguiente experimentación tiene por objetivo principal analizar cómo se comporta el modelo implementado. Es decir, ver cómo refleja los cambios esperados en un situación dada del problema modelado. En un alto horno real $r_i$ y $r_e$ no sufren cambios, lo que varía constantente son las $T_i(\theta_j)$ y las $T_e(\theta_j)$. Para simular esto y poder ver cómo varía la isoterma elegimos una buena discretización, la mayor que podamos costear en función de nuestra potencia de procesamiento, mantenemos $r_i$ y $r_e$ constantes y vamos variando las $T_i(\theta_j)$ y las $T_e(\theta_j)$ de diferentes formas para ver cómo se comporta la isoterma. Los casos que vamos a considerar son los siguientes:

1. aumenta $T_i(\theta_j)$ y $T_e(\theta_j)$ para todo $j$ de la discretización.
2. aumenta $T_i(\theta_j)$ y se mantienen constante $T_e(\theta_j)$ para todo $j$ de la discretización.
3. aumenta $T_i(\theta_j)$ y desciende $T_e(\theta_j)$ para todo $j$ de la discretización.

Elegimos estos tres casos porque creemos que cubren la mayoría de los comportamientos. A nivel teórico no hacemos distinción entre las $T_i(\theta_j)$ y las $T_e(\theta_j)$, por lo tanto tenemos dos componentes que pueden cambiar en un sentido u otro o mantenerse constantes. El caso 1 tomas que las dos componentes cambian y en el mismo sentido. El caso 2 que una componente cambia y la otra permanece constante. Finalmente, el caso 3 considera la situación en la que las dos componentes cambian pero en sentidos inversos.

### 2.2.4. Test de tiempo en función de la granularidad de la discretización

La siguiente experimentación tiene por objetivo analizar cómo y cuánto influye la granularidad de la discretización en el tiempo de ejecución de nuestros algoritmos. Sólo utilizamos el método de Gauss-Jordan ya que no nos interesa comparar Gauss-Jordan con LU, hay otro test específico para eso.

Para aislar la componente granularidad decidimos tomar $r_i$, $r_e$, $T_i(\theta_j)$, $T_e(\theta_j)$ e $iso$ constantes. Lo único que varía entre las instancias del test es la cantidad de ángulos y radios en la que discretizamos el horno. Tomamos cantidades de ángulos y radios siempre iguales porque sólo nos interesa la granularidad como cantidad de puntos. Es decir, que la cantidad de puntos en nuestra discretización va a ser:

$$puntos = m^2 = n^2 \tag{1}$$

Nuestro objetivo es comprobar que la complejidad temporal empírica es la esperada por un algoritmo de eliminación Gaussiana de $O(n^3)$. Además, dado que es un algoritmo en el cual los ciclos son siempre exhaustivos[2], no esperamos saltos demasiado agitantados entre un tiempo y el siguiente.

[2]Si miramos el pseudocódigo del algoritmo en la sección desarrollo, veremos que siempre entra a los *fors* y no sale hasta iterar todos los $i$.

# 3. Resultados

## 3.1. Test de isoterma en función de la granularidad de la discretización

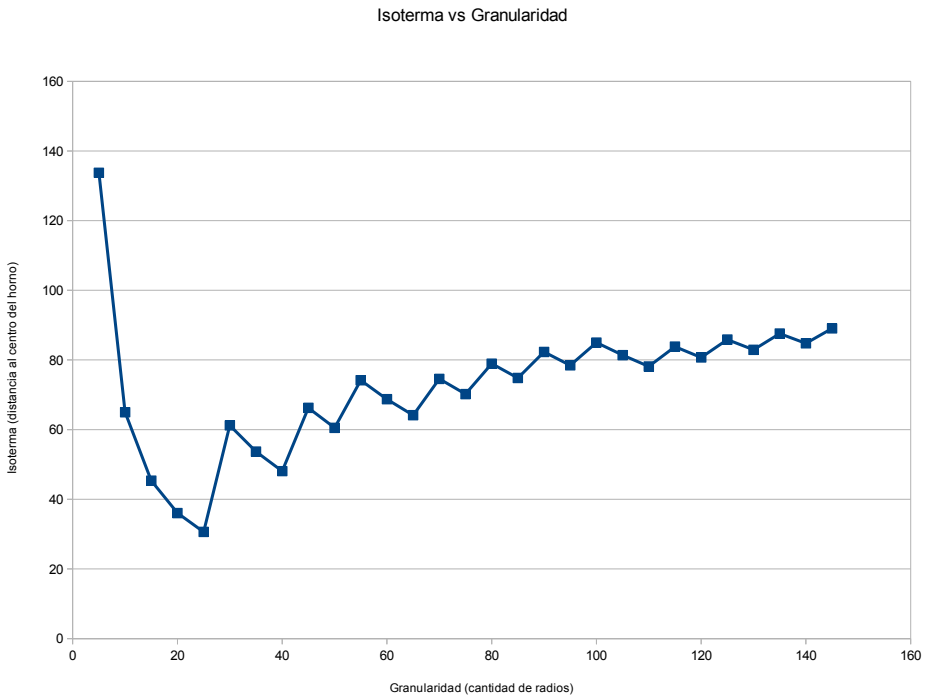


Figura 2: Resulados obtenidos aumentando la cantidad de radios de 5 en 5.

## 3.2. Test de comparación Gauss vs LU

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.

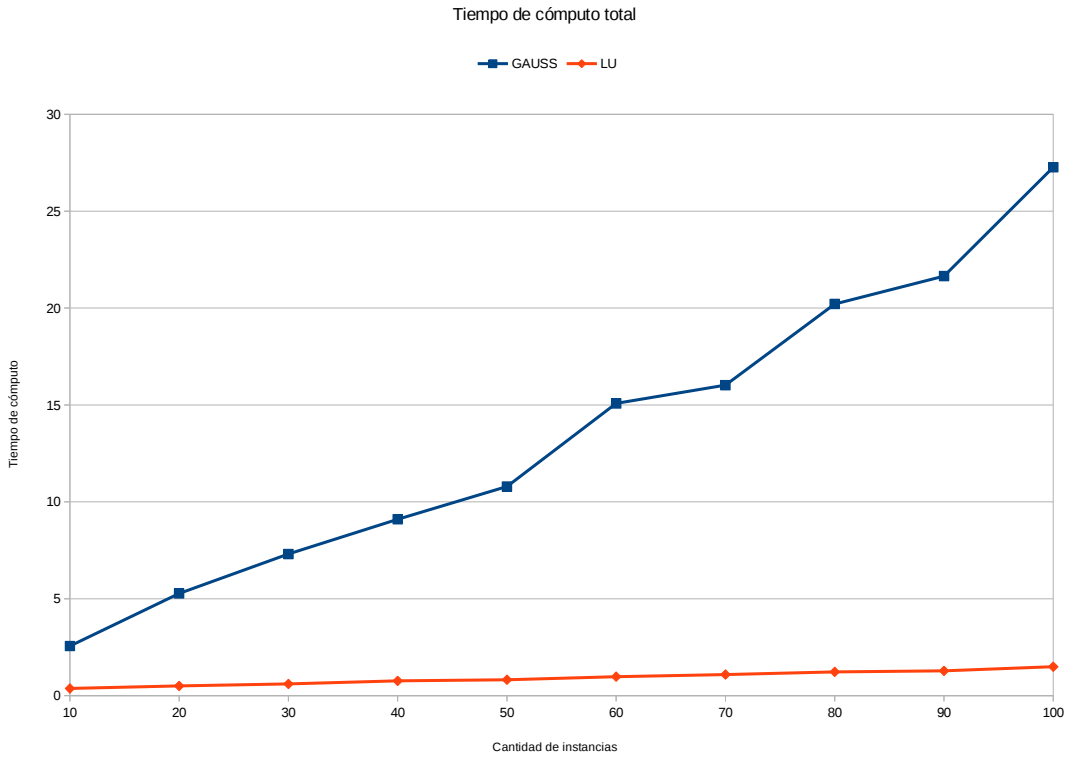


Figura 3: Resultados obtenidos usando matrices de 50 ángulos y 50 radios.

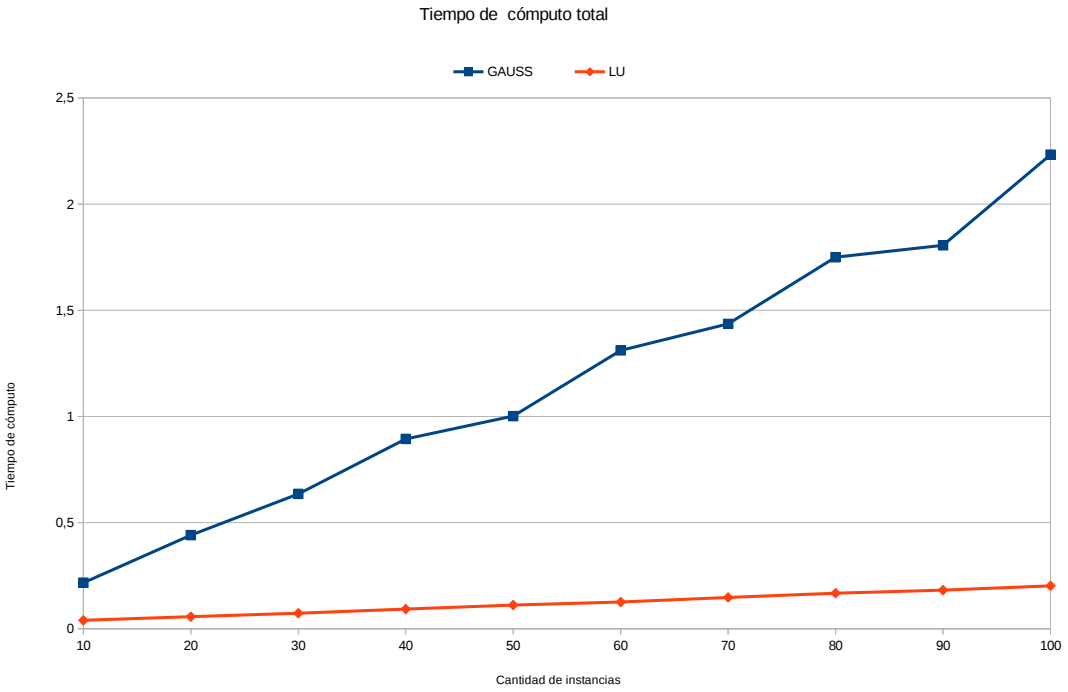


Figura 4: Resultados obtenidos usando matrices de 30 ángulos y 30 radios.

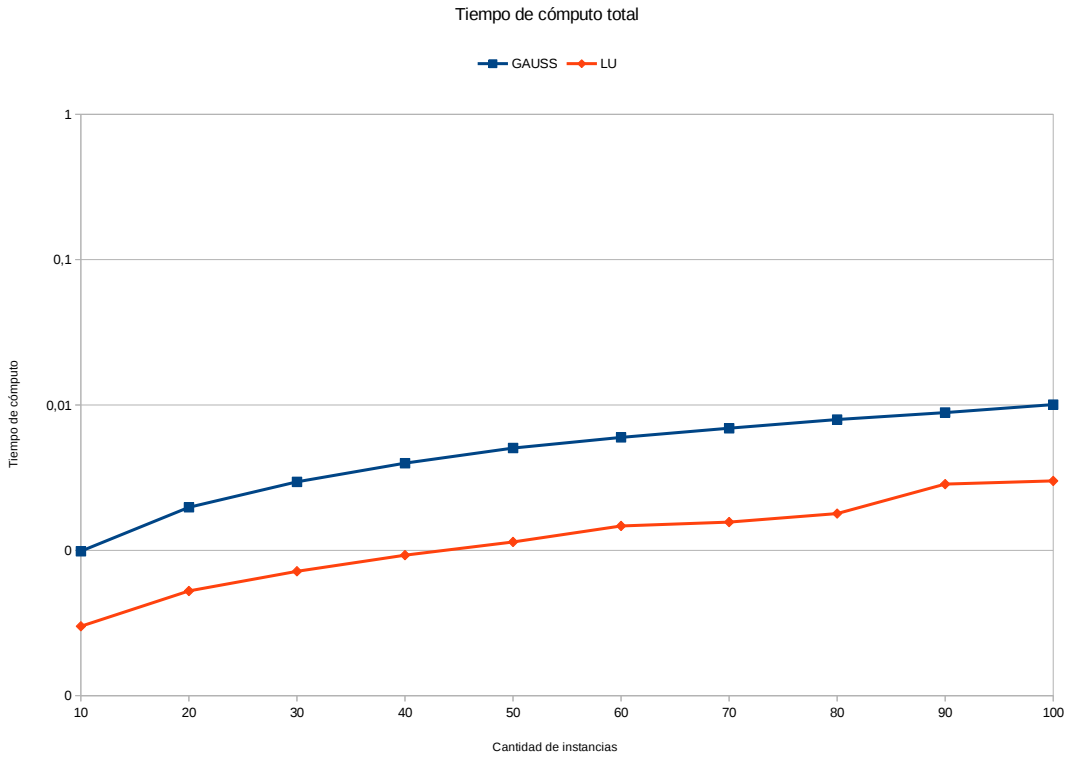


Figura 5: Resultados obtenidos usando matrices de 10 ángulos y 10 radios.

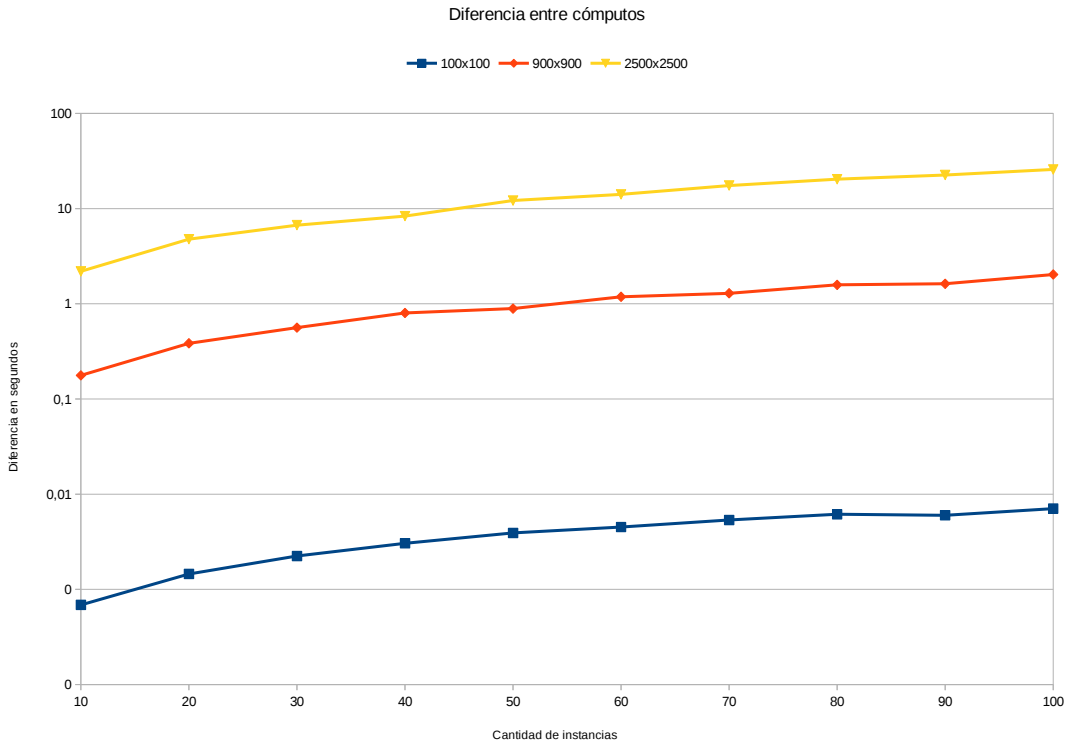


Figura 6: Diferencia entre el tiempo consumido por LU y Gauss de los gráficos anteriores. Uso de escala logarítmica.

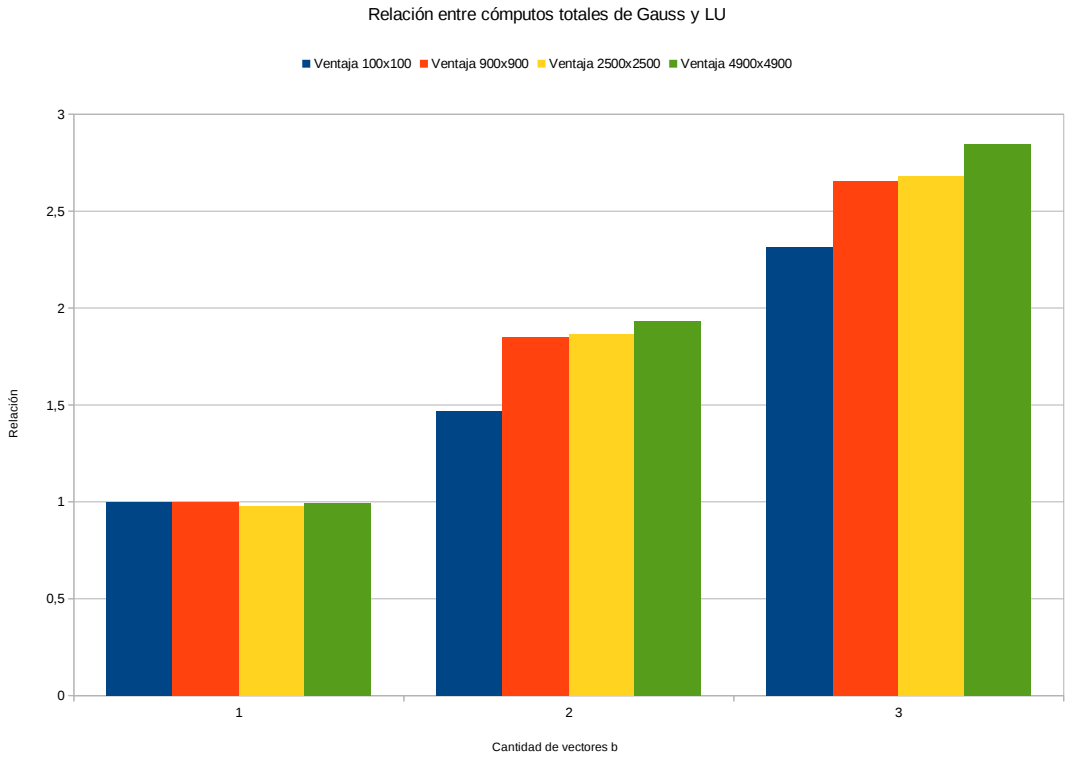


Figura 7: Relación entre el tiempo total consumido por Gauss y LU en función de la cantidad de vectores b.

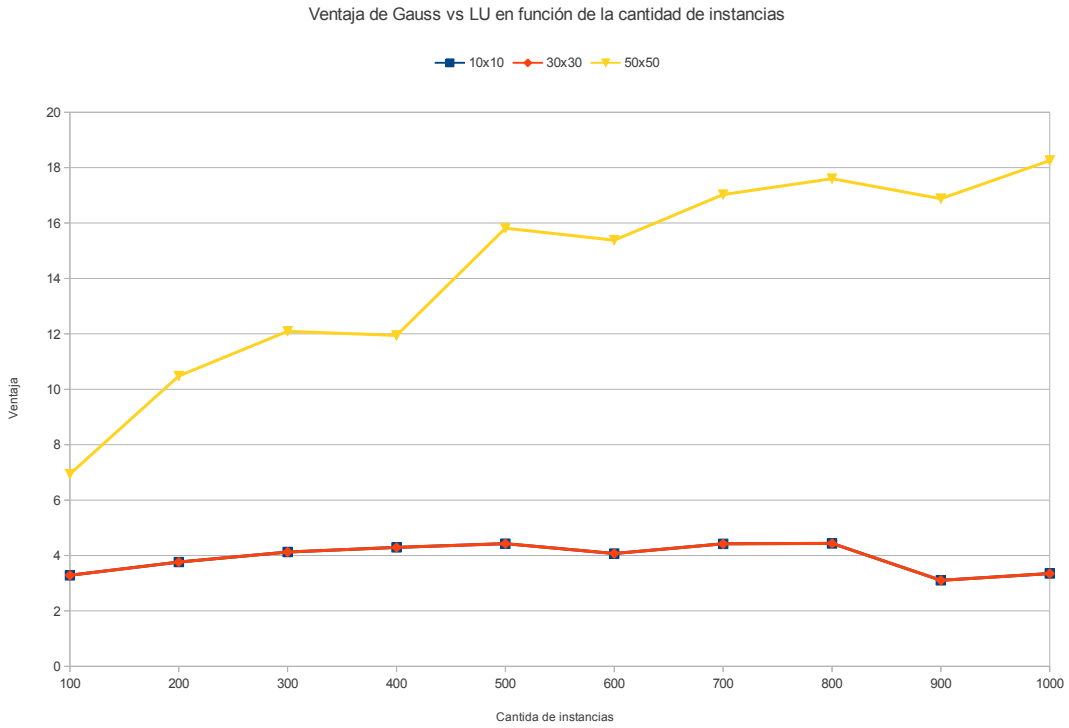


Figura 8: Ventaja de Gauss sobre LU en función de la cantidad de vectores b para matrices de 10, 30 y 50 (ángulos y radios).

## 3.3. Test de isoterma en función de las condiciones de borde

La temperatura interna y externa aumentan a la misma velocidad. La isoterma buscada en todos los casos es de $1500°C$.

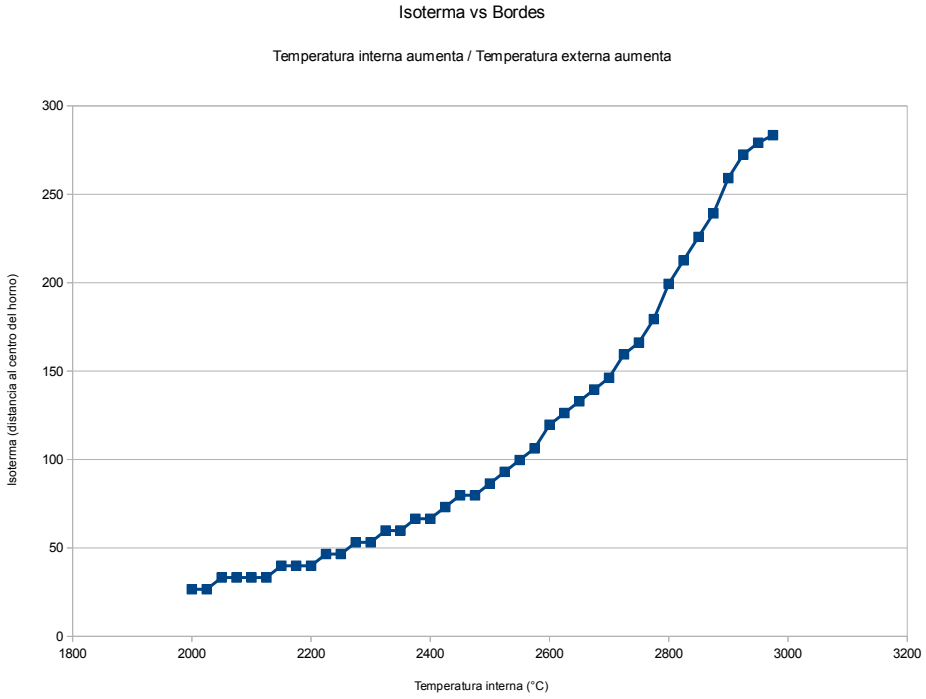


Figura 9: La temperatura externa e interna a aumentan con la misma proporción. La temperatura externa empieza en 0 $°C$

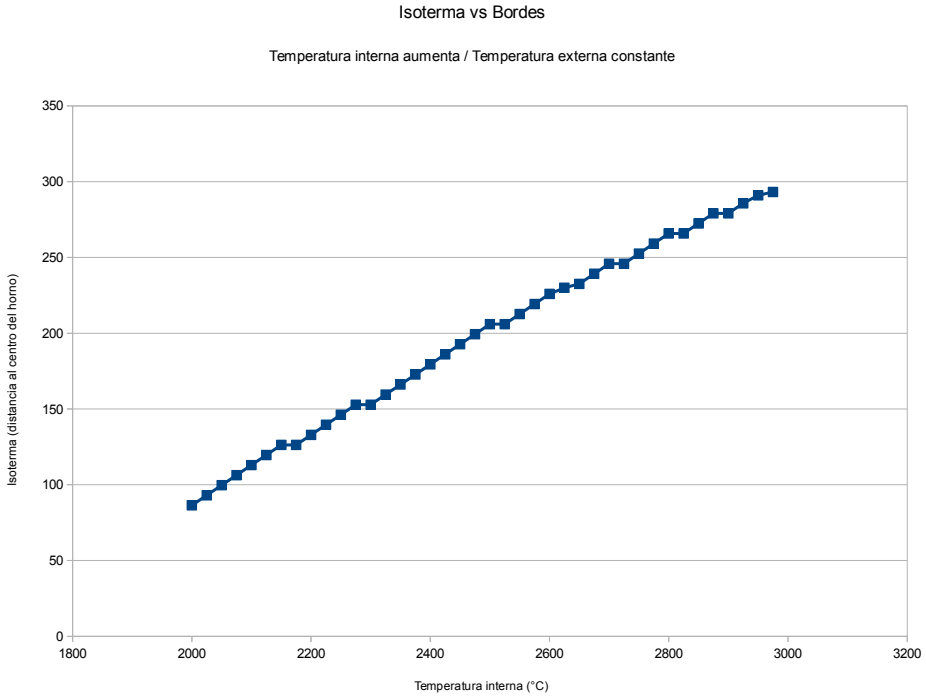


Figura 10: La temperatura externa se mantiene constante a 1000 $°C$

La temperatura externa desciende a un 40 % la velocidad a la que aumenta la temperatura interna.

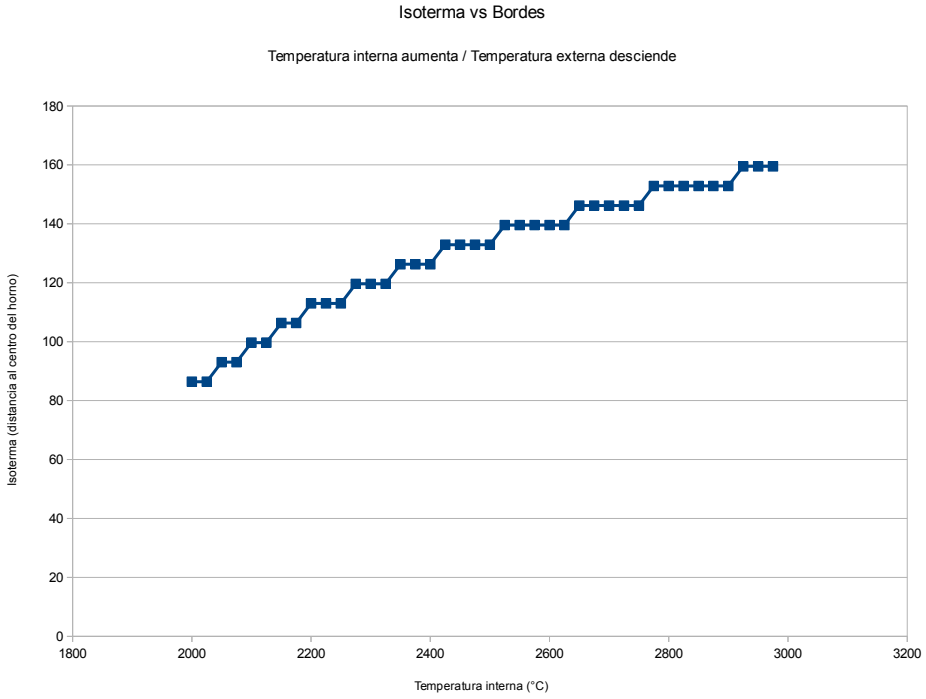


Figura 11: La temperatura externa desciende más lento que lo que aumenta la interna.

## 3.4. Test de tiempo en función de la granularidad de la discretización

En este test queríamos comprobar que la complejidad temporal empírica crecía por lo menos tan bien como una función cúbica ya que el algoritmo de Gauss tiene una complejidad temporal de $O(n^3)$.

Se midieron los tiempos para grupos de instancias de igual cantidad de ángulos y radios, ambos desde 30 hasta 100 saltando de a 1, y 5 instancias para cada uno. Se sacó promedio entre los 5 tiempos de las iteraciones del algoritmo y se muestran en el siguiente gráfico. Además, se agregó una función cúbica para comparar.

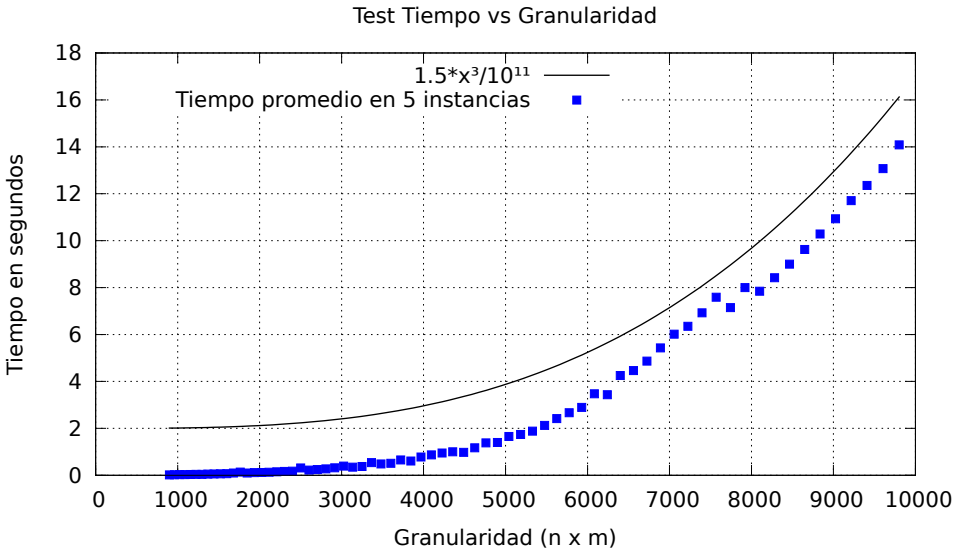


Figura 12: Comparación del tiempo de ejecución de diferentes granularidades comparando con una función cúbica.

# 4. Discusión

## 4.1. Análisis de comparación isoterma contra granularidad

En el gráfico 2 podemos ver claramente como en un principio aumentar un poco la granularidad nos cambia mucho el valor de la isoterma obtenida. Sin embargo, luego, a medida que la granularidad va en aumento, la isoterma va modificándose cada vez menos convergiendo a algún valor. Interpretamos que al aumentar la granularidad el valor obtenido para la isoterma converge a una magnitud, que corresponde a la isoterma real. Ésto coincide con la idea de que cuanta más precisión tengamos vamos a reducir el error de nuestros resultados.

## 4.2. Análisis de comparación Gauss contra LU

Como se puede apreciar en los gráficos 12, 4 y 5, 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 partir de ahora simplemente "instancias") a calcular aumenta. En el gráfico 6 se puede observar como la diferencia es proporcional tanto a la cantidad de instancias como al tamaño de las matrices, es decir, al aumentar la cantidad de instancias y/o el tamaño de la matriz LU aumenta la diferencia con respecto a Gauss. Para los cuatros tamaño de matrices probados la curva de crecimiento del tiempo total de cómputo en función de la cantidad de instancias es muy similar. Por otro lado en 7 se puede determinar que basta con resolver el sistema de ecuaciones con sólo dos vectores $b$ distintos para poder 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, es decir la *forward substitution* extra no agrega *overhead* al cálculo de la solución. EN el gráfico 8 se aprecia como evoluciona la ventaja de LU sobre Gauss. Para las matrices 10 y 30 ángulos y radios la ventaja es prácticamente constante, sin embargo la matrices de 50 ángulos y radios, la ventaja crece en función de la cantidad de instancias confirmando lo visto en los anteriores gráficos, la superioridad de LU sobre Gauss.

### 4.3. Análisis de comparación isoterma en función de las condiciones de borde

El gráfico 9 muestra claramente que al aumentar la temperatura externa e interna en la misma proporción la isoterma se mueve cada vez más rápido hacia el borde que se acerca a ella. En este caso es el borde externo. La temperatura externa empieza en 0 $°C$ y finaliza en 1000 $°C$. Esto coincide con lo que esperaríamos en una situación real. En el gráfico 10 mantuvimos la temperatura externa constante por lo tanto la isoterma siguió acercándose al borde exterior pero siempre al mismo ritmo. Interpretamos que se acerca con velocidad lineal porque es la velocidad a la que crece la temperatura interna. Finalmente, en el gráfico 11 vemos claramente cómo el movimiento de los bordes va afectando directamente a la posición de la isoterma y en sus mismas proporciones. A pesar de que la temperatura externa desciende mientras que la interna asciende, la isoterma sigue acercándose hacia el borde externo. Ésto se debe a que la temperatura externa desciende a un 40 % la velocidad a la que aumenta la interna. Después de analizar éstos tres gráficos concluimos que el modelo implementado coincide en gran parte con lo que el sentido común indica. El valor de la isoterma va modificándose de forma directamente proporcional a el cambio de las temperaturas internas y externas que ejercen influencia sobre ella en sentidos contrarios.

### 4.4. Análisis de comparación de granularidad y tiempo

Como era de esperarse, el tiempo de ejecución es creciente con la granularidad y, además, aparentaría ser cúbico con una función con multiplicadores extraños. Esto no debería ser de demasiada sorpresa dado que estamos midiendo en segundos y el tiempo es siempre menor a 15 segundos. Si tenemos en cuenta esto y además que la granularidad va desde $30^2$ hasta $100^2$, no parece tan alocado tener que dividir por $10^{11}$ y multiplicar por 2.

Éste fue un test satisfactorio dado que ocurrió lo que esperábamos: poder acotar el crecimiento de los tiempos por una función cúbica y sin resultados extraños inesperados. Sí hay algunas pequeñas irregularidades en los puntos, pero, aunque incomprobable, bien podría ser responsabilidad del scheduler del sistema operativo u otra de todas las variables que podrían haber afectado mínimamente los resultados, aún sacando promedio en 5 instancias.

## 5. Conclusiones

Como ya se mencionó, este trabajo tiene dos secciones principales: una es la resolución se sistema de ecuaciones lineales mediante técnicas algorítmicas matriciales, y la otra es el problema de estimar una isoterma a partir de una cantidad finita de puntos con sus correspondientes temperaturas. Para la parte inicial se estudió la eliminación gaussiana y la factorización LU. Pese a que son muy parecidas en su definición y en su implementación algorítmica, es preferible usar la segunda en caso de que se necesite resolver un sistema para una misma matriz $A$ pero con distintos vectores $b$ ya que reduce el costo computacional de manera contundente.

Una desventaja es que la factorización LU no siempre existe pero siempre que exista es preferible utilizarla antes que la eliminación gaussiana clásica. Más aun, cuanto más vectores b distintos haya, mayor será diferencia entre ambos algoritmos. Además, cabe señalar que la factorización LU no requiere espacio de memoria adicional ni agrega complejidad al algoritmo de eliminación gaussiana sin pivoteo.

En el caso de los tests con la isoterma, nuestros algoritmos se acercaron a lo que se esperaría de la realidad, o la realidad se acercó a lo que se esperaba de nuestros algoritmos, cómo se quiera verlo. Mientras más ricos seamos en la industria del tiempo de cómputo, más granularidad podremos permitirnos comprar, y más exacta será nuestra isoterma.

También fue el caso que la isoterma tendía a acercarse a la pared que más debía acercarse

según nuestras predicciones, confirmando empíricamente que nuestras acepciones sobre las leyes físicas y nuestro entendimiento de la fórmula de expansión del calor eran correctas.

Como trabajo práctico en general, fue una experiencia nueva. Aprendimos a prestarle más atención al por qué hacíamos las cosas en vez de cómo hacerlas.

# 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.

Notación útil:

$A(i|j)$ := Es la submatriz de $A$ que se obtiene de eliminar las filas de 1 a $i$ y las columnas de 1 a $j$.

$A^{(k)}$ := Es la matriz obtenida luego de realizar $k$ pasos de eliminación gaussiana sobre las filas de $A$.

Demostración:

Como A es diagonal dominante, se tiene que $|A_{ii}| \geq \sum_{j=1,i\neq j}^{n} |A_{ij}|$ para todo $j \neq i$, por lo tanto debe suceder que $A_{11} \neq 0$, si así no fuera, se tiene que $0 = |A_{11}| \geq |A_{1j}| \geq 0$ para todo $j \neq 2$ a $n$, por lo que $|A_{1j}| = 0$, y esto sucede si y solo si $A_{ij} = 0$, es decir, la columna 1 es nula y por lo tanto A no es inversible, pero A es inversible y en consecuencia $A_{11} \neq 0$. Teniendo en cuenta que $A_{11} \neq 0$, veamos ahora que al realizar eliminación gaussiana sobre $A$, la matriz $A(1|1)^{(1)}$ resulta ser diagonal dominante: Hay que probar que para todo $j$ vale:

$$\sum_{i\geq 2, i\neq j}^{n} |A_{ij}^{(1)}| \leq |A_{jj}^{(1)}|$$

Tenemos que:

$$\sum_{i\geq 2, i\neq j}^{n} |A_{ij}^{(1)}| = \sum_{i\geq 2, i\neq j}^{n} |A_{ij} - \frac{A_{1j}A_{i1}}{A_{11}}| \leq \sum_{i\geq 2, i\neq j}^{n} |A_{ij}| + |\frac{A_{1j}A_{i1}}{A_{11}}| = \sum_{i\geq 1, i\neq j}^{n} |A_{ij}| - |A_{1j}| + |\frac{A_{1j}}{A_{11}}|(\sum_{i\geq 2}^{n} |A_{i1}| - |A_{j1}|)$$

y como A es diagonal dominante:

$$\sum_{i\geq 1, i\neq j}^{n} |A_{ij}| \leq |A_{jj}| \qquad \text{y} \qquad \sum_{i\geq 2}^{n} |A_{i1}| \leq |A_{11}|$$

Finalmente:

$$\sum_{i\geq 1, i\neq j}^{n} |A_{ij}| - |A_{1j}| + |\frac{A_{1j}}{A_{11}}|(\sum_{i\geq 2}^{n} |A_{i1}| - |A_{j1}|) \leq |A_{jj}| - |A_{1j}| + |\frac{A_{1j}}{A_{11}}|(|A_{11}| - |A_{j1}|) =$$

$$|A_{jj}| - |\frac{A_{1j}A_{j1}}{A_{11}}| \leq |A_{jj} - \frac{A_{1j}A_{j1}}{A_{11}}| = |A_{jj}^{(1)}|$$

Por lo tanto $A(1|1)^{(1)}$ es diagonal dominante e inversible porque la matriz original lo era (si no lo fuera se podría triangular obteniendo al menos una fila con ceros, pero como $A$ ya tiene ceros en la primer columna, existiendo una triangulación de $A$ con una fila nula, lo que es un absurdo

pues $A$ ya era inversible), por lo que aplicando la demostración anterior para $A(1|1)^{(1)}$, se tiene que también es posible realizar el paso 2 de gauss sin pivoteo. Aplicando la demostración anterior recursivamente se obtiene que las matrices $A(2|2)^{(2)}$, ..., $A(k|k)^{(k)}$ son todas diagonal dominantes e inversible, donde $k$ es el último paso de la eliminación gaussiana, demostrando lo pedido.

# Referencias

[1] Richard Burden. **Numerical Analysis.** Brooks Cole, 2000.