DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

TP1 - Trabajamos y nos divertimos...

April 25, 2022 Métodos Numéricos

Grupo 8

Integrante	LU	Correo electrónico
Cappella Lewi, F. Galileo	653/20	galileocapp@gmail.com
Anachure, Juan Pablo	99/16	janachure@gmail.com

En este trabajo estudiamos una manera de calcular las temperaturas dentro de la pared de un alto horno, buscando estimar la posición de la isoterma. Utilizamos técnicas matriciales para resolver sistemas de ecuaciones lineales.

Palabras clave:

Alto Horno	Sistemas Matriciales
Eliminación Gausseana	Factorización LU

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2610 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (++54 +11) 4576-3300
https://exactas.uba.ar

Secciones

1 Introducción

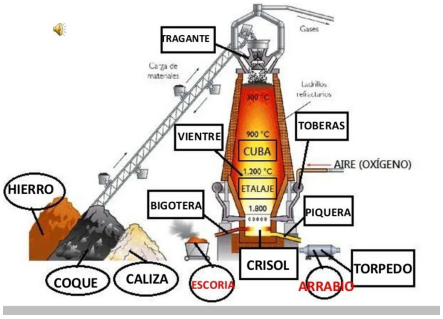

Fig. 1: Diagrama de las partes de un alto horno

Los altos hornos son un tipo de horno metalúrgico, usados para producir metales industriales. Están formados por una pared circular usualmente de acero y ladrillos refractarios, lo que permite fundir distintos metales dentro. En la Figura 2 se puede ver un diagrama de la sección horizontal de un horno.

Estos hornos suelen trabajar con temperaturas internas de 900ºC a 1300ºC [**?**] y en las paredes externas se suelen medir temperaturas de entre 50ºC y 200ºC. Dentro de la pared es importante la posición de la isoterma de 500ºC, ya que si esta se encuentra cerca del exterior (ver Sección 3) la integridad del horno podría estar en riesgo.

En este trabajo se estudia una manera de calcular las temperaturas dentro de la pared, buscando poder estimar la posición de la isoterma. Para ello se utiliza una discretización de la ecuación del calor de Laplace (Sección 2.1).

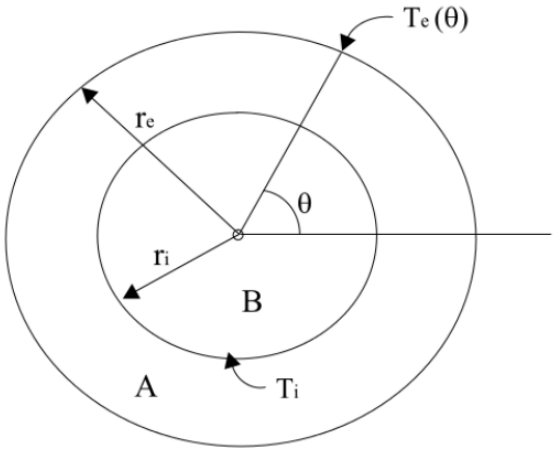

Fig. 2: Sección horizontal de un horno

El problema acerca de cómo encontrar la isoterma, y medir la peligrosidad se puede resolver utilizando un modelo que requiere de un conjunto de temperaturas medidas en diferentes ángulos. Esto es procesado utilizando un sistema de ecuaciones lineales, por lo que será de interés utilizar métodos que permitan encontrar soluciones a este tipo de sistemas, específicamente este trabajo esta orientado en la utilización de Eliminación Gaussiana y factorización LU. Finalmente, utilizando los resultados de dichos métodos se quiere evaluar la peligrosidad del horno.

2 Desarrollo

2.1 Modelado

La ecuación del calor Laplace afirma que en estado estacionario cualquier temperatura $T(r, \theta)$, dada por la posición dentro de la pared del horno en coordenadas polares (r, θ) que representan al radio y el ángulo polar, cumple la siguiente ecuación:

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

Para poder utilizarla para resolver el problema computacionalmente, discretizamos el dominio a coordenadas polares. Tomando una partición $0 = \theta_0 < \theta_1 < ... < \theta_n = 2\pi$ en n ángulos discretos con $\theta_k - \theta_{k-1} = \Delta\theta$ para $k = 1, ..., n$ 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 $j = 1, ..., m$. Tomamos $t_{j,k} = T(r_j, \theta_k)$ y transformamos a la ecuación (1) en un conjunto de ecuaciones lineales sobre las incógnitas $t_{j,k}$, aproximando así a T por la siguiente fórmula de diferencias finitas:

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

Que podemos reescribir como una combinación lineal de cinco temperaturas:

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

Ahora armamos un sistema de ecuaciones para cada $t_{j,k}$, que podemos representar matricialmente de la forma $At = b$. Donde $A \in \mathbb{R}^{(m+1)n \times (m+1)n}$ es una matriz con las primeras y últimas n filas iguales a la identidad (por las temperaturas externas e internas, que son datos), y el resto corresponde a una instancia de la ecuación (3). $t \in \mathbb{R}^{(m+1)n}$ es el vector de incógnitas con los radios "estirados" uno arriba del otro. Y $b \in \mathbb{R}^{(m+1)n}$ es el vector de resultados que tiene las temperaturas internas en los primeros n valores, las externas en los últimos n valores, y el resto todos ceros.
En la ecuación (4) mostramos un ejemplo de cómo queda el sistema con $r_i = 1,\ r_e = 3,\ mp1 = 3,\ n = 3,\ t_{i,1} = t_{i,2} = t_{i,3} = 1500,\ t_{e,1} = t_{e,2} = t_{e,3} = 150$.

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{2} & 0 & 0 & -\frac{2}{\pi^2} - \frac{3}{2} & \frac{1}{\pi^2} & \frac{1}{\pi^2} & 1 & 0 & 0 \\
0 & \frac{1}{2} & 0 & \frac{1}{\pi^2} & -\frac{2}{\pi^2} - \frac{3}{2} & \frac{1}{\pi^2} & 0 & 1 & 0 \\
0 & 0 & \frac{1}{2} & \frac{1}{\pi^2} & \frac{1}{\pi^2} & -\frac{2}{\pi^2} - \frac{3}{2} & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix} \cdot \begin{pmatrix} t_{1,1} \\ t_{1,2} \\ t_{1,3} \\ t_{2,1} \\ t_{2,2} \\ t_{2,3} \\ t_{3,1} \\ t_{3,2} \\ t_{3,3} \end{pmatrix} = \begin{pmatrix} 1500 \\ 1500 \\ 1500 \\ 0 \\ 0 \\ 0 \\ 150 \\ 150 \\ 150 \end{pmatrix} \tag{4}$$

Demostramos en el apéndice **??** que se puede aplicar la Eliminación Gaussiana sin pivotear filas.

2.2 Implementación

Implementamos este sistema en `C++`. Primero programamos matrices usando vectores de la `STL`. Luego hicimos funciones que resuelvan sistemas de ecuaciones usando Eliminación Gaussiana o factorización LU. Y finalmente recibimos los datos y guardamos los resultados en archivos pasados por parámetros.
También armamos una herramienta en `Python` para correr el programa y analizar los resultados, armando los gráficos vistos a través de este informe (como la Figura 3).
Todo el código se puede ver adjuntado al informe, junto con explicaciones de cómo usarlo.

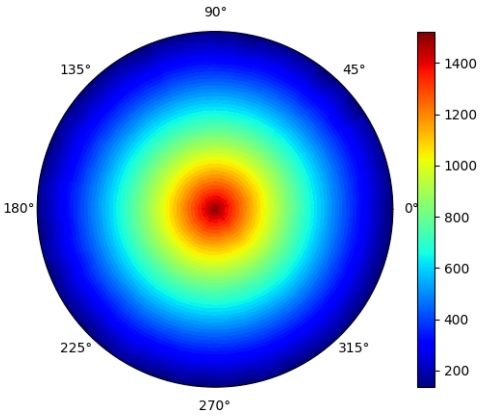

Fig. 3: Temperatura calculada dentro del horno (mp1 = 50, n = 50, Ti ≈ 1500ºC, Te ≈ 500ºC)

3 Estimando la isoterma

En la Sección 1 planteamos que la razón principal para buscar resolver este problema es querer calcular la posición de la isoterma de 500ºC, ya que si está muy cerca del exterior (ver Sección 3.1) la estructura del horno estaría en riesgo.

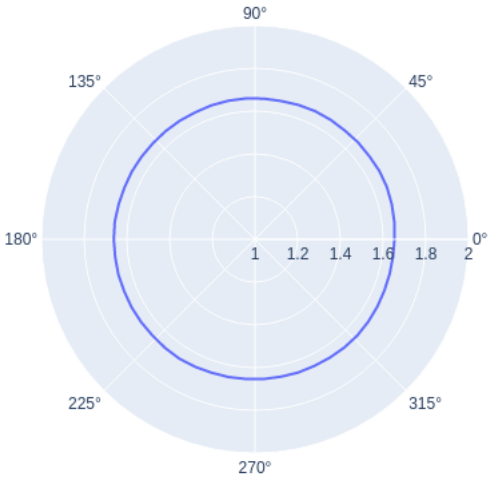

Fig. 4: Posición estimada de la isoterma (mp1 = 50, n = 50, Ti $\approx$ 1500ºC, Te $\approx$ 500ºC)

Para encontrarla buscamos por ángulo el primer punto donde la temperatura es menor a la buscada, y luego interpolamos linealmente la posición de la isoterma entre ese punto y el anterior.

3.1 Midiendo la peligrosidad

Una vez encontrada la isoterma, es importante decidir si el horno está en un estado peligroso o seguro.
Para esto habría que conocer más sobre cómo está armado un alto horno para poder sacar conclusiones sobre qué tan peligrosa es la posición de la isoterma. Aunque preliminarmente se puede ver (Figura 5) que si el punto de ruptura se encuentra antes de que la isoterma haya recorrido el 75% de la pared hacia afuera hay poco margen de error, ya que hasta ese punto la isoterma se mueve rápido hacia el exterior.

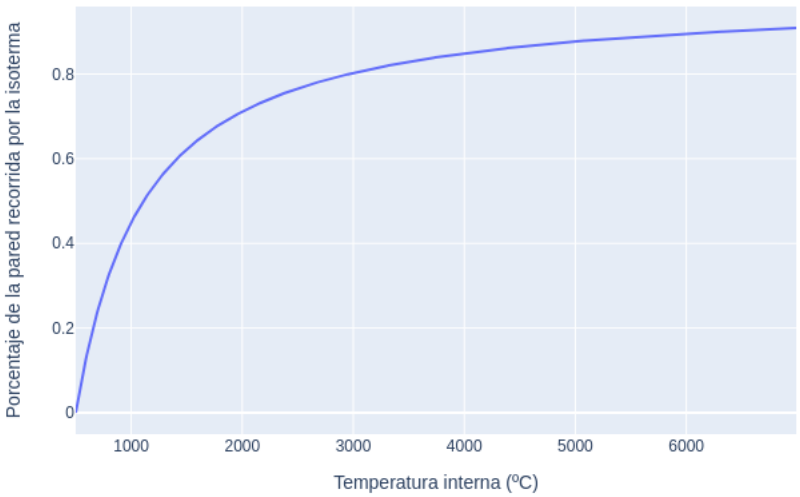

Fig. 5: Distancia de la isoterma al exterior del horno

4 Experimentación

4.1 Resultados esperados

La cátedra nos proveyó con una serie de tests, con su input y output esperados. Al correr nuestro programa con este input nos encontramos con que los resultados conseguidos no eran iguales a los esperados (ver Figura 6).
Para averiguar si este error es aceptable usamos el número de condición. Que es un número calculado a partir de la matriz que nos permite definir un rango de error alrededor del resultado esperado, dentro del cual, si se encuentran nuestros resultados, son aceptables y esperables por errores de redondeo (ver Sección 4.2).

Se puede notar que el error crece hacia el centro del horno. Esto se debe a que el algoritmo implementado para resolver el sistema despeja las temperaturas más internas en función de las más externas, por lo que los errores cometidos se van acumulando desde afuera hacia adentro.

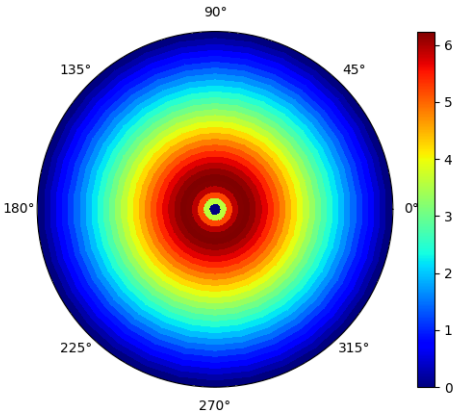

Fig. 6: Diferencia entre las temperaturas esperadas y las calculadas del test 1

Corrimos nuestro programa teniendo en cuenta el error y confirmamos que nuestros resultados caen dentro de este rango de error aceptable. Esto se puede repetir usando el script `analyze_expected.py`. Es muy relevante que la isoterma estimada entre ambos resultados no es muy diferente, por lo que este error no debería causar problemas.

4.2 Revisando errores de redondeo

Un problema común en la computación es que, al trabajar con números representados en una cantidad fija de dígitos binarios (bits), puede haber errores de redondeo. Buscamos cuantificar la diferencia entre dos tipos de representación. Para ello, queremos comparar los resultados entre usar `long double` es decir, utilizando 128-bits para guardar los números y utilizar `float` que usa 32-bits.

Armamos un sistema que genere coeficientes muy pequeños (menores que 10^{-6}) y utilice temperaturas muy altas ($T_i, T_e \geq 10^6$). De esta manera, deberían de notarse diferencias en los resultados entre las representaciones.

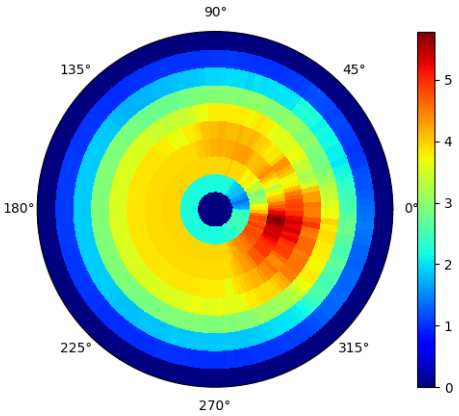

Fig. 7: Diferencia entre las temperaturas calculadas usando `long double` y `float`

En la Figura **??** se pueden ver las diferencias causadas. Parte de estos errores se deben a lo mencionado en la Sección 4.1 ya que los algoritmos propuestos tienen un leve error de redondeo. Observemos que los errores crecen hacia el centro del horno, esto sucede por cómo se resuelve el sistema, ya que al usar los resultados de las temperaturas más externas para calcular las más internas el error se acumula hacia el centro. Y no hay error en las temperaturas del radio interno y del radio externo, porque estos son datos de entrada y no calculados.

Este experimento puede ser repetido modificando la entrada en `data/rounding.in` y corriendo el script `analyze_rounding.py`.

4.3 Movimiento de la isoterma

4.3.1 Movimiento de la isoterma en base a la temperatura interna

Algo interesante a estudiar, es ver cómo varía la isoterma estimada en función a la temperatura interna que medimos. Se busca comprobar que la ubicación de la isoterma esta muy relacionada a la variabilidad de la temperatura interna. Motivado por este experimento, también resaltamos puntos de fusión de los metales más trabajados en altos hornos, de esta manera se puede observar la posición de la isoterma en cada uno. Los metales estudiados fueron aluminio, el cobre, el hierro, y el tungsteno [**?**].

Para armar el experimento variamos las temperaturas internas desde 500 hasta 7000 grados centígrados.
Como se puede ver en la Figura **??**, se observa una clara relación entre la temperatura interna del horno y la ubicación de la isoterma.

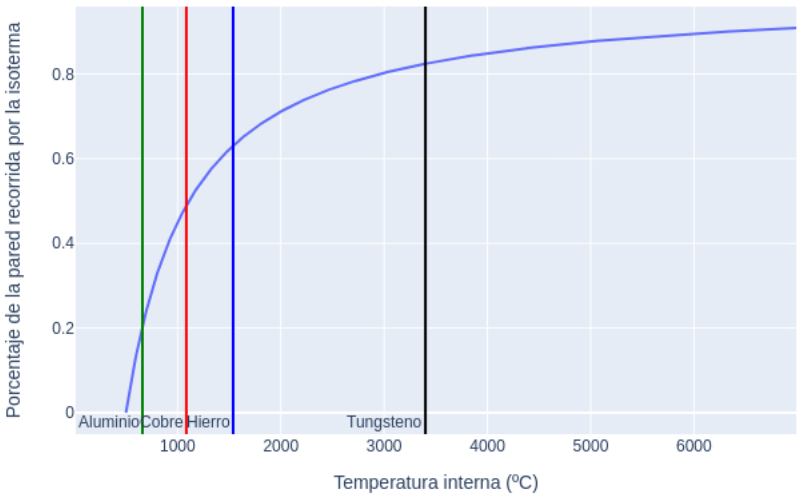

Fig. 8: Distancia de la isoterma al exterior del horno (resaltando temperatura de fusión de ciertos metales)

Finalmente, se puede notar la ubicación de los metales elegidos que aquellos que necesitan una temperatura del horno más baja, tienen la posición de su isoterma más cercana al centro, Mientras que las que requieren más calor tienen una isoterma más cercana a la pared exterior.

Este experimento puede ser repetido con el script `analyze_isotherm.py`.

4.3.2 Movimiento de isoterma en función de la cantidad de radios

Por como se planteó la estimación de la isoterma, nos parece que la cantidad de radios dentro de la pared del horno sobre los que se trabaja afectan a la posición estimada de la isoterma. Para estudiar el efecto que tiene esta variable, analizamos los resultados cuando cambiamos la cantidad de radios. Asumimos a las temperaturas internas y externas fijadas en 1500ºC y 150ºC respectivamente.

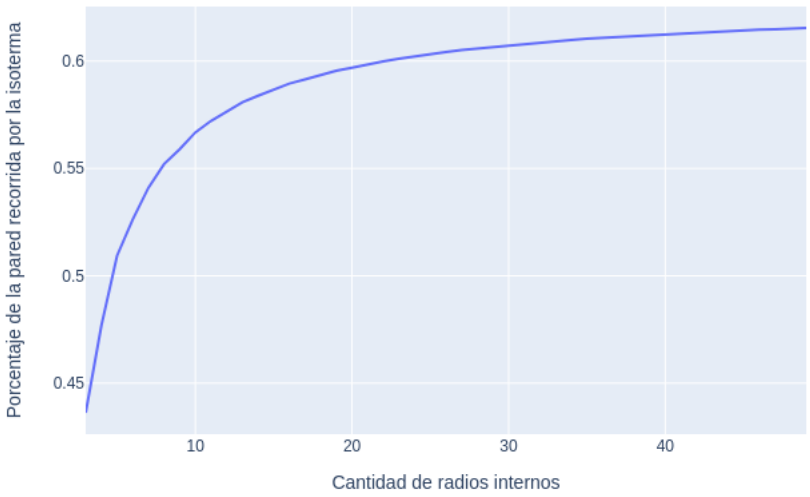

Fig. 9: Posición estimada de la isoterma

Como se ve en la Figura **??**, la posición varía bastante en base a la cantidad de radios. Preliminarmente parece convincente que si se calculan más radios se debería conseguir la estimación más correcta de la isoterma, pero habría que conseguir datos donde se sabe la posición de la isoterma para confirmarlo.

Este experimento puede ser repetido con el script `analyze_isotherm.py`.

5 Conclusión

En nuestra experimentación nos encontramos con errores en los cálculos. No creemos que se deba a un error en la metodología usada, ni en la implementación. Sino a un error esperable al estar discretizando una ecuación continua, y al usar computadoras para hacer cuentas. A futuro nos gustaría hacer otra implementación que aproveche diferentes técnicas de representación numérica para reducir los errores causados por la tecnología usada.

Pudimos corroborar que la posición estimada de la isoterma depende de parámetros elegidos arbitrariamente. Esto resalta que las decisiones tomadas al resolver un problema pueden afectar a los resultados, y tienen que ser consideradas.

También se debería analizar la estructura material de los altos hornos, para poder definir la peligrosidad.

I Justificación Gauss sin pivoteo

En este anexo demostramos que las matrices armadas (Sección 2.1) son diagonal dominantes y luego justificamos inductivamente que se puede aplicar el método de Eliminación Gaussiana sin reordenar filas.

Definición 1. *Sea $A \in \mathbb{R}^{n\times n}$. A se dice* ***diagonal dominante*** *si*

$$\sum {}^{j=1}_{j\neq i} n|A_{ij}| \leq |A_{ii}| \qquad \forall i = 1, ..., n$$

Lema 1. *Sea $A^{(0)} = A \in \mathbb{R}^{n\times n}$ una matriz diagonal dominante, con $A_{1,1} \neq 0$, y $A^{(1)}$ el resultado de aplicar un paso de Eliminación Gaussiana (sin pivoteo) sobre A. Entonces $A^{(1)}$ es diagonal dominante.*

Proof. Consideremos la k-ésima fila de $A^{(1)}$, queremos ver que:

$$\sum {}^{k=1}_{k\neq i} n|a^{(1)}_{i,k}| \leq |a^{(1)}_{i,i}|$$

Se tiene que:

$$|a^{(1)}_{k,k}| = |a_{k,k} - \frac{a_{k,1}}{a_{1,1}} a_{1,k}| \qquad \text{y} \qquad \sum_{\substack{i=1\\ i\neq j}}^{n} |a^{(1)}_{j,i}| = \sum_{\substack{i=2\\ i\neq j}}^{n} |a_{j,i} - \frac{a_{j,1}}{a_{1,1}} a_{1,i}|$$

Ahora veamos como es un paso de Eliminación Gaussiana:

$$\begin{aligned}
\sum_{\substack{k=1\\ k\neq i}}^{n} |a^{(1)}_{i,k}| &= \sum_{\substack{k=2\\ k\neq i}}^{n} |a_{i,k} - \frac{a_{i,1}}{a_{1,1}} a_{1,k}| \\
&\leq \sum_{\substack{k=2\\ k\neq i}}^{n} |a_{i,k}| + \left|\frac{a_{i,1}}{a_{1,1}}\right| \sum_{\substack{k=2\\ k\neq i}}^{n} |a_{1,i}| \\
&\leq (|a_{i,i}| - |a_{i,1}|) + \left|\frac{a_{i,1}}{a_{1,1}}\right| (|a_{1,1}| - |a_{1,i}|) \\
&= |a_{i,i}| - \left|\frac{a_{i,1}}{a_{1,1}}\right| |a_{1,i}| \\
&\leq \left|a_{i,i} - \frac{a_{i,1}}{a_{1,1}} a_{1,i}\right| = |a^{(1)}_{i,i}|
\end{aligned}$$

□

Ahora se ve como luego de realizar un paso de Eliminación Gaussiana la matriz resultante también es diagonal dominante.

Finalmente queda demostrar que, luego de realizar una iteración del algoritmo de Eliminación Gaussiana sin pivoteo sobre nuestra matriz, la matriz resultante queda bien definida. Entonces:

$$\begin{aligned}
|a_{i,i}| &= \beta \\
&= \left| -\frac{2}{(\Delta r)^2} + \frac{1}{r\Delta r} - \frac{2}{r^2(\Delta\theta)^2} \right| \\
&= \left| -\frac{1}{(\Delta r)^2} + \frac{1}{r\Delta r} - \frac{2}{r^2(\Delta\theta)^2} - \frac{1}{(\Delta r)^2} \right| \\
&= \left| -\frac{r-\Delta r}{r(\Delta r)^2} - \frac{2}{r^2(\Delta\theta)^2} - \frac{1}{(\Delta r)^2} \right| \\
&= \left| \frac{r-\Delta r}{r(\Delta r)^2} + \frac{2}{r^2(\Delta\theta)^2} + \frac{1}{(\Delta r)^2} \right|
\end{aligned}$$

Veamos que si tenemos $r_{int}, \Delta r > 0, j \geq 1$, se cumple que

$$r - \Delta r = (r_{int} + j\Delta r) - \Delta r = r_{int} + (j-1)\Delta r > 0$$

Entonces ahora sabemos que:

$$r - \Delta r > 0$$

Ahora veamos como queda la sumatoria sobre una fila. Notar que:

$$|a_{i,i}| = \frac{r-\Delta r}{r(\Delta r)^2} + \frac{2}{r^2(\Delta\theta)^2} + \frac{1}{(\Delta r)^2}$$

Ahora al sumar los módulos del resto de los coeficientes queda que:

$$\begin{aligned}
\sum_{\substack{k=1 \\ k\neq i}}^{(m+1)n} |a_{i,k}^j| &= |\alpha| + |\gamma| + 2|\chi| \\
&= \left| \frac{1}{(\Delta r)^2} - \frac{1}{r\Delta r} \right| + 2\left| \frac{1}{r^2(\Delta\theta)^2} \right| + \left| \frac{1}{(\Delta r)^2} \right| \\
&= \left| \frac{r-\Delta r}{r(\Delta r)^2} \right| + \frac{2}{r^2(\Delta\theta)^2} + \frac{1}{(\Delta r)^2} \\
\text{(Por lo probado antes)} &= \frac{r-\Delta r}{r(\Delta r)^2} + \frac{2}{r^2(\Delta\theta)^2} + \frac{1}{(\Delta r)^2} \\
&= |a_{i,i}|
\end{aligned}$$

Así de la demostración anterior se obtuvo que la matriz $A \in \mathbb{R}^{(m+1)n\times(m+1)n}$ de nuestro sistema es diagonal dominante y aparte no tiene ceros en la diagonal. Ahora resta ver que la Eliminación Gaussiana es aplicable y se puede utilizar sin pivoteo. Así solamente hay que ver que para los puntos del medio del horno puede aplicarse Eliminación Gaussiana y además la matriz resultante es diagonal dominante.

- En las primeras y últimas n filas, la matriz ya está triangulada, por lo que no hace falta aplicar Gauss.

Caso base: Notemos que $a_{1,1} = 1$ no es nulo, entonces se puede aplicar el primer paso de la Eliminación Gaussiana. Dado que $a_{1,2} = 0$ entonces si aplicamos un paso de Eliminación Gaussianaen en el lugar $a_{2,2}$ queda un coeficiente no nulo y dado que la matriz luego de haber procesado la primer fila sigue siendo diagonal dominante entonces puedo aplicar el siguiente paso de Eliminación Gaussiana.

HI: Tomamos como Hipótesis Inductiva que la matriz $A^{(k-1)}$, obtenida tras aplicar $k-1$ pasos de Eliminación Gaussiana sobre A, es diagonal dominante y su k-ésima fila es no nula. Esto implica que ${a^{(k-1)}}_{k,k} \neq 0$.

Paso inductivo: Vamos a escribir a $A^{(k-1)}$ por bloques de la siguiente manera:

$$A^{(k-1)} = \begin{pmatrix} {A^{(k-1)}}_{1,1} & B \\ C & {A^{(k-1)}}_{2,2} \end{pmatrix}$$

Con ${A^{(k-1)}}_{1,1} \in \mathbb{R}^{(k-1)\times(k-1)}$.
Luego de haber probado lo anterior, sabemos que ${A^{(k-1)}}_{1,1}$ y ${A^{(k-1)}}_{2,2}$ son matrices diagonal dominantes.
Como para armar $A^{(k-1)}$ ya aplicamos $(k-1)$ pasos de la Eliminación Gaussiana, se tiene que $C = 0$.
Entonces cuando se quiere aplicar un paso de Eliminación Gaussiana sobre ${A^{(k-1)}}_{2,2}$, dado que el elemento de la diagonal es no nulo, este paso se puede hacer sin pivotear la matriz, y utilizando el lema **??** se sigue que la matriz luego de este paso será diagonal dominante.

Solo resta probar que la fila $k+1$ de la matriz $A^{(k)}$ es no nula.
Si la fila k corresponde a uno de los puntos extremos de la pared del horno, es decir, si $1 \leq k \leq n$ ó $mn \leq k \leq (m+1)n$, tendrá un 1 en la diagonal y 0 en las demás posiciones. Entonces no cambia en ningún paso de la Eliminación Gaussiana.
Pero si la fila corresponde a una de los puntos internos de la pared, entonces como cada fila representa a una instancia de la ecuación de Laplace, se tiene que el elemento $a_{k+1,k+1+n-1}$ es el coeficiente de la temperatura $t_{j,k+1}$, que es no nulo.

II Efectividad de la factorización LU

Para resolver un sistema de ecuaciones cualquiera planteado como $Ax = b$, se puede usar el método de Eliminación Gaussiana o "Gauss", que tiene un tiempo de operación en el orden de $\mathcal{O}(n^3)$. Pero también se puede resolver usando la factorización LU, que son dos matrices tales que $A = LU$, donde L es triangular inferior y U es triangular superior. La forma de resolverlo así sería primero calcular un y tal que $Ly = b$ y luego resolver $Ux = y$. Esto toma $\mathcal{O}(n^2)$ operaciones.

Como vimos en el apéndice **??**, se puede aplicar Gauss sobre la matriz de nuestro sistema. Lo cual es condición suficiente para afirmar que tiene factorización $A = LU$ [**?**]. Que nos es muy útil cuando queremos resolver múltiples instancias de $At_i = b_i$ ya que la factorización LU se puede calcular aplicando la Eliminación Gaussiana una sola vez, y resolver cada instancia $LUt_i = b_i$ requiere muchas menos operaciones.
En la Figura **??** se puede confirmar que resolver una instancia del sistema con LU es definitivamente más rápido que resolverla con Gauss.

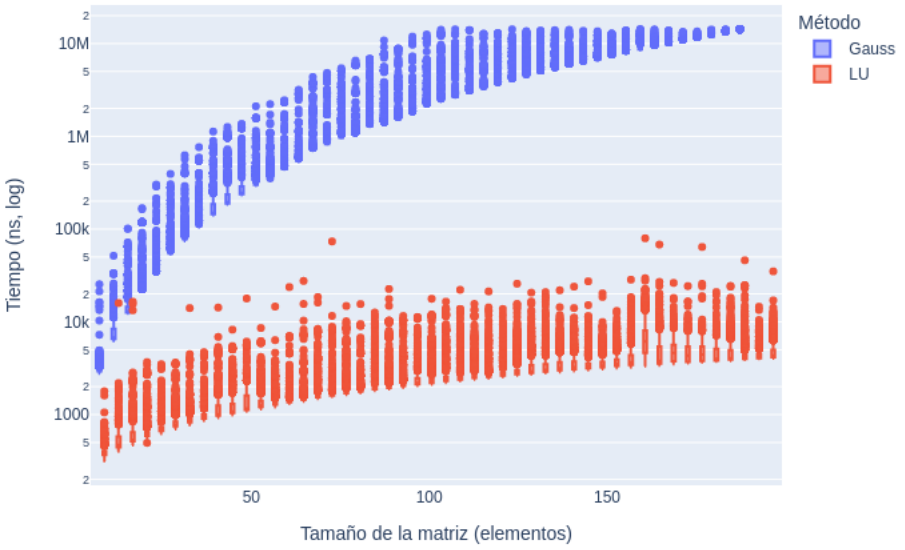

Fig. 10: Tiempo para resolver una instancia del sistema (1000 reps)

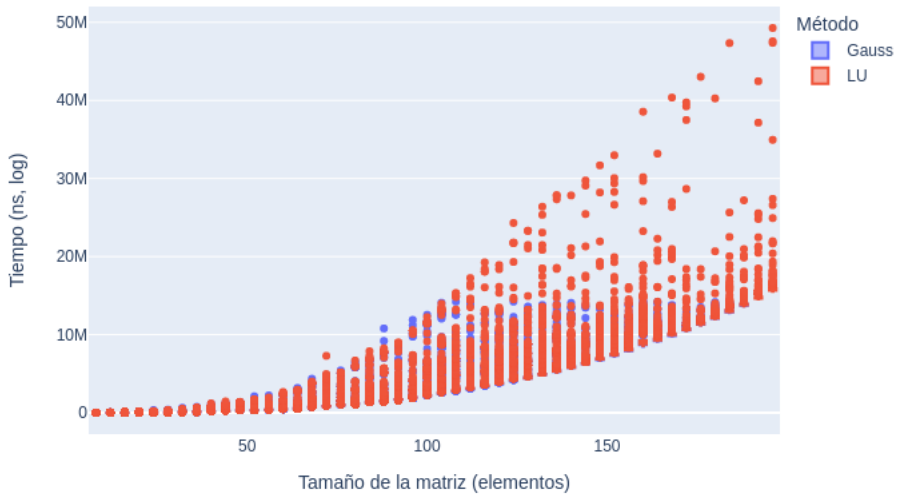

Fig. 11: Tiempo para resolver un sistema contando el tiempo para calcular LU (1000 reps)

Pero como mencionamos, para calcular la factorización hay que aplicar Gauss una vez, por lo que en realidad nos queda el resultado de la Figura **??** para resolver una instancia.
Es por esto que es importante revisar cuándo de verdad nos empieza a convenir usar la factorización LU. En la Figura **??** se ve que independiente del tamaño de la matriz, el tiempo usado para resolver una instancia del sistema es una fracción mínima del tiempo usado para calcular la factorización. Por lo que para dos o más instancias del sistema conviene calcular y usar la factorización LU (resaltado en la Figura **??**).
En la Figura **??** observamos que para resolver 9 instancias del sistema la diferencia de tiempo ya se vuelve muy notoria.

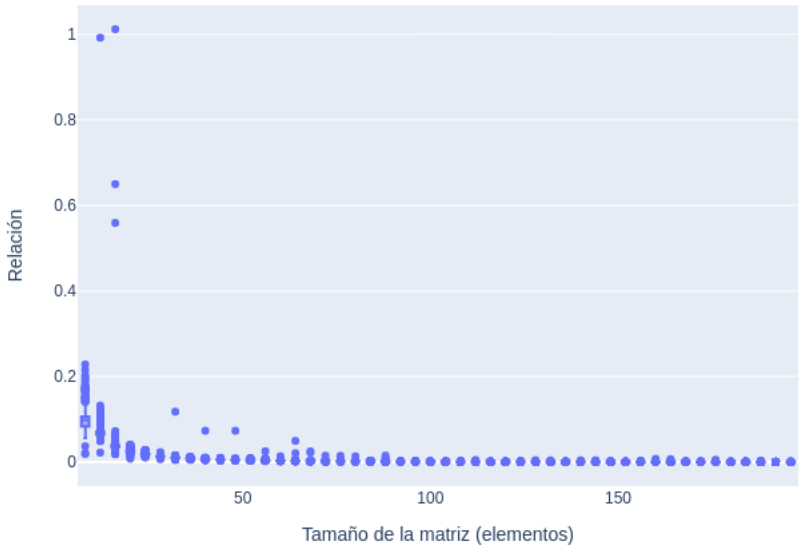

Fig. 12: Relación entre el tiempo para resolver el sistema y el tiempo para calcular la factorización LU (1000 reps)

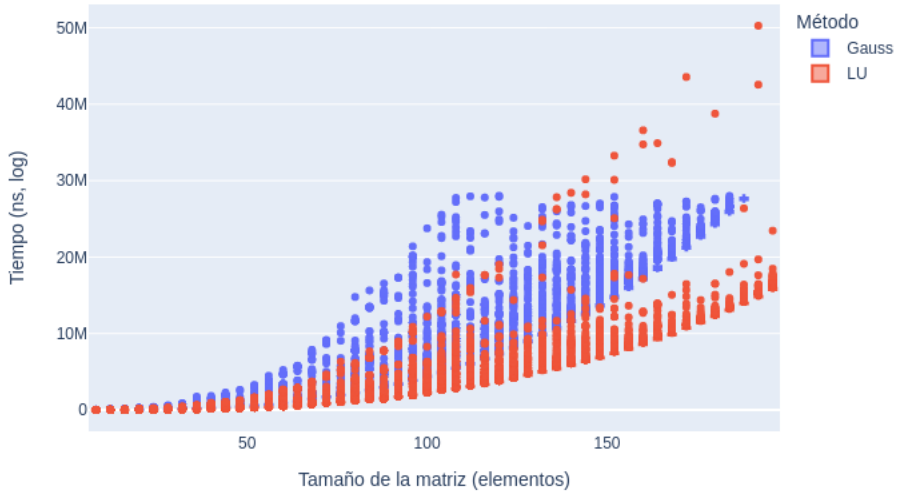

Fig. 13: Tiempo para resolver dos instancias del sistema teniendo en cuenta el tiempo de *LU* (1000 reps)

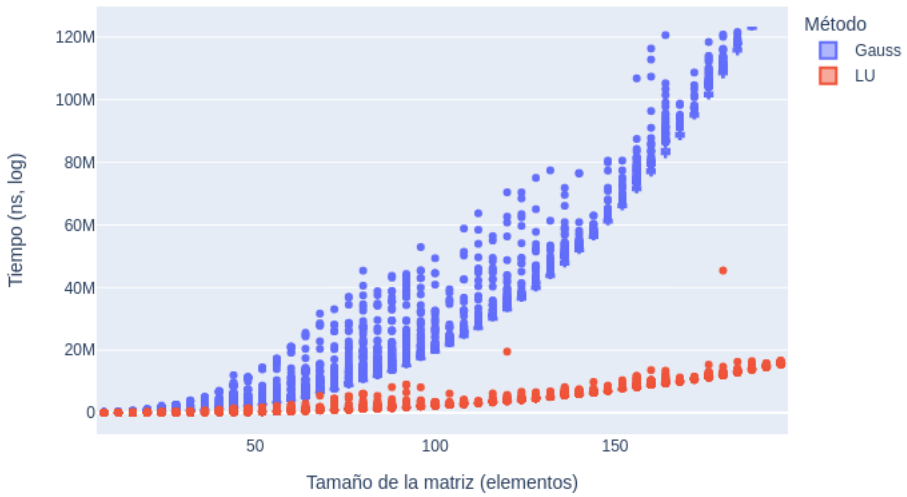

Fig. 14: Tiempo para resolver nueve instancias del sistema teniendo en cuenta el tiempo de LU (1000 reps)

Figuras

Bibliografía

[1] American Iron and Steel Institute (2005).

[2] R. Burden y J.D.Faires, Análisis numérico, International Thomson Editors, 2002.

[3] Temperatura de fusión de metales. `https://www.redproteger.com.ar/temp_fusion.htm`