DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

# TP1 - Trabajamos y nos divertimos...

April 16, 2022 Métodos Numéricos

**Grupo 8**


| Integrante | LU | Correo electrónico |
|---|---|---|
| Cappella Lewi, F. Galileo | 653/20 | galileocapp@gmail.com |
| Anachure, Juan Pablo | 99/16 | TODO@dc.uba.ar |





**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

Los altos hornos son un tipo de horno metalúrgico, usados para producir metales industriales. Están formado por una pared circular usualmente de acero y ladrillos refractarios [**?**] lo que permite fundir distintos metales en el centro. En la Figura **??** 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 [2]. 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 estudiamos una manera de calcular las temperaturas dentro de la pared, buscando poder estimar la posición de la isoterma.
Para ello utilizamos una discretización de la ecuación del calor de Laplace (Sección 2.1). Por lo que necesitamos conocer las dimensiones del horno (el radio interno y el radio externo), y las temperaturas medidas a diferentes ángulos en el interior y exterior del horno. Una vez conseguidos esos datos, armamos una matriz que nos permite estimar la temperatura en diferentes radios dentro de la la pared.

# 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 $(r, \theta)$ que representan al radio y el ángulo polar, cumple la siguiente ecuación: [2]

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

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:

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

Aunque para simplificar la notación vamos a nombrar cada coeficiente, por lo que nos queda:

$$0 = \alpha_r t_{j-1,k} + \beta_r t_{j,k} + \gamma_r t_{j+1,k} + \chi_r t_{j,k-1} + \chi_r t_{j,k+1} \tag{4}$$

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 ), y el resto corresponde a una instancia de la ecuación (4); $t \in \mathbb{R}^{(m+1)n}$ es el vector de incógntias 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 (5) mostramos un ejemplo de cómo queda el sistema con $r_i = 1,\ r_e = 3,\ mp1 = 3,\ n = 3,\ t_{i,1} = t_{i,2} = t_{i,3} = 1500,\ t_{e,1} = t_{e,2} = t_{e,3} = 150$.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & -\frac{2}{\pi^2} - \frac{3}{2} & \frac{1}{\pi^2} & \frac{1}{\pi^2} & 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{\pi^2} & -\frac{2}{\pi^2} - \frac{3}{2} & \frac{1}{\pi^2} & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{\pi^2} & \frac{1}{\pi^2} & -\frac{2}{\pi^2} - \frac{3}{2} & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} t_{1,1} \\ t_{1,2} \\ t_{1,3} \\ t_{2,1} \\ t_{2,2} \\ t_{2,3} \\ t_{3,1} \\ t_{3,2} \\ t_{3,3} \end{pmatrix} = \begin{pmatrix} 1500 \\ 1500 \\ 1500 \\ 0 \\ 0 \\ 0 \\ 150 \\ 150 \\ 150 \end{pmatrix} \tag{5}$$

Demostramos en el apéndice I que se puede aplicar la Eliminación Gausseana sin pivotear filas.

## 2.2 Implementación

Implementamos este sistema en `C++`. Primero programamos matrices usando vectores de la `STL`. Luego implementamos funciones que resuelvan sistemas de ecuaciones usando Eliminación Gausseana 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 1).
Todo el código se puede ver adjuntado al informe, junto con explicaciones de cómo usarlo.

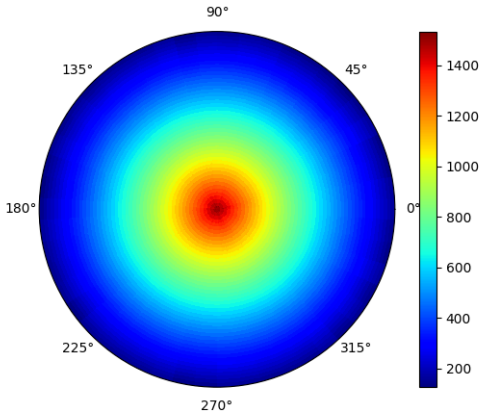


Fig. 1: Temperatura calculada dentro del horno

### 2.2.1 Optimización Matriz "en banda"

Por cómo construimos la matriz, luego aplicar la eliminación Gausseana, nos queda una matriz "en banda", lo que en este caso significa que en todas las filas sólo tiene máximo $n+1$ elementos diferentes a cero.
Por lo que podemos aprovechar y sólo operar sobre esos valores, reduciendo tanto la memoria usada por el programa, como la cantidad de operaciones hechas para resolver el sistema.
En la figura **??** se puede apreciar el espacio ahorrado por esta optimización.
Y en la figura **??** se nota que esta optimización no mejoró mucho el tiempo de operación. Esto se debe a que el órden del sistema no dejó de ser $\mathcal{O}(((m+1)n)^3)$.

# 3 Estimando la isoterma

En la sección 1 planteamos que la razón principal de querer resolver este problema es querer calcular la posición de la isoterma de $500^{\text{o}}$C, ya que si está "muy cerca" (ver sección 3.1) la estructura del horno estaría en riesgo.

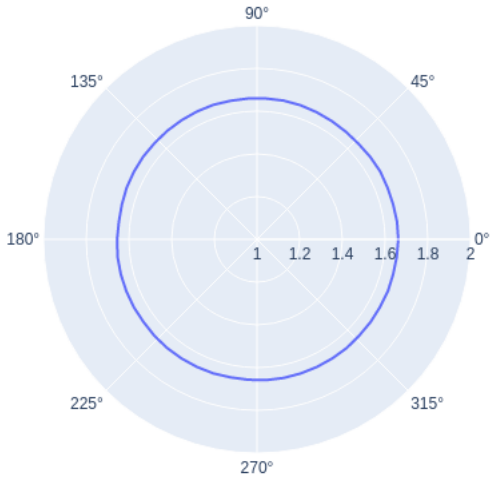


Fig. 2: Posición estimada de la isoterma

Para encontrarla buscamos dos puntos dentro de un mismo ángulo que "rodeen" la temperatura buscada, y luego aproximamos linealmente la posición donde se encuentra entre estos dos puntos. En la figura **??** representamos cómo hacemos esto.

### 3.1 Midiendo la Peligrosidad

Una vez encontrada la isoterma, es importante decidir si el horno está en un estado peligroso o seguro.
Pero 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 del 75% de la pared hay poco margen de error, ya que hasta ese punto la isoterma se mueve rápido hacia el exterior.

## 4 Experimentación

### 4.1 Resultados esperados

La cátedra nos proveyó con una serie de tests, con su input y un output esperado. Al correr nuestro programa con este input nos encontramos con que los resultados conseguidos no eran iguales a los esperados (ver figura 3).
Para confirmar 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 al rededor del resultado esperado dentro del cual, si se encuentran nuestros resultados, es aceptable y esperado por los 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 sumando desde afuera hacia adentro.

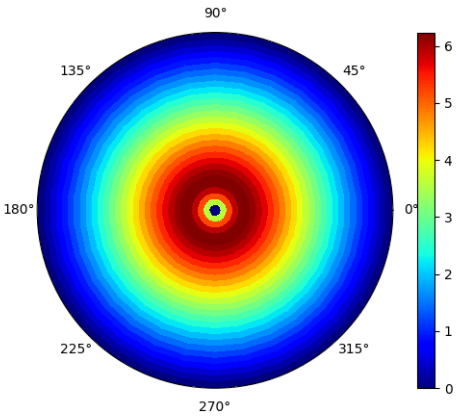


Fig. 3: Diferencia entre las temperaturas esperadas y las calculadas

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`. Y muy importante es 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), hay errores de redondeo [3].
Para experimentar sobre esto, armamos un sistema que generara coeficientes "muy chiquitos" ($\alpha, \beta, \gamma, \chi \leq 10^{-6}$) con temperaturas muy altas ($T_i, T_e \geq 10^6$) y comparamos los resultados entre usar `long double` que usa 128-bits para guardar los números y usar `float` que usa 32-bits.

En la figura 4 se pueden ver las diferencias causadas. Como vimos en la sección 4.1, el algoritmo que usamos para resolver el sistema causa que el error crezca hacia el centro del horno. También por razones similares a lo visto en la sección 4.1 estos errores no causan que la isoterma estimada se encuentre en un lugar muy diferente.

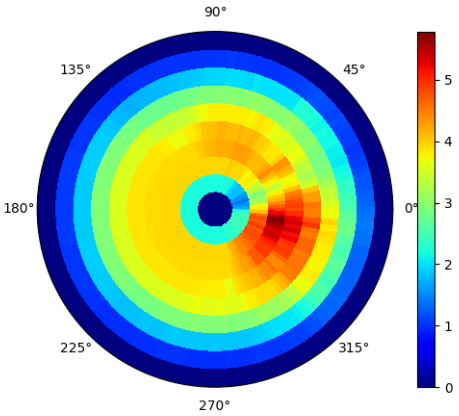


Fig. 4: Diferencia entre las temperaturas calculadas usando `long double` y `float`

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

Como diferentes metales necesitan diferentes temperaturas para ser fundidos, nos interesó ver la posición estimada de la isoterma para algunos de los metales más trabajados.
En la figura 5 se puede ver la posición de la isoterma en base a la temperatura interna, tiene marcadas en orden las temperaturas del aluminio, el cobre, el hierro, y el tungsteno.

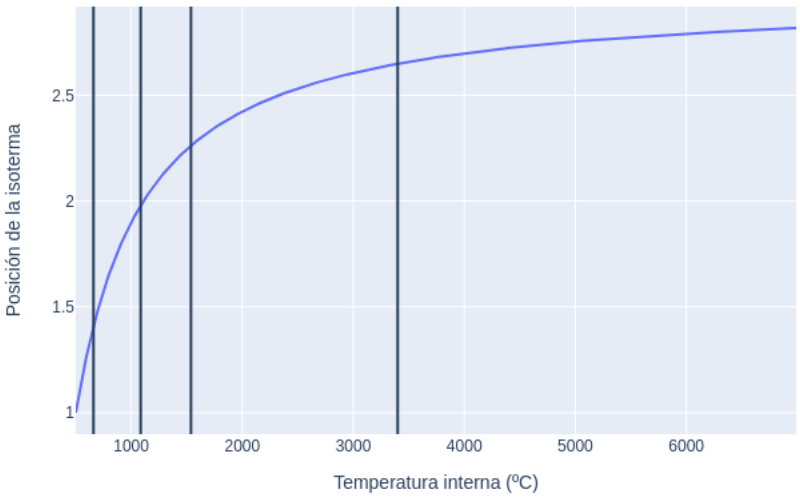


Fig. 5: Distancia de la isoterma al exterior del horno

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 radios. Asumimos a las temperaturas internas y externas fijadas en 1500ºC y 150ºC respectivamente.
Como se ve en la figura 6, la posición varía bastante en base a la cantidad de radios.

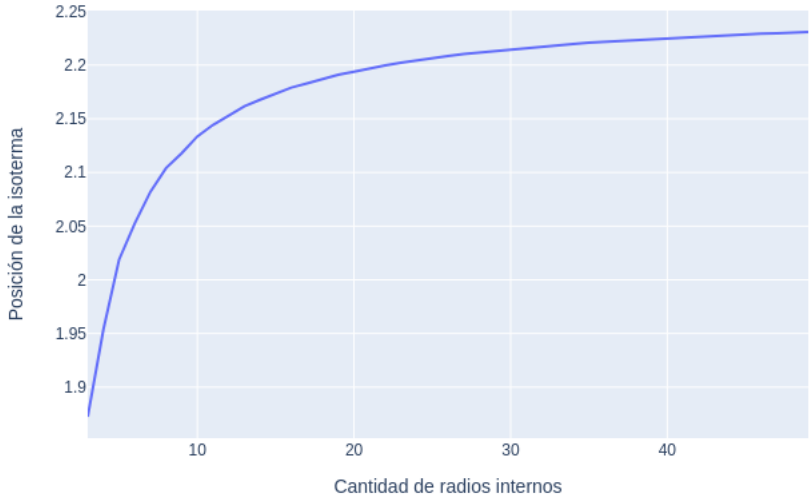


Fig. 6: Posición estimada de la isoterma

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

# 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 Gausseana sin reordenar filas.

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

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

**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 Gausseana (sin pivoteo) sobre A. Entonces $A^{(1)}$ es diagonal dominante.*

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

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

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 Gausseana:

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

□

Ahora se tiene que luego de realizar un paso de Eliminación Gausseana 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 Gausseana es aplicable y se puede utilizar sin pivoteo. Así solamente hay que ver que para los puntos del medio del horno puede aplicarse Eliminación Gausseana y ademas 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 Gausseana. Dado que $a_{1,2} = 0$ entonces si aplicamos un paso de Eliminación Gausseanaen 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 Gausseana.

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

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

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

Con $A_{1,1}^{(k-1)} \in \mathbb{R}^{(k-1)\times(k-1)}$.
Luego de haber probado lo anterior, sabemos que $A_{1,1}^{(k-1)}$ y $A_{2,2}^{(k-1)}$ son matrices diagonal dominantes.
Como para armar $A^{(k-1)}$ ya aplicamos $(k-1)$ pasos de la Eliminación Gausseana, se tiene que $C = 0$.
Entonces cuando se quiere aplicar un paso de Eliminación Gaussiana sobre $A_{2,2}^{(k-1)}$, dado que el elemento de la diagonal es no nulo, este paso se puede hacer sin pivotear la matriz, y utilizando el lema 1 se sigue que la matriz luego de este paso sera 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 Gausseana.
Pero si la fila corresponde a una de los puntos internos de la pared, entonces ya que 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 Gausseana 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 I, se puede aplicar Gauss sobre la matriz de nuestro sistema. Lo cual es condición suficiente para afirmar que tiene factorización $A = LU$ [4]. 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 Gausseana una sola vez con, y resolver cada instancia $LUt_i = b_i$ requiere muchas menos operaciones.
En la figura 7 se puede confirmar que resolver una instancia del sistema con LU es definitivamente más rápido que resolverla con Gauss.

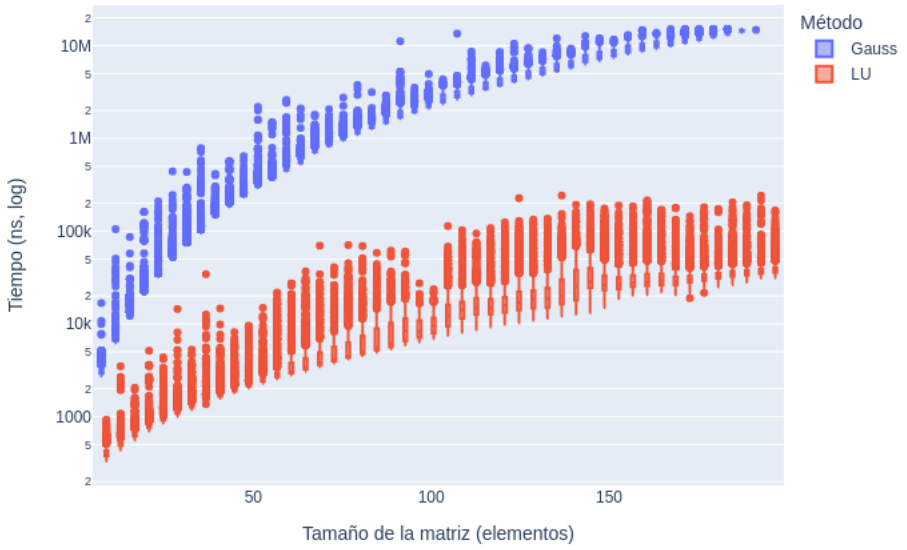


Fig. 7: Tiempo para resolver una instancia del sistema (1000 reps)

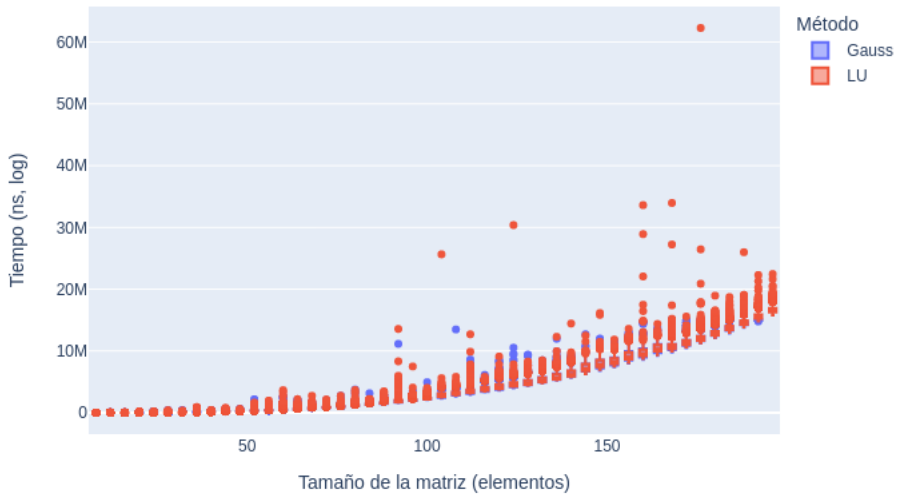


Fig. 8: Tiempo para resolver un sistema contando el tiempo para calcular $LU$ (1000 reps)

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 8 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 9 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 10).
En la figura 11 resaltamos que para resolver 9 instancias del sistema la diferencia de tiempo ya se vuelve muy notoria.

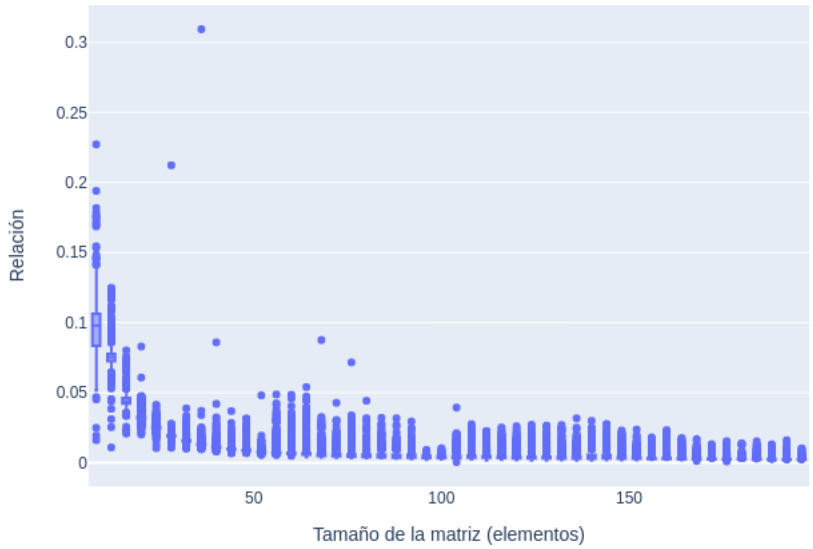


Fig. 9: Relación entre el tiempo para resolver el sistema y el tiempo para calcular la factorización LU (1000 reps)

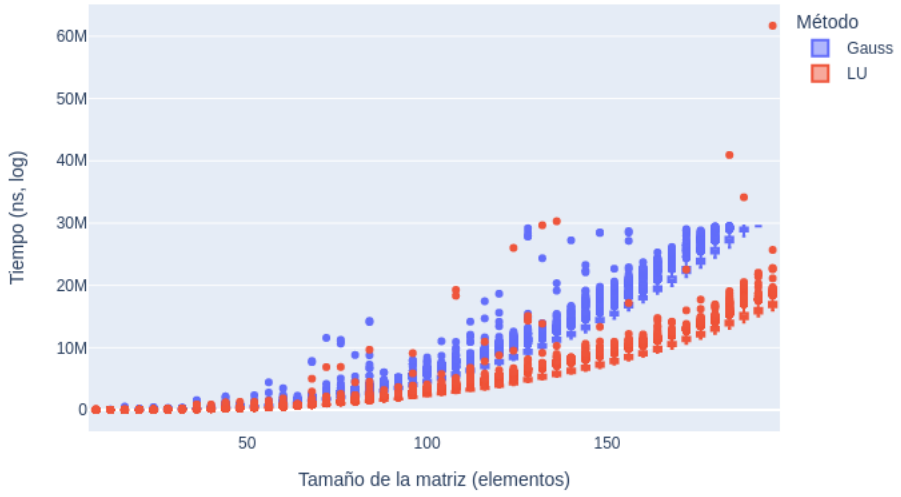


Fig. 10: Tiempo para resolver dos instancias del sistema teniendo en cuenta el tiempo de *LU* (1000 reps)

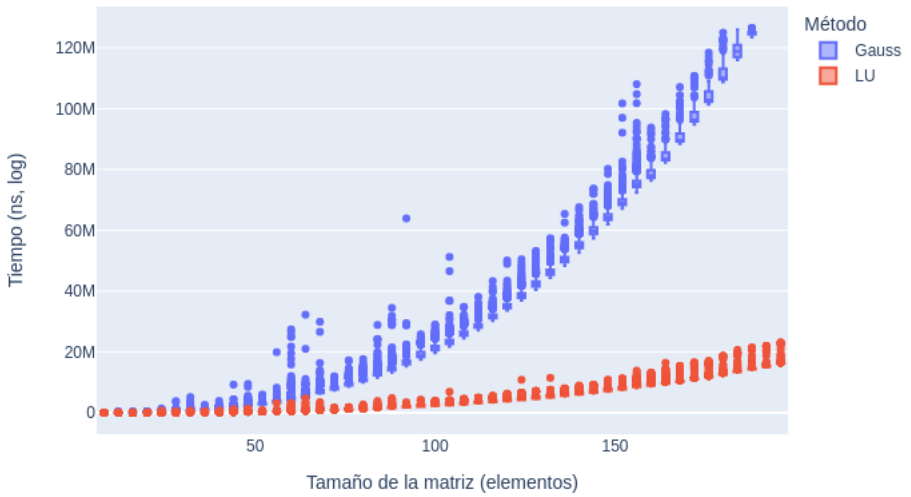


Fig. 11: Tiempo para resolver nueve instancias del sistema teniendo en cuenta el tiempo de $LU$ (1000 reps)

# Figuras



# Bibliografía


[1] American Iron and Steel Institute (2005), How A Blast Furnace Works. steel.org

[2] Cátedra de Métodos Numéricos (2022), Consigna del TP1

[3] TODO:

[4] Cátedra de Métodos Numéricos (2022), Clases Teóricas