# Métodos Numéricos

Trabajo Práctico 2

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

## Resolución de Sistemas de Ecuaciones Lineales



| Integrante | LU | Correo electrónico |
|---|---|---|
| Aronson, Alex | 443/08 | alexaronson@gmail.com |
| Nahabedian, Leandro | 250/08 | leanahabedian@hotmail.com |
| Ravasi, Nicolás | 53/08 | nravasi@gmail.com |

**Resumen:** En este trabajo analizaremos el funcionamiento de un horno, calculando la temperatura en cualquier punto dentro del mismo, y en particular la isoterma 400º C. Para esto realizamos un modelaje apropiado, reduciendo dicho problema a un sistema de ecuaciones lineales. Resolviendo el mismo y utilizando interpolación lineal, logramos hallar la isoterma. Además, incluimos en este informe, resultados comparativos que demuestran como es el comportamiento de nuestro algoritmo mediante gráficos.

**Palabras clave: eliminacion gaussiana isoterma interpolación lineal sistema de ecuaciones lineales**

# Contents

# 1 Introducción

Cuando nos encontramos frente a un problema en que trabajamos con ecuaciones en derivadas parciales, nos enfrentamos a analizar al relación que tiene una función con varias variables independientes y la derivada de la misma.

Generalmente se lidia con problemas de valor inicial, definidos por preguntas como cuáles son las variables dependientes que se propagan en el tiempo o cómo es la evolución de ellas, y también trabajamos con problema de valor de borde, definida con preguntas como cuáles son las variables, qué ecuaciones se satisfacen en el interior de una región de interes o qué ecuaciones se satisfacen por los puntos del borde de la región de interes. Como todas las condiciones de borde deben ser satisfechas simulateneamente, nos encontramos a la hora de resolver esto con un sistema de ecuaciones lineales, tema que hemos incorporado desde que comenzamos la materia.

Conociendo todo esto fuimos enfrentados al problema de estimar la isoterma 400º dentro de un horno de acero cilíndrico, al que le conocíamos que su borde externo es un círculo, mientras que su borden interno no tiene forma circular necesariamente.

Conociendo los valores de su pared interna y contando con un termómetro de su pared externa para medir los valores (que varían entre 50 y 200), armaremos al igual que lo hacen las ecuaciones diferenciales un sistema de ecuaciones lineales, luego discretizando el sistema resolveremos computacionalmente el problema de la manera que contamos a lo largo de este trabajo.

# 2 Desarrollo

## 2.1 Modelo del problema

Inicialmente tratamos de calcular la temperatura de cada parte del horno utilizando un modelado. El mismo se basa en dar una ecuación por cada interseccion entre radios y circulos alrededor del centro del horno.

Al inciar a pensar este problema, analizamos el funcionamiento del horno y la formula:

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

Donde podemos aproximar cada termino usando estas otras formulas:

$$\begin{aligned}
\frac{\partial^2 T(r,\theta)}{\partial r^2}(r_j,\theta_k) &\cong \frac{t_{j-1,k} - 2t_{jk} + t_{j+1,k}}{(\Delta r)^2} \\
\frac{\partial T(r,\theta)}{\partial r}(r_j,\theta_k) &\cong \frac{t_{jk} - t_{j-1,k}}{\Delta r} \\
\frac{\partial^2 T(r,\theta)}{\partial \theta^2}(r_j,\theta_k) &\cong \frac{t_{j,k-1} - 2t_{jk} + t_{j,k+1}}{(\Delta \theta)^2}
\end{aligned}$$

Luego la primer formula quedaria como

$$\frac{t_{j-1,k} - 2t_{jk} + t_{j+1,k}}{(\Delta r)^2} + \frac{1}{r}\frac{t_{jk} - t_{j-1,k}}{\Delta r} + \frac{1}{r^2}\frac{t_{j,k-1} - 2t_{jk} + t_{j,k+1}}{(\Delta \theta)^2} \cong 0$$

Ahora para analizar el funcionamiento de esta ecuación decidimos tomar un caso particular de un horno que tiene 3 radios y 3 circulos, para simplificar el análisis del problema. Las intersecciones del circulo más pegado al centro con los radios conocemos su valor, 1500 grados y las del círculo más alejado del centro lo sabemos porque son dato de entrada. En este ejemplo vamos a decir que tenemos los puntos $a, b, c$ ; $x, y, z$ ; $o, p, q$. donde $a$, $x$ y $o$ son los puntos donde se corta cada radio con el círculo mís interno al centro del horno y $c$, $z$, $q$ son los puntos que corresponden al círculo mas alejado del centro del horno.

Luego:
$t_q = 100$
$t_z = 50$
$t_c = 150$
$t_x, t_a, t_o = 1500$

Para hallar $t_b$: $t_b = t_{j,k}$

$$\frac{t_a - 2t_b + t_c}{(\Delta r)^2} + \frac{1}{r}\frac{t_b - t_a}{\Delta r} + \frac{1}{r^2}\frac{t_y - 2t_b + t_p}{(\Delta \theta)^2} \cong 0 \quad (1)$$

$t_y$:$t_y = t_{j,k}$

$$\frac{t_x - 2t_y + t_z}{(\Delta r)^2} + \frac{1}{r}\frac{t_y - t_x}{\Delta r} + \frac{1}{r^2}\frac{t_p - 2t_y + t_b}{(\Delta \theta)^2} \cong 0 \quad (2)$$

$t_p$:$t_p = t_{j,k}$

$$\frac{t_o - 2t_p + t_q}{(\Delta r)^2} + \frac{1}{r}\frac{t_p - t_o}{\Delta r} + \frac{1}{r^2}\frac{t_b - 2t_p + t_y}{(\Delta\theta)^2} \quad \cong \quad 0 \tag{3}$$

reemplazando $t_q$, $t_z$, $t_c$, $t_x$, $t_a$, $t_o$ por los valores que dimos anteriormente en la formula 1

$$\frac{1500 - 2t_b + 150}{25} + \frac{1}{r}\frac{t_b - 100}{\Delta r} + \frac{1}{r^2}\frac{t_y - 2t_b + t_p}{(\frac{2}{3}\pi)^2} \quad \cong \quad 0$$

Lo que queda por hacer es ordenar estas ecuaciones para que nos quede una sumatoria con productos de una incógnita por datos de la siguiente manera.

$$\frac{t_a}{(\Delta r)^2} - \frac{t_a}{r(\Delta r)} - \frac{2t_b}{(\Delta r)^2} + \frac{t_b}{r(\Delta r)} - \frac{2t_b}{r^2(\Delta\theta)^2} + \frac{t_c}{(\Delta r)^2} + \frac{t_y}{r^2(\Delta\theta)^2} + \frac{t_p}{r^2(\Delta\theta)^2} \quad \cong \quad 0$$

$$\frac{r - \Delta r}{r((\Delta r)^2)}t_a + \frac{-2r^2(\Delta\theta)^2 + r(\Delta r)(\Delta\theta)^2 - 2(\Delta r)^2}{r^2(\Delta r)^2(\Delta\theta)^2}t_b + \frac{1}{(\Delta r)^2}t_c + \frac{1}{r^2(\Delta\theta)^2}t_y + \frac{1}{r^2(\Delta\theta)^2}t_p \quad \cong \quad 0$$

Nótese que $t_a$ y $t_c$ son dato para nuestro caso particular pero no lo serán si tenemos más círculos en el horno y es una ecuación para algún punto en el medio del horno, es decir, no adyacente a algún borde. Entonces, tenemos una forma genérica de escribir una ecuación para un punto cualquiera del horno, instanciando los siguientes valores

$$\begin{aligned} t_b = t_{j,k} \quad \Rightarrow \quad & t_a = t_{j-1,k} \\ & t_c = t_{j+1,k} \\ & t_y = t_{j,k-1} \\ & t_p = t_{j,k+1} \end{aligned}$$

$$\frac{r - \Delta r}{r((\Delta r)^2)}t_{j-1,k} + \frac{-2r^2(\Delta\theta)^2 + r(\Delta r)(\Delta\theta)^2 - 2(\Delta r)^2}{r^2(\Delta r)^2(\Delta\theta)^2}t_{j,k} + \frac{1}{(\Delta r)^2}t_{j+1,k} + \frac{1}{r^2(\Delta\theta)^2}t_{j,k-1} + \frac{1}{r^2(\Delta\theta)^2}t_{j,k+1} \quad \cong \quad 0$$

Por lo tanto, tenemos una forma de representar cada intersección por una ecuación con 5 incógnitas. Ahora que tenemos dichas ecuaciones podemos crearnos un sistema de ecuaciones lineales $Ax = b$ para un horno donde hay $n$ radios y $m$ circulos donde $A \in \mathbb{R}^{n*mxn*m}$ es una matriz que representa los datos que estan asociados a cada incognita. El vector $x \in \mathbb{R}^{n*m}$ es el vector de incógnitas y el vector $b \in \mathbb{R}^{n*m}$ representa el valor de la temperatura en el caso de los puntos en lo que es conocida (bordes interior y exterior) y 0 en el caso en que es desconocida.

## 2.2 Inicialización del sistema de ecuaciones

Una vez obtenida la manera de encarar el problema, el primer paso fue lograr que nuestro programa inicializara la matriz de esta forma para poder proceder a resolverla. Para esto primero leemos la entrada desde un archivo de texto (se puede ingresar desde consola, pero es más rápido pasar un archivo de texto como entrada). Una vez hecho esto, calculamos cuántas ecuaciones tendrá la matriz, luego seteamos los datos que ya conocemos (las temperaturas de los bordes) en el vector $b$.

Hecho esto, falta calcular los coeficientes para cada ecuación. Para esto, sabemos que cada ecuación identifica unívocamente a un punto, de la manera que disponemos la matriz, los puntos se ordernan desde el más exterior, en orden (según cómo entraron de parámetro) pasando por cada punto que tiene ese radio, luego los puntos que se ubican un radio más adentro, y así sucesivamente hasta terminar en los puntos del borde interior del horno. De esta manera, para obtener el índice en la matriz que le corresponde al punto, sabemos que si el punto tiene coordenadas $(x, y)$, con $x$ el número de radio al que pertenece e $y$ el número de ángulo, entonces ese punto tiene el índice $x * m + y$ con $m$ la cantidad de ángulos.

Luego, basta aplicar las cuentas que resumimos anteriormente, para cada ecuación hay cinco coeficientes a setear:

$$\begin{aligned} m_{j-1,k} &:= \frac{r - \Delta r}{r((\Delta r)^2)} \\ m_{j,k} &:= \frac{-2r^2(\Delta\theta)^2 + r(\Delta r)(\Delta\theta)^2 - 2(\Delta r)^2}{r^2(\Delta r)^2(\Delta\theta)^2} \\ m_{j+1,k} &:= \frac{1}{(\Delta r)^2} \\ m_{j,k-1} &:= \frac{1}{r^2(\Delta\theta)^2} \\ m_{j,k+1} &:= \frac{1}{r^2(\Delta\theta)^2} \end{aligned}$$

donde $\Delta\theta$ y $\Delta r$ son constantes puesto que todos los círculos y los ángulos son equidistantes, por lo que se pueden calcular al principio de la ejecución y usar como datos.

Haciendo esto para cada ecuación nos queda una matriz bastante esparsa, donde las primeras $m$ filas y las últimas $m$ filas son como si fuera una matriz identidad y en las otras filas nunca hay más de 5 valores distintos de 0.

## 2.3 Triangulación de la matriz

Para resolver este sistema decidimos originalmente habíamos decidido utilizar factorización $QR$ para aprovechar la esparcidad de la matriz, es decir, hallar dos matrices $Q, R \in \mathbb{R}^{n*mxn*m}$ tal que $A = Q * R$ donde $Q$ es una matriz ortogonal y $R$ triangular superior. Se puede ver que $QRx = b$ se puede transcribir como $Rx = Q^{-1}b$ con lo cual con sólo encontrar la factorizacion $QR$ el problema estaría casi resuelto, ya que la inversa de $Q$ al ser ortogonal es $Q^t$.

Para realizar la factorización habíamos decidido usar el método de Householder, ya que el método de Givens tenia la complejidad de utilizar funciones trigonometricas que suelen darnos valores con mucha cantidad de decimales. Al representar estas funciones en la computadora hubiéramos necesitado truncar o redondear estos decimales perdiendo precisión.

Al tratar de codear el método de Householder encontramos muchos problemas ya que la implementación resultó muy compleja, y si bien llegamos a terminarlo y obtener una factorización, contrastando los resultados obtenidos contra Matlab evidenció que estos no eran correctos, y no nos fue posible encontrar en dónde radicaba el error. [1]

Dado que no pudimos hacer la factorización con $QR$, decidimos resolver el sistema mediante eliminación gaussiana, ya que consideramos que la implementación sería mucho más sencilla y además siendo la matriz esparsa, no habría que realizar demasiadas eliminaciones ya que la mayor parte de los datos ya iban a estar seteados en 0 y no iba a ser necesario hacer tantas cuentas.

La idea de este método es que por cada iteracion $i$ se complete la columna $i$-ésima con ceros desde la posicion $(i+1, i)$ hasta la posicion $(n*m, i)$. Al completar todas las iteraciones se obtiene una matriz triangulada.

¿Como logramos esto? Lo realizamos de la siguente manera. Primero debemos asegurar que en la posicion $(i, i)$ no haya un cero para el paso $i$. El $(1, 1)$ siempre vale 1, por la manera de construir la matriz, asi que podemos empezar a ejecutar el algoritmo que funciona de la siguiente manera:

1. Obtenemos ceros debajo del $(1, 1)$. Hacemos una resta de la fila (incluyendo el vector $b$) que queremos poner en cero con la fila 1 multiplicada con $m_{i,1}/m_{1,1}$ para lograr el cero. Ver cuenta en Apendice B, sección 6.2.2
2. Para todos los otros pasos debemos chequear que la posicion $(i, i)$ sea distinta de 0.
3. Si es cero intercambiamos esa fila con la primera que esté abajo que no tiene 0 en esa columna (si no hay ninguna, la matriz no es inversible y por lo tanto, no se puede resolver el sistema) y luego seguimos al paso 4
4. Si no es 0, repetimos la cuenta del paso 1 con la submatriz que se formo al colocar en 0 toda la primer columna menos el primer elemento, es decir, la matriz anterior sin contar las filas desde la primera hasta la $i$-esima ni la columna 1 hasta lai-esima.

Luego de realizar todos estos pasos la matriz inicial $M$ se transforma en una matriz $T$ donde T es triangular superior, debido a que el algoritmo se encargo de poner en 0 todos los valores de la matriz cuyos indices cumplen que $j < i$ donde $i$ es la fila y $j$ es la columna.

## 2.4 Resolución del sistema triangulado

Una vez triangulada la matriz, es fácil resolver el sistema mediante sustitución hacia atrás, para esto, obviamos las últimas $m$ filas (por la manera que está construida la matriz, las ecuaciones siempre quedan $1 * x_i = 1500$), y calculamos el valor de $t_{n-m}$ restando los coeficientes de las posiciones mayores a $n-m$ multiplicado por el valor de la temperatura (en este caso, 1500) al valor de b$n-m$. Esto nos da el valor de $t_{n-m}$, lo podemos setear en nuestro vector que contiene las temperaturas finales, y proceder hacia arriba, haremos el mismo proceso reemplazando con las temperaturas obtenidas hasta llegar hasta arriba.

Una vez obtenido el vector con las temperaturas, debemos calcular en qué posición estimamos que la temperatura vale 400º para cada ángulo. Para hacer esto, pasamos el vector a una matriz de $\mathbb{R}^{nxm}$, donde cada columna representa las temperaturas correspondientes a un ángulo determinado a lo largo de los diferentes puntos. Por lo tanto, podemos estimar dónde se ubicaría el punto de 400º encontrando los dos valores más cercanos al mismo, uno que esté por encima y uno que esté por debajo. Al tener estos dos valores, sabemos por la ecuación del calor, que el punto con la temperatura deseada está en algún lugar entre esos dos.

[1] Adjuntamos el código desarrollado en la carpeta entrega/src como *QR(no anda).cpp*

Para obtener ese punto, vamos a usar interpolación. Tal como lo dice la página que consultamos (seccion 7), para estimar el valor de una función desconocida teniendo dos valores de la misma uno por encima y uno por debajo, usamos la función

$$d_{400} = d_1 + \frac{400 - t_1}{t_2 - t_1} * (d_2 - d_1)$$

donde $d_1$ y $t_1$ representan a la distancia y temperatura respectivamente del primer punto (que está más cercano al centro) y $d_2$ y $t_2$ a la distancia y temperatura del punto más cercano al exterior del horno.

Con esta ecuación pudimos estimar el punto en el que la temperatura es $400^o$ para ese ángulo, haciéndolo para todos los ángulos tenemos la isoterma.

# 3 Resultados

A continuación incluiremos graficos comparativos cuyo objetivo es el de mostrar como funciona nuestro algoritmo en situaciones estándar y especiales que creemos merecen especial consideración. En todos los casos se usó un radio interior igual a 0.1 y un radio exterior igual a 1

## 3.1 Comparación entre discretizaciones con diferentes cantidades de ángulos

En este caso hicimos dos gráficos, para el primero establecimos la cantidad de ángulos fija en 10, y fuimos variando la cantidad de círculos de 20 en 20, desde 5 hasta 85. Las 10 temperaturas exteriores las generamos de manera aleatoria en una página de internet (Sección 7), obviamente teniendo en cuenta las restricciones que se aplican a las mismas.

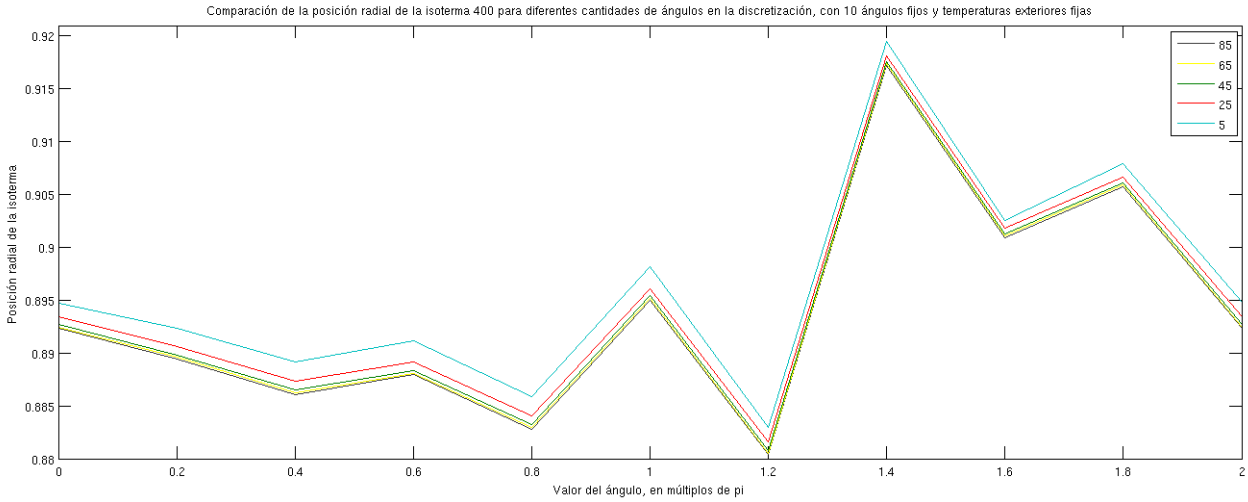


**Gráfico 1.1, comparación de la posición radial de la isoterma de 400º para diferentes cantidades de ángulos en la discretización, con 10 ángulos fijos y temperaturas exteriores fijas aleatorias. El eje Y empieza en 0.88 porque no pudimos introducir un corte de escala en MATLAB**

Luego graficamos otro set de datos donde nuevamente dejamos la cantidad de ángulos fija, esta vez en 30, usando las mismas variaciones de cantidades de círculos que en el experimento anterior. Nuevamente, las temperaturas exteriores fueron generadas de manera aleatoria por *random.org*

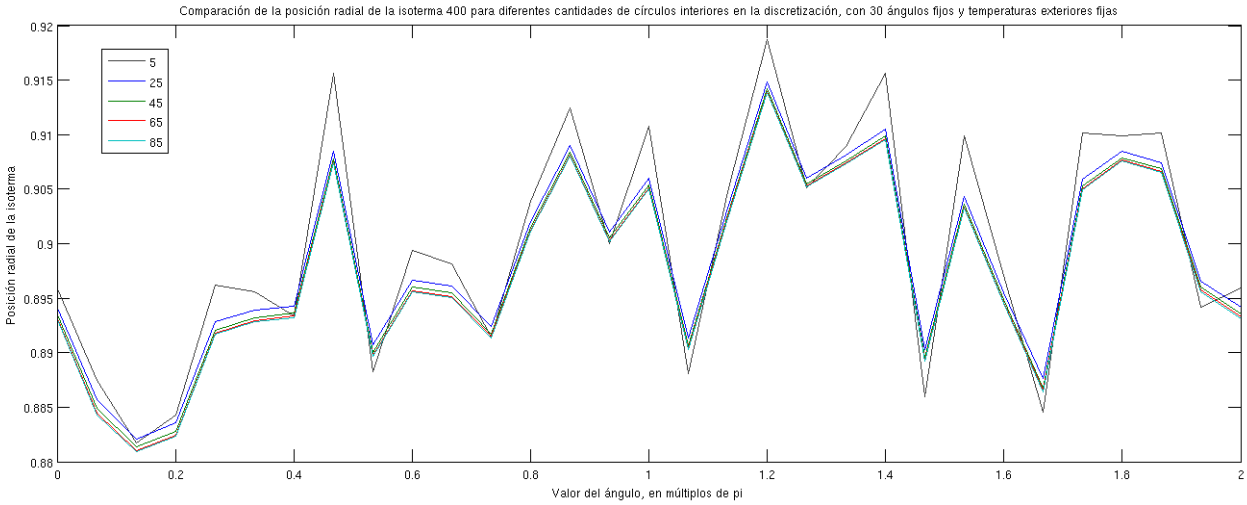


**Gráfico 1.2, comparación de la posición radial de la isoterma de 400º para diferentes cantidades de ángulos en la discretización, con 30 ángulos fijos y temperaturas exteriores fijas aleatorias. El eje Y empieza en 0.88 porque no pudimos introducir un corte de escala en MATLAB**

## 3.2 Caso especial

Nos pareció interesante estudiar el comportamiento de nuestro programa en un caso especial, esto es, en el que temperaturas exteriores consecutivas alternan entre 50 y 200 constantemente. Esto lo comparamos con un set de temperaturas exteriores generado al azar por *random.org*, pero con la restricción de que todas estuvieran entre 90 y 100, a manera de asegurar que la varianza sea menor a 10. En ambos casos se usaron 50 ángulos y 50 círculos.

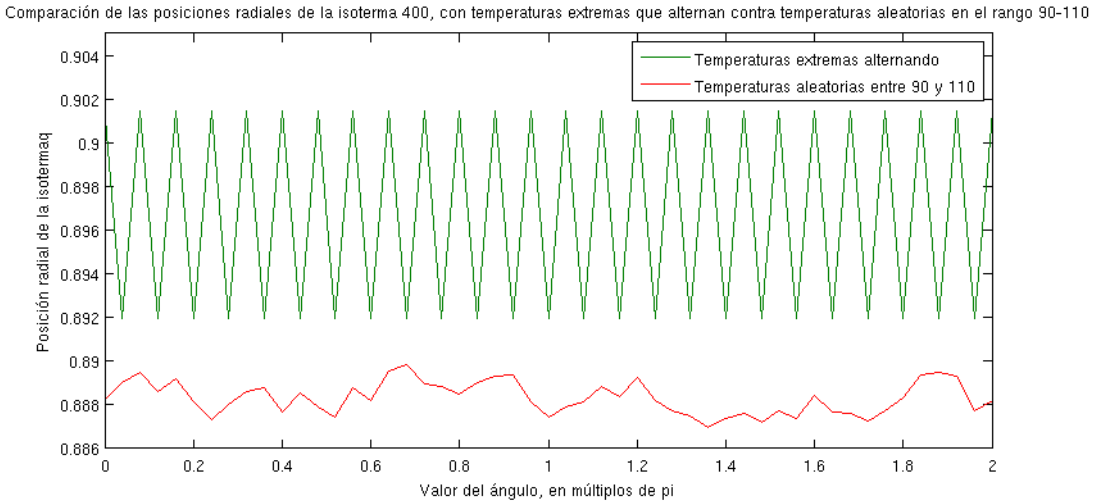


**Gráfico 2.1, comparación de las posicionen radiales de la isoterma 400 con temperaturas que alternan contra temperaturas aleatorias en el rango 90-100, en ambos casos usando 50 ángulos y 50 círculos. El eje Y empieza en 0.886 porque no pudimos introducir un corte de escala en MATLAB**

## 3.3 Tiempo transcurrido

Para este caso analizamos el tiempo que demora efectuar todos los cálculos para diferentes discretizaciones del problema. Para esto, empezamos con 5 ángulos y 5 círculos e incrementamos de 5 en 5 ambas cantidades (con lo cual el número de ecuaciones crece cuadráticamente), y usamos un set de temperaturas exteriores generado al azar por *random.org* Ante cada entrada del programa usamos la función *time* de UNIX y extrajimos el valor que nos devolvía del campo *real* para hacer este gŕafico. Tratamos de que todas las corridas se llevaran a cabo en un entorno similar, con los mismos procesos corriendo, de manera tal que eso no afecte los resultados obtenidos

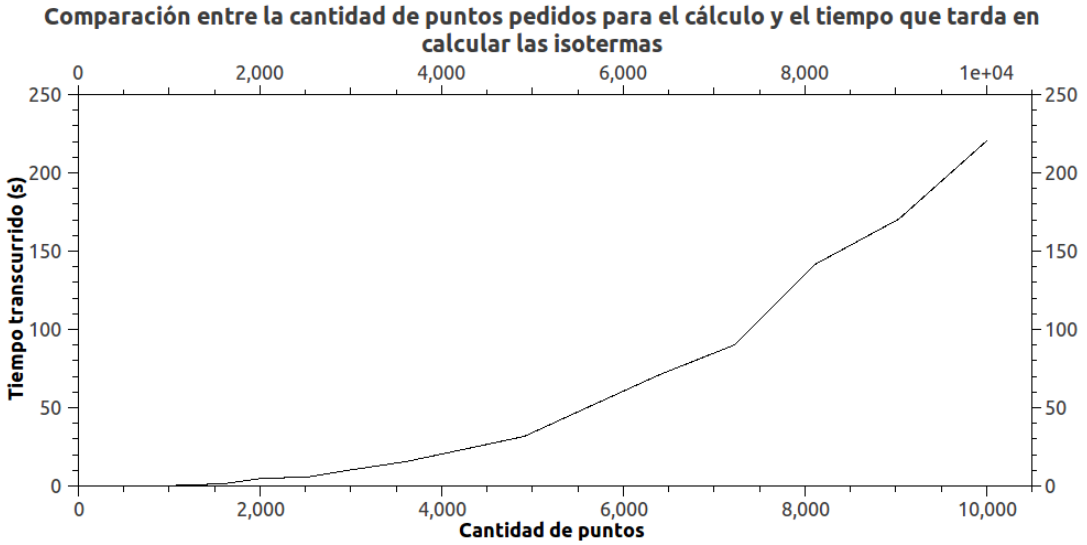


**Gráfico 3.1, Gráfico del tiempo insumido por el programa con diferentes cantidades de puntos a analizar**

# 4 Discusión

Como se puede observar en los gráficos 1.1 y 1.2, las curvas que grafican la isoterma de los diferentes puntos tienden a acercarse más y mas, con la de 5 círculos estando más alejada y ya las de 45, 65 y 85 siendo bastante más parecidas. Esto significa que la discretización es mucho más precisa, y que el valor verdadero (el que se obtendría si tuviéramos la función de la temperatura para ese radio) está siendo mejor aproximado mientras mayor cantidad de puntos tenga la discretización. Es decir, que con una cantidad infinita de puntos ésta sería igual a la función misma.

Igual es necesario recalcar que, aún con su imprecisión, una cantidad menor de puntos da igualmente un resultado bastante bien aproximado, ya que, por ejemplo, en el primer gráfico, en ningún caso el valor de la isoterma para 5 círculos se aleja en más de 0.01, o sea, un error relativo del 1%. Para el 2do gráfico observamos un comportamiento similar, el error relativo se mantiene alrededor del 1%. Por lo tanto, se puede asumir que, si bien es inexacto, incluso una discretización bastante chica nos aproxima bien el resultado.

En el gráfico 2.1 podemos ver que introducir un caso patológico que alterne temperaturas exteriores entre 50 y 200 causa una curva que fluctúa constantemente con cambios bruscos, en contraste con la curva que no es patológica que se comporta de manera más estable. Pero es interesante analizar que, a pesar de tener la mitad de las temperaturas exteriores en $50^o$, la isoterma sigue quedando más cerca del exterior del horno que la curva no patológica en todos los casos. Esto se debe, podemos inferir, a que al calcular los valores del radio en que la temperatura externa vale $50^o$, a los costados la misma vale $200^o$, lo cual modifica mucho el valor y esto hace que los $400^o$ se encuentren más afuera que lo que sucede con una temperatura exterior más estable.

Por su parte, el gráfico 3.1 exhibe que el tiempo que insume todo el algoritmo es al menos cuadrático con respecto a la cantidad de puntos. Analizando el algoritmo, sabemos que resolver la eliminación gaussiana tiene un costo $O(n^3)$, siendo $n$ la longitud de la matriz (en este caso, es cantidad de círculos x cantidad de ángulos, o sea, la cantidad de puntos totales), lo que significa que nuestro algoritmo completo también es $O(n^3)$. Probablemente si pudiéramos seguir analizando el comportamiento para más y más puntos observaríamos ese comportamiento cúbico, pero el problema es que una matriz tan grande causa que el sistema se quede sin memoria, por lo que no nos es posible proseguir nuestras mediciones mucho más allá de estos datos que obtuvimos.

Se puede notar que esta curva es bastante irregular, esto lo podemos atribuir al hecho que estamos midiendo algo tan inexacto como el tiempo insumido por un proceso en un sistema que corre muchísimos procesos a la vez, por lo tanto, existen muchos factores que no podemos controlar como memoria disponible, quantums asignados a nuestro proceso, etc., que afectan al resultado final.

Todos estos análisis los hicimos teniendo en cuenta que la pared interna del horno es perfectamente circular. Cabe pensar qué haríamos si ésta no lo fuera, es decir, si fuera irregular. Habiendo discutido el problema, decidimos que lo más logico, suponiendo que nos pasen la función que describa la forma de la pared, sería hacernos una función que nos devuelva si un punto, dada su posición en el horno, si está adentro de la pared o no. Con esta función, al construir la matriz, preguntaríamos si el punto está adentro o no a la hora de hacer las ecuaciones, si nos da *true*, directamente le seteamos la ecuación $x_i = 1500$, si no, estimamos el punto como venimos haciendo. Es posible que hubiera que adaptar el método de resolver escalonado para que soporte este cambio (el de eliminación gaussiana no creemos que requiera cambios), pero no creemos que suponga una gran modificación a nuestro programa.

# 5 Conclusión

Este trabajo nos sirvió en primer lugar para entender cómo se puede modelar un problema físico de aplicación práctica con un sistema de ecuaciones lineales, a manera de ver una forma de aplicarlas en la vida real. El problema, que nos pedía calcular los valores de la isoterma 400º para un horno, presentaba la inherente dificultad de que no sabíamos la ecuación de la temperatura para cualquier punto dado del horno, sólo contábamos con las dimensiones del mismo y las temperaturas en las paredes interior y exterior, por lo que nos fue necesario inferir la función mediante una discretización de la misma en puntos de una matriz que eran estimables.

Una vez que la discretización estaba hecha, donde cada punto se estima como dependiendo de si mismo y de sus cuatro vecinos, se obtuvo una matriz esparsa que podía ser resuelta por cualquiera de los métodos vistos en clase.

Otro de los objetivos de este TP era resolver un sistema de ecuaciones lineales, y fue en este momento que hubo que llevarlo a la práctica. Para lograrlo, intentamos resolver la matriz con factorizacion $QR$, pero al resultar esto imposible, recurrimos al método de eliminación Gaussiana que es de más fácil implementación. Este método nos permitió triangular la matriz, que luego se pudo resolver con sustitución hacia atrás, para obtener los valores de temperatura en cada punto discretizado. Con estos valores, fue posible interpolar el valor del radio donde la temperatura es 400º para cada radio, al tomar el valor inmediatamente superior e inmediatamente inferior para cada uno.

Al observar los resultados obtenidos, fue evidente que una mayor cantidad de puntos en la discretización nos proporciona un resultado más exacto, aún cuando una discretización con pocos puntos nos proporcionaba un valor bastante aproximado. Con esto pudimos concluir que mientras más tienda a infinito la cantidad de puntos de la discretización, más y mas se va a acercar al valor de la función, una "discretización infinita" (que obviamente no seria una discretización) tendría el valor exacto de la función de la temperatura.

Otra de las cosas que podemos concluir es el orden del algoritmo, empíricamente obtuvimos que es al menos cuadrático, pero con pocos datos para analizar puesto que cuando crece mucho el tamaño de la discretización la máquina no puede calcularlo al tener memoria finita. Igualmente, viendo el algoritmo sabemos que éste tiene complejidad $O(n^3)$, por lo que es de esperar que si pudiéramos analizarlo en detalle, nuestro algoritmo mostraría ese comportamiento empíricamente.

# 6 Apéndices

## 6.1 Apéndice A

Consideremos la sección horizontal de un horno de acero cilíndrico, como en la Figura 1. El sector A es la pared del horno, y el sector B es el horno propiamente dicho, en el cual se funde el acero a temperaturas elevadas. El borde externo de la pared es un círculo, pero el borde interno de la pared -que coincide con el borde del sector B- no necesariamente tiene forma circular. Inicialmente lo consideraremos también circular.

Supondremos que la temperatura del acero dentro del horno es constante e igual a 1500 C. Hay sensores ubicados en la pared externa del horno para medir la temperatura, y estos habitualmente indican una temperatura entre 50 C y 200 C.

El problema que debemos resolver consiste en estimar la isoterma de 400 C dentro de la pared del horno, para estimar la resistencia de la pared al momento de su puesta en marcha. Si esta isoterma está muy cerca de la pared externa del horno, existe peligro de que la estructura externa de la pared colapse.

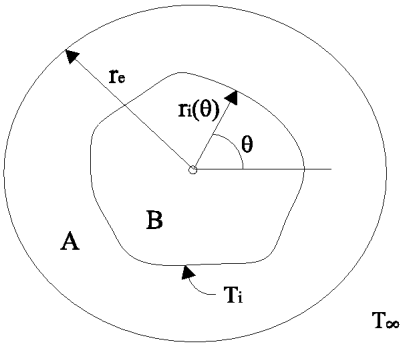


Figura 1: Sección del horno

El objetivo del trabajo práctico es implementar un programa que calcule la isoterma solicitada, conociendo las dimensiones del horno y las mediciones de temperatura en la pared exterior.

**El modelo**

Sea $r_e \in$ el radio exterior de la pared y sea $r_i : [0, 2\pi] \rightarrow$ el radio interior de la pared, que suponemos dependiente del ángulo (de manera tal que $r_i(\theta)$ es la distancia desde el centro hasta el borde del sector B en el ángulo $\theta \in [0, 2\pi]$). Llamemos $T(r, \theta)$ a la temperatura en el punto dado por las coordenadas polares $(r, \theta)$, siendo $r$ el radio y $\theta$ el ángulo polar de dicho punto. En el estado estacionario (luego de un tiempo suficientemente largo de operación del horno), esta temperatura satisface la ecuación del calor:

$$\frac{\partial^2 T(r,\theta)}{\partial r^2} + \frac{1}{r}\frac{\partial T(r,\theta)}{\partial r} + \frac{1}{r^2}\frac{\partial^2 T(r,\theta)}{\partial \theta^2} = 0 \tag{4}$$

Llamemos $T_i \in$ a la temperatura en el interior del horno (sector B) y $T_e \in$ a la temperatura en el extrior de la pared. Las condiciones de contorno del problema están dadas por:

$$T(r,\theta) = T_i, 0 < r \leq r_i \tag{5}$$

$$T(r_e,\theta) = T_e(\theta) \tag{6}$$

La condición (5) especifica que la temperatura en el borde interior del horno debe ser igual a la temperatura del acero en fundición, mientras que la condición (6) especifica que la temperatura debe ser igual a las

indicaciones de las termocuplas sobre la superficie exterior del horno.

El problema en derivadas parciales dado por la ecuación (4) con condiciones de contorno (5) y (6) permite encontrar la función $T$ de temperatura en la pared del horno (sector A), en función de los datos mencionados en esta sección.

**La resolución**

Para resolver este problema computacionalmente, discretizamos el dominio del problema (el sector A) en coordenadas polares. Consideremos una partición $0 = \theta_0 < \theta_1 < \cdots < \theta_n = 2\pi$ en $n$ ángulos discretos con $\theta_k - \theta_{k-1} = \Delta\theta$ para $k = 1, \ldots, n$, y una partición $0 = r_0 < r_1 < \cdots < r_m = r_e$ en $m + 1$ radios discretos con $r_j - r_{j-1} = \Delta r$ para $j = 1, \ldots, m$.

El problema ahora consiste en determinar el valor de la función $T$ en los puntos de la discretización $(r_j, \theta_k)$ que se encuentren dentro del sector A. Llamemos $t_{jk} = T(r_j, \theta_k)$ al valor (desconocido) de la función $T$ en el punto $(r_j, \theta_k)$.

Para encontrar estos valores, transformamos las ecuaciones (4) y (6) en un conjunto de ecuaciones lineales sobre las incógnitas $t_{jk}$, evaluando (4) en todos los puntos de la discretización que se encuentren dentro del sector A y evaluando (6) en todos los puntos de la discretización que se encuentren en el borde exterior de la pared. Al hacer esta evaluación, aproximamos las derivadas parciales de $T$ en (4) y (6) por medio de las siguientes fórmulas de diferencias finitas:

$$\begin{aligned}
\frac{\partial^2 T(r,\theta)}{\partial r^2}(r_j, \theta_k) &\cong \frac{t_{j-1,k} - 2t_{jk} + t_{j+1,k}}{(\Delta r)^2} \\
\frac{\partial T(r,\theta)}{\partial r}(r_j, \theta_k) &\cong \frac{t_{jk} - t_{j-1,k}}{\Delta r} \\
\frac{\partial^2 T(r,\theta)}{\partial \theta^2}(r_j, \theta_k) &\cong \frac{t_{j,k-1} - 2t_{jk} + t_{j,k+1}}{(\Delta \theta)^2}
\end{aligned}$$

Para cada ángulo $\theta_k$, llamamos $g_k$ al menor radio tal que el punto $(g_k, \theta_k)$ se encuentra dentro del sector A. Es importante notar que el valor de la incógnita $t_{g_k k}$ es conocido, dado que en la discretización el punto $(g_k, \theta_k)$ se encuentra sobre el borde interior de la pared (es decir, $t_{g_k k} = T_i$).

Al realizar este procedimiento, obtenemos un sistema de ecuaciones lineales que modela el problema discretizado. La resolución de este sistema permite obtener una aproximación de los valores de la función $T$ en los puntos de la discretización.

**Enunciado**

Se debe implementar un programa que tome como entrada los datos del problema y que calcule la temperatura en la pared del horno utilizando la técnica de resolución descripta en la sección anterior.

El programa debe tomar los datos de entrada desde un archivo de texto, cuyo formato queda a criterio del grupo. Es importante mencionar que los parámetros $n$ y $m$ de la discretización forman parte de los datos de entrada.

El programa debe generar el sistema de ecuaciones lineales planteado en la sección anterior, procediendo a su resolución por medio de cualquier método directo para la resolución de sistemas de ecuaciones lineales (es decir, un método no iterativo). El programa debe escribir la solución en un archivo, con un formato adecuado para su posterior graficación.

Sobre la base del resultado del sistema de ecuaciones, el programa debe obtener la isoterma de 400 C. El informe debe contener una descripción detallada del método computacional propuesto para obtener esta isoterma, junto con todas las decisiones que el grupo haya tomado con relación a este punto.

Se pide realizar experimentos con al menos dos instancias de prueba, generando distintas discretizaciones

para cada una. Se recomienda fijar $r_e = 1$ en estas instancias. Se sugiere que se presenten los resultados de estos experimentos en forma de gráficos de temperatura o gráficos de curvas de nivel, para ayudar a la visualización de los resultados. Como objetivo adicional obligatorio, se pide graficar el tiempo de resolución en función del tamaño de la discretización, para predecir los tiempos en discretizaciones más refinadas. Finalmente, qué espera que pase si el horno no presenta una geometría homogenea. Explicar en palabras cómo procedería a resolver el problema.

## 6.2 Apéndice B

### 6.2.1 Código fuente que genera y resuelve el sistema de ecuaciones

Incluimos a continuación el código (simplificado) relevante de las funciones usadas para generar y resolver el sistema de ecuaciones segun lo descripto en el desarrollo de este informe.

```
cin >> cant_radios >> cant_angulos >> radio_int >> radio_ext;

for(int i = 0; i<cant_angulos; i++){

  double temp;
  cin >> temp;
  temps_exterior.push_back(temp);
}

radio_dist = (radio_ext-radio_int) / (cant_radios-1);
angulo_dist = 2*M_PI / cant_angulos;

cant_ecus = cant_angulos * cant_radios;

matrix.resize(cant_ecus);
vecsol.resize(cant_ecus);

for (int i = 0; i < cant_ecus; i++){
  vecsol[i] = 0;
}


inicializar_matrix();

cout << "BEFORE GAUSS" << endl;
imprimir_matrix(matrix);

gauss(matrix);
cout << "AFTER GAUSS" << endl;

imprimir_matrix(matrix);

vd res= resolver_triangulado(matrix, vecsol);
```

```
for (unsigned int i =0; i< res.size(); i++ ){
  cout << "El valor x" << i << " es: " << res[i] << endl;
}

calcular_iso(res);
```

### 6.2.2 Código fuente de la eliminacion gaussiana

Incluimos a continuación el código relevante que realiza la eliminacion gaussiana

```
int tam = matrix.size();
for (int i = 0; i < tam-1; i++){
  for (int k = i+1; k < tam; k++){
    if (matrix[k][i] != 0){
      double factor = matrix[k][i]/matrix[i][i];
      for (int j= 0; j < tam; j++){
        matrix[k][j] -= matrix[i][j]*factor;
      }
      vecsol[k] -= vecsol[i]*factor;
    }
  }
}
```

### 6.2.3 Código fuente que resuelve la ecuación

A continuacion se presenta el código relevante que resuelve la ecuación

```
int tam = v.size();
vd res(tam,0);
res[tam-1] = v[tam-1];
for (int i = tam-2; i >= 0; i--){
  if ((i>=cant_angulos) && (i<cant_ecus-cant_angulos) && (v[i] != r[i][i])){
    double acum = 0;
    for (int j = tam-1; j > i; j--){
      acum += r[i][j]*res[j];
    }
    acum-=vecsol[i];
    res[i] = acum / (-r[i][i]);
  }else{
    res[i] = v[i];
  }
}
return res;
```

### 6.2.4 Código fuente que calcula la isoterma

Mas adelante incluimos el código relevante que calcula la isoterma

```
vd res;
vvd mat;
mat.resize(cant_radios);

for(int i =0; i<cant_radios; i++){
  mat[i].resize(cant_angulos);
}

for(int i =0; i<cant_radios; i++){
  for(int j =0; j<cant_angulos; j++){
    mat[i][j] = params[i*cant_angulos+j];
  }
}

for(int j =0; j<cant_angulos; j++){
  int i=0;
  while (mat[i][j]<400) i++;
  if (mat[i][j]==400){
    res.push_back(obtener_valor_radio(i));
  }else{
    double vmenor = mat[i-1][j];
    double vmayor = mat[i][j];
    double rmenor = obtener_valor_radio(i-1);
    double rmayor = obtener_valor_radio(i);

    double radio400 = rmenor + (400-vmenor)*(rmayor-rmenor)/(vmayor-vmenor);

    res.push_back(radio400);

  }
}

for (unsigned int i =0; i< res.size(); i++ ){
  cout << "La isoterma " << i << " es: " << res[i] << endl;
}
```

# 7 Referencias


- Linear interpolation
  `http://www.eng.fsu.edu/~dommelen/courses/eml3100/aids/intpol/index.html`
- Generador de números aleatorios
  `http://www.random.org/integers/?num=30&min=50&max=200&col=30&base=10&format=html&rnd=new`
- *Numerical Recipes: The Art of Scientific Computing, Third Edition (2007)*