Si bien la matemática no es una ciencia aplicada, es frecuentemente utilizada en la resolución de problemas de la industria o la ingeniería, entre otras disciplinas. Estas utilizan herramientas matemáticas para modelar problemas de la vida real, que frecuentemente involucran enormes cantidades de datos y variables, además de tener que obedecer comportamientos específicos según cada caso.
Para reproducir fielmente estos problemas la matemática emplea funciones, expresadas en sistemas de ecuaciones sobre las que se vuelcan todos los datos, y de las que se pretende obtener valores de incógnitas que permitan la proyección hacia el futuro de los comportamientos emulados.
Dadas las dimensiones de estos sistemas, es normal la utilización de herramientas informáticas en su manejo. Cuando los problemas son demasiado grandes o se manifiestan en conjuntos densos de valores, se lleva a cabo una discretización, que los vuelve resolubles para una computadora.
Los métodos que se emplean en dichas resoluciones pueden ser de dos tipos: directos o iterativos.
En este trabajo se utilizo un metodo directo, en particular el método de Gauss, uno de eliminación, y para obtener los valores se empleo la sustitución hacia atras.

Los métodos *directos* de resolución de sistemas lineales de ecuaciones son aquellos que permiten obtener la solución después de un número finito de operaciones aritméticas. Este número de operaciones es, obviamente, función del tamaño de la matriz.
Si las computadoras pudieran almacenar y operar con todas las cifras de los números reales, es decir, si emplearan una aritmética exacta, con los métodos directos se obtendría la solución exacta del sistema en un número finito de pasos.
Debido a que las computadoras trabajan con precisión finita, los errores de redondeo se propagan y la solución numérica obtenida siempre difiere de la solución exacta.
La cota del error, para una matriz y término independiente dados, se asocia por lo general al número de operaciones de cada método. Se pretende, por lo tanto, obtener métodos con el mínimo número de operaciones posible.
Una particularidad de los métodos directos es que siempre conducen, después de ciertas operaciones, a la resolución de uno o varios sistemas con solución inmediata. Es decir, sistemas donde la matriz es diagonal o triangular.

En el método de eliminación de Gauss el problema original, $\boldsymbol{Ax} = \boldsymbol{b}$**,** se transforma mediante permutaciones adecuadas y combinaciones lineales de cada una de las ecuaciones en un sistema de la forma $\boldsymbol{Ux} = \boldsymbol{c}$ donde $\boldsymbol{U}$ es una matriz triangular superior. Los elementos en la diagonal son llamados pivotes y se utilizan para eliminar a los elementos que se encuentran en su misma columna pero en filas mayores (es decir, los elementos bajo el pivote). Para esto se realiza el siguiente algoritmo:

ACA VA LA CUENTITA DE GAUSS

Como se puede apreciar cuando el pivote es cero el algoritmo no puede llevarse a cabo, con lo que es necesaria una permutación entre las filas por anular para obtener un pivote distinto de cero.
Esta permutación de filas no sólo tiene interés cuando el pivote es exactamente cero. Es obvio que valores pequeños del pivote pueden producir grandes errores de redondeo, ya que siempre se divide por el valor del pivote.
Por consiguiente, para reducir los errores de redondeo conviene escoger el pivote máximo en valor absoluto. Para ello, hay dos técnicas posibles:

Se toma como pivote el coeficiente mayor en valor absoluto de la columna k situado por debajo de la fila k inclusive. Para ello es necesario permutar las filas k y la correspondiente al pivote escogido en la matriz y su término independiente. Esta técnica se denomina método *de Gauss* con pivoteo parcial. Existen otras metodologías de pivoteo pero esta fue la empleada durante el presente trabajo.

La variante es extender la busqueda del coeficiente de mayor módulo no solo a la columna k, sino a toda la matriz no triangulada, es por ello que esta tecnica se conoce como pivoteo total.

Tras aplicar el algoritmo de Gauss a todas las filas del sistema nos queda una matriz triangular superior. Este nuevo sistema equivalente al original es de resolución inmediata, sólo es necesario aplicar el algoritmo de sustitución hacia atrás.

En cuanto a los métodos iterativos para resolver ecuaciones son aquellos en los cuales una primera aproximación es usada para calcular una segunda aproximación, la cual a su vez es usada para calcular una tercera aproximación, y así sucesivamente.

Desde un punto de vista general las matrices más usuales en las ciencias aplicadas y en ingeniería pueden englobarse en dos grandes categorías:

- Matrices llenas, pero no muy grandes. Por llenas se entiende que poseen pocos elementos nulos y por no muy grandes que el número de ecuaciones es de unos pocos miles a lo sumo. Estas matrices aparecen en problemas estadísticos, matemáticos, físicos e ingenieriles.

- Matrices vacías (ralas) y muy grandes. En oposición al caso anterior, vacías indica que hay pocos elementos no nulos y además están situados con una cierta regularidad. En la mayoría de estos casos el número de ecuaciones supera los miles y puede llegar en ocasiones a los millones. Estas matrices son comunes en la resolución de ecuaciones diferenciales de problemas de ingeniería.

Parece lógico que los métodos para resolver sistemas lineales de ecuaciones se adecuen a las categorías de matrices anteriormente expuestas. En general los *métodos directos* se aplican al primer tipo de matrices, mientras que los *métodos iterativos* se emplean con el segundo grupo. Es importante observar que no existen reglas absolutas y que todavía en la actualidad existe cierta controversia sobre los métodos óptimos a aplicar en cada caso.

Desarrollo

El problema particular a resolver en este trabajo es el comportamiento de un horno de acero, en lo que corresponde a la temperatura de su parte exterior. Este horno puede dividirse en tres “capas”, la primera, el horno propiamente dicho, esta a una temperatura constante y esta es un dato conocido; la segunda es la parte intermedia, la pared interior, su forma es arbitraria y viene dada por una función r, también conocida; la última capa es la pared exterior, y de ésta se desea conocer la temperatura. Por ser el horno circular (con radio conocido), para su representación se utilizaron coordenadas polares.
Se sabe que la temperatura se corresponde con una función T, para cualquier angulo y radio. La función es desconocida, con lo que para conocer la temperatura de la pared exterior es necesario hallarla. De T se sabe que cumple con tres ecuaciones:

LAPLACIANO IGUAL A CERO

TEMPERATURA DEL INTERIOR

ECUACION DE FLUJO

Siendo el dominio de T las coordenadas polares y R su imagen, se procede a realizar una discretización de la función, para poder aproximar las derivadas de las ecuaciones 1 y 3 por el método de las diferencias finitas. Como resultados, se obtuvieron las siguientes aproximaciones de

las derivadas de T:

INSERTAR APROXIMACIONES DE LAS DERIVADAS

Con estas aproximaciones y las ecuaciones antes detalladas se realizó el cálculo de la función T en cada punto de la discretización de la siguiente manera:

1)con las ecuaciones 1 y 3 se despejaron coeficientes para cada una de las variables. De esta forma cada punto de la discretización puede ser categorizado según las tres partes del horno mencionadas más arriba. Así, contamos con puntos del horno propiamente dicho, cuya temperatura es conocida (la temperatura interior del horno), puntos de la pared interior y los puntos del perímetro del horno, o de la pared exterior del mismo.
Los puntos de la pared interior se presentan como los mas complicados de manejar ya que luego de algunas manipulaciones algebraicas se encuentran expresados en función de otros cuatro puntos aledaños

ECUACIONES DESPEJADAS PARA QUE SE NOTE LO DE ARRIBA
En cuanto a los puntos dentro del horno, ya se encuentran "resueltos", expresados solo en función de la temperatura interior. Los puntos de la pared exterior quedan determinados por la ecuación de flujo, en función de un punto anterior, la temperatura exterior y las constantes térmicas H y K
ECUACION CORRESPONDIENTE

2)tras esta caracterización de cada punto interviniente se construye una matriz donde se vuelca toda esta información, con una ecuación para cada punto. La matriz, que será el eje de la resolución del problema, tiene dimensión *cantidadDePuntosDiscretizados*cantidadDePuntosDiscretizados*, y recopila todas las ecuaciones despejadas en el punto anterior.
3)Se puede apreciar gracias a las ecuaciones anteriores que la forma de la matriz resultante es predecible y se corresponde con la estructura del horno, ya que tiene una primera sección que asemeja la matriz identidad, con tan solo unos en la diagonal y ceros en el resto de las filas. Se nota un segundo gurpo de filas, que atañen a las variables de la pared interior del horno, estas filas tienen cinco coeficientes distintos de cero en un patrón definido, que asemeja una matriz banda un tanto peculiar en cuanto a su distribución. Esta alterna unas diagonales con ceros, una diagonal de coeficientes, otras diagonales de ceros, tres diagonales de coeficientes, unas diagonales más de ceros, una última diagonal de coeficientes y unas diagonales de ceros que terminan la matriz. Tras esta sección que podría catalogarse como pseudo-banda, se hallan las ecuaciones que describen los puntos de la pared exterior del horno, esta ultima serie de filas tiene tan solo dos coeficientes distintos de cero (y además son consecutivos), el correspondiente al punto descrito y su inmediato anterior en la sucesión de radios.
Nótese que el "b" de nuestro sistema se representa en otra matriz, organizada de igual forma que la principal, con lo que su primera parte es una columna que lleva el valor de la temperatura interior del horno, en la segunda solo hay ceros y en la tercera y última el valor de la ecuación de flujo. Con este sistema nos encontramos en condiciones de calcular todos los valores de T dentro de la discretización.

3)La resolución del conjunto de ecuaciones anterior se lleva a cabo en dos pasos, triangulación , a través de eliminación gaussiana con pivoteo parcial, y sustitución hacia atrás para despejar la solución pertinente a cada ecuación. Se volverá sobre este punto más adelante.

Con esto se obtiene un vector columna que responde al x (en una tercera matriz) de nuestro sistema $Ax = b$, que se utiliza para realizar los gráficos de distribución de la temperatura pertinentes.

Detalles de la implementación

Para este trabajo se implementaron dos clases (de C++):

la clase Matriz, que modela un sistema de ecuaciones en forma de matriz de long doubles, con los métodos necesarios para crear matrices (que se inicializan con sus coeficientes en cero), asignarles valores a cada posición, ver dichos valores, triangular (incluye un metodo privado para el pivoteo parcial) una matriz ya creada, asi como resolverla mediante la sustitución hacia atrás.

En su estructura esta clase esta sustentada por un arreglo de arreglos de long double, creados en forma dinámica durante la ejecución del programa. En cuanto al lenguaje C++, un arreglo no es más que un puntero a un espacio de memoria contiguo, con lo que esta implementación, bien puede pasar por un puntero a un puntero a long double. De hecho, esta es la forma en que C++ implementa su propio tipo nativo “matriz”, como un arreglo que tiene dos indices, uno de los cuales se utiliza con un multiplicador (para moverse por las filas) y otro como “off set” (para ubicar elementos en las columnas). Se decidió implementar una nueva clase para lograr una mayor abstracción y modularidad en el código, además de un ocultamiento de información delimitado por la parte pública y la privada.
Además de este puntero a puntero la clase cuenta con otros atributos, fil y col que denotan el número de filas y columnas respectivamente. Recordar que en C++ no se puede conocer en forma directa la dimensión de un arreglo.

Como sugerencia para una próxima revision de este trabajo resta explorar la posibilidad de utilizar la librería valarray, parte de la librería STL (librería estándar de plantillas), optimizada para computación numérica y manejo de matrices en su aspecto matemático.

Nota: a partir de este punto siempre que se emplee el término 'matriz', éste se refiere a la clase implementada y no al tipo nativo de C++.

En cuanto a los métodos de la clase podemos enumerar:

QUIZAS EN LUGAR DEL NOMBRE SE PODRIA PONER LA ARIDAD DE CADA METODO

constructor: crea en tiempo de ejecución la estructura mencionada, inicializando todos los valores de long double en cero. La dimensión de la matriz viene dada por los dos parámetros de la función, si éstos no son completados se crea por defecto una de 4*4.
Se cuenta también con un constructor por copia y un **operator**=, el primero utiliza fuertemente al segundo. El operador, copia los atributos fil y col de la matriz de la derecha en los de la izquierda, elimina la matriz “pisada” por la igualación y luego crea una nueva, en la que copia los valores de la original.

filas() y columnas(): se implementaron sendos métodos para conocer tanto el número de filas como de columnas de la matriz creada. Tan solo devuelven una copia de los atributos privados de la clase.

Ver: este método requiere dos parámetros que se utilizan, el primero para indexar una fila, y el segundo para hacer lo propio sobre una columna y se devuelve el elemento correspondiente de la matriz. Lleva una clausula **const** de forma que no se pueda modificar la matriz con este método.

Asignar: éste también requiere dos parámetros como Ver, y un tercero, un long double que será el nuevo valor de la posición seleccionada.

Tanto este método como Ver, lanzan una excepción si se le pasan parámetros por fuera del rango de

filas o de columnas de la matriz.

Triangular: recibe una matriz 'b' por referencia, que es el vector columna de la solución. Utilza los métodos privados pivotear y restarFilas, para emular el método de resolución de sistemas de ecuaciones lineales conocido como eliminación gaussiana. De esta manera transforma nuestro sistema en uno triangular superior, susceptible de ser despejado por “sustitución hacia atrás”.
Se comentan primero los métodos pivotear y restarFilas para luego volver sobre éste.

Pivotear: recibe un entero que denota la fila en la que se encuentra el algoritmo de Gauss, y busca mediante pivoteo parcial (ya explicado en la introducción) el mejor pivote disponible.
Se implemento este algoritmo en lugar del pivoteo total por ser más sencillo, además resta saber si la ganancia en precisión supera la perdida en tiempo que representa la busqueda a lo largo de toda la sub-matriz, en lugar de tan solo la columna pertinente.

RestarFilas: recibe un long double que es el coeficiente del pivote, y dos enteros, la fila a la que hay que anularle el valor de la columna del pivote, y la fila donde se encuentra el pivote. Este procedimiento se corresponde con el algoritmo enunciado en la introducción.

Este método se limita aplicar restarFilas con los elementos de la diagonal, hasta que algún pivote tiene valor cero, momento en el que se emplea pivotear.
Se eligió la eliminación gaussiana para obtener un sistema de resolución directa porque es el más conocido y estudiado, además, no tiene un costo temporal exagerado (es de orden cúbico), ni requiere memoria adicional a la necesaria para almacenar la matriz, tiene una implementación sencilla y se adapta perfectamente a cualquier matriz cuadrada.

Queda un aspecto muy importante a mejorar en futuras versiones de este programa: ni el algoritmo que implementa la eliminación gaussiana, ni la clase matriz, aprovechan la particular estructura antes mencionada que rige el sistema resultante del volcado de los datos y su tratamiento algebraico. Este conocimiento del patron que siguen las variables en la matriz podría ser utlizado para idear una estrategia de almacenamiento y mapeo de indices que permita ahorrar memoria no conservando los ceros. Se podría utilizar también para mejorar el rendimiento del algoritmo de triangulación, conservando la eliminación gaussiana.
Tampoco se pudo implementar un algoritmo que reduzca el espacio en memoria que ocupa la matriz triangulada, que podría ser acortado hasta un poco más de la mitad del consumo original.
Todas estas optimizaciones se discutieron durante la etapa de implementación, y se descartaron, en pos de cumplir con la fecha limite asignada durante la presentación del proyecto

La segunda clase que interviene en este trabajo es la que utiliza la matriz y sus métodos, y que modela el horno a través de una matriz de ecuaciones que describen los valores discretizados de la ya mencionada función T.
Entre sus atributos esta clase cuenta con todos los datos necesario para poder resolver el problema planteado, a saber:

temperaturas: una matriz con los valores y las características comentados durante este desarrollo
rad: un long double que representa el radio exterior del horno completo, desde el centro hasta la pared exterior.
angs y rads: don enteros, la cantidad de ángulos y radios en los que se discretiza el horno. Recordar que se utilizan coordenadas polares.
deltaR y deltaT: long doubles, la diferencia entre los diferentes radios y angulos respectivamente. deltaT se expresa en radianes.
ti, tinf: ambos enteros. Ti es la temperatura dentro del horno propiamente dicho, “donde se funde el

acero”, mientras que tinf es la temperatura del espacio no delimitado que esta afuera del horno, mas allá de la pared exterior. Tanto ti como tinf están expresadas en grados celcius (ºC)
k y h, dos long doubles, constantes de los materiales, k es la conductividad térmica de la pared del horno y h el coeficiente de transferencia de calor del borde de la pared.

```
Matriz* temperaturas;       //temperatura en cada punto discretizado del horno
        long double rad;            //radio del horno desde el centro al borde exterior
        int angs;                   //angulos en los que se divide el horno
        int rads;                   //cantidad de radios por angulo
        long double deltaR;         //delta Radio
        long double deltaT;         //delta Theeta
        int ti;                     //temperatura interior
        int tinf;                   //temperatura en el infinito
        long double k;              //constante k
        long double h;              //constante h
        int *bordeInterno;          //borde interno del horno.
```