Álgebra y matemáticas discretas

José Armando Cuesta Buitrago

Kevin Alejandro De Arco Cabezas

Universidad internacional de la Rioja (UNIR)

Ing. Informática

Tabla de contenido

Contenido

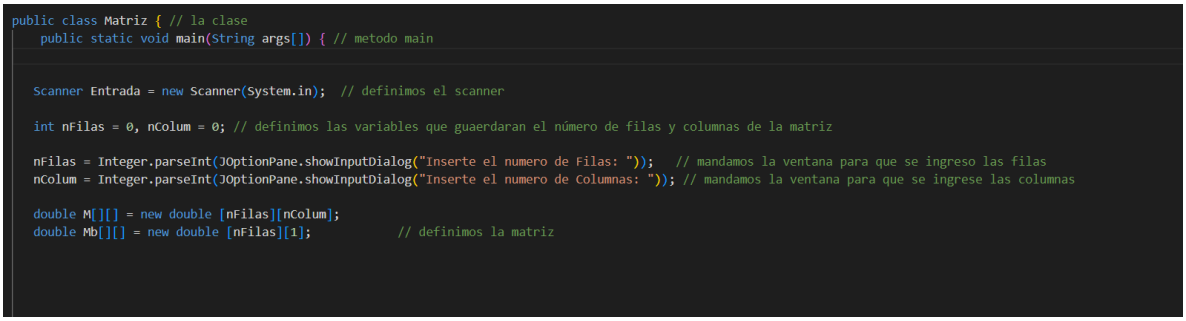
Objetivos

1) Crear un código en el lenguaje de programación de Java, el cual debe crear una matriz y resolver un sistema de ecuaciones a partir de la matriz dada.
2) Poder utilizar la eliminación gaussiana por el método del pivoteo parcial en la matriz creada para poder resolver el sistema de ecuaciones.
3) Mostrar al usuario la matriz y el resultado de esta después de usar el método de eliminación gaussiana.

Creación de la matriz

Para crear dicha matriz usaremos el lenguaje de Java, en un IDE con Java cargado, importamos algunos paquetes que usaremos para hacer algo más amigable la creación de esta matriz. Primero importamos “java.util.Scanner” para poder tomar datos por pantalla digitados por el usuario, ahora importamos “javax.swing.JOptionPane” con el propósito de hacer algo más amigable la creación de la matriz, pues esta importación nos servirá para hacer aparecer cuadros por pantalla y así tomar el dato del tamaño de la matriz, a pesar de que pudimos utilizar solo la función scanner para tomar esos datos, preferimos que el usuario pueda observar mejor y así se entere que a partir de ese momento la matriz va a empezar a crearse. Ahora empezamos a escribir código, empezamos por la clase a la cual se le llamó “Matriz” y también colocamos el método “main” para poder ejecutar todo el código que vamos a digitar. Empezamos digitando las variables y definiéndose, así como el scanner y la matriz, también “i” y “j” que nos servirán de ayuda para ejecutar los ciclos “for” y poder asignar los números dentro de una posición de la matriz, también vamos a tener las variables “nFilas” y “nColum” las cuales se usarán con el “JOption” para poder obtener el tamaño de la matriz, la matriz será definida como “double” pues así poder usar la matriz con una gran variedad de datos, “nFilas” y “nColum” se asignó como “int” pues son las variables que usaremos para guardar el tamaño de la matriz que nos proporciona el usuario. Usaremos la función del “JOption” para mandar la pantalla con un mensaje y la posibilidad de escribir valores, “nFilas = Integer.parseInt(JOptionPane.showInputDialog("Inserte el número de Filas: "));” es lo que

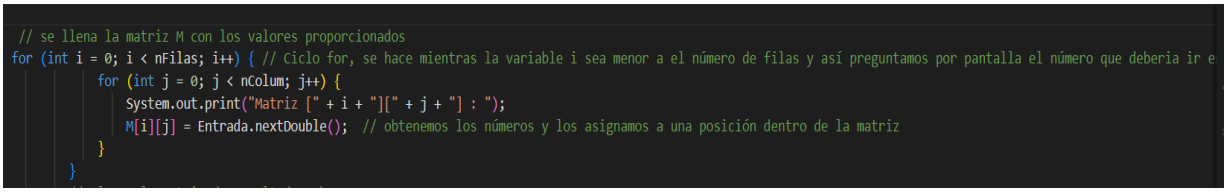
utilizaremos para mostrar la pantalla y guardar lo digitado en nuestra variable, utilizaremos este método para obtener los dos datos (filas y columnas). Crearemos un condicional “if” para asegurarnos que la matriz sea cuadrada, para ello colocamos el “if” y su condicional que será (nFilas == nColum) pues es ello una matriz cuadrada, es decir que la cantidad de filas y columnas sean iguales, en caso de ser falsa se mostrará una ventana emergente utilizando el “JOption” con el mensaje “La matriz debe ser cuadrada para usar eliminación gaussiana”. Una vez superada la condicional entonces empezaremos a crear la matriz dándole el tamaño ya obtenido guardado en “nFilas” y “nColum” teniendo el tamaño de la matriz creamos nuestro ciclo “for” para recorrer la matriz y empezar a digitar datos, el ciclo “for” funciona mientras que la variable “i” iniciada en cero que recorrerá las filas y se le da un aumento de uno después de recorrer la fila, esto terminará cuando “i” sea mayor al número de filas, lo mismo ocurre con “j” que es utilizado para las columnas, en el proceso de ese recorrido iremos mostrando por pantalla “Matriz” y la posición en la que nos encontramos, iniciando en cero(0) cero(0) para poder ir llenando las posiciones usamos la función scanner y lo asignamos a la matriz “M[i][j]”



```
public class Matriz { // la clase
    public static void main(String args[]) { // metodo main


   Scanner Entrada = new Scanner(System.in);  // definimos el scanner

   int nFilas = 0, nColum = 0; // definimos las variables que guaerdaran el número de filas y columnas de la matriz

   nFilas = Integer.parseInt(JOptionPane.showInputDialog("Inserte el numero de Filas: "));   // mandamos la ventana para que se ingreso las filas
   nColum = Integer.parseInt(JOptionPane.showInputDialog("Inserte el numero de Columnas: ")); // mandamos la ventana para que se ingrese las columnas

   double M[][] = new double [nFilas][nColum];
   double Mb[][] = new double [nFilas][1];           // definimos la matriz
```

Ahora mostraremos por pantalla la matriz creada con los datos y el tamaño antes digitados, primero mostraremos que vamos a imprimir la matriz y luego con ayuda de los

ciclos “for” empezaremos a mostrar toda la matriz y la imprimimos con la función “printf” para poder hacer que la matriz tenga un formato y se muestre de una manera más organizada como si fuera una lista para ello también usamos un salto de línea justo después de terminar de escribir el último dígito de la fila.



```
// se llena la matriz M con los valores proporcionados
for (int i = 0; i < nFilas; i++) { // Ciclo for, se hace mientras la variable i sea menor a el número de filas y así preguntamos por pantalla el número que deberia ir e
        for (int j = 0; j < nColum; j++) {
            System.out.print("Matriz [" + i + "][" + j + "] : ");
            M[i][j] = Entrada.nextDouble();  // obtenemos los números y los asignamos a una posición dentro de la matriz
        }
    }
```

Luego de crear nuestra matriz principal también vamos a crear la matriz b o la matriz de los términos independientes para hacer la matriz ampliada, para ello también usaremos los ciclos “for” que utilizaremos para escribir los números y también tendremos que crear las variables de matriz (Mb[][]) e inicializarla su tamaño el cual ya sabremos su cantidad de filas que será igual a las filas de la matriz principal, y sus columnas solo serán una pues solo son los resultados de nuestro sistema de ecuaciones, después de darle los valores a esa matriz procedemos a mostrar por pantalla la matriz ampliada con un mensaje avisando que mostraremos dicha matriz, la forma en que mostramos la matriz ampliada es la misma que la de la anterior, con ayuda de los ciclos “for”, pero esta tiene un “for” extra pues será la que nos ayudará a imprimir la matriz b “Mb” dentro del ciclo también hay un salto de línea e imprimimos con formato para que la matriz quede en formato de tabla.

```java
// Mostrar la matriz M
 System.out.println("\n La matriz es : ");
 for (int i = 0; i < nFilas; i++) {
     for (int j = 0; j < nColum; j++) {
         System.out.printf("%8.1f", M[i][j]);
     }
     System.out.println();
 }

 // Mostrar la matriz total (M junto con Mb)
 System.out.println("\n La matriz Total es : ");
 for (int i = 0; i < nFilas; i++) {
     for (int j = 0; j < nColum; j++) {
         System.out.printf("%8.1f", M[i][j]);
     }
     System.out.printf(" |%5.1f\n", Mb[i][0]);
 }
```

Cuando ya se tienen los valores agregados a la matriz, procedemos aplicar el método de eliminación gaussiana por el metodo pivoteo parcial, para esto primero debemos iterar sobre las columnas y filas de la matriz. El ciclo “For” se ejecutará hasta que se complete el proceso de eliminación gaussiana para todas las columnas o filas, lo que ocurra primero. Por lo tanto, el bucle se detendrá cuando k alcance el mínimo entre el número de filas (nFilas) y el número de columnas (nColum). Esto asegura que no intentemos acceder a índices fuera del rango de la matriz. como segundo paso debemos encontrar el numero con valor absoluto mayor o pivote máximo en la columna actual (k). Esto se hace recorriendo la columna desde la fila “k + 1” hasta la última fila (nFilas). Se actualiza el valor máximo (maxPivot) y se registra el índice de la fila correspondiente (maxIndex) donde se encuentra el pivote máximo, si el índice del pivote máximo (maxIndex) es diferente del índice actual (k), significa que se encontró un pivote más grande en otra fila. En este caso, se intercambian las filas k y maxIndex tanto en la matriz “M[][]” como en la matriz de

términos independientes “Mb[][]”. Esto se hace para mover el pivote máximo a la posición “k”, lo que ayuda a evitar divisiones por cero. Terminado el paso anterior, cuando se encuentra el pivote máximo y se realizan los intercambios necesarios, se procede a realizar la eliminación gaussiana en la columna actual (k). Se recorren las filas desde “k + 1” hasta la última fila (nFilas) y se realiza la eliminación gaussiana para reducir los elementos debajo del pivote a cero. Esto implica calcular un factor de eliminación y restar múltiplos adecuados de la fila actual (k) a las filas siguientes. También se actualizan los elementos correspondientes en la matriz de términos independientes “Mb[][]”, esta es la parte más importante del proceso dado que se realiza el proceso de eliminación gaussiana

```
// Vamos a hacer la eliminación gaussiana por el metodo pivotaje parcial

for (int k = 0; k < Math.min(nFilas, nColum); k++) {
 // Encontrar el pivote máximo en la columna k
 double maxPivot = Math.abs(M[k][k]);
 int maxIndex = k;
 for (int i = k + 1; i < nFilas; i++) {  //se corre un ciclo for que recorra la matriz M y encuentre el maximo numero de la columna k
     if (Math.abs(M[i][k]) > maxPivot) {
         maxPivot = Math.abs(M[i][k]);
         maxIndex = i;
     }
 }
 // Intercambiar filas si es necesario
 if (maxIndex != k) {  //si el numero que esta de primero en la columna es menor se va a intercambiar toda la columna
     double[] temp = M[k];
     M[k] = M[maxIndex];
     M[maxIndex] = temp;
     double tempMb = Mb[k][0];
     Mb[k][0] = Mb[maxIndex][0];
     Mb[maxIndex][0] = tempMb;
 }
 // Eliminación Gaussiana
 for (int i = k + 1; i < nFilas; i++) {
     double factor = M[i][k] / M[k][k];
     for (int j = k + 1; j < nColum; j++) {
         M[i][j] -= factor * M[k][j];
     }
     Mb[i][0] -= factor * Mb[k][0];
 }

}
```

Una vez se completa el proceso anterior se empieza a iterar o recorrer sobre la matriz “M[][]” y la matriz de resultados “Mb[][]” después de haber aplicado la eliminación gaussiana. Imprime los elementos de “M[][]” con un ancho de campo de 8 y un decimal utilizando “printf()”. Después de imprimir cada fila de “M[][]”, imprime el elemento correspondiente de “Mb[][]” con un ancho de campo de 5 y un decimal.

```
    // Mostrar la matriz M después de la eliminación gaussiana
System.out.println("\nLa matriz despues de la eliminacion gaussiana:");
for (int i = 0; i < nFilas; i++) {
  for (int j = 0; j < nColum; j++) {
    System.out.printf("%8.1f", M[i][j]);
  }
  System.out.printf(" |%5.1f\n", Mb[i][0]);
}
```

Cuando se finaliza el paso anterior se debe resolver el sistema triangular, para esto se debe iterar sobre las filas de la matriz “M[][]” en orden inverso. Para cada fila i, calcula la suma de los productos de los elementos de la fila i de “M[][]” y las soluciones conocidas soluciones”[j]” para las incógnitas que ya se han resuelto. Luego, calcula la solución para la incógnita correspondiente a la fila i y la guarda en el arreglo soluciones[].

```
// Resolución del sistema triangular
double[] soluciones = new double[Math.min(nFilas, nColum)];
for (int i = Math.min(nFilas, nColum) - 1; i >= 0; i--) {
  double sum = 0.0;
  for (int j = i + 1; j < Math.min(nFilas, nColum); j++) {
    sum += M[i][j] * soluciones[j];
  }
  soluciones[i] = (Mb[i][0] - sum) / M[i][i];
}
```

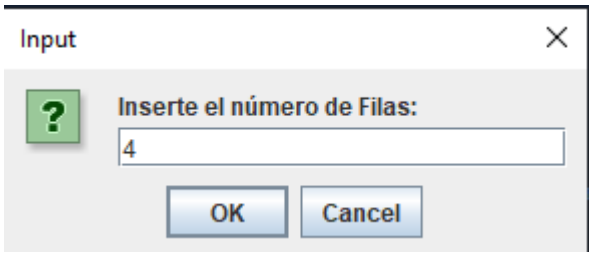
Como penúltimo paso se debe mostrar la matriz con los valores en 0 por debajo de la diagonal para esto se debe recorrer la matriz “M[][]” y se establecen todos los elementos por debajo de la diagonal principal a cero. Esto se logra mediante la condición “if (j < i)”. Luego, se imprime la matriz resultante con un formato específico utilizando “printf()” para garantizar que esté bien alineada y sea fácil de leer.

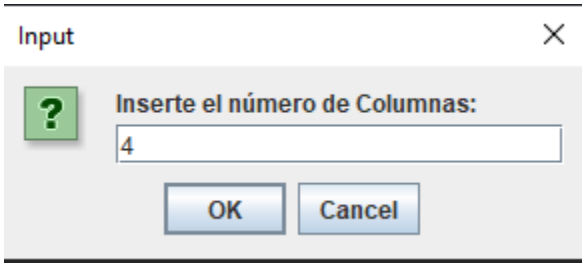
```
// Resultado de la matriz despues de la eliminacion gaussiana
System.out.println("\nResultado Matriz por debajo de la diagonal:");
for (int i = 0; i < nFilas; i++) {
  for (int j = 0; j < nColum; j++) {
    if (j < i) {
      M[i][j] = 0; //Se establecen los valores por debajo de la diagonal a 0
    }
    System.out.printf("%8.1f", M[i][j]);
  }
  System.out.println();
}
```

Para finalizar se deben mostrar los valores de las incógnitas (x1, x2, x3, x4); para esto se debe iterar sobre las soluciones encontradas para las incógnitas. Si una solución es un número entero (dentro de un margen de error de 1e-10), se redondea a ese número entero utilizando “Math.round()”. De lo contrario, se imprime como un número de coma flotante con dos decimales utilizando “String.format()”. Esto garantiza que las soluciones se muestren de manera correcta y precisa. Cada solución se imprime con un mensaje que indica a qué incógnita corresponde.

```
// Resultados despues de la eliminacion
System.out.println("\nLos valores de las incognitas son:");
for (int i = 0; i < Math.min(nFilas, nColum); i++) {
  if (Math.abs(soluciones[i] - Math.round(soluciones[i])) < 1e-10) {
    System.out.println("x" + (i + 1) + ": " + Math.round(soluciones[i]));
  } else {
    System.out.println(
      "x" + (i + 1) + ": " + String.format("%.2f", soluciones[i])
    );
  }
}
```

Ejecución del código






```
Matriz [0][0] : 6
Matriz [0][1] : -2
Matriz [0][2] : 2
Matriz [0][3] : 4
Matriz [1][0] : 12
Matriz [1][1] : -8
Matriz [1][2] : 6
Matriz [1][3] : 10
Matriz [2][0] : 3
Matriz [2][1] : -13
Matriz [2][2] : 9
Matriz [2][3] : 3
Matriz [3][0] : -6
Matriz [3][1] : 4
Matriz [3][2] : 1
Matriz [3][3] : -18
```

```
Matriz de resultado [0][0] : 16
Matriz de resultado [1][0] : 26
Matriz de resultado [2][0] : -19
Matriz de resultado [3][0] : -34
```

```
La matriz es :
    6,0    -2,0     2,0     4,0
   12,0    -8,0     6,0    10,0
    3,0   -13,0     9,0     3,0
   -6,0     4,0     1,0   -18,0

La matriz Total es :
    6,0    -2,0     2,0     4,0 | 16,0
   12,0    -8,0     6,0    10,0 | 26,0
    3,0   -13,0     9,0     3,0 |-19,0
   -6,0     4,0     1,0   -18,0 |-34,0
```

```
 La matriz despues de la eliminacion gaussiana:
    12,0    -8,0     6,0    10,0 | 26,0
     3,0   -11,0     7,5     0,5 |-25,5
    -6,0     0,0     4,0   -13,0 |-21,0
     6,0     2,0     0,4     0,3 |  0,3

 Resultado Matriz por debajo de la diagonal:
    12,0    -8,0     6,0    10,0
     0,0   -11,0     7,5     0,5
     0,0     0,0     4,0   -13,0
     0,0     0,0     0,0     0,3

 Los valores de las incognitas son:
 x1: 3
 x2: 1
 x3: -2
 x4: 1
```