

1.00Cla se22

Sistemas de ecuaciones lineales

Sistemas de ecuaciones lineales

$$3x_0 + x_1 - 2x_2 = 5$$

$$2x_0 + 4x_1 + 3x_2 = 35$$

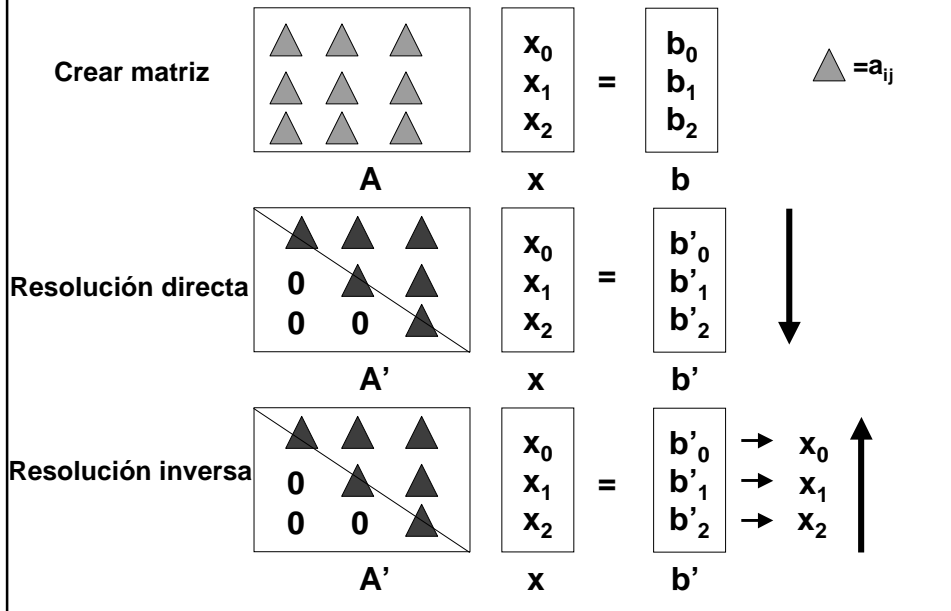
$$x_0 - 3x_1 = -5$$

$$\begin{array}{ccc|ccc} 3 & 1 & -2 & x_0 & & 5 \\ 2 & 4 & 3 & x_1 & = & 35 \\ 1 & -3 & 0 & x_2 & & -5 \end{array}$$

$$A \quad x = b$$

$$3 \times 3 \quad 3 \times 1 \quad 3 \times 1$$

Algoritmo para resolver sistemas lineales



Eliminación gaussiana: resolución directa

Q=	3	1	-2	5	Forma Q por comodidad Opciones básicas de filas: Multiplicar filas Sumar/restar filas
	2	4	3	35	
	1	-3	0	-5	
	A			b	

Hace que la columna 0 tenga ceros por debajo de la diagonal

Pivot= 2/3 →	3	1	-2	5	Fila1' = fila1 - (2/3)fila0 Fila2' = fila2 - (1/3)fila0
Pivot= 1/3 →	0	10/3	13/3	95/3	
	0	-10/3	2/3	-20/3	

Hace que la columna 1 tenga ceros por debajo de la diagonal

Pivot= 1 →	3	1	-2	5	Fila2'' = fila2' + 1*fila1
	0	10/3	13/3	95/3	
	0	0	15/3	75/3	

Eliminación gaussiana: resolución inversa

3	1	-2	5
0	10/3	13/3	95/3
0	0	15/3	75/3

$$(15/3)x_2 = (75/3)$$

$$x_2 = 5$$

3	1	-2	5
0	10/3	13/3	95/3
0	0	15/3	75/3

$$(10/3)x_1 + (13/3)*5 = (95/3) \quad x_1 = 3$$

3	1	-2	5
0	10/3	13/3	95/3
0	0	15/3	75/3

$$3x_0 + 1*3 - 2*5 = 5$$

$$x_0 = 4$$

Una complicación

0	1	-2	5
2	4	3	35
1	-3	0	-5

$$\text{Fila1}' = \text{fila1} - (2/0)\text{fila0}$$

Intercambia las filas: coloca el mayor elemento pivotante en la fila:

2	4	3	35
1	-3	0	-5
0	1	-2	5

Lo hace mientras procesamos cada columna.

Si no hay ningún elemento distinto de cero en una columna, la matriz no es de orden completo.

Eliminación gaussiana

```
public static void gaussian(Matrix a, Matrix b, Matrix x) {
    int i, j, n;
    n= a.getNumRows();           // Número de incógnitas
    Matrix q= new Matrix(n, n+1);
    for (i=0; i < n; i++) {
        for (j=0; j < n; j++)    // Forma la matriz q
            // q(i,j)= a(i,j)
            q.setElement(i, j, a.getElement(i, j));
        // q(i,n)= b(i,0)
        q.setElement(i, n, b.getElement(i, 0));
    }
    forward_solve(q);           // Realiza la eliminación gaussiana
    back_solve(q);              // Realiza sustitución inversa

    for (i=0; i<n; i++)
        // x(i,0)= q(i,n)
        x.setElement(i, 0, q.getElement(i, n));
}
```

```
private static void forward_solve(Matrix q) {
    int i, j, k, maxr, n;
    double t, pivot;
    n= q.getNumRows();

    for (i=0; i < n; i++) { // Busca la fila con el mayor elemento
        maxr= i;           // en esta columna, en la diagonal o bajo ella.
        for (j= i+1; j < n; j++)
            if (Math.abs(q.getElement(j,i)) >
                Math.abs(q.getElement(maxr,i)))
                maxr= j;
        if (maxr != i)    // Si la fila no es la actual, salta a otra
            for (k=i; k <= n; k++)
                { t= q.getElement(i,k); // t= q(i,k)
                  // q(i,k)= q(maxr, k)
                  q.setElement(i,k, q.getElement(maxr, k));
                  q.setElement(maxr, k, t); // q(maxr, k)= t
                }
        for (j= i+1; j < n; j++) // Calcula relación pivot
            { pivot= q.getElement(j,i)/q.getElement(i,i);
              for (k= n; k >=i; k--)
                  q.setElement(j, k, q.getElement(j,k)-
                                q.getElement(i,k)*pivot);
              // q(j,k) -= q(i,k)*pivot; Actualiza fila j bajo diagonal
            }
    }
}
```

Sustitución inversa

```
private static void back_solve(Matrix q)
{
    int j, k, n;
    double t; // t- temporal
    n= q.getNumRows();

    for (j=n-1; j >=0; j--) // Empieza en la última fila
    {
        t= 0.0;
        for (k= j+1; k < n; k++) // t += q(j,k)* q(k,n)
            t += q.getElement(j,k)* q.getElement(k,n);
        q.setElement(j, n,
            (q.getElement(j, n) -t)/q.getElement(j,j));
        // q(j, n)= (q(j, n) -t)/q(j,j);
    }
}
```

Progr. princ.

```
import Lecture21.Matrix;
import javax.swing.*;

public class GaussMain {
    public static void main(String[] args) {
        int i, j;
        double term;
        String input= JOptionPane.showInputDialog
            ("Introduzca el número de incógnitas");
        int n= Integer.parseInt(input);
        // Crea las matrices a, b, x
        Matrix a= new Matrix(n, n);
        Matrix b= new Matrix(n, 1);
        Matrix x= new Matrix(n, 1);
        // Introduce matriz a
        for (i = 0; i < a.getNumRows(); i++)
            for (j = 0; j < a.getNumCols(); j++) {
                input= JOptionPane.showInputDialog
                    ("Introduzca a["+i+"]["+j+"]");
                term= Double.parseDouble(input);
                a.setElement(i, j, term);
            }
    }
}
```

Progr. principal, 2

```
// Introduce el vector b como matriz de 1 columna
for (i = 0; i < b.getNumRows(); i++) {
    input= JOptionPane.showInputDialog
        ("Introduzca b["+i+"]");
    term= Double.parseDouble(input);
    b.setElement(i, 0, term);
}
gaussian(a, b, x);
System.out.println("Matriz a:");
a.print();
System.out.println("Vector b:");
b.print();
System.out.println("Vector solución x:");
x.print();
}
```

Variaciones

Múltiples lados derechos: aumenta Q, resuelve todas las ecuaciones a la vez

$$\left| \begin{array}{ccc|ccc} 3 & 1 & -2 & 5 & 7 & 87 \\ 2 & 4 & 3 & 35 & 75 & -1 \\ 1 & -3 & 0 & -5 & 38 & 52 \end{array} \right|$$

Inversión de matrices (no se suele hacer en la práctica)

$$\left| \begin{array}{ccc|ccc} 3 & 1 & -2 & 1 & 0 & 0 \\ 2 & 4 & 3 & 0 & 1 & 0 \\ 1 & -3 & 0 & 0 & 0 & 1 \end{array} \right| \longrightarrow \left| \begin{array}{ccc|ccc} \# & \# & \# & @ & @ & @ \\ 0 & \# & \# & @ & @ & @ \\ 0 & 0 & \# & @ & @ & @ \end{array} \right|$$

A

I

A⁻¹

Q

Ax=b

x= A⁻¹ b

Ejercicio

- Experimente con la aplicación de sistemas lineales. Descargue `GElim.java`:
 - Debe averiguar la ruta del directorio montado: en la vista Explorer, pulse sobre la pestaña FileSystems. La ruta de su directorio se mostrará como un enlace en su árbol del sistema de archivos (si dispone de varios directorios montados, seleccione el que esté más arriba en el árbol)
 - En el navegador Web, vaya a:
web.mit.edu/1.00/www/Lectures/Lecture22/GElim.java
Guarde el archivo descargado en su directorio (clic con el botón derecho, 'Save Link As' (guardar destino como))
 - Vuelva a la pestaña FileSystems de la vista Explorer. Haga clic con el botón secundario en el directorio y elija la opción 'Refresh' (actualizar). El archivo recién guardado deberá aparecer. Compile y ejecute `GElim.java`.

Aplicación de sistemas lineales

- Funciones de la aplicación:
 - Setup: construye una matriz cuadrada de dimensión N .
 - Undo: deshace la acción anterior (sólo se puede deshacer una vez).
 - Save: guarda la matriz actual.
 - Load: carga la matriz guardada.
 - Demo: carga una matriz preguardada; se trata de la matriz de ejemplo 3×3 que vimos en diapositivas anteriores.
 - Quit: cierra la aplicación
 - Pivot on Selection: Primero, debe seleccionar una celda haciendo clic en ella. A continuación, haga clic en "Pivot on Selection" para reducir a cero las filas por debajo de la selección.

Aplicación de sistemas lineales

- **Funciones de la aplicación(continuación):**
 - **Back Subst. on Selection:** Debe seleccionar una celda haciendo clic en ella. Para realizar una sustitución inversa correctamente, la celda seleccionada debe ser la única de su fila sin ceros (excepto para la celda "b"). Si no fuera el caso, el valor de la variable seleccionada se consideraría como una incógnita y no sería posible la sustitución inversa.
 - **Divide on selection:** divide toda la fila por el valor de la celda seleccionada.
 - **Solve:** resuelve directamente la matriz reduciéndola a su forma de "escalón de filas".
 - **Swap:** cambia la primera fila nombrada por la segunda.
 - **Commit:** multiplica una fila por una constante o sustituye la primera fila nombrada por una combinación de la segunda.

Ahora, la práctica

- **Experimente con las 3 matrices siguientes:**
 - El ejemplo de matriz 3x3 que vimos en diapositivas previas. Haga clic en "Demo" para cargarlo.

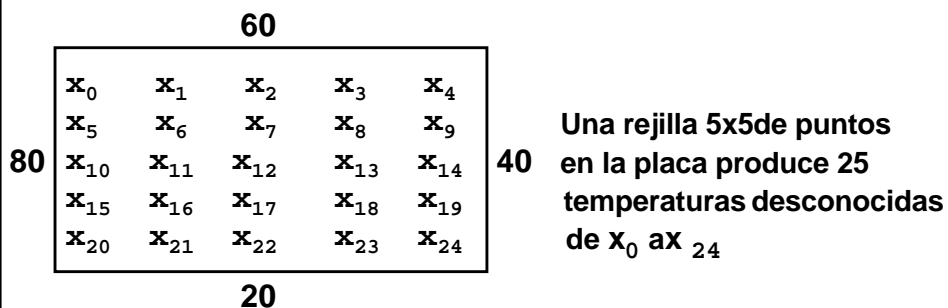
$$- \left| \begin{array}{ccc|c} 4 & 6 & -3 & 10 \\ 2 & 5 & 9 & 12 \\ 8 & 8 & -27 & 6 \end{array} \right|$$

$$- \left| \begin{array}{cccc|c} 12 & -13.5 & 3 & 0.5 & 2.75 \\ 8 & -9 & 4 & 2.5 & 3.5 \\ 3 & 6 & 1.5 & 2 & 4.25 \\ 2 & 1.5 & 4 & 12 & 6 \end{array} \right|$$

Uso de matrices

- Patrón común en software de ingeniería, científico y otro software analítico:
 - Generador de problemas(modelo,matriz de conjunto) :
 - Personalizado para una aplicación concreta(p. ej., transf. calor).
 - Usa la multiplicación, suma, etc. de matrices.
 - Solución al problema(sistema de ecuaciones lineales simultáneas):
 - Normalmente“preparado”: se puede obtener el código de una librería ya hecha o crear un código propio e incluirlo en una librería.
 - Generador de resultados(presenta los resultados en formato comprensible):
 - Personalizado para una aplicación concreta(a menudo, con gráficos, etc.)

Ejemplo de transferencia de calor



$$T = (T_{izq} + T_{dch} + T_{arriba} + T_{abajo})/4$$

Las temp. de los extremos son conocidas; las interiores, no. Esto da lugar a una matriz de 25x25 de ecuaciones lineales

Transferencia de calor,2

- **Nodo0:**

$$x_0 = (80 + x_1 + 60 + x_5)/4$$

$$4x_0 - x_1 - x_5 = 140$$

- **Nodo6:**

$$x_6 = (x_5 + x_7 + x_1 + x_{11})/4$$

$$4x_6 - x_5 - x_7 - x_1 - x_{11} = 0$$

- **Nodo interior:**

$$x_i = (x_{i-1} + x_{i+1} + x_{i-n} + x_{i+n})/4$$

$$4x_i - x_{i-1} - x_{i+1} - x_{i-n} - x_{i+n} = 0$$

Nodo	0	1	2	3	4	5	6	7
0	4	-1	0	0	0	-1	0	0
1	-1	4	-1	0	0	0	-1	0
2	0	-1	4	-1	0	0	0	-1
3	0	0	-1	4	-1	0	0	0
4	0	0	0	-1	4	0	0	0
5	-1	0	0	0	0	4	-1	0
6	0	-1	0	0	0	-1	4	-1
7	0	0	-1	0	0	0	-1	4

Transferencia de calor,3

25

$$\begin{matrix}
 a_{00} & a_{01} & a_{02} & \dots & a_{0,24} \\
 a_{10} & a_{11} & a_{12} & \dots & a_{1,24} \\
 a_{20} & a_{21} & a_{22} & \dots & a_{2,24} \\
 & & & \dots & \\
 a_{24,0} & a_{24,1} & & \dots & a_{24,24}
 \end{matrix}$$

$$\begin{matrix}
 x_0 \\
 x_1 \\
 x_2 \\
 \dots \\
 x_{24}
 \end{matrix}$$

$$\begin{matrix}
 b_0 \\
 b_1 \\
 b_2 \\
 \dots \\
 b_{24}
 \end{matrix}$$

25

A

x

b

Contiene 0, -1, 4
coeficientes en el
patrón (simple)

Temperaturas conocidas
(a menudo pero usan las
de los extremos al estar prox. a ellos)

25 temperaturas interiores desconocidas

Transf. de calor,4

Solución:

		60					
		67	61	57	54	50	
		68	60	54	50	45	
80		66	56	50	45	42	40
		61	49	43	39	38	
		50	38	33	31	32	
		20					

- Hay una clase `NumberFormatter` en Java para controlar el número de lugares decimales, etc.
- El ejemplo usa enteros para obtener un resultado más sencillo, pero en realidad, las temperaturas son dobles.
- Podría usar gráficos de color para mostrar gradientes.

Código de transf. de calor,1

```
package Lecture22;           // Incluye GaussMain.gaussian
import Lecture21.Matrix;    // Incluye la clase Matrix
public class Heat {         // Generador de problemas
    public static void main(String[] args) { // Temp. extremos
        double Te= 40.0, Tn=60.0, Tw=80.0, Ts=20.0;
        final int col= 5;
        final int row= 5;
        final int n= col * row;
        Matrix a= new Matrix(n,n);
        for (int i=0; i < n; i++)
            for (int j=0; j < n; j++) {
                if (i==j) // Elemento diagonal
                    a.setElement(i, j, 4.0);
                else ...
                    // Complete este código: defina elemento en 1.0
                    // si el nodo i está junto al nodo j
                    // Cuidado con los extremos (p. ej. el nodo
                    // 5 no es adyacente al nodo 4)
                else
                    a.setElement(i, j, 0.0);
            }
    }
}
```

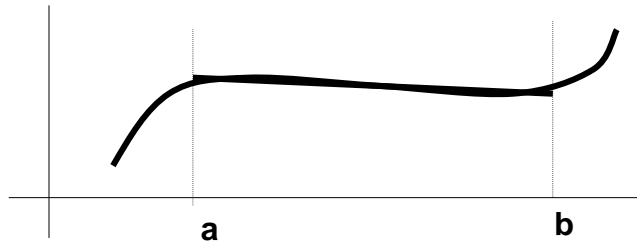
Código de transf. de calor,2

```
Matrix x= new Matrix(n, 1);           // Temps. desconocidas
for (int i=0; i < n; i++)
    x.setElement(i, 0, 0.0);         // Inicializa en ceros
Matrix b= new Matrix(n, 1);           // Temps. conocidas
for (int i=0; i < n; i++) {
    b.setElement(i, 0, 0.0);
    if (i < col)                      // Cerca del extremo norte
        b.setElement(i, 0, b.getElement(i,0)+Tn);
    // Complete el código para otros extremos, sin 'elses'
    // Sume siempre la temperatura del extremo a b
}
GaussMain.gaussian(a, b, x);         // Solución al problema
System.out.println("Parrilla de temperaturas:"); // Generador result.
// Muestre el resultado en una rejilla 5x5. Use enteros.
}

// Descargue Heat2.java del sitio Web y complete el código
// Descargue también Matrix.java y GaussMain.java
```

Otras aplicaciones

- Resolución de sistemas con miles o millones de ecuaciones lineales o desigualdades:
 - Redes, mecánica, fluidos, materiales, economía, etc.
 - Suelen linealizar sistemas en un rango definido



- Las rutinas de esta clase son válidas para cientos de ecuaciones:
 - No funcionan bien al buscar sistemas colineales. Consulte la ref. de fórmulas
- Si hay más: ref. fórmulas.