

Métodos Numéricos (SC-854)

Solución de sistemas de ecuaciones lineales

© M. Valenzuela 2007
(21 de agosto de 2007)

1. Matrices

Definición. 1 Una matriz $n \times m$ es un arreglo rectangular de elementos con n filas (o renglones) y m columnas en el cual no sólo es importante el valor de los elementos sino también su posición en el arreglo.

Ejemplo:

$$\mathbf{A} = (a_{ij}) = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix} \quad (1)$$

Las matrices se denotan con mayúsculas (con negritas) y sus elementos con minúsculas.

2. Vectores

Definición. 2 Una matriz $1 \times n$,

$$\mathbf{B} = [b_{11} \quad b_{12} \quad \cdots \quad b_{1n}], \quad (2)$$

se le llama un vector renglón de n dimensiones.

Definición. 3 Una matriz $n \times 1$,

$$\mathbf{C} = \begin{bmatrix} c_{11} \\ c_{12} \\ \vdots \\ c_{1n} \end{bmatrix}, \quad (3)$$

se le llama un vector columna de n dimensiones.

Usualmente, los subíndices innecesarios se omiten, de manera que

$$\mathbf{y} = [y_1 \quad y_2 \quad \cdots \quad y_m], \quad (4)$$

denota un vector renglón de m dimensiones, y

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad (5)$$

denota un vector columna de n dimensiones. Los vectores se denotan con minúsculas y negritas.

La norma de un vector \mathbf{x} se define de la siguiente manera:

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^n |x_i|^p \right)^{1/p} \quad (6)$$

Nótese que

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^n |x_i|^2} = |\mathbf{x}|, \quad (7)$$

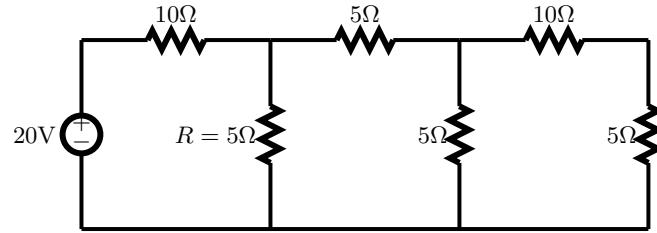


Figura 1: Circuito eléctrico que puede ser descrito mediante un sistema de ecuaciones simultáneas.

es decir, la magnitud del vector. Nótese también que

$$\|\mathbf{x}\|_{\infty} = \lim_{p \rightarrow \infty} \left(\sum_{i=1}^n |x_i|^p \right)^{1/p} = \max_i (|x_i|), \quad (8)$$

es decir, el máximo de los valores absolutos de las componentes x_i del vector.

3. Por qué se requieren métodos para resolver sistemas de ecuaciones lineales

Muchos problemas pueden ser descritos mediante un sistemas de ecuaciones lineales. Por ejemplo, considere el circuito eléctrico mostrado en la figura 1.

Las ecuaciones de malla que describen a este circuito son las siguientes:

$$\begin{aligned} 15i_1 - 5i_2 &= 20 \\ -5i_1 + 15i_2 - 5i_3 &= 0 \\ -5i_2 + 20i_3 &= 0 \end{aligned} \quad (9)$$

A partir de las ecuaciones de malla se pueden obtener todas las corrientes, voltajes, y pontencial de los elementos del circuito. Por ejemplo, la corriente de la resistencia R es $i_1 - i_2$.

4. Representación de sistemas de ecuaciones

Un sistema de ecuaciones lineales simulatáneas de la forma

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n \end{aligned}$$

puede representarse mediante una matriz $n \times (n + 1)$.

A partir de la matriz \mathbf{A} y el vector \mathbf{b}

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \quad (10)$$

$$\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \quad (11)$$

se forma la matriz aumentada

$$\tilde{\mathbf{A}} = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{array} \right] \quad (12)$$

Esta matriz aumentada representa la ecuación vectorial $\mathbf{Ax} = \mathbf{b}$.

5. Ejemplo de circuitos eléctricos

Definiendo \mathbf{R} , \mathbf{i} , y \mathbf{v} :

$$\mathbf{R} = \begin{bmatrix} 15 & -5 & 0 \\ -5 & 15 & -5 \\ 0 & -5 & 20 \end{bmatrix} \quad (13)$$

$$\mathbf{i} = \begin{bmatrix} i_1 \\ i_2 \\ i_3 \end{bmatrix} \quad \mathbf{v} = \begin{bmatrix} 20 \\ 0 \\ 0 \end{bmatrix} \quad (14)$$

Podemos expresar el juego de ecuaciones como

$$\mathbf{R}\mathbf{i} = \mathbf{v} = \begin{bmatrix} 15 & -5 & 0 \\ -5 & 15 & -5 \\ 0 & -5 & 20 \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \\ i_3 \end{bmatrix} = \begin{bmatrix} 20 \\ 0 \\ 0 \end{bmatrix} \quad (15)$$

Que puede representarse mediante la matriz aumentada:

$$\tilde{\mathbf{R}} = \begin{bmatrix} 15 & -5 & 0 & 20 \\ -5 & 15 & -5 & 0 \\ 0 & -5 & 20 & 0 \end{bmatrix} \quad (16)$$

6. Operaciones elementales de renglón

Como la matriz aumentada $\tilde{\mathbf{A}}$ representa un sistema de ecuaciones simultáneas, es posible realizar las siguientes operaciones elementales de renglón manteniendo las igualdades de las ecuaciones representadas:

- Multiplicar un renglón por una constante
- Multiplicar un renglón por una constante y sumarlo a otro renglón

Los métodos de soluciones de sistemas de ecuaciones aplican estas operaciones sobre la matriz aumentada en forma ordenada y repetida. En las siguientes secciones se explican los siguientes métodos:

- Eliminación gaussiana (Gauss)
- Gauss-Jordan
- Montante

| Function EGaussiana(A,b) | |
|--------------------------|--|
| 1 | $\mathbf{A} \leftarrow [\mathbf{A}b]$; |
| 2 | for $i \leftarrow 1$ to n do |
| 3 | // Hacer ceros abajo de pivote |
| 3 | for $j \leftarrow i + 1$ to n do |
| 4 | $\mathbf{A}(j, :) \leftarrow \mathbf{A}(j, :) - \frac{\mathbf{A}(i, :)\mathbf{A}(j, i)}{\mathbf{A}(i, i)}$; |
| 5 | for $i \leftarrow n$ downto 1 do |
| | // Hacer pivote 1 |
| 6 | $\mathbf{A}(i, :) \leftarrow \frac{\mathbf{A}(i, :)}{\mathbf{A}(i, i)}$; |
| | // Hacer ceros arriba de pivote |
| 7 | for $j \leftarrow i - 1$ downto 1 do |
| 8 | $\mathbf{A}(j, :) \leftarrow \mathbf{A}(j, :) - \mathbf{A}(i, :)\mathbf{A}(j, i)$; |
| 9 | $\mathbf{x} \leftarrow \mathbf{A}(:, n + 1)$; |
| 10 | return \mathbf{x} |

Figura 2: Pseudocódigo que implementa el método de eliminación gaussiana para solución de sistemas de ecuaciones lineales.

7. Eliminación Gaussiana

Eliminación gaussiana aplica operaciones de renglón para resolver un sistema de ecuaciones simultáneas; su pseudocódigo se presenta en la figura 2, y en la figura 3 su implementación en MATLAB.

Para cada renglón, se define el elemento $a_{i,i}$ de la matriz aumentada como el *pivote*. Eliminación gaussiana opera en dos fases. Primero, para cada renglón empezando por el primer renglón, hace ceros en los elementos debajo del pivote (líneas 3 y 4). Segundo, para cada renglón empezando por el último renglón, hace el pivote igual a 1, y hace ceros arriba del pivote (líneas 5 a 8). La solución al sistema de ecuaciones queda en la última columna de la matriz aumentada (línea 9).

A continuación se presenta la solución del ejemplo del circuito eléctrico mediante eliminación gaussiana.

$$\tilde{\mathbf{R}} = \begin{bmatrix} 15 & -5 & 0 & 20 \\ -5 & 15 & -5 & 0 \\ 0 & -5 & 20 & 0 \end{bmatrix} \sim \begin{bmatrix} 15 & -5 & 0 & 20 \\ 0 & 13.3333 & -5 & 6.6667 \\ 0 & -5 & 20 & 0 \end{bmatrix} \sim \quad (17)$$

$$\begin{bmatrix} 15 & -5 & 0 & 20 \\ 0 & 13.3333 & -5 & 6.6667 \\ 0 & 0 & 18.1250 & 2.5 \end{bmatrix} \sim \begin{bmatrix} 15 & -5 & 0 & 20 \\ 0 & 13.3333 & -5 & 6.6667 \\ 0 & 0 & 1 & 0.1379 \end{bmatrix} \sim \quad (18)$$

$$\begin{bmatrix} 15 & -5 & 0 & 20 \\ 0 & 13.3333 & 0 & 7.3563 \\ 0 & 0 & 1 & 0.1379 \end{bmatrix} \sim \begin{bmatrix} 15 & 0 & 0 & 22.7586 \\ 0 & 1 & 0 & 0.5517 \\ 0 & 0 & 1 & 0.1379 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & 0 & 1.5172 \\ 0 & 1 & 0 & 0.5517 \\ 0 & 0 & 1 & 0.1379 \end{bmatrix} \quad (19)$$

de donde se tiene que las corrientes de malla son:

$$\mathbf{i} = \begin{bmatrix} 1.5172 \\ 0.5517 \\ 0.1379 \end{bmatrix} \quad (20)$$

```

function x = EGauss(A,b)

% Implementacion del metodo de eliminacion
% gaussiana para resolver sistemas
% de ecuaciones lineales.
%
% x = EGauss(A,b)
%
% Regresa x, la solucion del sistema Ax=b.

% 22 enero 2007
% Manuel Valenzuela

% Crear la matriz aumentada
A = [A b];
n = size(A,1);

for i=1:n
    % Hacer ceros en la columna i debajo de la fila i
    for j=i+1:n
        A(j,:) = A(j,:) - A(i,:)*A(j,i)/A(i,i);
    end
end

for i=n:-1:1
    % Hacer uno el elemento i,i
    A(i,:) = A(i,+)/A(i,i);

    % Hacer ceros en la columna i arriba de la fila i
    for j=i-1:-1:1
        A(j,:) = A(j,:) - A(i,:)*A(j,i);
    end
end

x = A(:,n+1);

```

Figura 3: Implementación en MATLAB del método de eliminación gaussiana para solución de sistemas de ecuaciones lineales.

8. Método de Gauss-Jordan

En la figuras 4 y 5 se presenta en la implementación del método Gauss-Jordan. El método de Gauss-Jordan es similar a eliminación gaussiana, pero primero hace el pivote igual a 1, y luego hace ceros en toda la columna del pivote. En el método de Gauss-Jordan primero se hace el pivote igual a 1 se hace el pivote igual a 1 (línea 3), Después se hacen cero los elementos arriba y abajo del pivote líneas 4 a 6). La solución al sistema de ecuaciones queda en la última columna de la matriz aumentada (línea 7).

A contiunación se muestra la solución del ejemplo del circuito eléctrico mediante Gauss-Jordan.

$$\tilde{\mathbf{R}} = \begin{bmatrix} 15 & -5 & 0 & 20 \\ -5 & 15 & -5 & 0 \\ 0 & -5 & 20 & 0 \end{bmatrix} \sim \begin{bmatrix} 1 & -0.3333 & 0 & 1.3333 \\ -5 & 15 & -5 & 0 \\ 0 & -5 & 20 & 0 \end{bmatrix} \sim \quad (21)$$

$$\begin{bmatrix} 1 & -0.3333 & 0 & 1.3333 \\ 0 & 13.3333 & -5 & 6.6667 \\ 0 & -5 & 20 & 0 \end{bmatrix} \sim \begin{bmatrix} 1 & -0.3333 & 0 & 1.3333 \\ 0 & 1 & -0.3750 & 0.5000 \\ 0 & -5 & 20 & 0 \end{bmatrix} \sim \quad (22)$$

$$\begin{bmatrix} 1 & 0 & -0.1250 & 1.5000 \\ 0 & 1 & -0.3750 & 0.5000 \\ 0 & 0 & 18.1250 & 2.5000 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & -0.1250 & 1.5000 \\ 0 & 1 & -0.3750 & 0.5000 \\ 0 & 0 & 1 & 0.1379 \end{bmatrix} \sim \quad (23)$$

$$\begin{bmatrix} 1 & 0 & 0 & 1.5172 \\ 0 & 1 & 0 & 0.5517 \\ 0 & 0 & 1 & 0.1379 \end{bmatrix} \quad (24)$$

9. Pivote máximo

Los algoritmos presentados pueden encontrar el problema de que el pivote sea cero, causando una división entre cero. Para resolver este problema se pueden intercambiar renglones para colocar una elemento diferente de cero en la diagonal principal. En la figura 6 se presenta la implementación de Gauss-Jordan donde se escoge el elemento de máximo valor absoluto como pivote.

```

Function GaussJordan(A,b)

1 A ← [Ab] ;
2 for i ← 1 to n do
    // Hacer pivote 1
3     A(i,:) ←  $\frac{A(i,:)}{A(i,i)}$  ;
4     for j ← 1 to n do
5         if j ≠ i then
6             // Hacer ceros arriba y abajo del pivote
             A(j,:) ← A(j,:) - A(j,i)A(i,:) ;
7 x ← A(:, n + 1) ;
8 return x

```

Figura 4: Pseudocódigo que implementa de método Gauss-Jordan para la solución de sistemas de ecuaciones lineales.

```
function x = GaussJ(A,b)
```

```

% Implementacion del metodo Gauss-Jordan
% para resolver sistemas de ecuaciones
% lineales.
%
% x = GaussJordan(A,b)
%
% Regresa x, la solucion del sistema Ax=b.

```

```

% 22 enero 2007
% Manuel Valenzuela

```

```

% Se crea la matriz aumentada

```

```

A = [A b];
n = size(A,1);

```

```

for i=1:n
    % Dividir renglon entre el pivote
    A(i,:) = A(i, :)/A(i,i);
    % Hacer ceros en la columna i
    for j=1:n
        if i~j
            A(j,:) = A(j,:) - A(i, :)*A(j,i);
        end
    end
end

x = A(:,n+1);

```

Figura 5: Implementación en MATLAB del método Gauss-Jordan para solución de sistemas de ecuaciones lineales.

```

Function GaussJordan(A,b)
1 A ← [Ab] ;
2 for i ← 1 to n do
3   m ← max(|A(i : n, i)|) ;
4   k ← renglón de m en A ;
5   Intercambiar renglones i y k de A ;
   // Hacer pivote 1
6   A(i, :) ← A(i, :) / A(i, i) ;
7   for j ← 1 to n do
   // Hacer ceros arriba y abajo del pivote
8     if j ≠ i then
9       A(j, :) ← A(j, :) - A(j, i)A(i, :) ;
10 x ← A(:, n + 1) ;
11 return x

```

Figura 6: Pseudocódigo que implementa el método de Gauss-Jordan con pivote máximo para solución de sistemas de ecuaciones lineales.

10. Montante

El método de Montante, que se presenta en las figuras 7 y 8, resuelve un sistema de ecuaciones simultáneas haciendo operaciones que mantienen el número de decimales que tiene los datos originales hasta el último paso, donde se realiza la división entre el determinante.

Ejemplo del método Montante:

$$\tilde{\mathbf{A}} = \begin{bmatrix} 15 & -5 & 0 & 20 \\ -5 & 15 & -5 & 0 \\ 0 & -5 & 20 & 0 \end{bmatrix} \sim \begin{bmatrix} 15 & -5 & 0 & 20 \\ 0 & 200 & -75 & 100 \\ 0 & -75 & 300 & 0 \end{bmatrix} \sim \quad (25)$$

$$\begin{bmatrix} 200 & 0 & -25 & 300 \\ 0 & 200 & -75 & 100 \\ 0 & 0 & 3625 & 500 \end{bmatrix} \sim \begin{bmatrix} 3625 & 0 & 0 & 5500 \\ 0 & 3625 & 0 & 2000 \\ 0 & 0 & 3625 & 500 \end{bmatrix} \sim \quad (26)$$

$$\begin{bmatrix} 1 & 0 & 0 & 1.5172 \\ 0 & 1 & 0 & 0.5517 \\ 0 & 0 & 1 & 0.1379 \end{bmatrix} \quad (27)$$

11. Matriz inversa

Los métodos de eliminación gaussiana, Gauss-Jordan, y Montante pueden ser utilizados para encontrar la inversa de una matriz. En este caso, la matriz aumentada será la matriz de original y la matriz identidad.

$$\begin{bmatrix} 15 & -5 & 0 & 1 & 0 & 0 \\ -5 & 15 & -5 & 0 & 1 & 0 \\ 0 & -5 & 20 & 0 & 0 & 1 \end{bmatrix} \sim \begin{bmatrix} 15 & -5 & 0 & 1 & 0 & 0 \\ 0 & 200 & -75 & 5 & 15 & 0 \\ 0 & -75 & 300 & 0 & 0 & 15 \end{bmatrix} \sim \quad (28)$$

$$\begin{bmatrix} 200 & 0 & -25 & 15 & 5 & 0 \\ 0 & 200 & -75 & 5 & 15 & 0 \\ 0 & 0 & 3625 & 25 & 75 & 200 \end{bmatrix} \sim \begin{bmatrix} 3625 & 0 & 0 & 275 & 100 & 25 \\ 0 & 3625 & 0 & 100 & 300 & 75 \\ 0 & 0 & 3625 & 25 & 75 & 200 \end{bmatrix} \quad (29)$$

```

Function Montante(A,b)
1  A ← [Ab] ;
2  pivAnt ← 1 ;
3  for i ← 1 to n do
4      piv ← A(k,k) ;
5      for j ← 1 to n do
6          // Hacer ceros arriba y abajo del pivote
7          if j ≠ i then
8              A(j,:) ←  $\frac{\mathbf{A}(j,:)piv - \mathbf{A}(i,:\mathbf{A}(j,i)}{pivAnt}$  ;
9          pivAnt ← piv ;
10     x ←  $\frac{\mathbf{A}(:,n+1)}{piv}$  ;
11     det ← A(n,n) ;
12     return x, det

```

Figura 7: Pseudocódigo que implementa el método de Montante para la solución de sistemas de ecuaciones lineales.

```
function x = Mont(A,b)
```

```

% Implementacion del metodo Montante
% para resolver sistemas de
% ecuaciones lineales.
%
% x = Montante(A,b)
%
% Regresa x, la solucion del sistema Ax=b.

```

```

% 22 enero 2007
% Manuel Valenzuela

```

```

% Se crea la matriz aumentada

```

```

A = [A b];
n = size(A,1);

```

```

pivAnt = 1; % pivote inicial
for i=1:n
    % pivote actual
    piv = A(i,i);
    % Hacer ceros en la columna i
    for j=1:n
        if j~i
            A(j,:) = (A(j,)*piv - A(i,)*A(j,i))/pivAnt;
        end
    end
    % Guardar el pivote anterior
    pivAnt = piv;
end
% Dividir entre el ultimo pivote (determinante)
A = A/piv;

x = A(:,n+1);

```

Figura 8: Implementación en MATLAB del método de Montante para la solución de sistemas de ecuaciones lineales.

$$\sim \begin{bmatrix} 1 & 0 & 0 & 0.0759 & 0.0276 & 0.0069 \\ 0 & 1 & 0 & 0.0276 & 0.0828 & 0.0207 \\ 0 & 0 & 1 & 0.0069 & 0.0207 & 0.0552 \end{bmatrix} \quad (30)$$

La inversa son las últimas n columnas de la matriz aumentada:

$$\mathbf{A}^{-1} = \begin{bmatrix} 0.0759 & 0.0276 & 0.0069 \\ 0.0276 & 0.0828 & 0.0207 \\ 0.0069 & 0.0207 & 0.0552 \end{bmatrix} \quad (31)$$

12. Métodos Iterativos: Jacobi

Dado un sistema de ecuaciones de la forma:

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \quad (32)$$

$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \quad (33)$$

$$\vdots \quad (34)$$

$$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \quad (35)$$

si se despeja la variable x_i de cada ecuación se obtiene lo siguiente:

$$x_1 = -\frac{a_{12}}{a_{11}}x_2 - \frac{a_{13}}{a_{11}}x_3 - \cdots - \frac{a_{1n}}{a_{11}}x_n + \frac{b_1}{a_{11}} \quad (36)$$

$$x_2 = -\frac{a_{21}}{a_{22}}x_1 - \frac{a_{23}}{a_{22}}x_3 - \cdots - \frac{a_{2n}}{a_{22}}x_n + \frac{b_2}{a_{22}} \quad (37)$$

$$\vdots \quad (38)$$

$$x_n = -\frac{a_{n1}}{a_{nn}}x_1 - \frac{a_{n2}}{a_{nn}}x_2 - \cdots - \frac{a_{n,n-1}}{a_{nn}}x_{n-1} + \frac{b_n}{a_{nn}} \quad (39)$$

El sistema anterior, puede usarse como una fórmula recursiva, es decir,

$$x_1(t+1) = -\frac{a_{12}}{a_{11}}x_2(t) - \frac{a_{13}}{a_{11}}x_3(t) - \cdots - \frac{a_{1n}}{a_{11}}x_n(t) + \frac{b_1}{a_{11}} \quad (40)$$

$$x_2(t+1) = -\frac{a_{21}}{a_{22}}x_1(t) - \frac{a_{23}}{a_{22}}x_3(t) - \cdots - \frac{a_{2n}}{a_{22}}x_n(t) + \frac{b_2}{a_{22}} \quad (41)$$

$$\vdots \quad (42)$$

$$x_n(t+1) = -\frac{a_{n1}}{a_{nn}}x_1(t) - \frac{a_{n2}}{a_{nn}}x_2(t) - \cdots - \frac{a_{n,n-1}}{a_{nn}}x_{n-1}(t) + \frac{b_n}{a_{nn}} \quad (43)$$

puede usarse para obtener los valores de $x_i(t+1)$ en función de los valores de $x_i(t)$.

Si definimos la matriz \mathbf{T} y el vector \mathbf{c} de la siguiente manera,

$$\mathbf{T} = \begin{bmatrix} 0 & -\frac{a_{12}}{a_{11}} & -\frac{a_{13}}{a_{11}} & \cdots & -\frac{a_{1n}}{a_{11}} \\ -\frac{a_{21}}{a_{22}} & 0 & -\frac{a_{23}}{a_{22}} & \cdots & -\frac{a_{2n}}{a_{22}} \\ -\frac{a_{31}}{a_{33}} & -\frac{a_{32}}{a_{33}} & 0 & \cdots & -\frac{a_{3n}}{a_{33}} \\ \vdots & & & \ddots & \vdots \\ -\frac{a_{n1}}{a_{nn}} & -\frac{a_{n2}}{a_{nn}} & \cdots & -\frac{a_{n,n-1}}{a_{nn}} & 0 \end{bmatrix} \quad \mathbf{c} = \begin{bmatrix} \frac{b_1}{a_{11}} \\ \frac{b_2}{a_{22}} \\ \frac{b_3}{a_{33}} \\ \vdots \\ \frac{b_n}{a_{nn}} \end{bmatrix} \quad (44)$$

| Function Jacobi (A, b, x_0) | |
|--|--|
| 1 | Obtener matriz \mathbf{T} ; |
| 2 | Obtener vector \mathbf{c} ; |
| 3 | $\mathbf{x} \leftarrow \mathbf{x}_0$; |
| 4 | repeat |
| 5 | $\mathbf{x}_{\text{ant}} \leftarrow \mathbf{x}$; |
| 6 | $\mathbf{x} \leftarrow \mathbf{T}\mathbf{x} + \mathbf{c}$; |
| 7 | until $\frac{\ \mathbf{x} - \mathbf{x}_{\text{ant}}\ _{\infty}}{\ \mathbf{x}\ _{\infty}} < \varepsilon$; |
| 8 | return \mathbf{x} |

Figura 9: Pseudocódigo que implementa el método de Jacobi para resolver en forma iterativa sistemas de ecuaciones lineales. En la línea 7 se ha tomado la norma infinita, $\|\cdot\|_{\infty}$, para definir el criterio de terminación, pero es posible tomar otra norma.

se pueden escribir las ecuaciones recursivas en forma matricial:

$$\mathbf{x}(t+1) = \mathbf{T}\mathbf{x}(t) + \mathbf{c} \quad (45)$$

Para evitar el uso de la variable t podemos escribir la ecuación en forma de asignación:

$$\mathbf{x} \leftarrow \mathbf{T}\mathbf{x} + \mathbf{c} \quad (46)$$

En las figuras 9 y 10 se presenta la implementación del método de Jacobi.

13. Métodos Iterativos: Gauss-Seidel

En las ecuaciones recursivas, es posible utilizar inmediatamente los valores obtenidos para calcular los siguientes valores, es decir,

$$x_1(t+1) = -\frac{a_{12}}{a_{11}}x_2(t) - \frac{a_{13}}{a_{11}}x_3(t) - \dots - \frac{a_{1n}}{a_{11}}x_n(t) + \frac{b_1}{a_{11}} \quad (47)$$

$$x_2(t+1) = -\frac{a_{21}}{a_{22}}x_1(t+1) - \frac{a_{23}}{a_{22}}x_3(t) - \dots - \frac{a_{2n}}{a_{22}}x_n(t) + \frac{b_2}{a_{22}} \quad (48)$$

$$x_3(t+1) = -\frac{a_{31}}{a_{33}}x_1(t+1) - \frac{a_{32}}{a_{33}}x_2(t+1) - \dots - \frac{a_{3n}}{a_{33}}x_n(t) + \frac{b_3}{a_{33}} \quad (49)$$

$$\vdots \quad (50)$$

$$x_n(t+1) = -\frac{a_{n1}}{a_{nn}}x_1(t+1) - \frac{a_{n2}}{a_{nn}}x_2(t+1) - \dots - \frac{a_{n,n-1}}{a_{nn}}x_{n-1}(t+1) + \frac{b_n}{a_{nn}} \quad (51)$$

El utilizar los valores de x_i que se acaban de calcular para calcular los siguientes valores permite que el método converja más rápidamente a una solución.

Las ecuaciones recursivas se pueden escribir en forma matricial de la siguiente manera:

$$x_i \leftarrow \mathbf{T}(i, :)\mathbf{x} + c_i \quad (52)$$

donde $\mathbf{T}(i, :)$ representa la fila i de la matriz \mathbf{T} , y la regla debe aplicarse en orden para $i = 1, 2, \dots, n$.

En las figuras 11 y 12 se presenta la implementación del método de Gauss-Seidel.

```

function x = Jaco(A,b,x0)
% Implementacion del metodo Jacobi para la
% solucion de sistemas de ecuaciones, tomando
% como aproximacion inicial x0.
%
% x = Jaco(A,b,x0)
%
% Regresa x, la solucion del sistema Ax=b.
% El criterio de terminacion es que
% norm(x-xAnt,Inf)/norm(x,Inf)<eps.

% 17 agosto 2007
% Manuel Valenzuela

n = size(A,1);
eps = 0.001;

% Se obtienen el vector c y la matriz T
c = b./diag(A);
T = zeros(n);
for i=1:n
    T(i,:) = A(i,:)/A(i,i);
end
T = -T-eye(n);

x = x0;
xAnt = x;
while 1
    x = T*x + c;
    if (norm(x-xAnt,Inf)/norm(x,Inf)<eps)
        break
    end
    xAnt = x;
end

```

Figura 10: Implementación en MATLAB del método de Jacobi para resolver en forma iterativa sistemas de ecuaciones lineales.

| Function GaussSeidel(A,b,x0) | |
|------------------------------|--|
| 1 | Obtener matriz T ; |
| 2 | Obtener vector c ; |
| 3 | $\mathbf{x}_{ant} \leftarrow \mathbf{x}_0$; |
| 4 | repeat |
| 5 | for $i \leftarrow n$ do |
| 6 | $x_i \leftarrow \mathbf{T}(i, :)\mathbf{x} + c_i$; |
| 7 | $\mathbf{x}_{ant} \leftarrow \mathbf{x}$; |
| 8 | until $\frac{\ \mathbf{x} - \mathbf{x}_{ant}\ }{\ \mathbf{x}\ _\infty} < \varepsilon$; |
| 9 | return x |

Figura 11: Pseudocódigo que implementa el método de Gauss-Seidel para resolver en forma iterativa sistemas de ecuaciones lineales.

```

function x = GaussS(A,b,x0)
% Implementacion del metodo Gauss-Seidel para
% la solucion de sistemas de ecuaciones,
% tomando como aproximacion inicial x0.
%
% x = GaussSeidel(A,b,x0)
%
% Regresa x, la solucion del sistema Ax=b.
% El criterio de terminacion es que
% norm(x-xAnt,Inf)/norm(x,Inf)<eps.

% 17 agosto 2007
% Manuel Valenzuela

n = size(A,1);
eps = 0.001;

% Se obtienen el vector c y la matriz T
c = b./diag(A);

T = zeros(n);
for i=1:n
    T(i,:) = A(i,:)/A(i,i);
end
T = -T.-eye(n);

x = x0;
xAnt = x;
j = 0;
while 1
    j = j+1;
    for i=1:n
        x(i) = T(i,:)*x + c(i);
    end
    if (norm(x-xAnt,Inf)/norm(x,Inf)<eps)
        break
    end
    xAnt = x;
end

```

Figura 12: Implementación en MATLAB del método de Gauss-Seidel para resolver en forma iterativa sistemas de ecuaciones lineales.

14. Ejemplo de Jacobi y Gauss-Seidel

Para el ejemplo del circuito eléctrico se tiene que

$$\mathbf{T} = \begin{bmatrix} 0 & 0.3333 & 0 \\ 0.3333 & 0 & 0.3333 \\ 0 & 0.2500 & 0 \end{bmatrix} \quad \mathbf{c} = \begin{bmatrix} 1.3333 \\ 0 \\ 0 \end{bmatrix} \quad (53)$$

Jacobi:

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|-------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| i_1 | 1.3333 | 1.3333 | 1.4815 | 1.4815 | 1.5103 | 1.5103 | 1.5159 | 1.5159 | 1.5170 | 1.5170 |
| i_2 | 0 | 0.4444 | 0.4444 | 0.5309 | 0.5309 | 0.5477 | 0.5477 | 0.5509 | 0.5509 | 0.5509 |
| i_3 | 0 | 0 | 0.1111 | 0.1111 | 0.1327 | 0.1327 | 0.1369 | 0.1369 | 0.1377 | 0.1377 |

Gauss-Seidel

| | 1 | 2 | 3 | 4 | 5 |
|-------|--------|--------|--------|--------|--------|
| i_1 | 1.3333 | 1.4815 | 1.5103 | 1.5159 | 1.5170 |
| i_2 | 0.4444 | 0.5309 | 0.5477 | 0.5509 | 0.5516 |
| i_3 | 0.1111 | 0.1327 | 0.1369 | 0.1377 | 0.1379 |

En general, Gauss-Seidel es más rápido que Jacobi, es decir, converge en menos iteraciones a la solución correcta.