Universidad Nacional Agraria la Molina
Escuela de Postgrado
Maestría en Ingeniería de Recursos Hídricos

Métodos Numéricos en Recursos Hídricos

Sesión 5:
Álgebra Lineal Numérica.

Eusebio Ingol Blanco, Ph.D., A.M.ASCE
Introducción

• A matrix consists of a rectangular array of elements represented by a single symbol (example: $[A]$).
• An individual entry of a matrix is an element (example: $a_{23}$)

\[
\begin{align*}
    a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots &= b_1 \\
    a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots &= b_2 \\
    a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \cdots &= b_3 \\
    \vdots
\end{align*}
\]

\[
\begin{bmatrix}
    a_{11} & a_{12} & a_{13} & \cdots \\
    a_{21} & a_{22} & a_{23} & \cdots \\
    a_{31} & a_{32} & a_{33} & \cdots \\
    \vdots & \vdots & \vdots & \ddots
\end{bmatrix}
\begin{bmatrix}
    x_1 \\
    x_2 \\
    x_3 \\
    \vdots
\end{bmatrix}
= 
\begin{bmatrix}
    b_1 \\
    b_2 \\
    b_3 \\
    \vdots
\end{bmatrix}
\]
Introducción

• A horizontal set of elements is called a row and a vertical set of elements is called a column.
• The first subscript of an element indicates the row while the second indicates the column.
• The size of a matrix is given as $m$ rows by $n$ columns, or simply $m$ by $n$ (or $m \times n$).
• $1 \times n$ matrices are row vectors.
• $m \times 1$ matrices are column vectors.
Matrices Especiales

- Matrices where $m=n$ are called *square matrices*.
- There are a number of special forms of square matrices:

<table>
<thead>
<tr>
<th>Symmetric</th>
<th>Diagonal</th>
<th>Identity</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[A]=\begin{bmatrix} 5 &amp; 1 &amp; 2 \ 1 &amp; 3 &amp; 7 \ 2 &amp; 7 &amp; 8 \end{bmatrix}$</td>
<td>$[A]=\begin{bmatrix} a_{11} \ \end{bmatrix}$</td>
<td>$[A]=\begin{bmatrix} 1 \ \end{bmatrix}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Upper Triangular</th>
<th>Lower Triangular</th>
<th>Banded</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[A]=\begin{bmatrix} a_{11} &amp; a_{12} &amp; a_{13} \ a_{22} &amp; a_{23} &amp; \ \ a_{33} \end{bmatrix}$</td>
<td>$[A]=\begin{bmatrix} a_{11} \ a_{21} &amp; a_{22} \ a_{31} &amp; a_{32} &amp; a_{33} \end{bmatrix}$</td>
<td>$[A]=\begin{bmatrix} a_{11} &amp; a_{12} \ a_{21} &amp; a_{22} &amp; a_{23} \ a_{32} &amp; a_{33} &amp; a_{34} \ a_{43} &amp; a_{44} \end{bmatrix}$</td>
</tr>
</tbody>
</table>
Operación con Matrices

• Two matrices are considered equal if and only if every element in the first matrix is equal to every corresponding element in the second. This means the two matrices must be the same size.

• Matrix addition and subtraction are performed by adding or subtracting the corresponding elements. This requires that the two matrices be the same size.

• Scalar matrix multiplication is performed by multiplying each element by the same scalar.
The elements in the matrix \([C]\) that results from multiplying matrices \([A]\) and \([B]\) are calculated using:

\[
c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}
\]
The inverse of a square, nonsingular matrix \([A]\) is that matrix which, when multiplied by \([A]\), yields the identity matrix.

\[
[A][A]^{-1} = [A]^{-1}[A] = [I]
\]

The transpose of a matrix involves transforming its rows into columns and its columns into rows.

\[(a_{ij})^T = a_{ji}\]
Matrices provide a concise notation for representing and solving simultaneous linear equations:

\[ \begin{align*}
    a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\
    a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2 \\
    a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3
\end{align*} \]

\[
\begin{bmatrix}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23} \\
a_{31} & a_{32} & a_{33}
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix} =
\begin{bmatrix}
b_1 \\
b_2 \\
b_3
\end{bmatrix}
\]

\[ [A] \{x\} = \{b\]
• MATLAB provides two direct ways to solve systems of linear algebraic equations \([A]\{x\} = \{b\}:
  
  – Left-division
  \[ x = A\backslash b \]
  
  – Matrix inversion
  \[ x = \text{inv}(A) \ast b \]

• The matrix inverse is less efficient than left-division and also only works for square, non-singular systems.
Método de Grafico

- For small sets of simultaneous equations, graphing them and determining the location of the intercept provides a solution.
Método de Grafico

• Graphing the equations can also show systems where:
  a) No solution exists
  b) Infinite solutions exist
  c) System is ill-conditioned
Determinantes

• The determinant $D = |A|$ of a matrix is formed from the coefficients of $[A]$.
• Determinants for small matrices are:

1×1

$$|a_{11}| = a_{11}$$

2×2

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}$$

3×3

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11} \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12} \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13} \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$$

• Determinants for matrices larger than 3 x 3 can be very complicated.
Regla de Cramer

- *Cramer’s Rule* states that each unknown in a system of linear algebraic equations may be expressed as a fraction of two determinants with denominator $D$ and with the numerator obtained from $D$ by replacing the column of coefficients of the unknown in question by the constants $b_1, b_2, \ldots, b_n$. 
Regla de Cramer

• Find \( x_2 \) in the following system of equations:

\[
\begin{align*}
0.3x_1 + 0.52x_2 + x_3 &= -0.01 \\
0.5x_1 + x_2 + 1.9x_3 &= 0.67 \\
0.1x_1 + 0.3x_2 + 0.5x_3 &= -0.44
\end{align*}
\]

• Find the determinant \( D \)

\[
D = \begin{vmatrix}
0.3 & 0.52 & 1 \\
0.5 & 1 & 1.9 \\
0.1 & 0.3 & 0.5
\end{vmatrix} = 0.3 \begin{vmatrix}
1 & 1.9 \\
0.3 & 0.5
\end{vmatrix} - 0.52 \begin{vmatrix}
0.5 & 1.9 \\
0.1 & 0.5
\end{vmatrix} + 1 \begin{vmatrix}
0.5 & 1 \\
0.1 & 0.4
\end{vmatrix} = 0.0649
\]

• Find determinant \( D_2 \) by replacing \( D \)'s second column with \( b \)

\[
D_2 = \begin{vmatrix}
0.3 & -0.01 & 1 \\
0.5 & 0.67 & 1.9 \\
0.1 & -0.44 & 0.5
\end{vmatrix} = 0.3 \begin{vmatrix}
0.67 & 1.9 \\
-0.44 & 0.5
\end{vmatrix} - 0.01 \begin{vmatrix}
0.5 & 1.9 \\
0.1 & 0.5
\end{vmatrix} + 1 \begin{vmatrix}
0.5 & 0.67 \\
0.1 & -0.44
\end{vmatrix} = 0.0649
\]

• Divide

\[
x_2 = \frac{D_2}{D} = \frac{0.0649}{-0.0022} = -29.5
\]
Eliminación Naive Gauss

• For larger systems, Cramer’s Rule can become unwieldy.

• Instead, a sequential process of removing unknowns from equations using forward elimination followed by back substitution may be used - this is Gauss elimination.

• “Naive” Gauss elimination simply means the process does not check for potential problems resulting from division by zero.
Eliminación Naive Gauss

- **Forward elimination**
  - Starting with the first row, add or subtract multiples of that row to eliminate the first coefficient from the second row and beyond.
  - Continue this process with the second row to remove the second coefficient from the third row and beyond.
  - Stop when an upper triangular matrix remains.

- **Back substitution**
  - Starting with the *last* row, solve for the unknown, then substitute that value into the next highest row.
  - Because of the upper-triangular nature of the matrix, each row will contain only one more unknown.
Integración

```matlab
function x = GaussNaive(A,b)
    % GaussNaive: naive Gauss elimination
    % x = GaussNaive(A,b): Gauss elimination without pivoting.
    % input:
    % A = coefficient matrix
    % b = right hand side vector
    % output:
    % x = solution vector
    [m,n] = size(A);
    if m~=n, error('Matrix A must be square'); end
    nb = n+1;
    Aug = [A b];
    % forward elimination
    for k = 1:n-1
        for i = k+1:n
            factor = Aug(i,k)/Aug(k,k);
            Aug(i,k:nb) = Aug(i,k:nb) - factor*Aug(k,k:nb);
        end
    end
    % back substitution
    x = zeros(n,1);
    x(n) = Aug(n,nb)/Aug(n,n);
    for i = n-1:-1:1
        x(i) = (Aug(i,nb) - Aug(i,i+1:n)*x(i+1:n))/Aug(i,i);
    end
```
Usando la Función

Using the function into MATLAB

```matlab
>> A = [1,5,6;7,4,2;-3,6,7]
A =
    1  5  6
    7  4  2
   -3  6  7
>> b = [10;12;14];
>> b = [10;12;14]

b =
    10
    12
    14

>> [x] = GaussNaive(A,b)

x =
   -0.3692
    4.7692
   -2.2462
```

```matlab
>> A = [1,5,6;7,4,2;-3,6,7];
>> b = [10;12;14];
>> rref([A b])

ans =
  1.0000    0    0  -0.3692
    0    1.0000   0  4.7692
    0    0    1.0000 -2.2462
```