Cálculo Simbólico con MATLAB. Introducción a la Programación.

Bloque II: Vectores y Matrices. Operaciones con Vectores y Matrices.

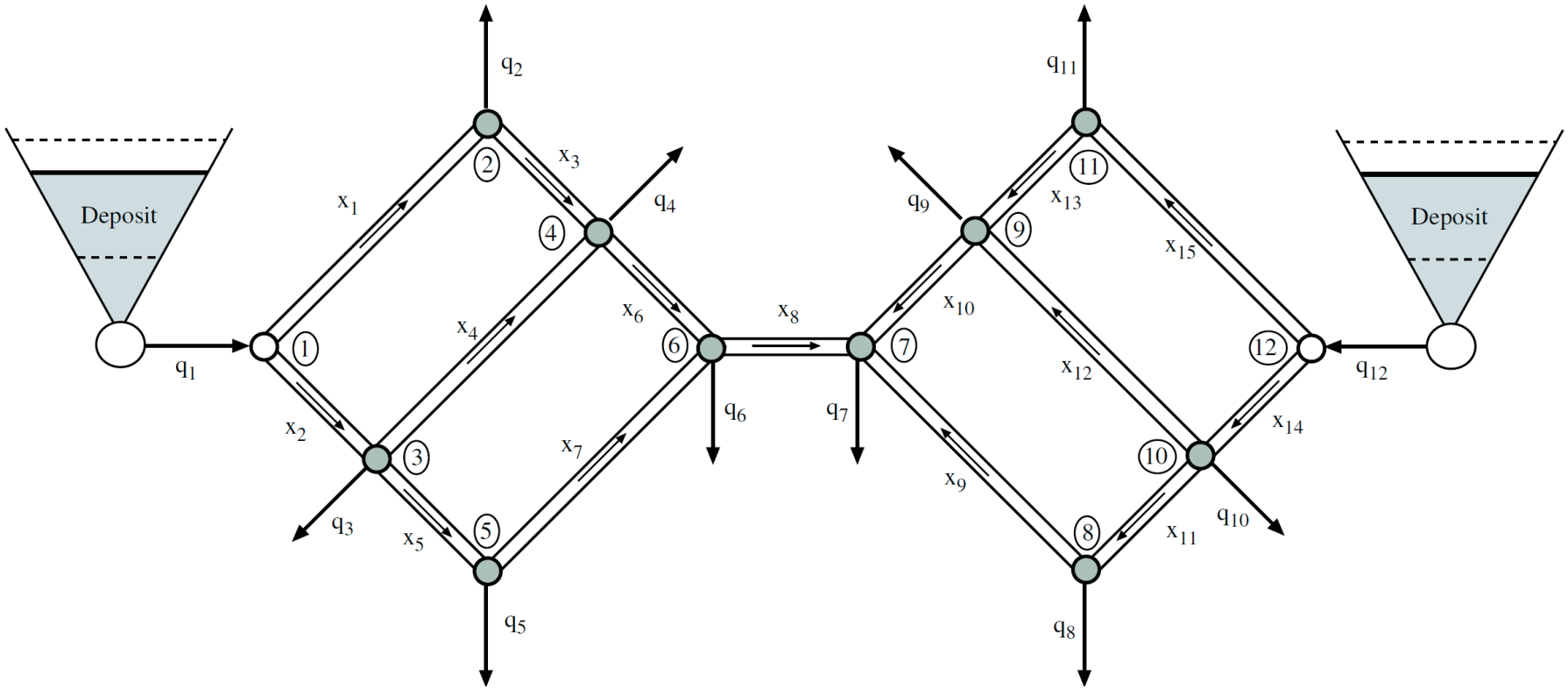
Cristina Solares

Universidad de Castilla-La Mancha

7 de septiembre de 2022

The Water Supply System.

The water supply system of a given city showing the unknowns, i.e. the flows associated with all pipes, and the node and flow numberings.



E. Castillo,R. E. Pruneda,C. Solares,R. Mínguez. *Interpreting linear systems of equalities and inequalities. Application to the water supply problem*. Numerical Lineal Algebra with Applications. Vol. 13 (5), pp. 361-397, 2005.

The Water Supply System.

Consider the water supply system of a given city, represented in above figure, that consists of the following elements:

- Pipes: They represent the paths to be followed by the water.
- Nodes: They are the points where the pipes intersect, and where water enters or leaves the flow network. We assume that there are two supply nodes, those coinciding with the two deposits, and the remaining nodes are consumption nodes.
- Data: The data of our problem are the amounts of flow that enter or leave each node, indicated by the arrows, and the topology of the network, depicted in previous figure.
- Unknowns: The unknowns in the water supply problem are the water flows in each of the pipes, that is, the number of unknowns coincides with the number of pipes.
- Equations: To derive the system of equations that model this problem we must establish the fluid balance at each node, i.e. assuming that there are no losses, the income flow + the output flow must be null at each node. This reveals that we have a system of equations with as many equations as nodes.

Establishing the node balance equations, in matrix form, we get:

$$\begin{pmatrix}
-1 & -1 & & & & & & & & & & & & & \\
1 & & -1 & & & & & & & & & & & & \\
 & 1 & & -1 & -1 & & & & & & & & & & \\
 & & 1 & 1 & & -1 & & & & & & & & & \\
 & & & & 1 & & -1 & & & & & & & & \\
 & & & & & 1 & 1 & -1 & & & & & & & \\
 & & & & & & & 1 & 1 & 1 & & & & & \\
 & & & & & & & & -1 & & 1 & & & & \\
 & & & & & & & & & -1 & & 1 & 1 & & \\
 & & & & & & & & & & -1 & -1 & & 1 & \\
 & & & & & & & & & & & & -1 & & 1 \\
 & & & & & & & & & & & & & -1 & -1
\end{pmatrix}
\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{15} \end{pmatrix}
=
\begin{pmatrix} -q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \\ q_8 \\ q_9 \\ q_{10} \\ q_{11} \\ -q_{12} \end{pmatrix}$$

which is a convenient banded matrix, the columns of which contain a one and a minus one, because each pipe has associated flow entering one node and leaving another node.

Creando una Matriz Numérica. Casos Particulares.

A continuación se muestra cómo **definir una matriz numérica** de dimensión 3×4 con MATLAB. Entre corchetes se incluyen las filas de la matriz que se pueden separar con ; o *enter*. Los elementos de cada fila se separan por uno (o más) espacios en blanco.

```
>> A = [1 2 3 4; 1 2 2 1; 1 0 0 5]
A =
     1     2     3     4
     1     2     2     1
     1     0     0     5
```

El número de filas(`m`) y columnas (`n`) se puede obtener con el comando `size`

```
>> [m n]=size(A)
m =
     3
n =
     4
```

El comando `length` devuelve el mayor valor entre `m` y `n`. En el caso de un vector devuelve el número de elementos de dicho vector.
A continuación se muestran **casos particulares de matrices** y su definición con MATLAB. La matriz cuyos elementos son todos unos se construye con el comando `ones`

```
>> A=ones(3)
A =
     1     1     1
     1     1     1
     1     1     1
```

```
>> ones(2,3)
ans =
     1     1     1
     1     1     1
```

Creando una Matriz Numérica. Casos Particulares.

La matriz cuadrada cuyos elementos son todos ceros se construye con el comando `zeros`

```
>> A=zeros(2)
A =
     0     0
     0     0
```

```
>> zeros(2,3)
ans =
     0     0     0
     0     0     0
```

La matriz I cuyos elementos son todos ceros salvo los de la diagonal principal que son unos se puede construir con el comando `eye`

```
>> I=eye(4)
I =
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

En MATLAB se pueden crear matrices de números aleatorios utilizando el comando `rand`

```
>> A=rand(2,3) %valores entre 0 y 1
A =
    0.9501    0.6068    0.8913
    0.2311    0.4860    0.7621
>> B=10*rand(2,2) %valores entre 0 y 10
B =
    8.1472    1.2699
    9.0579    9.1338
```

El comando `rng(seed)` especifica la semilla para el generador de números aleatorios.

Accediendo a los Elementos de una Matriz

A continuación se muestra cómo **acceder a los elementos**, filas y columnas de una matriz, con MATLAB. Se utiliza : para acceder a todas las filas (o columnas) de la matriz.

```
>> A = [1 2 3 4; 1 2 2 1; 1 0 0 5]
A =
     1     2     3     4
     1     2     2     1
     1     0     0     5
>> A(2,3) % elemento de la segunda fila y tercera columnna en A
ans =
     2

>> A(3,:) % tercera fila de A
ans =
     1     0     0     5
>> A(:,2) % segunda columna de A
ans =
     2
     2
     0
```

Accediendo a los Elementos de una Matriz. Submatriz de una Matriz

A continuación se muestra cómo **acceder a una submatriz** de una matriz dada

```
>> A([2,3],:) %submatriz de A formada por las filas 2 y 3
ans =
     1     2     2     1
     1     0     0     5

>> A(:,[2,3,4]) %submatriz de A formada por las columnas 2, 3 y 4
ans =
     2     3     4
     2     2     1
     0     0     5
```

Submatriz de una Matriz

```
>>  A([1,2],[1,2]) %submatriz de A formada por filas y columnas 1 y 2
ans =
     1     2
     1     2

>> A(2:3,2:4) %submatriz de A formada por filas (2 a 3) y columnas (2 a 4)
ans =
     2     2     1
     0     0     5
```

Para indicar la última fila o columna de una matriz podemos utilizar `end`

```
>> A(2:end,2:end)
ans =
     2     2     1
     0     0     5
```

Se puede crear una matriz de números enteros

```
>> B=[1:3;4:6;7:9]

B =

     1     2     3
     4     5     6
     7     8     9
```

Creando un Vector Numérico

Las matrices formadas por una sola fila o columna se denominan **vectores fila y columna**, respectivamente. Las filas y columnas de una matriz se pueden completar utilizando vectores fila y columna

```
>> b=[1;1;0] %vector columna
b =
     1
     1
     0
>> b(3) %tercer elemento del vector
ans =
     0
>> [A b] % completamos columnas
ans =
     1     2     3     4     1
     1     2     2     1     1
     1     0     0     5     0
>> b=[1,0,2,1] %vector fila
b =
     1     0     2     1
>> b(3) %tercer elemento del vector
ans =
     2
```

```
>> A=[A;b] % completamos filas
ans =
     1     2     3     4
     1     2     2     1
     1     0     0     5
     1     0     2     1
```

Una vez que se ha creado un vector o matriz, se pueden modificar sus elementos

```
>> A(1,1)=22;
>> A
A =
    22     2     3     4
     1     2     2     1
     1     0     0     5
     1     0     2     1
```

Es posible crear un vector de números enteros como sigue

```
>> b=1:10 %  valores enteros entre 1 y 10
b =
     1     2     3     4     5     6     7     8     9    10
>> b=1:2:10 % valores enteros entre 1 y 10 con un paso de 2 unidades
b =
     1     3     5     7     9
```

La función `linspace` crea un vector linealmente espaciado, utilice `help linspace` para ver como funciona.

Combinar Matrices

Se pueden combinar matrices **por filas o por columnas**. La matriz M es combinación de tres matrices 3×3 por columnas. Para hacer esta combinación, las matrices deben tener el mismo número de filas. La matriz N es combinación de tres matrices 3×3 por filas. Para hacer esta combinación, las matrices deben tener el mismo número de columnas.

```
>> A1=[1 1 1;2 2 2]; A2=[3 3 3;4 4 4]; A3=[5 5 5;6 6 6];
>> M=[A1, A2, A3]

M =

     1     1     1     3     3     3     5     5     5
     2     2     2     4     4     4     6     6     6

>> N=[A1; A2; A3]

N =

     1     1     1
     2     2     2
     3     3     3
     4     4     4
     5     5     5
     6     6     6
```

Para obtener la **diagonal** de una matriz A se puede utilizar el comando de MATLAB `diag`.

```
>> A=[1 1 3;1 0 2;2 0 5]; diag(A)
ans =
     1
     0
     5
```

El comando `diag` se puede utilizar para construir matrices diagonales. A continuación construimos una **matriz diagonal con valores numéricos y simbólicos**

```
>> syms a b
>> diag([1 a b])
ans =
[ 1, 0, 0]
[ 0, a, 0]
[ 0, 0, b]
```

Para obtener el **rango, determinante o inversa** de una función podemos utilizar los comandos `rank`, `det` o `inv`. Los **elementos de una matriz pueden ser numéricos y/o simbólicos**

```
>> rank(A)
ans =
     3
>>syms a b; B=[a 1 3;1 0 2;2 0 b]; det(B)
ans =
4 - b
```

```
>> inv(A)
ans =
     0     5    -2
     1     1    -1
     0    -2     1
```

Para calcular la **traspuesta** de una matriz se utiliza el operador ′ o la función de MATLAB `transpose`.

```
>> A = [1 2 3 4; 1 2 2 1; 1 0 0 5]; A'
ans =
     1     1     1
     2     2     0
     3     2     0
     4     1     5
```

Dado un vector (o una matriz) numérico o simbólico, podemos **aplicar a todos sus elementos funciones matemáticas** como `sin`,

```
>> u=[sym(pi) 2*pi pi/2]; sin(u)
ans =
[ 0, 0, 1]
```

Si queremos elevar todos los elementos de un vector (o una matriz) a una cierta potencia debemos utilizar el comando `.^`

```
>> v=[2 3 4]; v.^2
ans =
     4     9    16
```

Operaciones con Matrices y Vectores

Los operadores +, - y * también se pueden utilizar con matrices. Para **sumar y restar matrices** de la misma dimensión se utilizan los operadores $+$ y $-$

```
>> A = [1 2 3 4; 1 2 2 1; 1 0 0 5];
>> B = [1 0 1/5 1; 1 1 0 1; 1 5 2 0];
>> syms a; C = [1 0 1/5 1; 1 1 0 1; 1 5 2 a];
>> A+B
ans =
    2.0000    2.0000    3.2000    5.0000
    2.0000    3.0000    2.0000    2.0000
    2.0000    5.0000    2.0000    5.0000

>> A-C
ans =
[ 0,  2, 14/5,     3]
[ 0,  1,    2,     0]
[ 0, -5,   -2, 5 - a]
```

Para **multiplicar dos matrices** con MATLAB se utiliza el operador $*$

```
>> A = [1 2 3 4; 1 2 2 1; 1 0 0 5];
B = [1 0 3 1; 1 1 0 1; 1 5 2 0; 1 1 1 0];
>> C=A*B % producto de las matrices A y B
C =
    10    21    13     3
     6    13     8     3
     6     5     8     1
```

A continuación se muestra cómo **multiplicar un escalar por una matriz**, para ello se utiliza el operador $*$

```
>> A = [1 2 3 4; 1 2 2 1; 1 0 0 5];
>> C=2*A % producto de 2 por la matriz A
C =
     2     4     6     8
     2     4     4     2
     2     0     0    10
```

Para **multiplicar una matriz por un vector** con MATLAB se utiliza el operador $*$. Si el vector es un vector fila, debe ser traspuesto antes de realizar el producto

```
>> A = [1 2 3 4; 1 2 2 1; 1 0 0 5];
>> b = [1 0 3 1];
>> A*b'
ans =
    14
     8
     6
```

Si se quieren **multiplicar dos matrices (o dos vectores) elemento a elemento**, se puede utilizar $.*$

```
>> u=[1 2 3]; v=[2 2 2];u.*v
ans =
     2     4     6
```

```
>> A=[2 2;2 2];A.*A
ans =
     4     4
     4     4
```

Para calcular una **potencia entera de una matriz cuadrada** se puede utilizar el operador $*$ o el operador $\wedge$

```
>> A=[1 2; 3 0];
>> A*A
ans =
     7     2
     3     6
>> A^2
ans =
     7     2
     3     6
```

Para calcular una potencia entera, elemento a elemento, se puede utilizar $.\wedge$

```
>> A.^2

ans =

     1     4
     9     0
```

Operaciones con Matrices y Vectores

Para **sumar y restar vectores de la misma dimensión** se utilizan los operadores $+$ y $-$

```
>> u=[1 2 3];v=[4 5 6]; u+v
ans =
     5     7     9
>> u-v
ans =
    -3    -3    -3
```

Para **multiplicar un escalar por un vector** se utiliza el operador $*$

```
>> 2*u
ans =
     2     4     6
```

Para calcular el **producto escalar y vectorial de dos vectores**, MATLAB dispone de los comandos `dot` y `cross`

```
>> dot(u,v)
ans =
    32
>> cross(u,v)
ans =
    -3     6    -3
```

El producto escalar también se puede realizar haciendo

```
>> u*v'
ans =
    32
```

División de matrices

Hay dos tipos de división de matrices

- ./ over
- .\ under

```
>> X=[1 4; 5 7; 2 2];
>> Y=[1 1; 2 6; 3 4];
>> X./Y

ans =

   1.000000000000000   4.000000000000000
   2.500000000000000   1.166666666666667
   0.666666666666667   0.500000000000000

>> X.\Y

ans =

   1.000000000000000   0.250000000000000
   0.400000000000000   0.857142857142857
   1.500000000000000   2.000000000000000
```