Degree in Mechanical Engineering - Lab exercises

Prof. S. De Marchi

`Padova, 23 May 2017`

We start by introducing some useful matrices, commands and functions

1 Special matrices

```
A = zeros(2,3);
```

is a matrix 2×3 of all zeros.

```
A = eye(5);
```

is the identity matrix of order 5.

Suppose `A=[1 2 -2 4; 0 -1 6 4; 0.5 5 5 3; 0 0 3 10];`

```
d = diag(A);
```

gives the column vector of the diagonal elements, while

```
U = triu(A); L = tril(A);
```

gives the upper triangular part and the lower triangular part of `A`, respectively

2 Operation on rows or columns of matrices

- Given a matrix A of order n, the Matlab lines

  ```
  for i = 1:n
    A(i,j) = A(i,j)+1;
  end
  ```

  can be substituted by

  ```
  A(1:n,j) = A(1:n,j)+1;
  ```

  or, by using the operator **:**

  ```
  A(:,j) = A(:,j)+1;
  ```

- It is possible to **exchange rows or columns**. For example, by typing the line

```
A = B([1 3 2],:);
```

we create a matrix A having as the first row the first row of B, the second row is the third row of B and the third row is the second row of B. Similarly,

```
A = B(:,[1:3 5:6]);
```

create a matrix A whose columns are the first 3 columns of B then the fifth and the six of B.

- It is possible to **concatenate matrices**. For example,

  ```
  U = [A b];
  ```

  create a matrix U which is the concatenation of A and the column vector b (dimensions of A and b must be compatible).

- It is possible to **assign the same value to a submatrix**. For example,

  ```
  A(1:3,5:7) = 0;
  ```

  set to zero the submatrix formed by the first 3 rows and the columns from 5 to 7 of the matrix `A`.

- Another useful command for matrix manipulation is `max`. For example, `max` in the form

  ```
  [M, i] = max(A(2:7,j));
  ```

  returns the biggest element M of the j-th column of A (among the second and the seventh rows) and the position `i` of such element in the vector $[a_{2,j}, a_{3,j}, \ldots, a_{7,j}]^{\mathrm{T}}$.

Conclusion. All vectorial instructions that substitute `for` loops, should be preferable for the sake of Matlab efficiency!

3 Matrices with special structure

- *Toeplitz* matrix. `toeplitz`

$$\begin{bmatrix} c_1 & r_2 & r_3 & \ldots & r_n \\ c_2 & c_1 & r_2 & \ddots & r_{n-1} \\ \vdots & c_2 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & r_2 \\ c_n & c_{n-1} & \ldots & c_2 & c_1 \end{bmatrix}$$

This matrix can be defined by the command `toeplitz`. As an example

```
>> toeplitz([0,1,2,3],[0,-1,-2,-3])

ans =

     0    -1    -2    -3
     1     0    -1    -2
     2     1     0    -1
     3     2     1     0
```

- *Hankel* matrix. `hankel`

$$\begin{bmatrix} c_1 & c_2 & c_3 & \dots & c_n \\ c_2 & c_3 & c_4 & \iddots & r_2 \\ \vdots & c_4 & \iddots & \iddots & \vdots \\ \vdots & \vdots & \iddots & \iddots & r_{n-1} \\ r_1 & r_2 & \dots & r_{n-1} & r_n \end{bmatrix}$$

  This matrix can be defined by the command `hankel(c,r)`, where `c,r` are vectors of length n. Hankel matrices are symmetric, constant across the anti-diagonals, and have elements `H(i,j) = p(i+j-1)` where `p = [c r(2:end)]` completely determines the Hankel matrix. As an example

```
>> hankel([1,1/2,1/3,1/4],[1/4,1/5,1/6,1/7])

ans =

    1.0000    0.5000    0.3333    0.2500
    0.5000    0.3333    0.2500    0.2000
    0.3333    0.2500    0.2000    0.1667
    0.2500    0.2000    0.1667    0.1429
```

  which corresponds to the *Hilbert matrix* of order 4, `hilb(4)`. `hilb`

- *Vandermonde* matrix. `vander`

$$\begin{bmatrix} c_1^{n-1} & \dots & c_1^2 & c_1 & 1 \\ c_2^{n-1} & \dots & c_2^2 & c_2 & 1 \\ \vdots & & \ddots & \ddots & \vdots \\ c_n^{n-1} & \dots & c_n^2 & c_n & 1 \end{bmatrix}$$

```
>> vander([1 2 3 4])

ans =

     1     1     1     1
     8     4     2     1
    27     9     3     1
    64    16     4     1
```

- More special matrices can be found in *The Matrix Computation Toolbox* `http://www.maths.manchester.ac.uk/~higham/mctoolbox/`

4 Fundamental functions for matrix analysis

`det, eig, eye, diag, triu, tril, inv, norm, cond, spy, ,..`

5 The command `find`

One of the most useful command in Matlab is `find`. `find`

If we want to know which components of the vector `v=10:1:19` are ≥ 15, it suffices to type

```
>> find(v>=15)

ans =

     6     7     8     9    10
```

Now, we can do operations only on the specified elements

```
>> index=find(v>=15)

index =

     6     7     8     9    10

>> v(index)

ans =
```

```
    15    16    17    18    19

>> v(index)=v(index)-15

v =

    10    11    12    13    14     0     1     2     3     4
```

In the matrix case

```
>> A=[10,11;12,13]

A =

    10    11
    12    13

>> index=find(A<13)

index =

     1
     2
     3
```

The result of the command **find** is not a matrix, as one could expect, instead it is column vector. No problem arises in doing operations with the specified elements

```
>> A(index)=0

A =

     0     0
     0    13
```

For example, if we want to construct a matrix having elements corresponding only to the specified positions, it is necessary FIRST to initialize it with the proper dimensions

```
>> B=zeros(2)

B =

     0     0
     0     0
```

then to assign the values

```
>> B(index)=1

B =

     1     1
     1     0
```

where `index` was containing the values 1,2,3, as above.

5.1 Commands "norm" and "cond"

These two commands allow to compute the *norm* of an array and the *condition number*, κ, of a matrix. In particular the condition number is defined as

$$k(A) := \|A^{-1}\|\|A\|\,,$$

for any natural norm of A (1, 2, p, inf).

◇◇

6 Proposed excercises

1. Take the `Hilbert matrix` of order n, in Matlab is `H=hilb(n)`, with n chosen by the user. The matrix is symmetric.

   Compute its determinant for $n = 1 : 15$ and store the values in a vector `d` and also the corresponding eigenvalues by using the command `ee=eig(H)`. Plot in semilogarithmic scale the vector `d` and for all n the error norms $\|\texttt{det}(\texttt{H}) - \texttt{prod}(\texttt{ee})\|_2$ and $\|\texttt{trace}(\texttt{H}) - \texttt{sum}(\texttt{ee})\|_2$.

   What do you see?

2. Consider the vector `v=[4 1 zeros(1,n)]` and the matrix `A = toeplitz(v)`. Use the command `find` to see how many elements are different from zeros with $n = 1 : 10$. Can you derive a formula for the non-zero element of such a matrix? Using the command `nnz` we get the number of nonzero elements and then we can find the percentage of nonzero elements w.r.t. the $(n + 2)^2$ elements of the matrix.

   For $n = 10$, by using the command `spy`, plot the patterns of these elements. Can the matrix be considered *sparse*?

3. For $n = 1 : 50$, take the vectors

    - `x1=0:n;`
    - `x2=0:1/n:1;`
    - `x3=-0.5:1/n:0.5`

    Make a comparative plot, using the command `semilogy`, of the behavior of the condition numbers of the Vandermonde matrices based on the vectors `x1`, `x2` and `x3`, respectively. What do you see?

Time: **2 hours**.