Lecture 28 (hrs. 56,57) - November 27, 2025, 8:30 - 10:30 F3

(3.31) Scilab.

Scilab's built-in function *pinv* returns the pseudoinverse of a matrix. For example (see Example (3.27) in Lecture 27):

```
--> A = [1,1;1,1;1,1]

 A = [3x2 double]

   1.   1.
   1.   1.
   1.   1.

--> pinv(A)

 ans = [2x3 double]

   0.1666667   0.1666667   0.1666667
   0.1666667   0.1666667   0.1666667
```

The built-in function *backslash* (\) is used to solve a system of linear equations. Specifically, if $A \in R^{r\times c}$ is a matrix and $b \in R^{r}$ is a column, after the assignment:

$$x = A\backslash b$$

we have:[1]

- if $r = c$ and $c_1(A) \leqslant \dfrac{1}{10\,u}$

  then:

  x is an approximation of the solution of the system $A\,x = b$ computed with a procedure equivalent to the application of the EGPP, SA, SI procedures;

- if ($r = c$ and $c_1(A) > \dfrac{1}{10\,u}$) or $r > c$

  then:

  x is an approximation of an element of $S_{MQ}(A,b)$ - usually *not* the minimum norm one - computed with a procedure that uses a QR factorization of A.

For example (see Example (3.25) of Lecture 27):

```
--> A = [1,1;1,1]

 A = [2x2 double]

   1.   1.
   1.   1.
```

1 Let N be a norm in R^{n}. In *Scilab*, when $A \in R^{n\times n}$ is a *non-invertible* matrix, we set: $c_N(A) = +\infty$.

```
--> b = [1;0]

 b = [2x1 double]

   1.
   0.

--> x = A\b

 x = [2x1 double]

   0.5000000
   0.

--> y = pinv(A) * b

 y = [2x1 double]

   0.2500000
   0.2500000
```

The built-in function *qr* returns an approximation to a QR factorization of a matrix, even if it is not square. Specifically, if $A \in R^{r\times c}$ where $r > c$, after the assignment:

$$[Q,R] = qr(A)$$

the matrix $Q \in R^{r\times r}$ is an approximation of the orthogonal matrix calculated with the Householder method (Remark (2.21) of Lecture 17) applied to A and $R \in R^{r\times c}$ is a matrix all whose elements under the main diagonal are equal to zero (i.e it is an upper triangular matrix). For example:

```
--> A = [1,0;1,1;1,1]

 A = [3x2 double]

   1.   0.
   1.   1.
   1.   1.

--> [Q,R] = qr(A)

 Q = [3x3 double]

  -0.5773503   0.8164966  -8.756D-17
  -0.5773503  -0.4082483  -0.7071068
  -0.5773503  -0.4082483   0.7071068

 R = [3x2 double]

  -1.7320508  -1.1547005
   0.         -0.8164966
   0.          0.
```

To obtain an approximation of a QR factorization of A as defined in Definition (3.28) of Lecture 27 one can use the qr function as follows:

```
--> [U,T] = qr(A,'e')

 U = [3x2 double]

  -0.5773503   0.8164966
  -0.5773503  -0.4082483
  -0.5773503  -0.4082483

 T = [2x2 double]

  -1.7320508  -1.1547005
   0.         -0.8164966
```

The factors U,T are obtained from the factors Q,R by eliminating, respectively, the third column of Q and the third row of R. In fact, if we perform the product Q R column by column, we observe that, if q_1,q_2,q_3 are the columns of Q and r_{ij} are the elements of R, we have:

$$Q\,R = (r_{11}\,q_1 + 0\,q_2 + 0\,q_3 \;,\; r_{12}\,q_1 + r_{22}\,q_2 + 0\,q_3) = U\,T$$

(4) NUMERICAL METHODS FOR ORDINARY DIFFERENTIAL EQUATIONS

(4.01) <u>Example</u> (damped harmonic oscillator).

The motions of a damped harmonic oscillator are described by the *differential equation*:

(E) $$x''(t) + a\,x'(t) + b\,x(t) = 0$$

where the unknown is the real-valued *function* x(t). This is a *second-order* differential equation (linear, with constant coefficients, homogeneous). A *solution* to the equation is an *at least two-times differentiable* real-valued function y(t) that satisfies the equality $y''(t) + a\,y'(t) + b\,y(t) = 0$ for *all* t in R. The differential equation determines *all* possible motions of the oscillator (equation (E) has *infinitely many* solutions). Each of the motions is identified by the *initial conditions*:

(CI) $$x(t_0) = x_0 \quad , \quad x'(t_0) = v_0$$

The *Cauchy Problem* is the problem to *find the solutions to the differential equation that satisfy the initial conditions.*

The second-order differential equation (E) is *equivalent* to a *system of* two *first-order equations*. Equivalence in this case means that: if y(t) is a solution to equation (E), then, set:

$$u_1(t) = y(t) \quad , \quad u_2(t) = y'(t)$$

Then:

$$u_1'(t) = u_2(t) \quad , \quad u_2'(t) = - a\, u_2(t) - b\, u_1(t)$$

Hence the column $(u_1(t),\ u_2(t))^t$ is a solution of the system:

(S) $$x_1'(t) = x_2(t) \quad , \quad x_2'(t) = - a\, x_2(t) - b\, x_1(t)$$

On the contrary: let $(y_1(t),\ y_2(t))^t$ be a solution of system (S) and set $y(t) = y_1(t)$. We have: $y'(t) = y_1'(t) = y_2(t)$ and $y''(t) = y_1''(t) = y_2'(t) = - a\, y_2(t) - b\, y_1(t)$ that is:

$$y''(t) + a\, y'(t) + b\, y(t) = 0$$

hence $y(t)$ is a solution of equation (E). Moreover, $y(t)$ is a solution of the Cauchy Problem:

$$x''(t) + a\, x'(t) + b\, x(t) = 0 \quad ; \quad x(t_0) = x_0 \quad , \quad x'(t_0) = v_0$$

if and only if $(y(t),\ y'(t))^t$ is a solution of the Cauchy Problem:

$$x_1'(t) = x_2(t) \quad , \quad x_2'(t) = - a\, x_2(t) - b\, x_1(t) \quad ; \quad x_1(t_0) = x_0 \quad , \quad x_2(t_0) = v_0$$

(4.02) Remark.

The procedures we will describe are designed to approximate the solution of the Cauchy Problem:

(§) $$x'(t) = F(\ t,x(t)\ ) \quad , \quad x(t_0) = x_0$$

for t in a *bounded* interval $[t_0,t_f]$. The *unknown* of the problem is the function $x(t)$ with values in R^n; the *data* are: *the function* F defined in $R \times R^n$ with values in R^n, *the instants* t_0 and $t_f > t_0$ and *the column* x_0 in R^n.

The previous statement presupposes that the solution to problem (§) *exists and is unique*. A further hypothesis will also be necessary to *describe the numerical procedures*.

(4.03) Hypothesis (existence and uniqueness).

For every $\underline{t}$ in R and $\underline{x}$ in R^n there is only one solution to the differential equation:

$$x'(t) = F(\ t,x(t)\ )$$

which verifies the initial condition:

$$x(\underline{t}) = \underline{x}$$

We will denote such a solution by: $y(t;\ \underline{x},\underline{t})$.

(4.04) Definition (numerical method).

A *numerical method for approximating the solution of the Cauchy Problem* (§) *on* $[t_0,t_f]$ is a procedure that constructs, based on the value of a user-controlled parameter E, real numbers t(0) = t0,...,t(N) in $[t_0,t_f]$, columns x(0) = x0,...,x(N) in R^n and, for k = 0,...,N, suggests to use x(k) as an approximation of y(t(k); x0,t0).

The numbers t(0),...,t(N) are called *integration instants* and, for k = 0,...,N-1, the number h(k) = t(k+1) - t(k) is called the *integration step at instant* t(k).

A *Scilab* implementation of a numerical method has the following structure:

```
function [T,X] = NumericalMethod(x0,t0,tf,F,E)

k = 0; t(0) = t0; x(0) = x0;
while t(k) < tf,
   CHOOSE h(k) based on the value of E;
   COMPUTE x(k+1);
   t(k+1) = t(k) + h(k);
   k = k+1;
end;

endfunction
```

The output variables are, respectively, the row T and the matrix X such that:

T = (t(0),...,t(N))   ,   X = (x(0),...,x(N))

A numerical method is specified by the procedures of *choosing* h(k) and *computing* x(k+1).