Université de Picardie Jules Verne
Cours de l'Ecole Doctorale/Doctoral School's Lectures
Calculs Numériques/Numerical Compuations
2019-2020

Plotting with Scilab

- Simple plot
```
x = linspace(-% pi,% pi,11); // mesh of [-pi,pi] using 11 pts.
y = cos(x)./(1+x.∧ 2); // (% pi is a scilab constant)
clf() // clear the (current) graphic window
plot(x,y,’b’)
```
  We can modify the script by playing with the interval length and/or the discretisation parameter and/or the color of the plot line (’b’, ’g’, ’r’, ’k’, ’c’, ’m’ stand respectively for blue, green, red, black, cyan, magenta).

- More generally, The plot function can be used to draw one or several curves:
```
plot(x1,y1[,style1],x2,y2[,style2], ....)
```
  with style is an optional string to define color, line type, or symbol. strings in scilab are delimited by simple or double quote.
  Example
```
x = linspace(0,2% pi,31);
y1 = sin(x); y2 = cos(x);
scf(0); // select graphic window 0 to be the default graphic window
clf(); // clear the graphic window
plot(x,y1,"b-",x,y2,"r--"); // only lines
scf(1); // select graphic window 1 to be the default graphic window
clf(); // clear the graphic window
subplot(2,1,1); // split the graphic window and use subpart 1
plot(x,y1,"ro",x,y2,"bx"); // only symbols
subplot(2,1,2); // split the graphic window and use subpart 2
plot(x,y1,"r--o",x,y2,"g-x"); // both lines and symbols
```

- Plot3d plot3d(x,y,z,<optargs>)
  <u>Example</u> simple plot using z=f(x,y)
```
t=[0:0.3:2*% pi]’;
z=sin(t)*cos(t’);
plot3d(t,t,z)
```

	colors			line types		symbols			
k	black	c	cyan	-	solid	+	$+$	d	$\diamondsuit$
b	blue	m	magenta	--	dashed	x	$\times$	v	
r	red	y	yellow	:	dotted	o	$\bigcirc$	s	
g	green	w	white	-.	dashdot	*	$\ast$	*wedge*	$\triangle$

Matrices and arrays with Scilab

We can first define matrices and vectors from the coefficients as:

- ```
  A = [1, 2, 3; 4, 5, 6] // the character ; introduce the next row
  x = [0 ; 1; 0] // a column vector
  y = [exp(% pi), sin(% e)] // a row vector
  ```

We can aslo define matrices from other matrices

- ```
  B = [A ; y , 1]
  C = [B, x] // C can be built directly using [[A ; y, 1],x]
  ```

Some matrix/vector constructors and operators

- `linspace(a,b,n)` creates a uniform mesh of [a; b] with n points
  `zeros(m,n)` and `ones(m,n)` build `m`× `n` matrices of 0 and 1. `eye(m,n)` builds the `m`× `n` identity like matrix.
  If x is a vector `A=diag(x,k)` builds a diagonal like matrix by filling the k-th diagonal with the vector x.

Basic operations and transforms on matrices

- if A is a matrix but not a vector `diag(A,k)` extracts the diagonal number k as a column vector.
- `A*B` is the matrix product of the matrices A and B (or product between a scalar and a vector or matrix).
- `A'` performs the transposition of matrix A.
- the scalar product is `x'*y`, where `x` and `y` are columnvectors
- If A is a square invertible matrix you can solve the linear system `Ax = b` using `x = A\b` (a PA = LU factorization of the matrix, followed by an estimation of its condition number, and

  finally by solving the 2 triangular systems, are done in a transparent manner).
- Pointwize product `z=x.*y` returns the vector `z` of components $z_i = x_i * y_i$
  More generally one can consider `z=x./y` with $z_i = x_i/y_i$; `z=x.∧p` returns $z_i = x_i \wedge p$.
- The same with matrices of same size `A.*B`

<u>Example</u> We want to solve the linear system $Ax = b$ where

$$A = \frac{1}{h^2}\begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix}.$$

where $h = \dfrac{1}{n+1}$. Here A is the discretization matrix of the negative laplacian on the unit interval associated to homogeneous Dirichlet boundary conditions. We can define the right hand side (r.h.s in short) randomly as `b=rand(n,1)` (this is a column vector).
We can solve directly the system, compute the relative residual, the value of the associated quadratic functional $J(x) = \frac{1}{2} < Ax, x > - < b, x >$ at the solution. We can also build the bloc matrix

$$\begin{pmatrix} A & Id \\ Id & A \end{pmatrix}.$$

where Id is the $n \times n$ identity matrix.

```
n = 7;
v = -ones(1,n-1);
A = diag(v,-1) + 2*eye(n,n) + diag(v,1)
// another solution
// A = diag(v,-1) + diag(2*ones(1,n)) + diag(v,1)
b = rand(n,1);
x = A\ b
res = norm(A*x-b)/norm(b)
E = 0.5*x'*A*x - b'*x
y = rand(n,1);
F = 0.5*y'*A*y - b'*y
E < F
B = [ A , eye(n,n) ;...
eye(n,n), A ]
```

coefficients handling : the first element of a vector x is `x(1)` not `x(0)`

Example

```
A = rand(3,4) // create a matrix
A(2,2) = -1 // change coef (2,2) of A
c = A(2,3) // extract coef (2,3) of A and assign it to variable c
A(2,:)  // extract row 2 of A (it is assigned to ans)
A(2,:)  = ones(1,4) // change row 2 of A
A(:;3) = 0 // change column 3 of A
B = A([1,3],[1 2]) // extract submat (1,3)x(1,2) and assign it to B
A([1,3],[1 2]) = [-10,-20;-30,-40] // change the same sub-matrix
```

Here more complete assign/extraction syntaxes:

```
A(rowind,colind)= RHS // assign
var = A(rownd,colind) // extract (and assign to var)
```