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

Linear Algebra with Scilab

1 Solving Linear systems

1.1 The model problem

Consider the problem of solving a system of m linear equations with n unknowns:

$$\begin{array}{lll} a_{11}x_1 + a_{12}x_2 + \cdots & +a_{1n}x_n & = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots & +a_{2n}x_n & = b_2 \end{array}$$

$$a_{m1}x_1 + a_{m2}x_2 + \cdots \quad +a_{mn}x_n \quad = b_m$$

or, in matrix form,

$$Ax = b$$

where A is a $m \times n$ matrix with entries $a_{i,j}$, b is a given m-vector, and x is the n-vector to be determined. A special attention is given for m = n, that is when the number of equations coincides with the number of unknowns. In this case the system of equations has a unique solution x provided the matrix A is invertible, say $det(A) \neq 0$. We will assume that we deal with this situation in the sequel.

1.2 Solving Linear Equations with Scilab

When $m = n$ the algorithm used by Scilab and other software for solving linear equations is based on Gaussian elimination. Linear equations are solved in Scilab using the backslash, \, operator. If A is a matrix and b is a vector, then
`x = A \ b`
is the solution of the linear system $Ax = b$.

Example
Here is a simple example with a 3×3 random system: `A=rand(3,3)` Test is A invertible by computing `det(A)`. Then generate a random 3-vector b and solve $Ax = b$. Check the solution by computing $Ax - b$. Conclusion ?

The computations are done quickly, the inner codes are efficiently implemented. To haev an idea on how the time computation depends on the size of the matrices we can proceed as follows:

First we reserve an amount of memory with `stacksize(10000000)`. then, using a loop on the dimension we can

```
for n=1:100:1600
A=rand(n,n);
b=rand(n,1);
tic
A\b
toc
end
```

IMPORTANT !! In practice we never compute A^{-1} for solving teh linear system $Ax = b$. This should be too much expensive (complexity in $\mathcal{O}(n^3)$ vs $\mathcal{O}(n^2)$ for Gaussian elimination).

1.3 Vector norms

Use command `norm`

2 Eigenvalues and eigenvectors

Let A be a $n \times n$ matrix, a vector x is an eigenvector of A with eigenvalue λ when

$$Ax = \lambda x$$

The set of all eigenvalues is called the spectrum of A. To compute all teh eigenvalues, we can use

`spec(A)`
It returns the eigenvalues of A as components of a vector. Assigning a pair of values to result:

```
[evecs,evals] = spec(a)
```

Other computations can be done using the command `eigs` as follows:

- La commande `eigs` et ses options de base : les valeurs propres seules
  - `d=eigs(A)`: computes the 6 greatest (in modulus) eigenvalues of A
  - `d=eigs(A,k)` computes k greatest (in modulus) eigenvalues of A
  - `d=eigs(A,'lm')` : compute the greatest (in modulus) eigenvalue of A
  - `d=eigs(A,'sm')`: computes the smallest (in modulus) eigenvalue of A
  - `d=eigs(A,'lr')`: computes the greatest real part of the eigenvalues of A
  - `d=eigs(A,'li')` : computes the greatest imaginary part of the eigenvalues of A A
- Eigenvalues and eigenvectors
  - `[V,D]=eigs(A)`: computes the 6 greatest (in modulus) eigenvalues of A (array `D`) and teh corresponding eigenvectors (array `V`)
  - `[V,D]=eigs(A,k)`: the same with K eigen-elements