MA4011/MA7091 – Computer Class 1 – 25.01.2018

Matrices, Gaussian Elimination, Sparse matrices

We start by recalling how to introduce vectors and matrices in MATLAB. Once MATLAB is up and running, remind yourself how to introduce a 3×1-vector, a 3×3-matrix and 1×3-vector in MATLAB.

As we shall see in the next weeks, to compute an approximation to a partial differential equation, quite often we shall need to be able to solve *large* systems of linear equations. In matrix form, such a system can be written

$$Ax = b \tag{1}$$

where $A \in \mathbb{R}^{n\times n}$ is a square matrix, $x \in \mathbb{R}^n$ is the vector of unknowns, and $b \in \mathbb{R}^n$ is the right-hand side known vector. A known classical method for solving such linear systems of equations is the *Gaussian elimination*, which it is assumed we are all familiar with. If not, a revision is necessary at this point; a quick summary can be found in any book of basic linear algebra, or in the link:

`https : //en.wikipedia.org/wiki/Gaussian_elimination`

Task 1.
In MATLAB, Gaussian elimination is implemented; the command is "\". For example, to find the solution to the linear system (1) above, the command is $A\backslash b$.[1]

The following program sets up an $n \times n$-matrix A with random entries, a right-hand side n-vector b with random entries and it performs Gaussian elimination to find the solution n-vector x. The commands `tic` and `toc;` are used to find how long the solving of the linear system took (in seconds).

```
clear all;
n=64
A=rand(n,n);
b=rand(n,1);
tic;
x=A\b;
toc;
```

(Start a new M-file in MATLAB and copy and paste the code above. Run the program.) Write a loop which changes the value of n to $n = 2^k$ for $k = 7, 8, 9, 10, 11, 12, 13$ and records the times taking to solve each linear system. (You should use the help of MATLAB to understand how to change the `tic;` and `toc;` syntax appropriately to be able to save the time for each iteration of the loop.) What do you observe with the time it takes to solve the problem as a function of the size n? What happens when n becomes very large? Why?

Task 2.
Next, we shall use the command `diag` to construct another matrix A. (If you do not know what the commands in the program below do, you are advised to go to the MATLAB help and look them up, before trying this task.)

```
clear all;
n=10
e=ones(n,1);
A=diag(-2.*e,0)+diag(e(1:n-1),1)+diag(e(1:n-1),-1);
b=rand(n,1);
tic;
x=A\b;
toc;
```

(Start a new M-file in MATLAB and copy and paste the code above. Run the program.) Type `A` in the command line window to see what the matrix A looks like. This matrix has "many" entries equal to zero; in fact most entries are zero! Such matrices are called *sparse matrices*. Write a loop which changes the value of n to $n = 2^k$ for $k = 7, 8, 9, 10, 11, 12, 13$ and records the times taking to solve each linear system. What do you observe with the time it takes to solve the problem compared to the example above? For which value of k the problem becomes too big to be solved in the computer you are at? **Please turn over.**

[1] In fact, the command "\" is not just plain simple Gaussian elimination; it is a very complex algorithm that is based on Gaussian elimination, but with very many extra features that enable optimisation of various operations

Task 3.
Next, we shall use the command `spdiags` to construct the same matrix A above. This command is useful when we are working with sparse matrices, as it creates the matrix A in the so-called sparse format. That way we do not need to store all the zero entries! (If you do not know what the commands in the program below do, you are advised to go to the MATLAB help and look them up, before trying this task.)

```
clear all;
n=10
e = ones(n,1);
A = spdiags([e -2*e e], -1:1, n, n);
b=rand(n,1);
tic;
x=A\b;
toc;
```

Start a new M-file in MATLAB and copy and paste the code above. Run the program.) Type `A` in the command line window to see what the matrix A looks like. This matrix has "many" entries equal to zero; in fact most entries are zero! Such matrices are called *sparse matrices*. Write a loop which changes the value of n to $n = 2^k$ for $k = 7, 8, \ldots, 20$ and records the times taking to solve each linear system. What do you observe with the time it takes to solve the problem compared to the example above? For which value of k the problem becomes too big to be solved in the computer you are at? Why does this happen in view of the results from Task 2?