MATH 353: Engineering Mathematics III – Section 012

Spring 2013 (F.–J. Sayas) Lab # 10 April 26

Open Matlab and *move to the Desktop or to a folder where you can find your work* at the end of the session. Type these lines

```
>> diary myworkApril26
>> format long
>> format compact
```

1. When matrices are very large but contain very few non zero elements, a good option is to use a specific storage for *sparse* matrices. The information is given only for non-zero entries, by listing in what row and column the element is stored, and what element is stored in that location. For instance, to introduce the matrix

$$\begin{bmatrix} 0 & 0.1 & 0 & 0 & 0 \\ -1 & -2 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.7 & 0 & 0 \\ 4.1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

we write

```
A=sparse([1 2 4 5 2 2],...
         [2 2 3 1 1 4],...
         [0.1 -2 0.7 4.1 -1 2],...
         5,5);
```

Note that the elements have not been introduced in order, and that at the end we tell Matlab the size of the matrix. Repeat the same argument to introduce the matrix

$$\mathrm{A} = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & -2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix}$$

If you want to be sure that the matrix is the one you thought you had introduced, do `full(A)`.

2. Use the previously defined matrix to solve the linear system

$$\begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ 3 \\ 2 \\ 4 \end{bmatrix}$$

3. Learn how to use the comand `spdiags` (as in sparse–diagonal) in order to create and store matrices with very few non-zero diagonals. Once you have figured it out, create a script that constructs the $N \times N$ matrices

$$\begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix} \qquad \text{and} \qquad \begin{bmatrix} r(x_1) & & & & \\ & r(x_2) & & & \\ & & \ddots & & \\ & & & r(x_{N-1}) & \\ & & & & r(x_N) \end{bmatrix}$$

for any given N and function r. Here the points x_i are given by the formula

$$x_i = a + h\,i, \quad i = 1, \ldots, N, \qquad \text{where} \qquad h = \frac{b-a}{N+1},$$

and a, b are also given. (This means that at the beginning of your script you have to fix a, b, the function r and the value N. After that, create the two matrices.)

4. **The whole guacamole.** The final goal of this lab is to numerically solve the boundary value problem

$$\left[\begin{array}{ll} -y'' + r(x)y = f(x) & a \le x \le b, \\ y(a) = y_a, & \\ y(b) = y_b. & \end{array}\right.$$

The Finite Difference method that we explored in class consists of choosing N, defining

$$h = \frac{b-a}{N+1}, \qquad x_i = a + i\,h \quad i = 1, \ldots, N,$$

and creating and solving a system $\mathrm{A}\mathbf{y} = \mathbf{b}$, where

$$\mathrm{A} = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix} + h^2 \begin{bmatrix} r(x_1) & & & & \\ & r(x_2) & & & \\ & & \ddots & & \\ & & & r(x_{N-1}) & \\ & & & & r(x_N) \end{bmatrix}$$

and

$$\mathbf{b} = h^2 \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{N-1}) \\ f(x_N) \end{bmatrix} + \begin{bmatrix} y_a \\ 0 \\ \vdots \\ 0 \\ y_b \end{bmatrix}.$$

Once you have solved the system, you will have two vectors, $\mathbf{x}$ (with the points x_i) and $\mathbf{y}$ (with the approximate values $y_i \approx y(x_i)$). You are finally going to create an extended version of these vectors so that you have $(a, x_1, \ldots, x_N, b)$ and $(y_a, y_1, \ldots, y_N, y_b)$. (Read on in the next page.)

At the end you should have **a function** with the following prototype

```
function [y,x]=FiniteDiffBVP(r,f,interval,bvalues,N)

% [y,x]=FiniteDiffBVP(r,f,[a b],[ya yb],N)
%
% Input:
%     r,f    : vectorized functions
%     [a,b]  : 2-vector with interval
%     [ya,yb]: 2-vector with boundary values
%     N      : number of interior points in discretization grid
% Output:
%     y      : (N+2)-vector [ya, y_1,...,y_N, yb]   y_i \approx y(x_i)
%     x      : (N+2)-vector [a,  x_1,...,x_N, b]    x_i = a+ i h
```

5. To check that your code works, use the following example

$$\left[\begin{array}{ll} -y'' + (1+x^2)y = x^2\, e^x & \quad 0 \le x \le 2, \\ y(0) = 1, & \\ y(2) = e^2. & \end{array}\right.$$

The exact solution is $y(x) = e^x = \exp(x)$. Check the graphs of the exact and approximate solutions. Compute also the errors

$$E_h = \max_{0 \le i \le N+1} |y(x_i) - y_i| = \mathcal{O}(h^2).$$