MATH 5485 Introduction to Numerical Methods I

Instructor: *Alexander Shapeev* Fall 2013

Matlab Tutorial

September 6, 2013

1 Starting Matlab

In `terminal`, type:

```
ashapeev@lind40-09 (/home/ashapeev) % mkdir num_meth
ashapeev@lind40-09 (/home/ashapeev) % cd num_meth
ashapeev@lind40-09 (/home/ashapeev) % matlab
```

This creates `num_meth` directory for your files, enters this directory, and starts Matlab.

(Note: you can start it directly in a Dashboard, or whatever it is called, but you then need to change directory in Matlab.)

2 Matrices and Vectors

- Use “`[`” and “`]`” to create a matrix, “`,`” to separate entries and “`;`” to separate lines:

```
>> A = [1,0,0; 1,1,1; 1,2,3]

A =

     1     0     0
     1     1     1
     1     2     3
```

- Likewise, create a column-vector:

```
>> b = [1; 2; 4]

b =

     1
     2
     4
```

Now try the following:

- Matrix-vector multiplication `x = A*b`
- Solving linear systems `x = A\b`
- Matrix and vector transpose `A'` and `b'`
- Scalar product of x and b: `b'*x`
- Component-wise product of x and b:

```
>> b .* x

ans =

     1
     0
     4
```

- Component-wise square of b:

```
>> b.^2

ans =

     1
     4
    16
```

- Identity matrix: `eye(3)`
- Diagonal of a matrix: `diag(A)`
- Compontents a vector and a matrix:

```
>> A(3,2)

ans =

     2

>> b(3)

ans =

     4
```

- Submatrices and subvectors:

```
>> A(2:3,1:2)

ans =

     1     1
     1     2

>> b(2:3)

ans =

     2
     4
```

- Vector of integers from 1 to 5:
  `>> 1:5`

- Vector of six numbers from 2 to 3:
  `>> linspace(2,3,6)`

- Although our matrix `A` is 3×3, we can still assign, e.g., an extra line element-by-element:

  ```
  >> A(4,1)=4; A(4,2)=6; A(4,3)=9; A

  A =

       1     0     0
       1     1     1
       1     2     3
       4     6     9
  ```

- To see how to concatenate vectors and matrices, see Matlab documentation on square brackets, e.g., by typing

  ```
  >> doc paren
  ```

3 Plotting Functions

- Set of bunch of x coordinates:
  `>> x = linspace(0,2*pi,100);`

- (Note that ";" is used to suppress the output!)

- Calculate the y coordinates:
  `>> y = sin(x);`

- (Note that “`sin`” is applied component-wise to the vector `x`.)
- Plot:
  ```
  >> plot(x,y);
  ```
- Next: plot another graph on top:
  ```
  >> hold on;
  >> plot(x,cos(x),'g');
  >> plot(x, sin(x).^2, 'r--');
  >> hold off;
  ```
- (Note: you may need to manually switch to the “Figure 1” window to see the graph.)
- (Note that “`'g'`” tells Matlab to plot in green, and “`'r--'`” is “red dashed”.)
- (Recall: “`.^`” stands for “component-wise squared”.)

4 Scripts

Let us start over by cleaning up:

```
>> clear all
>> close all
```

A script is a bunch of Matlab commands saved in a “`.m`” file.

- Create a script from the top menu, by clicking “New script” (or in Windows, simply by pressing “Ctrl-N”). This opens a File Editor (before that you were working in Command Window).
- Key in the following two lines:

  ```
  x = linspace(0,2*pi,n);
  plot(x,sin(x));
  ```

  and save as “`plotsin.m`”. A .m file is a Matlab script file that you can run from the Command Window.
- To run the script simply type `plotsin`:

  ```
  >> plotsin
  Undefined function or variable 'n'.

  Error in plotsin (line 1)
  x = linspace(0,2*pi,n);
  ```
- (See that it tell you that n is undefined, with details on which line triggered the error.)
- To use our script correctly, we first should set n:

  ```
  >> n=100;
  >> plotsin;
  ```

5 Functions

Functions are ".m" files with the a special first line:

- Create the following script:

  ```
  function x0 = sqrt_approx(a, x0, Niter)
  for i=1:Niter
      x0 = 0.5*(x0 + a/x0);
  end
  ```

  and save it as "sqrt_approx.m"

- (Notice that the function does calculations in x0 and returns it.)

- Run the function:

  ```
  >> x = sqrt_approx(2,1,3)

  ans =

      1.4142

  >> x-sqrt(2)

  ans =

     2.1239e-06
  ```

- (We see that it gives a pretty good approximation to $\sqrt{2}$.)

- Notice: we used a "for" loop. You can learn more by typing

  ```
  >> doc for
  ```

  or going to the Matlab documentation from the top menu.

- You can modify the function to include the error tolerance:

  ```
  function x1 = sqrt_approx(a, x0, Niter, tol)
  for i=1:Niter
  x1 = 0.5*(x0 + a/x0);
      if(abs(x1-x0)<tol)
          return;
      end
      x0 = x1;
  end
  warning('Maximum number of iterations reached');
  ```

and play with the function:

```
>> sqrt_approx(2,1,3,1e-5)
Warning: Maximum number of iterations reached
> In sqrt_approx at 9

ans =

    1.4142

>> sqrt_approx(2,1,10,1e-5)

ans =

    1.4142
```

6 Working with files

- Download `numnum.txt` and `matr.txt` (go to `www.shapeev.com`, click on “Teaching” and see the entry for the class today). The file `numnum.txt` reads:
  `5 3`

- Try the following:

```
fid = fopen(’numnum.txt’);
x = fscanf(fid, ’%f’);
fclose(fid);
x
```

- (Don’t put “;” in the last line to see the result.) The Matlab variable `x` should be a column-vector with the two numbers.

- Try reading the numbers one-by-one:

```
fid = fopen(’numnum.txt’);
a = fscanf(fid, ’%f’, 1);
b = fscanf(fid, ’%f’, 1);
fclose(fid);
a
b
```

- (The third parameter to `fscanf` tells how many numbers to read.)

- Exercise: the file `matr.txt` has the following structure. The first line are two numbers, `n` and `m`. The following `n` lines contain `m` numbers each. Your task is to read `matr.txt` into three variables: `n`, `m`, and `A`, where `A` is the corresponding `n`×`m` matrix.