Introduction to Matlab

Praveen. C
praveen@math.tifrbng.res.in

Tata Institute of Fundamental Research
Center for Applicable Mathematics
Bangalore 560065
http://math.tifrbng.res.in/~praveen

IFCAM Summer School on Numerics and Control of PDE
22 July - 2 August, 2013
IISc, Bangalore

In the following slides, the symbol

```
>>
```

denotes the matlab command prompt.

Variables: Come into existence when you assign a value

```
>> x=1
```

To prevent the value from being printed to screen, end the line with a colon

```
>> x=1;
```

You can now use the variable `x` in other statements

```
>> y=sin(x)
```

A **row vector**

```
>> x = [1,2,3,4]
>> y=sin(x)
```

Note that Matlab computed `sin` on every element of the vector `x`

A **column vector**

```
>> x = [1; 2; 3; 4]
>> y = sin(x)
```

Output y inherits dimensions of input x

Matrix

```
>> x = [1, 2, 3, 4; 5, 6, 7, 8]
>> y=sin(x)
```

Line continuation

```
>> x = [1, 2, 3, 4; ...
        5, 6, 7, 8]
>> y=sin(x)
```

Adding vectors

```
>> x = [1, 2, 3, 4]
>> y = [5, 6, 7, 8]
>> z = x + y
```

x and y must have same dimensions. The following is wrong

```
>> x = [1, 2, 3, 4]
>> y = [5; 6; 7; 8]
>> z = x + y
```

To find dimensions

```
>> size(x)
>> size(y)
```

Transpose a vector or matrix

```
>> z = x + y'
>> size(y')
```

Find all variables

```
>> who
```

Deleting all existing variables

```
>> clear all
>> who
```

Matrix-vector multiplication

```
>> x = [1; 2]
>> A = [1, 2; 3, 4]
>> y = A*x
```

Matrix-matrix operations

```
>> B = [5, 6; 7, 8]
>> C = A + B
>> D = A*B
```

Elementwise operation

$$z = x \sin(y)$$

```
>> x = [1, 2, 3, 4]
>> y = [5, 6, 7, 8]
>> z = x .* sin(y)
```

A more complicated example

$$z = \frac{x^2 \sin(y)}{\cos(x+y)}$$

```
>> z = x.^2 .* sin(y) ./ cos(x+y)
```

Multiply matrices element-wise

```
>> E = A .* B
```

`A` and `B` must have same size

Zero vector/matrix

```
>> x = zeros(4,1)
>> A = zeros(3,3)
```

Ones vector/matrix

```
>> x = ones(4,1)
>> A = ones(3,3)
```

Identity matrix

```
>> A = eye(4)
```

Random vector/matrix

```
>> x = rand(1,3)
>> A = rand(3,2)
```

Documentation

```
>> help rand
```

Plotting

Making a uniform grid

```
>> x = linspace(0, 2*pi, 10)
>> y = sin(x)
```

Plot a line graph

```
>> plot(x, y, '-')
```

Plot a symbol graph

```
>> plot(x, y, 'o')
```

Plot a line and symbol graph

```
>> plot(x, y, 'o-')
```

Plotting

Multiple graphs

```
>> x = linspace(0, 2*pi, 100);
>> y = sin(x);
>> z = cos(x);
>> plot(x, y, 'b-', x, z, 'r--')
>> xlabel('x')
>> ylabel('y,z')
>> legend('x versus y', 'x versus z')
>> title('x versus y and z')
```

Plotting

Subplots

```
>> x = linspace(0, 2*pi, 100);
>> y = sin(x);
>> z = cos(x);
>> subplot(1,2,1)
>> plot(x, y, 'b-')
>> xlabel('x')
>> ylabel('y')
>> subplot(1,2,2)
>> plot(x, z, 'r--')
>> xlabel('x')
>> ylabel('z')
```

For more, use `help`

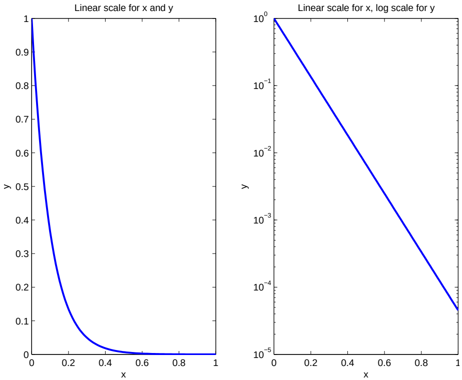
```
>> help plot
```

Logarithmic plots

Suppose we want to plot

$$y = \exp(-10x), \qquad x \in [0, 1]$$

Then y varies between 10^{-4} to 1. A normal plot does not clearly show the function, as seen in the left figure.



Logarithmic plots

```
>> x = linspace(0,1,100);
>> y = exp(-10*x);
>> figure(1)
>> plot(x,y)
```

We can use logarithmic scale for the y axis

```
>> figure(2)
>> semilogy(x,y)
```

Now the variation of y is clearly seen. Study the matlab file `logplot.m` included in the `matlab` directory.

Also check out these other functions for logarithmic plots
`semilogx, loglog`

Sparse matrices

Suppose the matrix A has mostly zero entries

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 2 \\ 3 & 0 & 0 \end{bmatrix}$$

We will store only the non-zero entries.

Create a sparse matrix

```
>> A = sparse(3,3)
```

At this stage A is empty (zero matrix). Fill in non-zero entries

```
>> A(1,2) = 1;
>> A(2,3) = 2;
>> A(3,1) = 3;
```

To get normal matrix

```
>> B = full(A)
```

To convert normal matrix to sparse matrix

```
>> C = sparse(B)
```

Sparse matrices

Sparse diagonal matrix

$$A = \mathrm{diag}[1, -2, 1] = \begin{bmatrix} -2 & 1 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 1 & -2 \end{bmatrix} \in \mathbb{R}^{n\times n}$$

```
>> n = 10;
>> e = ones(n,1);
>> A = spdiags([e, -2*e, e], -1:1, n, n);
```

Sparse identity matrix

```
>> A = speye(5)
```

Eigenvalues and eigenvectors

$$Ax = \lambda x$$

```
>> A = rand(100,100);
>> lambda = eig(A);
>> plot(real(lambda), imag(lambda), 'o')
```

To get eigenvectors

```
>> [V,D] = eig(A);
```

Columns of `V` contain eigenvectors,

$$V = [e_1, e_2, \ldots, e_n] \in \mathbb{R}^{n\times n}, \qquad e_j \in \mathbb{R}^n$$

`D` is diagonal matrix with eigenvalues on the diagonal

$$D = \text{diag}[\lambda_1, \lambda_2, \ldots, \lambda_n]$$

$$Ae_j = \lambda_j e_j \qquad \Longrightarrow \qquad AV = VD$$

Eigenvalues and eigenvectors

Generalized eigenvalues/vectors

$$Ax = \lambda Bx$$

```
>> A = rand(10,10);
>> B = rand(10,10);
>> lambda = eig(A,B);
>> [V,D] = eig(A,B);
```

Sparse matrices

For large, sparse matrices, we may want to find only few eigenvalues, e.g., those with largest magnitude.

```
>> A = rand(10,10);
>> lambda = eigs(A,2)
```

To get eigenvectors and eigenvalues

```
>> [V,D] = eigs(A,2)
```

Similarly, to get generalized eigenvectors/values

Eigenvalues and eigenvectors

```
>> A = rand(10,10);
>> B = rand(10,10);
>> lambda = eigs(A,B,2)
>> [V,D] = eigs(A,B,2)
```

If matrix is **non-symmetric**, then we may want to compute eigenvalues with **largest real part**

```
>> lambda = eigs(A,B,2,'LR')
>> [V,D] = eigs(A,B,2,'LR')
```

Other options available are

```
'SR', 'LI', 'SI'
```

Numerical example: `eigtest.m`

Compute eigenvalues and eigenfunctions

$$-u''(x) = \lambda u(x), \qquad x \in (0,1)$$

$$u(0) = u(1) = 0$$

Exact eigenvalues and eigenfunctions

$$u_n(x) = \sin(n\pi x), \qquad \lambda_n = \pi^2 n^2, \qquad n = 1, 2, \ldots$$

Use finite difference method: form a grid

$$0 = x_0 < x_1 < x_2 < \ldots < x_{N+1} = 1, \qquad x_j - x_{j-1} = h = \frac{1}{N+1}$$

$$-\frac{u_{j-1} - 2u_j + u_{j+1}}{h^2} = \lambda u_j, \qquad j = 1, 2, \ldots, N$$

$$u_0 = u_{N+1} = 0$$

Numerical example: `eigtest.m`

Define

$$U = [u_1, u_2, \ldots, u_N]^\top, \qquad A = \text{diag}[-1, 2, -1] \in \mathbb{R}^{N \times N}$$

then the finite difference approximation is

$$AU = \lambda U$$

Excercises

1. Study the program `eigtest.m`
2. Run `eigtest.m`; first change your matlab directory to the directory `matlab`

```
>> pwd % This shows your current working directory
>> ls  % This shows contents of directory
```

You should be able to see the `eigtest.m` file in this directory. Now run it

```
>> eigtest
```

Numerical example: `eigtest.m`

3. Compare numerical and exact eigenvalues/eigenfunctions
   (Eigenfunctions are exact at the grid points. Can you explain why ?)
4. Make a copy of the file `eigtest.m` as `eigtest2.m`
   In `eigtest2.m`, replace the function `eig` with `eigs` and compute the 5 smallest eigenvalues

Solving system of ODE using ode15s

$$\frac{\mathrm{d}y}{\mathrm{d}t} = \mathsf{fun}(t, y, a, b, c, \ldots), \qquad \mathtt{T0} \le t \le \mathtt{TFINAL}$$

$$y(\mathtt{T0}) = y0$$

Write a matlab program `fun.m` which computes right hand side

```
function f = fun(t, y, a, b, c, ...)
```

`tspan`	`[T0, TFINAL]` or `[T0, T1, ..., TFINAL]` or `T0:dT:TFINAL`
`y0`	Initial condition $y(\mathtt{T0})$
`options`	`options = odeset('RelTol',1e-8,'AbsTol',1e-8);`

Solve ode

```
[t, Y] = ode15s(@fun, tspan, y0, options, a, b, c, ...)
```

`Y(:,i)` $=$ i'th component of solution at different times specified in `tspan`

Numerical example: `odetest.m`

This program solves the inverted pendulum problem which we will study in next lecture. We will solve the following non-linear ODE

$$\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \dot{z}_3 \\ \dot{z}_4 \end{bmatrix} = \begin{bmatrix} z_2 \\ \frac{1}{D}[ml\cos z_3(cz_4 - mgl\sin z_3) + (I + ml^2)(-kz_2 + mlz_4^2 \sin z_3)] \\ z_4 \\ \frac{1}{D}[(M+m)(-cz_4 + mgl\sin z_3) - ml\cos z_3(-kz_2 + mlz_4^2\sin z_3)] \end{bmatrix}$$

where

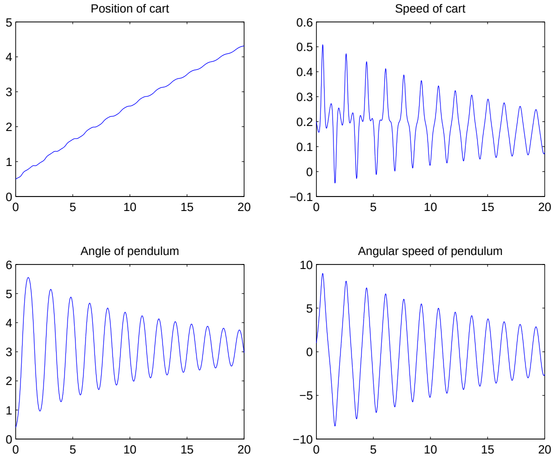
$$D = (M+m)(I+ml^2) - m^2l^2\cos^2 z_3$$

The values of various parameters are set in file `parameters.m`

Excercises

1. Study the programs: `fbo.m, odetest.m`
   `fbo.m` implements the right hand side function of the ODE
   `odetest.m` is the driver program which solves the ODE and plots the solution.
2. Run `odetest.m`; you will obtain solution as shown in figure below

Numerical example: `odetest.m`



Numerical example: `odetest.m`

3. Implement a program to solve the following problem

$$\begin{bmatrix} \dot z_1 \\ \dot z_2 \\ \dot z_3 \\ \dot z_4 \end{bmatrix} = \begin{bmatrix} z_2 \\ \frac{1}{D}[ml\cos z_3(cz_4 - mgl\sin z_3) + (I+ml^2)(F - kz_2 + mlz_4^2\sin z_3)] \\ z_4 \\ \frac{1}{D}[(M+m)(-cz_4 + mgl\sin z_3) - ml\cos z_3(F - kz_2 + mlz_4^2\sin z_3)] \end{bmatrix}$$

where

$$F = \alpha u - \beta z_2$$

$$u = -Kz, \qquad K = \begin{bmatrix} -10 & -16.1615 & -71.8081 & -15.2885 \end{bmatrix}$$

The value of α, β are set in `parameters.m` file.

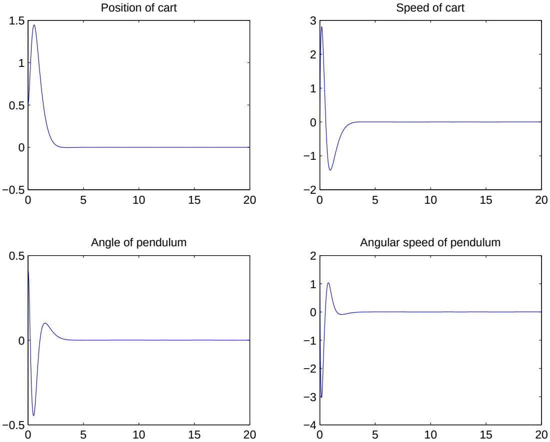
- Copy `fbo.m` as `fbf.m`, e.g. in Unix/Linux

```
cp fbo.m fbf.m
```

- You have to pass α, β in the arguments to `fbf` function.
- Modify `fbf.m` to include the force F
- Copy `odetest.m` as `odetest2.m`

Numerical example: `odetest.m`

- Modify `odetest2.m` to now use `fbf` instead of `fbo` and make sure to pass α, β
- Run `odetest2.m`; you should obtain solution as shown in figure below



Some checks

We will need some functions from the Control System toolbox. Check that you have this toolbox by typing following command

```
>> help lqr
```

If you get the message

```
lqr not found
```

then you do not have this toolbox. Talk to one of us.