LAB 5

Spring 2011, BESE- 15 A&B

Fundamental Concepts

Objective

The aim of this introductory lab is to introduce you to the basic functions in the Matlab and Numerical Methods with Matlab toolbox. By the end of today's lab, you should be able to understand the Symbolic calculation and Taylor Series files.

Submission Requirements

You are expected to complete the assigned tasks within the lab session and show them to the lab engineer/instructor. Some of these tasks are for practice purposes only while others (marked as '*Exercise*' or '*Question*') have to be answered in the form of a lab report that you need to prepare. Following guidelines will be helpful to you in carrying out the tasks and preparing the lab report.

Guidelines

- In the exercises, you have to put the output in your Lab report. You may add screen print to the report by using the 'Print Screen' command on your keyboard to get a snapshot of the displayed output. This point will become clear to you once you actually carry out the assigned tasks**.**

- Name your reports using the following convention:
  Lab#_Rank_YourFullName
  - '#' replaces the lab number
  - '*Rank*' replaces Maj/Capt/TC/NC/PC
  - '*YourFullName*' replaces your complete name.
- You need to submit the report even if you have demonstrated the exercises to the lab engineer/instructor or shown them the lab report during the lab session.

Symbolic Computations

Matlab can do symbolic computations, which means exact calculations using symbols as in Algebra or Calculus. Symbolic computations are performed with variables of the class sym (defined by the Symbolic Toolbox).

Defining functions and basic operations

Before doing any symbolic computation, one must declare the variables used to be symbolic:

```
>>syms x y                            %Create Variables x, y of sym class
or
>>x=syms('x');y=sym('y')
```

A function is defined by simply typing the formula:

```
>> f = cos(x) + 3*x^2                 %Construct symbolic expressions
f =cos(x)+3*x^2
or
>>r=5.1*x^3+a*x^2+b*x+c               %Defines a cubic polynomial in x with a sym
r = 51/10*x^3+a*x^2+b*x+c

>>s=sin(x)*exp(-x)-sinh(x)*x^2
s= sin(x)*exp(-x)-sinh(x)*x^2
```

In MATLAB the expressions f, g, r and q are nothing than other symbolic variables and you can combine them to define further expressions:

```
>>r*s+a*c+1
ans = (51/10*x^3+a*x^2+b*x+c)*(sin(x)*exp(-x)-sinh(x)*x^2)+a*c+1
```

The subs command
This can replace variables in an expression by other variables or numbers:

```
>>subs(r,a,c)                         %Replace variable a with variable c
ans = 51/10*x^3+c*x^2+b*x+c

>>subs(r,a,1)                         %Replace variable a with number 1
ans = 51/10*x^3+x^2+b*x+c
```

For multiple substitutions you need parantheses:

```
>>subs(r,{a,b},{c,1})                 %Replace a by c and b by 1.
ans = 51/10*x^3+c*x^2+x+c
```

If a substitution is made such that all variables are replaced by numbers, the answer is a floating-point number (class double):

```
>>subs(s,x,1)
ans = -0.8656
```

```
>> subs(f,pi)
ans = 28.6088
```

If define another function **>> g = exp(-y^2)** then we can **compose** the functions, The compose(f,g) returns f(g(y)) where f = f(x) and g = g(y):

```
>> h = compose(g,f)
```

i.e. h(x) = g(f(x)). Since f and g are functions of different variables, their product must be a function of two variables:

```
>> k = f*g
>> subs(k,{x,y},{0,1})
```

The diff command
Simple calculus operations, like differentiation can be done:

```
>>diff(r)                %Derivative of r with respect to default variable is found
ans = 153/10*x^2+2*a*x+b

>>diff(r,x)              %Derivative of r with respect to x
ans = 153/10*x^2+2*a*x+b

>>diff(r,a)              %Derivative of r with respect to a
ans = x^2

>>diff(r,2)              %2nd Derivative of r with respect to x (default)
ans = 153/5*x+2*a

>>diff(r,a,2)            %2nd Derivative of r with respect to a
ans = 0

>> f1 = diff(f)
```

The int command
This command finds the indefinite integral (antiderivative) of a function. The synatx is basically the same as for diff. Examples:

```
>>int(r)
ans = 51/40*x^4+1/3*a*x^3+1/2*b*x^2+c*x

>>int(r,a)
ans = 51/10*a*x^3+1/2*a^2*x^2+b*x*a+a*c

>> F = int(f)
F =sin(x)+x^3
```

Plotting a symbolic function can be done as follows:

```
>> ezplot(f)
```

For the domain can be specified:

```
>> ezplot(g,-10,10)
>> ezplot(g,-2,2)
```

Matlab allows you to do simple algebra. For instance:

```
>> poly = (x - 3)^5
poly =(x-3)^5
```

Expand distributes products over sums and applies other identities involving functions of sums

```
>> polyex = expand(poly)
polyex =x^5-15*x^4+90*x^3-270*x^2+405*x-243
```

```
>> polysi = simple(polyex)
polysi =(x-3)^5
```

Another useful property of symbolic functions is that you can substitute numerical vectors for the variables:

```
>> X = 2:0.1:4;
>> Y = subs(polyex,X);
>> plot(X,Y)
```

Finding roots

Consider the following polynomial and wish to find the roots of this polynomial:
$f(x) = 2x^2 + 4x - 8$

```
>> syms x
>> f=2*x^2 + 4*x -8;
>> solve(f,x)
ans =
5^(1/2)-1
-1-5^(1/2)
```

Alternately, you may use the following lines in Matlab to perform the same calculation:

```
>> f=[2 4 -8];
>> roots(f)
```

Matlab returns:

```
ans =
-3.2361
1.2361
```

Matrix Symbolic Calculation

First we need to define the symbolic variables:

```
>> syms a b c d e f g h
```

Matrix **A** is then defined as:

```
>> A=[a b; c d]
A =
[ a, b]
[ c, d]
```

Matrix **B** is defined as:

```
>> B=[e f;g h]
B =
[ e, f]
[ g, h]
```

The addition of these two matrices yields:

```
>> C=A+B
C =
[ a+e, b+f]
[ c+g, d+h]
```

The product of **A** and **B** is:

```
>> D=A*B
D =
[ a*e+b*g, a*f+b*h]
[ c*e+d*g, c*f+d*h]
```

If we wish to evaluate a specific matrix numerically, we simply assign the numeric values to the appropriate variable then use the command **eval** as demonstrate below.

```
>> a=1;b=2;c=3;d=4;e=5;f=6;e=7;f=8;g=9;h=0;
>> eval(A)
ans =
1 2
3 4

>> eval(B)
ans =
7 8
9 0

>> eval(C)
ans =
8 10
12 4
```

The inverse of **A** can be expressed symbolically:

```
>> D=inv(A)
D =
[ d/(a*d-b*c), -b/(a*d-b*c)]
[ -c/(a*d-b*c), a/(a*d-b*c)]
```

Numerically, **D** is expressed by

```
>> Dn=eval(inv(A))
Dn =
-2.0000 1.0000
1.5000 -0.5000
```

Taylor Series

Taylor function returns the Taylor series approximation

- r = taylor(f)
- r = taylor(f,n,v)
- r = taylor(f,n,v,a)

f	is a symbolic expression representing a function
v	specifies the independent variable in the expression
a	about which the series will represented
n	(n-1)-order Maclaurin polynomial approximation

```
>>syms x
>>f = 1/(5+4*cos(x))
>>T = taylor(f,8)
```

```
T =1/9+2/81*x^2+5/1458*x^4+49/131220*x^6
```

Exercise 1

Write a script file to find the 1st , 2nd and 3rd derivative of $z = \sin^2(\alpha) + \cos^2(\alpha)$

Exercise 2

Write a script to generate the Taylor series as created by the built in function.