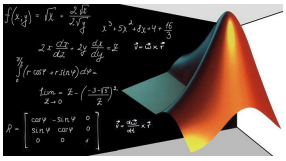
Advanced Concepts in MATLAB:

Symbolic Mathematics



PhD Course Work 2014

Under the Faculty of Engineering

Instructor: Dr. G. L. Prajapati

IET-DAVV, Indore

Objectives

- Create and manipulate symbolic variables
- Factor and simplify mathematical expressions
- Solve symbolic expressions
- Solve systems of equations
- Determine the symbolic derivative of an expression
- Integrate an expression symbolically

MuPad

- MATLAB's symbolic capability is based on the MuPad software
- MuPad is a GUI driven MATLAB package that helps you do algebra, calculus, as well as to graph and visualize functions.
  - Originally produced by SciFace Software and purchased by the Mathworks in 2008.
- Previous versions of MATLAB, before 2007b, used Maple as the symbolic engine

Symbolic Manipulation

- MuPAD® is available as part of the Symbolic Math Toolbox™ in MATLAB® – which is an option in the professional version
- Included with the Student Edition

Capabilities

- Manipulate symbolic expressions to
  - Simplify
  - Solve symbolically
  - Evaluate numerically
  - Take derivatives
  - Integrate
  - Perform linear algebraic manipulations
- More advanced features include
  - LaPlace transforms
  - Fourier transforms

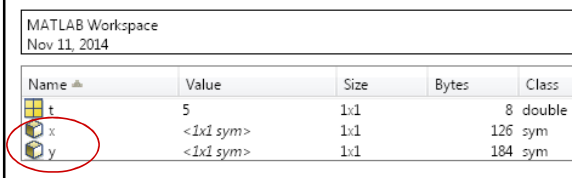
Creating Symbolic Variables

- Two approaches
  - Use the sym command to create
    - Single variable
    - Expression
    - Equation
  - Use the syms command to create
    - Single variables
    - Compose expressions and equations from the variables you've defined

Defining Symbolic Variables

- Define x as a symbolic variable
  - **x=sym('x')** or
  - **syms x**
- Use x to create a more complicated expression
  - **y = 2*(x+3)^2/(x^2+6*x+9)**

y = 2*(x+3)^2/(x^2+6*x+9)



y also becomes a symbolic variable because x is a symbolic variable in the expression

The syms command can create multiple variables

- **syms Q R T k0**
- Use these variables to create another symbolic variables

k=k0*exp(-Q/(R*T))

Notice that we used standard algebraic operators – the array operators (.*, ./ and .^) are not used in symbolic algebra

Create an entire expression with the sym command

- **E=sym('m*c^2')**
- Since m and c have not been specifically defined as symbolic variables, they are not stored
- **E** was set equal to a character string, defined by the single quotes inside the function.

Substitution: the subs function

- Once we have a symbolic expression, we may want to substitute values into it, we use the subs function. **Syntax**

  R = subs(S)

  R = subs(S, new) %substitutes Default Variable with new

  R = subs(S,old,new)

R = subs(S) %replaces all occurrences of variables in the symbolic expression S with values obtained from the calling function, or the MATLAB workspace.

Substitution …

```
>> syms a b c x
>> f = a*x^2 + b*x + c
f =
a*x^2 + b*x + c
 >> g = subs(f, a, 3)
g =
 3*x^2 + b*x + c
```

```
>> h = subs(f, {a, b, c}, {1, 2, 3})
h =
x^2 + 2*x + 3
>> subs(h, x, -2)
ans =
   3
```

```
>> f % The symbolic expression itself is unchanged.
f =
a*x^2 + b*x + c
```

Substitution …

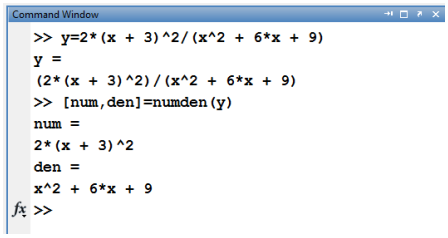
- Substituting an expression,

```
>> y=x-1;
>> h = x^2 + 2*x + 3;
>> subs(h, x, y)

ans =

2*x + (x - 1)^2 + 1
```

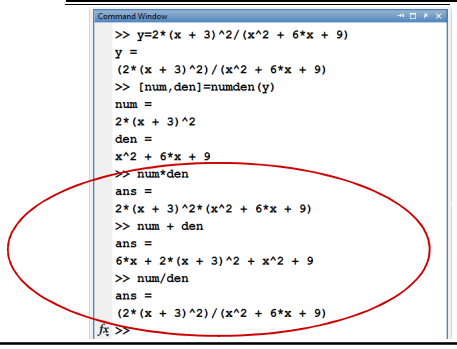
```
>> simplify(ans)
ans =
x^2 + 2
>> h+ans
 ans =

2*x^2 + 2*x + 5
```

Extracting Numerators and Denominators

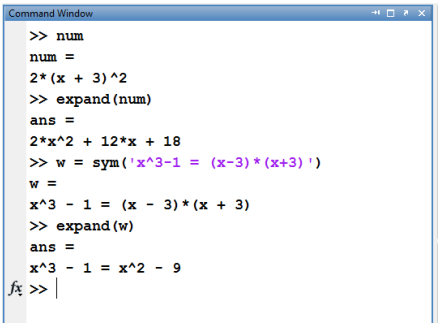
- The numden function extracts the numerator and denominator from an **expression**



We can combine symbolic variables using standard algebraic operators



Symbolic expansion of polynomials and elementary functions



Function solve

solve(eq)

- The input to solve can be either symbolic expressions or strings. If eq is a symbolic expression (x^2-2*x+1) or a string that does not contain an equal sign ('x^2-2*x+1'), then solve(eq) solves the equation eq=0 for its default variable

Example

```
>> syms x y;
>> y=sym('x^2+12*x+10')
y =
   x^2+12*x+10
>> solve(y)
ans =
   - 26^(1/2) - 6
     26^(1/2) - 6
```

solve …

- Use MATLAB's symbolic capability to solve an equation
  - k = k0*exp(-Q/RT)
  - Solve for Q

Solutions

Hand solution

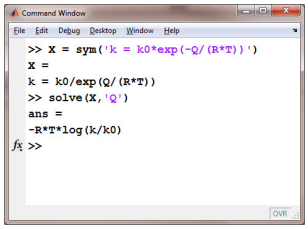
$$k = k_0 \exp\left(\frac{-Q}{RT}\right)$$

$$\frac{k}{k_0} = \exp\left(\frac{-Q}{RT}\right)$$

$$\ln\left(\frac{k}{k_0}\right) = \frac{-Q}{RT}$$

$$Q = RT \ln\left(\frac{k_0}{k}\right)$$

MATLAB Solution



Differentiation

- The diff function is used to determine the symbolic derivative of a symbolic expression f.
  - diff(f) returns the derivative of f w.r.t. the default independent variable
  - diff(f, t) returns the derivative of f w.r.t. the variable t
  - diff(f, n) returns the nth derivative of f w.r.t the default independent variable
  - diff(f, t, n) returns the nth derivative of f w.r.t the variable t

Differentiation …

```
>> syms x y z t;
>> f=2*x^2+3*y-9*(z-t)
f =
2*x^2 + 9*t + 3*y - 9*z
>> diff(f) %differentiate f with respect to x (the
   default variable)
ans =
4*x
>> diff(f,2) % differentiate f twice with respect to x
ans =
4
```

Differentiation …

```
f =
2*x^2 + 9*t + 3*y - 9*z
>> diff(f,z,2) %
   differentiate f twice
   with respect to z
ans =
0
```

```
>> A=[3*sin(x) -cos(x^2);
   cos(2*x) -sin(x)]
A =
[ 3*sin(x), -cos(x^2)]
[ cos(2*x),   -sin(x)]
>> diff(A)
ans =
[     3*cos(x), 2*x*sin(x^2)]
[ (-2)*sin(2*x),     -cos(x)]
```

Integration

- MATLAB supports the following symbolic integration capabilities; the int function attempts to find a function F such that diff(F) = f; diff(F) being the derivative of F.
  - int(f) returns the integral of the expression f w.r.t. the independent variable
  - int(f, t) returns the integral of f w.r.t. the variable t
  - int(f, a, b) returns the integral of f w.r.t. the independent variable over the interval [a,b]; a and b numerical
  - int(f, t, a, b) returns the integral of f w.r.t to t over the interval [a,b]; a and b numerical
  - Int(f, 'm', 'n') returns the integal of f w.r.t. the independent variable over the interval [m,n] where m and n are symbolic expressions.

Integration …

```
>> syms x y a b
>> f=sin(2*x+y)
f =
sin(2*x + y)
>> int(f) %integrate f
   with respect to
   default variable x
ans =
-cos(2*x + y)/2
```

- >> int(f,pi/2,pi) %integrate f with respect to default variable x, with limits pi/2 to pi

```
ans =
-cos(y)
```

Integration …

```
>> syms x y a b
>> f=sin(2*x+y)
f =
sin(2*x + y)
>> int(f,y)
   %integrate f with
   respect to y
ans =
-cos(2*x + y)
```

```
>> int(f,y,pi/2,pi) %integrate f with
   respect to y with limits pi/2 to pi
ans =
cos(2*x) - sin(2*x)
>> int(f,a,b) %integrate f with
   respect to x with limits a and b
ans =
cos(2*a + y)/2 - cos(2*b + y)/2
```

Integration …

```
>> int(f,'a','b')
   %integrate f with
   respect to x over
   the interval [a b]
   where a and b
   are symbolic
   expressions
ans =
cos(2*a + y)/2 -
   cos(2*b + y)/2
```

```
>> int(f,'a+2','b+3')
   %integrate f with
   respect to x with
   limits a+2 and b+3
ans =
cos(2*a + y + 4)/2 -
   cos(2*b + y + 6)/2
```

Integration …

```
>> A=[3*sin(x) -cos(x);
   cos(2*x) -sin(x)]
 A =

[ 3*sin(x), -cos(x)]
[ cos(2*x), -sin(x)]
```

```
>> int(A) %
   integrating a
   symbolic
   matrix/array

ans =
[ (-3)*cos(x), -sin(x)]
[  sin(2*x)/2,  cos(x)]
```

System of Linear Equations

```
>> syms x y z f1 f2 f3
>> f1=2*x+8*y+2*z-26
 f1 =
 2*x + 8*y + 2*z - 26
 >> f2=4*y+3*x+6*z-12
f2 =
 3*x + 4*y + 6*z - 12
>> f3=6*x+4*z+2*y-14
 f3 =
 6*x + 2*y + 4*z - 14
```

```
>> [x y z]=solve(f1,f2,f3)
   %important to order the
   output parameters in
   lexicographic manner
x =
2
y =
3
z =
-1
```

Solving differential equations

- MATLAB's symbolic toolbox provides capabilities to solve differential equations.
- **dsolve** is the function that we need to explore. Consider the following,
  Suppose we have the equation dx/dt = -a*x and we want to solve for x.
- Use D to represent differentiation against the independent variable.
- D2, D3, and so on are used to denote repeated differentiation. Any letters following Ds are dependent variables.
- The equation dx/dt = -a*x is represented as: Dx = - a*x
- The equation $d^2y/dt^2 = 0$ is represented as: D2y = 0
- The independent variable is assumed as *t*, unless specified

Solving differential equations …

```
>> syms x y f t
>> dsolve('Dx = - a*x')
 ans =
 C2/exp(a*t)  % note the
   undetermined constant
   C2

>> dsolve('Df = f + sin(t)')

ans =
 C24*exp(t) - sin(t)/2 -
   cos(t)/2
```

```
>>dsolve('Dy = a*y')
 ans =
 C28*exp(a*t)
>> dsolve('Dy = a*y',
   'y(0) = b')
ans =
b*exp(a*t)
```

Solving differential equations …

```
>> [x y] =
    dsolve('Dx = y',
    'Dy = -x')
 x =

(C30*i)/exp(i*t) -
    C29*i*exp(i*t)

y =

C29*exp(i*t) +
    C30/exp(i*t)
```

```
>> dsolve('D2y -2*Dy-3*y =0')
 ans =
C37*exp(3*t) + C38/exp(t)
 >> dsolve('D2y -2*Dy-3*y
    =0','y(0)=0,y(1)=1','x')
 ans =
 1/(exp(x)*(1/exp(1) - exp(3))) -
    exp(3*x)/(1/exp(1) - exp(3))
 >> simple(ans)
ans =
 (1/exp(x) - exp(3*x))/(1/exp(1) - exp(3))
```

Exercise

Problem: Use MATLAB to create a plot of the function $f(x) = \sin(x^2)\cos(x)$ and its derivative $f'(x)$ on the same set of axes over the interval $-\pi \leq x \leq \pi$. The x-axis tick marks should be at points with coordinates $-\pi, -\pi/2, 0, \pi/2, \pi$.

Solution

- %Matlab Code
- clear all;
- clc;
- % Generating the data to be plotted
- syms x
- domf=linspace(-pi,pi,60);
- domg=domf;
- f= @(x) sin(x.^2).*cos(x);
- dfx=diff(f(x), x);
- g=@(x) subs(dfx);

Solution …

- % Plotting the generated data
- clf; %Clear current figure window
- figure(1);
- PLOT1=plot(domf,f(domf));
- hold on
- PLOT2=plot(domg,g(domg));
- hold off

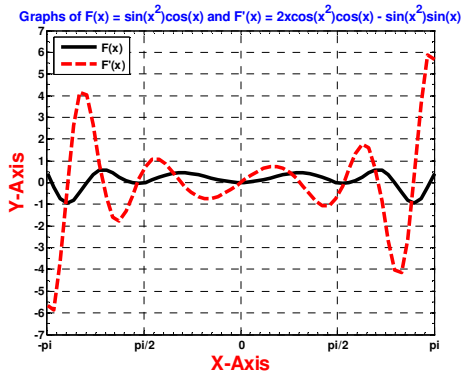
Solution …

- %Derssing up your plot
- title({'Graphs of F(x) = sin(x^2)cos(x) and F''(x) = 2xcos(x^2)cos(x) - sin(x^2)sin(x)'}, 'color', 'blue', 'FontName', 'mathematica', 'FontWeight','bold','FontSize', 12)
- legend('F(x)','F''(x)','Location','NorthWest')
- set(PLOT1,'color', 'black','LineWidth',3,'LineStyle','-')
- set(PLOT2,'color', 'red','LineWidth',3,'LineStyle','--')

Solution …

- set(gca, 'XTick', [-pi:pi/2:pi],'XMinorTick', 'on', 'XTickLabel', {'-pi','pi/2','0','pi/2','pi'},'FontName', 'mathematica', 'FontWeight','bold')
- set(gca, 'YTick', [-7:7],'YMinorTick', 'on','FontName', 'mathematica', 'FontWeight','bold')
- xlabel('X-Axis','Color','red','FontName', 'mathematica', 'FontWeight','bold', 'FontSize', 16)
- ylabel('Y-Axis','Color','red','FontName', 'mathematica', 'FontWeight','bold', 'FontSize', 16)
- axis([-pi pi -7 7])
- grid on

Figure 1



Plotting Symbolic Expressions

- MATLAB provides an easy way to plot symbolic expressions, it is called ezplot. Suppose S is a symbolic expression of x, then

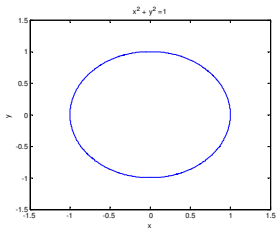
  >>ezplot(S, [xmin, xmax])

  will plot S between the limits of xmin and xmax.

  >> ezplot(S) always uses the range [-2 *pi, 2*pi]
- You can use the commands xlabel, ylabel, and title to add x and y labels, etc., in the usual way. Also use grid on, hold on and hold off, subplots, etc.

ezplot supports implicit plotting

- The equation for a circle can be expressed implicitly as:
  - $x^2 + y^2 = 1$
- You could solve for y, but it's not necessary with ezplot
- **ezplot('x^2 + y^2 =1',[-1.5,1.5])**

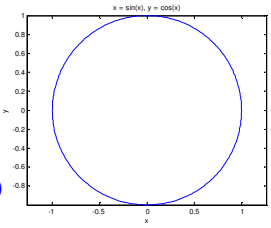


ezplot supports parametric equation graphs

The equation for a circle can be expressed parametrically as:

- x=sin(t)
- y=cos(t)

>>ezplot('sin(x)','cos(x)')



Symbolic Plot Types		
ezplot	Function plotter	if z is a function of x **ezplot(z)**
ezmesh	Mesh plotter	if z is a function of x and y **ezmesh(z)**
ezmeshc	Combined mesh and contour plotter	if z is a function of x and y **ezmeshc(z)**
ezsurf	Surface plotter	if z is a function of x and y **ezsurf(z)**
ezsurfc	Combined surface and contour plotter	if z is a function of x and y **ezsurfc(z)**
ezcontour	Contour plotter	if z is a function of x and y **ezcontour(z)**
ezcontourf	Filled contour plotter	if z is a function of x and y **ezcontourf(z)**
ezplot3	3-D parametric curve plotter	if x is a function of t if y is a function of t if z is a function of t ezplot3(x,y,z)
ezpolar	Polar Coordinate plotter	if r is a function of θ ezpolar(r)

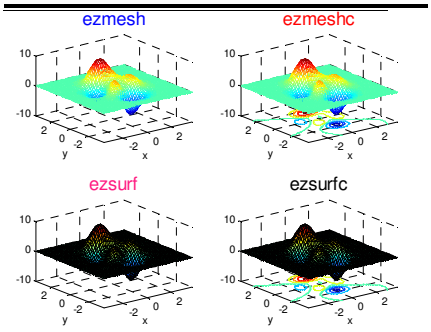
Demonstration of these Plots

- clear all
- close all
- clc
- z1=sym('3*(1-x)^2*exp(-(x^2)-(y+1)^2)');
- z2=sym('-10*(x/5-x^3-y^5)*exp(-x^2-y^2)');
- z3=sym('-1/3*exp(-(x+1)^2-y^2)');
- z=z1+z2+z3;

Demonstration of these Plots…

- subplot(2,2,1)
- ezmesh(z);
- title('ezmesh','Color', 'blue','FontSize',16);
- subplot(2,2,2)
- ezmeshc(z);
- title('ezmeshc','Color', 'red','FontSize',16);
- subplot(2,2,3)
- ezsurf(z);
- title('ezsurf','Color', [1 .1 .5],'FontSize',16);
- subplot(2,2,4)
- ezsurfc(z);
- title('ezsurfc','Color', 'black','FontSize',16);

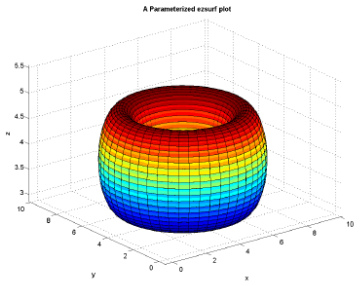
Output Figure: Demonstration of these Plots…



Another Example

- x=sym('4+(3+cos(v))*sin(u)');
- y=sym('4+(3+cos(v))*cos(u)');
- z=sym('4+sin(v)');
- ezsurf(x,y,z);
- title('\bfA Parameterized ezsurf plot');

Another Example [output figure]



Thanks