INTRODUCTION TO SCIENTIFIC & ENG. COMPUTING
BIL 108E, CRN 44448

Week.10_Appl. of curve fitting- Approxm. & Inregration

Week.11_ Symbolic Mathematics

Dr. Feyzi HAZNEDAROĞLU

İstanbul Teknik Üniversitesi - İnşaat Fakültesi
Room : 250A, e-mail :haznedar@itu.edu.tr

29 - 04 - 2011

TENTATIVE SCHEDULE

Wk	Date	Topics
1	Feb. 11	Introduction to Scientific and Engineering Computing
2	Feb. 18	Introduction to Program Computing Environment
3	Feb. 25	Variables, Operations and Simple Plot
4	Mar. 04	Algorithms and Logic Operators
5	Mar. 11	Flow Control, Errors and Source of Errors
6	Mar. 18	Functions & Linear Algebra
7	Mar. 25	Arrays
8	Apr. 01	Solving of Simple Equations
9	Apr. 08 -15	Polynomials Examples Exam 1
10	Apr. 15 -22	Applications of Curve Fitting
11	Apr. 29	Applications of Interpolation
12	May 06	Applications of Numerical Integration
13	May 13	Symbolic Mathematics
14	-- -- --	Ord Dif.Equations (ODE) solution with Built-in Functions

Week 11

- LECTURE # 11

SYMBOLIC MATHEMATICS

- Symbolic Objects
- Creating Symbolic Variables and Expressions Performing Symbolic Computations

SYMBOLIC MATH TOOLBOX

SYMBOLIC MATH TOOLBOX

- Symbolic Math Toolbox software lets you to perform symbolic computations within the MATLAB numeric environment. It provides tools for solving and manipulating symbolic math expressions and performing variable-precision arithmetic.
- Symbolic Math Toolbox software also includes the MuPAD language, which is optimized for handling and operating on symbolic math expressions.

SYMBOLIC OBJECTS

- Symbolic objects are a special MATLAB data type introduced by the Symbolic Math Toolbox software.
- They allow you to perform mathematical operations in the MATLAB workspace analytically, without calculating numeric values.
- Symbolic objects present
  symbolic **variables**, symbolic **numbers**,
  symbolic **expressions** and symbolic **matrices**.

SYMBOLIC VARIABLES

- To declare variables x and y as symbolic objects use the syms command:

```
syms x y
```

- You can manipulate the symbolic objects according to the usual rules of mathematics. For example:

```
x + x + y
ans =
2*x + y
```


SYMBOLIC NUMBERS

- Symbolic Math Toolbox software also enables you to convert numbers to symbolic objects. To create a symbolic number, use the sym command:

```
a = sym ('8')
```

Note: Symbolic results are not indented.

SYMBOLIC NUMBERS

- The following example illustrates the difference between a standard double-precision MATLAB data and the corresponding symbolic number.
- The MATLAB command `sqrt(2)` returns a double-precision floating-point number:

```
ans =
    1.4142
```

- On the other hand, if you calculate a square root of a symbolic number 2:

```
a = sqrt(sym(2))
```

you get the precise symbolic result:

```
a =
2^(1/2)
```

SYMBOLIC NUMBERS

- To evaluate a symbolic number numerically, use the `double` command:

```
double(a)
ans =
   1.4142
```

- You also can create a rational fraction involving symbolic numbers:

```
sym(2)/sym(5)
ans =
    2/5
```


CREATING SYMBOLIC VARIABLES AND EXPRESSIONS

- The sym command creates symbolic variables and expressions.

- For example, the commands

```
x = sym('x');
a = sym('alpha');
```

create a symbolic variable *x* with the value *x* assigned to it in the MATLAB workspace and a symbolic variable *a* with the value *alpha* assigned to it.

- An alternate way to create a symbolic object is to use the syms command:

```
syms x;
a = sym('alpha');
```


CREATING SYMBOLIC VARIABLES AND EXPRESSIONS

- `findsym` determines variables in symbolic expression or matrix.

- Note: `findsym` is not recommended. Use `symvar` instead.

```
findsym(S)
findsym(S, n)
```

- `findsym(S, n)` returns the `n` variables alphabetically closest to `x`.

CREATING SYMBOLIC VARIABLES AND EXPRESSIONS

You can use `sym` or `syms` to create symbolic variables.

- The `syms` command:
  - Does not use parentheses and quotation marks: `syms x` can create multiple objects with one call.
  - Serves best for creating individual single and multiple symbolic **variables**
- The `sym` command:
  - Requires parentheses and quotation marks: `x = sym('x')`. When creating a symbolic number with 10 or fewer decimal digits, you can skip the quotation marks:

    `f = sym (5).`
  - Creates one symbolic object with each call.
  - Serves best for creating symbolic **numbers** and symbolic **expressions**.
  - Serves best for creating symbolic **objects in functions** and **scripts**.

CREATING SYMBOLIC VARIABLES AND EXPRESSIONS

- Suppose you want to use a symbolic variable to represent the golden ratio . $\rho = \frac{1+\sqrt{5}}{2}$
- The command
  `rho = sym('(1 + sqrt(5))/2');` achieves this goal.
- Now you can perform various mathematical operations on rho. For example,
  `f = rho^2 - rho – 1` returns;

```
f =
    (5^(1/2)/2 + 1/2)^2 - 5^(1/2)/2 - 3/2
```

- Note: Do not use `syms` command to create a symbolic expression that is a constant.

CREATING SYMBOLIC OBJECTS WITH IDENTICAL NAMES

- You can use the `syms` command to clear variables of definitions that you previously assigned to them in your MATLAB session

CREATING A MATRIX OF SYMBOLIC VARIABLES

- You can create a symbolic matrix A whose elements are a, b, and c, using the commands:

```
syms a b c;

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

- Using symbolic objects is very similar to using regular MATLAB numeric objects.

CREATING SYMBOLIC VARIABLES AND EXPRESSIONS

- A particularly effective use of `sym` is to convert a matrix from numeric to symbolic form.
- To determine what symbolic variables are present in an expression, use the `symvar` command.
- For example, given the symbolic expressions f and g defined by

```
syms a b n t x z;
f = x^n;
g = sin(a*t + b);
```

you can find the symbolic variables in *f* by entering:

```
symvar(f)
ans =
    [ n, x]
```

SIMPLIFYING SYMBOLIC

- Symbolic Math Toolbox provides a set of simplification functions allowing you to manipulate an output of a symbolic expression.

- For example, the following polynomial of the golden ratio

```
rho
rho = sym('(1 + sqrt(5))/2');
f = rho^2 - rho - 1
```

You can simplify this answer by entering `simplify(f)` and get a very short answer:

```
ans =
     0
```


SUBSTITUTING IN SYMBOLIC EXPRESSIONS

yerine koymak

- You can substitute a numeric value for a symbolic variable or replace one symbolic variable with another using the `subs` command.

- For example, to substitute the value x = 2 in the symbolic expression

```
syms x;
```

- `f = 2*x^2 - 3*x + 1;` enter the command

```
subs(f, 2)
ans =
     3
```

=8-6+1

SUBSTITUTING IN MULTIVARIATE EXPRESSIONS

- When your expression contains more than one variable, you can specify the variable for which you want to make the substitution.

yerine geçme,

- For example, to substitute the value x = 3 in the symbolic expression

```
syms x y;
f = x^2*y + 5*x*sqrt(y);
```

enter the command

```
subs(f, x, 3)
ans =
    9*y + 15*y^(1/2)
```

- `subs(f, y, x)`

```
ans =
    x^3 + 5*x^(3/2)
```


- CONVERSION
- `sym2poly` is used to get a row vector containing the numeric coefficients of the polynomial.

```
syms x;
f = x^3 - 15*x^2 - 24*x + 350;
b = sym2poly(f)
b =
1 -15 -24 350
```

- `poly2sym` converts polynomial coefficient vector to symbolic polynomial.

- The command `poly2sym([ 1 3 2 ])` returns;

```
ans =
x^2 + 3*x + 2
```

SYMBOLIC MATH TOOLBOX

- CONVERSION

- You can convert the standard double-precision variable into a symbolic object:

```
t = 0.1;
sym(t)
ans =
    1/10
```


SYMBOLIC MATH TOOLBOX

- CONVERSION
- The technique for converting floating-point numbers is specified by the optional second argument, which can be 'f', 'r', 'e' or 'd'.

```
t = 0.1;
```
⇒ fractional

- `sym(t, 'f')`
```
ans = 3602879701896397/36028797018963968
```
- `sym(t, 'r')` ⇒ ratio
```
ans = 1/10
```
- `sym(t, 'e')` ⇒ natural logarithmic
```
ans = eps/40 + 1/10
```
- `sym(t, 'd')`
```
ans = 0.1000000000000000055511151231257 8
```


SYMBOLIC MATH TOOLBOX

DIGITS

- To change the number of significant digits, use the `digits` command:

```
digits(7);
sym(t, 'd')
ans =
    0.1
```


SYMBOLIC MATH TOOLBOX

DIFFERENTIATING SYMBOLIC EXPRESSIONS

- With the Symbolic Math Toolbox software, you can find;
  - ✓ Derivatives of single-variable expressions,
  - ✓ Partial derivatives,
  - ✓ Second and higher order derivatives,
  - ✓ Mixed derivatives.

DIFFERENTIATING SYMBOLIC EXPRESSIONS

- To differentiate a symbolic expression, use the `diff` command.

Türev

```
syms x;
f = sin(x)^2;
diff(f)
ans =
    2*cos(x)*sin(x)
```


DIFFERENTIATING SYMBOLIC EXPRESSIONS

EXAMPLE:

-
```
syms x y;
f = sin(x)^2 + cos(y)^2;
diff(f, y)
ans =
    (-2)*cos(y)*sin(y)
```

-
```
diff(f, y, 2)
diff(diff(f, y))
ans =
    2*sin(y)^2 - 2*cos(y)^2
```


INTEGRATING SYMBOLIC EXPRESSIONS

- You can perform symbolic integration including:
  - ✓ **Indefinite** and **definite** integration
  - ✓ Integration of multivariable expressions

-
```
syms x;
f = sin(x)^2;
```

- To find the **indefinite integral**, enter

```
int(f)
ans =
    x/2 - sin(2*x)/4
```


INTEGRATING SYMBOLIC EXPRESSIONS

EXAMPLE:

-
```
syms x y n;
f = x^n + y^n;
int(f, y)
ans =
   x^n*y + (y*y^n)/(n + 1)
```

- To find a **definite integral**, pass the limits of integration as the final two arguments of the `int` function:

```
syms x y n;
f = x^n + y^n;
int(f, 1, 10)
ans =
piecewise([n = -1, log(10) + 9/y],...
[n <> -1, (10*10^n - 1)/(n + 1) + 9*y^n])
```

SOLVING EQUATIONS

- You can solve different types of symbolic equations including:
  - ✓ **Algebraic equations** with **one** symbolic variable
  - ✓ **Algebraic equations** with **several** symbolic variables
  - ✓ **Systems of algebraic equations**

SOLVING EQUATIONS

EXAMPLE:

- You can find the values of variable *x* for which the following expression is equal to zero:

```
syms x; solve(x^3 - 6*x^2 + 11*x - 6)
ans =
     1
     2
     3
```

- By default, the `solve` command assumes that the right-side of the equation is equal to zero.
-
```
syms x;
solve('x^3 - 6*x^2 + 11*x - 5 = 1')
```


SOLVING EQUATIONS

EXAMPLE:

-
```
syms x y;
f = 6*x^2 - 6*x^2*y + x*y^2 - x*y + y^3 - y^2;
```

- with respect to a symbolic variable y:

```
solve(f, y)
ans =
     1
     2*x
    -3*x
```


SOLVING EQUATIONS

EXAMPLE:

-
```
syms x y z;
```
-
```
[x, y, z]=solve('z=4*x', 'x=y', 'z=x^2+y^2')
```

```
x =
    0
    2
y =
    0
    2
z =
    0
    8
```

DEFAULT SYMBOLIC VARIABLE

- When performing substitution, differentiation, or integration, if you do not specify a variable to use, MATLAB uses a default variable.
- The default variable is basically the one closest alphabetically to *x*.
- To find which variable is chosen as a default variable, use the `symvar(expression, 1)` command.

```
syms s t;
g = s + t;
symvar(g, 1)
ans =
      t
```


SYMBOLIC FUNCTION PLOTTING

- You can create different types of graphs including:
  - ✓ Plots of **explicit functions**
  - ✓ Plots of **implicit functions**
  - ✓ **3-D parametric plots**
  - ✓ **Surface plots**

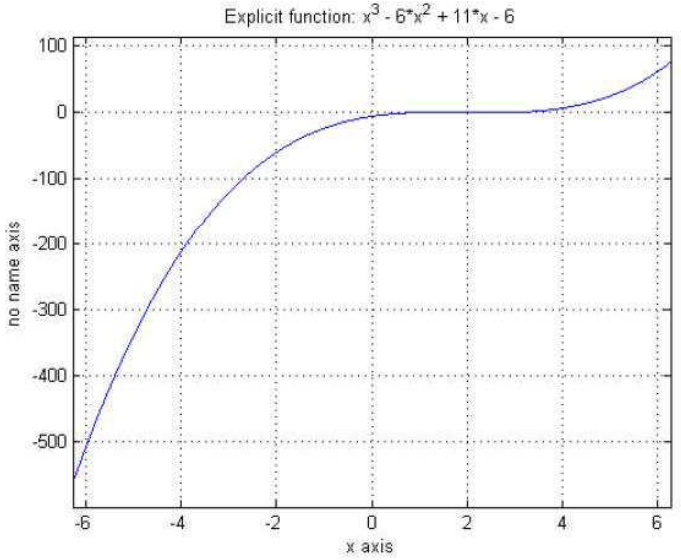
EXPLICIT FUNCTION PLOTTING

Belirtik, açık

The simplest way to create a plot is to use the `ezplot` command:

```
syms x;
ezplot(x^3 - 6*x^2 + 11*x - 6);
hold on;
xlabel('x axis');
ylabel('no name axis');
title('Explicit function: x^3 -6*x^2 +11*x -6');
grid on;
hold off
```


- EXPLICIT FUNCTION PLOTTING



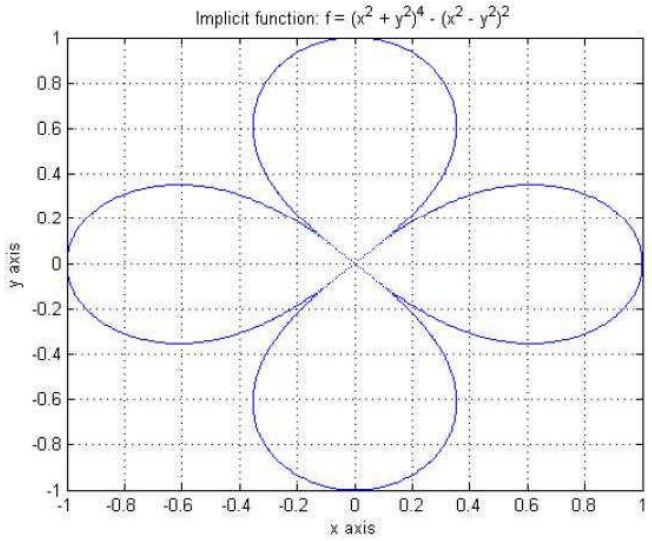
- IMPLICIT FUNCTION PLOTTING

kesin, kapalı

- Create a plot for the following implicit function over the domain $-1 < x < 1$:

```
syms x y;
f = (x^2 + y^2)^4 - (x^2 - y^2)^2;
ezplot(f, [-1 1]);
hold on;
xlabel('x axis');
ylabel('y axis');
title('Implicit function: f=(x^2+y^2)^4 -(x^2-y^2)^2');
grid on;
hold off
```


- IMPLICIT FUNCTION PLOTTING

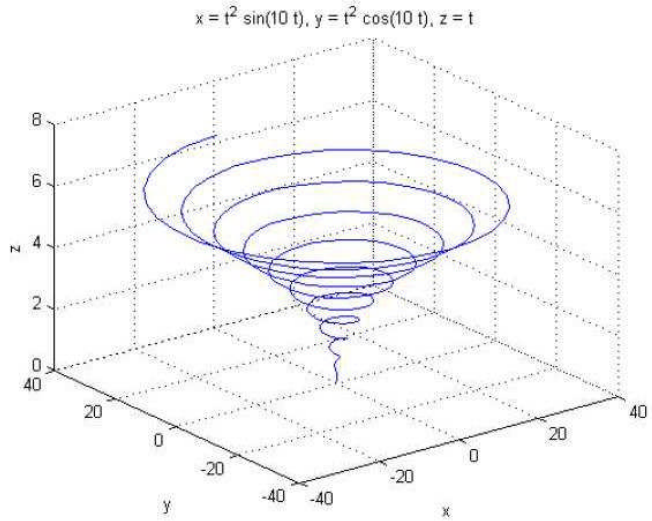


3-D PLOT

- To create a 3-D plot, use the `ezplot3` command.

```
syms t;
ezplot3(t^2*sin(10*t), t^2*cos(10*t), t);
```

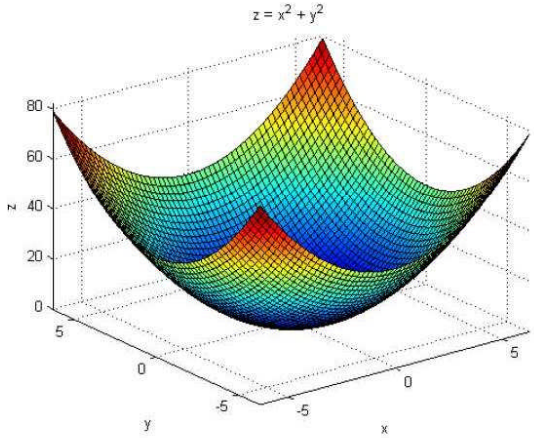

3-D PLOT



SURFACE PLOT

- If you want to create a surface plot, use the `ezsurf` command.
- For example, to plot a paraboloid $z = x^2 + y^2$, enter:
- ```
  syms x y;
  ezsurf(x^2 + y^2);
  hold on;
  zlabel('z');
  title('z = x^2 + y^2');
  hold off
  ```


- SURFACE PLOT



`pretty`

- The command `pretty(T)` prints *T* in a format resembling typeset mathematics:
- T = (49*x^6) / 131220 + (5*x^4) / 1458 + (2*x^2) / 81 + 1 / 9

```
      6       4       2
   49x      5x      2x      1
---------- + -------- + ---- + -----
131220    1458     81      9
```


SIMPLIFICATIONS

- Here are three different symbolic expressions.

```
syms x
f = x^3 - 6*x^2 + 11*x - 6;
g = (x - 1)*(x - 2)*(x - 3);
h = -6 + (11 + (-6 + x)*x)*x;
```

- Here are their *prettyprinted* forms, generated by

```
pretty(f);
pretty(g);
pretty(h)
```

- x^3 - 6 x^2 + 11 x - 6
  (x - 1)*(x - 2)*(x - 3)
  x (x (x - 6) + 11) - 6
```

SIMPLIFICATIONS

- These expressions are three (3) different representations of the same mathematical function, a cubic polynomial in ***x***.
- Polynomial form, factored form, Horner or nested form

- Polynomial form is simply a linear combination of the powers of ***x***.
- Factored form displays the roots of the polynomial and is the most accurate for numerical evaluation near the roots.
- For numerical evaluation, Horner or nested form involves the fewest arithmetic operations and is the most accurate for some other ranges of ***x***.

SIMPLIFICATIONS

`collect`

- The statement `collect(f)` views *f* as a polynomial in its symbolic variable,

```
syms x;
f = x*(x*(x - 6) + 11) - 6;  ⇒   symbolic
collect(f)
ans =
    x^3 - 6*x^2 + 11*x – 6   ⇒   polynomial
```

```
syms x t;
f = (1+x)*t + x*t;
collect(f)
ans =
    (2*t)*x + t
```


SIMPLIFICATIONS

`expand`

- The statement `expand(f)` distributes products over sums and applies other identities involving functions of sums

```
syms x y;
f = cos(x + y);
expand(f)
ans =
    cos(x)*cos(y) - sin(x)*sin(y)
```


SIMPLIFICATIONS

`horner`

- The statement `horner(f)` transforms a symbolic polynomial *f* into its horner, or nested, representation as shown in the following examples. içiçe geçmiş

```
syms x;
f = x^3 - 6*x^2 + 11*x - 6;
horner(f)
ans =
    x*(x*(x - 6) + 11) - 6
```

SIMPLIFICATIONS

`factor`

- If *f* is a polynomial with rational coefficients, the statement `factor(f)` expresses *f* as a product of polynomials of lower degree with rational coefficients.

- ```
  syms x;
  f = x^6 + 1;
  factor(f)
  ans =
      (x^2 + 1)*(x^4 - x^2 + 1)
  ```


LINEAR ALGEBRA

- Basic algebraic operations on symbolic objects are the same as operations on MATLAB objects.

- ```
  syms t;
  G = [cos(t) sin(t); -sin(t) cos(t)]
  A = G^2
  A =
  [ cos(t)^2 - sin(t)^2, 2*cos(t)*sin(t)]
  [ -2*cos(t)*sin(t), cos(t)^2 - sin(t)^2]
  ```


- LINEAR ALGEBRA & OPERATIONS

- ```
  A = simple(A)
  A =
  [  cos(2*t), sin(2*t)]
  [ -sin(2*t), cos(2*t)]
  ```

- The command `H = hilb(3)` generates the 3-by-3 Hilbert matrix. With format short, MATLAB prints

  ```
  H =
  1.0000 0.5000 0.3333
  0.5000 0.3333 0.2500
  0.3333 0.2500 0.2000
  ```


ALGEBRAIC OPERATIONS

- Converting H to a symbolic matrix

  `H = sym(H)`

  gives

  ```
  H =
  [  1,  1/2, 1/3]
  [ 1/2, 1/3, 1/4]
  [ 1/3, 1/4, 1/5]
  ```
  ```

ALGEBRAIC OPERATIONS

`inv(H)` produces

```
ans =
[   9, -36 ,   30]
[ -36, 192 , -180]
[  30,-180 ,  180]
```

```
det(H)
ans=
    1/2160
```


SYMBOLIC SUMMATION

- You can compute symbolic summations, when they exist, by using the `symsum` command. For example, the p-series

$1+\frac{1}{2^2}+\frac{1}{3^2}+\dots$
sums to $\frac{\pi^2}{6}$, while the geometric series
$1+x^2+x^3+\dots$
sums to $\frac{1}{(1-x)}$, provided $|x|<1$.

-
```
syms x k
s1 = symsum(1/k^2, 1, inf)
s2 = symsum(x^k, k, 0, inf)
s1 =
    pi^2/6
s2 =
   piecewise([1 <= x, Inf], [abs(x) < 1, -1/(x -1)])
```


TAYLOR SERIES

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

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

- which is all the terms up to, but not including, order eight in the Taylor series for f (x):

$$\sum_{n=0}^{\infty}(x-a)^n\frac{f^{(n)}(a)}{n!}$$

TAYLOR SERIES

These commands

```
syms x
g = exp(x*sin(x))
t = taylor(g, 12, 2);
```

- generate the first 12 nonzero terms of the Taylor series for g about x = 2.

SINGLE DIFFERENTIAL EQUATION

- The function `dsolve` computes symbolic solutions to ordinary differential equations.
- The equations are specified by symbolic expressions containing the letter ***D*** to denote differentiation.
- The symbols *D2, D3, ... DN*, correspond to the second, third, ... , *Nth* derivative, respectively.
- Thus, ***D2y*** is the toolbox equivalent of d^2y/dt^2.

SINGLE DIFFERENTIAL EQUATION

- The dependent variables are those preceded by *D* and the default independent variable is *t*.
- Note that names of symbolic variables should not contain *D*.
- The independent variable can be changed from *t* to some other symbolic variable by including that variable as the last input argument.
- Initial conditions can be specified by additional equations. If initial conditions are not specified, the solutions contain constants of integration, *C1, C2,* etc.

```
y = dsolve('Dy = t*y', 'y(0) = 2')
y =
    2*exp(t^2/2)
```


SINGLE DIFFERENTIAL EQUATION

- Nonlinear equations may have multiple solutions, even when initial conditions are given:

```
x = dsolve('(Dx + x)^2 = 1', 'x(0) = 0')
```

results in;

```
x =
1/exp(t) - 1
1 - 1/exp(t)
```


SINGLE DIFFERENTIAL EQUATION

- Here is a second-order differential equation with two initial conditions, and the default independent variable changed to *x*. The commands

```
y = dsolve('D2y = cos(2*x) - y', 'y(0) = 1',
'Dy(0) = 0', 'x');
simplify(y)
```

produce

```
ans =
   1 - (8*(cos(x)/2 - 1/2)^2)/3
```

SINGLE DIFFERENTIAL EQUATION

- The key issues in this example are the order of the equation and the initial conditions. To solve the ordinary diferential equation

$$\frac{d^3u}{dx^3}$$

$u(0) = 1, \quad u'(0) = -1, \quad u''(0) = \pi,$

- with x as the independent variable, type

```
u = dsolve('D3u = u', 'u(0) = 1', 'Du(0) = -1', 'D2u(0) = pi', 'x')
```

```
u = (pi*exp(x))/3 - (cos((3^(1/2)*x)/2)*(pi/3 -1))/exp(x/2) . . .
      - (3^(1/2)*sin((3^(1/2)*x)/2)*(pi + 1))/(3*exp(x/2))
```


SEVERAL DIFFERENTIAL EQUATIONS

- The function `dsolve` can also handle several ordinary differential equations in several variables, with or without initial conditions. For example, here is a pair of linear, first-order equations:

- `S = dsolve('Df = 3*f + 4*g', 'Dg = -4*f + 3*g')`

SEVERAL DIFFERENTIAL EQUATIONS

- The computed solutions are returned in the structure *S*. You can determine the values of *f* and *g* by typing;

```
f = S.f
g = S.g

f =
C2*cos(4*t)*exp(3*t) + C1*sin(4*t)*exp(3*t)
g =
C1*cos(4*t)*exp(3*t) - C2*sin(4*t)*exp(3*t)
```


SEVERAL DIFFERENTIAL EQUATIONS

- If you prefer to recover *f* and *g* directly as well as include initial conditions, type;

```
[f, g] = dsolve('Df= 3*f+4*g, Dg= -4*f+3*g', . .
'f(0) = 0, g(0) = 1')

f =
sin(4*t)*exp(3*t)
g =
cos(4*t)*exp(3*t)
```

References for Week 11

1 Symbolic Math Toolbox User's Guide, The Mathworks Inc., 2010.

Have a nice Week End