Introduction to MATLAB

Symbolic Math

MATLAB's Symbolic Toolbox Allows You To:

- Enter expressions or equations in symbolic form with symbolic data types
- Expand or simplify symbolic expressions
- Find symbolic roots, limits, minima, maxima, etc.
- Differentiate and integrate symbolic functions
- Generate Taylor series of functions (among other tools)
- Solve algebraic and differential equations symbolically
- Solve simultaneous sets of equations symbolically (even some nonlinear ones)
- Plot 2D and 3D symbolic expressions or functions

Symbolic Objects

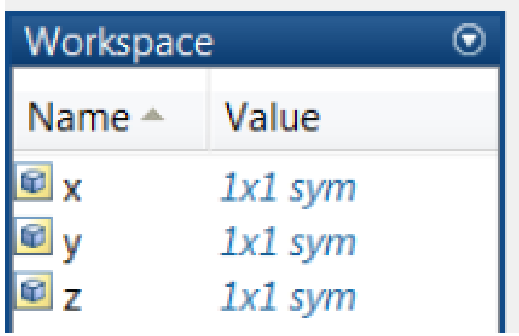
- A symbolic object is a data structure that stores a string representation of a symbolic expression or equation.
- A symbolic object is used to represent symbolic variables.
- A symbolic object is created using **`syms`** and/or **`sym( )`** commands.
- A symbolic object can be converted to a number object using the **`double( )`** function (provided it doesn't have any symbolic variables in it).
- A symbolic object can be converted to a string object using the **`char( )`** function.

Creating Symbolic Variables

- In order to use the symbolic toolbox, you must create symbolic variables or expressions.
- Use the **syms** command to this:

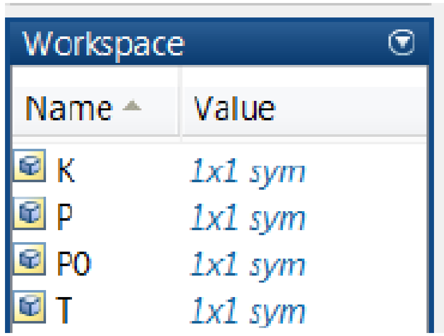
```
>> syms x      %Create x as symbolic variable
>> syms x y z %Create multiple symbolic variables
```

- Notice the variables of type *1x1 sym* created in the workspace after these commands are executed



Creating a Symbolic Expression with Symbolic Variables

- After symbolic variables have been defined/declared:
  `>> syms K T P0`
- They can be used to create a symbolic expression:
  `>> P = P0*exp(K*T)`
- This creates a symbolic expression that includes the exponential function. It could represent the exponential growth of a population.
- Note: A symbolic expression is NOT an equation (i.e. there is no equal "==" sign). When a symbolic expression is used, MATLAB often views expressions as an equation by assuming that the expression equals 0.
  For example, `P0*exp(K*T)== 0`



Creating Symbolic Equations

- It is possible to write equations with the symbolic toolbox

```
>> syms P V n R T
>> ideal_gas_law = (P*V == n*R*T)
```

- MATLAB returns:

```
    ideal_gas_law =
      P*V == R*T*n
```

- Note: As with logical comparison statements, the "**==**" is just MATLAB saying the two sides are equivalent. Remember, a single "**=**" is the assignment operator so it won't work in this case.

Difference Between Symbolic and Standard Numbers

Command Window

```
>> sqrt(2)

ans =

    1.4142

>> sqrt(sym(2))

ans =

2^(1/2)
```

Command Window

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

a =

2^(1/2)


>> double(a)

ans =

    1.4142
```

Command Window

```
>> a = sym(3)/sym(10)

a =

3/10


>> b = 3/10

b =

    0.3000
```

Workspace

Stack

Name	Value
a	<1x1 sym>
b	0.3

Useful Symbolic Functions

Function	Description
`solve( )`	solve equation(s)
`subs( )`	replace a symbolic variable with a numeric value or another symbolic variable
`double( )`	convert symbolic number to an actual number
`char( )`	convert symbolic expression/equation to a string
`vpa( )`	reformat symbolic fraction numbers (common and often unwieldy) to symbolic decimal numbers having a specified significant figures
`poly2sym( )`	create symbolic polynomial from array of coefficients
`pretty( )`	makes equation pretty (ASCII art)
`ezplot( )`	plots symbolic equation
`ezsurf( )`	surface plot of a symbolic equation
`symsum( )`	evaluates the sum of a series
`diff( )`	differentiates an equation*
`limit( )`	finds the limit of an equation*
`int( )`	integrates an equation*

* These also have different, non-symbolic uses so be careful when looking them up in the help files

Functions for Manipulating Symbolic Expressions

Function	Description
`numden( )`	separate the numerator and denominator of a quotient
`expand( )`	expand the products of factors in an expression
`factor( )`	factor an expression into a product of terms.*
`collect( )`	collect coefficients based on the specified variable
`simplify( )`	find a simpler form of the equation

* These also have different, non-symbolic uses so be careful when looking them up in the help files

Symbolic Solving with `solve( )`

- For expressions, the **`solve( )`** function sets an expression equal to zero and then solves the resulting equation(s) for its roots.
```
>> syms x %declare x as symbolic
>> ex1 = x^2 - 9 %expression
>> solve(ex1,x)%solve ex1 for x
```
- For equations, the **`solve( )`** function solves them as entered
```
>> eq1 = (x^2 - 9 == 0)%equation
>> solve(eq1,x)%solve eq1 for x
```
- Both methods yield in the same result:
```
    ans = 3
         -3
```

`solve( )` Example

- Solving an equation symbolically

```
>> syms a b c x
>> solve(a*x^2 + b*x + c == 0,x)
ans = -(b+(b^2-4*a*c)^(1/2))/(2*a)
      -(b-(b^2-4*a*c)^(1/2))/(2*a)
>> pretty(ans)
```

```
/              2          \
|   b + sqrt(b  - 4 a c)  |
| - --------------------  |
|           2 a           |
|                         |
|              2          |
|   b - sqrt(b  - 4 a c)  |
| - --------------------  |
|           2 a           |
\                         /
```

- Solving for a variable besides x

```
>> solve(a*x^2 + b*x + c,a)
ans = -(c + b*x)/x^2
```

Solving Systems of Equations

- You can use **`solve( )`** to find the solution of a system of equations

```
>> syms x y z
>> eq1 = (11*x + 25*y - z == 10);
>> eq2 = (-x + 61*y + 2*z == 5);
>> eq3 = (x - y - z == -1);
>> [x y z] = solve([eq1,eq2,eq3],[x y z])

  ans =
           x = 571/564    y = 19/564    z = 93/47
```

- The **`solve( )`** function produces symbolic output. You can convert the output to numerical values with the **`double( )`** command.

```
>> double(x)
ans = 1.0124
```

Replacing a Variable with a Number with **`subs( )`**

- Example:

```
>> syms x y z %declare all symbolic variable
>> f = 2*x + y^2 + z; %create symbolic expression

>> subs(f,x,4) %replace x with 4
```
Yields: `ans = 8 + y^2 + z`
```
>> subs(f,y,2) %replace y with 2
```
Yields: `ans = 2*x + 4 + z`

- Notes:
  - The original expression, **`f`**, is unchanged so each **`subs( )`** command is unrelated and ONLY one number is substituted each time.
  - If a symbolic variable is not specified, MATLAB chooses the letter closest to x in the alphabet.
  - The term order may be different than listed in the example

Replacing a Variable with Another Variable or Expression

- The **subs( )** command also allows you to replace a symbolic variable with another symbolic variable (e.g. **y**), or symbolic expression (e.g. **y^4+z**, **sin(y)** , etc.)
- Example:

```
>> syms a b c x y;
>> f = a*x^2 + b*x + c;
>> yf = subs(f,x,sin(y)) %replace x
with sin(y)
```

Yields:

```
   yf = a*sin(y)^2 + b*sin(y) + c
```

Multiple Variable Substitutions (Method 1)

- Often substituting for multiple variables is needed. This method substitutes/replaces one variable at a time using **subs( )** and overwrites the original expression (**f** for this example).

```
>> syms a b c x %declare all symbolic variable
>> f = a*x^2 + b*x + c; %original expression

>> f = subs(f,x,4) %replace x with 4
Yields:   f = 16*a + 4*b + c          %updated f
>> f = subs(f,a,1) %replace a with 1
Yields:   f = 4*b + c + 16            %updated f
>> f = subs(f,b,2) %replace b with 2
Yields:   f = c + 24                  %updated f
>> f = subs(f,c,3) %replace c with 3
Yields:   f = 27                      %updated f
```

Multiple Variable Substitutions (Method 2)

- **subs( )** can also be used to do multiple substitutions in one command. This is done by grouping the variables and their substitutes (other variables, symbolic expressions or numerical values) in brackets:

  subs(sym_exp,[substitutant],[substitutes])

- Example:

```
>> syms a b c x
>> f = a*x^2 + b*x + c;
>> f = subs(f,[a b c x],[1 2 3 4]) %a=1,b=2,c=3,x=4
```

- Yields:

```
f = 27
```

- Notes:
  - All equation variables/expressions must be symbolic
  - The example above overwrites the original **f**. If you want to keep the original expression, assign the **subs( )** output to a different variable.
  - Use **double( )** to convert **f** from a symbolic number to a standard MATLAB number

Converting and Reformatting Symbolic Expressions

- **`double( )`**
  - Convert symbolic number to an actual number
  - Symbolic input **CANNOT** contain any variables
  - See solving systems of equations slides for example
- **`char( )`**
  - Convert symbolic expression/equation to a string
  - Often used to insert symbolic expression as a string into a title, legend, etc.
  - See 2D **`ezplot( )`** slide for example
- **`vpa( )`**
  - Reformat symbolic fraction numbers (common and often unwieldy) to decimal numbers having a specified significant figures.
  - Symbolic input **CAN** contain both numbers AND/OR variables
  - Returned object is still symbolic.
  - E.g. **`f = (22219*x)/1000`** → **`f = 22.21*x`**
  - See 2D **`ezplot( )`** slide for example

Creating Polynomial with **poly2sym( )**

- The **poly2sym( )** function uses an array of coefficients to create a polynomial:

```
>> coeff = [4 -1 3 2]; %matrix of coefficients
>> b = poly2sym(coeff); %create symbolic polynomial from coeff

b = 4*x^3 - x^2 + 3*x + 2
```

- The function **sym2poly( )** is the inverse of **poly2sym( )**

```
>> sym2poly(b)
ans = 4 -1 3 2
```

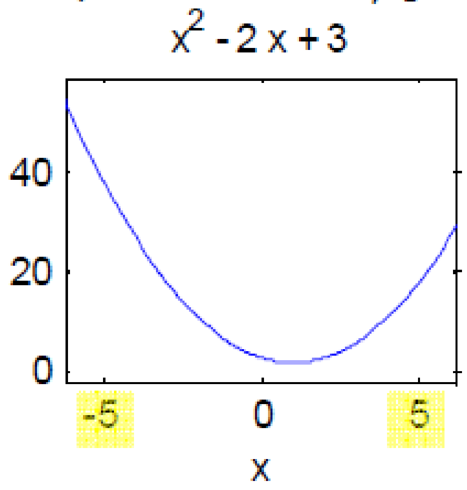
Plotting Symbolic Expressions

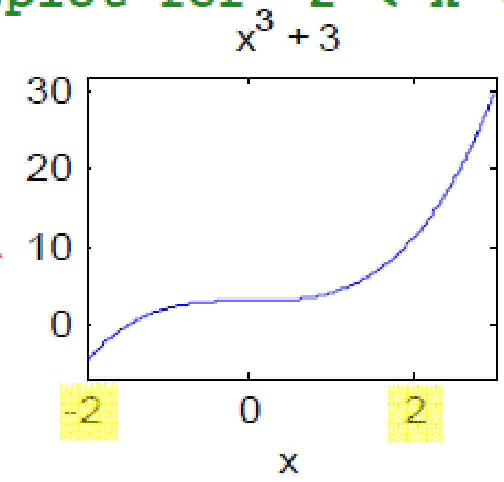
- Symbolic expressions can be plotted without having to generate x or y data points.
- This is often the fastest way to plot an equation in MATLAB
- **`ezplot( )`** is used as follows:
  - **`ezplot(f);`**
  - **`ezplot(f,[xmin xmax]);`**
  - **`ezplot(f, [xmin xmax ymin ymax]);`**
  - **`ezplot(…,figure);`**
- Other useful symbolic based plotting functions
  - **`ezplot3( )`** / **`ezsurf( )`** / **`ezmesh( )`** / etc.

2D Plots of Symbolic Expressions

- Plotting 2D symbolic expressions in MATLAB is done with the **`ezplot( )`** command.
- Example:

```
syms x;  y = x^2 - 2*x +3;
ezplot(y) %plot y for -2*pi < x < 2*pi (default)
ezplot('x^3 + 3',[-2 3]) %plot for -2 < x < 3
```





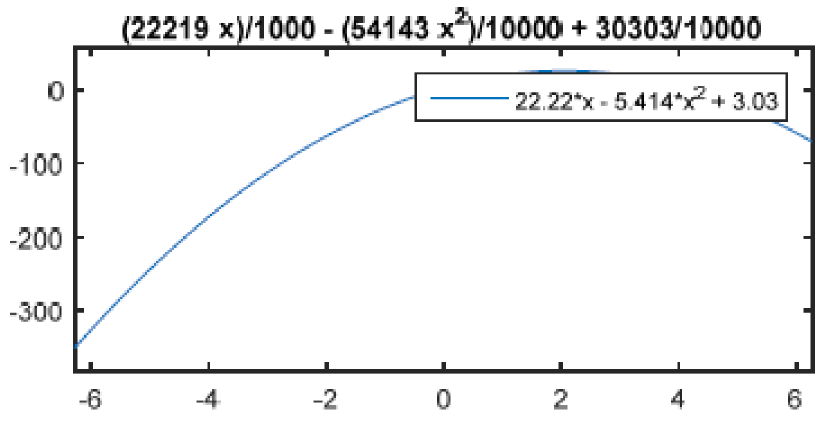
be careful when looking them up in the help files

Example using **ezplot( )**, **vpa( )**, and **char( )**

```
coeff = [-5.4143 22.219 3.0303]%polynomial coefficients
eqPoly = poly2sym(coeff) %conv. to symbolic expression
ezplot(eqPoly,[-6 6])    %plot polynomial
eq4 = vpa(eqPoly,4)      %conv. fractions to 4 sig fig
eqString = char(eq4)     %conv. eq4 from sym to string
legend(eqString)         %use string for legend
```

Notes:

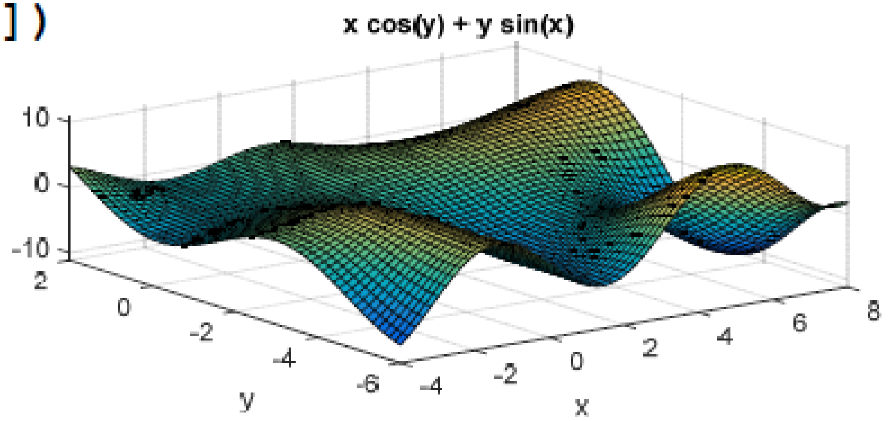
- The title contains default format with numbers as often unwieldy fractions, while the legend has the more compact format generated using **vpa( )** and **char( )**.
- Plots can be manipulated just like standard plots. E.g. **hold on**, **title( )**, **xlabel( )**, **text( )**, etc.



3D Surface Plots of Symbolic Expressions

- Creating a surface plot of a 3D symbolic functions is done using **`ezsurf( )`**.
- The syntax when z is a function of x and y is:

```
>> syms x y
>> z = y*sin(x) + x*cos(y);
>> ezsurf(z,[-4 8 -6 2])
```



- Notice how the x and y axis limits are defined similar to the **`axis( )`** function

Symbolic Summations with `symsum( )`

- `symsum( )` is a symbolic function to do summations ($\sum$)
- Usage:
  ```
  symsum(f, n, a, b); % sum wrt f from n=a to n=b
  ```
- Example $\sum_0^2 \frac{(-1)^n}{x^n}$:
  ```
  >> syms x n
  >> f = (-1)^n/(x^n);
  >> symsum(f,n,0,2)
  ans = 1/x^2 - 1/x + 1
  ```

Symbolic Derivatives with `diff( )`

Mathematical Operator	MATLAB Command
$\frac{df}{dx}$	`diff(f) or diff(f,x)`
$\frac{df}{da}$	`diff(f,a)`
$\frac{d^2 f}{db^2}$	`diff(f,b,2)`

Examples

`f`	`diff(f,x)`
`x^n`	`n*x^(n-1)`
`sin(a*x + b)`	`a*cos(b + a*x)`
`exp(a*x)`	`a*exp(a*x)`

Note: Make sure all variables are declared using syms beforehand

Symbolic Limits with `limit( )`

Mathematical Operator	MATLAB Command
$\lim_{x \to 0} f(x)$	`limit(f,x)`
$\lim_{x \to a} f(x)$	`limit(f,x,a)`
$\lim_{x \to a^-} f(x)$	`limit(f,x,a,'left')`
$\lim_{x \to a^+} f(x)$	`limit(f,x,a,'right')`

Note: Make sure all variables are declared using syms beforehand

- Example:
```
>> syms x
>> f = sin(x)/x;
>> limit(f,x,0)
```
- Yields:
```
ans = 1
```

Symbolic Integrals with `int( )`

Mathematical Operator	MATLAB Command
$\int f(x)\mathrm{dx}$	**`int(f,x)`**
$\int_a^b f(x)\mathrm{dx}$	**`int(f,x,a,b)`**

Examples

f	int(f,x)
`x^n`	`x^(n+1)/n+1`
`1/x`	`log(x)`
`1/(1+x^2)`	`atan(x)`

f	a, b	int(f,x,a,b)
`x^7`	`0, 1`	`1/8`
`1/x`	`1, 2`	`log(2)`
`exp(-x^2)`	`0, inf`	`1/2*pi^(1/2)`

Note: Make sure all variables are declared using `syms` beforehand

Numerator and Denominators with **numden( )**

- The **numden( )** command is used to separate the numerator and denominator of a quotient.
- Example:

```
>> syms x
>> y = 2*(x + 3)^2/(x^2 + 6*x + 15);
>> pretty(y) %make y pretty
>> [numerator, denominator] = numden(y)
```

- Yields:

```
numerator = 2*(x +3)^2
denominator = x^2 + 6*x + 15
```

- If the equation can be reduced to a simpler form, MATLAB will reduce it – e.g. try it with 9 instead of 15

Manipulating expressions with **`expand( )`**

- The **`expand( )`** function is used to expand an expression by expanding the products of factors in an expression.
- Example:

```
>> syms x
>> expand(2*(x +3)^2)
ans = 2*x^2 + 12*x + 18
```

`expand( )` Examples

f	expand(f)
`a*(x+y)`	`a*x + a*y`
`(x-1)*(x-2)*(x-3)`	`x^3 - 6*x^2 + 11*x -6`
`x*(x*(x-6)+11)-6`	`x^3 - 6*x^2 + 11*x -6`
`exp(a+b)`	`exp(a)*exp(b)`
`cos(x+y)`	`cos(x)*cos(y) - sin(x)*sin(y)`
`cos(3*acos(x))`	`4*x^3 - 3*x`

Note: Make sure all variables are declared using syms beforehand

Manipulating expression with `factor( )`

- The **`factor( )`** function is used to factor an expression into a product of terms.
- Example:

```
>> syms x
>> factor(x^2 + 6*x + 9,x)
ans = (x + 3)^2
```

Manipulating expression with `collect( )`

- This function is used to collect coefficients based on the specified variable.

f	collect(f,x)
`(x-1)*(x-2)*(x-3)`	`x^3 - 6*x^2 + 11*x - 6`
`x*(x*(x-6)+11)-6`	`x^3 - 6*x^2 + 11*x - 6`
`(1+x)*t + x*t`	`(2*t)*x + t`
`x^2*y + y*x - x^2 - 2*x`	`(y - 1)*x^2 + (y - 2)*x`

Note: Make sure all variables are declared using syms beforehand

Manipulating expression with `simplify( )`

- The **`simplify( )`** function uses the Maple simplification algorithm to simplify each part of an expression
- Example:

```
>> syms a b c
>> simplify(exp(c*log(sqrt(a+b))))
```

- Yields:

```
    ans = (a + b)^(c/2)
```