# Symbolic computations with MATLAB

**Georgios Georgiou**
*Department of Mathematics and Statistics*
*University of Cyprus*

Πανεπιστήμιο Κύπρου
Τμήμα Μαθηματικών
και Στατιστικής

# Symbolic variables

## The sym function

x=sym('x') creates symbolic variable x for x

a=sym('alpha') creates symbolic variable a for alpha

pi=sym('pi') creates symbolic variable pi for π

s=sym('sqrt(2)') creates symbolic variable s for $\sqrt{2}$

e=sym('exp(1)') creates symbolic variable e for e

delta=sym(0.001) creates the symbolic number delta for 0.001=1/1000

# The syms function

**syms** creates symbolic variables (like **sym**) but does not display the results in the command window.

It is very convenient when we have to define a number of symbolic variables.

For example

```
>> syms alpha beta gamma
```

is equivalent to

```
>> alpha=sym('alpha'); beta=sym('beta'); gamma=sym('gamma');
```

# The sym function

A symbolic variable may correspond to a **symbolic expression** like

```
>> b=sym('2*k*pi*x/R')
b =
2*k*pi*x/R
```

or a **symbolic function** like

```
>> y=sym('3*x^2*sin(x)')
y =
3*x^2*sin(x)
```

or a **symbolic equation** like

```
>> Elaw=sym('E = m*c^2')
Elaw =
E=m*c^2
```

# The sym function

**sym can be called with a second input argument defining the type of the symbolic variable:**

| Command | Result |
|---|---|
| x=sym(‘x’,’real’) | x is real |
| x=sym(‘x’,’positive’) | x is positive |
| x=sym(‘x’,’unreal’) | x is a formal variable |

```
>> syms x y z positive
```

**Is equivalent**

```
>> x=sym('x','positive'); y=sym('y','positive') z=sym('z','positive');
```

# Example 1

```
>> syms x y
>>
>> f=x^2+x-5*x+6-4
f =
x^2 - 4*x + 2
>>
>>
>> g=(x+y)^2-x/y
g =
(x + y)^2 - x/y
>> pretty(g)

          2   x
  (x + y)  - -
              y
```

Simplification!

Pretty form!

# Example 2

```
>> syms a b
>> A=[a b; b a]
A =
[ a, b]
[ b, a]

>> det(A)
ans =
a^2 - b^2

>> eig(A)
ans =
 a + b
 a - b
>>
```

Determinant in symbolic form.

Eigenvalues

# The subs function

**subs substitutes the current value(s) of symbolic variable(s) in a given symbolic expression.**

**Example:** $\boxed{y = ae^{x} + be^{-x}}$

```
>> syms a b x
>> y=a*exp(x)+b*exp(-x)
y =
a*exp(x) + b/exp(x)
```

**If we assign a value to a, we can get the special form of y as follows:**

```
>> a=2; subs(y)
ans =
2*exp(x) + b/exp(x)
```

# The subs function

**If the prescribed variables are 2 or more subs is used as follows:**

```
subs(y, {a, b, …} {valueofa, valueofb, ... })
```

**or equivalently**

```
subs(y, {a,b, ….} [valueofa, valueofb,  ..])
```

**where the 2nd and the 3rd input argument are cell arrays.**

**So, in our example we can type:**

```
>> subs(y,{a,b},{2,-1})
ans =
2*exp(x) - 1/exp(x)
>>
>> subs(y,{a,b},{3,2})
ans =
2/exp(x) + 3*exp(x)
```

# findsym

**findsym(S) returns the symbolic variables of a symbolic expression in lexicographic order. The constants i, j, and pi are not variables.**

**findsym(S,n) returns the first n variables closer to x or X.**

1)

```
>> findsym( a*x+b *z +c *w)
ans =
a,b,c,w,x,z
>> findsym( a*x+b *z +c *w, 4)
ans =
x,w,z,c
```

**2)**

```
>> findsym( (x^2-a^2)/(x-a) )
ans =
a,x
>> findsym( (sin(x+h)-sin(x))/h )
ans =
h,x
>> findsym( pi* i + x* j +y)
ans =
x,y
```

# help symbolic

We can see a complete list of all the symbolic functions with

```
>> help symbolic
```

Some names correspond to both non-symbolic and symbolic functions. Such an example is diff:

With

```
>> help diff
```

we obtain help for the non-symbolic function, whereas with

```
>> help sym/diff
```

we get help for the symbolic function.

# simplify

**simplify(S)** tries (with a lot of success!) to simplify symbolic expression **S** in 1 step (default).

**simplify(S,N)** tries to simplify **S** in **N** steps.

## Example 1

```
>> simplify( exp(log(x^2)) )
ans =
x^2
>>
>> simplify( cos(x)^2 + sin(x)^2 )
ans =
1
>> simplify( (x+1)*x*(x-1) )
ans =
x^3 – x
```

# Example 2

**We will show that**

$$\begin{vmatrix} a & b+c & a^2 \\ b & a+c & b^2 \\ c & a+b & c^2 \end{vmatrix} = (a-b)(a-c)(b-c)(a+b+c)$$

```
>> syms a b c
>> A=[ a b+c a^2; b c+a b^2; c a+b c^2]
A =
[ a, b + c, a^2]
[ b, a + c, b^2]
[ c, a + b, c^2]
>> det(A)
ans =
a^3*b - a^3*c - a*b^3 + a*c^3 + b^3*c - b*c^3
>> simplify(ans)
ans =
(a - b)*(a - c)*(b - c)*(a + b + c)
```

# expand

**expand** expands polynomials, trigonometric, exponential and logarithmic functions.

**Examples**

**expand((x+1)^3)** returns **x^3 + 3*x^2 + 3*x + 1**

**expand(exp(x+y))** returns **exp(x)*exp(y)**

**expand(sin(x+y))** returns **cos(x)*sin(y) + cos(y)*sin(x)**

**expand(sinh(x+y))** returns **cosh(x)*sinh(y) + cosh(y)*sinh(x)**

# Example

Compute sin(a+b), cos(a+b) and tan(a+b):

```
>> syms a b
>> expand( sin(a+b) )
ans =
cos(a)*sin(b) + cos(b)*sin(a)
>>
>> expand( cos(a+b) )
ans =
cos(a)*cos(b) - sin(a)*sin(b)
>>
>> expand( tan(a+b) )
ans =
-(tan(a) + tan(b))/(tan(a)*tan(b) - 1)
>> pretty(ans)

    tan(a) + tan(b)
  - ----------------
    tan(a) tan(b) - 1
```

Pretty form!

# simple

**simple** searches for simplest form of a symbolic expression or matrix.

$$f = \tan^2\left(\frac{1}{2}\cos^{-1} x\right) x(x-1)(x+1)$$

```
>> syms x
>> f=tan( acos(x)/2 )^2* x* (x-1)* (x+1)
f =
x*tan(acos(x)/2)^2*(x - 1)*(x + 1)
>> simplify(f)
ans =
-x*(x - 1)^2
>> simple(f)
simplify:
-x*(x - 1)^2
radsimp:
…….
```

**Try with MATLAB to see the rest!**

# factor

**factor** factorizes a symbolic expression.

```
>> factor( x^3 - 6*x^2 + 11*x - 6 )
ans =
(x - 3)*(x - 1)*(x - 2)

>> factor(x^12-1)
ans =
(x - 1)*(x + 1)*(x^2 + x + 1)*(x^2 + 1)*(x^2 - x + 1)*(x^4 - x^2 + 1)
>> pretty(ans)

                 2          2       2          4    2
 (x - 1) (x + 1) (x  + x + 1) (x  + 1) (x  - x + 1) (x  - x  + 1)
```

# collect

**collect(S,v)** collects all terms of **S** as a polynomial of **v**.

**collect(S)** collects all terms of **S** as a polynomial of the variable determined by **findsym** (e.g. **x**)

# Example

$$S = x^2 - 4x^2 \cos y + 2x \cos^2 y + 3\cos^2 y$$

```
>> syms x y
>> S=x^2-4*x^2*cos(y)+2*x-x*(cos(y))^2+3*(cos(y))^2
S =
2*x - x*cos(y)^2 - 4*x^2*cos(y) + 3*cos(y)^2 + x^2
>>
>> collect(S)
ans =
(1 - 4*cos(y))*x^2 + (2 - cos(y)^2)*x + 3*cos(y)^2
>>
>> collect(S,cos(y))
ans =
(3 - x)*cos(y)^2 + (-4*x^2)*cos(y) + x^2 + 2*x
```

# limit

```
limit(F)
```

$$\lim_{y \to 0} F$$

y=findsym(F)

```
limit(F,a)
```

$$\lim_{y \to a} F$$

```
limit(F,x,a)
```

$$\lim_{x \to a} F$$

```
limit(F,x,a,'left')
```

$$\lim_{x \to a^-} F$$

```
limit(F,x,a,'right')
```

$$\lim_{x \to a^+} F$$

# Example

$$\lim_{x\to\infty}\frac{x^2+t}{2x^2-2x+t^2}$$

```
>> limit( (x^2+t)/(2*x^2-2*x+t^2), inf )
ans =
1/2
```

$$\lim_{t\to-\infty}\frac{x^2+t}{2x^2-2x+t^2}$$

```
>> limit( (x^2+t)/(2*x^2-2*x+t^2), t, -inf )
ans =
0
```

# The diff function

diff(S) finds the derivative with respect to findsym(S)

```
>> syms x t
>> diff(4*x^3)
ans =
12*x^2
>> diff( cos(2*x) )
ans =
-2*sin(2*x)
>>
>> int( t^6 )
ans =
1/7*t^7

>> diff(x^a,a)
ans =
x^a*log(x)
```

# The diff function

**diff(S,n,x)** or **diff(S,n,x)** finds $\boxed{\dfrac{d^n S}{dx^n}}$

## Example

**Find the first 4 derivatives of**

$$\boxed{y = \log(ax + b)}$$

```
>> syms a b x
>> for i=1:4
diff( log(a+b*x), i)
end
ans =
b/(a + b*x)
ans =
-b^2/(a + b*x)^2
ans =
(2*b^3)/(a + b*x)^3
ans =
-(6*b^4)/(a + b*x)^4
```

# The int function

| expression | |
|---|---|
| `int(f)` or<br>`int(sym(f))` | $\int f dx$<br>όπου x=findsym(f) |
| `int(f,v)` | $\int f dv$ |
| `int(f,a,b)` | $\int_a^b f dx$<br>όπου x=findsym(f) |
| `int(f,v,a,b)` | $\int_a^b f dv$ |

# Example 1

```
>> syms a x
>> int( exp(a*x) )
ans =
exp(a*x)/a
>> pretty(ans)

  exp(a x)
  --------
     a
>>
>> int( x*exp(a*x) )
ans =
(exp(a*x)*(a*x - 1))/a^2
>> pretty(ans)

  exp(a x) (a x - 1)
  ------------------
          2
         a
```

$$\int e^{ax}dx = \frac{e^{ax}}{a}$$

$$\int xe^{ax}dx = \frac{(ax-1)e^{ax}}{a^2}$$

# Example 2

We will calculate

$\int x \sin ax \, dx,$ $\int \frac{x dx}{x^2 - a^2}$ and $\int x J_0(x) dx$

```
>> int( x*sin(a*x) )
ans =
(sin(a*x) - a*x*cos(a*x))/a^2
>> int( x/(x^2-a^2) )
ans =
log(x^2 - a^2)/2
>> int( x*besselj(0,x) )
ans =
x*besselj(1, x)
```

# The double function

**double returns the numerical value of a symbolic expression which may not be computed symbolically.**

**With**

```
>> int( log(1+tan(x)), 0, pi/4 )
ans =
(pi*log(2))/8
```

**we find that** $$\int_0^{\pi/4} \log(1+\tan x)\,dx = \frac{\pi}{8}\log 2$$

**The numerical value of the integral is found as follows:**

```
>> double(ans)
ans =
   0.272198261287950
```

# Example: Elliptic integral

```
>> syms x
>> int( sqrt(2-sin(x)^2) )
Warning: Explicit integral could not be found.
> In sym.int at 64
ans =
int((2 - sin(x)^2)^(1/2), x)

>> int( sqrt(2-sin(x)^2), 0, pi )
Warning: Explicit integral could not be found.
> In sym.int at 64
ans =
int((2 - sin(x)^2)^(1/2), x = 0..pi)

>> double(ans)
ans =
   3.820197789027712
```

$$\boxed{\int \sqrt{2-\sin^2 x}\,dx}$$

$$\boxed{\int_0^{\pi} \sqrt{2-\sin^2 x}\,dx}$$

**Numerical value**

# The solve function

**solve** solves either a single or a system of algebraic equations.

**solve('eqn')** : solve eqn wrt findsym(eqn)

**solve('eqn',var)** : solve eqn wrt var

```
>> solve(a*x+b)
ans =
-b/a

>> solve(a*x+b,a)
ans =
-b/x

>> solve('a*x+b=a+x')
ans =
(a - b)/(a - 1)
```

$$ax + b = 0$$

$$ax + b = a + x$$

Used ' ' for this equation!

# Example

$$\boxed{ax^2 + bx + c = 0}$$

```
>> syms a b c x
>> y=solve(a*x^2+b*x+c)
y =
 -(b + (b^2 - 4*a*c)^(1/2))/(2*a)
 -(b - (b^2 - 4*a*c)^(1/2))/(2*a)
```

To find the roots for given values of the parameters we use **subs**.

```
>> subs(y, {a,b,c}, [1,0,-1] )
ans =
   -1
    1
>> subs(y, {a,b,c}, [1,3,2] )
ans =
   -2
   -1
```

# Systems of algebraic equations

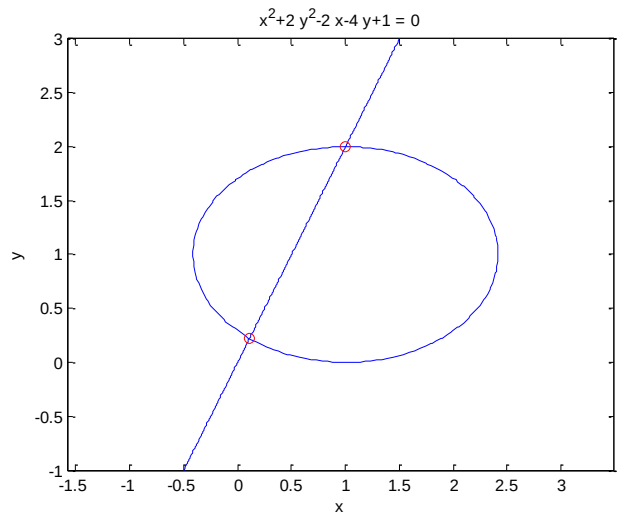
```
solve('eqn1','eqn2',...,'eqnN')

solve('eqn1','eqn2',...,'eqnN',var1,var2,...,varN)
```

# Example

**Find the common points of** $x^2+2y^2-2x+4y+1=0$ **and** $y=2x$

```
>> [x,y]=solve('x^2+2*y^2-2*x-4*y+1','y-2*x')
x =
   1
 1/9
y =
   2
 2/9
```

```
>> ezplot('y-2*x',[-1 3]); axis equal;
>> set(gcf,'Color','w'); hold on
>> ezplot('x^2+2*y^2-2*x-4*y+1')
>> plot(subs(x),subs(y),'ro')
```



# digit and vpa

When it is not possible for solve to find a symbolic solution, then it searches for a numerical solution with an accuracy of 32 digits. This is the default value that we can see using

```
>> digits
```

We can change the accuracy to n digits using

```
>> digits(n)
```

The function vpa (variable precision arithmetic) calculates the value of a symbolic expression with the accuracy specified by digit.
The result is a symbolic number.

# Example 1

**With the following command we get π and e with an accuracy of 32 digits!**

```
>> vpa(pi)
ans =
3.1415926535897932384626433832795
>> vpa(exp(1))
ans =
2.7182818284590455348848081484903
```

**We can change the accuracy by selecting the desired number D of accurate digits.**

```
>> vpa(S,D)
```

# Example 2

```
>> solve('sin(x)-x+0.1')
ans =
0.85375015664086577428139327460646
>> digits(48)
>> solve('sin(x)-x+0.1')
ans =
0.85375015664086577428139327460646152542415836906 1
```

***Thank you!!***