Table of Contents


```
close all; clearvars; clc
```

SYMBOLIC NUMBERS

>> for exact representations (unlike floating-point numbers), thus calculations on symbolic numbers are exact

```
Val = sym(pi);
      % assign exact value of pi
Exact = sin(2*Val)
      % returns 0 (exact)
Approx = sin(2*pi)
      % return -2.4e-16 (approximated)
fprintf(" ------- \n");
```

```
Exact =

0


Approx =

  -2.4493e-16

 -------
```

SYMBOLIC VARIABLES (pt.1)

>> to create variables with sequential name

```
syms(sym('k',[1 9]))
      % to create a matrix of 1x9 symbolic variables named k1, k2, k3, etc
k1
      % returns k1 (pure symbolic reference)
k2 = 60;
      % reassign variable (no more symbolic)
k2
      % returns 60
fprintf(" ------- \n");
```

```
k1 =
```

```
k1
```

```
k2 =

    60

 -------
```

EVAL FUNCTION

>> to evaluate code expressions, useful in loop cases to handle variables with sequential name (see section above)

```
eval(['k' num2str(5) ' = k' num2str(5) '+  412'])
      % returns "k5 = k5+412" (executes k5 = k5 + 412 through "eval")
eval(['k' num2str(2) ' = k' num2str(2) '+  25'])
      % returns "k2 = 85" (execute k2 = k2 + 25 through "eval")
fprintf(" ------- \n");
```

```
k5 =

k5 + 412
```

```
k2 =

    85

 -------
```

SYMBOLIC VARIABLES (pt.2)

>> to handle functions estimation/solution

```
syms a b c x
      % define symbolic variables
f(x) = a*x^2+b*x+c;
      % define function through symbolic variables
sol_x = solve(f == 0,x)
      % estimate x-solutions so that f=0 (assuming a,b,c as constants)
sol_a = solve(f == 0,a)
      % estimate a-solution so that f=0 (assuming x,b,c as constants)
f(2)
      % estimate f(x) value for x=2
```

```
sol_x =

-(b + (b^2 - 4*a*c)^(1/2))/(2*a)
-(b - (b^2 - 4*a*c)^(1/2))/(2*a)
```

```
sol_a =

-(c + b*x)/x^2

ans =

4*a + 2*b + c
```

Published with MATLAB® R2022a