```
% SYMBOLIC EXPRESSIONS AND NUMERICAL INTEGRATION IN MATLAB

%1)
help syms;

%2)
syms t;
func=sin(2*pi*t);

%3)
%{
Two variables are are stored in workspace with the name
func:- with value 1*1 sym
and
t:- with 1*1 sym
%}

%4)
help subs;

%5)
t=-1:0.01:1;
y=subs(func,t);
disp(y);

%6)
help int;
h1=int(func);
h2=int(func*func);
y1=subs(h1,1)-subs(h1,0);
y2=subs(h2,1)-subs(h2,0);
disp(y1);
disp(y2);

%FOURIER SERIES ANALYSIS
%1)
syms t;
w0=100*pi;
T0=2*pi/w0;
K=-5:5;

%2)
x1=cos(w0*t);
[Ak1,w1]=FourierAnalysis(x1,T0,K);

x2=sin(w0*t);
[Ak2,w2]=FourierAnalysis(x2,T0,K);

disp(Ak1);
disp(Ak2);

%3)
```

```
syms t;
t=-5:0.01:5;
xs=(abs(mod(t,1))<=0.25)+(mod(t,1)~=0).*(abs(mod(-t,1))<=0.25);
y=subs(xs);
plot(t,y);

%4)
syms t;
T0=1;
K=-10:10;
Xs=1;
[Ak,W]=FourierAnalysis2(Xs,T0,K);
disp(Ak);
disp(W);
%All the Ak's are real.

%5)
stem(W,abs(Ak));

%6)
%{
The output is verified to be correct, but only after some neccesary changes
to the FourierAnalysis() function because otherwise, since the function is
define as a piecewise function it can't be integrated using the int()
function. Hence FourierAnalysis2() is designed.
%}

%FOURIER SERIES SYNTHESIS EQUATION AND GIBB'S PHENOMENON

%1)
t1=-2:0.01:2;
subplot(231);
plot(t1,X(1));
subplot(232);
plot(t1,X(5));
subplot(233);
plot(t1,X(10));
subplot(234);
plot(t1,X(25));
subplot(235);
plot(t1,X(50));
subplot(111);

%2)
%{
Observation:-
a) From the plot we observe that the maximum overshoot is:-

                max(xN(t))              Percentage Overshoot
N=1             1.13662                 13.662%
N=5             1.09332                 9.332%
N=10            1.09116                 9.116%
N=25            1.08996                 8.996%
N=50            1.08956                 8.956%
```

```
b) For each value of N we observe that the amplitude of intersection
   point is 0.5.

c) For each value of N we observe that the time difference is 0.8.
%}

%3)
%{
N=1 and N=5 are not in conformity with Gibb's phenomenon as they are
highly deviated from a square wave. N=10 is somewhat in conformity as it
tends to be a square wave.N=25 and N=50 almost satisfies Gibb's
phenomenon as they are almost fully square waves.
%}



%FUNCTION FourierAnalysis()


function [Ak,w]=FourierAnalysis(x,T0,K)
    Ak=K;
    w=K;
    syms t;
    for i=1:length(K)
        k=K(i);
        w0=k*2*pi/T0;
        w(i)=w0;
        func=int(x*exp(-(1i)*w0*t))/T0;
        Ak(i)=subs(func,T0)-subs(func,0);
    end
end

function [Ak,w]=FourierAnalysis2(x,T0,K)
    Ak=K;
    w=K;
    syms t;
    for i=1:length(K)
        k=K(i);
        w0=k*2*pi/T0;
        w(i)=w0;
        func=int(x*exp(-(1i)*w0*t))/T0;
        Ak(i)=subs(func,T0/4)-subs(func,-T0/4);
    end
end

function x=X(N)
    syms t;
    xs=1;
    K=-N:N;
    [Ak,W]=FourierAnalysis2(xs,1,K);
    t1=-2:0.01:2;
    x=t1*0;
    for y=1:length(K)
```

```
        ak=Ak(y);
        w=W(y);
        x=x+ak*exp((1i)*w*t1);
    end
end
```

```
 SYMS   Short-cut for constructing symbolic variables.
    SYMS arg1 arg2 ...
    is short-hand notation for creating symbolic variables
       arg1 = sym('arg1');
       arg2 = sym('arg2'); ...
    or, if the argument has the form f(x1,x2,...), for
    creating symbolic variables
       x1 = sym('x1');
       x2 = sym('x2');
       ...
       f = symfun(sym('f(x1,x2,...)'), [x1, x2, ...]);
    The outputs are created in the current workspace.

    SYMS  ... ASSUMPTION
    additionally puts an assumption on the variables created.
    The ASSUMPTION can be 'real', 'rational', 'integer', or 'positive'.
    SYMS  ... clear
    clears any assumptions on the variables created, including those
    made with the ASSUME command.

    SYMS ... [nrows,ncols] matrix
    declares the variables symbolic matrices

    SYMS({symvar1, symfun1,  ...})
    is equal to the call SYMS 'symvar1' 'symfun1'  ...

    SYMS({symvar1, symfun1, ...}, ASSUMPTION)
    is equal to the call SYMS 'symvar1' 'symfun1'  ... ASSUMPTION

    SYMS([symvar1, symvar2,  ...])
    is equal to the call SYMS 'symvar1' 'symvar2'  ...

    SYMS([symvar1, symvar2, ...], ASSUMPTION)
    is equal to the call SYMS 'symvar1' 'symvar2'  ... ASSUMPTION

    Each input argument must begin with a letter and must contain only
    alphanumeric characters.

    S = SYMS returns a cell array containing the names of the symbolic
    variables in the workspace. Without an output argument, SYMS lists
    the symbolic variables in the workspace.

    Example 1:
       syms x beta real
    is equivalent to:
       x = sym('x','real');
       beta = sym('beta','real');
```

```
   To clear the symbolic objects x and beta of 'real' or 'positive'
   status, type
      syms x beta clear

   Example 2:
      syms x(t) a
   is equivalent to:
      a = sym('a');
      t = sym('t');
      x = symfun(sym('x(t)'), [t]);

   Example 3:
      syms({sym('u'), sym('v'), sym('w')})
   is equivalent to:
      syms u v w

   Example 4:
      syms({symfun('u(t)',sym('t')), symfun('v(t)',sym('t')),
symfun('w(t)',sym('t'))})
   is equivalent to:
      syms u(t) v(t) w(t)

   Example 5:
      syms({sym('u'), sym('v'), sym('w')}, 'real')
   is equivalent to:
      syms u v w real

   Example 6:
      syms([sym('u'), sym('v'), sym('w')])
   is equivalent to:
      syms u v w

   Example 7:
      Clear all symbolic variables in the workspace:
      syms u v w
      S = syms;
      cellfun(@clear, S);

   Example 8:
      Create a symbolic matrix variable
      syms A [3 4] matrix
   is equivalent to:
      A = symmatrix('A', [3 4]);

   See also SYM, SYMFUN, SYMMATRIX.
   Deprecated API:
   The 'unreal' keyword can be used instead of 'clear'.

   Documentation for syms
      doc syms


SUBS   Symbolic substitution.
   SUBS(S,OLD,NEW) replaces OLD with NEW in the symbolic expression S.
```

OLD is a symbolic variable, a string representing a variable name, or an expression. NEW is a symbolic or numeric variable or expression.

SUBS(S,VALUES), where VALUES is a STRUCT, replaces the symbolic variables in S which are field names in VALUES by the corresponding entries of the struct.

SUBS(S) replaces all the variables in the symbolic expression S with values obtained from the calling function, or the MATLAB workspace.

SUBS(S,NEW) replaces the free symbolic variable in S with NEW.

*If OLD and NEW are vectors or arrays of the same size, each element of OLD is replaced by the corresponding element of NEW. If S and OLD are scalars and NEW is an array or cell array, the scalars are expanded to produce an array result. If NEW is a cell array of numeric matrices, the substitutions are performed elementwise (i.e., subs(x*y,{x,y},{A,B}) returns A.*B when A and B are numeric).*

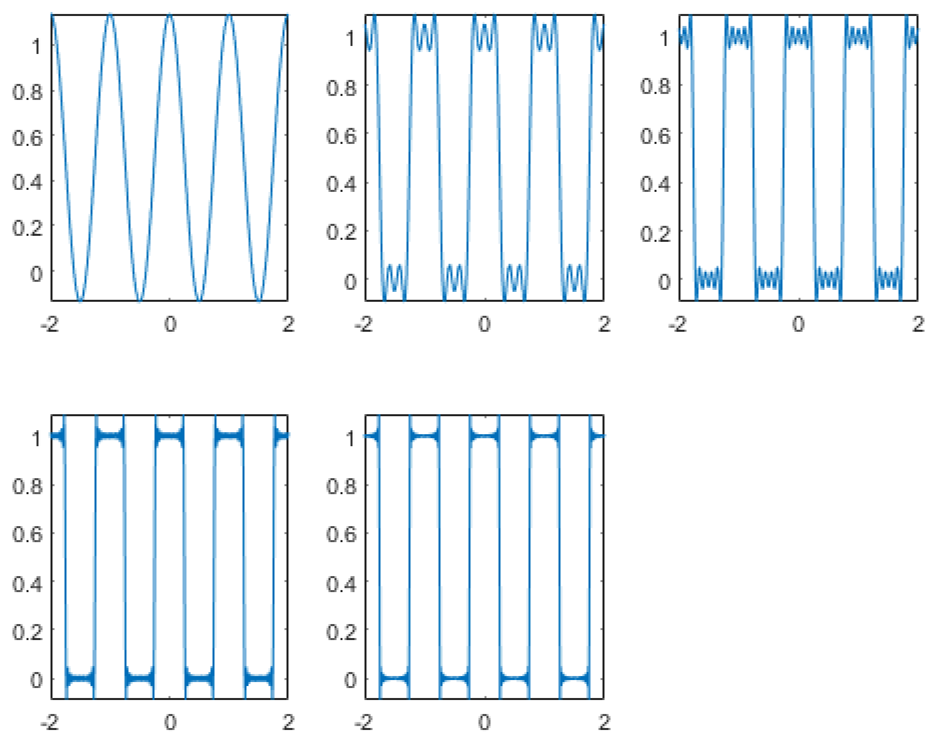
If SUBS(S,OLD,NEW) does not change S, then SUBS(S,NEW,OLD) is tried. This provides backwards compatibility with previous versions and eliminates the need to remember the order of the arguments. SUBS(S,OLD,NEW,0) does not switch the arguments if S does not change.

```
Examples:
  Single input:
    Suppose a = 980 and C1 = 3 exist in the workspace.
    The statement
       y = dsolve('Dy = -a*y')
    produces
       y = exp(-a*t)*C1
    Then the statement
       subs(y)
    produces
       ans = 3*exp(-980*t)

  Single Substitution:
    subs(a+b,a,4) returns 4+b.

  Multiple Substitutions:
    subs(cos(a)+sin(b),{a,b},[sym('alpha'),2]) or
    subs(cos(a)+sin(b),{a,b},{sym('alpha'),2}) returns
    cos(alpha)+sin(2)

  Scalar Expansion Case:
    subs(exp(a*t),'a',-magic(2)) returns

    [   exp(-t), exp(-3*t)]
    [ exp(-4*t), exp(-2*t)]

  Multiple Scalar Expansion:
    subs(x*y,{x,y},{[0 1;-1 0],[1 -1;-2 1]}) returns
    [  0, -1]
```

```
      [  2,  0]

    See also SYM/SUBEXPR, SYM/VPA, SYM/DOUBLE.

    Documentation for subs
       doc subs

    Other uses of subs

       sym/subs

[0, sin(pi/50), sin(pi/25), sin((3*pi)/50), sin((2*pi)/25), 5^(1/2)/4
 - 1/4, sin((3*pi)/25), sin((7*pi)/50), sin((4*pi)/25), sin((9*pi)/50),
 (2^(1/2)*(5 - 5^(1/2))^(1/2))/4, sin((11*pi)/50), sin((6*pi)/25),
 sin((13*pi)/50), sin((7*pi)/25), 5^(1/2)/4 + 1/4, sin((8*pi)/25),
 sin((17*pi)/50), sin((9*pi)/25), sin((19*pi)/50), (2^(1/2)*(5^(1/2)
 + 5)^(1/2))/4, sin((21*pi)/50), sin((11*pi)/25), sin((23*pi)/50),
 sin((12*pi)/25), 1, sin((12*pi)/25), sin((23*pi)/50), sin((11*pi)/25),
 sin((21*pi)/50), (2^(1/2)*(5^(1/2) + 5)^(1/2))/4, sin((19*pi)/50),
 sin((9*pi)/25), sin((17*pi)/50), sin((8*pi)/25), 5^(1/2)/4 + 1/4,
 sin((7*pi)/25), sin((13*pi)/50), sin((6*pi)/25), sin((11*pi)/50), (2^(1/2)*(5
 - 5^(1/2))^(1/2))/4, sin((9*pi)/50), sin((4*pi)/25), sin((7*pi)/50),
 sin((3*pi)/25), 5^(1/2)/4 - 1/4, sin((2*pi)/25), sin((3*pi)/50),
 sin(pi/25), sin(pi/50), 0, -sin(pi/50), -sin(pi/25), -sin((3*pi)/50),
 -sin((2*pi)/25), 1/4 - 5^(1/2)/4, -sin((3*pi)/25), -sin((7*pi)/50),
 -sin((4*pi)/25), -sin((9*pi)/50), -(2^(1/2)*(5 - 5^(1/2))^(1/2))/4, -
sin((11*pi)/50), -sin((6*pi)/25), -sin((13*pi)/50), -sin((7*pi)/25), -
 5^(1/2)/4 - 1/4, -sin((8*pi)/25), -sin((17*pi)/50), -sin((9*pi)/25), -
sin((19*pi)/50), -(2^(1/2)*(5^(1/2) + 5)^(1/2))/4, -sin((21*pi)/50), -
sin((11*pi)/25), -sin((23*pi)/50), -sin((12*pi)/25), -1, -sin((12*pi)/25),
 -sin((23*pi)/50), -sin((11*pi)/25), -sin((21*pi)/50), -(2^(1/2)*(5^(1/2)
 + 5)^(1/2))/4, -sin((19*pi)/50), -sin((9*pi)/25), -sin((17*pi)/50), -
sin((8*pi)/25), - 5^(1/2)/4 - 1/4, -sin((7*pi)/25), -sin((13*pi)/50),
 -sin((6*pi)/25), -sin((11*pi)/50), -(2^(1/2)*(5 - 5^(1/2))^(1/2))/4, -
sin((9*pi)/50), -sin((4*pi)/25), -sin((7*pi)/50), -sin((3*pi)/25), 1/4 -
 5^(1/2)/4, -sin((2*pi)/25), -sin((3*pi)/50), -sin(pi/25), -sin(pi/50),
 0, sin(pi/50), sin(pi/25), sin((3*pi)/50), sin((2*pi)/25), 5^(1/2)/4
 - 1/4, sin((3*pi)/25), sin((7*pi)/50), sin((4*pi)/25), sin((9*pi)/50),
 (2^(1/2)*(5 - 5^(1/2))^(1/2))/4, sin((11*pi)/50), sin((6*pi)/25),
 sin((13*pi)/50), sin((7*pi)/25), 5^(1/2)/4 + 1/4, sin((8*pi)/25),
 sin((17*pi)/50), sin((9*pi)/25), sin((19*pi)/50), (2^(1/2)*(5^(1/2)
 + 5)^(1/2))/4, sin((21*pi)/50), sin((11*pi)/25), sin((23*pi)/50),
 sin((12*pi)/25), 1, sin((12*pi)/25), sin((23*pi)/50), sin((11*pi)/25),
 sin((21*pi)/50), (2^(1/2)*(5^(1/2) + 5)^(1/2))/4, sin((19*pi)/50),
 sin((9*pi)/25), sin((17*pi)/50), sin((8*pi)/25), 5^(1/2)/4 + 1/4,
 sin((7*pi)/25), sin((13*pi)/50), sin((6*pi)/25), sin((11*pi)/50), (2^(1/2)*(5
 - 5^(1/2))^(1/2))/4, sin((9*pi)/50), sin((4*pi)/25), sin((7*pi)/50),
 sin((3*pi)/25), 5^(1/2)/4 - 1/4, sin((2*pi)/25), sin((3*pi)/50), sin(pi/25),
 sin(pi/50), 0, -sin(pi/50), -sin(pi/25), -sin((3*pi)/50), -sin((2*pi)/25),
 1/4 - 5^(1/2)/4, -sin((3*pi)/25), -sin((7*pi)/50), -sin((4*pi)/25), -
sin((9*pi)/50), -(2^(1/2)*(5 - 5^(1/2))^(1/2))/4, -sin((11*pi)/50), -
sin((6*pi)/25), -sin((13*pi)/50), -sin((7*pi)/25), - 5^(1/2)/4 - 1/4, -
sin((8*pi)/25), -sin((17*pi)/50), -sin((9*pi)/25), -sin((19*pi)/50), -
(2^(1/2)*(5^(1/2) + 5)^(1/2))/4, -sin((21*pi)/50), -sin((11*pi)/25), -
```

```
sin((23*pi)/50), -sin((12*pi)/25), -1, -sin((12*pi)/25), -sin((23*pi)/50),
 -sin((11*pi)/25), -sin((21*pi)/50), -(2^(1/2)*(5^(1/2) + 5)^(1/2))/4,
 -sin((19*pi)/50), -sin((9*pi)/25), -sin((17*pi)/50), -sin((8*pi)/25),
 - 5^(1/2)/4 - 1/4, -sin((7*pi)/25), -sin((13*pi)/50), -sin((6*pi)/25),
 -sin((11*pi)/50), -(2^(1/2)*(5 - 5^(1/2))^(1/2))/4, -sin((9*pi)/50),
 -sin((4*pi)/25), -sin((7*pi)/50), -sin((3*pi)/25), 1/4 - 5^(1/2)/4, -
sin((2*pi)/25), -sin((3*pi)/50), -sin(pi/25), -sin(pi/50), 0]

 int - States from CIC filter
    This MATLAB function returns the states of a CIC filter in matrix form,
    rather than as the native filtstates object.

    Syntax
      integerstates = int(hm.states)

    See also filtstates.cic, dsp.CICDecimator, dsp.CICInterpolator

    Introduced in DSP System Toolbox in R2011a
    Documentation for int
       doc int

    Other uses of int

       filtstates/int    sym/int

0

1/2

  Columns 1 through 7

         0         0         0         0    0.5000         0    0.5000

  Columns 8 through 11

         0         0         0         0

  Columns 1 through 4

   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i

  Columns 5 through 8

   0.0000 + 0.5000i   0.0000 + 0.0000i   0.0000 - 0.5000i   0.0000 + 0.0000i

  Columns 9 through 11

   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i

  Columns 1 through 7

         0    0.0354         0   -0.0455         0    0.0637         0

  Columns 8 through 14
```

```
  -0.1061         0    0.3183    0.5000    0.3183         0   -0.1061

  Columns 15 through 21

         0    0.0637         0   -0.0455         0    0.0354         0

  Columns 1 through 7

  -62.8319  -56.5487  -50.2655  -43.9823  -37.6991  -31.4159  -25.1327

  Columns 8 through 14

  -18.8496  -12.5664   -6.2832         0    6.2832   12.5664   18.8496

  Columns 15 through 21

   25.1327   31.4159   37.6991   43.9823   50.2655   56.5487   62.8319

Warning: Imaginary parts of complex X and/or Y arguments ignored.
Warning: Imaginary parts of complex X and/or Y arguments ignored.
Warning: Imaginary parts of complex X and/or Y arguments ignored.
Warning: Imaginary parts of complex X and/or Y arguments ignored.
```



Published with MATLAB® R2023a