Introduction to Matlab Programming

Zsolt Csaba Johanyák

http://johanyak.hu

v1.4

Contents

1 User Input and If Statement – Conditional Branch

Calculate the roots of the equation

$$a \cdot x^2 + b \cdot x + c = 0$$

```
%
% Calculate the Roots of a Second Order Equation
% a*x^2+b*x+c=0
%

% Ask the user for the parameters.
a=input('a=')
b=input('b=')
c=input('c=')
% Calculate the discriminant.
D=b^2 - 4*a*c;
% Calculate the real and complex roots.
if D>=0
    x1= -(b + D^(1/2))/(2*a)
    x2= -(b - D^(1/2))/(2*a)
else
     real=-b/2/a;
     imaginary=(-D)^(1/2)/2/a;
     c1=[num2str(real) '+' num2str(imaginary) '*i']
     c2=[num2str(real) '-' num2str(imaginary) '*i']
end
```

```
a=1
b=3
c=1
```

```
x1 =   -2.6180
x2 =   -0.3820
```

2 Using Symbolic Math

2.1 Solving an Equation

Calculate the roots of the equation

$$a \cdot x^2 + b \cdot x + c = 0$$

using symbolic math.

```
%
% Solving an Equation using Symbolic Tools
%

% Clear workspace (delete previously defined variables).
clear all
% Define symbolic variable.
syms a b c x
% Solve equation.
solve(a*x*x+b*x+c)
```

```
ans =

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

2.2 Indefinite Integral

$$\int \sin(x)dx,\ \int \frac{1}{1+(6x-1)^2}dx,\ \int e^x \cdot \sin(x)^2 dx,\ \int \frac{1}{1+\sin(3x)^2}dx,\ \int \frac{4}{\sqrt{3x^2+10x+9}}dx$$

```
%
% Symbolic Integration
%
syms x
int(sin(x))

int(1/(1+(6*x-1)^2))

int(exp(x)*sin(x)^2)

int(1/(1+sin(3*x)^2))

int(4/(sqrt(3*x*x+10*x+9)))
```

```
ans = -cos(x)
ans = ((6*x - 1)*((6*x - 1)^2 + 3))/18
ans = -(exp(x)*(cos(2*x) + 2*sin(2*x) - 5))/10
ans = (2^(1/2)*(3*x - atan(tan(3*x))))/6 + (2^(1/2)*atan(2^(1/2)*tan(3*x)))/6
ans = (4*3^(1/2)*log(3^(1/2)*(x + 5/3) + (3*x^2 + 10*x + 9)^(1/2)))/3
```

2.3 Definite Integral

$$\int_0^1 \sin(x)dx,\ \int_0^\pi \sin(x)^2 dx,\ \int_1^2 \sin(\ln(x))dx,\ \int_0^3 \frac{1}{\sqrt{(4x-1)^2+11}}dx$$

```
syms x
int(sin(x),0,1)
double(int(sin(x),0,1))

double(int(sin(x)^2,0,pi))

double(int(sin(log(x)),1,2))

double(int(1/sqrt((4*x-1)^2+11),0,3))
```

```
ans = 1 - cos(1)
ans =    0.4597
ans =    1.5708
ans =    0.3697
ans =    0.5528
```

2.4 Derivative of a Function

$$\frac{d\,\sin(x)}{dx}$$

```
%
% Symbolic Derivative Calculation
%

syms x
diff(sin(x))
```

```
ans = cos(x)
```

3 Creating Functions

```
function [x,y]=polar2rect(r,theta)
% Function for coordinate transformation from polar to rectangular
% coordinates.
% r     - radius.
% theta - angle in radians.

x=r*cos(theta);
y=r*sin(theta);% Cumulative integration
```

>> [x,y]=polar2rect(50,pi/4)

x = 35.3553

y = 35.3553Results =

4 Vectorized Operations – Element by Element Multiplication of Vectors/Matrices

```
function [ f ] = CompFunc( x,y )
% Function applying vectorized operations.
% Element by element vector/matrix multiplication.
f=y./exp(x.*x+y.*y);
```

>> >> CompFunc([1 2],[0 1])

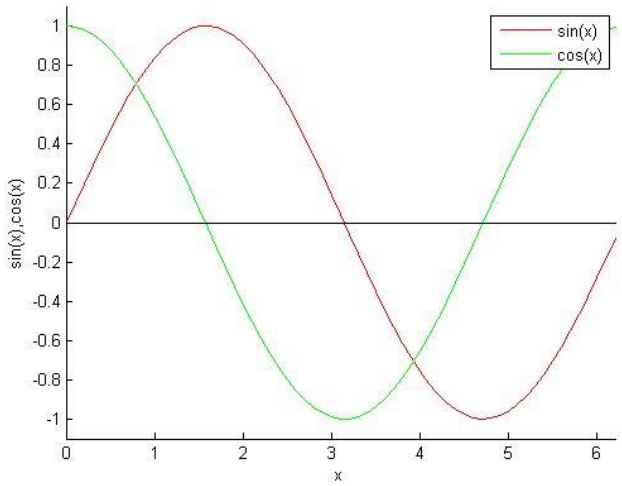
ans = 0 0.0067

5 Graphical Representation

5.1 Graphical 2D Representation of a Function

```
%
% Graphical representation of a Function
%

% Define a uniform distributed set of x values with a step of 0.1. Create a
% row vector with these values.
x=0:0.1:2*pi;
% Calculate the values of the trigonometrical funtions sin and cos these x
% values (vector based calculation).
s=sin(x);
c=cos(x);
% Create a window.
h=figure;
% Do not delete the previously drawn fugures.
hold all
% Plot the sin function with continuous red line.
plot(x,s,'-r')
% Plot the sin function with continuous green line.
plot(x,c,'-g')
% Define label for the abscissa.
xlabel('x')
% Define label for the ordinate.
ylabel('sin(x),cos(x)')
% Define the showed area (lower and upper bounds in both directions).
axis([x(1) x(63) -1.1 1.1])
% Draw horizontal axis.
plot([x(1) x(63)],[0 0],'-k')
```

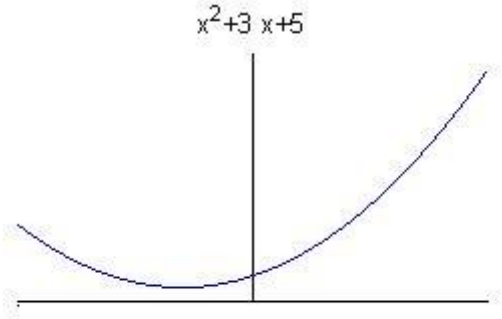
```
% Draw vertical axis.
plot([0 0],[-1.1 1.1],'-k')
% Switch off the border lines.
box off
% Set background color white.
set(h,'Color','w')
% Show legend.
legend('sin(x)','cos(x)')
```



5.2 Customizing the Graphical Representation and using ezplot

```
%
% Customizing the Graphical Representation
%

% Create window.
h=figure;
% Keep all drawings.
hold all
% Switch off the grid.
grid off
% Switch off the upper and right hand side lines of the border.
box off
% Name of the figure (appears in the title bar of the window).
set(h,'Name','Function representation')
% Switch off the showing of the number of the window.
set(h,'NumberTitle','off')
% Set background color to white.
set(h,'Color','white')
% Set the measurement units to cm.
set(h,'Units','centimeters')
% Get the position and size information of the figure.
pos=get(h,'Position');
% Redefine the size.
```

```
pos(3:4)=[8 5];
% Set the size and position onformation.
set(h,'Position',pos)
%
% Switch off the original horizontal and vertical axis.
%
% Find the handle of the axes object.
hA = findobj(h,'type','axes');
% Set the color of both axis to white.
set(hA,'XColor','w')
set(hA,'YColor','w')

% Plot the new vertical and horizontal axis.
plot([0 0],[0 50],'-k')
plot([-5 5],[0 0],'-k')
% Set the lower and upper bounds of the showed area.
axis([-4.5 4.5 0 40])

% Plot the function.
ezplot('x^2+3*x+5',[-5,5])
```



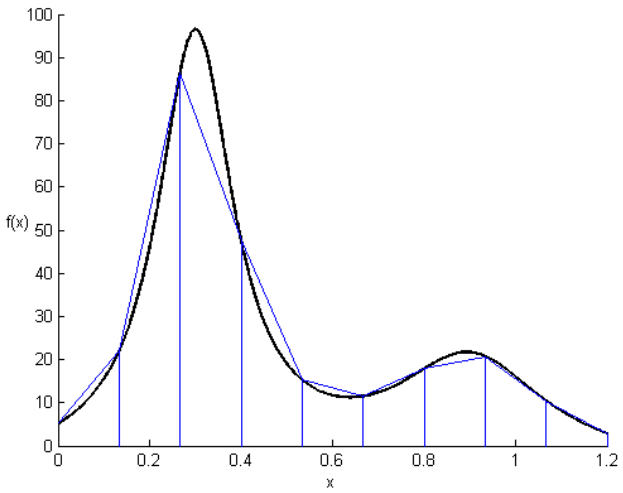
6 Numerical Methods

6.1 Numerical Integration

```
%
% Numerical integretion using the trapezoidal method
%
% Calculate the values of the function with high resolution.
x0=linspace(0,1.2,2000);
y0=humps(x0);
% Caculate the values of the funtion for the trapezoidal method with low
% resolution.
x=linspace(0,1.2,10);
y=humps(x);
% Calculate the integral value with the low resolution.
A=trapz(x,y)
% Graphical representation.
% Create window.
h=figure;
```

```
% Set background color.
set(h,'color','w')
% Set displayed area.
axis([0 1.2 0 100])
% Show labels.
xlabel('x')
ylabel('f(x)','Rotation',0)
hold all
% Plot the humps function
plot(x0,y0,'-k','linewidth',2)
% Plot the piecewise linear approximation of the original function.
plot(x,y,'-b')
% Plot the vertical lines at the endpoints of the subintervals.
for i=1:length(x)
    plot([x(i) x(i)],[0 y(i)],'-b')
end
```

```
A =   31.4152
```



6.2 Cumulative Integration with the Trapezoidal Method

```
%
% Cumulative integration
%
% Task: calculate the velocity and the displacement values from the
%       measured acceleration values.

% Time [s]
t=0:1:10;
% Measured acceleration values [m/s2]
a=[0 2 4 7  11 17 24 32 41 48 51];
```

```
% Velocity [m/s]
v=cumtrapz(t,a);
% Displacement [m]
x=cumtrapz(t,v);

% Create and show a matrix that contains all values.
Results=[t' a' v' x']
```

```
Results =
         0         0         0         0
    1.0000    2.0000    1.0000    0.5000
    2.0000    4.0000    4.0000    3.0000
    3.0000    7.0000    9.5000    9.7500
    4.0000   11.0000   18.5000   23.7500
    5.0000   17.0000   32.5000   49.2500
    6.0000   24.0000   53.0000   92.0000
    7.0000   32.0000   81.0000  159.0000
    8.0000   41.0000  117.5000  258.2500
    9.0000   48.0000  162.0000  398.0000
   10.0000   51.0000  211.5000  584.7500
```

6.3 Numerical Differentiation

```
%
% Numerical differentiation
%

% Example task: calculate the velocity and acceleration values from the
%               displacement values.

% Number of available displacement values.
n=length(x);
% Velocity calculation from the dislocation using the
% forward method.

% Preallocate a vector filled with zeros.
vf=zeros(1,n);
% Calculate the velocity.
for i=1:n-1
   vf(i)=(x(i+1)-x(i))/(t(i+1)-t(i));
end
% Only n-1 values can be calculated using this method. Therefore put a NaN
% at the end of the vector.
vf(end)=NaN;
% Append the new velocity vector to the end of the matrix as a column.
Results=[Results vf']

% Velocity calculation from the dislocation using the
% backward method.
vb=zeros(1,n);
% Only n-1 values can be calculated using this method. Therefore put a NaN
% into the first element of the vector.
```

```
vb(1)=NaN;
for i=2:n
   vb(i)=(x(i)-x(i-1))/(t(i)-t(i-1));
end
% Append the new velocity vector to the end of the matrix as a column.
Results=[Results vb' ]

% Velocity calculation from the dislocation using the
% central method.
vc=zeros(1,n);
% Only n-1 values can be calculated using this method. Therefore put a NaN
% into the first element and at the end of the vector.
vc(1)=NaN;
vc(end)=NaN;
for i=2:n-1
   vc(i)=(x(i+1)-x(i-1))/(t(i+1)-t(i-1));
end
% Append the new velocity vector to the end of the matrix as a column.
Results=[Results vc']

% Measure the quality of the approximation by calculating the mean square
% error for each case.
%
% Mean square of errors in case of the forward method.
MSEF=sum((Results(1:n-1,3)-Results(1:n-1,5)).^2)/(n-1)
% Mean square of errors in case of the backward method.
MSEB=sum((Results(2:n,3)-Results(2:n,6)).^2)/(n-1)
% Mean square of errors in case of the central method.
MSEC=sum((Results(2:n-1,3)-Results(2:n-1,7)).^2)/(n-2)
```

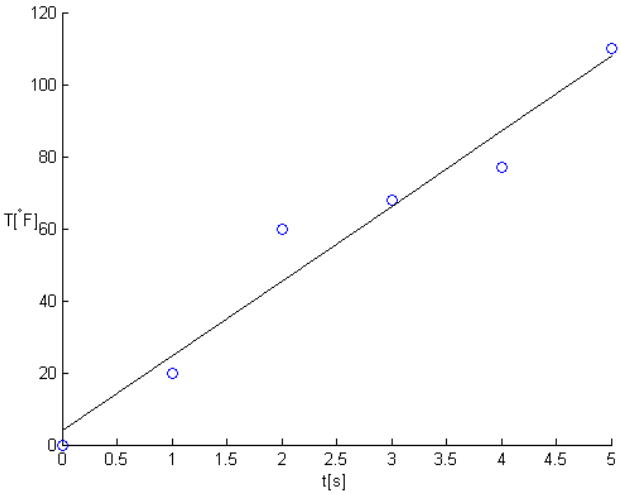
```
Results =

         0         0         0         0    0.5000
    1.0000    2.0000    1.0000    0.5000    2.5000
    2.0000    4.0000    4.0000    3.0000    6.7500
    3.0000    7.0000    9.5000    9.7500   14.0000
    4.0000   11.0000   18.5000   23.7500   25.5000
    5.0000   17.0000   32.5000   49.2500   42.7500
    6.0000   24.0000   53.0000   92.0000   67.0000
    7.0000   32.0000   81.0000  159.0000   99.2500
    8.0000   41.0000  117.5000  258.2500  139.7500
    9.0000   48.0000  162.0000  398.0000  186.7500
   10.0000   51.0000  211.5000  584.7500       NaN


Results =

         0         0         0         0    0.5000       NaN
    1.0000    2.0000    1.0000    0.5000    2.5000    0.5000
    2.0000    4.0000    4.0000    3.0000    6.7500    2.5000
    3.0000    7.0000    9.5000    9.7500   14.0000    6.7500
    4.0000   11.0000   18.5000   23.7500   25.5000   14.0000
    5.0000   17.0000   32.5000   49.2500   42.7500   25.5000
    6.0000   24.0000   53.0000   92.0000   67.0000   42.7500
    7.0000   32.0000   81.0000  159.0000   99.2500   67.0000
    8.0000   41.0000  117.5000  258.2500  139.7500   99.2500
    9.0000   48.0000  162.0000  398.0000  186.7500  139.7500
   10.0000   51.0000  211.5000  584.7500       NaN  186.7500


Results =

         0         0         0         0    0.5000       NaN       NaN
    1.0000    2.0000    1.0000    0.5000    2.5000    0.5000    1.5000
    2.0000    4.0000    4.0000    3.0000    6.7500    2.5000    4.6250
    3.0000    7.0000    9.5000    9.7500   14.0000    6.7500   10.3750
    4.0000   11.0000   18.5000   23.7500   25.5000   14.0000   19.7500
```

```
    5.0000   17.0000   32.5000   49.2500   42.7500   25.5000   34.1250
    6.0000   24.0000   53.0000   92.0000   67.0000   42.7500   54.8750
    7.0000   32.0000   81.0000  159.0000   99.2500   67.0000   83.1250
    8.0000   41.0000  117.5000  258.2500  139.7500   99.2500  119.5000
    9.0000   48.0000  162.0000  398.0000  186.7500  139.7500  163.2500
   10.0000   51.0000  211.5000  584.7500       NaN  186.7500       NaN

MSEF =  182.1062

MSEB =  182.1062

MSEC =    2.1337
```

6.4 Linear regression

```
%
% Curve fitting using linear regression
%

% Time-temperature.
t=0:5;
% Measured temperature values [Fahrenheit].
T=[0 20 60 68 77 110];
% Create window.
h=figure;
hold all
% Set background color.
set(h,'color','w')
% Plot measured values with cicles.
plot(t,T,'ob')
% Show labels.
xlabel('t[s]');
ylabel('T[^{\circ}F]','Rotation',0)
% Linear regression - T=a1*t+a2 - calculate parameters.
a=polyfit(t,T,1);
% Calculate the poits of the line corresponding to the predefined time values.
Tregr=polyval(a,t);
% Plot calculated points connected by lines.
plot(t,Tregr,'-k');
% Calculate mean square error.
MSE=sum((T-Tregr).^2)/length(t)
```

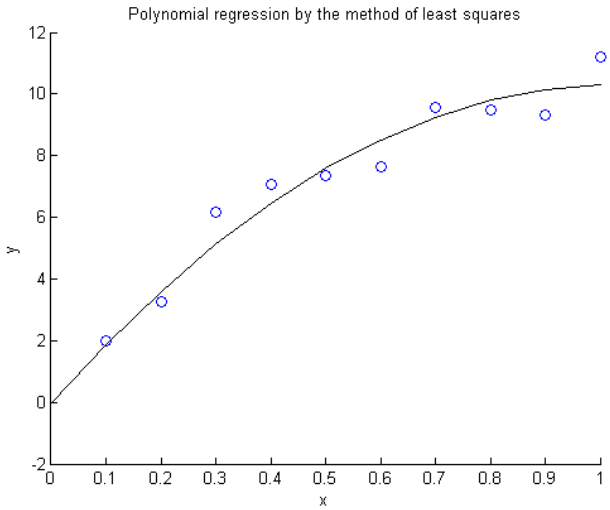
```
MSE =    59.4698
```



6.5 Regression with Higher Order Polynomial

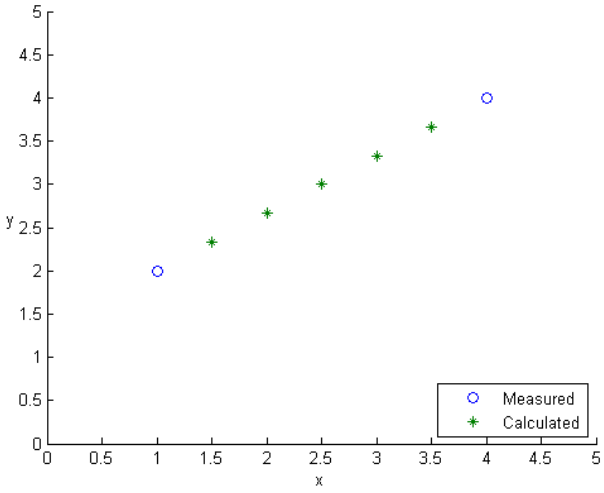
```
%
% Curve fitting using regression with higher order polynomial
%

% Measured values
x=0:0.1:1;
y=[-0.447 1.978 3.280 6.160 7.08 ...
    7.34 7.66 9.56 9.48 9.30 11.20];
% Graphical representation.
% Create window.
h=figure;
hold all
% Set background.
set(h,'Color','w');
% Set title.
title('Polynomial regression by the method of least squares')
% Set labels.
xlabel('x')
ylabel('y')
% Plot points using circles.
plot(x,y,'ob');
% Regression with second order polynomial.
a=polyfit(x,y,2);
% Calculate points corresponding to the polynomial.
yregr=polyval(a,x);
% Plot curve.
plot(x,yregr,'-k');
% Calculate mean square error.
MSE=sum((y-yregr).^2)/length(y)
```

```
MSE =    0.3882
```



6.6 Linear Interpolation

```
%
% Linear interpolation
%

% Measured values.
x=[1 4];
y=[2 4];
% Graphical representation.
h=figure;
set(h,'color','w');
xlabel('x')
ylabel('y','Rotation',0)
hold all;
% Plot measured values with circles.
plot(x,y,'o');
axis([0 5 0 5])
% Create a vector containing the values for which the interpolated values
% should be calculated.
x1=[1.5 2 2.5 3 3.5];
% Do the interpolation.
y1=interp1(x,y,x1);
% Plot interpolated points with stars.
plot(x1,y1,'*');
% plot(x1,y1,'-k');
legend('Measured','Calculated',4)
```



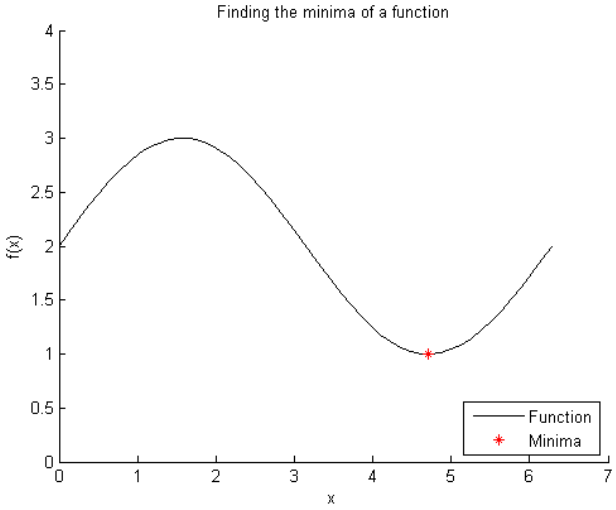
6.7 Find the Minima of a Function

```
%
% Find the minima of the function "newfc" on the interval [0,2pi]
%

xmin=fminbnd(@newfc,0,2*pi)

% Graphical representation
h=figure;
hold all
box off
set(h,'color','w');
title('Finding the minima of a function');
xlabel('x')
ylabel('f(x)')
% Calculate the function for 50 points
x=linspace(0,2*pi,50);
y=newfc(x);
% Plot function.
plot(x,y,'-k')
axis([0 7 0 4])
% Display minima.
plot(xmin,newfc(xmin),'*r')
legend('Function','Minima',4)
```

```
xmin =

    4.7124
```

The newfc.m file:

```
function [ y ] = newfc( x )
  y=sin(x)+2;
end
```



One can also use an inline function instead of applying a conventional function:

```
newfc=@(x)sin(x)+2
xmin=fminbnd(newfc,0,2*pi)
```

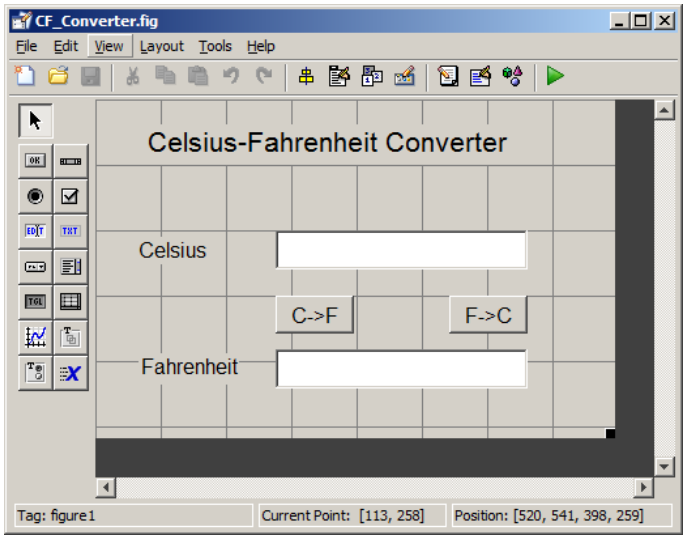
7 Creating a GUI Application

7.1 Temperature Converter

Taks: create a GUI based application that can convert temperature values between Celsius and Fahrenheit units.

File/New/GUI/Blank GUI/OK

File/Save as …/CF_Converter.fig



Components and Properties -> Property Inspector

Static Text	String: Celsius-Fahrenheit Converter
Static Text	String: Celsius
Static Text	String: Fahrenheit
Edit Text	String: <leave empty>
	Tag: etCelsius
	Font Size: 12
Edit Text	String: <leave empty>
	Tag: etFahrenheit
	Font Size: 12
Push Button	String: C->F
	Tag: pbCF
	Font Size: 12
Push Button	String: F->C
	Tag: pbFC
	Font Size: 12

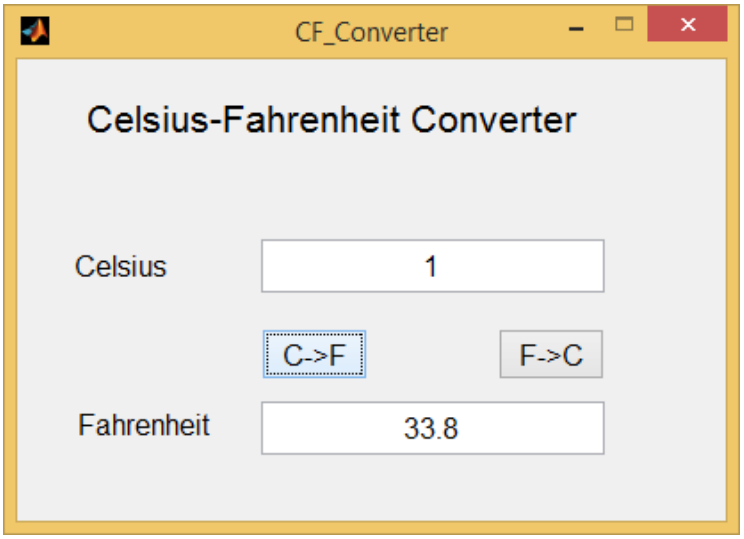
Align Objects

```
function varargout = CF_Converter(varargin)
% CF_CONVERTER MATLAB code for CF_Converter.fig
%      CF_CONVERTER, by itself, creates a new CF_CONVERTER or raises the existing
%      singleton*.
%
%      H = CF_CONVERTER returns the handle to a new CF_CONVERTER or the handle to
%      the existing singleton*.
```

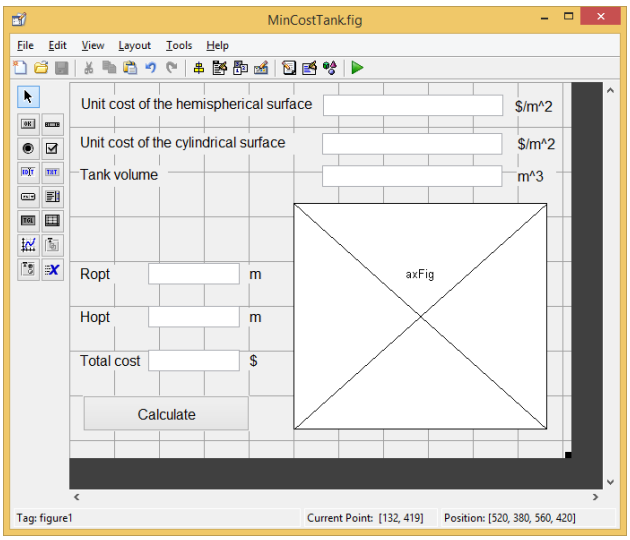
```
%
%      CF_CONVERTER('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in CF_CONVERTER.M with the given input arguments.
%
%      CF_CONVERTER('Property','Value',...) creates a new CF_CONVERTER or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before CF_Converter_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to CF_Converter_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help CF_Converter

% Last Modified by GUIDE v2.5 08-Apr-2015 17:27:01

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @CF_Converter_OpeningFcn, ...
                   'gui_OutputFcn',  @CF_Converter_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before CF_Converter is made visible.
function CF_Converter_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to CF_Converter (see VARARGIN)

% Choose default command line output for CF_Converter
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes CF_Converter wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = CF_Converter_OutputFcn(hObject, eventdata, handles)
```

```
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function etCelsius_Callback(hObject, eventdata, handles)
% hObject    handle to etCelsius (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of etCelsius as text
%        str2double(get(hObject,'String')) returns contents of etCelsius as a double

% --- Executes during object creation, after setting all properties.
function etCelsius_CreateFcn(hObject, eventdata, handles)
% hObject    handle to etCelsius (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function etFahrenheit_Callback(hObject, eventdata, handles)
% hObject    handle to etFahrenheit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of etFahrenheit as text
%        str2double(get(hObject,'String')) returns contents of etFahrenheit as a double

% --- Executes during object creation, after setting all properties.
function etFahrenheit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to etFahrenheit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pbCF.
function pbCF_Callback(hObject, eventdata, handles)
% hObject    handle to pbCF (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Celsius=str2double(get(handles.etCelsius,'String'));
if isnan(Celsius)
    errordlg('Please enter a correct value for degrees in Celsius!')
```

```
else
    Fahrenheit=((9/5)*Celsius)+32;
    set(handles.etFahrenheit,'String',num2str(Fahrenheit));
end

% --- Executes on button press in pbFC.
function pbFC_Callback(hObject, eventdata, handles)
% hObject    handle to pbFC (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Fahrenheit=str2num(get(handles.etFahrenheit,'String'));
if isnan(Fahrenheit)
    errordlg('Please enter a correct value for degrees in Fahrenheit!')
else
    Celsius=(5/9)*(Fahrenheit-32);
    set(handles.etCelsius,'String',num2str(Celsius));
end
```



7.2 Minimum Cost Tank Design



The values of the Tag properties for the components that are used programmatically.

etCylindrical

etHemispherical

etVolume

etRopt

etHopt

etTC

axFig

pbCalculate

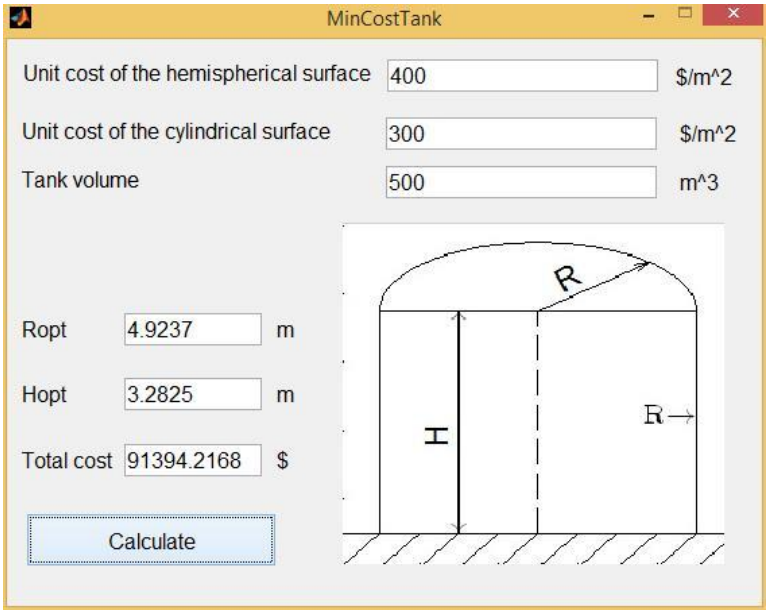
```
function varargout = MinCostTank(varargin)
% MINCOSTTANK MATLAB code for MinCostTank.fig
%      MINCOSTTANK, by itself, creates a new MINCOSTTANK or raises the existing
%      singleton*.
%
%      H = MINCOSTTANK returns the handle to a new MINCOSTTANK or the handle to
%      the existing singleton*.
%
%      MINCOSTTANK('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MINCOSTTANK.M with the given input arguments.
%
%      MINCOSTTANK('Property','Value',...) creates a new MINCOSTTANK or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before MinCostTank_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to MinCostTank_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help MinCostTank

% Last Modified by GUIDE v2.5 14-Apr-2015 15:04:24

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @MinCostTank_OpeningFcn, ...
                   'gui_OutputFcn',  @MinCostTank_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
```

```
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before MinCostTank is made visible.
function MinCostTank_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to MinCostTank (see VARARGIN)

% Choose default command line output for MinCostTank
handles.output = hObject;

im=imread('Tank.jpg');
axis(handles.axFig);
image(im);
box off
axis off
axis image;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes MinCostTank wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = MinCostTank_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function etHemispherical_Callback(hObject, eventdata, handles)
% hObject    handle to etHemispherical (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of etHemispherical as text
%        str2double(get(hObject,'String')) returns contents of etHemispherical as a double


% --- Executes during object creation, after setting all properties.
function etHemispherical_CreateFcn(hObject, eventdata, handles)
% hObject    handle to etHemispherical (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function etCylindrical_Callback(hObject, eventdata, handles)
% hObject    handle to etCylindrical (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of etCylindrical as text
%        str2double(get(hObject,'String')) returns contents of etCylindrical as a double


% --- Executes during object creation, after setting all properties.
function etCylindrical_CreateFcn(hObject, eventdata, handles)
% hObject    handle to etCylindrical (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function etVolume_Callback(hObject, eventdata, handles)
% hObject    handle to etVolume (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of etVolume as text
%        str2double(get(hObject,'String')) returns contents of etVolume as a double


% --- Executes during object creation, after setting all properties.
function etVolume_CreateFcn(hObject, eventdata, handles)
% hObject    handle to etVolume (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function etRopt_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to etROpt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of etROpt as text
%        str2double(get(hObject,'String')) returns contents of etROpt as a double


% --- Executes during object creation, after setting all properties.
function etROpt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to etROpt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function etHopt_Callback(hObject, eventdata, handles)
% hObject    handle to etHopt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of etHopt as text
%        str2double(get(hObject,'String')) returns contents of etHopt as a double


% --- Executes during object creation, after setting all properties.
function etHopt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to etHopt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function etTC_Callback(hObject, eventdata, handles)
% hObject    handle to etTC (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of etTC as text
%        str2double(get(hObject,'String')) returns contents of etTC as a double


% --- Executes during object creation, after setting all properties.
function etTC_CreateFcn(hObject, eventdata, handles)
% hObject    handle to etTC (see GCBO)
```

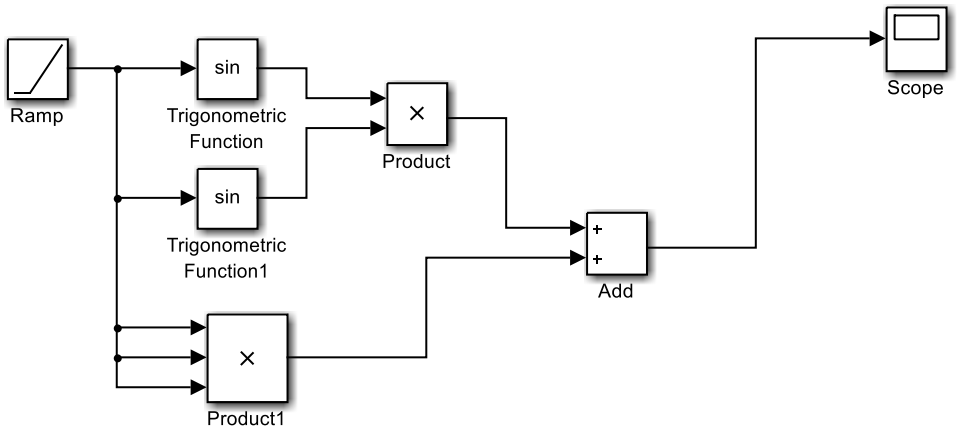
```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pbCalculate.
function pbCalculate_Callback(hObject, eventdata, handles)
% hObject    handle to pbCalculate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
CH=str2double(get(handles.etHemispherical,'String'));
if isnan(CH)
    errordlg('Please enter a correct value for the unit cost of the hemispherical part');
    return
end
CC=str2double(get(handles.etCylindrical,'String'));
if isnan(CH)
    errordlg('Please enter a correct value for the unit cost of the cylindrical part');
    return
end
V=str2double(get(handles.etVolume,'String'));
if isnan(CH)
    errordlg('Please enter a correct value for the tank volume');
    return
end
%
% Radius calculation for the minimum cost tank design problem.
%
% Total cost.
CT=@(R)2*CC*(V-2/3*pi*R.^3)./R+CH*2*pi*R.^2;

Ropt=fminbnd(CT,0,6.2035);
Hopt=(V-2/3*pi*Ropt^3)/pi/Ropt^2;
TotalCost=CT(Ropt);

set(handles.etRopt,'String',num2str(Ropt));
set(handles.etHopt,'String',num2str(Hopt));
set(handles.etTC,'String',num2str(TotalCost));
```

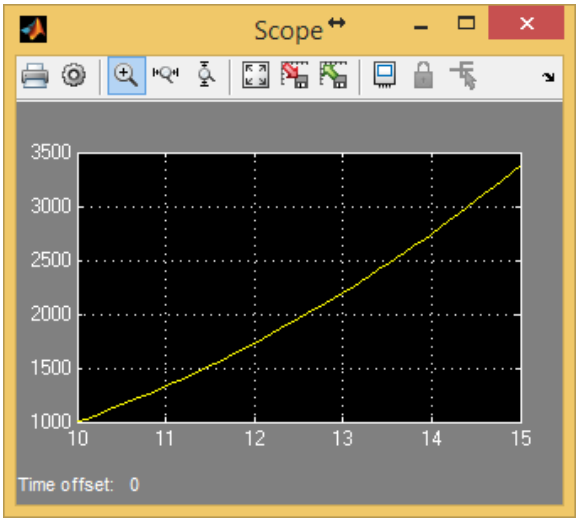


8 Simulink Basics

8.1 Graphical Representation of a Function

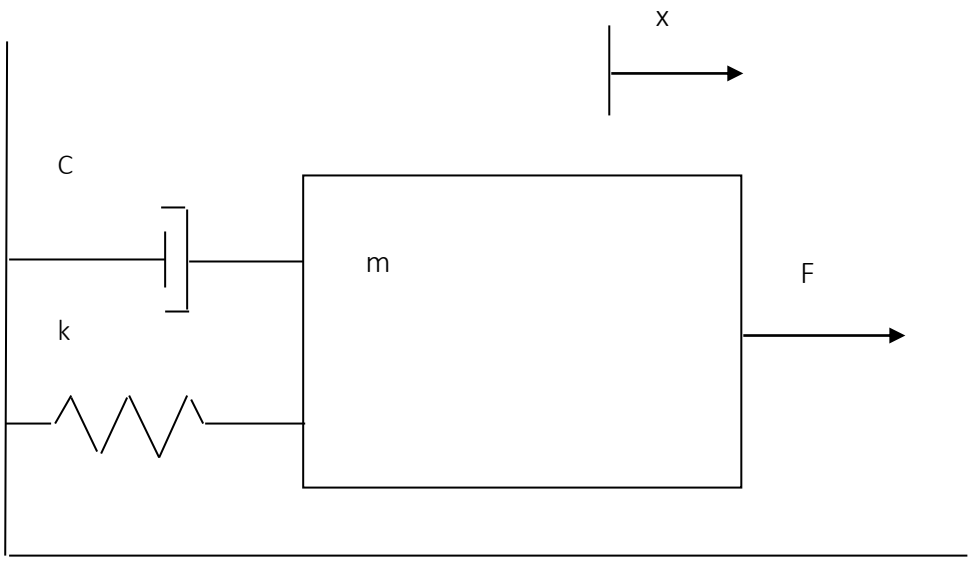
$f(x) = \sin^2(x) + x^3$





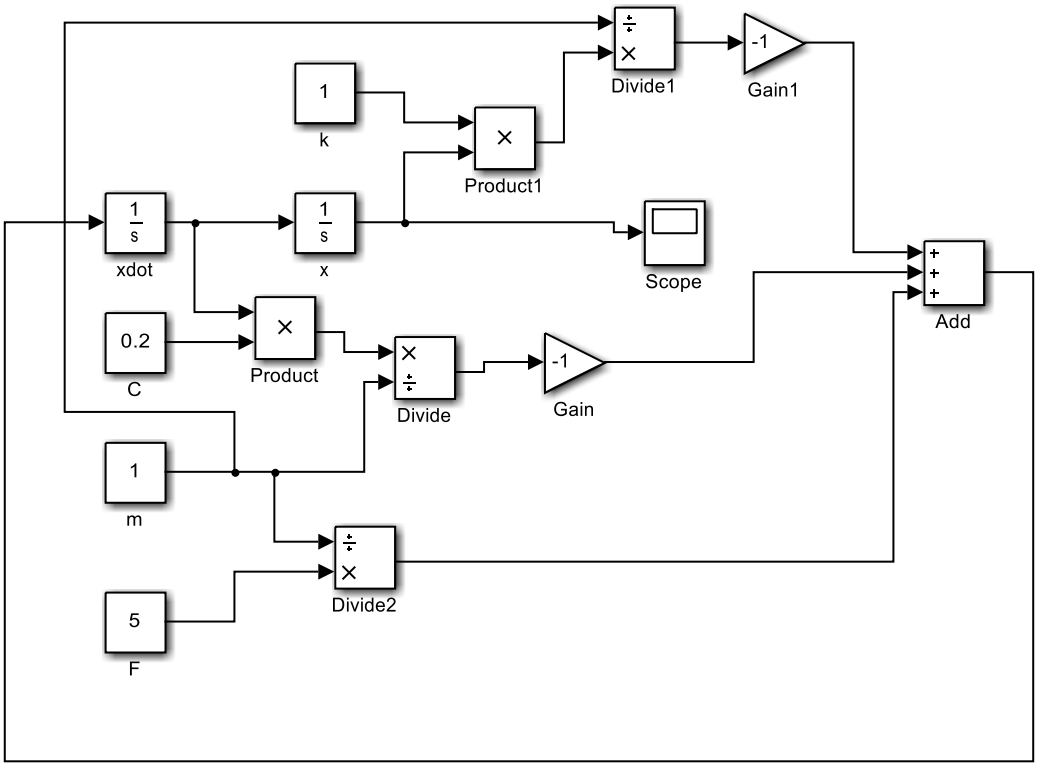
8.2 Cart model

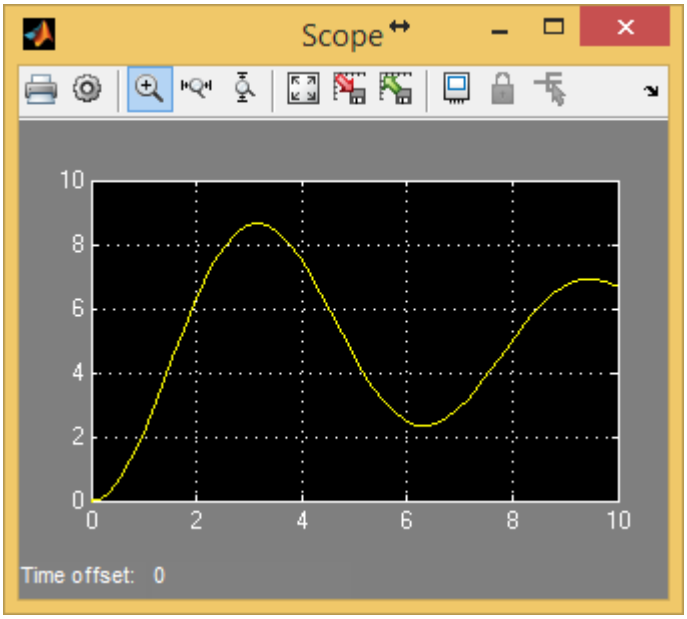
Mass spring damper example



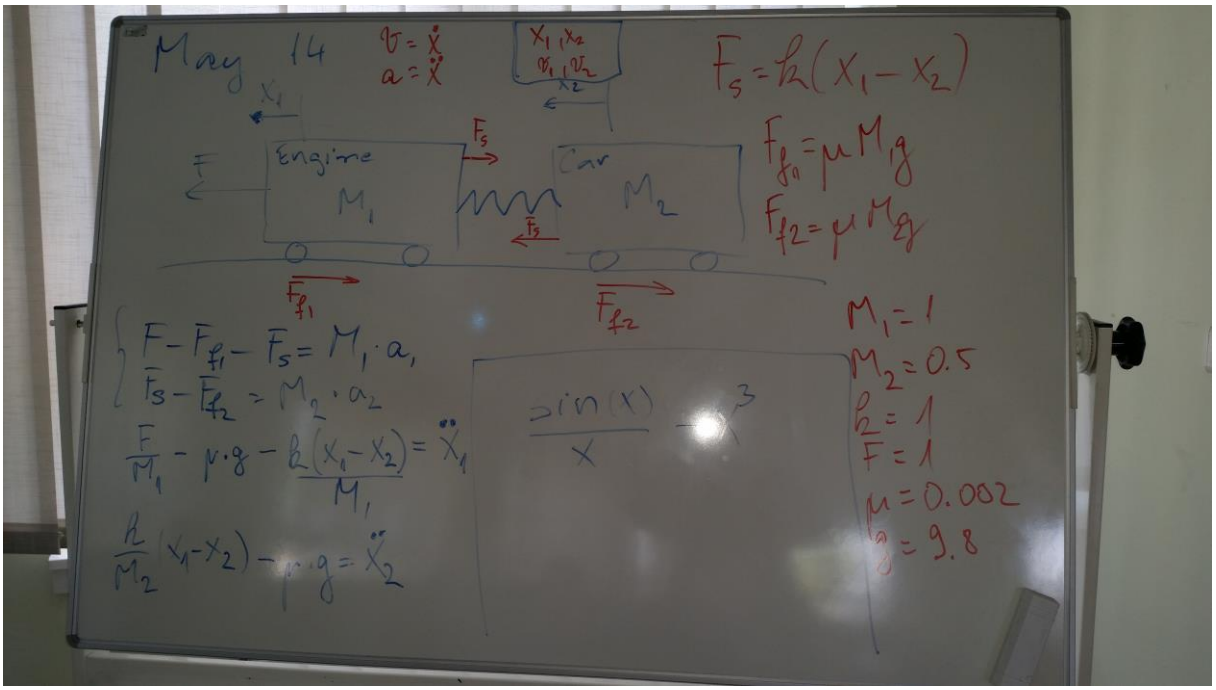
$$F - C \cdot v - k \cdot x = m \cdot a$$

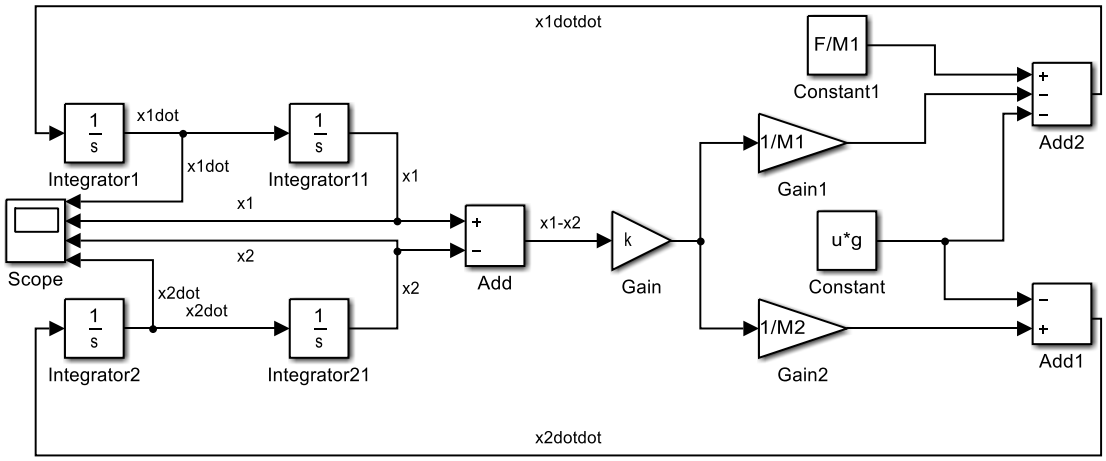
$$F - C \cdot x' - k \cdot x = m \cdot x''$$

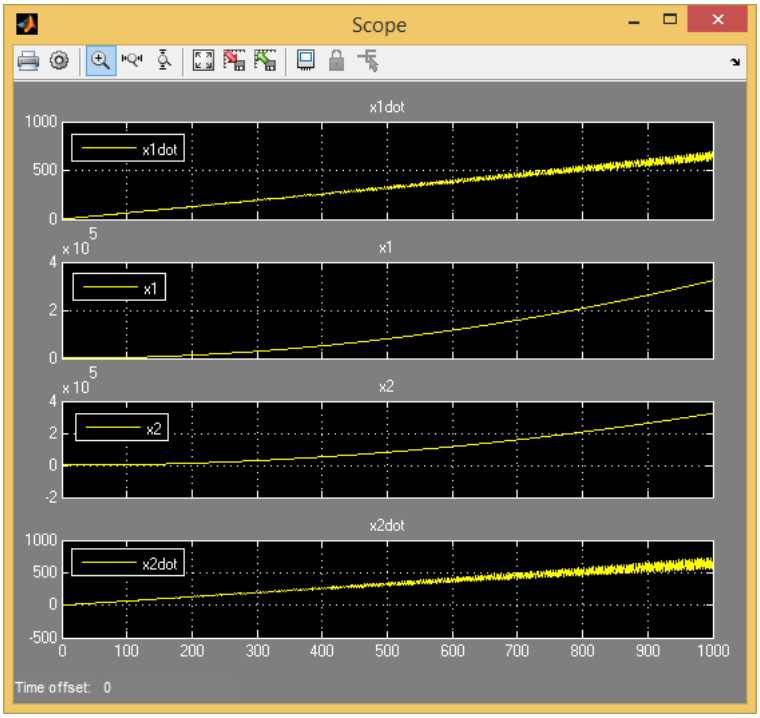

1
k
Product1
Divide1
-1
Gain1
1
s
xdot
1
s
x
Scope
0.2
C
Product
Divide
-1
Gain
Add
1
m
Divide2
5
F


Scope
10
8
6
4
2
0
0
2
4
6
8
10
Time offset: 0

8.3 Train System






Scope
x1dot
1000
500
0
x1dot
x 10^5
x1
4
2
0
x1
x 10^5
x2
4
2
0
-2
x2
x2dot
1000
500
0
-500
x2dot
0
100
200
300
400
500
600
700
800
900
1000
Time offset: 0