```
% % Mfile name
%   mtl_inp_sim_direct.m

% Language: Matlab 7.4.0 (R2007a)
% Revised:
%   June 30, 2008

% Authors:
%   Nathan Collier, Luke Snyder, Autar Kaw
%   University of South Florida
%   kaw@eng.usf.edu
%   Website: http://numericalmethods.eng.usf.edu

% Purpose
%   To illustrate the method of direct interpolation using Matlab.

% Keywords
%   Interpolation

% Clearing all data, variable names, and files from any other source and
% clearing the command window after each sucesive run of the program.
clc
clear all
clf

% Inputs:
%    This is the only place in the program where the user makes the changes
%    based on their wishes.

% Enter arrays of x and y data and the value of x at which y is desired.

    % Array of x-data
    x=[10 0 20 15 30 22.5];

    % Array of y-data
    y=[227.04 0 517.35 362.78 901.67 602.97];

    % Value of x at which y is desired
    xdesired = 16;

% ************************************************************************

disp(sprintf('\n\nDirect Method of Interpolation'))
disp(sprintf('\nUniversity of South Florida'))
disp(sprintf('United States of America'))
disp(sprintf('kaw@eng.usf.edu'))
disp(sprintf('\nNOTE: This worksheet illustrates the use of Matlab to demonstrate'))
disp(sprintf('the concepts of the direct method of interpolation.'))

disp(sprintf('\n*************************Introduction*************************'))

disp(sprintf('\nThe following simulation illustrates the direct method of interpolation'))
disp(sprintf('and its applications.  Given n data points of y versus x, the value of y'))
disp(sprintf('is found at a particular value of x using either first, second or third'))
disp(sprintf('order interpolation.  It is necessary to first pick the needed data points'))
disp(sprintf('and then use those points to interpolate the data.  The simulation will'))
disp(sprintf('choose from the user specified points, and determine which data points'))
disp(sprintf('are closest to "xdesired" and that also bracket this number.  The direct'))
disp(sprintf('method of interpolation is then used to determine the value of y at the'))
disp(sprintf('specified value of x.'))
format short

disp(sprintf('\n\n*************************Input Data*****************************'))
fprintf('\n');
disp('x array of data:')
x
disp('y array of data:')
```

```
y
disp(sprintf('The value of x at which y is desired, xdesired = %g',xdesired))

disp(sprintf('\n**************************Simulation*****************************'))

% Determining whether or not "xdesired" is a valid point to ask for given
% the range of the x data.
high = max(x);
low = min(x);
if xdesired > high
    disp(sprintf('\nThe value entered for "xdesired" is too high. Please pick a smaller valu
    disp(sprintf('that falls within the range of x data.'))
    break;
elseif xdesired < low
    disp(sprintf('\nThe value entered for "xdesired" is too low.  Please pick a larger value
    disp(sprintf('that falls within the range of x data.'))
    break;
else
    disp(sprintf('\nThe value for "xdesired" falls within the given range of xdata.  The'))
    disp(sprintf('simulation will now commence.'));

% The following considers the x and y data and selects the two closest points to xdesired
% that also bracket it.
n = numel(x);
comp = abs(x-xdesired);
c=min(comp);

for i=1:n
    if comp(i)==c;
        ci=i;
    end
end

% The following sequence of if statements determines if the value examined
% in the x array is greater than or less than the value "xdesired".  Once
% this is determined, the remaining lines find the necessary points around
% the "xdesired" variable.
if x(ci) < xdesired
    q=1;

    for i=1:n
        if x(i) > xdesired
            ne(q)=x(i);
            q=q+1;
        end
    end
    b=min(ne);

    for i=1:n
        if x(i)==b
            bi=i;
        end
    end
end

if x(ci) > xdesired
    q=1;

    for i=1:n
        if x(i) < xdesired
            ne(q)=x(i);
            q=q+1;
        end
    end
    b=max(ne);

    for i=1:n
        if x(i)==b
```

```
            bi=i;
        end
    end
end

% If more than two values are needed, the following selects the subsequent values and puts
% them into a matrix, preserving the orginal data order.
for i = 1:n
    A(i,2)=i;
    A(i,1)=comp(i);
end

A=sortrows(A,1);
for i=1:n
    A(i,3)=i;
end

A=sortrows(A,2);
d=A(1:n,3);
if d(bi)~=2
    temp=d(bi);
    d(bi)=1;
    for i=1:n
        if i ~= bi & i ~= ci & d(i) <= temp
            d(i)=d(i)+1;
        end
        d(ci)=1;
    end
end

%%%%%%%%% LINEAR INTERPOLATION %%%%%%%%%

% Pick two data points
datapoints=2;
p=1;
for i=1:n
    if d(i) <= datapoints
        xdata(p)=x(i);
        ydata(p)=y(i);
        p=p+1;
    end
end

%Setting up the equations to find coefficients of the linear interpolant
M=[1 xdata(1)
    1 xdata(2)];
Y=[ydata(1)
    ydata(2)];

%Coefficients of linear interpolant
A=inv(M)*Y;
z=sym('z');
a0=sym('a0');
a1=sym('a1');
fl = a0 + a1*z;
fl=subs(fl,a0,A(1));
fl=subs(fl,a1,A(2));
fxdesired=subs(fl,z,xdesired);
fprev=fxdesired;

% Displaying the outputs:
disp(sprintf('\nLINEAR INTERPOLATION:'))
disp(sprintf('\nx data chosen: x1 = %g, x2 = %g',xdata(1),xdata(2)))
disp(sprintf('\ny data chosen: y1 = %g, y2 = %g',ydata(1),ydata(2)))

% Displaying the determined coefficients a0,a1.
disp(sprintf('\nCoefficients of linear interpolation: a0 = %g, a1 = %g',A(1),A(2)))
```

```
% Using concactenation to correctly display the equation in the command
% window.
fprintf('\n');
str1 = ['Final Function: f(x) = ',num2str(A(1))];
if A(2) > 0
    str2 = [' + ',num2str(A(2)),'x'];
else
    str2 = [' ',num2str(A(2)),'x'];
end

finalstr = [str1,str2];
disp(finalstr);

% Final value of function at "xdesired".
disp(sprintf('\nFunction value at xdesired = %g, f(%g) = %g',xdesired,xdesired,fxdesired))

% Creating an inline function to plot the results with.
func = inline([num2str(A(1)),'+',num2str(A(2)),'*z']);

% Figures displaying the interpolation and data points.
axis on
figure(1)
subplot(2,1,1)
fplot(func,[min(xdata),max(xdata)]);
hold on
title('Linear Interpolation (Data Points Used)','Fontweight','bold');
xlabel('x data','Fontweight','bold');
ylabel('y data','Fontweight','bold');
plot(xdata,ydata,'ro','MarkerSize',8','MarkerFaceColor',[1,0,0]);
plot(xdesired,fxdesired,'kx','Linewidth',2,'MarkerSize',12');
hold off

% Creating the second figure showing the full data set and interpolation.
axis on
subplot(2,1,2)
fplot(func,[min(xdata),max(xdata)]);
title('Linear Interpolation (Full Data Set)','Fontweight','bold')
xlabel('x data','Fontweight','bold');
ylabel('y data','Fontweight','bold');
hold on
plot(x,y,'ro','MarkerSize',8','MarkerFaceColor',[1,0,0])
plot(xdesired,fxdesired,'kx','Linewidth',2,'MarkerSize',12')
xlim([min(x) max(x)])
ylim([min(y) max(y)])
hold off

%%%%%%%%% QUADRATIC INTERPOLATION %%%%%%%%%

% Pick three data points
datapoints=3;
p=1;
for i=1:n
    if d(i) <= datapoints
        xdata(p)=x(i);
        ydata(p)=y(i);
        p=p+1;
    end
end

%Setting up the equations to find coefficients of the quadratic interpolant
M=[1 xdata(1) xdata(1)^2
    1 xdata(2) xdata(2)^2
    1 xdata(3) xdata(3)^2];

Y=[ydata(1)
    ydata(2)
    ydata(3)];
```

```
%Coefficients of quadratic interpolant
A=inv(M)*Y;
z=sym('z');
a0=sym('a0');
a1=sym('a1');
a2=sym('a2');
fq = a0 + a1*z + a2*z^2;
fq=subs(fq,a0,A(1));
fq=subs(fq,a1,A(2));
fq=subs(fq,a2,A(3));
fxdesired=subs(fq,z,xdesired);

fnew=fxdesired;
ea=abs((fnew-fprev)/fnew*100);
if ea >= 5
    sd=0;
else
    sd=floor(2-log10(abs(ea)/0.5));
end

% Displaying the results for Quadratic Interpolation.
disp('------------------------------------------------------------------------')
disp(sprintf('\nQUADRATIC INTERPOLATION:'))
disp(sprintf('\nx data chosen: x1 = %g, x2 = %g, x3 = %g',xdata(1),xdata(2),xdata(3)))
disp(sprintf('\ny data chosen: y1 = %g, y2 = %g, y3 = %g',ydata(1),ydata(2),ydata(3)))

% Displaying the determined coefficients a0, a1, a2.
disp(sprintf('\nCoefficients of Quadratic Interpolation: a0 = %g, a1 = %g, a2 = %g',A(1),A(2

% Using concactenation to properly display the final function in the
% command window.
fprintf('\n')
str1 = ['Final Function: f(x) = ',num2str(A(1))];
if A(2) > 0
    str2 = [' + ',num2str(A(2)),'x'];
else
    str2 = [' ',num2str(A(2)),'x'];
end

if A(3) > 0
    str3 = [' + ',num2str(A(3)),'x^2'];
else
    str3 = [' ',num2str(A(3)),'x^2'];
end

% Putting all the strings together
finalstr = [str1,str2,str3];
disp(finalstr);

% Final value of the function at "xdesired".
disp(sprintf('\nFunction value at xdesired = %g, f(%g) = %g',xdesired,xdesired,fnew))

% Displaying the number of significant digits that can be taken and
% absolute relative approximate error.
disp(sprintf('\nAbsolute relative approximate error, abrae = %g%%',ea))
disp(sprintf('\nNumber of significant digits at least correct, sig_dig = %g',sd))

% Creating the inline function for plotting to take place.
func = inline([num2str(A(1)),'+',num2str(A(2)),'*z+',num2str(A(3)),'*z^2']);

% Creating the plots for quadratic interpolation.
axis on
figure(2)
subplot(2,1,1)
fplot(func,[min(xdata),max(xdata)]);
title('Quadratic Interpolation (Data Points Used)','Fontweight','bold')
xlabel('x data','Fontweight','bold');
ylabel('y data','Fontweight','bold');
```

```
hold on
plot(xdata,ydata,'ro','MarkerSize',8','MarkerFaceColor',[1,0,0])
plot(xdesired,fxdesired,'kx','Linewidth',2,'MarkerSize',12')
hold off

% Creating the second graph for quadratic interpolation.
axis on
subplot(2,1,2)
fplot(func,[min(xdata),max(xdata)])
title('Quadratic Interpolation (Full data set)','Fontweight','bold')
xlabel('x data','Fontweight','bold');
ylabel('y data','Fontweight','bold');
hold on
plot(x,y,'ro','MarkerSize',8','MarkerFaceColor',[1,0,0])
plot(xdesired,fxdesired,'kx','Linewidth',2,'MarkerSize',12')
xlim([min(x) max(x)])
ylim([min(y) max(y)])
hold off
fprev=fnew;


%%%%%%%%% CUBIC INTERPOLATION %%%%%%%%%

% Pick four data points
datapoints=4;
p=1;
for i=1:n
    if d(i) <= datapoints
        xdata(p)=x(i);
        ydata(p)=y(i);
        p=p+1;
    end
end

%Setting up the equations to find coefficients of the quadratic interpolant
M=[1 xdata(1) xdata(1)^2 xdata(1)^3
    1 xdata(2) xdata(2)^2 xdata(2)^3
    1 xdata(3) xdata(3)^2 xdata(3)^3
    1 xdata(4) xdata(4)^2 xdata(4)^3];

Y=[ydata(1)
    ydata(2)
    ydata(3)
    ydata(4)];

%Coefficients of quadratic interpolant
A=inv(M)*Y;
z=sym('z');
a0=sym('a0');
a1=sym('a1');
a2=sym('a2');
a3=sym('a3');
fc = a0 + a1*z + a2*z^2+a3*z^3;
fc=subs(fc,a0,A(1));
fc=subs(fc,a1,A(2));
fc=subs(fc,a2,A(3));
fc=subs(fc,a3,A(4));
fxdesired=subs(fc,z,xdesired);

fnew=fxdesired;
ea=abs((fnew-fprev)/fnew*100);
if ea >= 5
    sd1=0;
else
    sd1=floor(2-log10(abs(ea)/0.5));
end

% Displaying the results for Cubic Interpolation.
```

```
disp('------------------------------------------------------------------------')
disp(sprintf('\nCUBIC INTERPOLATION:'))
disp(sprintf('\nx data chosen: x1 = %g, x2 = %g, x3 = %g, x4 = %g',xdata(1),xdata(2),xdata(3
disp(sprintf('\ny data chosen: y1 = %g, y2 = %g, y3 = %g, y4 = %g',ydata(1),ydata(2),ydata(3

% Displaying the determined coefficients a0, a1, a2, a3.
disp(sprintf('\nCoefficients of Quadratic Interpolation: a0 = %g, a1 = %g, a2 = %g, a3 = %g'

% Using concactenation to display the final function properly in the
% command window.
fprintf('\n');
str1 = ['Final Function: f(x) = ',num2str(A(1))];%g + %gx + %gx^2 + %gx^3',A(1),A(2),A(3),A(
if A(2) > 0
    str2 = [' + ',num2str(A(2)),'x'];
else
    str2 = [' ',num2str(A(2)),'x'];
end

if A(3) > 0
    str3 = [' + ',num2str(A(3)),'x^2'];
else
    str3 = [' ',num2str(A(3)),'x^2'];
end

if A(4) > 0
    str4 = [' + ',num2str(A(4)),'x^3'];
else
    str4 = [' ',num2str(A(4)),'x^3'];
end

% Putting all the strings together.
finalstr = [str1,str2,str3,str4];
disp(finalstr);

% Final value of the function at "xdesired".
disp(sprintf('\nFunction value at xdesired = %g, f(%g) = %g',xdesired,xdesired,fnew))

% Displaying the number of significant digits that can be taken and
% absolute relative approximate error.
disp(sprintf('\nAbsolute relative approximate error, abrae = %g%%',ea))
disp(sprintf('\nNumber of significant digits at least correct, sig_dig = %g',sd1))

% Creating the inline function to allow for plotting to take place.
func = inline([num2str(A(1)),'+',num2str(A(2)),'*z+',num2str(A(3)),'*z^2+',num2str(A(4)),'*z

% Plotting the cubic interpolation and data points.
axis on
figure(3)
subplot(2,1,1)
fplot(func,[min(xdata),max(xdata)])
title('Cubic Interpolation (Data Points Used)','Fontweight','bold')
xlabel('x data','Fontweight','bold');
ylabel('y data','Fontweight','bold');
hold on
plot(xdata,ydata,'ro','MarkerSize',8','MarkerFaceColor',[1,0,0])
plot(xdesired,fxdesired,'kx','Linewidth',2,'MarkerSize',12')
hold off

axis on
subplot(2,1,2)
fplot(func,[min(xdata),max(xdata)])
title('Cubic Interpolation (Full Data Set)','Fontweight','bold')
xlabel('x data','Fontweight','bold');
ylabel('y data','Fontweight','bold');

hold on
plot(x,y,'ro','MarkerSize',8','MarkerFaceColor',[1,0,0])
plot(xdesired,fxdesired,'kx','Linewidth',2,'MarkerSize',12')
```

```
xlim([min(x) max(x)])
ylim([min(y) max(y)])
hold off


end
```

```
Error: A BREAK statement appeared outside of a loop. Use RETURN instead.
```

Published with MATLAB® 7.3