Homework 7, ME3215 Spring 2022

Table of Contents

Curve fitting part 1: polynomial & spline interpolation

```
function HW7Report %ignore this
```

HW7_1 Linear spline equations

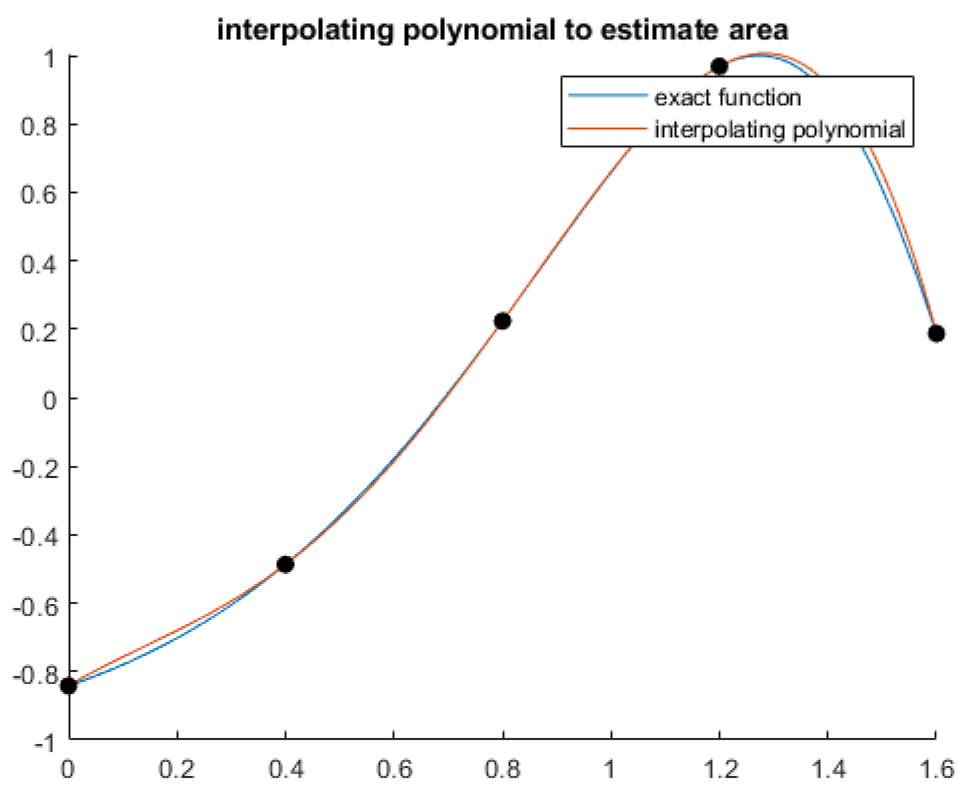
```
%Showing interpolation equations
 close all;clc;clear all

%data
x=[-4 -2 0 1 2 3]';
y=[-10 3 8 25 52 36]';

% (a) linear spline equation si = fi + (fi+1-f1)/(xi+1-xi) * (x-xi)
for i = 1:length(x)-1 %there are n-1 segments for n points
    coefs=(y(i+1)-y(i))/(x(i+1)-x(i));
    fprintf('s%g =%4.2f x + %2g for %g<=x<=%g\n',i,coefs,y(i)-
coefs*x(i),x(i:i+1));
end

% (b) Interpolate at x=-3.5, 0.8, 2.3.
i=1; coefs=(y(i+1)-y(i))/(x(i+1)-x(i));  xx=-3.5;
fprintf('\nAt x=%g, y= %4.2f * (%g) + %2g = %g\n',xx,coefs,xx,y(i)-
coefs*x(i), coefs*xx + y(i)-coefs*x(i))

i=3; coefs=(y(i+1)-y(i))/(x(i+1)-x(i));  xx=0.8;
fprintf('At x=%g, y= %4.2f * (%g) + %2g = %g\n',xx,coefs,xx,y(i)-
coefs*x(i), coefs*xx + y(i)-coefs*x(i))

i=5; coefs=(y(i+1)-y(i))/(x(i+1)-x(i));  xx=2.3;
fprintf('At x=%g, y= %4.2f * (%g) + %2g = %g\n',xx,coefs,xx,y(i)-
coefs*x(i), coefs*xx + y(i)-coefs*x(i))
```

```
s1 =6.50 x + 16 for -4<=x<=-2
s2 =2.50 x +  8 for -2<=x<=0
s3 =17.00 x +  8 for 0<=x<=1
s4 =27.00 x + -2 for 1<=x<=2
s5 =-16.00 x + 84 for 2<=x<=3
```

```
At x=-3.5, y= 6.50 * (-3.5) + 16 = -6.75
At x=0.8, y= 17.00 * (0.8) +  8 = 21.6
At x=2.3, y= -16.00 * (2.3) + 84 = 47.2
```

HW7_2: Comparing polynomial fit to sin(ex-p(x)-2)

```
clear;clc;close all
%the real function
f=@(x)sin(exp(x)-2);
%the "Data points"
x=[0 .4 .8 1.2 1.6 ];
y=[-0.8415 -0.4866 0.2236 0.9687 0.1874];
%polynomial fit
a = polyfit(x,y,4);
%the function, the data points & the polynomial fit
hold on
fplot(f,[0 1.6])
xx=linspace(0,1.6);
yy=polyval(a,xx);
plot(xx,yy)
plot(x,y,'ko','markerfacecolor','k')
title('interpolating polynomial to estimate area')
legend('exact function','interpolating polynomial')
```

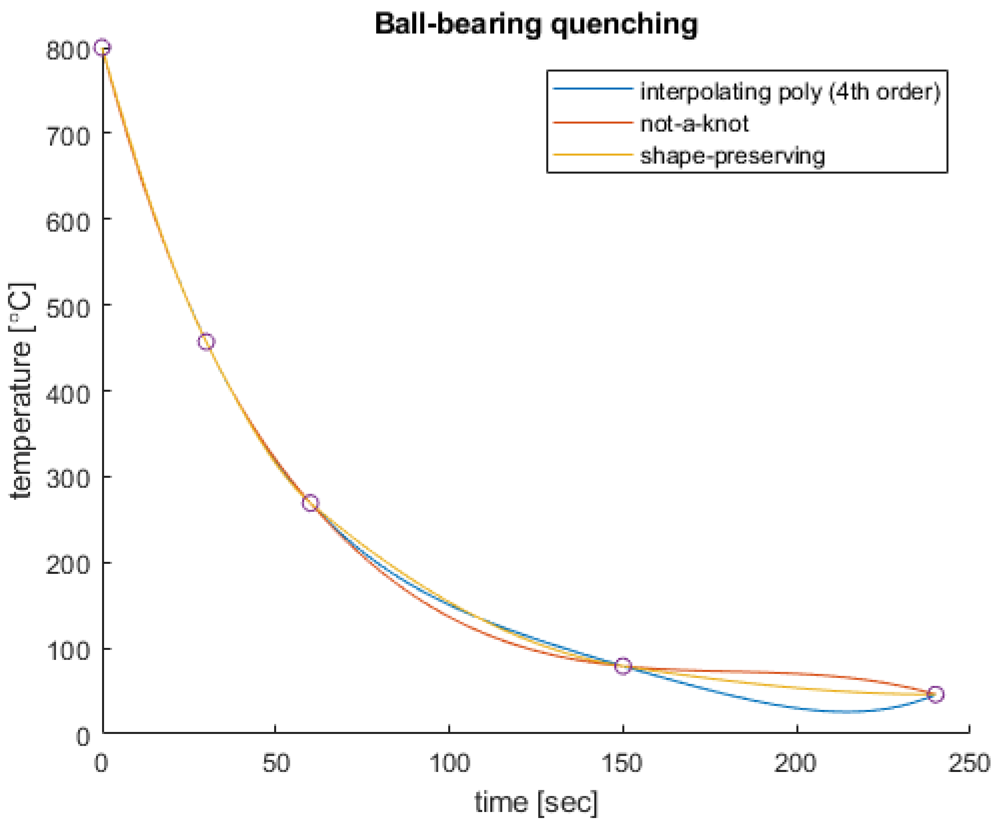


```
%save the coefs
a4=a(1);a3=a(2);a2=a(3);a1=a(4);a0=a(5);
%construct the integral of the polynomial
fi=@(x)1/5*a4*x.^5 + 1/4*a3*x.^4 + 1/3*a2*x.^3 + 1/2*a1*x.^2 + a0*x;
%there is actually a built-in function for this
b=polyint(a);
%evaluate the integral of the polynomial (either way)
Iest = fi(1.6)-fi(0);
Iest = polyval(b,1.6)-polyval(b,0);
%the polynomial is a good approximation
%the function cannot be integrated analytically
fprintf('Integral of interpolating polynomial is %.4f\n',Iest)
```

```
Integral of interpolating polynomial is 0.2406
```

HW7_3 Ball bearing quenching

```
clear;clc;close all

%data
t=[0 30 60 150 240];
T=[800 457 269 79 46];

a=polyfit(t,T,4);   %polynomial fit--warning!!
%evaluation
tt=linspace(0,240);     %100 evenly spaced points
TTc=polyval(a,tt);      %evaluation of polynomial
TTs=interp1(t,T,tt,'spline');%evaluation of piecewise cubic--not-a-
knot
TTp=interp1(t,T,tt,'pchip'); %evaluation of piecewise cubic--shape
 pres
%plotting
hold on
plot(tt,TTc,tt,TTs,tt,TTp)
plot(t,T,'o')
legend('interpolating poly (4th order)','not-a-knot','shape-
preserving')
title('Ball-bearing quenching')
xlabel('time [sec]')
ylabel('temperature [\circC]')
```




```
%shape-preserving is the best one, based on its shape.
%the not-a knot bumps up at the end. the spline dips down then curves
 up.
%neither of these is a good model for the temperature trend.

fprintf('The temperature after 3 minutes is %.4f
\n',interp1(t,T,180,'pchip'))
```

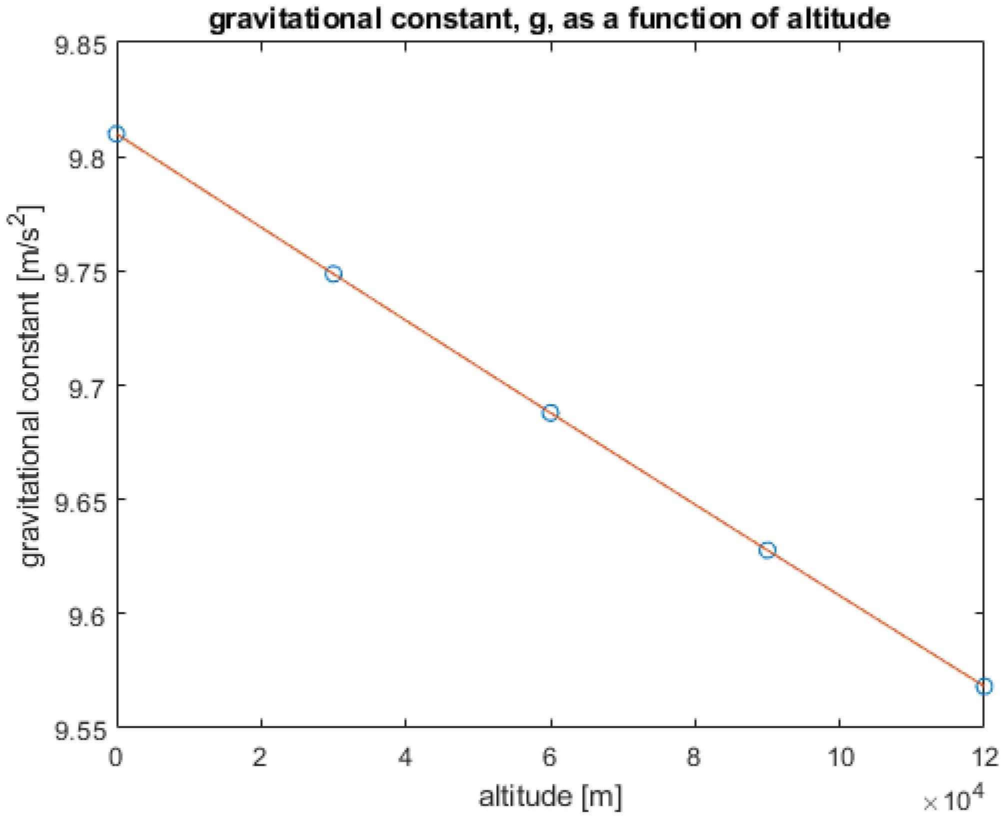
The temperature after 3 minutes is 62.1136

HW7_4: Gravitational constant

```
clear;clc;close all
y=(0:30:120)*1000; %altitude, m
g=[9.81 9.7487 9.6879 9.6278 9.5682];

%curve fit
a = polyfit(y,g,4);

%plot
plot(y,g,'o')
hold on
xx=linspace(y(1),y(end));
yy=polyval(a,xx);
plot(xx,yy)
```

```
xlabel('altitude [m]');ylabel('gravitational constant [m/s^2]')
title('gravitational constant, g, as a function of altitude')

%interpolate the polynomial
g55k=polyval(a,55000);
fprintf('\npolynomial interpolation: The gravitational constant at
 y=55000 m is %.4f m/s^2\n\n',g55k)

%linear interpolation instead
g55ki=interp1(y,g,55000);
fprintf('linear interpolation: The gravitational constant at y=55000 m
 is %.4f m/s^2\n\n',g55ki)

%inverse interpolation
y97i=interp1(g,y,9.75);
fprintf('linear interpolation: The altitude when g=9.75m/s^2 is %.0f m
\n',y97i)

% %NOT ASSIGNED we can "center and scale" to fix
% [P,S,MU] = polyfit(y,g,4);
% meany=MU(1);stdy=MU(2);
% figure
% %let's use polyval to plot
% xx=linspace(0,120)*1000;
% yy=polyval(P,((xx-meany)/stdy)); %scale & center
% plot(y,g,'*',xx,yy)
% title('scaled and centered...no warning message')
% %interpolate
% g55k=polyval(a,(55000-meany/stdy));
% fprintf('The gravitational constant at y=55000 m is %.4f m/
s^2\n',g55k)
```

```
Warning: Polynomial is badly conditioned. Add points with distinct X
 values,
reduce the degree of the polynomial, or try centering and scaling as
 described
in HELP POLYFIT.

polynomial interpolation: The gravitational constant at y=55000 m is
 9.6980 m/s^2

linear interpolation: The gravitational constant at y=55000 m is
 9.6980 m/s^2

linear interpolation: The altitude when g=9.75m/s^2 is 29364 m
```

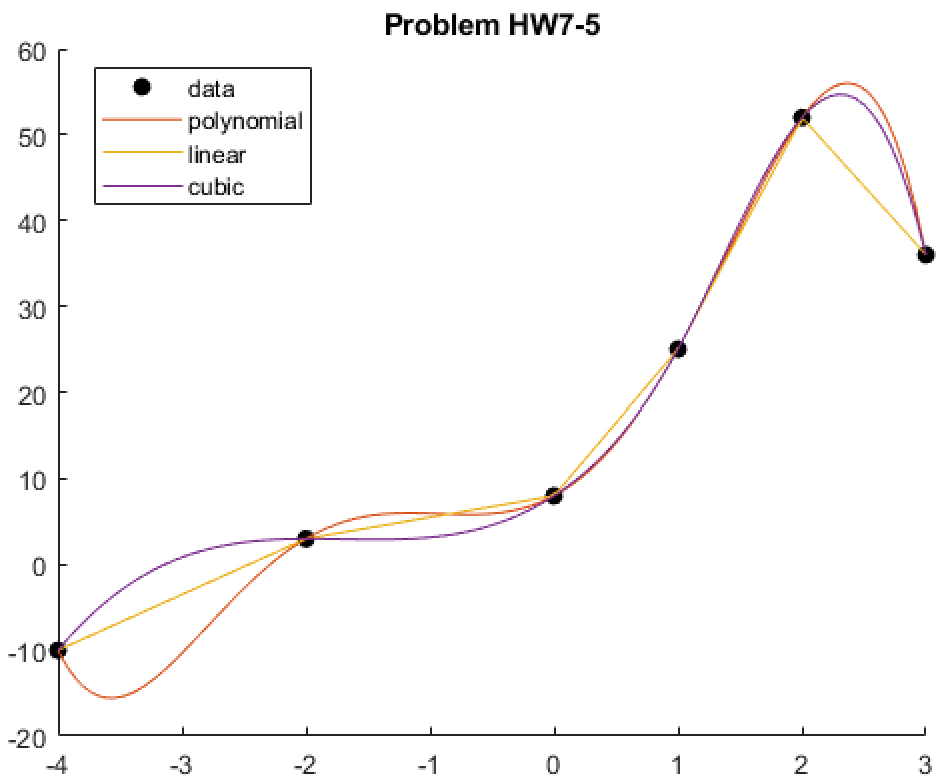


HW7_5: Interpolating polynomial with Vandermonde matrix

```
close all;clc;clear all

%data
x=[-4 -2 0 1 2 3]';
y=[-10 3 8 25 52 36]';

%or using backslash on Vandermonde matrix
Z = [x.^5 x.^4 x.^3 x.^2 x ones(size(x))];
a=Z\y;
%the function handle:
a5=a(1);a4=a(2);a3=a(3);a2=a(4);a1=a(5);a0=a(6);
f=@(x)a5*x.^5+a4*x.^4+a3*x.^3+a2*x.^2+a1*x+a0;

% print the equation of the polynomial, 5th order
fprintf('y = %3.2f x^5 %3.2g x^4 %+3.2g x^4 %+3.2g x^4 %+3.2g x %+3.2g
\n\n',a)

xx=linspace(x(1), x(end));
yy_l=interp1(x,y,xx);
yy_c=interp1(x,y,xx,'spline');
```

```
%plotting
hold on
plot(x,y,'ko','markerfacecolor','k')    %data pts
fplot(f,[min(x) max(x)]) %polynomial
plot(xx,yy_l)            %linear
plot(xx,yy_c)            %cubic

legend('data','polynomial','linear', 'cubic','location','best')
title('Problem HW7-5')
```

y = -0.23 x^5 -0.87 x^4 +2 x^4 +8.3 x^4 +7.7 x +8




```
end
```

Published with MATLAB® R2020a