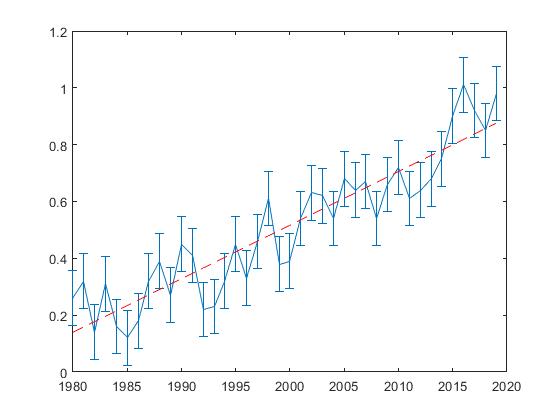
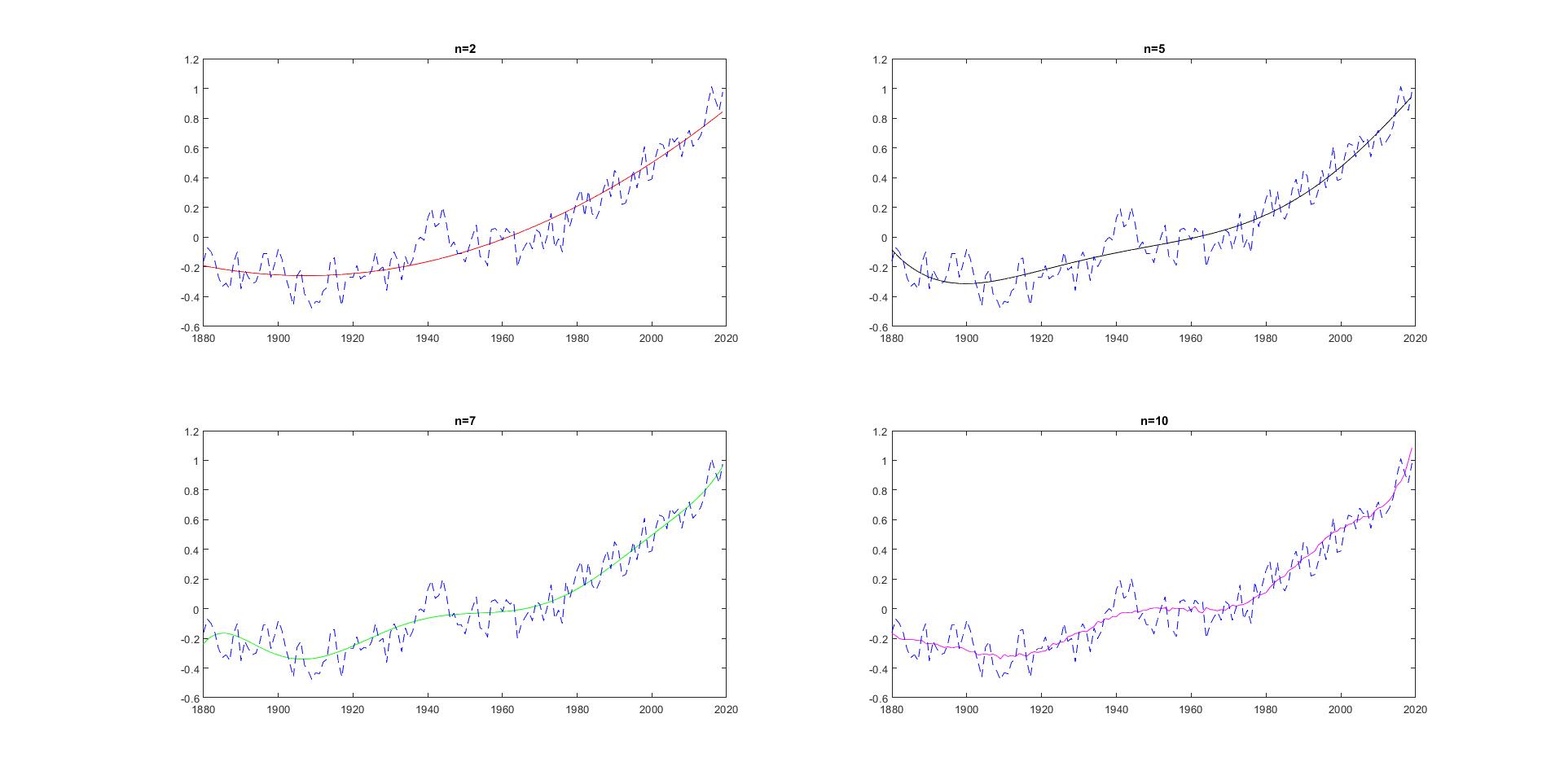
**Math Methods Hw 12**

**Jordan Mugglin, Alex Gonzalez, Tristan Noble**

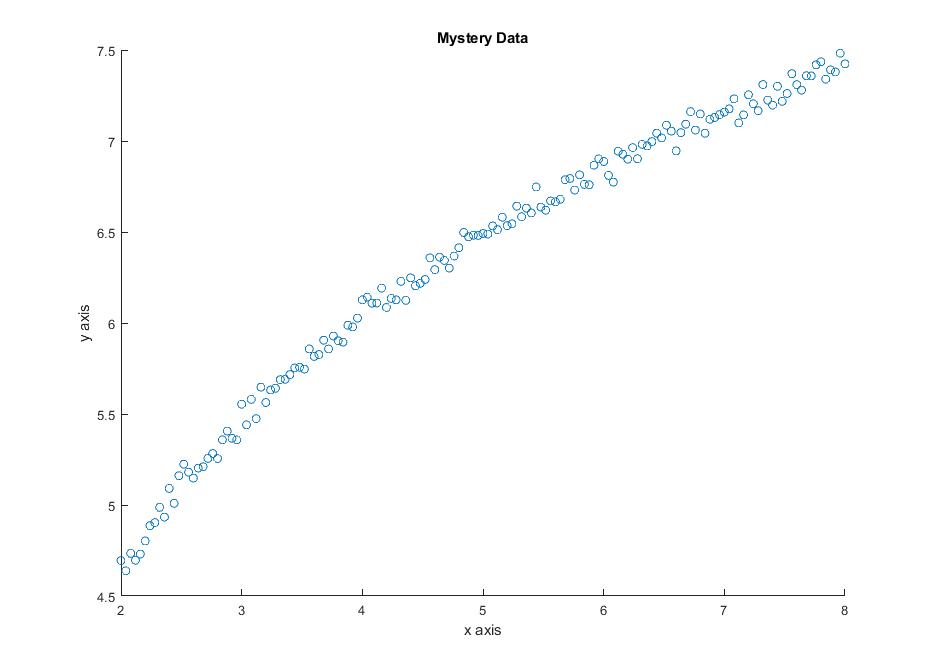
1.

1. Difference in means = 0.4817, uncertainty is 0.0304.
   1. Uncertainty is calculated by calculating the standard error from the standard deviation and sample size pre and post 1950, and then propagating those two uncertainties together.
2. CONF{0.4746 < μ < 0.4888}
3. P = 1.11022\*10^(-16)
   1. The integral evaluated to find this was
4. We used an initial guess of .05, which was too small. To get an rchi^2 as close to 1 as possible, we used an error of .096.
5. The n value we think is smallest while keeping the best representation of the data is n=7.



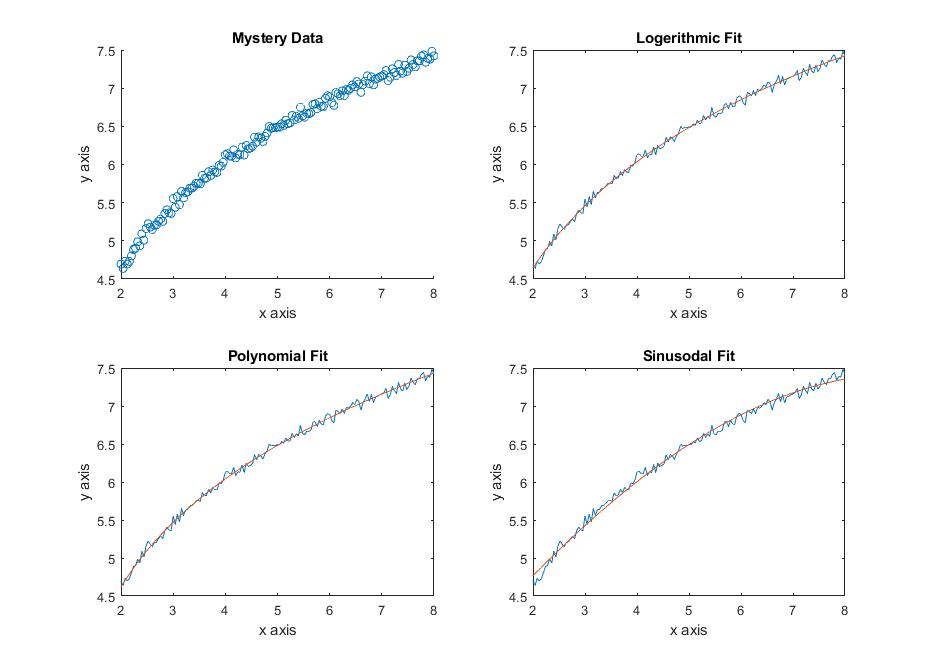
2.

1. These vary for each point, and are included in the graph
2. The function that fits best is the gaussian function. It looks like it fits best, and it has the lowest MSE.
3. The uncertainty in the peak center is 6.4787, and is greater than the spacing between points. This seems possible because if the error was smaller than the spacing between points it would be possible for data to not fit the curve.

3

a)

b) We could use logarithmic, exponential, or the beginning of a sine wave

c) 

Code!

% mathMethods\_hw12\_FINISHED.m

% Alex G, Jordan M, and Tristan N

%% #1

%Math Methods Homework 12, Problem 1. Climate Stuff%

%% Part a

dataPre1950 = climData(1:70,2);

dataPost1950 = climData(71:140,2);

muPre = mean(dataPre1950);

muPost = mean(dataPost1950);

stdPre = std(dataPre1950);

stdPost = std(dataPost1950);

SEpre = stdPre/sqrt(2\*70-2);

SEpost = stdPost/sqrt(2\*70-2);

%These two are the answer to part A%

muDifference = muPost - muPre;

muSEDiff = sqrt(SEpre^2+SEpost^2);

%% Part b

% 95 percent confidence interval

C = 1.96;

stdDiff = muSEDiff\*sqrt(2\*2-2);

k = C\*stdDiff/sqrt(140);

confmin = muDifference - k;

confmax = muDifference + k;

%% part c

% Probability of it being an anomaly%

% Lets assume that the mean of Reality is muPre1950, and calculate the

% probability of the temperature avg after that being as big as it is.

muReal = muPre;

stdReal = stdPre;

%Evaluated the cumulative distributive fxn in Mathematica, gave this result

P = 1.11022\*10^(-16);

%% part d

% Lets look back to 1980

% Lets say the error bar guess is .05 degrees

x = [1:1:40]';

x1 = climData(101:140,2);

y1 = 1980:2019;

y1T = transpose(y1);

c = polyfit(y1T,x1,1);

cest = polyval(c,y1T);

figure(1)

err = .0960\*ones(40,1);

errorbar(1980:2019,climData(101:140,2),err)

hold on

plot(y1T,cest,'r--')

hold off

%I'd say this guess is a little too small

[a,da,sig\_a2,rchi2] = weightedPoly(1,y1T,x1,err);

% Yeah, too small. Lets guess a uncertainty of .1

% After guessing around, final error bar we're going with is .096 %

%% Part e

years = climData(1:140,1);

temp = climData(1:140,2);

%Polynomial of degree 2

p2 = polyfit(years,temp,2);

p2est = polyval(p2,years);

%Polynomial of degree 5

p5 = polyfit(years,temp,5);

p5est = polyval(p3,years);

%Polynomial of degree 7

p7 = polyfit(years,temp,7);

p7est = polyval(p7,years);

%Polynomial of degree 10

p10 = polyfit(years,temp,10);

p10est = polyval(p10,years);

subplot(2,2,1)

plot(years,p2est,'r')

hold on

plot(years,temp,'b--')

title('n=2')

hold off

subplot(2,2,2)

plot(years,p5est,'k')

hold on

plot(years,temp,'b--')

title('n=5')

hold off

subplot(2,2,3)

plot(years,p7est,'g')

hold on

plot(years,temp,'b--')

title('n=7')

hold off

subplot(2,2,4)

plot(years,p10est,'m')

hold on

plot(years,temp,'b--')

title('n=10')

hold off

%n=7 is the lowest # polynomial that effectively conveys the details, in

%our opinion

%///////////////////////////////////////////////////////

%///////////////////////////////////////////////////////

%% #2

%% Part a

Error = spectrum(1:100,2)/mean(spectrum(1:100,2));

%% Part b

x=spectrum(1:100,1);

y=spectrum(1:100,2);

w=(1./y);

[betag,Rg,Jg,CovBg,MSEg,ERRORMODELINFO] = nlinfit(x,y,@gaussfit,[45,8,30,10],'Weights',w);

[betal,Rl,Jl,CovBl,MSEl,ERRORMODELINFO] = nlinfit(x,y,@lorentzfit,[45,8,30,10],'Weights',w);

[betas,Rs,Js,CovBs,MSEs,ERRORMODELINFO] = nlinfit(x,y,@sincfit,[45,8,30,10],'Weights',w);

subplot(3,1,1)

plot(x,lorentzfit(betal,x))

hold on;

errorbar(x,y,Error,'o')

subplot(3,1,2)

plot(x,gaussfit(betag,x))

hold on;

errorbar(x,y,Error,'o')

subplot(3,1,3)

plot(x,sincfit(betas,x))

hold on;

errorbar(x,y,Error,'o')

%Thefunction that fits best is the gaussian function.

%It looks like it is the best fit, as well as it has the lowest MSE.

%% Part c

StandardUg=sqrt(diag(CovBg));

%The uncertainty in the peak center is [6.47871253834100],

%and is greater than the spacing between points.

%This seems posible because if the error was smaler than the spacing between points it would be posible for data to not fit the curve.

%///////////////////////////////////////////////////////

%///////////////////////////////////////////////////////

%% #3

% a)

% mysteryCurve( X , Y , dy)

x = table2array(mysteryCurve(1:151, 1)); % pulling rows from column 1

y = table2array(mysteryCurve(1:151, 2));

dy = table2array(mysteryCurve(1:151, 3));

subplot(2,2,1)

scatter(x, y)

title('Mystery Data')

xlabel('x axis')

ylabel('y axis')

% b)

% we could use logerithmic, exponential, or the beginning of a sine wave

% c)

%% Fit Data - logerithmic

subplot(2,2,2)

logFun = @(beta, x) (beta(1) \* log(x) + beta(2))

logInitials = [1,1]

logCoeffs = nlinfit(x, y, logFun, logInitials)

log\_y = logFun(logCoeffs, x);

plot(x,y,...

x, log\_y)

title('Logerithmic Fit')

xlabel('x axis')

ylabel('y axis')

logDiff = rms(log\_y - y)

%% Fit Data - Polynomial

subplot(2,2,3)

polyFun = @(beta, x) (beta(1)\*(x - beta(2)).^beta(3))

polyInitials = [.1,1,.1]

polyCoeffs = nlinfit(x, y, polyFun, polyInitials)

poly\_y = polyFun(polyCoeffs, x);

plot(x,y,...

x, poly\_y)

title('Polynomial Fit')

xlabel('x axis')

ylabel('y axis')

polyDiff = rms(poly\_y - y)

%% Fit Data - Sinusodal

subplot(2,2,4)

sineFun = @(beta, x)(beta(1)\*sin(beta(2)\*x + beta(3)))

sineInitials = [.1,1,.1]

sineCoeffs = nlinfit(x, y, sineFun, sineInitials)

sine\_y = sineFun(sineCoeffs, x);

plot(x,y,...

x, sine\_y)

title('Sinusodal Fit')

xlabel('x axis')

ylabel('y axis')

sineDiff = rms(sine\_y - y)