Max Kramer

I affirm that I have adhered to the honor code on this assignment.

*Hello again, scientist! I'll do all my writing in italics, and problems for you will be in **bold.** Comment your code, and* explain your ideas in plaintext. *As a general rule, I expect you to do at least as much writing as I do. Code should be part of your solution, but I expect variables to be clear and explanation to involve complete sentences. Cite your sources; if you work with someone in the class on a problem, that's an extremely important source. Don't work alone.*

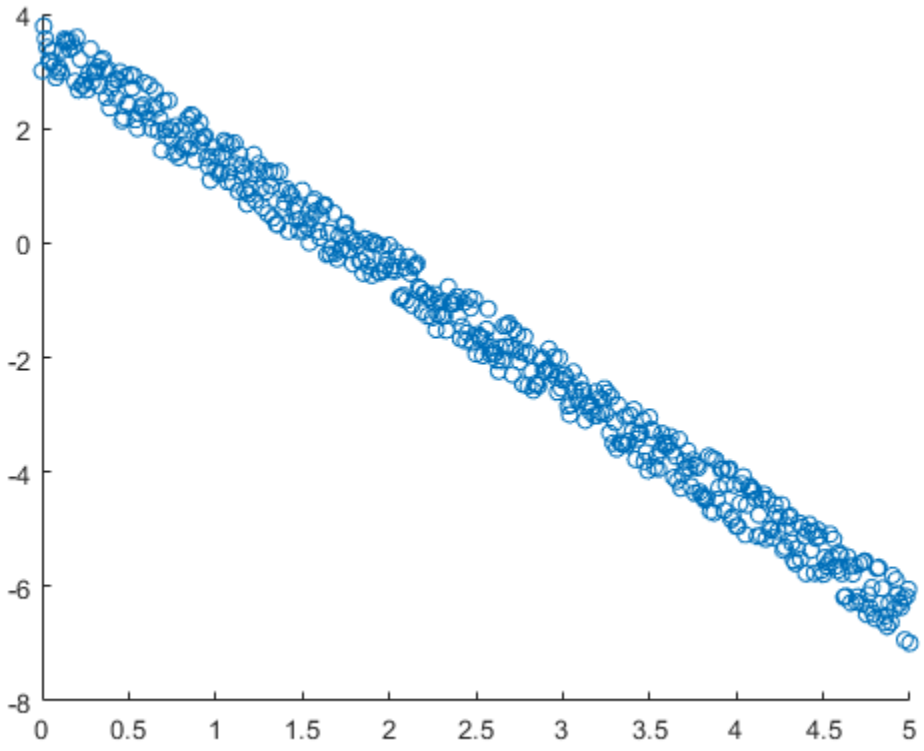
Problem F.02: Find the function.

Import the file data1.csv as data1. Use a semicolon.

```
data1 = csvread("data1.csv");
```

data1 represents the x-y coordinates of some unknown function. ***Show me a scatterplot of data1.***

```
scatter(data1(:,1),data1(:,2))
```



The scatter command plots points on the x-y plane. The first column of data1 is plotted along the x-axis and the second column along the y-axis.

Looks linear, huh? Let's find the ordinary line of best fit. I'll walk you through this one. When you're ready, ***uncomment the following.***

```
inputs = data1(:,1); outputs = data1(:,2);
syms x;
functionone = 1; % this is silly, but I'm thinking of this as the
 function y=1
functionx = x; % same deal, this is the function y = x
X = [subs(functionone,x,inputs) subs(functionx,x,inputs)]; % a linear
 transformation is determined...
```

*You're trying to best-fit the equation X*b = outputs.* ***Set up the normal equations as in Lay 6.6, and solve for b.***

```
Xtx = X' * X;
Xty = X' * outputs;

b = Xtx^(-1) * Xty
```

```
b =

 31885321295194723730585980 51/906131450786347587993600000
   -472298847106902807933321 49/235971731975611351040000000
```

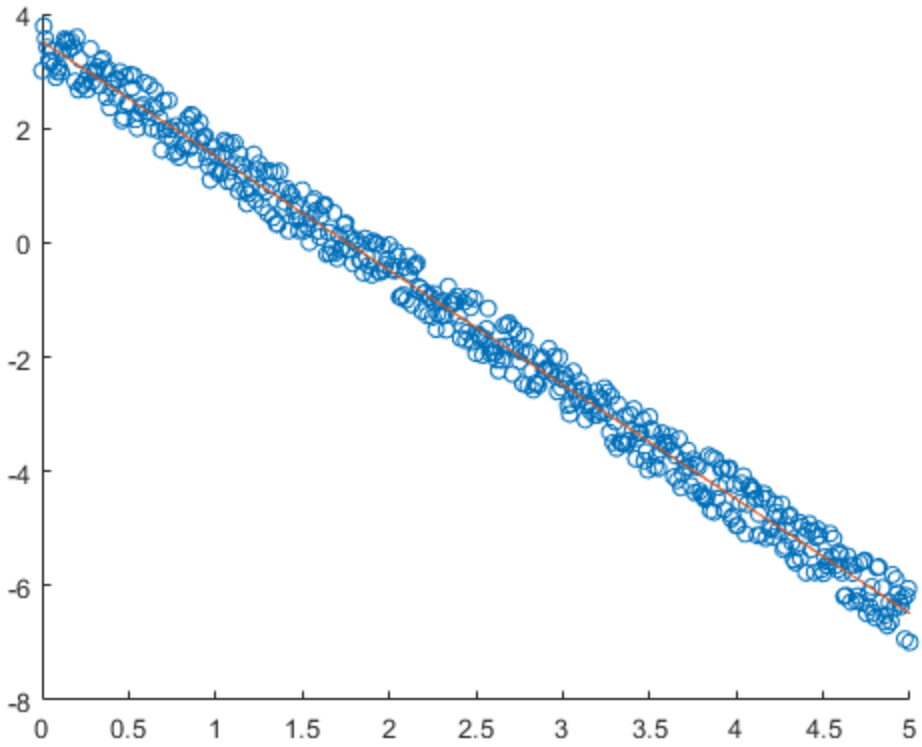
Then ***uncomment the following*** *to find "the" line of best fit; that is, the line that best approximates y as a function of x.*

```
f1 = [1 x]*b
```

```
f1 =

31885321295194723730585980 51/906131450786347587993600000 -
 (472298847106902807933321 49*x)/235971731975611351040000000
```

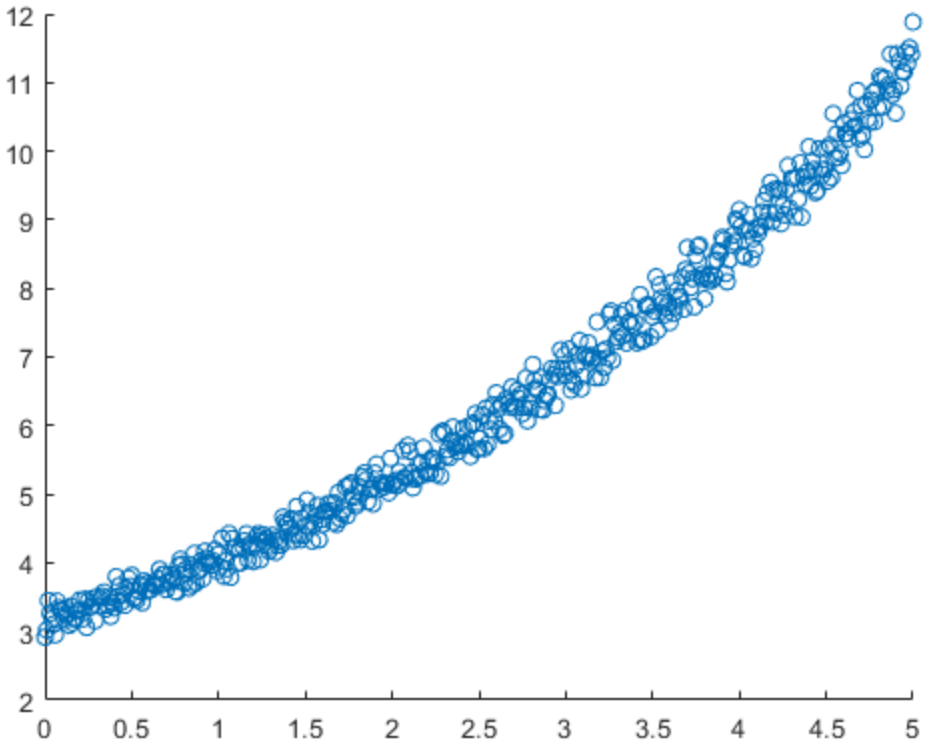
Plot data1 and f1 on the same set of axes.

```
figure;
scatter(data1(:,1),data1(:,2))
hold on
fplot(f1,[0 5])
```



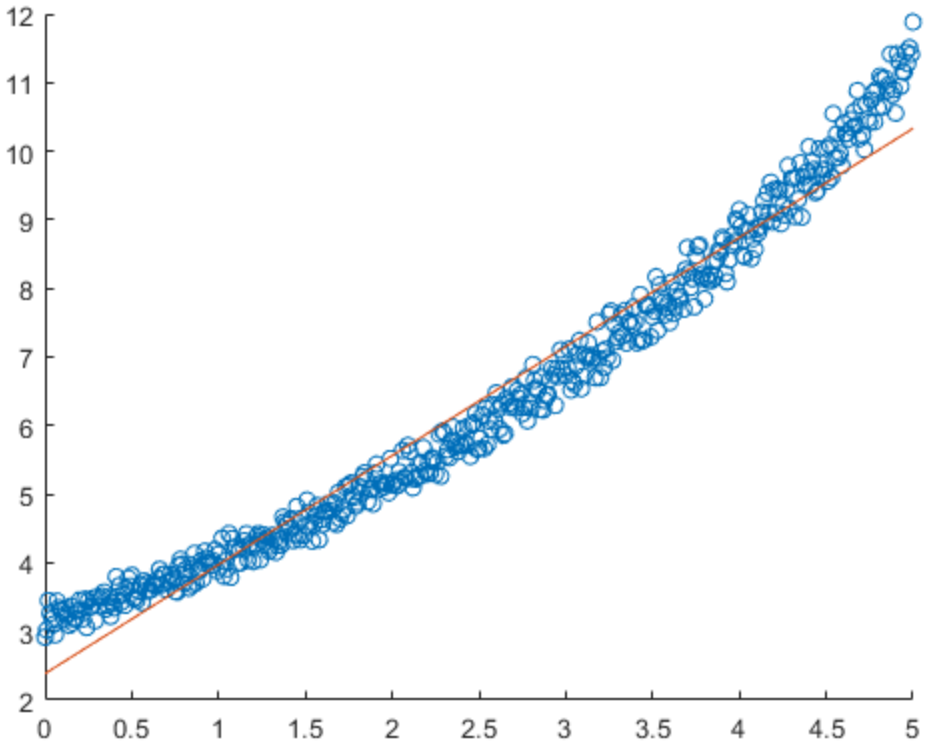
Cool, right? Let's move on. ***Import data2 and plot it.***

```
data2 = csvread("data2.csv");
figure;
scatter(data2(:,1),data2(:,2))
```



Find the ordinary line of best fit. Explain why it is bad.

```
inputs2 = data2(:,1); outputs2 = data2(:,2);
X2 = [subs(functionone,x,inputs2) subs(functionx,x,inputs2)];

Xtx2 = X2' * X2;
Xty2 = X2' * outputs2;

b2 = Xtx2^(-1) * Xty2;
f2 = [1 x]*b2;

figure;
scatter(data2(:,1),data2(:,2))
hold on
fplot(f2,[0 5])
```

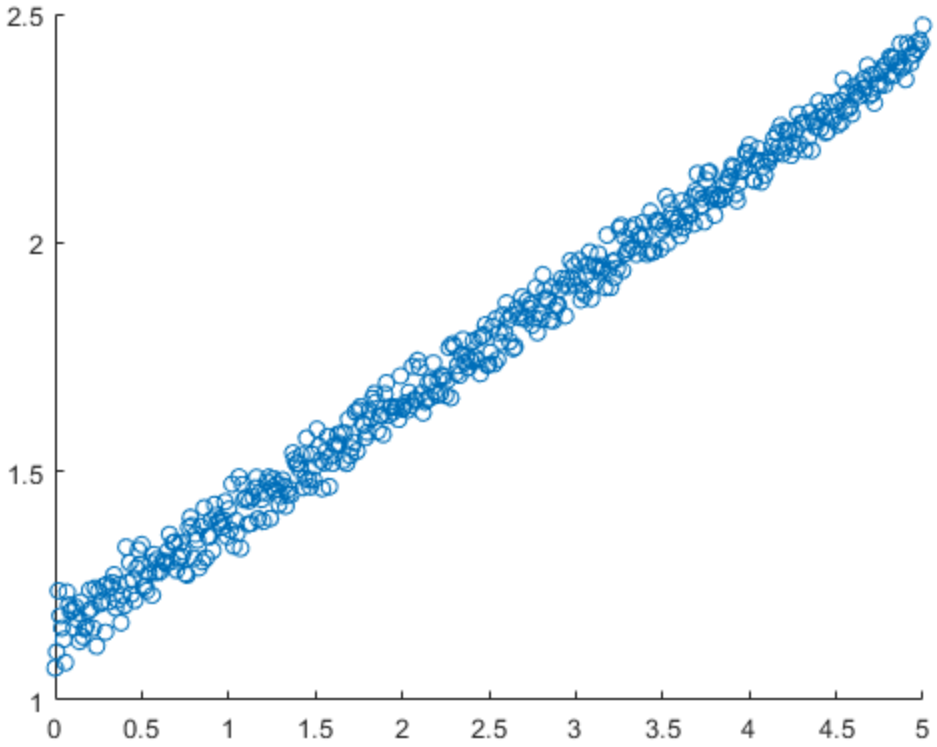


f2 represents the equation of the ordinary line of best fit to data2. This line is not a particularly useful model fit as the shape of the data do not suggest that the hidden function is linear.

*Okay it turns out that data2 is pretty close to some exponential function, y = a*b^x. Unfortunately this isn't linear. But y'know what is? log(y) = x*log(b) + a. So let's find that.*

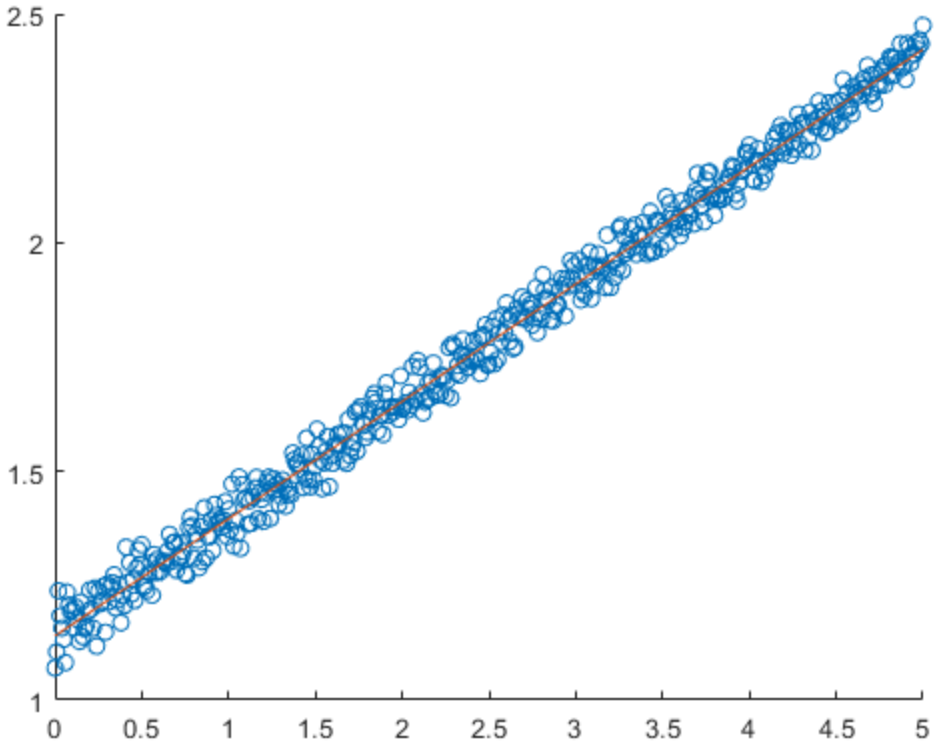
Plot the first column against the log of the second column. *This is called a semi-log plot. (It's common in biostatistics, where exponential growth is the norm.)*

```
figure;
scatter(data2(:,1), log(data2(:,2)))
```



Sweet. ***Do linear regression on that.***

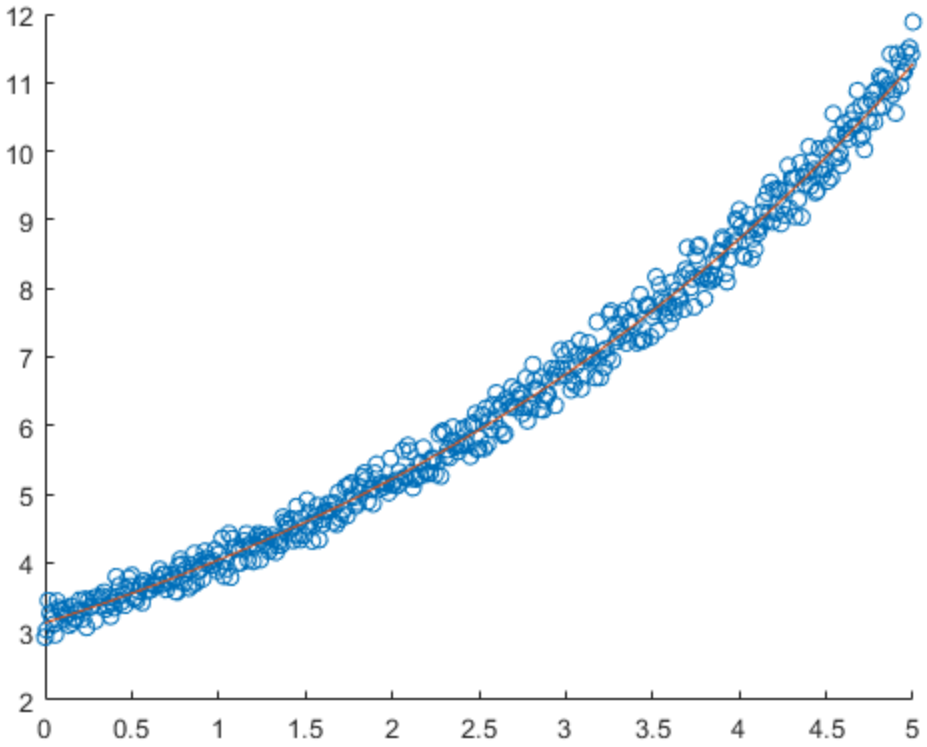
```
inputs2log = data2(:,1); outputs2log = log(data2(:,2));
X2log = [subs(functionone,x,inputs2log) subs(functionx,x,inputs2log)];

Xtx2log = X2log' * X2log;
Xty2log = X2log' * outputs2log;

b2log = Xtx2log^(-1) * Xty2log;
f2log = [1 x]*b2log;

figure;
scatter(data2(:,1),log(data2(:,2)))
hold on
fplot(f2log,[0 5])
```



The linear regresion is the same procedure but using log(data2(:,2)). This model fits a line to the linear relationship observed in the semi-log plot.

Using your linear regression, ***find a curve f2 that closely follows data2. Plot data2 and f2 on the same axes.***

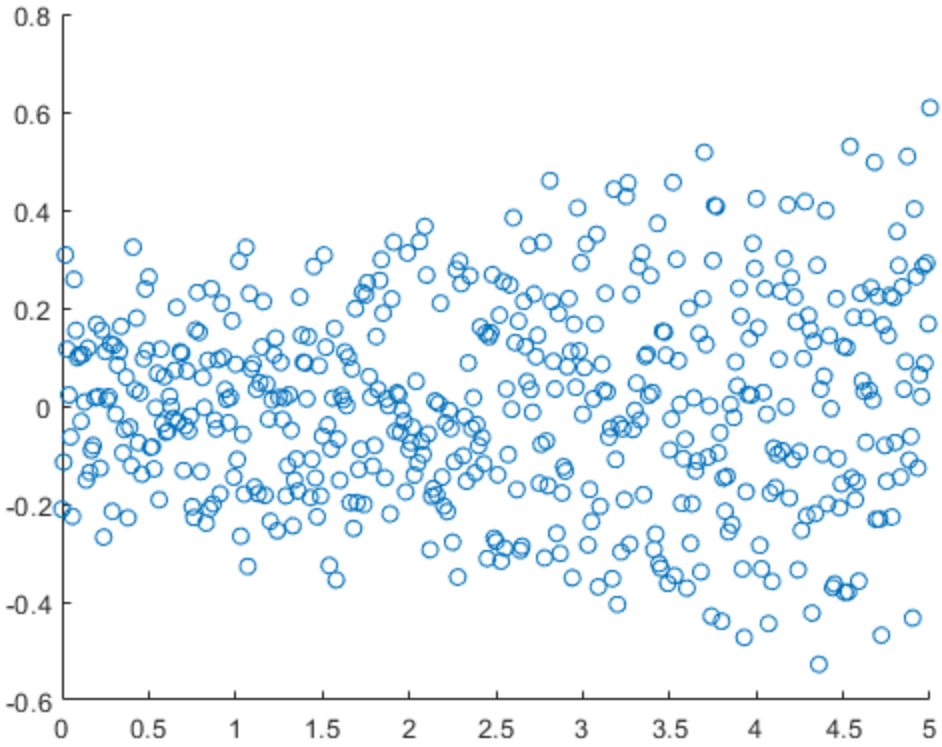
```
figure;
scatter(data2(:,1),data2(:,2))
hold on
f2 = exp(f2log);
fplot(f2, [0 5])
```



The exponentiation of the function f2log fits a curve of the form y = a * b^x to the original data2.

While this might look pretty good, using a linear fit to model exponential growth is often dangerously misleading. ***Show me a scatterplot of data2's inputs against the residual error.*** *If you've done this right, the residuals should get worse over time. This is because a log scale compresses large y-values into a much smaller range than they actually are, so small errors on a log plot correspond to massive real-world prediction errors. (This is why infectious disease models have such huge error bars; they're trying their best to predict the exponential future. The difference between 100K and 1M is only a difference of 1 on a log_10 scale.)*

```
residual = outputs2 - double(subs(f2,x,inputs2));
figure;
scatter(inputs2,residual)
```

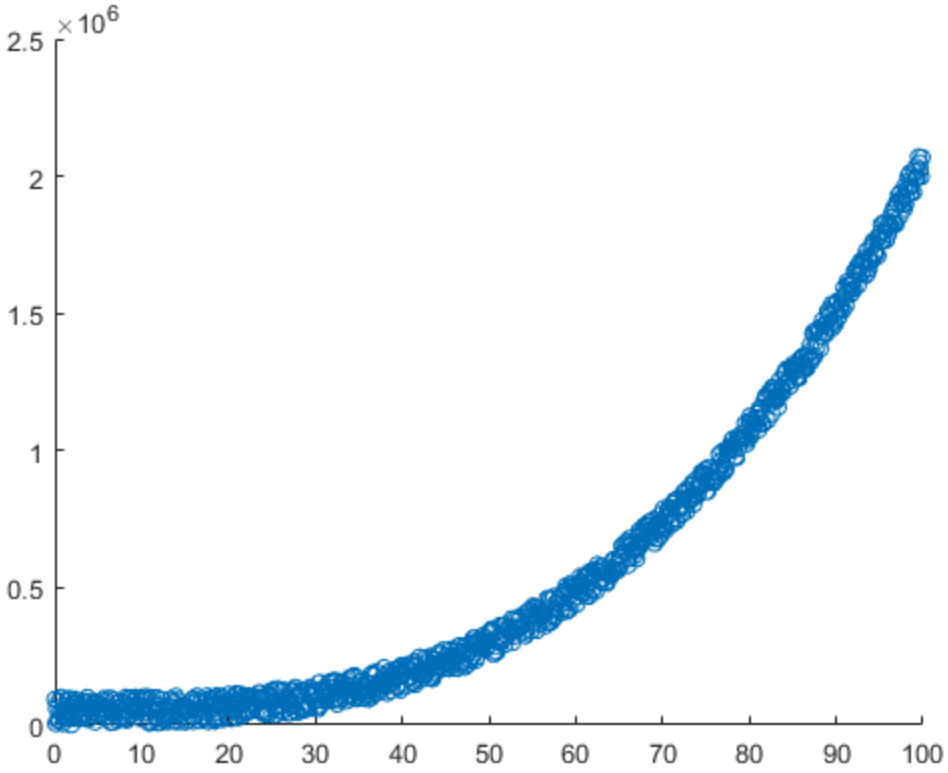


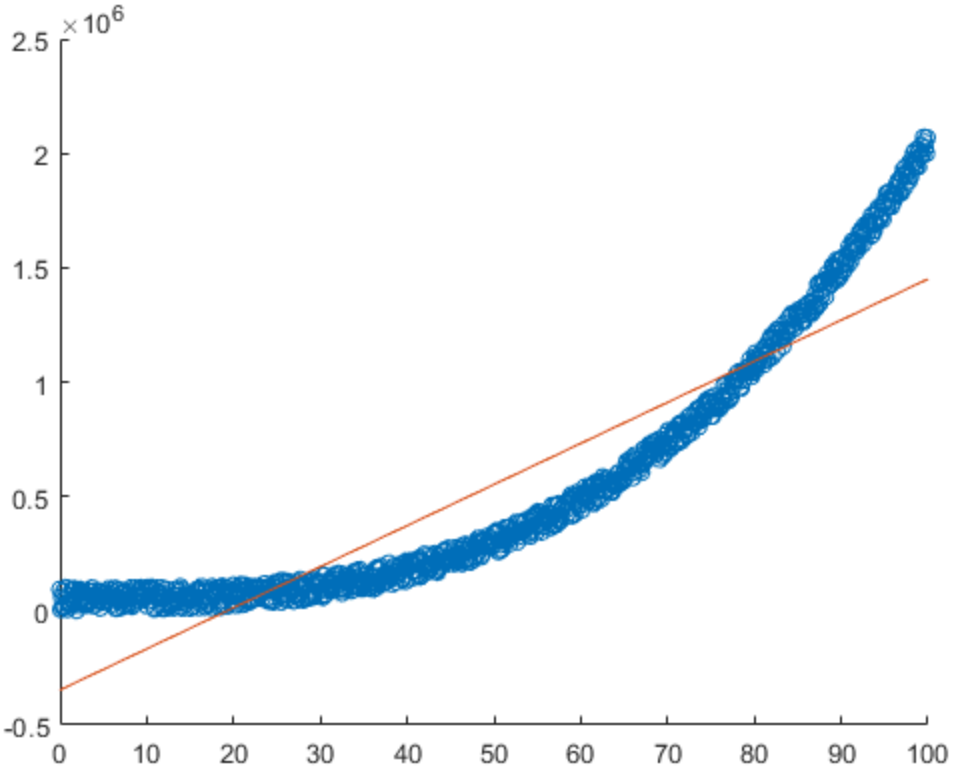
The residuals are plotted below. It is clear that the model demonstrates heteroschedasticity, as the residuals blow up at higher values. This is a violation of the assumptions of linear regression.

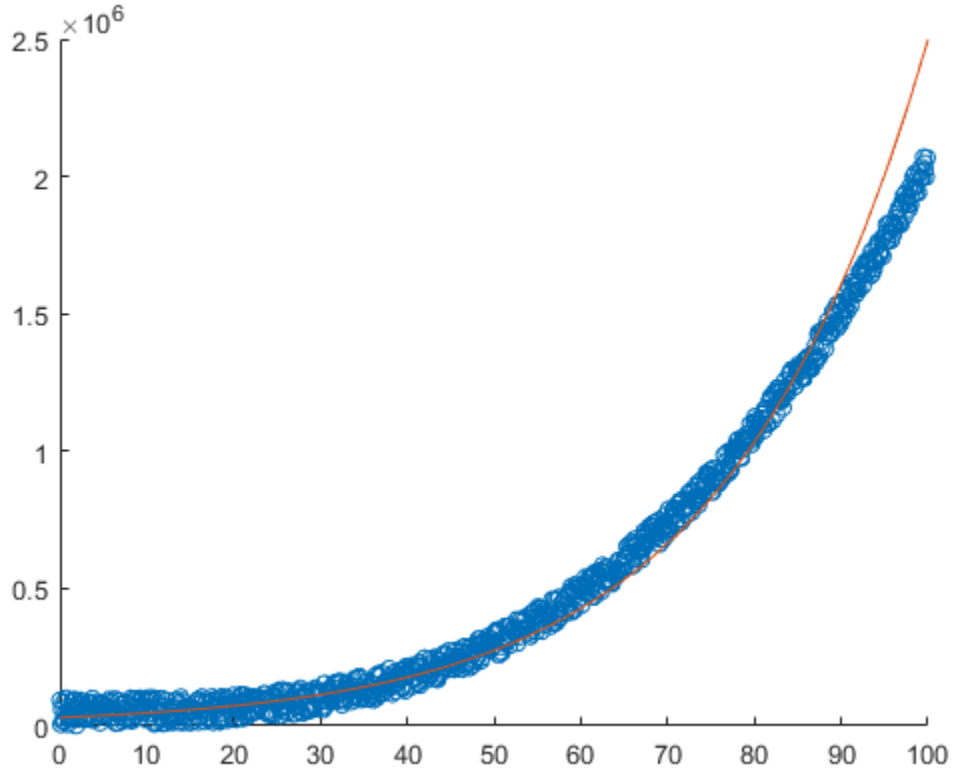
Import data3 and have a look at it. Verify that data3 doesn't fit either a linear or an exponential model particularly well.

```
data3 = csvread("data3.csv");
figure;
scatter(data3(:,1),data3(:,2))

% Linear Model
inputs3 = data3(:,1); outputs3 = data3(:,2);
X3 = [subs(functionone,x,inputs3) subs(functionx,x,inputs3)];

Xtx3 = X3' * X3;
Xty3 = X3' * outputs3;

b3 = Xtx3^(-1) * Xty3;
f3 = [1 x]*b3;

figure;
scatter(data3(:,1),data3(:,2))
hold on
fplot(f3,[0 100])

% Exponential Model
inputs3log = data3(:,1); outputs3log = log(data3(:,2));
```

```
X3log = [subs(functionone,x,inputs3log) subs(functionx,x,inputs3log)];

Xtx3log = X3log' * X3log;
Xty3log = X3log' * outputs3log;

b3log = Xtx3log^(-1) * Xty3log;
f3log = [1 x]*b3log;

figure;
scatter(data3(:,1),data3(:,2))
f3exp = exp(f3log);
hold on
fplot(f3exp,[0 100])
```




×10^6
2.5
2
1.5
1
0.5
0
-0.5
0
10
20
30
40
50
60
70
80
90
100

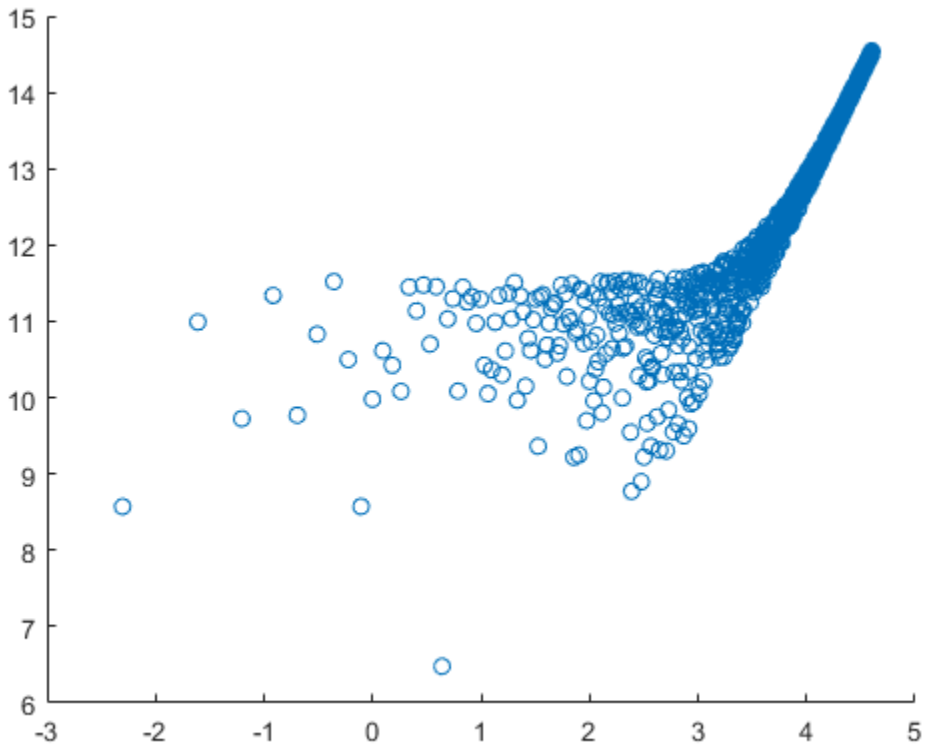

×10^6
2.5
2
1.5
1
0.5
0
0
10
20
30
40
50
60
70
80
90
100

The resulting linear and exponential models do not fit the data particularly well. The exponential model was found using the semi-log data as in the previous section.

In fact data3 is pulled from a polynomial of degree ***n*** *for some number* ***n****. But what's* ***n****? Well, if you have a function y=a*x^n and you take log of BOTH sides, you get log(y) = n*log(x)+log(a). When x is really big, the lower-order terms of the polynomial are small and drop out.*

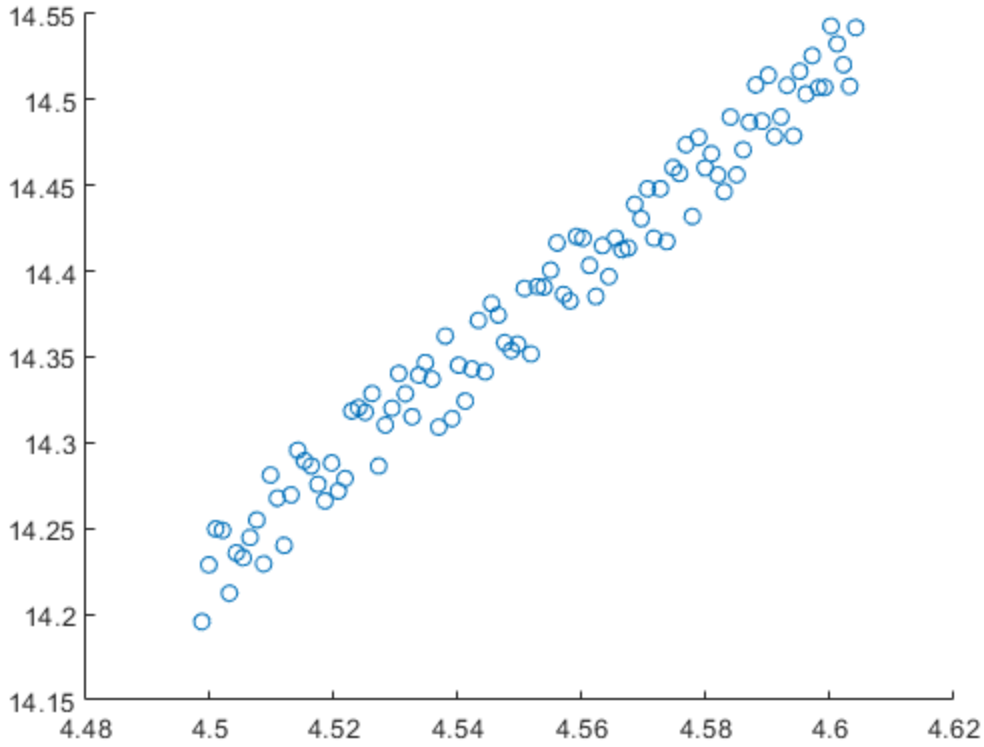
Plot log of the first column against log of the second column. *This is called a log-log plot, and it's very common in economic modeling.*

```
figure;
scatter(log(data3(:,1)), log(data3(:,2)))
```



Like most real-world data, this one's pretty noisy. ***Truncate the data so that you only see the linear chunk at the end and plot that.*** *This requires a bit of a judgment call as to where you should truncate.*

```
data3denoise = log(data3(end:-1:end-100,:)); % LAST 100 ROWS
figure;
scatter(data3denoise(:,1),data3denoise(:,2))
```

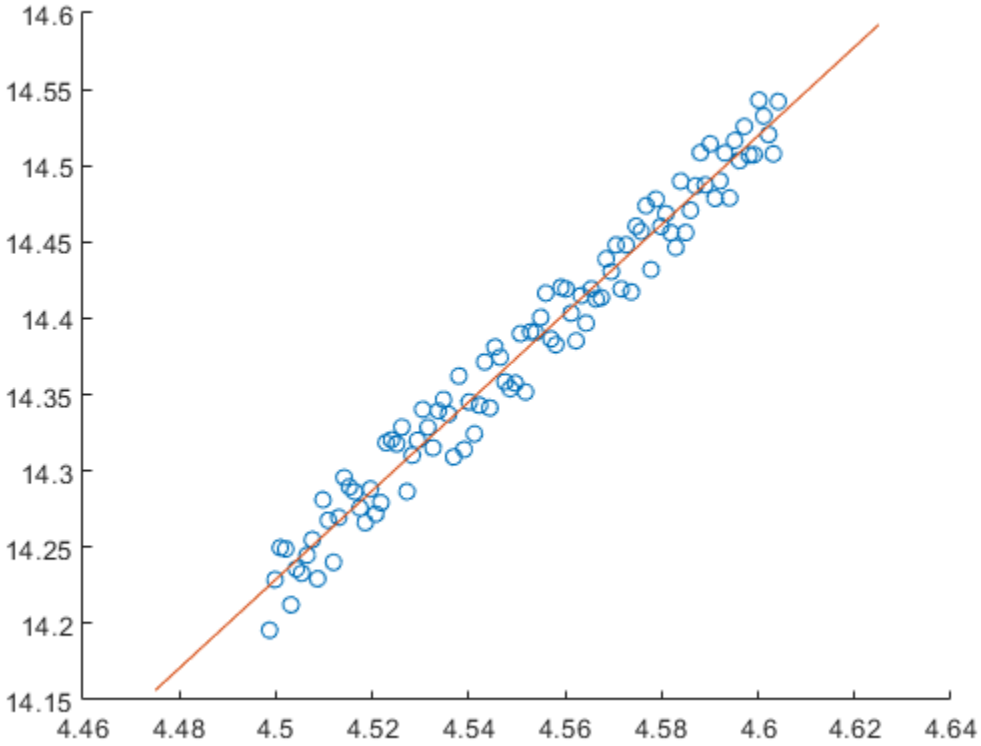


The last 100 entries were selected as they show a very linear relationship.

Do linear regression on that. Then find the degree and leading coefficient of a polynomial f3 and plot it against data3. *Even though this isn't the "exact" polynomial, real-world data is often better estimated by just using the leading term. Retainin lower-order information is almost always what's called "overfitting," which is great at predicting the past and terrible at predicting the future. Anyone can predict the past.*

```
inputs3denoise = data3denoise(:,1); outputs3denoise =
 data3denoise(:,2);
X3denoise = [subs(functionone,x,inputs3denoise)
 subs(functionx,x,inputs3denoise)];

Xtx3denoise = X3denoise' * X3denoise;
Xty3denoise = X3denoise' * outputs3denoise;

b3denoise = Xtx3denoise^(-1) * Xty3denoise;
f3denoise = [1 x]*b3denoise;

exp(1.148) % Had to hardcode b3denoise(1,1) as exp() refused to work
 on the enormous fraction.

figure;
scatter(data3denoise(:,1),data3denoise(:,2))
hold on
fplot(f3denoise,[4.475 4.625])
```

```
ans =

    3.1519
```



The degree of the polynomial is the slope of the line of best fit to the log-log data. The slope of the best fit line is approximately 3, which would indicate that the degree of the polynomial is 3. The leading coefficient is the exponentiation of the intercept of the line of best fit to the log-log data. The exp() of the intercept in this case is approximately 3.1517, so the leading coefficient on the polynomial is 3.1517.

Published with MATLAB® R2019b