The following MatLab code was prepared by Eric Sullivan, Carroll College. The comments explain exactly what the code does.

```matlab
clear; clc; clf;
% This is an example of how to use MatLab to make a parameter estimation.
% This requires the optimization toolbox.

% I'm going to build some noisy data on the function y(x) = 2 e^(-0.1x).
x = 0:1:20;
y = 2*exp(-0.1*x)+0.2*rand(size(x));

% This is the code to do the parameter estimation.
% This first line defines the function where k(n) is the n-th parameter to
% be estimated.
ProposedFunction = @(k,x) k(1)*exp(k(2)*x);
% This next line does the heavy lifting to perform the least squares fit.
% The first argument is the function defined above. The next argument is
% the collection of initial guesses for the parameters. The last two
% arguments are the x and y data sets.
[o1,o2,o3] = lsqcurvefit(ProposedFunction,[1,-1],x,y)
% output 1 (o1) is the collection of parameters
% output 2 (o2) is (maybe) the sum of squares
% output 3 (o3) is the collection of all of the residuals

% Now we'll plot the data along with the fit on the left and the residuals
% on the right.
xx = 0:0.01:20;
FitFunction = ProposedFunction(o1,xx);
subplot(1,2,1)
plot(x,y,'ro',xx,FitFunction,'b--')
legend('Data','Fit')
subplot(1,2,2)
plot(x,o3,'k*')
```
We show the same process in Mathematica.

We offer some toy model with noise.

\[
f(x) = 2 \exp[-0.01 \times] + 0.2 \times \text{Random[]}
\]

We generate data with this noisy model.

\[
data = \text{Table}[\{x, f[x]\}, \{x, 0, 20, 1\}];
\]

We plot the data.

\[
dataPlot = \text{ListPlot}[\text{data}, \text{PlotRange} \to \{\{0, 21\}, \{0, 3\}\}, \text{AxesLabel} \to \{\text{x}, \text{f(x)}\}]
\]

We seek to find the parameters for a model which will fit this toy data. We have two parameters to estimate, \(k_1\) and \(k_2\).

\[
\text{model}[\_\_] = k_1 \times \exp[-k_2 \times]
\]

\(e^{-k_2 x} \times k_1\)

We form the sum of square errors between our data (\(\text{data}[\_][2]\) for \(i^{th}\) data point, the second coordinate) and our model prediction (\(\text{model}[\text{data}[\_][1]]\) for the value of the model at the \(i^{th}\) data point, the first coordinate).

\[
\text{SSE}[k_1\_, k_2\_] = \text{Sum}[(\text{data}[\_][2] - \text{model}[\text{data}[\_][1]])^2, \{i, 1, \text{Length[\text{data}]}\}]
\]

We use Mathematica's \text{FindMinimum} command to determine the parameters \(k_1\) and \(k_2\) which minimize the sum of square errors between our data and our model. We call the result \text{sol}. \text{sol} shows the actual minimum value of the sum of square errors as well as returns the parameter values, \(k_1\) and \(k_2\), which give that minimum.
sol = FindMinimum[SSE[k1,k2],{k1,1},{k2,1}]

\[7.13487 \times 10^{-7}, \{k1 \to 2.04036, k2 \to 0.00978149\}\]

We substitute the values of our parameters k1 and k2 (as found in the second set of rules in sol) into our model to have a final model final(x).

final[x_] = model[x]/.sol[[2]]

2.04036 e^{-0.00978149 x}

We plot the resulting final (x) model.

finalPlot = Plot[final[x],{x,0,20},PlotStyle->{Thickness[.001],Hue[.9]},PlotRange->{0,21},

Finally, we compare our final best fit (in terms of minimizing our sum of square error function) model with our data.

Show[finalPlot, dataPlot]