

Parameter Identification for Radioactive Decay

Radioactive decay can be modeled by a differential equations for the amount of radioactive material, $u(t)$, and this has the following form

$$u' = -d u, \text{ with } d = \text{the decay rate and } u(0) = \text{initial amount.}$$

In the basic calculus course the d and $u(0)$ are given and the solution of this differential equation is

$$u(t) = u(0) e^{-dt}.$$

The objective of this lesson is to determine $u(0)$ and d based on a number of observations of the amounts, u_i , at a particular times, t_i , where $i = 1, \dots, n$ and n is the number of observations. We would like to choose $u(0)$ and d , called parameters, so that for all i

$$u_i = u(0)e^{-dt_i}.$$

If there are more than two observations, $n > 2$, this may not be possible. We hope to be able to find d and $u(0)$ so that all the differences

$$u_i - u(0)e^{-dt_i}$$

are as small as possible. One way to do this is to find $u(0)$ and d so that the least squares function is a minimum:

$$\min_{u(0), d} \sum_{i=1}^n (u_i - u(0)e^{-dt_i})^2.$$

This can often be done by a search algorithm, which is implemented in Matlab by `fminsearch.m`

Outline of Parameter Identification via Matlab's `fminsearch`.

1. Function definitions.
 - a. Find the function and its parameters....you may need to use `dsolve`.
 - b. Create a Matlab function file, `sol`, for this as a of function its independent variable and its parameters.
 - c. Create a Matlab function file, `lsfct`, for the least squares function
2. Execute `fminsearch`.
3. Use the output, the numerical values of the parameters, of `fminsearch`.
 - a. Put the parameters into the function in 1a.
 - b. Create a table of values for this function.
 - c. Create a graph of this function.

1a. EDU>> dsolve('Dy = -d*y','y(0)= y0')
ans =
y0*exp(-d*t)

1b. Contents of sol2.m
function r =sol2(x1,x2,t)
r = x1*exp(-x2*t);

1c. Contents of lsfct2.m
function lsr2=lsfct2(xx)
t = [10 11 12 13 14 15 16] ;
amount = [101 95 90 86.5 82 79 78.2];
lsr2 = 0;
for i=1:7
lsr2 = lsr2 + (amount(i)-sol2(xx(1),xx(2),t(i)))^2;
end

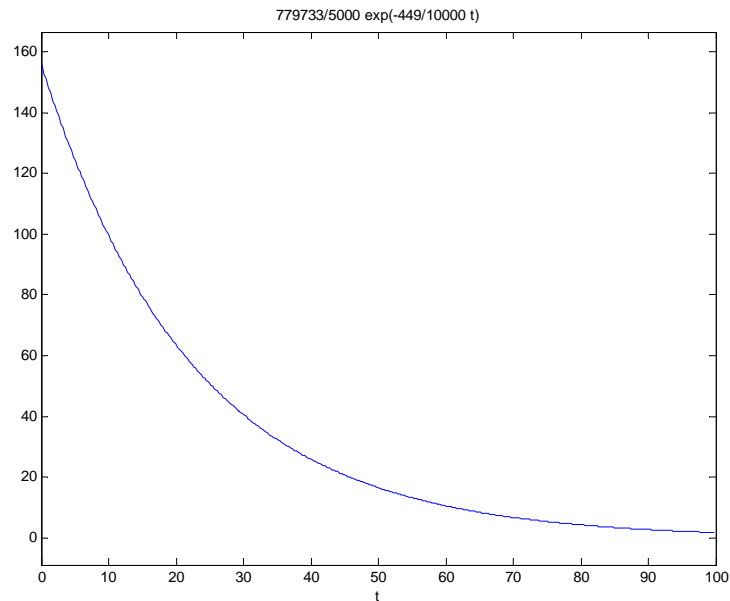
2. Execute fminsearch
EDU>> [xx fctval] = fminsearch('lsfct2', [120 .03])
xx =
155.9466 0.0449
fctval =
9.7357

So, the initial amount is 155.9466, the decay rate is 0.0449, and for these values the least squares function has a minimum value equal to 9.7357.

3a. EDU>> amountfct = dsolve('Dy = -.0449*y','y(0) = 155.9466')
amountfct =
779733/5000*exp(-449/10000*t)

3b. Form the tables of times, computed amounts and amount data
EDU>> times = 10:16;
EDU>> for i = 10:16
amount(i) = subs(amountfct,'t',i);
end
EDU>> amount_data = [101 95 90 86.5 82 79 78.2];
EDU>> table = [times' amount(10:16)' amount_data']
table =
10.0000 99.5354 101.0000
11.0000 95.1651 95.0000
12.0000 90.9867 90.0000
13.0000 86.9918 86.5000
14.0000 83.1722 82.0000
15.0000 79.5204 79.0000
16.0000 76.0289 78.2000

3c. Graph the amount function
EDU>> ezplot(amountfct, [0 100])



By using the graph one can predict the amounts of radioactive material at future times. For example, at time equal to 50 the radioactive material is about equal to 20. The accuracy of this modeling process can be critiqued at a number of points. First, the measured data may have errors; here the reader may find it interesting to experiment with some variations of the measured data. Second, `fminsearch` is an implementation of the Nelder-Mead search algorithm, which may not always give accurate "solutions"; the interested reader should consult the Matlab help file on `fminsearch`. Third, there is the possibility of a number of choices of the parameters that could minimize the least squares function. The following Matlab code, `graphlsfct2.m`, will generate 3D and contour plots for the about least squares function as a function of the two parameters for the initial amount and decay rate. Note the long valley in the graph of this function, that indicates a number of possible minimizers.

```

begin1 = 150;
end1 = 160;
begin2 = .03;
end2 = .05;
n1 = 60;
n2 = 60;
for j = 1:n2
    x2(j) = begin2 +(j-1)*(end2-begin2)/n2;
    for i = 1:n1
        x1(i) = begin1 +(i-1)*(end1-begin1)/n1;
        u(i,j) = lsfc2([x1(i) x2(j)]);
    end
end
end
%mesh(x1,x2,u)
contour(x1,x2,u',[8 9 9.5 10 10.5 11 12])

```

