Laboratory 6: System Identification (Parameter Estimation)

Part 1: Introductory material

Recall the Lotka-Volterra model we have seen in Lab #1. The (nonlinear) differential equations describing the dynamic model are:

$$\frac{dH(t)}{dt} = rH(t) - aH(t)P(t)$$

$$\frac{dP(t)}{dt} = bH(t)P(t) - mP(t)$$

where the two state variables H(t) and P(t) represent the density (amount) of prey and predators respectively. The model includes intrinsic and interaction parameters.

It is an interesting and potentially very useful exercise to determine the optimal parameter values for a certain dataset. This "model training" process is *system identification*, in that a model is fit to real data in an attempt to determine under which circumstances that model is a good representation for the real phenomenon. Data have been collected over nearly a century by the Hudson Bay Company about the population dynamics of Canadian lynx and hare. A portion of the data is follows (as reported in the instructional website http://www-rohan.sdsu.edu/~jmahaffy/courses/f00/math122/labs/labj/lotvol.xls):

Year	Hares	Lynx
1900	30	4
1901	47.2	6.1
1902	70.2	9.8
1903	77.4	35.2
1904	36.3	59.4
1905	20.6	41.7
1906	18.1	19
1907	21.4	13
1908	22	8.3
1909	25.4	9.1
1910	27.1	7.4
1911	40.3	8
1912	57	12.3
1913	76.6	19.5
1914	52.3	45.7
1915	19.5	51.1
1916	11.2	29.7
1917	7.6	15.8
1918	14.6	9.7
1919	16.2	10.1
1920	24.7	8.6

The task at hand is to fit the Lotka-Volterra model to these data, so as to determine optimal parameter values r, a, b and m for this particular case of population dynamics.

This can be accomplished using the MATLAB program `fminsearch` (see `help fminsearch` for more details). The following MATLAB script accomplishes this goal:

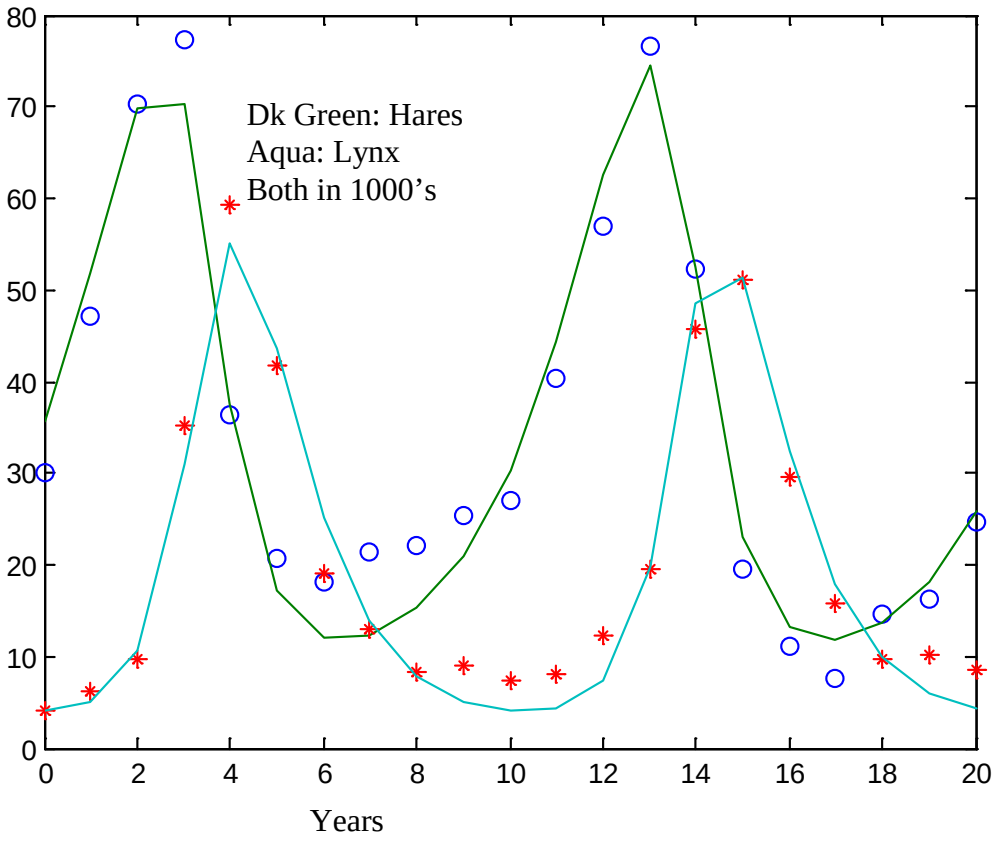
```
function identification
```

```
 clear all; close all;
% Data are from
% http://www-rohan.sdsu.edu/~jmahaffy/courses/f00/math122/labs/labj/lotvol.xls
% Year   Hares (x1000)   Lynx(x1000)
global data
data=...
[1900   30  4
1901    47.2    6.1
1902    70.2    9.8
1903    77.4    35.2
1904    36.3    59.4
1905    20.6    41.7
1906    18.1    19
1907    21.4    13
1908    22  8.3
1909    25.4    9.1
1910    27.1    7.4
1911    40.3    8
1912    57  12.3
1913    76.6    19.5
1914    52.3    45.7
1915    19.5    51.1
1916    11.2    29.7
1917    7.6 15.8
1918    14.6    9.7
1919    16.2    10.1
1920    24.7    8.6];
% guess the parameters [r a b m] and initial conditions[hares lynx]:
guesses = [1.00 0.02 0.02 1.00 35.7 3.99];
% optimize the parameters and initial conditions:
[estimates, J] = fminsearch(@obj,guesses);
disp(estimates);
% graph the model behavior using these optimized values:
tspan = data(:,1)-1900;
y0 = [estimates(5) estimates(6)];
options=[];
[t,y] = ode23s(@lotkavolterraODE,tspan,y0,options,estimates);
plot(t,data(:,2),'o',t,y(:,1),t,data(:,3),'*',t,y(:,2))

function J = obj(guesses)
global data
% determine the model behavior with the guesses values:
tspan = data(:,1)-1900;
y0 = [guesses(5) guesses(6)];
options=[];
[t,y] = ode23s(@lotkavolterraODE,tspan,y0,options,guesses);
plot(t,data(:,2),'o',t,y(:,1),t,data(:,3),'*',t,y(:,2)) ; drawnow
% determine the weighted least-squares by comparing model to data:
J = sum(((data(:,2)-y(:,1))).^2)+sum(((data(:,3)-y(:,2))).^2)
% note that the above function assumes a certain error model.

function dydt = lotkavolterraODE(t,y,p)
% ode for the model:
dydt = [ p(1)*y(1)-p(2)*y(1)*y(2)
         p(3)*y(1)*y(2)-p(4)*y(2) ];
```

The following items are worth a mention:

a) Note how the estimated initial conditions for the ODE system are slightly different than the first data value, to accommodate the past history of the dataset. Note that we have chosen to estimate the initial conditions. The initial measurement is a noisy measurement just like all the other time points and should not have a greater impact on model course than the others.
b) The optimal estimate is determined through nonlinear least squares, defined in the function `J`, and both populations are fit simultaneously (so that the optimal parameter value will depend on both time courses);
c) Note also the use of `tspan` in the function `J`: the selection of a full vector `tspan`, as opposed to just a range, allows one to obtain the solution of the differential equation at same time points where the data was measured, instead of having to deal with the variable time step in the output characteristic of MATLAB integrators. This is almost always necessary when fitting model to a data.
d) We have used the stiff integrator `ode23s` to accommodate the nonlinear nature of the system across a wide spectrum of parameter values.

The resulting plot at the optimal parameter estimates is below: the blue circles represent the prey population, and the red stars represent the predator population.
The final value of the objective function is 599.231,
and the optimal parameter vector is: [0.4822 0.0249 0.0276 0.9275 34.7982 3.8320].



You are encouraged to acquire some practice with the estimator (for example, by changing slightly the starting values and inspecting the results).

Part 2: Laboratory assignment

In my lab, we perform experiments where we measure the strength of single receptor-ligand bonds under force. In this 485 lab, you will be analyzing one of our experiments from the paper of the week.

1. **First we consider the experiments**. We pull the bonds one at a time, increasing the force linearly at rate *r* until the bond ruptures, and record the rupture force. We do this hundreds of times in order to get a large number of single events, so that the data may be statistically significant in spite of the stochastic nature of single bonds. We make a histogram of the results that gives the number of attempts that rupture in a particular force bin. Most attempts result in no detectable ruptures either because no bond forms or if one does it is hidden by the noise (< 20 pN), and we do not keep the data in these bins. We assume that the measurable ruptures are due to single bonds, both because Poisson's law of small numbers says that if something happens with probability *p*, it will happen twice with probability p^2, and because two or more bonds are usually visible as two separate rupture force peaks which can be isolated or removed from the data. Thus, we interpret the histograms as single bond rupture forces. In these experiments, we measure bond rupture forces at three different loading rates as indicated below. We also measure the number of ruptures in a control experiment in which specific bonds cannot form due to a soluble inhibitor; we have found that this number is always similar at all loading rates, so we only took one set of data.

bins	300 pN/s	3000 pN/s	30,000 pN/s	negative control
20-40	5	8	4	9
40-60	4	7	5	6
60-80	5	8	4	6
80-100	5	5	4	4
100-120	5	3	6	3
120-140	14	7	5	3
140-160	25	11	8	3
160-180	9	27	8	3
180-200	2	9	20	2
200-220	2	3	11	1
220-240	1	2	4	1
240-260	1	0	1	2
260-280	1	0	2	1
280-300	0	0	2	1

Graph the four sets of data. You can use a bar graph, but it is also fine just to graph with line graphs as usual, using the middle of the force bins as the values for the x-axis.

2. **Next we consider the model and how to compare it to the data.** The specific bonds dissociate with a first-order rate constant and so can be modeled with a simple first-order ODE: $\frac{dS_i(t)}{dt} = -k \cdot S_i(t)$, where $S_i(t)$ is the expected number of bonds surviving until time t, and we use the subscript to indicate the different test conditions (different loading rates). However, the rate constant depends on force: $k(f) = \mathrm{k} \cdot \exp(f/f_s)$, where k is the dissociation rate without force and f_s is the force sensitivity of dissociation. Recalling that force increases linearly ($f = r_i t$, with a different r_i for each condition *i*), we get: $\frac{dS_i(t)}{dt} = -\mathrm{k} \cdot \exp(r_i t/f_s) \cdot S_i(t)$. At this point, we have to further manipulate the model to compare it to data, which is almost always the case in real

applications. To translate the model to one where force is the independent variable instead of time, multiply both sides by of each ODE by $\frac{dt}{df} = \frac{1}{r_i}$, and $tr_i = f$ (both from the definition of the loading rate), to get a new ODE in force:

$$\frac{dS_i(t)}{df} = -\frac{\mathrm{k}}{r_i} \cdot \exp\left(\frac{f}{f_s}\right) \cdot S_i(f)$$

Recall that $S_i(f)$ for the expected number of a bonds *surviving* until the force has reached f, but the data is the number of bonds *that rupture within a designated range of forces*. Thus, we use the diff command to calculate the probability of rupture in the bin from f to $f + \Delta f$

$$R_i(f, f + \Delta f) = S_i(f) - S_i(f + \Delta f)$$

Also remember that you do not know $S_i(0)$ since some bonds may have broken in the range where you could not detect bond ruptures, and because the measurements are a noisy representation of the probability distribution. Finally, remember that there are three datasets with different loading rates r that you must fit simultaneously. So you will also need to estimate the value of the three initial conditions. *(Note: you could solve this simple ODE analytically, and then fit the algebraic rather than numeric ODE solution model to the data, but please don't do this; you need practice using the ODE solver together with the fitting algorithm, and it will take you extra time to do the analytic solution).*

3. **Solve the model with initial guesses**: Use the following wild guesses: $k = 20\ \mathrm{sec}^{-1}$ and $f_s = 30$ pN. Use the sum of the number of ruptures in condition i as your guess of $S_i(0)$, the total number of bonds in each condition. Solve the ODE model. Plot it on the same graph as the data to compare the two. The model should not fit the data, because you have not performed parameter estimation.

4. **Estimate the parameters ignoring the negative control**. Using the model in Part 1 as a template, fit the model simultaneously to the three sets of experimental data (ignore the control for now), using the initial estimates from given above. Assume a constant standard deviation for the errors for your weighting scheme. Plot the data and model prediction versus force and visually inspect the result. Also note the value of the objective function that corresponds to this fit. Try several different initial guesses to see if you can get some guesses that lead to a fit that is clearly wrong and at least two different sets of initial guesses that lead to a reasonable fit the data. *(Hint: Alter the k and f_s guesses. If the first guesses were way off, change them dramatically.*

    a. Plot the data and model on the same graph for the best set of parameters you found. Explain how you know which set of parameters was best.
    b. What are the parameters and initial conditions and are they physiologically possible? If these values are not reasonably possible, do something to change this, and try again until they are and explain what you did. (this can vary based on the guesses used, so you should always ask this.)
    c. Determine and plot the residuals and/or weighted residuals and use them in some way to evaluate your weighting scheme and the model performance. How did the model perform? Could any specific areas of the fit be improved? Be sure to state how you draw your conclusions.

5. **Address the negative control data**. There are several ways to address the negative control data, which is due to nonspecific adhesion between the proteins and surfaces in the experiment. One method is to subtract the control data from the experimental data before fitting. However, we will use an alternative method, which is to simultaneously fit a model to your control data, which is

added to the experimental model. Because the control data looks approximately like an exponential decay, we will use a decay model as an empirical model (meaning we don't know the underlying mechanism). Thus, we use an algebraic equation: $N(f) = N_0 \exp(-af)$, where both N_0 and a are the same for all four data sets and will be determined by parameter estimation. Note that this exponential decay of rupture number with force is very different from the exponential effect of force on the unbinding rate, and it is an algebraic rather than an ODE expression for the bond survival. Now fit the four sets of data simultaneously, fitting the control data to $R_4(f, f+\Delta f) = N(f) - N(f+\Delta f)$, while fitting the three experimental sets to: $R_i(f, f+\Delta f) = N(f) + S_i(f) - N(f+\Delta f) - S_i(f+\Delta f)$.

a. Fit the data using this model, show your fit plot, state the new parameters, the value of the objective function, and show the residuals and/or weighted residuals. *Hint: make sure you try enough initial guesses to get the best estimate for this model. To get an initial guess for a, note that the negative control data decays to ½ by about 100 pN, so a ~ 1/100 = 0.01.*
b. How can you decide whether this approach is a better way to fit the data, and what do you conclude? *Hints: How many parameters did you fit for this model versus the original model? How many parameters per data set? Can you compare the objective functions to decide which model is better? Can you use the (weighted) residuals?*

6. **Use a different weighting scheme, with the part 5 model**. Now you realize that the data is a counting process, and the major source of noise may be the random, small numbers. You assume the *variance* (σ^2) is proportional to the data or model value.
   a. Plot the model and data using the new weighting scheme
   b. What are parameters and initial conditions? How are they different from before?
   c. Plot the residuals or weighted residuals. How did the distribution of the residuals or weighted residuals change?
   d. How did the value of the objective function (J) change when you changed the weighting?
   e. What information tells you whether your new weighting scheme is better or worse and what conclusion do you draw? Did this change address any concerns you raised in part c? Can you think of any weighting schemes that might be better?

HINTS:

1. When you want to match an ODE result to a set of data at defined time points, you can pass the ODE solver a tspan (or fspan) that matches your data that you will eventually compare.
2. Your optimization will run much faster if you don't print anything out (figures included) during the process. But, you will not be able to monitor it. So, you may want to print at some points of trouble shooting and not at others. You can pass a maximum number of iterations to fminsearch, which will cause it to run quickly but imperfectly while you to debug what is done with the output.
3. Use the function diff to take the finite difference of a vector. To approximate dy/dx, use diff(y)./diff(x). Be careful when you have matrices.