Estimating the values of parameters

If and as you *use* your mathematical model, you're going to need to determine the values of the parameters.

There are primarily two ways of going about this although you might also do a mixture of the two; you determine some of the parameters by one method and verify them by the other method, or use the first method on some of the parameters and the second method on the rest of the parameters.

1. If you understand the physical meaning of the parameters you may be able to devise experiments to measure those values directly or you may have historical data indicating what their values are. This is, perhaps, the most satisfying way to determine the values of the parameters.

2. You may be able to observe the process evolving and gather data on its evolution. In the deterministic case, you search for those values of the parameters that make the model match the data most closely. In the stochastic case, there are different techniques for estimating the parameters, the most common of which is to search for those values of the parameters that make the data most likely.

The first method is particular to your model. There's not much we can say about it in general.

Remember though, you must be very careful that you have interpreted the parameters correctly; you need to ensure that the quantities you are measuring are, indeed, the values of the parameters.

We'll explore here how you can go about the second method.

Estimating Parameters in the Case of a Deterministic Model

- Let $\Omega \subset \mathbb{R}^K$ denote parameter space (so there are K parameters and each entry in Ω is a vector whose components are the values of all the parameters in the model).
- Let $\delta : \Omega \to \Delta$ denote the theoretical values of the data that are observed. (In other words, $\delta(\omega)$ constitutes the values predicted by the model when the parameters are ω.) The space Δ could be a variety of different kinds of objects. For example, it could be a single number that is observed at the 'completion' of the process or it could be the values of the variables themselves, some subset of the variables, or some function of the variables at a discrete set of points in time.
- Let $D \in \Delta$ denote the data that are collected.

We'd like to find $\omega \in \Omega$ such that $\delta(\omega) = D$. However, in general there is no such ω. This is because

a) the model is only an approximation of reality and

b) our measurements are almost certainly not exact.

So what we do instead is find $\omega \in \Omega$ such that the 'distance' between $\delta(\omega)$ and D is as small as possible.

What do we mean by 'distance'?

We need to choose a metric d on Δ. (In other words, d is a notion of the distance between two elements in Δ.) How can we choose d?

- If you have a *model* for why $\delta(\omega)$ is not exactly equal to D, this can inform your choice of d and may allow you to give *confidence intervals* for the parameter values. For example, this may be the case if you think your model is pretty realistic and the difference between $\delta(\omega)$ and D is primarily due to measurement error.

- The choice of d depends on what the elements in Δ look like; they could be functions, numbers, vectors, or a mixture of these. Most commonly they are vectors. In this case typical choices of d are some kind of L^p-norm:

$$d_1(D, \delta(\omega)) = |D_1 - \delta(\omega)_1| + \ldots + |D_m - \delta(\omega)_m|$$

$$d_p(D, \delta(\omega)) = (|D_1 - \delta(\omega)_1|^p + \ldots + |D_m - \delta(\omega)_m|^p)^{1/p}$$

$$d_\infty(D, \delta(\omega)) = \max_{1 \leq i \leq m} |D_i - \delta(\omega)_i|$$

- In fact, when the elements in Δ are vectors, the default choice is to use the L^2 norm:

$$d(D, \delta(\omega)) = \sqrt{(D_1 - \delta(\omega)_1)^2 + \ldots + (D_m - \delta(\omega)_m)^2}.$$

When we use this metric we are minimizing the *least squares distance* between the actual data D and the data as predicted by the model $\delta(\omega)$.

There are a variety of reason to use this metric, not least of which are this function is amenable to analysis (unlike absolute values and minima) and there is a well-established theory of least squares error analysis in statistics.

However, if different entries in D are measured in different units, there could be compelling reasons to use some kind of weighted L^2 norm for the metric.

Once you have chosen the metric, you have an optimization problem to do: you need to find $\omega \in \Omega$ that minimizes $\phi(\omega) = d(\delta(\omega), D)$.

How difficult it is to do this depends on the nature of the function ϕ which, in turn, depends on δ and the metric d.

If the situation is simple enough and you have a nice formula for the function $\delta(\omega)$ you may be able to do this analytically.

More often, you won't understand the function δ exactly and you'll need to do this optimization problem numerically.

Which numerical algorithm you use depends on the nature of the function ϕ and the nature of the space Ω. Numerical methods can typically involve the evaluation of ϕ at thousands of points $\omega \in \Omega$.

If Ω is a nice subset of $\mathbb{R}^n$ (this is usually the case) and you have a formula for ϕ and can find the derivatives of ϕ then you can use gradient descent methods. These methods can be very efficient.

More commonly ϕ will depend on the values of the variables over time and the only way to determine $\phi(\omega)$ for a given value of ω is to simulate the model, read off the value of $\delta(\omega)$ and thereby calculate $\phi(\omega)$. Methods of doing the optimization in these cases include simulated annealing and the genetic algorithm. This can be exceedingly time consuming.

Example 1: Recall the problem we discussed in the introduction where our goal was to determine the drug dosages for a patient with asthma.

In our model we tracked the concentration of drug c as a function of time t. The parameters in the model were:

- The weight W of the patient in kilograms.
- The dosages d_0 and d in mg.
- The time between doses T_0 and T in hours.
- The decay constant k in hours^{-1}.

This means that parameter space Ω looks like the positive 'sixty-fourth'-tant inside $\mathbb{R}^6$. However, control d, d_0, T and T_0 and we measure W, so the only parameter that we really want to estimate is k. So, effectively our parameter space is $\Omega = \mathbb{R}_+$.

To determine k, data was collected on a patient who weighed 50 kg and was given an initial dose of 300 mg and no other doses. In other words, $W = 50$, $d_0 = 300$, and $T_0 = \infty$. (The values of d and T are irrelevant.) In this case the model predicts that

$$c = c_0 e^{-kt} = \left(\frac{2d_0}{W}\right) e^{-kt} = 12e^{-kt}.$$

The data collected were the concentrations c at times $t = 1, 3, 5, \ldots, 19$. In other words

$$\Delta = \mathbb{R}_+^{10}$$

and the map $\delta : \Omega = \mathbb{R}_+ \to \Delta = \mathbb{R}_+^9$ is

$$\delta(k) = \left(12e^{-k}, \quad 12e^{-3k}, \quad 12e^{-5k}, \quad \ldots, \quad 12e^{-19k}\right).$$

The actual data collected were:

$$D = \Big(10.0, \quad 7.0, \quad 5.0, \quad 3.5, \quad 2.5, \quad 2.0, \quad 1.5, \quad 1.0, \quad 0.7, \quad 0.5\Big)$$

We needed to choose $k \in \mathbb{R}_+$ that minimizes $\phi(k) = d(\delta(k), D)$ for some metric d on Δ. Let's use the least squares metric:

$$\begin{aligned}\phi(k) &= d(\delta(k), D) \\ &= \sqrt{(12e^{-k} - 10)^2 + (12e^{-3k} - 7)^2 + \ldots + (12e^{-19k} - 0.5)^2}\end{aligned}$$

We have a nice formula for ϕ in this case and it is easy enough to find the derivative ϕ'. Despite that, it is still hard to do the minimization problem analytically since it's hard to solve for that value of k where $\phi'(k) = 0$.

However, it's no problem to do the minimization problem in Matlab using, say, `fminsearch`. Since this function is so simple, this is almost immediate and requires only twenty or so function evaluations.

First we create an anonymous function that calculates the value of ϕ. Suppose we call that function `phi`.

```
>> D = [10, 7, ...  , 0.5];

>> phi = @(k) norm(12*exp(-(1:2:19)*k) - D);
```

Then we use `fminsearch` to find the minimum of `phi`. The inputs to `fminsearch` are the following.

- The function to be minimized `phi`.
- An initial value of k. The search algorithm starts evaluating the function at this initial value and keeps looking in a neighborhood of that value, looking for the minimum of the function. When you use `fminsearch` you should try multiple initial values as a test to see how well it's performing. We start with $k = 1$ for lack of better information.

```
>> [k, fval] = fminsearch(phi,1)
```

This returns the value of k at which ϕ is a minimum in the variable `k` and the minimum value of ϕ in the variable `fval`.

```
k =

    0.1722

fval =

    0.3908
```

In fact, since it's easy to find ϕ' we could have used a numerical method that employed this knowledge. However, it wasn't necessary, since this was so fast and appears to be accurate based on searching from different starting values.

If you recall, when we did this example, this was not the value of k that we got. Instead of 0.1722 we got 0.1633.

This was because we used a different metric on Δ!

What we did there was take logs to get

$$\ln c = \ln 12 - kt.$$

Then we used least squares distance on the values of $\ln c$. In other words we had

$$\begin{aligned}\tilde{\phi}(\omega) &= \tilde{d}(\delta(k), D) \\ &= \big((\ln 12 - k - \ln 10)^2 + (\ln 12 - 3k - \ln 7)^2 + \dots \\ &\qquad \dots + (\ln 12 - 19k - \ln(0.5))^2\big)^{1/2}\end{aligned}$$

In this case the function $\tilde{\phi}^2$ is quadratic in k, so the minimization problem can be done analytically.

Which value of k should we use? It's not clear which is better.

- In this case we expect the model to be pretty close to reality, so that the errors will primarily be due to measurement error. The metric d assumes the errors are the same irrespective of the value of c. The second metric $\tilde{d}$ treats the errors as if they were proportional to c; in other words, the second metric tolerates larger values of the errors when c is large. A biologist might be able to tell you which is a better model for the errors.
- Maybe it doesn't matter; the practical difference between 0.1722 and 0.1633 may be negligible. A sensitivity analysis may help you determine if this is the case.

Example: *(A typical (failed) story of parameter estimation.)*

A model for the population of a predator and a prey has the form

$$x' = \alpha x - \beta xy$$

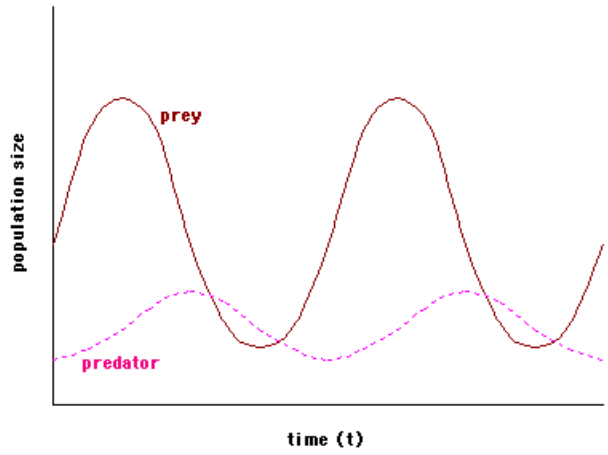
$$y' = -\gamma y + \delta xy$$

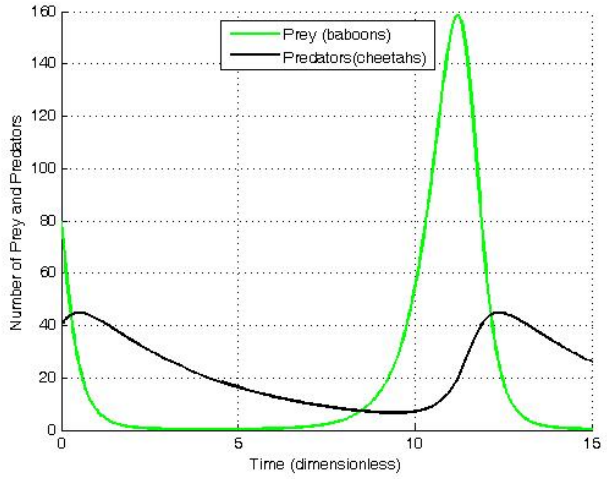
where x is the number of prey, y is the number of predator and α, β, γ and δ are parameters. This model was posited by Volterra in the 1920's to explain the populations of selachians (predator fish) in the years 1914 - 1923.

The assumptions underlying the model are:

- The prey has ample food at all times with a natural percentage growth rate (in the absence of the predator) equal to α.
- The predator depends entirely on the prey as its only food supply. It has a natural percentage death rate of γ.
- Even if the populations are large there are no (other) environmental constraints on their sizes.
- The environment is closed (there is no migration) and the time period is short enough that the environment is unchanging.
- The populations of the predator and the prey are well-mixed.

These equations are hard to solve analytically but can be analyzed qualitatively. Solutions are periodic with the population of the predator peaking a quarter period after the peak of the prey. Here are examples of what typical solutions look like.





The snow shoe hare and lynx populations in Canada form a predator-prey system that might be modeled by this system.

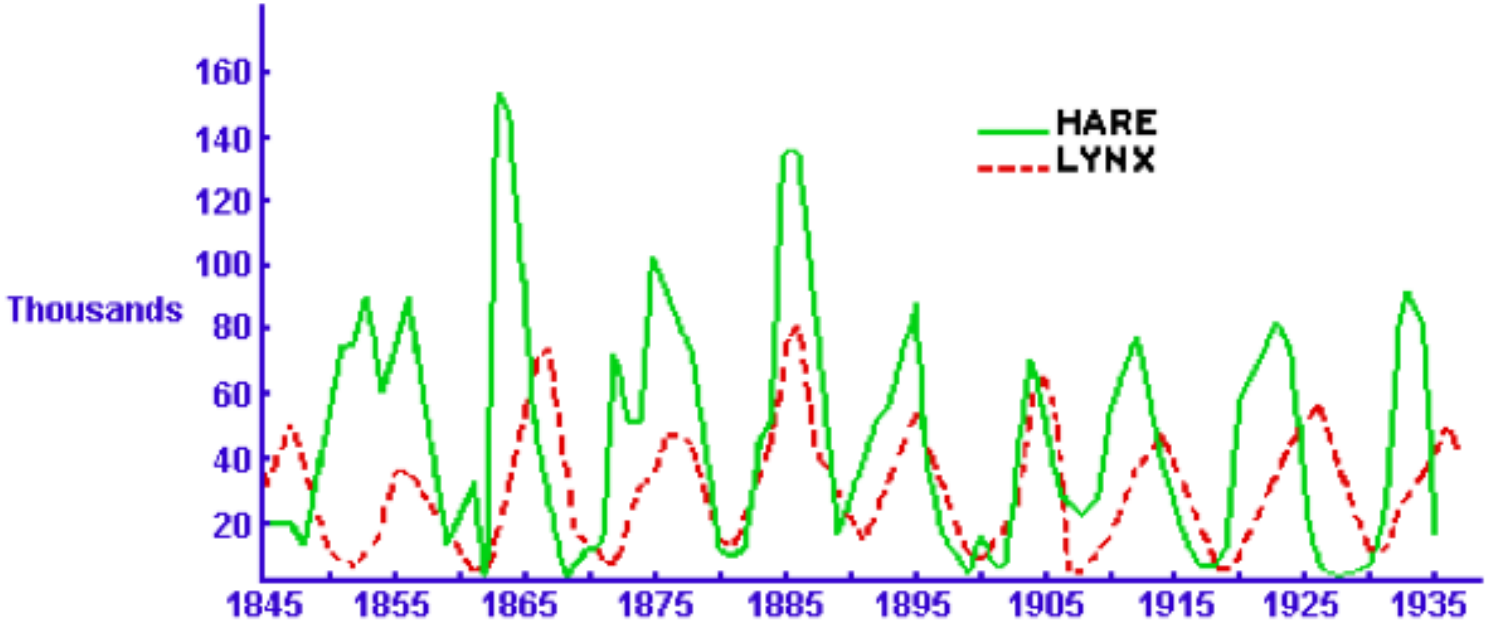
Data of the Canadian hare and lynx populations were collected by the Hudson's Bay Company.

Can we find values of the parameters in the model that provide a match to the data?

Snow Shoe Hare and Lynx Population in Canada (in thousands)

Year	Hare	Lynx	Year	Hare	Lynx
1845	20	32	1893	52	37
1847	20	50	1895	83	50
1849	52	12	1897	18	35
1851	83	10	1899	10	12
1853	64	13	1901	9	12
1855	68	36	1903	65	25
1857	83	15	1905	45	62
1859	12	12	1907	30	49
1861	36	6	1909	25	7
1863	150	6	1911	27	7
1865	110	65	1913	77	20
1867	60	70	1915	25	43
1869	7	40	1917	10	11
1871	10	9	1919	10	6
1873	70	20	1921	46	20
1875	100	34	1923	80	37
1877	92	45	1925	20	43
1879	70	40	1927	8	50
1881	10	15	1929	6	30
1883	11	15	1931	6	15
1885	137	60	1933	20	18
1887	137	80	1935	83	40
1889	18	26	1937	12	48
1891	22	18			

A plot of the data looks like:



Looks relatively promising given the limitations of the model.

The parameters are all the numbers we need to know in order to simulate the model. These are α, β, γ and δ as well as the initial conditions $x(0)$ and $y(0)$. These are all positive, so parameter space is $\Omega = \mathbb{R}^6_+$.

Let's measure time in years. Since the differential equations are autonomous (i.e. the right-hand side doesn't depend on time at all), we are free to choose $t = 0$ to be any time we wish, so let's let $t = 0$ correspond to the year 1945 (this is the year that the data start).

The data are the values of $x(t)$ and $y(t)$ at each point in time $t = 0, 2, 4, \ldots, 92$. In other words Δ is the set of 2×47 matrices with positive real entries and

$$D = \begin{pmatrix} 20 & 20 & 52 & \ldots & 83 & 12 \\ 32 & 50 & 12 & \ldots & 40 & 48 \end{pmatrix}.$$

The map $\delta : \Omega \to \Delta$ takes the solutions to the differential equations $x(t)$ and $y(t)$ when the parameters have the values in ω and finds their values at each of the points in time $t = 0, 2, 4, \ldots, 92$. In other words

$$\delta(\omega) = \begin{pmatrix} x(0) & x(2) & x(4) & \ldots & x(92) \\ y(0) & y(2) & y(4) & \ldots & y(92) \end{pmatrix}.$$

Method 1: Since the populations of the hare and lynx are comparable, let's use a simple least squares distance as our estimate of distance in Δ. (If the populations were different sizes and we therefore expected the error in the measurement of one to be significantly different from the error in the measurement of the other we might use a weighted least squares distance where we weight the one with the large error less than the one with the smaller error.) Thus, we need to find $\omega = (\alpha, \beta, \gamma, \delta, x(0), y(0)) \in \Omega$ such that

$$\begin{aligned}\phi(\omega) &= d(\delta(\omega), D) \\ &= \sum_{i,j} \left(\delta(\omega)_{ij} - D_{ij}\right)^2 \\ &= \sum_{j} \left(x(2j) - D_{1j}\right)^2 + \sum_{j} \left(y(2j) - D_{2j}\right)^2\end{aligned}$$

is a minimum.

Since we cannot solve the differential equations analytically, we don't have formulae for $x(2j)$ and $y(2j)$ in terms of the parameter values.

However, given parameter values we can find $x(2j)$ and $y(2j)$ numerically by solving the differential equations numerically! In other words, we *can* evaluate the function ϕ, so, in theory, we can still use `fminsearch` to do the optimization.

To do this we create an m-file that calculates the value of ϕ just as we did in the example above. In this case however, we don't have a simple formula for ϕ; to calculate the value of ϕ the m-file has to use an ode solver. The most common ode solver to use in Matlab is `ode45`. We called this function `phipredprey`.

Notice: because we will be using `fminsearch` the m-file for ϕ has a single input whose value is a vector of length six rather than six different inputs.

```
function [err] = phipredprey(par)


alpha = par(1);

beta = par(2);

gamma = par(2);

delta = par(4);

x0 = par(3);

y0 = par(4);


D = zeros(2, 47);

D(1, :)  = [20, 20, 52, ...  12];

D(2, :)  = [32, 50, 12, ...  48];
```

```
pprhs = @(t,x) [alpha*x(1)-beta*x(1)*x(2);
-gamma*x(2)+delta*x(1)*x(2)];

t = 0:2:92;
[~, deltaomega] = ode45(pprhs, t, [x0; y0]);

err = sum(sum((D - deltaomega').^2));
end
```

Once we have created this m-file we can search for the parameter values that minimize ϕ using `fminsearch`. We need to give `fminsearch` an initial guess; we choose $\alpha = \beta = \gamma = \delta = 1$ arbitrarily and $x(0) = 20$ and $y(0) = 32$ because these are the corresponding values in the data.

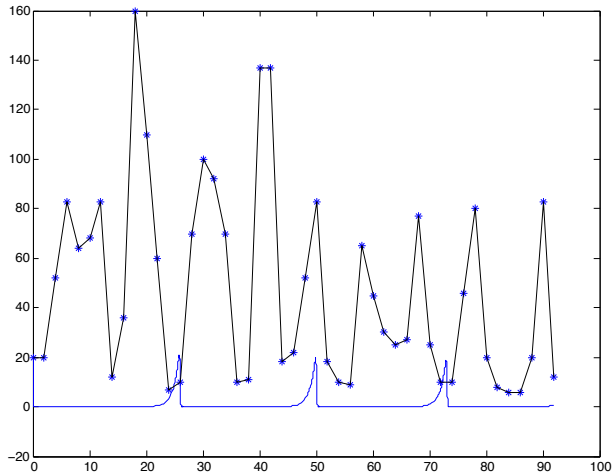
```
>> [param, err, exitf, info] = fminsearch(@phipredprey,
[1, 1, 1, 1, 20, 32])
```

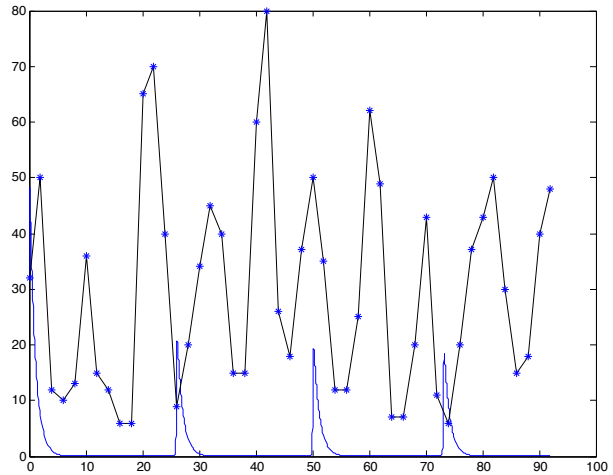
Matlab returns a result quite quickly, but the values of the parameters are suspiciously close to their starting values and the error is quite large given the values in D.

```
param =

Columns 1 through 5

1.0160 1.0187 0.9896 1.0022 20.2799

Column 6

31.8618

err =

2.3045e+05

exitf =

1

info =

iterations:  195

funcCount:  433
```

Moreover, if we plot the solutions to the differential equations with these parameter values it doesn't look good.

```
>> pprhs = @(t,x) [param(1)*x(1)-param(2)*x(1)*x(2);
-param(3)*x(2)+param(4)*x(1)*x(2)];

>> x0 = param(5); y0 = param(6);

>> [tim, soln] = ode45(pprhs, [0, 92], [x0; y0]);

>> figure(1)

>> plot(tim, soln(:, 1))

>> hold on

>> plot(0:2:92, D(1,:), '*')

>> plot(0:2:92, D(1, :), 'k')
```





The problem is probably that ϕ is very bumpy and the algorithm has gotten stuck in a local minimum.

One way to explore this hypothesis is to do the optimization again with a different initial guess.

I ran it with various initial guesses.

When I started with `[1, 1, 1, 1, 1, 1]` it ran for at least an hour; I quit the program before it exited.

With other initial guesses it converged but often to parameter values that were negative.

To deal with the negative parameter values we could:

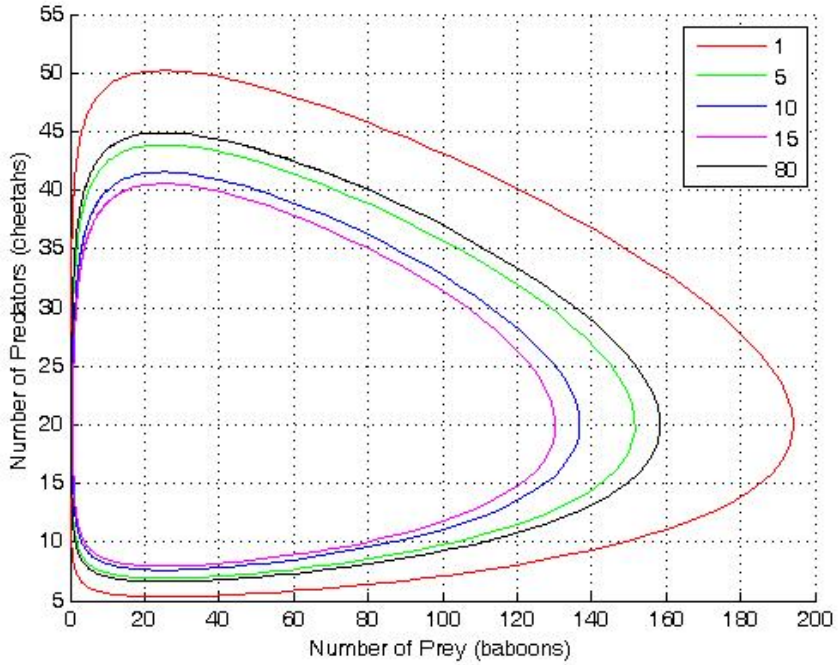
- Use a constrained optimization algorithm. Matlab has one in its optimization toolbox (I'm not sure if that comes with the Matlab package provided by ITS). This will probably be even slower.
- Write $\alpha = e^{\tilde{\alpha}}$ etc and use `fminsearch` to search for $\tilde{\alpha}$.

I didn't try either of these.

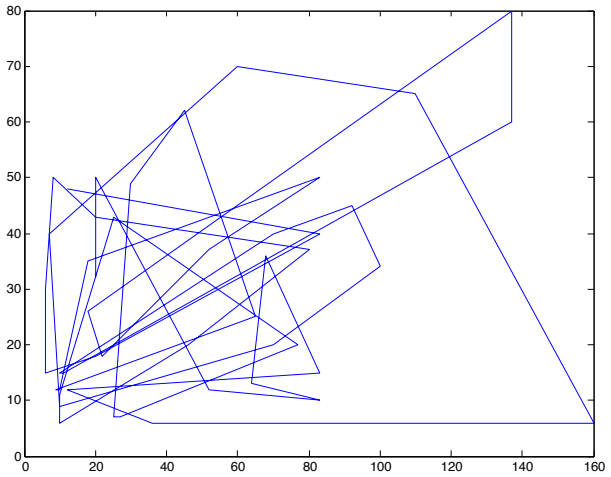
It's not looking good, but to confirm that the problem is with the procedure, we can test the procedure by choosing parameter values and *simulating data from the model*, then use the procedure to see if we can't reconstruct the parameter values we used.

I haven't tried this.

Method 2: Since the system of differential equations is autonomous (i.e. the right-hand side doesn't depend explicitly on time), it makes sense to look at the solutions in the (x, y)-plane. Here is a typical picture of the family of solutions in the (x, y)-plane.



When we plot the data in the (x, y)-plane we get:



This doesn't look so hopeful though we wouldn't expect a very clean fit because

a) the model is only very approximate (for example, only about 75% of the lynx' diet is the snowshoe hare instead of 100%) and

b) the measurements are far from precise (they are estimates of the populations based on furs traded by the company).

We can actually find an expression for the solution curves in the (x, y)-plane as follows. First we divide the two equations by each other to get an expression for dy/dx:

$$\frac{dy}{dx} = \frac{-\gamma y + \delta xy}{\alpha x - \beta xy} = \frac{-\frac{\gamma}{x} + \delta}{\frac{\alpha}{y} - \beta}$$

Then we solve by separation of variables to get:

$$\int \left(\frac{\alpha}{y} - \beta\right) dy = \int \left(-\frac{\gamma}{x} + \delta\right) dx$$

In other words, the solution curves have the form

$$\alpha \ln y - \beta y + \gamma \ln x - \delta x = K$$

for different constants K.

Let's ignore time and see if we can't find the curve of this form that most closely fits the data.

Method 2a: In other words, we look for α, β, γ and δ such that the numbers

$$\alpha \ln D(2,j) - \beta D(2,j) + \gamma \ln D(1,j) - \delta D(1,j)$$

are as close to a constant as possible as j ranges between 1 and 47. This means that we want the variance of this collection of numbers to be as small as possible.

Let

$$\tilde{\phi}(\alpha, \beta, \gamma, \delta) = \mathrm{Var}_{j=1}^{47} \left(\alpha \ln D(2, j) - \beta D(2, j) + \gamma \ln D(1, j) - \delta D(1, j) \right)$$

To find the minimum of $\tilde{\phi}$ we first make an m-file that computes its value from the parameters α, β, γ, and δ (we called this `phi2 -` it is easy to write). Then we call `fminsearch` to find the values of the parameters that minimize its value.

```
>> par = fminsearch(@phi2, [1,1,1,1])

par =

1.0e-04 *

0.2434 0.0107 0.3438 0.004

>> par = fminsearch(@phi2, [.1,.1,.1,.1])

par =

1.0e-04 *

0.4240 0.0100 -0.2108 -0.0043
```

We're getting negative parameter values again but we already know how to deal with that problem. The more serious problem is that the function $\tilde{\phi}$ must again be quite bumpy because we get different minimizers when we start at different starting points.

Looking more closely, notice that the parameter values it is finding are all very small. Does this make sense?

What's happening is the following.

The function $\tilde{\phi}$ doesn't directly measure how well the data fit the curve; instead it measures how close the left-hand side of the equation is to a constant.

To understand how the variance of the left-hand side of the equation relates to how well the data fit one of these curves we need to understand how the curves change when the constant K is varied. This will depend on the parameter values.

If there are parameter values where the curve changes a lot with small changes in K, then we could have all our data points lying on curves whose K-values are really close to each other, so the variance will be small without the data points lying close to any of the curves.

Looking at the formula for $\tilde{\phi}$ we see that when the parameter values are all small, the value of the left-hand side will be small for all our data points. In other words, tiny changes in K are producing drastic differences in the physical solution curves, so we can get $\tilde{\phi}$ to be small without the data lying close to any one solution curve.

In summary, the value of $\tilde{\phi}$ isn't really measuring what we want it to measure; we need a function that measures the physical distance between the data and the best-fitting solution curve for those parameter values.

Method 2b: Given the parameter values α, β, γ, and δ, the solution curves to the differential equation are the level sets of the function

$$f(x, y) = \alpha \ln x - \beta y + \gamma \ln x - \delta x.$$

The gradient of this function,

$$\nabla f(x, y) = \left(\frac{\gamma}{x} - \delta, \frac{\alpha}{y} - \beta \right),$$

points in the direction perpendicular to the solution curves and its magnitude indicates the rate at which the value of the function changes when you move in this direction.

In other words,

$$\frac{|f(x,y)-K|}{\left[\left(\frac{\gamma}{x}-\delta\right)^2+\left(\frac{\alpha}{y}-\beta\right)^2\right]^{1/2}}$$

is an estimate for the physical distance between the point (x, y) and the level curve on which the value of the function is equal to K.

Given α, β, γ and δ, we first want to find K that minimizes the sum of the squares of these distances where (x, y) ranges over the points in the data set. The level curve with this value of K is the best-fitting solution curve for those parameter values.

Then we need to find the parameters α, β, γ and δ, that minimize the sum of squares with this best value of K.

In other words, we need to minimize the function

$$\tilde{\tilde{\phi}}(\omega, K) = \sum_{j=1}^{47} \frac{[\alpha \ln D(2,j) - \beta D(2,j) + \gamma \ln D(1,j) - \delta D(1,j) - K]^2}{\left(\frac{\gamma}{D(1,j)} - \delta\right)^2 + \left(\frac{\alpha}{D(2,j)} - \beta\right)^2}$$

We write an m-file for this function (call it `phi3`) and run it through `fminsearch`.

Here is the m-file for $\tilde{\tilde{\phi}}$. This time I forced the parameters to be positive.

```
function [dist] = phi3(par)

alpha = exp(par(1));

beta= exp(par(2));

gamma = exp(par(3));

delta = exp(par(4));

k = par(5);

D = zeros(2, 47);

D(1, :)  = [20, 20, 52, ...  12];

D(2, :)  = [32, 50, 12, ...  48];

const = alpha*log(D(2,:))  - beta*D(2,:)  +
gamma*log(D(1,:))  - delta*D(1,:);
```

```
num = abs(const - k);

den = sqrt((gamma./D(1,:)  - delta).^2 + (alpha./D(2, :)
- beta).^2);

dist = sum(num./den);

end
```

Here are some results.

```
>> [par, fval] = fminsearch(@phi3, [10, 10, 10, 10, 10])

par =

Columns 1 through 4

-29.6607 23.9316 26.0754 -0.6217

Column 5

23.5161

fval =

650.7738
```

```
>> [par, fval] = fminsearch(@phi3, [1, 1, 1, 1, 1])

Exiting:  Maximum number of function evaluations has
been exceeded - increase MaxFunEvals option.  Current
function value:  610.250140

par =

1.0e+03 *

-4.0556 0.0056 0.0083 -0.2765 6.4492

fval =

610.2501
```

```
>> [par, fval] = fminsearch(@phi3, [1, 10, 5, 14, -5])
par =
-1.6497 34.5651 36.7089 -402.5583 -108.6188
fval =
650.7738
```

We are getting different parameter values for different initial guesses. However, notice that the value of the function is quite similar in each case. This is promising.

Could it be that different parameter values can yield the same solution curves?

Yes! If we rescale time, the parameter values will scale but the solution curves will remain the same.

Method 2c: So, let's force $\alpha = 1$.

```
>> [par, fval] = fminsearch(@(par)phi3([0, par]), [10,
5, 14, -5])

par =

19.8308 21.9746 -38.5454 6.5929

fval =

650.7738

>> [par, fval] = fminsearch(@(par)phi3([0, par]), [-3,
0, 6, 2])

par =

-6.9562 -0.0006 -3.7020 6.0128

fval =

514.8197
```

```
>> [par, fval] = fminsearch(@(par)phi3([0, par]), [0, 0,
0, 0])

par =

32.0928 34.2366 -68.4023 -10.1172

fval =

650.7738
```

We have two very different sets of values of the parameters giving us the same error and another very different set of values of the parameters giving us a lower error. :-(

Are there still multiple parameter values that yield the same solution curves? To have the same solution curves we need dy/dx at each point (x, y) in the plane to be the same. Recall

$$\frac{dy}{dx} = \frac{-\frac{\gamma}{x} + \delta}{\frac{\alpha}{y} - \beta}$$

along the solution curves. Thus, $(\alpha, \beta, \gamma, \delta)$ yields the same solution curves as $(\tilde{\alpha}, \tilde{\beta}, \tilde{\gamma}, \tilde{\delta})$ if and only if, for all points (x, y) in the plane

$$\frac{-\frac{\gamma}{x} + \delta}{\frac{\alpha}{y} - \beta} = \frac{-\frac{\tilde{\gamma}}{x} + \tilde{\delta}}{\frac{\tilde{\alpha}}{y} - \tilde{\beta}}$$

This will be the case if and only if

$$\left(-\frac{\gamma}{x}+\delta\right)\left(\frac{\tilde{\alpha}}{y}-\tilde{\beta}\right)=\left(-\frac{\tilde{\gamma}}{x}+\tilde{\delta}\right)\left(\frac{\alpha}{y}-\beta\right).$$

We have equality for every point (x, y) in the plane if and only if the coefficients of $1/(xy)$, $1/x$, and $1/y$ and the constant term are all equal. In other words the solution curves are the same if and only if

$$\gamma\tilde{\alpha}=\tilde{\gamma}\alpha$$

$$\gamma\tilde{\beta}=\tilde{\gamma}\beta$$

$$\delta\tilde{\alpha}=\tilde{\delta}\alpha$$

$$\delta\tilde{\beta}=\tilde{\delta}\beta.$$

It's not hard to see that we have this if and only if, for some constant c,

$$(\tilde{\alpha},\tilde{\beta},\tilde{\gamma},\tilde{\delta})=c(\alpha,\beta,\gamma,\delta).$$

In other words, by setting $\alpha = 1$ we should now have different solution curves for each set of parameter values.

This means that the problem is not non-uniqueness of the solution curves. Rather, there could be three issues.

- There are very different parameter values that yield different solution curves but these different curves are are equally close to the data.
- Our measurement of how close the data are to a solution curve is only an estimate; it could be that the estimate is way off for some solution curves (i.e. some values of K) and therefore we get a 'small' value for the distance even when that distance is large.
- The function $\tilde{\tilde{\phi}}$ is very bumpy and we're getting stuck in a local minimum.

Method 2d: Let's see if we can't narrow down the appropriate values of the parameters. For that we have the following theorem.

Theorem: Let $x(t), y(t)$ be a periodic solution to the differential equation with period T. Let

$$\bar{x} = \frac{1}{T}\int_0^T x(t)\,dt, \qquad \bar{y} = \frac{1}{T}\int_0^T y(t)\,dt$$

denote the average values of x and y. Then $\bar{x} = \gamma/\delta$ and $\bar{y} = \alpha/\beta$. In particular, the average values of x and of y are the same for every solution curve and are determined completely by the parameters.

Proof: From the differential equation we have that $y'/y = -\gamma + \delta x$. Integrating with respect to time from 0 to T we get

$$\ln(y(0)) - \ln(y(T)) = -\gamma T + \delta \int_0^T x(t)\,dt$$

But $y(0) = y(T)$ so

$$\frac{1}{T}\int_0^T x(t)\,dt = \frac{\gamma}{\delta}.$$

A similar calculation shows the corresponding result for the average value of y.

So, let's calculate the average value of x and y from the data; this will help narrow down our search for the parameters.

However, the average is taken over one period of oscillation. We don't know how long one period is! Eyeballing the data it looks like one period might be around 9 years or so and we have data that is equally spaced in time over 90 years. Since the data seem to roughly cover an integer number of periods and there are enough periods covered that it probably wouldn't affect the average much if it really ends up covering a non-integer number of periods, we'll estimate the average of x and y simply by averaging the corresponding numbers in the data.

We get

$$\bar{x}_{\text{data}} = 48.96$$

$$\bar{y}_{\text{data}} = 28.85$$

So now we'll look for parameters such that

$$\frac{\gamma}{\delta} = \bar{x}_{\text{data}}$$

$$\text{and} \qquad \frac{\alpha}{\beta} = \bar{y}_{\text{data}}$$

Since we are looking for parameters with $\alpha = 1$, then we choose

$$\beta = \frac{1}{\bar{y}_{\text{data}}}.$$

So then we set $\gamma = \delta \bar{x}_{\text{data}}$ and seek to minimize $\tilde{\tilde{\phi}}$ over δ and K.

We still get wildly varying values of δ and K and, moreover, the error now is larger; it ranges from about 5,000 to 40,000 (instead of 650).

It's not surprising that the error is larger since the estimates we have for β and γ are estimates and are certainly not the exact values that yield the best fitting approximation. However, it's discouraging that the values of δ and K still vary wildly.

It would be good to try to sketch some of these solution curves that we are finding along with the data.

Each curve is given as the set of solutions (x, y) to an equation in x and y:

$$\alpha \ln y - \beta y + \gamma \ln x - \delta x = K.$$

Given values of the parameters we can plot the curve in Matlab as the level set of the function in x and y on the left.

```
>> [x, y] = meshgrid(0:1:180, 0:1:100);
>> x(1:5, 1:5)
ans =
0   1   2   3   4
0   1   2   3   4
0   1   2   3   4
0   1   2   3   4
0   1   2   3   4
>> y(1:3, 1:3)
ans =
0   0   0
1   1   1
2   2   2
```

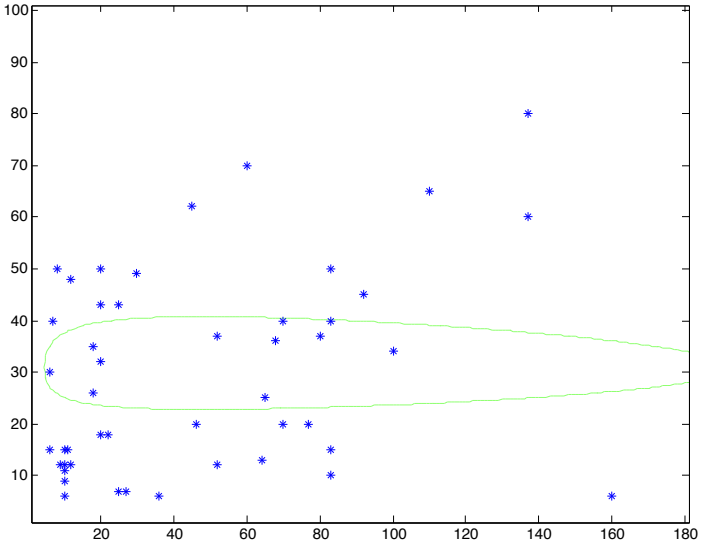
Here are the values of the parameters that we got from one of the optimization runs.

```
>> alpha = 1; beta = 1/avdata(2);

>> delta = exp(-7.4618); gamma = delta*avdata(1);

>> k = 2.4310;
```

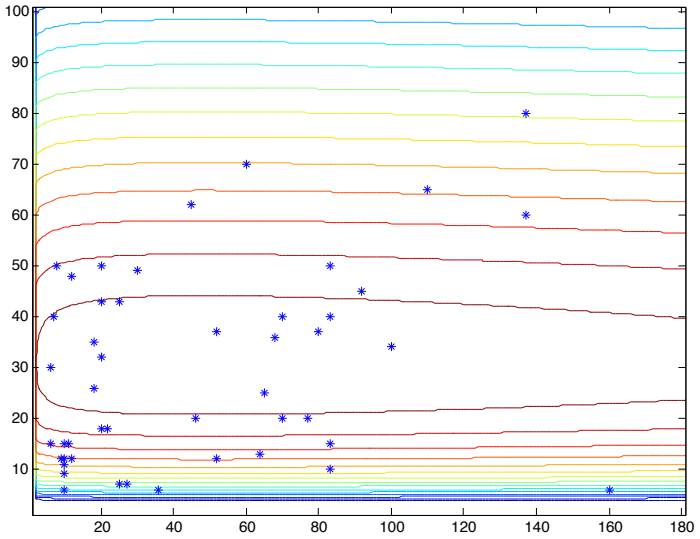
Now we use these values of the parameters to plot the solution curve along with the data.

```
>> z = alpha*log(y) - beta*y + gamma*log(x) - delta*x;

>> contour(z, [k, k])

>> hold on, plot(D(1,:), D(2,:), '*')
```

Here's what it looks like.



If we look at different solution curves with different values of k (using `contour(z, [1:.1:2.5])`) we get the following picture.



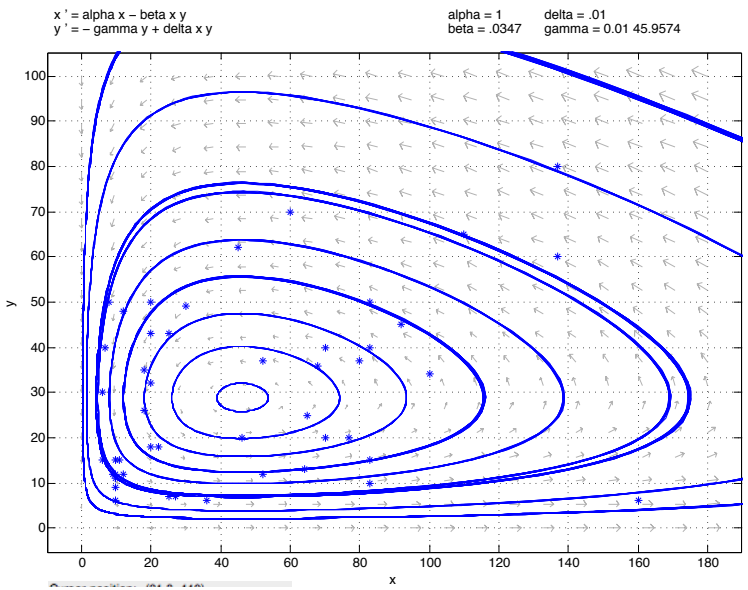
It's pretty clear that many different solution curves could fit the data equally well (or badly). This means that the problem with the optimization is that the function $\tilde{\tilde{\phi}}$ is very bumpy.

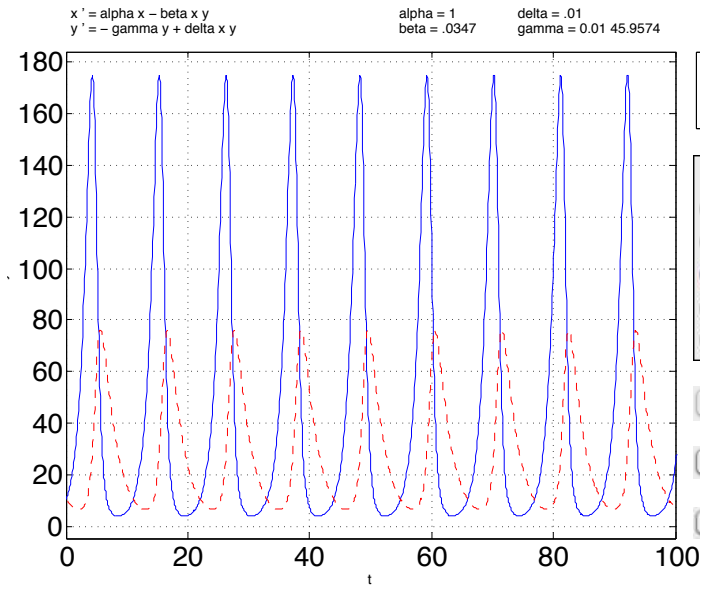
Method 3: Let's just explore the process by hand. By this I mean, let's play with the parameters and look at the different solution curves we get and compare with the data.

For this we can use a Matlab program developed by John Polking at Rice University for looking at solutions to a system of two differential equations. The program is called pplane8 and can be downloaded from his website at

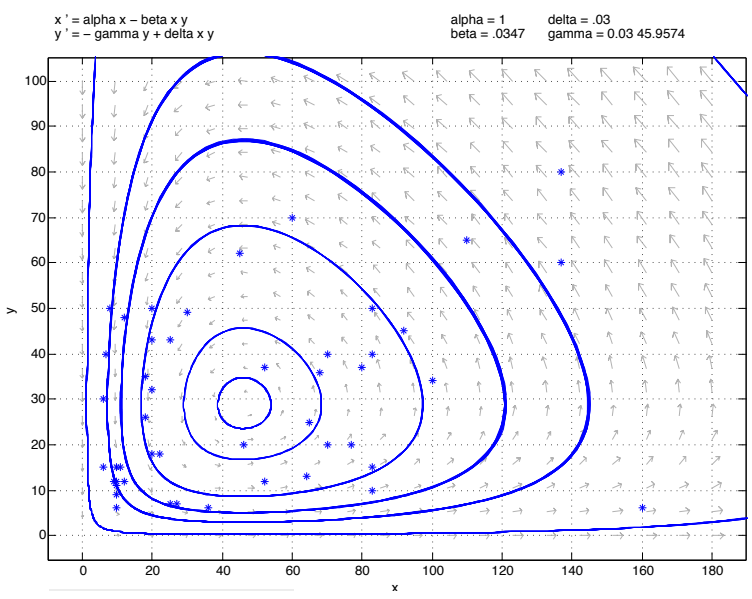
`http://math.rice.edu/~polking/`

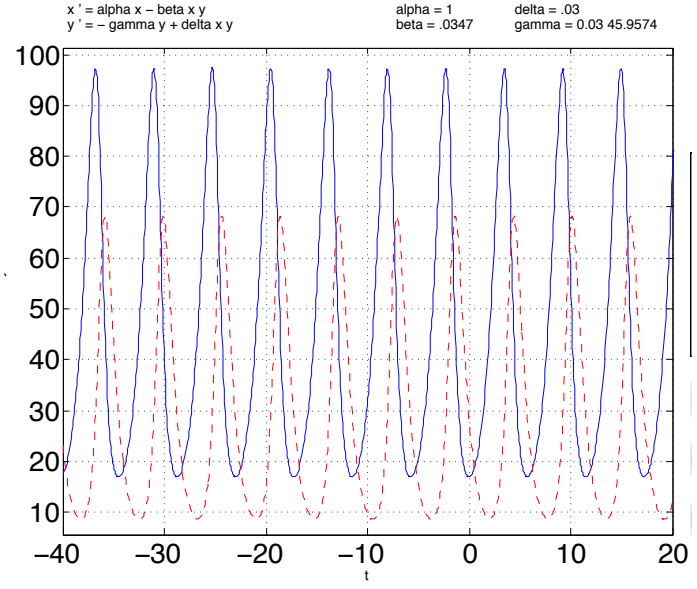
Here are some plots of the solution curves for different cases of the parameters. First we show $\alpha = 1$, $\beta = 1/\bar{y}_{\text{data}}$, $\gamma = 0.01$ and $\delta = \gamma \bar{x}_{\text{data}}$. We plot some solutions in the (x, y)-plane along with the data and then plot both x and y as a function of t for a promising-looking solution.





This looks pretty good. Using `pplane` you can explore what happens to the solution curves when you vary the parameters. In the next picture we keep α and β the same, change γ to be 0.03 and have $\delta = \gamma \bar{x}_{\text{data}}$ as above. Here are some solution curves along with the data as well as a promising-looking solution plotted against time. Notice that the shape of the solution curves has changed a little; the curves stretch less in the x-direction and more in the y-direction.





These parameters look promising. However, by setting $\alpha = 1$ we rescaled time and it looks like our time scale is a little off; the period of the data looked like it was around 9 years whereas the period in our solutions looks like it's about 11 years.

So, let's go back to our original method of estimating the parameters, but now let's set $\alpha = \beta\bar{y}_{\text{data}}$ and $\gamma = \delta\bar{x}_{\text{data}}$ and try to find the values of β, δ, $x(0)$ and $y(0)$ that minimize ϕ.

```
function [err] = phipredprey(par)

beta = exp(par(1));

delta = exp(par(2));

x0 = par(3);

y0 = par(4);

D = zeros(2, 47);

D(1,:)  = [20, 20, 52, ..., 83, 12];

D(2,:)  = [32, 50, 12, ..., 40, 48];

xbar = mean(D(1,:));

ybar = mean(D(2,:));

alpha = beta*ybar;

gamma = delta*xbar;
```

```
pprhs = @(t,x) [alpha*x(1)-beta*x(1)*x(2);
-gamma*x(2)+delta*x(1)*x(2)];

t = 0:2:92;

[ , deltaomega] = ode45(pprhs, t, [x0; y0]);

err = sum(sum((D - deltaomega').2));

end
```

```
>> [param, err, exitf, info] = fminsearch(@phipredprey,
[log(1/ybar), log(.01), 20, 32])

param =

-3.3588 -4.0876 21.0246 31.2201

err =

1.0094e+05

exitf =

1

info =

iterations:  128

funcCount:  273

algorithm:  'Nelder-Mead simplex direct search'

message:  [1x194 char]
```

Here are the values we obtained:

$$\alpha = 1.0403$$

$$\beta = 0.0348$$

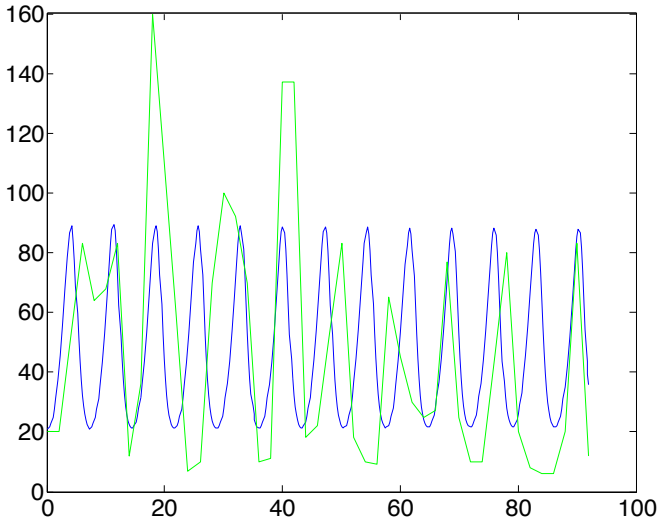
$$\gamma = 0.7922$$

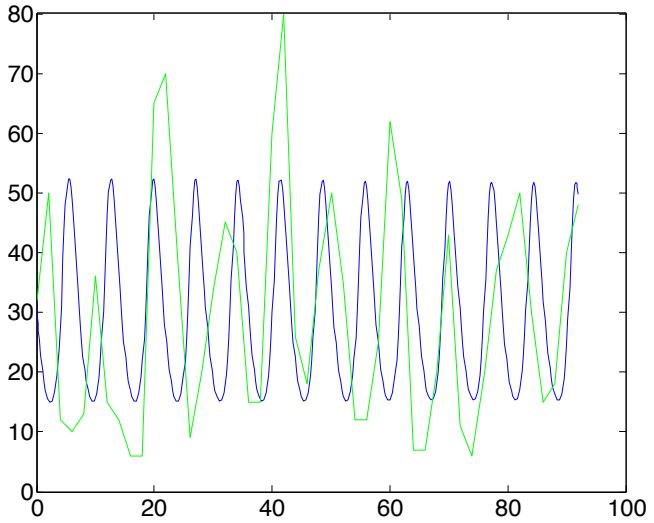
$$\delta = 0.0168$$

$$x0 = 21.0246$$

$$y0 = 31.2201$$

Here are plots of the solutions along with the data.


160
140
120
100
80
60
40
20
0
0
20
40
60
80
100


80
70
60
50
40
30
20
10
0
0
20
40
60
80
100

Determining parameters from data when the model is stochastic.

When the model is stochastic every time you simulate the model you get different results. This means that you wouldn't expect a simulation to match the data you collect. In this case, the most common approach to estimating the parameters is to find those values of the parameters that make the data *most likely*.

This can be a lot easier said than done. Here's an example where's it's feasible.

Example:

Recall that the parameters in the Reed-Frost model of an epidemic are p and N.

The parameter N is generally known in any application of the model; the parameter p depends on the nature of the disease and perhaps also on the behavior of the small group of people to which the Reed-Frost model is being applied.

Suppose we have observed the disease spreading through two different families; in one family of 4 people, three people total ended up getting infected. In the other family of 3 people, all three people ended up getting infected. Can we use this data to estimate p?

First of all, to use this data to estimate p we need to assume that the value of p in the two cases and in the case to which we want to apply the data is the same. This is reasonable if all the families behave similarly.

We want to find that value of p that maximizes the probability of obtaining the data. So, first we need to calculate this probability.

In a family of 4 there are two ways that 3 people can get infected. These are shown below where each triple denotes the values (S, I, R) at each time step.

$$(3, 1, 0) \to (1, 2, 1) \to (1, 03)$$

and

$$(3, 1, 0) \to (2, 1, 1) \to (1, 1, 2) \to (1, 0, 3)$$

To calculate the probability of the first way notice that $X_0 = 2$ where $X_0 \sim \text{Bin}(3, p)$, $X_1 = 0$ where $X_1 \sim \text{Bin}(1, 1 - (1-p)^2)$, and X_0 and X_1 are independent. Thus, the probability is

$$\left[3p^2(1-p)\right](1-p)^2 = 3p^2(1-p)^3.$$

To calculate the probability of the second way notice that $X_0 = 1$ where $X_0 \sim \text{Bin}(3, p)$, $X_1 = 1$ where $X_1 \sim \text{Bin}(2, p)$, $X_2 = 0$ where $X_2 \sim \text{Bin}(1, p)$, and X_0, X_1, and X_2 are independent. Thus, the probability is

$$\left[3p(1-p)^2\right]\left[2p(1-p)\right](1-p) = 6p^2(1-p)^4.$$

Thus the probability that three people get sick when there are four people is

$$\begin{aligned} 3p^2(1-p)^3 + 6p^2(1-p)^4 &= 3p^2(1-p)^3\left[1+2(1-p)^2\right] \\ &= 3p^2(1-p)^3(2p^2-4p+3). \end{aligned}$$

Now, let's look at the family of three people. There are two ways that all three people can get sick in this case. They are:

$$(2, 1, 0) \to (0, 2, 1) \to (0, 0, 3)$$

and

$$(2, 1, 0) \to (1, 1, 1) \to (0, 1, 2) \to (0, 0, 3)$$

In the first case, $X_0 = 2$ where $X_0 \sim \text{Bin}(2, p)$ and then $(0, 2, 1)$ goes to $(0, 0, 3)$ with probability 1. Thus the probability of this occurring p^2.

In the second case, $X_0 = 1$ where $X_0 \sim \text{Bin}(2, p)$, $X_1 = 1$ where $X_1 \sim \text{Bin}(1, p)$, and with probability 1, $(0, 1, 2)$ goes to $(0, 0, 3)$. Thus the probability of this occurring is $2p(1-p)p = 2p^2(1-p)$.

Thus the probability that everyone gets sick when there are 3 people is

$$p^2 + 2p^2(1-p) = p^2(3-2p)$$

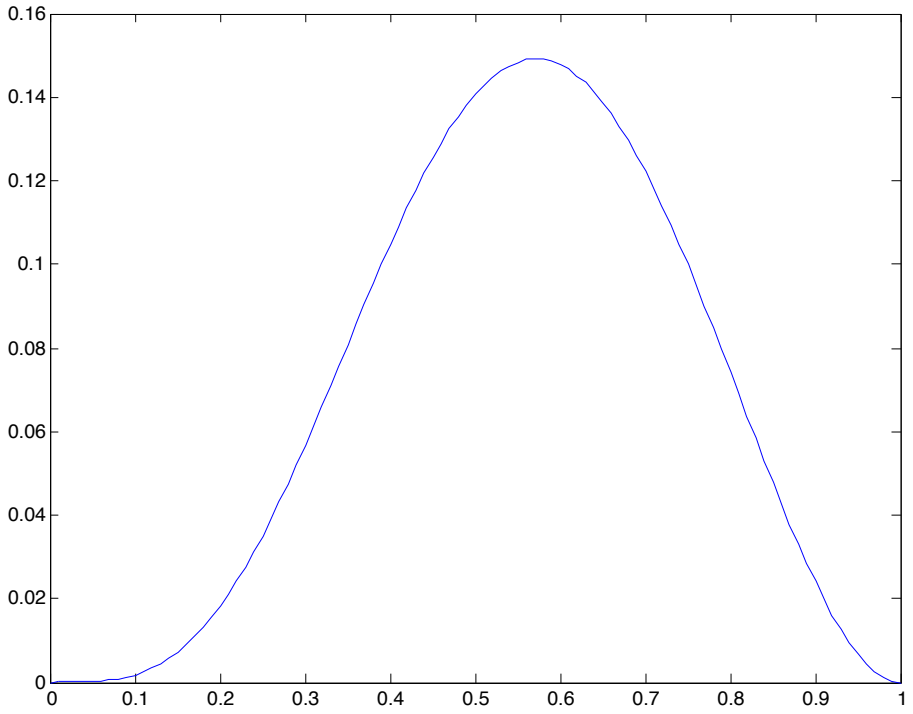
We have to assume that these two events we observed are independent. So the probability of observing the data is the product of these two probabilities

$$L(p) = 3p^4(1-p)^3(2p^2-4p+3)(3-2p).$$

This is called the likelihood function. We want to find the value of p that maximizes this function.

This is a polynomial of degree 10, so it's unlikely to be able to do this analytically. So, let's use Matlab.

We require p to lie between 0 and 1. The graph of L is shown below. It looks very well-behaved.



We want to maximize L so that means we want to minimize $-L$.

In order to use `fminsearch` but constrain p to lie between 0 and 1 we look for a function that will convert a real number between $-\infty$ and ∞ to a number between 0 and 1. Such a function is

$$p = \frac{\arctan(x)}{\pi} + \frac{1}{2}.$$

So we will minimize $-L(p(x))$ where x can be any real number.

```
function [l] = likly(x)
p = (1/pi)*atan(x) + 0.5;
l = - 3*p.^4.*(1-p).^2.*(2*p.^2-4*p+3).*(3-2*p);
end
>> [x, likelihood] = fminsearch(@likly, 0.5)
x =
0.2286
likelihood =
-0.1492
>> p = (1/pi)*atan(x) + 0.5
p =
0.5715
```

That's about what we expected from the graph.

The example above was exceptional because we were able to calculate the likelihood function and it was relatively simple.

More often than not it's difficult to calculate the likelihood function or, even if you can get an expression for it, it may involve many integrals, so it is time consuming to calculate its value given specific values of the parameters.

In these cases you can consider estimating the value of the likelihood function for a given set of parameters by simulating the model many times and seeing how often you observe the data (or something close to the data).