Parameter Estimation Example

Kurt Bryan and SIMIODE

A very simple example of fitting a function or model to data by using least squares.

The Data: Here are some hypothetical data in the form of (t,y) pairs:

```
bolttimes = [0.165, 1.85, 2.87, 3.78, 4.65, 5.50, 6.32, 7.14, 7.96, 8.79, 9.69];
dists = 0:10:100
```

A quick plot of (time, distance) pairs:

```
scatter(bolttimes,dists);
```

Solving The ODE: Solve the Hill-Keller ODE, treating both P and k as unknown. Also use initial condition v(0.165) = 0.

```
syms v(t); %Declare v(t) as symbolic function
syms k;
syms P;
ode = diff(v(t),t) == P-k*v(t) %Define the ODE
t0 = 0.165;
vsol(t) = dsolve(ode,v(t0)==0) %Incorporate initial condition
```

Integrate to obtain the position in terms of t, k, and P:

```
syms tau;
X(t,k,P) = int(vsol(tau),tau,0.165,t)
```

Let's guess a value k = 1 and P = 11 and plot X(t) with the data.

```
fplot(X(t,1,11),[0 9.69],'-r')
hold on;
scatter(bolttimes,dists);
hold off;
```

The Optimal Choice for k and P: Not bad, but we can do better by forming a sum of squares SS and minimizing with respect to k.

```
syms SS(k,P)
SS(k,P) = sum((X(bolttimes,k,P)-dists).^2);
```

To get a sense of where the minimum is, plot this SS(k,P) as a function of k and P, or better yet, plot log(SS(k,P)). . We already know the minimum is somehwere around k = 1 and P = 11.

```
fsurf(log(SS),[0.7 1.1 8.0 12.0])
```

Rotating the graph around shows k around 0.85 and P around 10.3 looks promising. So set d(SS)/dk = 0 and d(SS)/dP = 0 and use Matlab's ***vpasolve*** command to find a good solution

```
dSSdkeqn = diff(SS,k)==0;
```

```
dSSdPeqn = diff(SS,P)==0;
kPbest = vpasolve([dSSdkeqn,dSSdPeqn],[k,P],[0.9; 10.5]) %Initial guess k = 0.9, P = 10.5
```

The residual is

```
kbest = kPbest.k
Pbest = kPbest.P
SS(kbest,Pbest)
```

Use these values in X(t) to plot and compare to the data

```
fplot(X(t,kbest,Pbest),[0 9.69],'-r')
hold on;
scatter(bolttimes,dists);
hold off;
```

Alternatively, we can minimize SS with respect to k and P by using Matlab's built-in optimization routines, although this requires the Optimization Toolbox.

```
SSf = matlabFunction(SS,'Vars',{[k P]});  %Converts the symbolic function "SS" to a traditional
[kP,fval] = fminunc(SSf,[0.9 10.5]) %Initial guess k = 0.9, P = 10.5
```