The Hill-Keller Model

Kurt Bryan and SIMIODE

A Matlab script to fit the Hill-Keller model v'(t) = P - k*v(t) to the data from Usain Bolt's 2008 Beijing gold medal Olympic race. We focus on the case P = 11 meters per second squared and adjust k to fit the data, in a least-squares sense.

The Data: Here is the data for Usain Bolt's 2008 Beijing race. The data consists of pairs [t, d], where d is 0, 10, 20, ..., 100 meters and t is the corresponding split time.

```
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, using P = 11 and treating "k" as unknown. Also use initial condition v(0.165) = 0.

```
syms v(t); %Declare v(t) as symbolic function
P = 11;
syms k;
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 and k.

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

Let's guess a value of k and plot X(t) with the data.

```
fplot(X(t,1),[t0,9.69],'-r')
hold on;
scatter(bolttimes,dists);
hold off;
```

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

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

To get a sense of where the minimum is, plot this expression as a function of k:

```
fplot(SS(k),[0.7 1.1])
```

Somewhere around k = 0.93 it appears. Find using Calculus 1 techniques. Compute d(SS)/dk, set to zero, and solve (numerically, using Matalb's ***vpasolve*** command). Tell Matlab to look between 0.85 and 0.95.

```
eqn = diff(SS,k)==0;
kbest = vpasolve(eqn,k,[0.85,0.95])
```

The residual is

```
SS(kbest)
```

Use this value in X(t) to plot and compare to the data

```
fplot(X(t,kbest),[t0,9.69],'-r')
hold on;
scatter(bolttimes,dists);
hold off;
```

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

```
SSf = matlabFunction(SS,'Vars',{k}); %Converts the symbolic function "SS" to a traditional matl
kbest = fminbnd(SSf,0.8,1.0)
```