CS4220 Assignment 7   Due: 5/2/13 (Thur) at 11pm

You must work either on your own or with one partner. You may discuss background issues and general solution strategies with others, but the solutions you submit must be the work of just you (and your partner). If you work with a partner, you and your partner must first register as a group in CMS and then submit your work as a group. Each problem is worth 5 points. One point may be deducted for poor style.

Topics: Numerical optimization, `fminsearch`, systems of nonlinear equations, `fsolve`, nonlinear least squares, `lsqnonlin`

1 Costliest Tetrahedron

A tetrahedron has its four vertices on the unit sphere and its four faces are colored red, blue, green, and cyan. Let R, G, B, and C be the areas of these faces in square feet and let r, g, b, and c be the cost of painting one square foot in the respective colors. The cost of the tetrahedron is given by

$$Cost \;=\; rR + gG + bB + cC.$$

By making effective use of `fminsearch`, implement the following function:

```
  function [Cost,R,G,B,C,nFevals] = Tetra(r,g,b,c)
% r,g,b,c specifies the cost per square foot of red, green, blue, and cyan paint.
% Cost returns the value of the most expensive tetrahedron that is circumscribed
% by the unit sphere and whose four faces are painted red, green, blue and
% cyan. The areas of the red, green, blue, and cyan faces are reported
% in R,G,B,and C. The number of times that fminsearch calls the objective
% function is returned in nFevals.
```

Before the call to `fminsearch`, set

```
options = optimset('MaxFunEvals',1000,'TolFun',.00000001,'TolX',.00000001);
```

Some math facts. If a triangle has sides a, b, and c, then its area is given by

$$A \;=\; \sqrt{s(s-a)(s-b)(s-c)} \qquad s \;=\; \frac{a+b+c}{2}$$

A point on the unit sphere with latitude θ with $-\pi/2 \le \theta \le \pi/2$ and longitude ϕ with $-\pi \le \phi \le \pi$ has xyz coordinates $(\cos(\theta)\cos(\phi), \cos(\theta)\sin(\phi), \sin(\theta))$.

Submit your implementation of `Tetra` to CMS. A test script `P1` is available on the website.

2 A Dynamical System

We describe a dynamical system that models the life cycle of the flour beetle. The scene at time t is defined by a triplet of values:

$L(t)$: the number of beetles in the larval stage

$P(t)$: the number of beetles in the pupal stage

$A(t)$: the number of beetles in the adult stage

The situation at time $t+1$ depends upon a number of factors. In particular, Adults snack on larva and pupa and larva snack on other larva.

Why can't we all just get along!

Let's get precise by talking the about the model parameters that define the interactions. Between time t and $t+1$ each adult "order up" b new larva. With probability $e^{-c_{ea}A(t)}$ they are *not* eaten by another adult and with probability $e^{-c_{el}L(t)}$ they are *not* eaten by another larvae. A beetle is in the larval stage for exactly one time unit so

$$L(t+1) \;=\; b\,A(t)\,e^{-c_{ea}A(t)}\,e^{-c_{el}L(t)}. \tag{1}$$

The constants c_{ea} and c_{el} are *rate constants.*

A larva graduates to the pupa stage with probability $1-\mu_L$. A beetle is in the pupa stage for exactly one time unit and so

$$P(t+1) \;=\; L(t)(1-\mu_L). \tag{2}$$

An adult survives the time step with probability $1-\mu_A$. The probability that a pupa makes it to the adult stage is $e^{-c_{pa}t}$. Thus,

$$A(t+1) \;=\; P(t)e^{-c_{pa}A(t)} + A(t)(1-\mu_A). \tag{3}$$

Define the function F as follows

$$F(x) \;=\; F\left(\begin{bmatrix} L \\ P \\ A \end{bmatrix}\right) \;=\; \begin{bmatrix} b\,A\,e^{-c_{ea}A}\,e^{-c_{el}L} \\ L(1-\mu_L) \\ Pe^{-c_{pa}A} + A(1-\mu_A) \end{bmatrix}.$$

The script `P2` can be used to check out the iteration

$$x_{k+1} \;=\; F(x_k)$$

for various choices of the model parameters and starting vectors. Here, the 3-vector x_k reports the values of L, P and A after k time steps. (FYI: a time step is about 14 days.) By playing with the script, observe that the L, P, and A values sometimes reach a "steady state". The goal in this problem is to characterize when this happens.

We say x is an *equilibrium point* of the flour beetle dynamical system if $\widehat{F}(x) = 0$ where

$$\widehat{F}(x) \;=\; F(x) - x.$$

We can use `fsolve` to compute this vector. It can be shown that if $c_{el} = 0$ and $b > \mu_A/(1-\mu_L)$, then

$$x_{fixed} \;=\; F(x_{fixed})$$

where

$$x_{fixed} \;=\; \begin{bmatrix} L_{fixed} \\ P_{fixed} \\ A_{fixed} \end{bmatrix} \;=\; \begin{bmatrix} \log(b(1-\mu_L)/\mu_A)/(c_{ea}+c_{pa}) \\ b\,A_{fixed}\,e^{-c_{ea}A_{fixed}} \\ L_{fixed}\,(1-\mu_L) \end{bmatrix}.$$

It is reasonable to use x_{fixed} as a starting vector.

Let $J(x)$ be the Jacobian of F at x. We say that x_* is a *stable equilibrium* if when the equilibrium values for L, P, and A are slightly perurbed, then subsequent time steps restore the equilibrium. It turns out that this property hinges on the eigenvalues of F's Jacobian at the equilibrium point. We offer a hint as to why this is the case. From Taylor series we have

$$F(x_k) \;\approx\; F(x_*) + J(x_*)\cdot(x_k - x_*) \;=\; x_* + J(x_*)\cdot(x_k - x_*)$$

If $x_{k+1} = F(x_k)$ this says that

$$x_{k+1} - x_* \;\approx\; J(x_*)(x_k - x_*) \;\approx\; [J(x_*)]^k(x_0 - x_*)$$

We know that $[J(x_*)]^k \to 0$ if the spectral radius of $J(x_*)$ is less than one. Thus, if x_0 is a slight perturbation of x_* then subsequent iterates work there way back to x_*.

Modify P2 so that it sheds light on this phenomena. Here are the details. Set $c_{el} = c_{ep} = c_{ea} = 0.01$ and $\mu_A = \mu_L = 0.5$. For $b = 1.5:0.5:16$ your modified P2 should

- Apply `fsolve` to find a zero of $\widehat{F}$. Use x_{fixed} for a starting value. The function passed to `fsolve` should also supply the Jacobian of $\widehat{F}$. Use the MATLAB online documentation to figure out how to do this.
- Use `eig` to compute the spectral radius ρ of F's Jacobian at the equilibrium point that you compute using `fsolve`.
- If $\rho < 1$, then plot in a single window the values that L, P, and A take on at time steps $0, ..., 200$ with starting value x_{fixed}. Use `title` to display the value of b and e and `xlabel` to display the equilibrium values of L, P, and A.
- If $\rho \geq 1$, then plot in a single window the values of L, P, and A at time steps $0, ..., 200$ assuming that the simulation starts at the equilibrium point. Use `title` to display the value of b and e and `xlabel` to display the equilibrium values of L, P, and A. This gives us a chance to see divergence from the equilibrium point.
- Include an `shg` command and a `pause` command after your `plot` command to facillitate grading.

Make your implementation life easier by noting that $\widehat{F}$ and F (and their respective Jacobians) are highly related. Submit you final version of P2 to CMS.

3 Predicting Season Length

The seasons do not have equal length. For example, for the seasons that start in 2013, Spring = 92.75 days, Summer = 93.65, Autumn = 89.85, and Winter = 88.99. Check out `StartTimes.m` for more details. The goal in this problem is to build a model that can predict when the seasons start. The model you will use is the *eccentric model* and it has two parameters. Download `ShowEccentric.m` and run (for example) `ShowEccentric(-.2,pi/6)`. The white dot moves around the unit circle at a uniform rate. By playing with the eccentric point $x_c, 0)$ and the "crosshair" tilt angle ϕ we see that the arclength between the season points can be made to vary. For example, if the arclength between the "Spring dot" and the "Summer Dot" is 100 degrees, then the model is predicting that the length of spring is about $(100/360)\cdot 365.25$ days.

Complete the following function so that it performs as specified:

```
  function [xcStar,phiStar,maxErr] = BestEccentric(k)
% k is an integer that satisfies 1<=k<=20
% xcStar and phiStar are the optimal eccentric model parameters
% from the standpoint of predicting the first 4k season events.
% maxErr returns the max(|s(1)-shat(1)|,...,|s(4k)-shat(4k)|}
% where s(i) is the observed start time of the i-th event and
% shat(i) is the predicted start time of the i-th event obtained
% by the eccentric model with parameters xcStar and phiStar.
```

Make effective use of `lsqnonlin`. The essential trigonometry that you need can be found in `ShowEccentric`. The function `StartTimes()` provides the data that you need to fit. A test script `P3` is available. Submit `BestEccentric` to CMS.