Matlab Scripts for Tumor Dynamics Module *

L.G. de Pillis and A.E. Radunskaya

August 31, 2002

1 Matlab Script Descriptions

To run any script, type its name at the Matlab prompt (without the .m extension). Be sure the scripts that must be called by the demo program are either in your Matlab path, or in your current directory.

Scripts for Qualitative Analysis Module:

QualDemo1.m This script file can be used to draw null-clines and equilibria for the system of ODE's described in [KMTP94]. If parameter values are changed, they must be changed in both *BifDemo1.m* and *BifDemo3.m*. This demo calls scripts:

- *BifDemo1.m*
- *BifDemo3.m*
- *NullE.m*
- *NullT.m*
- *NullIntersect.m*
- *KuzEquil.m*
- *EquilStability.m*
- *KuzJac.m*

Scripts for Numerics Module:

NumDemo1.m Demonstrates Euler's method. This demo calls scripts:

- *ODE0.m*
- *ODE0soln.m*

NumDemo2.m Uses Euler's method to solve an example from [KMN89]. This demo calls scripts:

- *ODE2.m*

*This work was supported in part by a grant from the W.M. Keck Foundation

- *ODE2soln.m*

NumDemo3.m Compares stiff and nonstiff solvers. This demo calls scripts:

- *stiffODE.m*
- *stiffODEsoln.m*

Scripts for Parameter Estimation Module:

ParDemo1.m Illustrates Newton's method. Be prepared to input a range over which to plot your function in the form [a b], as well as an initial guess, which should be any real number $x \in$[a b]. This demo allows you to loop through inputting several initial guesses until you type in "9999", which ends the demo. This demo calls scripts:

- *fEx.m*
- *fExP.m*

ParDemo2.m Estimates parameters for the logistic differential equation. This demo calls scripts

- *logisticdist.m*
- *logisticsoln.m*

ParDemo3.m Estimates remaining parameters of the system in [KMTP94]. The distance function is a sum of squares. The system is evaluated at all of the times given in the data, for each of the three initial conditions. The differences between these computed values and the data are squared and summed, and this value is minimized over the parameter values, using the built-in Matlab routine fminsearch. The initial values for the minimization routine are those given in the paper [KMTP94]. (In general, the modeler will not be provided with such a good initial guess!) As before, our estimations differ from those in [KMTP94], but we can attribute this to discrepancies in the actual data sets used. Warning! Due to the number of computations involved, this minimization can take quite a while, depending on the speed of the processor used. On my Dell INSPIRON 3700 laptop, for example, it took 7 minutes to run. This demo calls scripts:

- *kuzdist.m*
- *kuzode.m*
- *systemsoln.m*

Scripts for Bifurcation Module:

BifDemo1.m Draws the E and T nullclines. The figure numbers in the comments refer to Figure 2 in [KMTP94]. To shift the T-nullcline over to the left, multiply α, or par(1), by a constant, $C < 1$, and multiply β, or par(2), by $1/C$. This demo calls scripts:

- *NullE.m*
- *NullT.m*

BifDemo2.m This script file can be used to generate the graphic describing the qualitative changes in the shape of the E-nullcline as parameter values are changed.

BifDemo3.m This script file can be used to find equilibria and their eigenvalues, and to generate a plot showing the location and stability of the computed equilibria. This demo calls scripts:

- *NullE.m*
- *NullT.m*
- *NullIntersect.m*
- *KuzEquil.m*
- *EquilStability.m*
- *KuzJac.m*

BifDemo4.m This is a Matlab file for drawing stable and unstable manifolds. This routine returns the values of the computed saddle points. The file uses perturbations in the directions of the eigenvectors of the linearized system. The solutions are only computed if they stay with a certain range in the positive quadrant. Note that the subroutine files which compute the equilibria and their stability might be useful on their own. This demo calls scripts:

- *KuzEquil.m*
- *NullIntersect.m*
- *NullE.m*
- *NullT.m*
- *EquilStability.m*
- *KuzJac.m*
- *OutOfRange.m*
- *NonDimKuzOde.m*

BifDemo5.m This script file can be used to generate a bifurcation diagram. The demo calls scripts:

- *NullE.m*
- *NullT.m*
- *NullIntersect.m*
- *KuzEquil.m*
- *EquilStability.m*
- *KuzJac.m*
- *NonDimKuzOde.m*

2 Matlab Script Files

Script files are listed in alphabetical order.

BifDemo1.m

```
% Script file for graphing Null-Clines of the system described in
% Kuznetsov's paper: nonlinear dynamics of
% immunogenic tumors, Bulletin of Mathematical Biology Vol.56, 1994.
%
% This file calls the two routines NullE.m and NullT.m.
% To run the file, type it's name - without the .m extension -
% in the Command Window, followed by a carriage return.
%
% Author: A.E.Radunskaya
% August, 2002
%
%------------------------------------------------------------------------%
% The parameter values for the (non-dimensionalized) system are
% given in the vector "par", as noted below:
% par(1)=alpha;
% par(2)=beta;
% par(3)=delta;
% par(4)=eta;
% par(5)=mu;
% par(6)=rho;
% par(7)=sigma;

% The following four parameter sets correspond to parts a,b,c and d of Figure 2 of the paper
% Select one parameter set by ``uncommenting" that line.

% The parameters from Kuznetsov's paper, identified as 'Normal' fall into
% the category described in Figure 2 a): rho > (sqrt(eta*mu) + sqrt(mu))^2
% changing the value of alpha, while keeping alpha*beta the same,
% changes the number of equilibria by
% moving the y-intercept of the T-nullcline.
par=[1.636 .002 .3743 20.19 .00311 1.131 .1181];

% These parameters produce Figure 2b): (sqrt(eta*mu) - sqrt(mu))^2 < rho < (sqrt(eta*mu) + sqrt(mu))^2
% and rho > eta*mu
 %par=[1.636 .002 .3743 20.19 .00311 .5 .1181];

%These parameters produce Figure 2c): (sqrt(eta*mu) - sqrt(mu))^2 < rho < (sqrt(eta*mu) + sqrt(mu))^2
% and rho < eta*mu
%  par=[.2 .006 .5 20.19 .0311 .5 .1181];

% These parameters produce Figure 2d): rho <= (sqrt(eta*mu) -sqrt(delta))^2
%par=[1.636 .002 .3743 20.19 .00311 .05 .1181];

alpha=par(1);
beta=par(2);
delta=par(3);
```

```
eta=par(4);
mu=par(5);
rho=par(6);
sigma=par(7);

% The following lines find the nullclines and graph them.
% Some fiddling needs to be done so that points on either side of the
% asymptotes are NOT connected.

T=-100:1:700;
FofT=NullE(T,par);
I=find(FofT >=0); % Separates out the positive values of f(T)
AI=find(diff(I)>1); % Separates disconnected positive components
plot(FofT(I(1:AI)),T(I(1:AI)),'r');
hold on
if AI < length(I)
    plot(FofT(I(AI+1:end)),T(I(AI+1:end)),'r'); % if there are two components
end;
K=find(FofT < 0 ); % Separates out the negative values of f(T)
AK=find(diff(K)>1); % Separates disconnected negative components
plot(FofT(K(1:AK)),T(K(1:AK)),'r');
if AK < length(K)
    plot(FofT(K(AK+1:end)),T(K(AK+1:end)),'r'); % if there are two components
end;
GofT=NullT(T,par);
plot(GofT,T,'b')
plot([-1 max(FofT)],[0 0],'b',[0 0],[-100 700],'k'); % T=0 in blue, vertical axis in black
axis manual
axis([-1 3 -100 700]);
```

BifDemo2.m

```
% Script file to draw the bifurcation curves as determined
% by an analysis of the intersections of the graphs of the
% two nullcline functions f(T) and g(T).
%
% Author: A.E.Radunskaya
% August, 2002
%
% These parameters are those calculated as ‘‘normal":
par=[1.636 .002 .3743 20.19 .00311 1.131 .1181];

alpha=par(1);
beta=par(2);
delta=par(3);
eta=par(4);
mu=par(5);
rho=par(6);
sigma=par(7);

delta=0:.01:.5;
rho1=(sqrt(eta*mu)*ones(size(delta))-sqrt(delta)).^2;
rho2=eta*mu*ones(size(delta));
```

```
rho3=(sqrt(eta*mu)*ones(size(delta))+ sqrt(delta)).^2;

plot(delta,rho1,'r-',delta,rho2,'b-',delta,rho3,'g');
```

BifDemo3.m

```
% Script file for finding the equilibria for the system of ode's
% described in Kuznetsov's paper: nonlinear dynamics of
% immunogenic tumors, Bulletin of Mathematical Biology Vol.56, 1994.
%
% The equilibria are stored in the matrix Equil, one equilibrium per row.
% The eigenvalues are stored in the matrix Ev.
% The vector Stability stores the stability information:
%       Stability(i) = 1 iff the ith equilibrium is stable,
%       Stability(i) = -1 iff the ith equilibrium is unstable,
%       Stability(i) = 0 if the stability of the ith equilibrium is undetermined
%
% Author: A.E.Radunskaya
% August, 2002
%
% The parameter values for the (non-dimensionalized) system are
% given in the vector "par", as noted below:
% par(1)=alpha;
% par(2)=beta;
% par(3)=delta;
% par(4)=eta;
% par(5)=mu;
% par(6)=rho;
% par(7)=sigma;

% The parameters from Kuznetsov's paper:
par=[1.636 .002 .3743 20.19 .00311 1.131 .1181];

Equil=KuzEquil(par);
[Ev,Stability]=EquilStability(Equil,par);

for i=1:length(Stability)   % for each equilibrium
    switch Stability(i)
    case{1}, symbol='b*'; mcol='b';
    case{-1}, symbol='ro'; mcol='r';
    case{0}, symbol='g^'; mcol='g';
    end;
    plot(Equil(i,1),Equil(i,2),symbol,'MarkerFaceColor',mcol);
    hold on
end;
```

BifDemo4.m

```
% Draw stable and unstable manifolds through the saddle equilibria
% in the 2-dimensional ODE system given in NonDimKuzOde
%
% Author: A.E.Radunskaya
```

```
% August 2, 2002
%
function Saddles=DrawManifolds(par)
% Parameters of the system are in the vector par as follows:
% alpha=par(1);
% beta=par(2);
% delta=par(3);
% eta=par(4);
% mu=par(5);
% rho=par(6);
% sigma=par(7);

% This routines needs these M-files: KuzEquil, NullIntersect,
% NullE, NullT, EquilStability, KuzJac, NonDimKuzOde and OutOfRange

Equil=KuzEquil(par);
[Ev, Stability]=EquilStability(Equil,par);
I=find(Stability==-1);
Saddles=[Equil(I,1) Equil(I,2)];
[r,c]=size(Saddles);
options=odeset('Events',@OutOfRange);
for i=1:r
    i
    J=KuzJac(Saddles(i,:),par);
    [V,L]=eig(J);
    [v,ind]=min(diag(L));  % Find the stable eigenvector
    y1=Saddles(i,:) + V(:,ind)'/norm(V(:,ind));
    [val,term,dir]=OutOfRange(0,y1,[]);
    if val~=0
    [T1,Y1]=ode45(@NonDimKuzOde,[0 -20],y1,options,par); % Solve backwards for stable manifold
    else
    Y1=y1;
    end;
    y2=Saddles(i,:)-V(:,ind)'/norm(V(:,ind));
    [val,term,dir]=OutOfRange(0,y2,[]);
    if val~=0
    [T2,Y2]=ode45(@NonDimKuzOde,[0,-20],y2,options,par);
    else
    Y2=y2;
    end;
    [v,j]=max(diag(L));  % Find the unstable eigenvector
    y3=Saddles(i,:)+V(:,j)'/norm(V(:,j));
    [val,term,dir]=OutOfRange(0,y3,[]);
    if val~=0
    [T3,Y3]=ode45(@NonDimKuzOde,[0,50],y3,options,par); % Solve forward for the unstable manifold
    else
    Y3=y3;
    end;
    y4=Saddles(i,:)-V(:,j)'/norm(V(:,j));
    [val,term,dir]=OutOfRange(0,y1,[]);
    if val~=0
    [T4,Y4]=ode45(@NonDimKuzOde,[0,50],y4,options,par);
    else
    Y4=y4;
```

```
    end;
   plot(Y1(:,1),Y1(:,2),'m',Y2(:,1),Y2(:,2),'m',...
        Y3(:,1),Y3(:,2),'c',...
        Y4(:,1),Y4(:,2),'c','LineWidth',1);
    hold on
    clear Y1 Y2 Y3 Y4
end
axis manual
axis([0 5 0 500]);
plot(Saddles(:,1),Saddles(:,2),'mo','MarkerFaceColor','m')
```

BifDemo5.m

```
% Script file for drawing a bifurcation diagram.
% In this case, we compute the value of the tumor population
% at the equilibria of the system given in the ODE file NonDimKuzOde
% as the parameter sigma, the immune source rate, is varied.
% The other parameter values are those determined to be 'normal'.
%
% Unstable equilibria are plotted as dashed lines, and stable equilibria
% as solid lines.
%
% This file uses NonDimKuzOde,  KuzEquil, EquilStability and KuzJac
% NullIntersect, NullE, and NullT are called by KuzEquil
%
% Author: A.E.Radunskaya
% August, 2002
%
par=[1.636 .002 .3743 20.19 .00311 1.131 .1181];
% par(1)=alpha;
% par(2)=beta;
% par(3)=delta;
% par(4)=eta;
% par(5)=mu;
% par(6)=rho;
% par(7)=sigma;

sigma=0:.005:1.2;
for i=1:length(sigma)
    i
    par(7)=sigma(i);
    Equil=KuzEquil(par);
    [Ev, Stability]=EquilStability(Equil,par);
  EQ{i}=Equil;
  STAB{i}=Stability';
end;

% Plotting
for i=1:length(sigma)
     for j=1:length(STAB{i})
          switch STAB{i}(j)
            case{1}, symbol='b.';
            case{-1}, symbol='r.';
```

```
            case{0}, symbol='g';
          end;
          plot(sigma(i),EQ{i}(j,2),symbol)
          hold on
      end;
  end
```

EquilStability.m

```
% This function determines the stability of a set of equilibria
% by computing the Jacobian of the ODE and computing its eigenvalues.
% The eigenvalues are stored in the array, and the stability is stored
% in the vector Stability, with a '1' signifying stable, and a -1 signifying unstable.
%
% Author:A.E.Radunskaya
% August, 2002
%
function [Ev, Stability]=EquilStability(Equil,par)

% Equil is an array, each row of which is an equilibrium of the system in question.
% The Jacobian is computed in the routine "KuzJac".


[r,c]=size(Equil);  %  the number of rows in Equil, r, is the number of equilibria.

for i=1:r   % compute the stability of each equilibrium
    J=KuzJac(Equil(i,:),par);
    Ev(i,:)=eig(J)';
    if max(real(Ev(i,:))) < 0  % if all eigenvalues are negative, the equilibrium is stable
        Stability(i) = 1;
    elseif max(real(Ev(i,:))) > 0  % if there is at least one positive eigenvalue the equilibrium is unst
        Stability(i)=-1;
    else Stability(i) = 0;  % otherwise, the stability is undetermined
    end;  % if eigenvalues are all negative
end;  % for each equilibrium
```

fEx.m

```
% Matlab code to define the function
% f(x) = 4*x*sin(x)*(3-2*x)
% for Demo of Newton's Method in Parameter Estimation Module
%
% Author: L.G. de Pillis
% Date: August 2002

function [solution] = f(x)
solution = 4*x.*sin(x).*(3-2*x);
```

fExP.m

```
% Matlab code to define the derivative of the function
% f(x) = 4*x*sin(x)*(3-2*x)
% which is
% fp(x) = 4*sin(x)*(3-4*x) + 4*x*cos(x)*(3-2*x)
% for Demo of Newton's Method in Parameter Estimation Module
%
% Author: L.G. de Pillis
% Date: August 2002

function [solution] = fp(x)
solution = 4*sin(x).*(3-4*x) + 4*x.*cos(x).*(3-2*x);
```

kuzdist.m

```
% Calculates a distance function used to estimate parameters
% for the 2-dimensional system of equations described in
% the article by Kuznetsov et al "Nonlinear dynamics of immunogenic tumors ..."
% in the Bulletin of Mathematical Biology, Vol. 56, No 2, 1994
%
% Author: A.E.Radunskaya
% July 22, 2002

function d = kuzdist(par1,data_1, data_2, data_3, par2);
% vector par1 = [p,g,m,n]
% each data_i should have the form: data = [t1 t2 ... tn; d1 d2 ... dn], i.e. 2 rows, n cols
p = par1(1);
g = par1(2);
m = par1(3);
n = par1(4);
% These are the data from Figures 1a),b), and c) in the article
dataArray={data_1 data_2 data_3};

d=0; % initialize sum of squares to zero

for i = 1:3   % for each data set, i.e. each Figure
    %assume (t1,d1) = (0,x0)
    x0 = [3.2*(10^5) dataArray{i}(2,1)];
    % solve the ode numerically at the given times
    [T,x] = systemsoln(par1, x0, dataArray{i}(1,1:end),par2);
    % add to sum of squares
    d = d + norm(x(:,2)' - dataArray{i}(2,1:end))^2;
end;
% Rescale the computed distance for computational reasons
d = d*(10^(-15));
```

KuzEquil.m

```
% This function determines the equilibria of the system of two-dimensional
% equations given in the m-file ``Kuzode".  It does so by finding
% the intersection of the two null-clines described in the m-files NullE and NullT,
% as well as the portion of the T-nullcline: {T=0}.
```

```
% This is accomplished by the subroutine ‘‘NullIntersect"
%
% This is the system described in Kuznetsov’s paper: nonlinear dynamics of
% immunogenic tumors, Bulletin of Mathematical Biology Vol.56, 1994
%
% Author: A.E.Radunskaya
% August, 2002
%
function Equil=KuzEquil(par)

alpha=par(1);
beta=par(2);
delta=par(3);
eta=par(4);
mu=par(5);
rho=par(6);
sigma=par(7);

% Initialize the array containing computed equilibria.
Equil=[];

% Loop through possible starting values for the routine fzero.
% If a new zero is found, store it in the array Equil.

options=optimset('TolX',.000001);
for startT=.1:10:500  % go one beyond the carrying capacity for the tumor cells
    [z,fval]=fzero(@NullIntersect,startT,options,par);
    ez=alpha*(1-beta*z);
    if abs(fval) < .0001  % if fzero has been successful
        if isempty(Equil)
            Equil(1,2)=z;
            Equil(1,1)=ez;
            Lastz=z;
            i=1;
        elseif min(abs(ez-Equil(:,1))>.001)  % if we haven't yet found this equilibrium, store it
            i=i+1;
            Equil(i,2)=z;
            Equil(i,1)=ez;
            Lastz=z;
        end;  % if isempty or if not yet found
    end;   % if fval is small enough
end; % for loop

% Add the tumor-free equilibrium.
Equil(i+1,2)=0;
Equil(i+1,1)=sigma/delta;
```

KuzJac.m

```
% This function returns the Jacobian of the ODE described in
% Kuznetsov’s paper: nonlinear dynamics of
% immunogenic tumors, Bulletin of Mathematical Biology Vol.56, 1994
```

```
% at the equilibrium given by the vector Eq, with parameter values given in
% the vector par
%
% Author: A.E.Radunskaya
% August, 2002
%
function J=KuzJac(Eq,par);

alpha=par(1);
beta=par(2);
delta=par(3);
eta=par(4);
mu=par(5);
rho=par(6);
sigma=par(7);

E=Eq(1);
T=Eq(2);

J(1,1) = (rho*T)/(eta+T) - mu*T - delta;
J(1,2)= (rho*eta*E)/((eta + T)^2) - mu*E;
J(2,1)= -T;
J(2,2)=alpha - 2*alpha*beta*T - E;
```

kuzode.m

```
% Calculates a distance function used to estimate parameters
% for the 2-dimensional system of equations described in
% the article by Kuznetsov et al "Nonlinear dynamics of immunogenic tumors ..."
% in the Bulletin of Mathematical Biology, Vol. 56, No 2, 1994
%
% Author: A.E.Radunskaya
% July 22, 2002

function xdot=kuzode(tn,x,par1,par2)

% Equations 4a) and 4b) in the article

p=par1(1);
g=par1(2);
m=par1(3);
n=par1(4);

a=par2(1);
b=par2(2);
s=par2(3);
d=par2(4);

E=x(1);
T=x(2);

xdot=[s+(p*E*T/(g+T))-m*E*T-d*E;
      a*T*(1-b*T)-n*E*T];
```

logisticdist.m

```
% Calculates a distance function used to estimate parameters
% for the 2-dimensional system of equations described in
% the article by Kuznetsov et al "Nonlinear dynamics of immunogenic tumors ..."
% in the Bulletin of Mathematical Biology, Vol. 56, No 2, 1994
%
% Author: A.E.Radunskaya
% July 22, 2002

function d = logisticdist(x,data);
% vector x = [a,b]
% data should have the form: data = [t1 t2 ... tn; d1 d2 ... dn], so 2 rows, n cols
a = x(1);
b = x(2);
x0 = data(2,1); %assume (t1,d1) = (0,x0)
xt = logisticsoln(data(1,2:end),x0,a,b);
d = norm(xt - data(2,2:end))^2; % sum of squares
```

logisticsoln.m

```
% Calculates the solution to the logistic differential equation,
% used to estimate parameters for the 2-dimensional system of equations described in
% the article by Kuznetsov et al "Nonlinear dynamics of immunogenic tumors ..."
% in the Bulletin of Mathematical Biology, Vol. 56, No 2, 1994
%
% Author: A.E.Radunskaya
% July 22, 2002

function x = logisticsoln(t,x0,a,b);
c = 1/x0 - b;
x = 1./(c.*exp(-a*t) + b);
```

NonDimKuzOde.m

```
function xdot=NonDimKuzOde(t,x,par)

% The ODE for the non-dimensionalized system of differential equations in
% Kuznetsov's paper: nonlinear dynamics of
% immunogenic tumors, Bulletin of Mathematical Biology Vol.56, 1994

alpha=par(1);
beta=par(2);
delta=par(3);
eta=par(4)*ones(size(x(2,:)));
mu=par(5);
rho=par(6);
sigma=par(7)*ones(size(x(2,:)));

xdot=[sigma + (rho*x(1,:).*x(2,:))./(eta + x(2,:)) - mu*x(1,:).*x(2,:) - delta*x(1,:);
       alpha * x(2,:) .*(ones(size(x(2,:))) - beta*x(2,:)) - x(1,:).*x(2,:)];
```

NullE.m

```
function E=NullE(T,par)
% This function determines the value of the effector cells along
% the null-cline: dE/dt = 0 for a given value of T, using the
% (non-dimensional) parameters in the vector par.
%
% This is equation (7) in Kuznetsov's paper: nonlinear dynamics of
% immunogenic tumors, Bulletin of Mathematical Biology Vol.56, 1994


alpha=par(1);
beta=par(2);
delta=par(3);
eta=par(4);
mu=par(5);
rho=par(6);
sigma=par(7);

E=sigma./(delta + mu*T - (rho*T./(eta + T)));
```

NullIntersect.m

```
function z=NullIntersect(T,par)
% Find the intersection of the two nullclines
% described by NullE and NullT
z=NullE(T,par)-NullT(T,par);
```

NullT.m

```
function E=NullT(T,par)
% This function determines the value of the effector cells along
% the null-cline: dT/dt = 0 for a given value of T not equal to zero, using the
% (non-dimensional) parameters in the vector par.
%
% This is the equation for g(y) in Kuznetsov's paper: nonlinear dynamics of
% immunogenic tumors, Bulletin of Mathematical Biology Vol.56, 1994


alpha=par(1);
beta=par(2);
delta=par(3);
eta=par(4);
mu=par(5);
rho=par(6);
sigma=par(7);


E=alpha*(1 - beta*T);
```

NumDemo1.m

```
% Main Matlab routine to call Forward Euler method.
% Demo example from BFR p.188
%
% Author: L.G. de Pillis
% Date: August 7, 2002

tinit = 0.00;
tfinal = 1.50;
tstep = 0.1;
tspan = [tinit:tstep:tfinal];
y0 = 1.0;

%Demo of Exponential DE: MATLAB vs EULER solve
[t1,y1] = ode45(@ODE0,tspan,y0);

h = .01; %Specific stepsize needed for EulerSolve routine
[t2,y2] = EulerSolve(@ODE0,tspan,y0,h);


figure(1);
plot(t1,y1,'ro');
hold on;
plot(t2,y2,'g-');
plot(t1,ODE0soln(t1),'m+'); %true solution
T1 = sprintf('Euler Solver h = %g',h);
legend('ODE45 solver',T1,'True Solution');
T = sprintf('ODE dy/dt = 3y, y0 = %g',y0);
title(T);
```

NumDemo2.m

```
% Main Matlab routine to call various ODE solvers,
% both built-in Matlab solvers and self-written solvers.
% Functions will be stiff or non-stiff.
% Run to compare Euler method to other methods.  Try to
% modify stepsize h and IC y0 for the Stiff Eqn, so you
% can demo how Euler goes unstable.
%
% Author: L.G. de Pillis
% Date: August 7, 2002
% Example from Kahaner, Moler, Nash p. 290
tinit = 0.00;
tfinal = 6.00;
tstep = 0.1;
tspan = [tinit:tstep:tfinal];
y0 = exp(-5);
h = 0.10001;
[t7,y7] = EulerSolve(@ODE2,tspan,y0,h);
figure(3); clf; hold off;
plot(t7,y7,'b-',t7,ODE2soln(t7),'m+');
```

```
legend('Euler solver','True Solution');
T = sprintf('EM h = %g, ODE dy/dt = -10(t-1)y, y0 = %g',h,y0);
xlabel('Time t');
ylabel('Solution y(t)');
title(T);
```

NumDemo3.m

```
% Main Matlab routine to compare EM to BE
% Function in "stiffODE" borrowed from Heath p.402
%
% Plots results in table to standard output.
%
% Also plots results graphically.
% Use both built-in Matlab solvers and self-written solvers.
% Modify stepsize h and IC y0 for the Stiff Eqn, so you
% can demo how Euler goes unstable.
%
% Author: L.G. de Pillis
% Date: August 7, 2002

tinit = 0.00;
tfinal = 0.50;
tstep = 0.1;
tspan = [tinit:tstep:tfinal];

%Print table
%Note: it may be effective to print the values in a table:
T1 = sprintf('Time t:       %4.2f     %4.2f     %4.2f    %4.2f     %4.2f \n', t4(1:5));
T2 = sprintf('Exact Soln:   %4.2f     %4.2f     %4.2f    %4.2f     %4.2f \n', stiffODEsoln(t4(1:5)));
fprintf(1,T1);
fprintf(1,T2);

%Demo of Stiff DE: MATLAB vs EULER solve
%First EM
y0 = 0.99;
h = .1; %Specific stepsize needed for EulerSolve routine
[t5,y5] = EulerSolve(@stiffODE,tspan,y0,h);
T3 = sprintf('EM y0=%3.2f:   %4.2f     %4.2f     %4.2f    %4.2f     %4.2f \n', y0, y5(1:5));
fprintf(1,T3);

%Demo of Stiff DE: MATLAB vs EULER solve
%Second EM
y0 = 1.01;
h = .1; %Specific stepsize needed for EulerSolve routine
[t5,y5] = EulerSolve(@stiffODE,tspan,y0,h);
T3 = sprintf('EM y0=%3.2f:   %4.2f     %4.2f     %4.2f    %4.2f     %4.2f \n', y0, y5(1:5));
fprintf(1,T3);

%Demo of Stiff DE: MATLAB vs EULER solve
%First BE
y0 = 0.00;
h = .1; %Specific stepsize needed for EulerSolve routine
```

```
[t6,y6] = BackEulerSolve(@stiffODE,tspan,y0,h);
T4 = sprintf('BE y0=%3.2f:   %4.2f     %4.2f     %4.2f    %4.2f     %4.2f \n', y0, y6(1:5));
fprintf(1,T4);

%Demo of Stiff DE: MATLAB vs EULER solve
%Second BE
y0 = 2.00;
h = .1; %Specific stepsize needed for EulerSolve routine
[t6,y6] = BackEulerSolve(@stiffODE,tspan,y0,h);
T4 = sprintf('BE y0=%3.2f:   %4.2f     %4.2f     %4.2f    %4.2f     %4.2f \n', y0, y6(1:5));
fprintf(1,T4);

%===If desired, Run Codes and plot solutions to
%===visually compare ODE15s with EM and BE
y0 = 1.0;
%y0 = 0.99; %perturb the ICs
%y0 = 1.01; %perturb the ICs
%y0 = 0.00; %perturb the ICs
%y0 = 2.00; %perturb the ICs

[t4,y4] = ode15s(@stiffODE,tspan,y0); %use y0=1 for stiffODE
%Note: for Heath p.402 example, must have h < .02 for stability.  Try modifying this.
%h = .001;  %Specific stepsize needed for EulerSolve routine
h = .1;     %Specific stepsize needed for EulerSolve routine
[t5,y5] = EulerSolve(@stiffODE,tspan,y0,h);
[t6,y6] = BackEulerSolve(@stiffODE,tspan,y0,h);

figure(2); clf; hold off;
plot(t4,y4,'ro');
hold on;
plot(t5,y5,'g-');
plot(t6,y6,'b-');
plot(t4,stiffODEsoln(t4),'m+'); %true solution
T1 = sprintf('Euler Solver h = %g',h);
T2 = sprintf('BackEuler Solver h = %g',h);
legend('ODE15s solver',T1,T2,'True Solution');
T = sprintf('ODE dy/dt = -100y + 100t + 101, y0 = %g',y0);
title(T);
xlabel('t');
ylabel('y(t)');
```

ODE0.m

```
% Example of an ODE
% Written: July 29, 2002
% Author: L.G. de Pillis

function yp = ODE0(t,y)

yp = -y + t + 1;
```

ODE0soln.m

```
% Example of an ODE solution
% Written: July 29, 2002
% Author: L.G. de Pillis
% Taken from BFR p. 188

function y = ODE0soln(t,y)

%solution of yp = -y + t + 1;
y = t + exp(-t);
```

ODE2.m

```
% Example of an ODE, borrowed from KMN p.289
% Written: July 29, 2002
% Author: L.G. de Pillis

function yp = ODE2(t,y)

yp = -10*(t-1)*y;
```

ODE2soln.m

```
% Example of an ODE solution
% Borrowed from KMN p. 287, ex.8.4, also p. 290
% Written: July 29, 2002
% Author: L.G. de Pillis

function y = ODE2soln(t,y)

%solution of yp = -10*(t-1)*y;
y = exp(-5.*((t-1).^2));
```

OutOfRange.m

```
function [VALUE,ISTERMINAL,DIRECTION]=OutOfRange(T,Y,flag);
% Determines whether to stop the Ode Solver when the solution
% is out of the window given in WINDOW=[Xmin Xmax Ymin Ymax]
% This function is used as the Events function in the ODE options
%
WINDOW=[0 5 0 500];
ISOUT=min([Y(1)-WINDOW(1) WINDOW(2)-Y(1) Y(2)-WINDOW(3) WINDOW(4)-Y(2)]);
VALUE=ISOUT>=0;
ISTERMINAL=1;
DIRECTION=0;
```

ParDemo1.m

```
% Matlab program to carry out Newton's method
% for finding the root of a function f(x).
```

```
% Use, for example, fEx for the function and fExP for the
% derivative of the function to be evaluated.
% Author: Lisette de Pillis
% Date: September 15, 1993
% Updated: August 29, 2002

fprintf('NEWTONs METHOD \n')
fprintf('This is a program to determine the root of \n')
fprintf('the function f(x) (user determined) given \n')
fprintf('the user specified initial guess x0. \n')
fprintf('\n')
%-------------------------------------------------------------
range = input('Input a range [a b] over which to plot the function: ');

TOL = .00001; %Error tolerance
M   = 1000;   %Maximum iterations allowed
f   = 'fEx';  %Name of function to be evaluated
fp  = 'fExP'; %Name of derivative of function to be evaluated

%Start to PLOT results
hold off; clf;
x = [range(1):(range(2)-range(1))/50:range(2)];
plot(x,feval(f,x),'b');
hold on;
%Plot the x-axis
plot([x(1) x(end)],[0 0],'k');

p0 = input('Initial root approximation, x_0  =  ');
while (p0 ~= 9999)
p(1) = p0;
i = 2;
while (i <= M+1)
  p(i) = p0 - feval(f,p0)/feval(fp,p0);
%  if  ((abs(p0 - p(i)) < TOL)  & (feval(f,p(i)) < TOL) ) %use for stricter convergence
  if  (abs(p0 - p(i)) < TOL)
    fprintf('\n ')
    fprintf('Solution x = %g. \n ',  p(i))
    fprintf('Function f(x) = %g. \n ', feval(f,p(i)))
    break
  end; % End If
  p0 = p(i);
  i = i+1;
end; %End While
if (i>M+1)
  fprintf('Method failed after M iterations, M = %g. \n', M)
else
  fprintf('Method succeeded in M iterations, M = %g. \n', i)
end;

%PLOT results
vec1 = [];
vec2 = zeros(1,length(p));
for i=1:length(p)
  vec1 = [vec1 p(i) p(i)];
```

```
  vec2(2*i) = fEx(p(i));
end; %for
plot(vec1(1),vec2(1),'ro','MarkerFaceColor','r'); %plot initial guess
legend('f(x)','x-axis','Initial Guess x0');
%Plot the iterations
plot(vec1,vec2,'r'); %plot line segments
plot(vec1(2:end),vec2(2:end),'ro'); %put dots at subsequent iterations
xlabel('x'); ylabel('f(x)'); title('Newton Method Iteration');

p0 = input('Initial root approximation, x_0 (type 9999 to finish) =  ');
end; %while
```

ParDemo2.m

```
% Main Matlab script file for estimating the parameters
% a and b, as is done in
% the article by Kuznetsov et al "Nonlinear dynamics of immunogenic tumors ..."
% in the Bulletin of Mathematical Biology, Vol. 56, No 2, 1994
%
% Author: A.E.Radunskaya
% Date: July 22, 2002

% This file calls the function 'logisticdist', which in turn calls 'logisticsoln'.
% The data below is taken off of the graph in Figures 1a) in the article.
%
% The data is in the following form:
% first row = time values, second row = number of tumor cells

data = [0     (14/13)     (20/13) (33/13)     (39/13)     (45.5/13)    (58/13);
  (500000) (10^(22/3)) (10^8)  (10^(25/3)) (10^(42/5)) (10^(76/9))  (10^(77/9))];

data(1,:)=20*data(1,:); % scale time values by 20

% Find best fitting parameter values, returned in x.

% Note: the data input above is derived from Kuznetsov's  paper.
% Our starting a and b values are Kuznetsov's results.

options=optimset('MaxFunEvals',10000,'MaxIter',10000);
[x,fval] = fminsearch(@logisticdist,[0.18,2*10^(-9)],options,data);
tvec = [0:0.1:data(1,end)+1]; % go one time step beyond the data time
semilogy(tvec,logisticsoln(tvec,data(2,1),x(1),x(2)),'b-',data(1,:),data(2,:),'ro');
```

ParDemo3.m

```
% Main Matlab script file for estimating the parameters
% p,m,n and g, as is done in
% the article by Kuznetsov et al "Nonlinear dynamics of immunogenic tumors ..."
% in the Bulletin of Mathematical Biology, Vol. 56, No 2, 1994
%
% Author: A.E.Radunskaya
```

```
% Date: July 22, 2002

% This file calls the function 'kuzdist', which in turn calls 'kuzode'.
% The data below is taken off of the graphs in Figures 1 b), c) and d) in the article.
%
% The data is in the following form: first row = time values, second row = actual data
% these represent Kuznetsov's data derived from varying the initial
% number of tumor cells. These three data sets correspond to parts
% a), b), and c) of figure 1 in K's paper.

data_1 = [0     (14/13)     (20/13) (27/13)     (32/13)     (45/13)    (58/13)   (71/13);
       (10^(5+(9/13))) (10^(5+(24/13))) (10^(5+(35/13))) (10^(5+(34/13))) (10^(5+(32/13)))...
       (10^(5+(19/13)))  (10^(5+(12/13))) (10^(5+(11/13)))]; % logarithms are converted

data_1(1,:)=20*data_1(1,:); % scale time values by 20

data_2 = [0          1            (37/24)     (7/2);
         (10^(27/4)) (10^(95/12)) (10^(49/6)) (10^(15/2))];

data_2(1,:)=20*data_2(1,:);

data_3 = [0           (8/13)        (14/13)        2             3            (46/13)      (59/13)      (
         (10^(100/13)) (10^(108/13)) (10^(112/13)) (10^(109/13)) (10^(102/13)) (10^(99/13)) (10^(81/13))

data_3(1,:)=20*data_3(1,:);

par1guess = [0.125 (2.019 * (10^7)) (3.422*(10^(-10))) (1.101*(10^(-7)))];
% we use K.'s estimated values as our starting points for the parameters p,g,m, and n,
% which should be contained in that order in the vector par1guess.

par2 = [0.18 (2*(10^(-9))) (1.3*(10^4)) 0.0412];
% the vector par2 contains the variables a, b, s, d which are passed to the ODE solver.

x0_1 = [(3.2*(10^5)) data_1(2,1)];
x0_2 = [(3.2*(10^5)) data_2(2,1)];
x0_3 = [(3.2*(10^5)) data_3(2,1)];

% Find best fitting parameter values, returned in x.
% We want fval to be as close to zero as possible, since we want
% the parameters we find to minimize the value of the function
% kuzdist.  "kuzdist" is actually a function which is the
% sum of squares of differences between given data points at given time points
% and the solutions of an ODE evaluated at those time points.

options=optimset('Display','iter', 'MaxFunEvals',1200);
[par1,fval] = fminsearch(@kuzdist,par1guess,options,data_1,data_2,data_3,par2);
tvec_1 = [0:0.1:data_1(1,end)+1]; % go one time step beyond the data time
tvec_2 = [0:0.1:data_2(1,end)+1];
tvec_3 = [0:0.1:data_3(1,end)+1];
clf
[T1,X1]=systemsoln(par1,x0_1,tvec_1,par2);
[T2,X2]=systemsoln(par1,x0_2,tvec_2,par2);
[T3,X3]=systemsoln(par1,x0_3,tvec_3,par2);
```

```
semilogy(tvec_1,X1(:,2),'b-',data_1(1,:),data_1(2,:),'bo');
hold on
semilogy(tvec_2,X2(:,2),'r-',data_2(1,:),data_2(2,:),'ro');
semilogy(tvec_3,X3(:,2),'g-',data_3(1,:),data_3(2,:),'go');
```

QualDemo1.m

```
% Script file for plotting the nullclines and equilibria for the system of ode's
% described in Kuznetsov's paper: nonlinear dynamics of
% immunogenic tumors, Bulletin of Mathematical Biology Vol.56, 1994.
%
% The file will generate several workspace variables, including
%       Equil: a matrix whose rows are the equilibria of the system
%       Ev:    a matrix whose rows are the eigenvalues of the equilibria
%       par:   a vector containing the 7 system parameters:
%           par=[alpha, beta, delta, eta, mu, rho, sigma].
%
% This file calls the two script files: BifDemo1.m and BifDemo3.m
% If another parameter set is to be used, it must be changed in BOTH files.
%
% Author: A.E.Radunskaya
% August, 2002
%
clf
BifDemo1
BifDemo3
xlabel('E: Effector Cells')
ylabel('T: Tumor Cells')
```

stiffODE.m

```
% Example of a stiff ODE, borrowed from Heath p.402
% Written: July 29, 2002
% Author: L.G. de Pillis

function yp = sODE(t,y)

yp = -100*y + 100*t + 101;
```

stiffODEsoln.m

```
% Solution to stiffODE equation
% Borrowed from Heath p.402

function y = soln(t)

y = 1 + t;
```

systemsoln.m

```
function [T,x]=systemsoln(par1,x0,tn,par2)

% Numerically solves the system from
% for the 2-dimensional system of equations described in
% the article by Kuznetsov et al "Nonlinear dynamics of immunogenic tumors ..."
% in the Bulletin of Mathematical Biology, Vol. 56, No 2, 1994
%
% Author: A.E.Radunskaya
% July 22, 2002
%
% The parameter values are in the vectors par1 and par2,  with initial
% conditions E_0=x0(1), T_0=x0(2) at time values in the vector tn.
%
% We assume the parameters a, b, s, and d are given in par2, in that order
% and the remaining parameters are in par1.


p=par1(1);
g=par1(2);
m=par1(3);
n=par1(4);

[T,x]=ode45(@kuzode,tn,x0,[],par1, par2);
```

References

[KMN89] David Kahaner, Cleve Moler, and Stephen Nash. *Numerical Methods and Software*. Prentice Hall Series in Computational Mathematics. Prentice-Hall, 1989.

[KMTP94] Vladmir A. Kuznetsov, Iliya A. Makalkin, Mark A. Taylor, and Alan S. Perelson. Nonlinear dynamics of immunogenic tumors: Parameter estimation and global bifurcation analysis. *Bulletin of Mathematical Biology*, 56(2), 1994.