**Annotated Model Code**

**\*all code scripted & carried out in MATLAB unless otherwise stated**

*Stability Analysis for Predator-Prey Models:*

%% Stability Analysis Plot (R-M Pred-Prey) %%

syms r N K c P s e d

% The model equations:

dN = r\*N\*(1-N/K) - c\*N\*P/(s+N);

dP = e\*(c\*N\*P/(s+N)) - (d\*P);

% Find the equilibrium (set both equations to zero and solve for N,P)

Eq = solve( r\*N\*(1-N/K) - c\*N\*P/(s+N) == 0,...

e\*(c\*N\*P/(s+N)) - (d\*P) == 0,...

N,P);

% Structure Eq holds the results in fields N, P.

% There are three roots: first one has N>0 & P>0, second is both extinct,

% third is P=0, N=K. So the first is what we want.

% Jacobian. For local stability analysis, we require the Jacobian to be

% calculated with respect to the state variables (so it represents rate of

% return to equilibrium due to minor perturbations of state vars)

J = jacobian([dN; dP], [N P]);

% Jacobian must be evaluated at the equilibrium. Substitute in the

% equilibrium solution:

J = subs(J,[N,P],[Eq.N(1),Eq.P(1)]);

% Now evaluate the Jacobian over a range of the values you are interested

% in:

% c vs. d:

Cs = 0.5:0.5:10;

Ds = 0.01:0.01:0.2;

% c vs. K:

%Cs = 0.5:0.5:10;

%Ks = 0.1:0.1:2;

% c vs. s:

%Cs = 0.5:0.5:10;

%ss = 0.1:0.1:2;

% Other parameters are just fixed at a constant value: (again, just making

% up numbers for now)

rc = 0.1;

Kc = 0.7;

sc = 1;

ec = 0.1;

%dc = 0.1;

% Loop over the two parameters of interest and calculate the eigenvalues:

Eigs = nan(length(Cs),length(Ds)); % pre-allocate the variable

for i = 1:length(Cs)

for j = 1:length(Ds)

try

J\_num = subs(J, [c, d, r, K, s, e], [Cs(i),Ds(j),rc,Kc,sc,ec]); % substitute in the values for each parameter

Eigs(i,j) = max(real(eig(eval(J\_num)))); % calculate the maximum (dominant)

%eigenvalue (have to use eval to make it actually calculate the value

%rather than report the symbolic expression)

end

end

end

% Make a pretty figure

figure(1)

clf

hold on

pcolor(Cs,Ds,Eigs) % plot the eigenvalues

shading interp

contour(Cs,Ds,Eigs,[-1,0,1],'linecolor',[0.85 0.85 0.85]) % plot contour lines to denote critical regions

colormap(gray); % make gray scale

set(gca,'tickdir','out','ticklength',[0.015 0.015],'fontsize',12)

xlabel('Predator attack rate (a)','fontsize',14)

ylabel('Predator death rate (d)','fontsize',14)

ch=colorbar;

lb = ylabel(ch, 'Eigenvalue ({\lambda})','fontsize',14);

set(ch,'tickdir','out','ticklength',0.015)

axis square

%axis equal

brighten(0.3); % The colors brighten when value is between 0 and 1,

%and they darken when value is between -1 and 0

print -dpng NumericalStabilityAnalysisPlot\_AvsD\_3.png;

*Predator-Prey (Type II) Model Equations:*

function RMPredPreyTypeII = PredPreyTypeII(t,x,Params)

% Rosenzweig-MacArthur (1963) predator-prey model (Type II functional response):

% dN/dt = rN(1-N/K) - aNP/(1+hN) % change in prey (N) over time

% dP/dt = e(aNP/(1+hN)) - dP % change in predators (P) over time

% where r = per capita growth rate, K = carrying capacity, a = per capita

% attack rate, h = handling time,

% e = biomass conversion efficiency (prey consumed resulting in predator

% offspring), d = predator death rate

% Starting param values:

a = Params(1);

r = Params(2);

d = Params(3);

h = Params(4);

e = Params(5);

K = Params(6);

% Generates a vector of zeros the same size as the vector x. The 0's are

% placeholders for dN/dt and dP/dt values:

RMPredPreyTypeII = zeros(size(x));

% Input vector, x: x(1) = N and x(2) = P

% Calculates dN/dt, the first value in the RMPredPrey vector:

RMPredPreyTypeII(1) = ((r\*x(1))\*(1-(x(1)/K))) - ((a\*x(1)\*x(2))/(1+(h\*x(1))));

% Calculates dP/dt, the second value in the RMPredPrey vector:

RMPredPreyTypeII(2) = e\*((a\*x(1)\*x(2))/(1+(h\*x(1)))) - (d\*x(2));

end

*Predator-Prey (Type III) Model Equations:*

function RMPredPreyTypeIII = PredPreyTypeIII(t,x,Params)

% Rosenzweig-MacArthur (1963) predator-prey model (Type III functional response):

% dN/dt = rN(1-N/K) - aN^2P/(1+hN^2) % change in prey (N) over time

% dP/dt = e(aN^2P/(1+hN^2)) - dP % change in predators (P) over time

% where r = per capita growth rate, K = carrying capacity, a = per capita

% attack rate, h = handling time,

% e = biomass conversion efficiency (prey consumed resulting in predator

% offspring), d = predator death rate

% Starting param values:

a = Params(1);

r = Params(2);

d = Params(3);

h = Params(4);

e = Params(5);

K = Params(6);

% Generates a vector of zeros the same size as the vector x. The 0's are

% placeholders for dN/dt and dP/dt values:

RMPredPreyTypeIII = zeros(size(x));

% Input vector, x: x(1) = N and x(2) = P

% Calculates dN/dt, the first value in the RMPredPrey vector:

RMPredPreyTypeIII(1) = ((r\*x(1))\*(1-(x(1)/K))) - ((a\*(x(1)^2)\*x(2))/(1+(h\*(x(1)^2))));

% Calculates dP/dt, the second value in the RMPredPrey vector:

RMPredPreyTypeIII(2) = e\*((a\*(x(1)^2)\*x(2))/(1+(h\*(x(1)^2)))) - (d\*x(2));

end

*Predator-Prey Disturbance Model:*

function [ExtinctProb] = PredPreyDisturbanceModel(dyn,freqmax,parm,mu,sd,thr,fig)

%% Rosenzweig-MacArthur (1963) predator-prey model with varying disturbance frequency and magnitude %%

% time sequence:

tf = 500;

tspan = 0:1:500;

for i = 1:2500

switch dyn % pred-prey dynamics

case 'noncyclic2' % non-cyclic coexistence (Type II)

% parameters (Params 1-6 specified in function PreyPreyTypeII.m):

Params = [5; 0.1; 0.1; 1; 0.1; 0.7];

% initial prey and pred. pop. sizes (respectively)

% (coexistence steady state equilibrium (mean) pop. values):

x\_temp = [0.2502 0.0164];

case 'cyclic2' % cyclic coexistence (Type II)

% parameters:

Params = [5; 0.1; 0.1; 1; 0.1; 2];

% initial prey and pred. pop. sizes

% (stable limit cycles equilibrium pop. values):

x\_temp = [0.3214 0.0224];

case 'noncyclic3' % non-cyclic coexistence (Type III)

% parameters (Params 1-6 specified in function PreyPreyTypeIII.m):

Params = [5; 0.1; 0.1; 1; 0.1; 5];

% initial prey and pred. pop. sizes (respectively)

% (coexistence steady state equilibrium pop. values):

x\_temp = [0.5049 0.0448];

case 'cyclic3' % cyclic coexistence (Type III)

% parameters:

Params = [0.9; 0.1; 0.6; 1; 1; 15];

% initial prey and pred. pop. sizes

% (stable limit cycles equilibrium pop. values):

x\_temp = [3.5672 0.3419];

end

% disturbance frequency:

freqn = 500;

freq = rand(1,freqn) <= freqmax;

freq(end) = false;

% create empty vectors to hold the ODE results

T = [];

X = [];

if freq == 0

[t\_temp, x\_temp] = ode45(@(t,x) PredPreyTypeII(t,x,Params),tspan,x\_temp);

% Type II differential equations specified in file PredPreyTypeII.m

% PredPreyTypeII.m = Rosenzweig-MacArthur predator-prey model (Type II functional response)

% Type III differential equations specified in file PredPreyTypeIII.m

% PredPreyTypeIII.m = Rosenzweig-MacArthur predator-prey model (Type III functional response)

T = [T; t\_temp]; % add in the new results to the growing vector of results

X = [X; x\_temp];

else

starts = find(freq);

[t\_temp, x\_temp] = ode45(@(t,x) PredPreyTypeII(t,x,Params),[0,starts(1)],x\_temp);

T = [T; t\_temp]; % add in the new results to the growing vector of results

X = [X; x\_temp];

for s = 1:length(starts)

switch parm

case 'attack'

% disturbance magnitude (normally dist rand num): (mean, stdev, row, column)

mag = normrnd(mu,sd,1,500);

Param\_tmp = Params;

% apply disturbance: increase pred. attack rate- Params 1 = c

Param\_tmp(1) = Params(1) + mag(starts(s));

% keeps abundances non-negative

x\_temp = max(0,x\_temp);

[t\_temp, x\_temp] = ode45(@(t,x) PredPreyTypeII(t,x,Param\_tmp),[starts(s), starts(s)+1],x\_temp(end,:));

case 'mort'

% disturbance magnitude (normally dist rand num): (mean, stdev, row, column)

mag = normrnd(mu,sd,1,500);

Param\_tmp = Params;

% apply disturbance: increase/decrease (+/-) pred. mortality rate- Params 3 = d

Param\_tmp(3) = Params(3) - mag(starts(s));

% keeps abundances non-negative

x\_temp = max(0,x\_temp);

[t\_temp, x\_temp] = ode45(@(t,x) PredPreyTypeII(t,x,Param\_tmp),[starts(s), starts(s)+1],x\_temp(end,:));

case 'abundance'

% apply disturbance: knock down abundance- N, P

mag = normrnd(mu,sd,1,500);

x\_tmp = x\_temp(end,:) \* mag(starts(s));

% keeps abundances non-negative

x\_tmp = max(0,x\_tmp);

[t\_temp, x\_temp] = ode45(@(t,x) PredPreyTypeII(t,x,Params),[starts(s), starts(s)+1],x\_tmp);

end

T = [T; t\_temp]; % add in the new results to the growing vector of results

X = [X; x\_temp];

if s < length(starts)

if starts(s+1) > starts(s)+1

[t\_temp, x\_temp] = ode45(@(t,x) PredPreyTypeII(t,x,Params),[starts(s)+1, starts(s+1)],x\_temp(end,:));

T = [T; t\_temp]; % add in the new results to the growing vector of results

X = [X; x\_temp];

end

else

[t\_temp, x\_temp] = ode45(@(t,x) PredPreyTypeII(t,x,Params),[starts(s), tf],x\_temp(end,:));

T = [T; t\_temp]; % add in the new results to the growing vector of results

X = [X; x\_temp];

end

end

end

T\_outer{i} = T; % store each simulation

X\_outer{i} = X;

end

% probability of extinction:

if strcmp(dyn, 'noncyclic2')

Extinctions = nan(length(X\_outer),2); % pre-allocate

for i = 1:length(X\_outer)

X = X\_outer{i};

Equilib = [0.2502 0.0164];

Threshold = thr\*Equilib;

Extinctions(i,1) = any(X(:,1) <= Threshold(1,1));

Extinctions(i,2) = any(X(:,2) <= Threshold(1,2));

end

elseif strcmp(dyn, 'cyclic2')

Extinctions = nan(length(X\_outer),2);

for i = 1:length(X\_outer)

X = X\_outer{i};

Equilib = [0.3214 0.0224];

Threshold = thr\*Equilib;

Extinctions(i,1) = any(X(:,1) <= Threshold(1,1));

Extinctions(i,2) = any(X(:,2) <= Threshold(1,2));

end

elseif strcmp(dyn, 'noncyclic3')

Extinctions = nan(length(X\_outer),2);

for i = 1:length(X\_outer)

X = X\_outer{i};

Equilib = [0.5049 0.0448];

Threshold = thr\*Equilib;

Extinctions(i,1) = any(X(:,1) <= Threshold(1,1));

Extinctions(i,2) = any(X(:,2) <= Threshold(1,2));

end

elseif strcmp(dyn, 'cyclic3')

Extinctions = nan(length(X\_outer),2);

for i = 1:length(X\_outer)

X = X\_outer{i};

Equilib = [3.5672 0.3419];

Threshold = thr\*Equilib;

Extinctions(i,1) = any(X(:,1) <= Threshold(1,1));

Extinctions(i,2) = any(X(:,2) <= Threshold(1,2));

end

else

end

ExtinctProb = mean(Extinctions);

if fig == 0

else

if fig == 1

f = 1 + (2500-1).\*rand(1,1);

f = round(f);

% Plot one of the pop. time series iterations

% f = iteration number

% Plots prey values in blue with circles and predator values in red with \*'s:

fig = figure;

plot(T\_outer{1,f},X\_outer{1,f}(:,1),'k',T\_outer{1,f},X\_outer{1,f}(:,2),'color',repmat(0.7,[1,3]),'LineWidth',3)

xlabel('Time')

ylabel('Population Size')

title('Predator-Prey Population Time Series')

legend('Prey', 'Predator')

dateformat = 'mm-dd-yy';

date = datestr(now,dateformat);

filename = strcat(date, '\_Pop Time Series\_', num2str(f), '.eps');

set(gca,'tickdir','out','ticklength',[0.02 0.02])

print(fig, '-depsc2', filename);

% Plot one of the phase portraits of the iterations

% Plots the phase portrait of prey values versus predator values:

fig2 = figure;

plot(X\_outer{1,f}(:,1),X\_outer{1,f}(:,2))

xlabel('Prey')

ylabel('Predator')

title('Predator-Prey Phase Portrait')

date2 = datestr(now,dateformat);

filename2 = strcat(date2, '\_Phase Portrait\_', num2str(f), '.eps');

print(fig2, '-depsc2', filename2);

end

end

end

*Predator-Prey Disturbance Model Script:*

%% PredPreyDisturbanceModel Script %%

% Inputs:

dyn = 'noncyclic2';

% 'noncyclic2' = non-cyclic coexistence (Type II)

% 'cyclic2' = cyclic coexistence (Type II)

% 'noncyclic3' = non-cyclic coexistence (Type III)

% 'cyclic3' = cyclic coexistence (Type III)

freqmax = 0:0.1:1;

% disturbance frequency, 0 to 100%

parm = 'abundance';

% 'attack' = disturbance applied to pred. attack rate parameter

% 'mort' = disturbance applied to pred. mort. parameter

% 'abundance' = disturbance applied to knocking down populations

mu = 0.1:0.1:1;

% if noncyclic2/cyclic2/noncyclic3 -> 'attack': 0:0.5:5, 0 to 100% of baseline parameter value

% if cyclic3 -> 'attack': 0:0.09:0.9

% if noncyclic2/cyclic2/noncyclic3 -> 'mort': 0:0.01:0.1, 0 to 100% of baseline parameter value

% if cyclic3 -> 'mort': 0:0.06:0.6

% 'abundance': 0.1:0.1:1, 90% knockdown to 0%

sd = 0;

% sd = 0, OR sd = mu -> turn off sd and set sd input to mu(m)

thr = 0.05;

% 5% of equilib. pop. value

% 10% (relevant fisheries threshold) too high for some dynamics

% 1% too low for some dynamics

for m = 1:length(mu)

for f = 1:length(freqmax)

[ExtinctProb(m,f,1:2)] = PredPreyDisturbanceModel(dyn,freqmax(f),parm,mu(m),sd,thr,0);

end

end

% Save output:

csvwrite('.csv', ExtinctProb)

% Append ExtinctProb to file:

%dlmwrite('filename.csv',ExtinctProb,'-append');

*Predator-Prey Disturbance Model Figures Script:*

%% PredPreyDisturbanceModel Figures Script %%

% column = frequency, row = magnitude

%Prey = ExtinctProb(:,:,1);

%Predator = ExtinctProb(:,:,2);

ExtinctProb = csvread('.csv');

Prey = ExtinctProb(:,1:11);

Predator = ExtinctProb(:,12:end);

figure;

contourf(Prey);

colormap(gray); % make gray scale

title(' - Prey');

c = colorbar('northoutside');

%set(c, 'YTick', 0:0.2:1,'YTickLabel', 0:0.2:1);

LabelText = 'Probability of Prey Extinction';

ylabel(c, LabelText);

xlabel('Disturbance Frequency');

ylabel('Disturbance Magnitude');

set(gca,'XTick',1:1:11,'XTickLabel',0:0.1:1);

set(gca,'YTick',1:1:11,'YTickLabel',0:0.1:1);

% YTick: attack & mort -> 1:1:11, abundance -> 1:1:10

% YTickLabel: attack & mort -> 0:0.1:1, abundance -> 0:0.1:0.9

set(gca,'tickdir','out','ticklength',[0.02 0.02]) % this will make the

%ticks point outwards, can adjust their length with the

%values in that vector

axis square

brighten(0.3); % The colors brighten when value is between 0 and 1,

%and they darken when value is between -1 and 0

%print -depsc2 \_Prey.eps;

%print -djpeg \_Prey.jpg;

%figure;

%contourf(Predator);

%title('Cyclic - Predator');

%c2 = colorbar('northoutside');

%set(c2, 'YTick', 0:0.2:1,'YTickLabel', 0:0.2:1);

%LabelText2 = 'Probability of Predator Extinction';

%ylabel(c2, LabelText2);

%xlabel('Disturbance Frequency');

%ylabel('Disturbance Magnitude');

%set(gca,'XTick',1:1:11,'XTickLabel',0:0.1:1);

%set(gca,'YTick',1:1:11,'YTickLabel',0:0.1:1);

% YTick: attack & mort -> 1:1:11, abundance -> 1:1:10

% YTickLabel: attack & mort -> 0:0.1:1, abundance -> 0:0.1:0.9

%axis square

%print -depsc2 filename2.eps;

%print -djpeg TypeIIINon-CyclicAbundanceSD0\_thr5pct\_Take2\_pred.jpg;

*Predator-Prey Disturbance Model Global Sensitivity Analysis:*

function [ExtinctProb] = PredPreyDisturbanceModel\_GSA(c,r,d,s,e,K,dyn,freqmax,parm,mu,sd,thr,fig)

%% Rosenzweig-MacArthur (1963) predator-prey model with varying disturbance frequency and magnitude %%

% time sequence:

tf = 500;

tspan = 0:1:500;

for i = 1:2500

switch dyn % pred-prey dynamics

case 'noncyclic2' % non-cyclic coexistence (Type II)

% parameters (Params 1-6 specified in function PreyPreyTypeII.m):

Params = [c; r; d; s; e; K];

% initial prey and pred. pop. sizes (respectively)

% (coexistence steady state equilibrium (mean) pop. values):

x\_temp = [0.2502 0.0164];

case 'cyclic2' % cyclic coexistence (Type II)

% parameters:

Params = [c; r; d; s; e; K];

% initial prey and pred. pop. sizes

% (stable limit cycles equilibrium pop. values):

x\_temp = [0.3214 0.0224];

case 'noncyclic3' % non-cyclic coexistence (Type III)

% parameters (Params 1-6 specified in function PreyPreyTypeIII.m):

Params = [c; r; d; s; e; K];

% initial prey and pred. pop. sizes (respectively)

% (coexistence steady state equilibrium pop. values):

x\_temp = [0.5049 0.0448];

case 'cyclic3' % cyclic coexistence (Type III)

% parameters:

Params = [c; r; d; s; e; K];

% initial prey and pred. pop. sizes

% (stable limit cycles equilibrium pop. values):

x\_temp = [3.5672 0.3419];

end

% disturbance frequency:

freqn = 500;

freq = rand(1,freqn) <= freqmax;

freq(end) = false;

% create empty vectors to hold the ODE results

T = [];

X = [];

if freq == 0

[t\_temp, x\_temp] = ode45(@(t,x) PredPreyTypeII(t,x,Params),tspan,x\_temp);

% Type II differential equations specified in file PredPreyTypeII.m

% PredPreyTypeII.m = Rosenzweig-MacArthur predator-prey model (Type II functional response)

% Type III differential equations specified in file PredPreyTypeIII.m

% PredPreyTypeIII.m = Rosenzweig-MacArthur predator-prey model (Type III functional response)

T = [T; t\_temp]; % add in the new results to the growing vector of results

X = [X; x\_temp];

else

starts = find(freq);

[t\_temp, x\_temp] = ode45(@(t,x) PredPreyTypeII(t,x,Params),[0,starts(1)],x\_temp);

T = [T; t\_temp]; % add in the new results to the growing vector of results

X = [X; x\_temp];

for s = 1:length(starts)

switch parm

case 'attack'

% disturbance magnitude (normally dist rand num): (mean, stdev, row, column)

mag = normrnd(mu,sd,1,500);

Param\_tmp = Params;

% apply disturbance: increase pred. attack rate- Params 1 = c

Param\_tmp(1) = Params(1) + mag(starts(s));

% keeps abundances non-negative

x\_temp = max(0,x\_temp);

[t\_temp, x\_temp] = ode45(@(t,x) PredPreyTypeII(t,x,Param\_tmp),[starts(s), starts(s)+1],x\_temp(end,:));

case 'mort'

% disturbance magnitude (normally dist rand num): (mean, stdev, row, column)

mag = normrnd(mu,sd,1,500);

Param\_tmp = Params;

% apply disturbance: increase pred. mortality rate- Params 3 = d

Param\_tmp(3) = Params(3) + mag(starts(s));

% keeps abundances non-negative

x\_temp = max(0,x\_temp);

[t\_temp, x\_temp] = ode45(@(t,x) PredPreyTypeII(t,x,Param\_tmp),[starts(s), starts(s)+1],x\_temp(end,:));

case 'abundance'

% apply disturbance: knock down abundance- N, P

mag = normrnd(mu,sd,1,500);

x\_tmp = x\_temp(end,:) \* mag(starts(s));

% keeps abundances non-negative

x\_tmp = max(0,x\_tmp);

[t\_temp, x\_temp] = ode45(@(t,x) PredPreyTypeII(t,x,Params),[starts(s), starts(s)+1],x\_tmp);

end

T = [T; t\_temp]; % add in the new results to the growing vector of results

X = [X; x\_temp];

if s < length(starts)

if starts(s+1) > starts(s)+1

[t\_temp, x\_temp] = ode45(@(t,x) PredPreyTypeII(t,x,Params),[starts(s)+1, starts(s+1)],x\_temp(end,:));

T = [T; t\_temp]; % add in the new results to the growing vector of results

X = [X; x\_temp];

end

else

[t\_temp, x\_temp] = ode45(@(t,x) PredPreyTypeII(t,x,Params),[starts(s), tf],x\_temp(end,:));

T = [T; t\_temp]; % add in the new results to the growing vector of results

X = [X; x\_temp];

end

end

end

T\_outer{i} = T; % store each simulation

X\_outer{i} = X;

end

% probability of extinction:

if strcmp(dyn, 'noncyclic2')

Extinctions = nan(length(X\_outer),1); % pre-allocate

for i = 1:length(X\_outer)

X = X\_outer{i};

Equilib = 0.2502;

Threshold = thr\*Equilib;

Extinctions(i,1) = any(X(:,1) <= Threshold);

end

elseif strcmp(dyn, 'cyclic2')

Extinctions = nan(length(X\_outer),1);

for i = 1:length(X\_outer)

X = X\_outer{i};

Equilib = 0.3214;

Threshold = thr\*Equilib;

Extinctions(i,1) = any(X(:,1) <= Threshold);

end

elseif strcmp(dyn, 'noncyclic3')

Extinctions = nan(length(X\_outer),1);

for i = 1:length(X\_outer)

X = X\_outer{i};

Equilib = 0.5049;

Threshold = thr\*Equilib;

Extinctions(i,1) = any(X(:,1) <= Threshold);

end

elseif strcmp(dyn, 'cyclic3')

Extinctions = nan(length(X\_outer),1);

for i = 1:length(X\_outer)

X = X\_outer{i};

Equilib = 3.5672;

Threshold = thr\*Equilib;

Extinctions(i,1) = any(X(:,1) <= Threshold);

end

else

end

ExtinctProb = mean(Extinctions);

if fig == 0

else

if fig == 1

f = 1 + (2500-1).\*rand(1,1);

f = round(f);

% Plot one of the pop. time series iterations

% f = iteration number

% Plots prey values in blue with circles and predator values in red with \*'s:

fig = figure;

plot(T\_outer{1,f},X\_outer{1,f}(:,1),'bo:',T\_outer{1,f},X\_outer{1,f}(:,2),'r\*-')

xlabel('Time')

ylabel('Population Size')

title('Predator-Prey Population Time Series')

legend('Prey', 'Predator')

dateformat = 'mm-dd-yy';

date = datestr(now,dateformat);

filename = strcat(date, '\_Pop Time Series\_', num2str(f), '.jpg');

print(fig, '-djpeg', filename);

% Plot one of the phase portraits of the iterations

% Plots the phase portrait of prey values versus predator values:

fig2 = figure;

plot(X\_outer{1,f}(:,1),X\_outer{1,f}(:,2))

xlabel('Prey')

ylabel('Predator')

title('Predator-Prey Phase Portrait')

date2 = datestr(now,dateformat);

filename2 = strcat(date2, '\_Phase Portrait\_', num2str(f), '.jpg');

print(fig2, '-djpeg', filename2);

end

end

end

*Predator-Prey Disturbance Model Global Sensitivity Analysis Script:*

%% PredPreyDisturbanceModel\_GSA\_Script %%

% Inputs:

% a and b = 10% below and 10% above baseline param value, respectively

% Param = (b-a)\*rand(1,1000) + a

% [c=5; r=0.1; d=0.1; s=1; e=0.1; K=0.7] % Type II non-cyclic param values

cs = (5.5-4.5)\*rand(1,1000) + 4.5; % pred. attack param

rs = (0.11-0.09)\*rand(1,1000) + 0.09; % prey growth param

ds = (0.11-0.09)\*rand(1,1000) + 0.09; % pred. mort. param

ss = (1.1-0.9)\*rand(1,1000) + 0.9; % satiation constant param

es = (0.11-0.09)\*rand(1,1000) + 0.09; % pred. biomass conversion efficiency param

Ks = (0.77-0.63)\*rand(1,1000) + 0.63; % prey carrying capacity param

dyn = 'noncyclic2';

% 'noncyclic2' = non-cyclic coexistence (Type II)

freqmax = 0.1;

% disturbance frequency: Low = (0.1), Med. = (0.5), High = (1.0)

parm = 'attack';

% 'attack' = disturbance applied to pred. attack rate parameter

mu = 0.5;

% L = 10%, M = 50%, H = 100% of baseline parameter value

% if noncyclic2: 'attack' -> mu = [0.5 2.5 5]

sd = 0;

% dist. mag. sd = 0

thr = 0.05;

% 5% of equilib. pop. value

for j = 1:1000

[ExtinctProb(j)] = PredPreyDisturbanceModel\_GSA(cs(j),...

rs(j),ds(j),ss(j),es(j),Ks(j),dyn,freqmax,parm,mu,sd,thr,0);

end

X = [cs(:),rs(:),ds(:),ss(:),es(:),Ks(:)];

ExtinctProb = ExtinctProb(:); % convert to 1 column

Data = [ExtinctProb, X]; % merge data

% Save output:

dateformat = 'mm-dd-yyyy';

date = datestr(now,dateformat);

fname = strcat('DataforGSA\_',dyn,'\_',parm,'\_mu',num2str(mu),...

'\_freqmax',num2str(freqmax),'\_',date,'.csv');

csvwrite(fname,Data);

% num2str converts a numeric value to a string

% strcat concatenates multiple strings

*TreeBagger Global Sensitivity Analysis Script:*

%% TreeBagger GSA Script %%

% Load the data set and split it into predictor and response arrays:

data = csvread('DataforGSA\_exclusion\_comp1\_mu0.04\_freqmax1\_07-22-2017.csv');

Y = data(:,2);

X = data(:,3:end);

rng(1); % For reproducibility

% For regression, the general rule is to the set leaf size to 5 and

%select one third of the input features for decision splits at random.

%In the following step, verify the optimal leaf size by comparing mean

%squared errors obtained by regression for various leaf sizes. oobError

%computes MSE versus the number of grown trees. You must set OOBPred

%to 'On' to obtain out-of-bag predictions later.

leaf = [5 10 20 50 100];

col = 'rbcmy';

figure

for i=1:length(leaf)

b = TreeBagger(50,X,Y,'Method','R','OOBPred','On',...

'MinLeafSize',leaf(i));

plot(oobError(b),col(i))

hold on

end

xlabel 'Number of Grown Trees'

ylabel 'Mean Squared Error'

legend({'5' '10' '20' '50' '100'},'Location','NorthEast')

hold off

%print

% grow a larger ensemble with 100 trees and use it to estimate feature importance.

b = TreeBagger(100,X,Y,'Method','R','OOBVarImp','On','MinLeafSize',20);

% Inspect the error curve again to make sure nothing went wrong during training.

figure

plot(oobError(b))

xlabel 'Number of Grown Trees'

ylabel 'Out-of-Bag Mean Squared Error'

%print

% Prediction ability should depend more on important features than

%unimportant features. You can use this idea to measure feature importance.

% For each feature, permute the values of this feature across every

%observation in the data set and measure how much worse the MSE becomes

%after the permutation. You can repeat this for each feature.

%Using the following code, plot the increase in MSE due to permuting

%out-of-bag observations across each input variable.

%The OOBPermutedVarDeltaError array stores the increase in MSE averaged

%over all trees in the ensemble and divided by the standard deviation

%taken over the trees, for each variable.

fig = figure;

bar(b.OOBPermutedVarDeltaError)

xlabel 'Feature Number'

ylabel 'Out-of-Bag Feature Importance'

% Save parameter importance bar graph:

dateformat = 'mm-dd-yyyy';

date = datestr(now,dateformat);

filename = strcat(date, '\_Parameter Importance Bar Graph\_', '.jpg');

print(fig, '-djpeg', filename);

% Save parameter importance values to .txt file:

ParamImportance = b.OOBPermutedVarDeltaError;

dlmwrite('.txt',ParamImportance,',');

%The larger this value, the more important the variable. Impose an

%arbitrary cutoff (e.g., 0.7 or 1) to select the most important features.

idxvar = find(b.OOBPermutedVarDeltaError>0.7)

%The OOBIndices property of TreeBagger tracks which observations are out

%of bag for what trees. Using this property, you can monitor the fraction

%of observations in the training data that are in bag for all trees. The

%curve starts at approximately 2/3, which is the fraction of unique

%observations selected by one bootstrap replica, and goes down to 0 at

%approximately 10 trees.

finbag = zeros(1,b.NTrees);

for t=1:b.NTrees

finbag(t) = sum(all(~b.OOBIndices(:,1:t),2));

end

finbag = finbag / size(X,1);

figure

plot(finbag)

xlabel 'Number of Grown Trees'

ylabel 'Fraction of In-Bag Observations'

%Using just the most powerful features, determine if it is possible

%to obtain a similar predictive power. To begin, grow 100 trees on these

%features only.

biv = TreeBagger(100,X(:,idxvar),Y,'Method','R',...

'OOBVarImp','On','MinLeafSize',20);

figure

plot(oobError(biv))

xlabel 'Number of Grown Trees'

ylabel 'Out-of-Bag Mean Squared Error'

%print

figure

bar(biv.OOBPermutedVarDeltaError)

xlabel 'Feature Index'

ylabel 'Out-of-Bag Feature Importance'

%print

%These most powerful features give the same MSE as the full set,

%and the ensemble trained on the reduced set ranks these features

%similarly to each other. If you remove features 1 and 2 from the

%reduced set, then the predictive power of the algorithm might not

%decrease significantly.

%To find outliers in the training data, compute the proximity matrix

%using fillProximities.

biv = fillProximities(biv);

%The method normalizes this measure by subtracting the mean outlier

%measure for the entire sample. Then it takes the magnitude of this

%difference and divides the result by the median absolute deviation

%for the entire sample.

figure

histogram(biv.OutlierMeasure)

xlabel 'Outlier Measure'

ylabel 'Number of Observations'

%By applying multidimensional scaling to the computed matrix of

%proximities, you can inspect the structure of the input data and

%look for possible clusters of observations. The mdsProx method returns

%scaled coordinates and eigenvalues for the computed proximity matrix.

%If you run it with the Colors name-value-pair argument, then this method

%creates a scatter plot of two scaled coordinates.

figure(8)

[~,e] = mdsProx(biv,'Colors','K');

xlabel 'First Scaled Coordinate'

ylabel 'Second Scaled Coordinate'

%Assess the relative importance of the scaled axes by plotting the

%first 20 eigenvalues.

figure

bar(e(1:20))

xlabel 'Scaled Coordinate Index'

ylabel 'Eigenvalue'

*Elasticity Analysis (in R):*

## Calculating Elasticities ##

# Read in data [Dissertation Ch. 1]

Data = read.csv("TypeIICyclicMortSD0\_thr5pct\_2500sims.csv", header= FALSE)

# header=FALSE because first row does not contain variable names

# row -> magnitude, column -> frequency

Data\_prey = Data[ ,c(1:11)] # prey results

#Data\_pred = Data[ ,c(12:22)] # predator results

Mag = c(0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1)

#Mag = c(0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9) # if dist. applied to abundance

Freq = c(0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1)

#Formulas for calculating elasticities for mag. and freq:

# from row i to row i-1, and column j to column j-1 --> "DOWNHILL"

# Freq: ((E(i,j-1)-E(i,j))/E(i,j))/((Freq(j-1)-Freq(j))/Freq(j))

# Mag: ((E(i-1,j)-E(i,j))/E(i,j))/((Mag(i-1)-Mag(i))/Mag(i))

# from row i-1 to row i, and column j-1 to column j --> "UPHILL"

# Freq: ((E(i,j)-E(i,j-1))/E(i,j-1))/((Freq(j)-Freq(j-1))/Freq(j-1))

# Mag: ((E(i,j)-E(i-1,j))/E(i-1,j))/((Mag(i)-Mag(i-1))/Mag(i-1))

# pre-allocate

eF = matrix(NA, nrow = 11, ncol = 11) #nrow=10 if dist. applied to abundance

# for loop to calculate elasticities

for (i in 1:11) # i in 1:10 if dist. applied to abundance

for (j in 2:11)

{

eF[i,j] = ((Data\_prey[i,j-1]-Data\_prey[i,j])/Data\_prey[i,j]) /

((Freq[j-1]-Freq[j])/Freq[j]) # DOWNHILL

#eF[i,j] = ((Data\_prey[i,j]-Data\_prey[i,j-1])/Data\_prey[i,j-1]) /

#((Freq[j]-Freq[j-1])/Freq[j-1]) # UPHILL

}

eF[is.nan(eF)] = 0 #replaces NaN's (resulting from 0/0) with 0's

eF[eF < 0] = 0 # replace negative values with 0

eF = abs(eF) # this is here because I kept getting negative thetas

# make .csv file for eF (match prefix (before \_eF) to file above)

#write.csv(eF, file = "\_eF.csv")

# pre-allocate

eM = matrix(NA, nrow = 11, ncol = 11) #nrow=10 if dist. applied to abundance

# for loop to calculate elasticities

for (i in 2:11) # i in 2:10 if dist. applied to abundance

for (j in 1:11)

{

eM[i,j] = ((Data\_prey[i-1,j]-Data\_prey[i,j])/Data\_prey[i,j]) /

((Mag[i-1]-Mag[i])/Mag[i]) # DOWNHILL

#eM[i,j] = ((Data\_prey[i,j]-Data\_prey[i-1,j])/Data\_prey[i-1,j]) /

#((Mag[i]-Mag[i-1])/Mag[i-1]) # UPHILL

}

eM[is.nan(eM)] = 0 #replaces NaN's (resulting from Div/0) with 0's

eM[eM < 0] = 0 # replace negative values with 0

eM = abs(eM) # this is here because I kept getting negative thetas

# make .csv file for eM (match prefix (before \_eM) to file above)

#write.csv(eM, file = "\_eM.csv")

#plot

# make .png file for plot (match prefix (before .png) to file above)

#png(filename=".png")

plot(eF,eM,xlim=c(0,8),ylim=c(0,8))

abline(0,1) # slope of 1 indicates theta = 45 degrees

#dev.off()

# Convert Cartesian coordinates to polar:

# eF is x and eM is y; convert to get (r,theta)

# Use Pythagorean theorem to get r

# and solve for theta using the inverse tangent function -> theta = tan^-1(y/x)

# calculate Rs and Thetas

# pre-allocate

Rs = matrix(NA, nrow = 11, ncol = 11) #nrow=10 if dist. applied to abundance

Thetas\_rad = matrix(NA, nrow = 11, ncol = 11) #nrow=10 if dist. applied to abundance

Thetas = matrix(NA, nrow = 11, ncol = 11) #nrow=10 if dist. applied to abundance

# Use Radians to Degrees function (atan function gives radians)

rad2deg = function(rad) {

return((180 \* rad) / pi)

}

# for loop

for (i in 1:11) # i in 1:10 if dist. applied to abundance

for (j in 1:11)

{

Rs[i,j] = sqrt(eF[i,j]^2 + eM[i,j]^2)

Thetas\_rad[i,j] = atan(eM[i,j]/eF[i,j])

Thetas[i,j] = rad2deg(Thetas\_rad[i,j])

}

Thetas[is.nan(Thetas)] = 45 #atan(0/0) and atan(Inf/Inf) result in NaN; replaces NaN's with 45 degrees

# Note: atan(0/integer) results in theta of 0 degrees; atan(integer/0) results in theta of 90 degrees

# when y=0, vector points in x direction (0 degrees); when x=0, vector points in y direction (90 degrees)

# The angle (relative to horizontal) theta of that combined vector

# will tell you the relative elasticity (is it > 45º or < 45º?)

# make .csv file for Rs (match prefix (before \_Rs) to file above)

#write.csv(Rs, file = "\_Rs.csv")

# make .csv file for Thetas (match prefix (before \_Thetas) to file above)

write.csv(Thetas, file = "TypeIICyclicMortSD0\_thr5pct\_2500sims\_Thetas.csv")

*Stability Analysis for Competition Models:*

%% Stability Analysis Plot (L-V Comp) %%

syms ro N1 Ko a N2 rt Kt B

% The model equations:

dN1 = ro\*N1\*((Ko-N1-(a\*N2))/Ko);

dN2 = rt\*N2\*((Kt-N2-(B\*N1))/Kt);

% Find the equilibrium (set both equations to zero and solve for N1,N2)

Eq = solve( ro\*N1\*((Ko-N1-(a\*N2))/Ko) == 0,...

rt\*N2\*((Kt-N2-(B\*N1))/Kt) == 0,...

N1,N2);

% Structure Eq holds the results in fields N1, N2.

% There are four roots. We want the one where both are > 0 (the 4th one).

% Jacobian. For local stability analysis, we require the Jacobian to be

% calculated with respect to the state variables (so it represents rate of

% return to equilibrium due to minor perturbations of state vars)

J = jacobian([dN1; dN2], [N1 N2]);

% Jacobian must be evaluated at the equilibrium. Substitute in the

% equilibrium solution:

J = subs(J,[N1,N2],[Eq.N1(4),Eq.N2(4)]);

% Now evaluate the Jacobian over a range of the values you are interested

% in:

% a vs. B:

%as = 0.08:0.08:0.8;

%Bs = 0.12:0.12:1.2;

as = 0.1:0.1:1;

Bs = 0.15:0.15:1.5;

%as = 0.25:0.25:2.5;

%Bs = 0.25:0.25:2.5;

% Other parameters are just fixed at a constant value: (again, just making

% up numbers for now)

ro = 0.1;

Ko = 2;

rt = 0.2;

Kt = 1;

% Loop over the two parameters of interest and calculate the eigenvalues:

Eigs = nan(length(as),length(Bs)); % pre-allocate the variable

for i = 1:length(as)

for j = 1:length(Bs)

try

J\_num = subs(J, [a, B, ro, Ko, rt, Kt], [as(i),Bs(j),ro,Ko,rt,Kt]); % substitute in the values for each parameter

Eigs(i,j) = max(real(eig(eval(J\_num)))); % calculate the maximum (dominant)

%eigenvalue (have to use eval to make it actually calculate the value

%rather than report the symbolic expression)

end

end

end

% Make a pretty figure

figure(1)

clf

hold on

pcolor(as,Bs,Eigs) % plot the eigenvalues

shading interp

contour(as,Bs,Eigs,[-1,0,1])%,'linecolor',[0.85 0.85 0.85]) % plot contour lines to denote critical regions

colormap(gray); % make gray scale

set(gca,'tickdir','out','ticklength',[0.015 0.015],'fontsize',12)

xlabel('Comp. Coeff. 1-2 (a12)','fontsize',14)

ylabel('Comp. Coeff. 2-1 (a21)','fontsize',14)

ch=colorbar;

lb = ylabel(ch, 'Eigenvalue ({\lambda})','fontsize',14);

set(ch,'tickdir','out','ticklength',0.015)

axis square

%axis equal

brighten(0.3); % The colors brighten when value is between 0 and 1,

%and they darken when value is between -1 and 0

% approx. 0 eig --> non-zero coexist

% next region --> approaching exclusion

% next region (highest eigs) --> exclusion

% blank/white regions --> exlcusion, something else (e.g., faster approach to exclusion)?

print -dpng NumericalStabilityAnalysisPlot\_Comp\_avsB2.png;

*Competition Model Equations:*

function LVCompetition = LVComp(t,x,Params)

% Lotka-Volterra Competition Model (2 species):

% dN1/dt = r1 \* N1 \* ((K1 - N1 - (a\*N2))/K1)) % change in species 1 population over time

% dN2/dt = r2 \* N2 \* ((K2 - N2 - (B\*N1))/K2)) % change in species 2 population over time

% where r = intrinsic rate of increase, a = per capita effect of species 2

% on the population growth of species 1, B = effect of species 1 on the

% growth of species 2, and K = carrying capacity.

% Starting parameter values:

ro = Params(1);

Ko = Params(2);

a = Params(3);

rt = Params(4);

Kt = Params(5);

B = Params(6);

% Generates a vector of zeros the same size as the vector x. The 0's are

% placeholders for dN1/dt and dN2/dt values:

LVCompetition = zeros(size(x));

% Input vector, x: x(1) = N1 and x(2) = N2

% Calculates dN1/dt, the first value in the LVPredPrey vector:

LVCompetition(1) = ro \* x(1) \* ((Ko - x(1) - (a\*x(2)))/Ko);

% Calculates dN2/dt, the second value in the LVPredPrey vector:

LVCompetition(2) = rt \* x(2) \* ((Kt - x(2) - (B\*x(1)))/Kt);

end

*Competition Disturbance Model:*

function [ExtinctProb] = CompDisturbanceModel(dyn,freqmax,parm,mu,sd,thr,fig)

%% Competition model (2 species) with varying disturbance frequency and magnitude %%

% time sequence:

tf = 500;

tspan = 0:1:500;

for i = 1:2500

switch dyn % competition dynamics

case 'coexist' % stable coexistence

% parameters (Params 1-6 specified in function LVComp.m):

Params = [0.1; 2; 0.4; 0.2; 1; 0.2];

% initial Species 1 and Species 2 pop. sizes (respectively)

% (stable coexistence steady state equilibrium pop. values):

x\_temp = [1.69 0.66];

case 'exclusion' % competitive exclusion

% parameters:

Params = [0.1; 2; 0.4; 0.2; 1; 0.6];

% initial Species 1 and Species 2 pop. sizes

% (competitive exclusion equilibrium pop. values):

x\_temp = [1.93 0.04];

end

% disturbance frequency:

freqn = 500;

freq = rand(1,freqn) <= freqmax;

freq(end) = false;

% create empty vectors to hold the ODE results

T = [];

X = [];

if freq == 0

[t\_temp, x\_temp] = ode45(@(t,x) LVComp(t,x,Params),tspan,x\_temp);

% differential equations specified in file LVComp.m

% LVComp.m = Lotka-Volterra Competition Model (2 species)

T = [T; t\_temp]; % add in the new results to the growing vector of results

X = [X; x\_temp];

else

starts = find(freq);

[t\_temp, x\_temp] = ode45(@(t,x) LVComp(t,x,Params),[0,starts(1)],x\_temp);

T = [T; t\_temp]; % add in the new results to the growing vector of results

X = [X; x\_temp];

for s = 1:length(starts)

switch parm

case 'comp1'

% disturbance magnitude (normally dist rand num): (mean, stdev, row, column)

mag = normrnd(mu,sd,1,500);

Param\_tmp = Params;

% apply disturbance to 1 comp. coeff. parameter (a - Params(3))

Param\_tmp(3) = Params(3) + mag(starts(s));

% keeps abundances non-negative

x\_temp = max(0,x\_temp);

[t\_temp, x\_temp] = ode45(@(t,x) LVComp(t,x,Param\_tmp),[starts(s), starts(s)+1],x\_temp(end,:));

case 'comp2'

% disturbance magnitude (normally dist rand num): (mean, stdev, row, column)

mag1 = normrnd(mu,sd,1,500);

mag2 = normrnd(mu2,sd2,1,500);

Param\_tmp = Params;

% apply disturbance to both comp. coeff. parameters (a - Params(3); B - Params(6))

Param\_tmp(3) = Params(3) + mag1(starts(s));

Param\_tmp(6) = Params(6) + mag2(starts(s));

% keeps abundances non-negative

x\_temp = max(0,x\_temp);

[t\_temp, x\_temp] = ode45(@(t,x) LVComp(t,x,Param\_tmp),[starts(s), starts(s)+1],x\_temp(end,:));

case 'compB'

% disturbance magnitude (normally dist rand num): (mean, stdev, row, column)

mag = normrnd(mu,sd,1,500);

Param\_tmp = Params;

% apply disturbance to 1 comp. coeff. parameter (B - Params(6))

Param\_tmp(6) = Params(6) + mag(starts(s));

% keeps abundances non-negative

x\_temp = max(0,x\_temp);

[t\_temp, x\_temp] = ode45(@(t,x) LVComp(t,x,Param\_tmp),[starts(s), starts(s)+1],x\_temp(end,:));

case 'abundance'

% apply disturbance: knock down abundance- N1, N2

mag = normrnd(mu,sd,1,500);

x\_tmp = x\_temp(end,:) \* mag(starts(s));

% keeps abundances non-negative

x\_tmp = max(0,x\_tmp);

[t\_temp, x\_temp] = ode45(@(t,x) LVComp(t,x,Params),[starts(s), starts(s)+1],x\_tmp);

end

T = [T; t\_temp]; % add in the new results to the growing vector of results

X = [X; x\_temp];

if s < length(starts)

if starts(s+1) > starts(s)+1

[t\_temp, x\_temp] = ode45(@(t,x) LVComp(t,x,Params),[starts(s)+1, starts(s+1)],x\_temp(end,:));

T = [T; t\_temp]; % add in the new results to the growing vector of results

X = [X; x\_temp];

end

else

[t\_temp, x\_temp] = ode45(@(t,x) LVComp(t,x,Params),[starts(s), tf],x\_temp(end,:));

T = [T; t\_temp]; % add in the new results to the growing vector of results

X = [X; x\_temp];

end

end

end

T\_outer{i} = T; % store each simulation

X\_outer{i} = X;

end

% probability of extinction:

if strcmp(dyn, 'coexist')

Extinctions = nan(length(X\_outer),2); % pre-allocate

for i = 1:length(X\_outer)

X = X\_outer{i};

Equilib = [1.69 0.66];

Threshold = thr\*Equilib;

Extinctions(i,1) = any(X(:,1) <= Threshold(1,1));

Extinctions(i,2) = any(X(:,2) <= Threshold(1,2));

end

elseif strcmp(dyn, 'exclusion')

Extinctions = nan(length(X\_outer),2);

for i = 1:length(X\_outer)

X = X\_outer{i};

Equilib = [1.93 0.04];

Threshold = thr\*Equilib;

Extinctions(i,1) = any(X(:,1) <= Threshold(1,1));

Extinctions(i,2) = any(X(:,2) <= Threshold(1,2));

end

else

end

ExtinctProb = mean(Extinctions);

if fig == 0

else

if fig == 1

f = 1 + (2500-1).\*rand(1,1);

f = round(f);

% Plot one of the pop. time series iterations

% f = iteration number

% Plots Species 1 values in blue with circles and Species 2 values in red with \*'s:

fig = figure;

plot(T\_outer{1,f},X\_outer{1,f}(:,1),'b',T\_outer{1,f},X\_outer{1,f}(:,2),'r')

xlabel('Time')

ylabel('Population Size')

title('Competition (2 Species) Population Time Series')

legend('Species 1', 'Species 2')

dateformat = 'mm-dd-yy';

date = datestr(now,dateformat);

filename = strcat(date, '\_Pop Time Series\_', num2str(f), '.eps');

set(gca,'tickdir','out','ticklength',[0.02 0.02])

print(fig, '-depsc2', filename);

% Plot one of the phase portraits of the iterations

% Plots the phase portrait of Species 1 values versus Species 2 values:

fig2 = figure;

plot(X\_outer{1,f}(:,1),X\_outer{1,f}(:,2))

xlabel('Species 1')

ylabel('Species 2')

title('Competition Phase Portrait')

date2 = datestr(now,dateformat);

filename2 = strcat(date2, '\_Phase Portrait\_', num2str(f), '.eps');

print(fig2, '-depsc2', filename2);

end

end

end

*Competition Disturbance Model Script:*

%% CompDisturbanceModel Script %%

% Inputs:

dyn = 'coexist';

% 'coexist' = stable coexistence (1 species at higher equilibrium/K (Species 1) --> r vs. K)

% 'exclusion' = competitive exclusion (1 winner (Species 1) --> much larger comp coeff)

freqmax = 0.4; %0:0.1:1;

% disturbance frequency, 0 to 100%

parm = 'abundance';

% 'comp1' = disturbance applied to 1 comp. coeff. parameter (a)

% 'comp2' = disturbance applied to 2/both comp. coeff. parameters (a & B)

% 'compB' = disturbance applied to 1 comp. coeff. parameter (B)

% 'abundance' = disturbance applied to knocking down populations

mu = 0.9; %0:0.04:0.4;

% if coexist/exclusion -> 'comp1' mu = 0:0.04:0.4, 0 to 100% of baseline parameter value

% if coexist -> 'comp2' mu(a) 0:0.04:0.4, mu(B) = 0:0.02:0.2, 0 to 100% of baseline parameter value

% if exclusion -> 'comp2' mu(a) = 0:0.04:0.4, mu(B) = 0:0.06:0.6

% 'abundance': 0.1:0.1:1, 90% knockdown to 0%

%mu2 = 0:0.02:0.2;

sd = 0;

% sd = 0, OR sd = mu -> turn off sd and set sd input to mu(m)

%sd2 = 0;

thr = 0.05;

% 10% (0.1) is relevant fisheries threshold [too high for some dynamics]

% 5% seems just right for most circumstances

% 1% seems too low for most

%for m = 1:length(mu)

%for f = 1:length(freqmax)

%[ExtinctProb(m,f,1:2)] = CompDisturbanceModel(dyn,freqmax(f),parm,mu(m),sd,thr,0);

[ExtinctProb] = CompDisturbanceModel(dyn,freqmax,parm,mu,sd,thr,1);

%end

%end

% Save output:

%csvwrite('.csv', ExtinctProb)

% csvwrite('filename.csv', ExtinctProb)

% e.g., 'Coexist\_Comp1\_sd0\_thr10pct.csv'

*Competition Disturbance Model Figures Script:*

%% CompDisturbanceModel Figures Script %%

% row = magnitude, column = frequency

ExtinctProb = csvread('CoexistComp1SD0\_thr5pct.csv');

%Species1 = ExtinctProb(:,:,1);

%Species2 = ExtinctProb(:,:,2);

Species1 = ExtinctProb(:,1:11);

Species2 = ExtinctProb(:,12:end);

figure;

contourf(Species1);

title('CoexistComp1SD0\_thr5pct - Species 1');

c = colorbar('northoutside');

LabelText = 'Probability of Extinction';

ylabel(c, LabelText);

xlabel('Disturbance Frequency');

ylabel('Disturbance Magnitude');

set(gca,'XTick',1:1:11,'XTickLabel',0:0.1:1);

set(gca,'YTick',1:1:11,'YTickLabel',0:0.1:1);

% YTick: parameter -> 1:1:11, abundance -> 1:1:10

% YTickLabel: parameter -> 0:0.1:1, abundance -> 0:0.1:0.9

axis square

%print -depsc2 filename.eps;

print -djpeg CoexistComp1SD0\_thr5pct\_Species1.jpg;

figure;

contourf(Species2);

title('CoexistComp1SD0\_thr5pct - Species 2');

c2 = colorbar('northoutside');

LabelText2 = 'Probability of Extinction';

ylabel(c2, LabelText2);

xlabel('Disturbance Frequency');

ylabel('Disturbance Magnitude');

set(gca,'XTick',1:1:11,'XTickLabel',0:0.1:1);

set(gca,'YTick',1:1:11,'YTickLabel',0:0.1:1);

% YTick: parameter -> 1:1:11, abundance -> 1:1:10

% YTickLabel: parameter -> 0:0.1:1, abundance -> 0:0.1:0.9

axis square

%print -depsc2 filename2.eps;

print -djpeg CoexistComp1SD0\_thr5pct\_Species2.jpg;

*Competition Disturbance Model Global Sensitivity Analysis:*

function [ExtinctProb] = CompDisturbanceModel\_GSA(ro,Ko,a,rt,Kt,B,dyn,freqmax,parm,mu,sd,thr,fig)

%% Competition model (2 species) with varying disturbance frequency and magnitude %%

% time sequence:

tf = 500;

tspan = 0:1:500;

for i = 1:2500

switch dyn % competition dynamics

case 'coexist' % stable coexistence

% parameters (Params 1-6 specified in function LVComp.m):

Params = [ro; Ko; a; rt; Kt; B];

% initial Species 1 and Species 2 pop. sizes (respectively)

% (stable coexistence steady state equilibrium pop. values):

x\_temp = [1.69 0.66];

case 'exclusion' % competitive exclusion

% parameters:

Params = [ro; Ko; a; rt; Kt; B];

% initial Species 1 and Species 2 pop. sizes

% (competitive exclusion equilibrium pop. values):

x\_temp = [1.93 0.04];

end

% disturbance frequency:

freqn = 500;

freq = rand(1,freqn) <= freqmax;

freq(end) = false;

% create empty vectors to hold the ODE results

T = [];

X = [];

if freq == 0

[t\_temp, x\_temp] = ode45(@(t,x) LVComp(t,x,Params),tspan,x\_temp);

% differential equations specified in file LVComp.m

% LVComp.m = Lotka-Volterra Competition Model (2 species)

T = [T; t\_temp]; % add in the new results to the growing vector of results

X = [X; x\_temp];

else

starts = find(freq);

[t\_temp, x\_temp] = ode45(@(t,x) LVComp(t,x,Params),[0,starts(1)],x\_temp);

T = [T; t\_temp]; % add in the new results to the growing vector of results

X = [X; x\_temp];

for s = 1:length(starts)

switch parm

case 'comp1'

% disturbance magnitude (normally dist rand num): (mean, stdev, row, column)

mag = normrnd(mu,sd,1,500);

Param\_tmp = Params;

% apply disturbance to 1 comp. coeff. parameter (a - Params(3))

Param\_tmp(3) = Params(3) + mag(starts(s));

% keeps abundances non-negative

x\_temp = max(0,x\_temp);

[t\_temp, x\_temp] = ode45(@(t,x) LVComp(t,x,Param\_tmp),[starts(s), starts(s)+1],x\_temp(end,:));

case 'comp2'

% disturbance magnitude (normally dist rand num): (mean, stdev, row, column)

mag1 = normrnd(mu,sd,1,500);

mag2 = normrnd(mu2,sd2,1,500);

Param\_tmp = Params;

% apply disturbance to both comp. coeff. parameters (a - Params(3); B - Params(6))

Param\_tmp(3) = Params(3) + mag1(starts(s));

Param\_tmp(6) = Params(6) + mag2(starts(s));

% keeps abundances non-negative

x\_temp = max(0,x\_temp);

[t\_temp, x\_temp] = ode45(@(t,x) LVComp(t,x,Param\_tmp),[starts(s), starts(s)+1],x\_temp(end,:));

case 'compB'

% disturbance magnitude (normally dist rand num): (mean, stdev, row, column)

mag = normrnd(mu,sd,1,500);

Param\_tmp = Params;

% apply disturbance to 1 comp. coeff. parameter (B - Params(6))

Param\_tmp(6) = Params(6) + mag(starts(s));

% keeps abundances non-negative

x\_temp = max(0,x\_temp);

[t\_temp, x\_temp] = ode45(@(t,x) LVComp(t,x,Param\_tmp),[starts(s), starts(s)+1],x\_temp(end,:));

case 'abundance'

% apply disturbance: knock down abundance- N1, N2

mag = normrnd(mu,sd,1,500);

x\_tmp = x\_temp(end,:) \* mag(starts(s));

% keeps abundances non-negative

x\_tmp = max(0,x\_tmp);

[t\_temp, x\_temp] = ode45(@(t,x) LVComp(t,x,Params),[starts(s), starts(s)+1],x\_tmp);

end

T = [T; t\_temp]; % add in the new results to the growing vector of results

X = [X; x\_temp];

if s < length(starts)

if starts(s+1) > starts(s)+1

[t\_temp, x\_temp] = ode45(@(t,x) LVComp(t,x,Params),[starts(s)+1, starts(s+1)],x\_temp(end,:));

T = [T; t\_temp]; % add in the new results to the growing vector of results

X = [X; x\_temp];

end

else

[t\_temp, x\_temp] = ode45(@(t,x) LVComp(t,x,Params),[starts(s), tf],x\_temp(end,:));

T = [T; t\_temp]; % add in the new results to the growing vector of results

X = [X; x\_temp];

end

end

end

T\_outer{i} = T; % store each simulation

X\_outer{i} = X;

end

% probability of extinction:

if strcmp(dyn, 'coexist')

Extinctions = nan(length(X\_outer),2); % pre-allocate

for i = 1:length(X\_outer)

X = X\_outer{i};

Equilib = [1.69 0.66];

Threshold = thr\*Equilib;

Extinctions(i,1) = any(X(:,1) <= Threshold(1,1));

Extinctions(i,2) = any(X(:,2) <= Threshold(1,2));

end

elseif strcmp(dyn, 'exclusion')

Extinctions = nan(length(X\_outer),2);

for i = 1:length(X\_outer)

X = X\_outer{i};

Equilib = [1.93 0.04];

Threshold = thr\*Equilib;

Extinctions(i,1) = any(X(:,1) <= Threshold(1,1));

Extinctions(i,2) = any(X(:,2) <= Threshold(1,2));

end

else

end

ExtinctProb = mean(Extinctions);

if fig == 0

else

if fig == 1

f = 1 + (2500-1).\*rand(1,1);

f = round(f);

% Plot one of the pop. time series iterations

% f = iteration number

% Plots Species 1 values in blue with circles and Species 2 values in red with \*'s:

fig = figure;

plot(T\_outer{1,f},X\_outer{1,f}(:,1),'bo:',T\_outer{1,f},X\_outer{1,f}(:,2),'r\*-')

xlabel('Time')

ylabel('Population Size')

title('Competition (2 Species) Population Time Series')

legend('Species 1', 'Species 2')

dateformat = 'mm-dd-yy';

date = datestr(now,dateformat);

filename = strcat(date, '\_Pop Time Series\_', num2str(f), '.jpg');

print(fig, '-djpeg', filename);

% Plot one of the phase portraits of the iterations

% Plots the phase portrait of Species 1 values versus Species 2 values:

fig2 = figure;

plot(X\_outer{1,f}(:,1),X\_outer{1,f}(:,2))

xlabel('Species 1')

ylabel('Species 2')

title('Competition Phase Portrait')

date2 = datestr(now,dateformat);

filename2 = strcat(date2, '\_Phase Portrait\_', num2str(f), '.jpg');

print(fig2, '-djpeg', filename2);

end

end

end

*Competition Disturbance Model Global Sensitivity Analysis Script:*

%% CompDisturbanceModel\_GSA\_Script %%

% Inputs:

% a and b = 10% below and 10% above baseline param value, respectively

% Param = (b-a)\*rand(1,1000) + a

% [ro=0.1; Ko=2; a=0.4; rt=0.2; Kt=1; B=0.2] % coexist param values

% [ro=0.1; Ko=2; a=0.4; rt=0.2; Kt=1; B=0.6] % exclusion param values

ro = (0.11-0.09)\*rand(1,1000) + 0.09; % Species 1 intrinsic rate of increase

Ko = (2.2-1.8)\*rand(1,1000) + 1.8; % Species 1 carrying capacity

a = (0.44-0.36)\*rand(1,1000) + 0.36; % per capita effect of species 2 on the population growth of species 1

rt = (0.22-0.18)\*rand(1,1000) + 0.18; % Species 2 intrinsic rate of increase

Kt = (1.1-0.9)\*rand(1,1000) + 0.9; % Species 2 carrying capacity

B = (0.22-0.18)\*rand(1,1000) + 0.18; % effect of species 1 on the growth of species 2

dyn = 'coexist';

% 'coexist': competitive coexistence

% 'exclusion': competitive exclusion

freqmax = 0.1;

% disturbance frequency: Low = (0.1), Med. = (0.5), High = (1.0)

parm = 'comp1';

% 'comp1': dist. applied to 1 comp. coeff. parameter (a)

% 'abundance': dist. applied to both. pop.'s abundances

mu = 0.04;

% L = 10%, M = 50%, H = 100% of baseline parameter value

% if 'coexist', 'comp1' -> mu = [0.04 0.2 0.4]

% if 'exclusion', 'comp1' -> mu = [0.04 0.2 0.4]

% if 'coexist', 'abundance' -> mu = [0.9 0.5 0.1]

% if 'exclusion', 'abundance' -> mu = [0.9 0.5 0.1]

sd = 0;

% dist. mag. sd = 0

thr = 0.1;

% 10% of equilib. pop. value

for j = 1:1000

[ExtinctProb(j,1:2)] = CompDisturbanceModel\_GSA(ro(j),Ko(j),a(j),rt(j),...

Kt(j),B(j),dyn,freqmax,parm,mu,sd,thr,0);

end

X = [ro(:),Ko(:),a(:),rt(:),Kt(:),B(:)];

Data = [ExtinctProb, X]; % merge data

% Save output:

dateformat = 'mm-dd-yyyy';

date = datestr(now,dateformat);

fname = strcat('DataforGSA\_',dyn,'\_',parm,'\_mu',num2str(mu),...

'\_freqmax',num2str(freqmax),'\_',date,'.csv');

csvwrite(fname,Data);

% num2str converts a numeric value to a string

% strcat concatenates multiple strings