**// Ratio-Dependent Functional Response** (based onEquation 1.14 in Arditi & Ginzburg 2012)

// Functional response is the number of prey consumed per predator in this time period.

// alpha: slope of the function at origin; the rate at which the prey population is made available

// to the predator population: the prey death rate due to predation cannot exceed alpha

// htime: handling time; 1/asymptote

Function RDFuncResp(PreyPop,PredPop,alpha,htime):real;

begin

IF (PreyPop=0) or (PredPop=0)

then FuncResp := 0

else FuncResp := (alpha \* PreyPop) / (PredPop + alpha \* htime \* PreyPop);

end;

**/////////////////////////////////// start ///////////////////////////////////////////////**

**// CHECK PARAMETERS**

If any of the following is <0 then it is an error condition:

PreyPop, PredPop, // population sizes at current time step

PreyRmax, PreyKcap, // prey density dependence (Rmax & K in the absence of predator)

PredRmax, PredKcap, // predator density dependence (not quite: Rmax is an optional maximum)

alpha, htime, // functional response parameters

PredRnoprey, EffConst, // for the predator equation (see below)

PreyMatrix, PREDmatrix // stage matrices for prey and predator; eigenvalues must be 1.0

**/////////////////// FOR PREY MODEL**

// PreyRmax (max growth rate) and PreyKcap are assumed to be those when there is no predation

// NOTE: This function assumes that the eigenvalue of the PREY stage matrix is 1.0,

// The following loop checks if the user-specified Rmax is possible (when N is small, EV>=Rmax)

// REMOVE or CHANGE the following loop if eigenvalue of the matrix (EV)=Rmax

for j := 1 to Stages do {columns}

begin

TotalSurv := 0.0; // sum of all survivals from j when N~0 (when R=Rmax)

for i := 1 to Stages do {rows}

TotalSurv := TotalSurv + PreyMatrix^[i]^[j] \* PreyConstraintsMatrix^[i]^[j] \* PreyRmax;

if TotalSurv>1.0 then

begin

userDDErrorCode := 3;

userDDErrorMsg := 'Prey Rmax is too high.';

exit

end;

end; //for j (columns)

if PreyRmax<=1 then

begin

userDDErrorCode := 3;

userDDErrorMsg := 'Prey Rmax is too low.';

exit

end;

if PreyKcap <= 0 then

lMult := 0

else // see equation for "Scramble" in section 8.4 of the User Manual

begin

lMult := (-ln(PreyRmax) / PreyKcap) \* PreyPop;

if Abs(lMult)<exp\_biggest

then lMult := exp(lMult)

else if lMult>0 then lMult := pow\_biggest

else lMult := 1/pow\_biggest;

// At this point lmult~1 if N is small, and lmult=(1/Rmax) if N=K. So:

// REMOVE the following line if eigenvalue of the matrix (EV)=Rmax;

// KEEP it if EV=1. These are the only two options.

lMult := PreyRmax \* lMult;

end;

// Now reduce lMult further to account for predation mortality

IF (PreyPop=0) or (PredPop=0)

then PredMort := 0

else PredMort := RDFuncResp(PreyPop,PredPop,alpha,htime) \* (PredPop/PreyPop);

lMult := lMult / exp(PredMort);

// Finally, multiply all elements of PREY stage matrix with lMult:

for i := 1 to Stages do

for j := 1 to Stages do

PreyMatrix^[i]^[j] := lMult \* PreyMatrix^[i]^[j];

**///////////////////// FOR PREDATOR MODEL**

// EffConst is the efficiency constant that converts number of prey eaten to number of predators;

// it accounts for the trophic efficiency of energy transfer, as well as differences in body mass.

// PredRnoprey is the predator growth rate (eigenvalue of the matrix) when there is no prey.

if PredRnoprey >= 1 then

begin

userDDErrorCode := 3;

userDDErrorMsg := 'Predator must decline when there is no prey!';

exit

end;

if EffConst >=1 then

begin

This is a warning condition, but not necessarily an error, because there are some cases the predator has smaller body mass than the prey.

end;

lMult := exp(EffConst \* RDFuncResp(PreyPop,PredPop,alpha,htime)) \* PredRnoprey;

// from Pred\_R := exp(EffConst \* RDFuncResp(PreyPop,PredPop,alpha,htime)) / exp(mu);

// NOTE: This equation assumes that the eigenvalue of the PREDATOR stage matrix is 1.0,

// and all stage matrix elements are proportionally affected by predation.

// Thus, lMult is the eigenvalue of the predator stage matrix at this time step.

// If the user specified an Rmax, lMult should not be larger than Rmax (Rmax must be >1)

if (lMult > PredRmax) and (PredRmax > 1) then lMult:=PredRmax;

// Multiply all elements of PREDATOR stage matrix with lMult:

for i := 1 to Stages do

for j := 1 to Stages do

PREDMatrix^[i]^[j] := lMult \* PREDMatrix^[i]^[j];

// Finally, add ceiling-type density dependence (e.g., to simulate space limitation)

If PredPop > PredKcap then

begin

............................................

end;

// =========================================================================================

//NOT USED BUT KEEP IT AS A FUNCTION

Function RickerDDmultiplier(PopSize, Rmax, Kcap): real;

var lMult:real;

begin

if Kcap <= 0 then lMult := 0

else {see equation for "Scramble" in section 8.4 of the User Manual}

begin

lMult := (-ln(Rmax) / Kcap) \* PopSize;

if Abs(lMult)<exp\_biggest

then lMult := exp(lMult)

else if lMult>0 then lMult := pow\_biggest

else lMult := 1/pow\_biggest;

// At this point lmult~1 if N is small, and lmult=(1/Rmax) if N=K. So:

// REMOVE the following line if eigenvalue of the matrix (EV)=Rmax;

// KEEP it if EV=1. These are the only two options.

lMult := Rmax \* lMult;

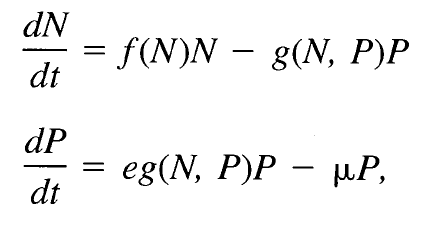
end;

RickerDD := lMult;

end;

**Derivation of the Predation Functions**

Standard predator-prey model:



where *f(N)* is the prey density dependence and *g*() is the functional response.

Rewriting as per-capita rates:

(1/*N*)(*dN/dt*) = *f(N) - g(N,P) (P/N)*

(1/*P*)(*dP/dt*) = *eg(N,P) - μ*

*g*(•) : functional response; the number of prey consumed per predator per unit time.

*g*(•) *P* : total number of prey consumed per unit time

*g*(•) (*P/N*) : proportion of prey consumed per unit time; mortality rate due to predation

To discretize, use the same approach as for DD (see Box).

**Prey equation:**

(1/*N*)(*dN/dt*) = *f(N) - g(N,P) (P/N)*

*R*prey =exp(*f(N) - g(N,P) (P/N*))

*R*prey =exp(*f(N*)) */* exp( *g(N,P) (P/N*))

where exp(*f(N*)) is the same Ricker function as in Metapop.

So, first do something similar to what is done in RAMAS for DD (multiply all stage matrix elements with lMult), then divide all stage matrix with

exp( *g(N,P) (P/N*)). Combining,

lMult := PreyRmax \* (-ln(PreyRmax) / PreyKcap) \* PreyPop; // prey density dependence

PredMort := FuncResp() \* (PredPop/PreyPop);

lMult := lMult / exp(PredMort);

// Finally, multiply all elements of PREY stage matrix with lMult

See below for FuncResp().

Note: By the way, the reason for the convoluted lMult equation is to save computation time. When K and Rmax do not change (as in the initial versions) and computers are slow (as when the initial versions were released), the simulations can be sped up by calculating Rmax\*(-ln(Rmax)/Kcap)only once for the simulation (also, it is not really that much convoluted than a power function).

**Predator equation:**

(1/*P*)(*dP/dt*) = *eg(N,P) - μ*

*R*predator =exp(*eg(N,P) - μ*).

*R*predator =exp(*eg(N,P*)) / exp( *μ*)

*R*predator =exp(*eg(N,P*)) \* (1/exp( *μ*))

*μ* is the decline rate of the predator when there is no prey. This is an instantaneous rate, and is converted to a multiplier (i.e., the eigenvalue of the predator matrix or finite rate of predator pop growth when PreyPop=0) by calculating 1/exp(*μ*). The user is then asked to specify PredRnoprey, which is assumed to be 1/exp(*μ*), and is much more intuitive. So:

// PredRnoprey is the predator growth rate (eigenvalue of the matrix) when there is no prey.

lMult := exp( e \* FuncResp()) \* PredRnoprey;

// Multiply all elements of PREDATOR stage matrix with lMult

See below for FuncResp().

**BOX:** Discretization used in RAMAS for Ricker DD:



(1/N)(dN/dt) = r (1-N/K)

Rt = (Nt+1/Nt) = exp(r(1-Nt/K))

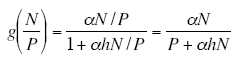
(Nt+1/Nt) = exp(r)^(1-Nt/K)

Nt+1/Nt = R^(1-Nt/K)

, which is equivalent to function used in RAMAS Metapop, which in turn is equivalent to the Ricker function (see RAMAS Metapop manual, section 4.5).

**Functional response:**

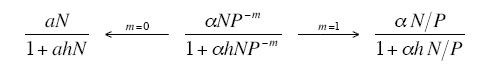
**For ratio-dependent functional response**, use equation 1.14 in Arditi & Ginzburg (2012):



where α (alpha) is the slope of the function at origin. For ratio-dependent models, it is the rate at which the prey population is made available to the predator population: the prey death rate due to predation cannot exceed alpha (Arditi & Ginzburg 2012). The handling time *h* is the time spent by the predator for handling each prey encountered, and during which it stops searching; *h* = (1/asymptote).

**FuncResp := (alpha \* PreyPop) / (PredPop + alpha \* htime \* PreyPop);**

**For general predator-dependent functional response**, use equation 2.10 in Arditi & Ginzburg (2012):



where *m* is the mutual interference coefficient. This gives ratio-dependent FR if *m*=1, and prey-dependent FR if *m*=0, and allows for intermediate values. BUT, the meaning and units of alpha (is the slope of the function at origin) change when *m* changes! At the prey-dependent (*m*=0) end, *a* is called searching efficiency (or attack rate), and can be interpreted as the proportion of prey killed per predator per unit of searching time (Arditi & Ginzburg 2012). For α (alpha) at the ratio-dependent end, see above.

**alphaNP := alpha \* PreyPop \* power(PredPop, MutualInterferenceCoeff);**

**FuncRespGen := alphaNP / (1 + htime \* alphaNP);**

**For Akçakaya (1992) ratio-dependent functional response**, use equation 4 in Akcakaya (1992):



R1 and R2 are the intrinsic (maximum) rates of increase of prey and predator populations, respectively;

X\* is the level of per capita resources of prey at which the prey population is stationary in the absence of predation (i.e., the amount of vegetation available per herbivore per unit of time at the carrying capacity without predators); similarly,

Yd is the number of prey per predator at which level the predator population is stationary.

K is a coefficient that is related to the energy efficiency of the predator as defined in Eq. 2

Function FuncRespAkcakaya92(PreyPop,PredPop,PredRnoprey,PredRmax,K,Yd):real;

**// local variables**

var **R2, D, Y, Navail : real;**

begin

IF (PreyPop=0) or (PredPop=0)

then FuncRespAkcakaya92 := 0

else begin

if PreyRefuge<0 then PreyRefuge:=0.003\*PreyPop;

if PreyRefuge<PreyPop then **Navail :=** PreyPop-PreyRefuge

else **Navail := 0;**

R2 := exp(PredRmax);

D := ln(1/PredRnoprey);

Y := Navail / PredPop;

FuncRespAkcakaya92 := ((R2+D)\*K) / (Y+(Yd\*R2)/D);

end;

end;

Parameters:

PreyPop : start with 100000

PredPop : start with 100

Prey and predator stage matrix : first try scalar (just=1); in general should have an eigenvalue of 1

PreyRmax = 21;

PredRmax = 1.33

PredRnoprey = 0.002 to 0.067

Yd = 850 to 3950

K = 630

EffConst = 1 (because K is used instead; effectively EffConst=1/630)

PreyRefuge = -1 (to start; then try other values)