Appendix B of E. Pachepsky, R. M. Nisbet, and W. W. Murdoch 'Between discrete and continuous. Consumer-resource dynamics with synchronized reproduction.' Mathematica (version 5) code to calculate the overcompensation and consumer-resource stability boundaries of the semi-discrete model as a function of $\mu$ and $\alpha$, for different values of $\rho$.

The code calculates the stability boundaries in pieces.

- data1 - bottom part of the consumer-resource boundary (heavy solid line in figure 2a);
- data2 - top part of the consumer-resource boundary (dotted line in figure 2a);
- data3 - top of the left arm of the overcompensation boundary (dashed line in figure 2a) until it crosses the consumer-resource boundary;
- data4 - bottom of the left arm of the overcompensation boundary (dashed line in figure 2a) until the minimum of the boundary;
- data5 - right arm of the overcompensation boundary (dashed line in figure 2a).

The output gives some warnings, but they are simply about names of variables, and not anything that matters.

Some parameter definitions:

- $J_{11}, J_{12}, J_{21}, J_{22}$ - elements of the Jacobian from the linear stability analysis of the model.
- Cond1 - condition for existence of a positive equilibrium;
- Cond2 - transition from stability to overcompensation cycles;
- Cond3 - transition from stability to consumer-resource cycles;
- Period - period of population cycles on the consumer-resource stability boundary (Nisbet and Gurney, Ecological Dynamics, 1998)

(* Define a convenient quantity e1 *)

```mathematica
e1[w_, t_] := Exp[r t - a w (1 - Exp[-d t])]/d;
```

(* Set parameter values *)

```mathematica
r = 20;
T = 1.0; Theta = 1.0; K = 1.0;
aMax = 35;
```

(* The equations in the code are written for the unscaled version of the model. However, since T, Theta and K are set to 1, the scaled and unscaled versions are equivalent, and in this case r corresponds to $\rho$; d corresponds to $\mu$; a corresponds to $\alpha$ *)

(* First find the first point where the resource-consumer boundary is crossed, to seed aBegin correctly *)

```mathematica
data = {};
For[d = 0.01, d < 0.01, d += 0.4,
    Print["r is ", r];
For[a = 5.0, a ≥ 2, a -= 0.5,
    Print["a is ", a];
    (* Equilibrium *)
```
\[ \text{wEq} = r d (T - (\text{Exp}[d T] - 1) / \text{Theta} / a / K) / (a (1 - \text{Exp}[-d T])); \]
\[ \text{vEq} = N[K / r (\text{el}[	ext{wEq}, T] - 1) / \text{NIntegrate}[	ext{el}[	ext{wEq}, t], \{t, 0, T\}]]; \]
Print["W equilibrium is ", \text{wEq}];
Print["V equilibrium is ", \text{vEq}];
(* Commented out is the code to actually run the model. Uncomment to see what the dynamics within year actually looks like.
V0=\text{vEq}; W0=\text{wEq};
runModel := \text{NDSolve}[\]
\[ \{ \partial_t R[t] = r R[t] (1 - R[t] / K) - a R[t] S[t], \]
\[ \partial_t S[t] = -d S[t], \]
\[ \partial_t B[t] = a R[t], \]
\[ R[0] := V0, \]
\[ S[0] := W0, \]
\[ B[0] = 0, \}
\]
\[ \{ \text{R}, \text{S}, \text{B}\}, \{t, 0, T\}; \]
\] \text{Rtotal} = \{ \text{V0} \}; \text{Stotal} = \{ \text{W0} \};
\]
For[y = 0, y < 300, y++,
\text{sampleRun} = \text{runModel};
\text{V0} = \text{R[T] /. sampleRun}[[1, 1]]; \]
\text{W0} = (\text{Theta} \text{B[T]} + 1) \text{S[T]} /. \{ \text{sampleRun}[[1, 3]], \text{sampleRun}[[1, 2]] \};
\text{Rtotal} = \text{Append}[\text{Rtotal}, \text{V0}];
\text{Stotal} = \text{Append}[\text{Stotal}, \text{W0}];
]
\text{ListPlot}[\text{Rtotal}, \text{PlotJoined} \rightarrow \text{True}];
\text{ListPlot}[\text{Stotal}, \text{PlotJoined} \rightarrow \text{True}];
\text{Vmin} = \text{Min}[\text{Rtotal}]; \text{Vmax} = \text{Max}[\text{Rtotal}];
\text{Wmin} = \text{Min}[\text{Stotal}]; \text{Wmax} = \text{Max}[\text{Stotal}];
Print["V \in (", \text{Vmin}, ", ", \text{Vmax}, ", \), W \in (", \text{Wmin}, ", ", \text{Wmax}, ", \)\];
(* Explicit expressions for the Jacobian *)
\text{rvEqK} = r \text{vEq} / K;
\text{J11} = 1 / \text{el}[	ext{wEq}, T];
\text{J12} = N[a \text{vEq} / d (\text{Exp}[-d T] - (\text{rvEqK NIntegrate}[	ext{el}[	ext{wEq}, t] \text{Exp}[-d t], \{t, 0, T\}] + 1) / (\text{rvEqK NIntegrate}[	ext{el}[	ext{wEq}, t], \{t, 0, T\}] + 1))];
\text{J21} = \text{wEq} (1 - \text{Exp}[-d T]) / \text{vEq} - \text{Theta} a \text{wEq} \text{Exp}[-d T] \text{rvEqK NIntegrate}[	ext{el}[	ext{wEq}, t] \text{NIntegrate}[	ext{el}[	ext{wEq}, z], \{z, 0, t\}] + 1) / (\text{rvEqK NIntegrate}[	ext{el}[	ext{wEq}, z], \{z, 0, t\}] + 1)^2, \{t, 0, T\}];
\text{J22} = 1 - \text{Exp}[-d T] \text{Theta} a^2 / \text{d vEq wEq NIntegrate}[	ext{el}[	ext{wEq}, t] \text{NIntegrate}[	ext{el}[	ext{wEq}, z] \text{Exp}[-d z] - \text{Exp}[-d t], \{z, 0, t\}] + (1 - \text{Exp}[-d t])) / (\text{rvEqK NIntegrate}[	ext{el}[	ext{wEq}, z], \{z, 0, t\}] + 1)^2, \{t, 0, T\};
\text{Period} = 2 \pi / \text{ArcCos}[(\text{N[J11]} + \text{N[J22]}) / 2];
(*Print["J11 is ", \text{N[J11]}, ", \nJ12 is ", \text{N[J12]}, ", \nJ21 is ", \text{N[J21]}, ", \nJ22 is ", \text{N[J22]}, ", \nPeriod is ", \text{Period});*)
\text{B1} = -(\text{N[J11]} + \text{N[J22]});
\text{B2} = \text{N[J11]} \text{N[J22]} - \text{N[J12]} \text{N[J21]};
\text{Cond1} = \text{B1} + \text{B2} + 1;
\text{Cond2} = 1 - \text{B1} + \text{B2};
\text{Cond3} = 1 - \text{B2};
(*Print["B1 is ", \text{B1}, ", \nB2 is ", \text{B2});*)
Print["\nCond1 is ", \text{Cond1}, ", Cond2 is ", \text{Cond2}, ", \nCond3 is ", \text{Cond3}];
data = \text{Append}[\text{data}, \{d, a, \text{vEq}, \text{wEq}, \text{Cond1}, \text{Cond2}, \text{Cond3}\}];
Print[data];

(* Now start up the consumer-resource boundary *)

(* Equilibrium *)

wEq = r d (T - (Exp[-d T] - T - 1) / Theta / a / K) / (a (1 - Exp[-d T]));

vEq = K / r (el[wEq, T] - 1) / NIntegrate[el[wEq, t], {t, 0, T}];

rvEqK = r vEq / K;

(* Explicit expressions for the Jacobian *)

J11 = 1 / el[wEq, T];

J12 = a vEq / d (Exp[-d T] - (rvEqK NIntegrate[el[wEq, t] Exp[-d t], {t, 0, T}] + 1) / (rvEqK NIntegrate[el[wEq, t], {t, 0, T}] + 1));

J21 = wEq (1 - Exp[-d T]) / vEq - Theta a wEq Exp[-d T] rvEqK

NIntegrate[el[wEq, t] NIntegrate[el[wEq, z], {z, 0, t}] / (rvEqK NIntegrate[el[wEq, z], {z, 0, t}] + 1) ^ 2, {t, 0, T}];

J22 = 1 - Exp[-d T] Theta a^2 / d vEq wEq NIntegrate[el[wEq, t] (rvEqK NIntegrate[el[wEq, z] (Exp[-d z] - Exp[-d t]), {z, 0, t}] + (1 - Exp[-d t])) / (rvEqK NIntegrate[el[wEq, z], {z, 0, t}] + 1) ^ 2, {t, 0, T}];

B1 = -(J11 + J22);

B2 = J11 J22 - J12 J21;

Cond1 = B1 + B2 + 1;

Cond2 = 1 - B1 + B2;

Cond3 = 1 - B2;

(* See if we are at the boundary *)

Cond3Prev = Cond3New;

Cond3New = Cond3;

LogisticStabilityBoundaries_mu_aNew.nb
If[Cond3Prev*Cond3New < 0, test = 1; 
Period = 2 Pi/ArcCos[(J11 + J22)/2]; 
Print["d: ", d, " a: ", a]; 
Print["Weq: ", wEq, " Veq: ", vEq, 
" Cond2: ", Cond2, " Cond3: ", Cond3, " Period ", Period]; 
If[d > 2*dBreak, 
testd = 1; 
]; 
If[Cond2 > 0, 
data1 = Append[data1, {d, a, vEq, wEq, Cond1, Cond2, Cond3, Period}], 
If[dBreakCount < 1, dBreak = d; dBreakCount = 1;]; 
data2 = Append[data2, {d, a, vEq, wEq, Cond1, Cond2, Cond3, Period}]; 
]; 
]; 
]; 
]; 
Print[data1]; 
Print[data2]; 

(* Now go along the overcompensation boundary *)
Print["OVERCOMPENSATION BOUNDARY."]; 

d = data1[[Dimensions[data1, 1], 1]][[1]]; 
a = data1[[Dimensions[data1, 1], 2]][[1]]; 
dCurrent = d; 
aCurrent = a; 

pr = 6; (* decimals in precision goal *)
dStep = d/80; 
aStep = a/50; 
(* adjust dStep and aStep to make sure that 
Cond2 is very close to 0 when the boundary is crossed *)
data3 = {}; 
Print["Calculating overcompensation boundary."]; 
Print["Get the left arm of the boundary."]; 
For[a = aCurrent, a ≤ aMax, a += aStep, 
Cond2Prev = 1; 
Cond2New = 1; 
test = 0; 
While[test < 1, d += dStep; 
(* Equilibrium *)

wEq = r d (T - (Exp[d T] - 1) / (Theta a K)) / (a (1 - Exp[-d T])); 
vEq = K / r (el[wEq, T] - NIntegrate[el[wEq, t], {t, 0, T}, PrecisionGoal → pr]; 
(* Calculate the following for faster computation *)
rvK = r vEq / K; 
(* Explicit expressions for the Jacobian *)
J11 = 1/el[wEq, T]; 
J12 = a vEq / d 
(Exp[-d T] - (rvK NIntegrate[el[wEq, t] Exp[-d t], {t, 0, T}, PrecisionGoal → pr] + 1) /
\( NIntegrate[e1[wEq, t], \{t, 0, T\}, \text{PrecisionGoal} \to pr] + 1)\)

\( J_{21} = wEq \left(1 - \text{Exp}\left[-d T\right]\right) / vEq - \text{Theta a wEq Exp}\left[-d T\right] \text{rvK} NIntegrate[e1[wEq, t] NIntegrate[e1[wEq, z], \{z, 0, t\}, \text{PrecisionGoal} \to pr] / (\text{rvK} NIntegrate[e1[wEq, z], \{z, 0, t\}, \text{PrecisionGoal} \to pr] + 1)^2, \{t, 0, T\}, \text{PrecisionGoal} \to pr] \)

\( J_{22} = 1 - \text{Exp}\left[-d t\right] \text{Theta a}^2 \text{vEq wEq NIntegrate[e1[wEq, t] (rvK NIntegrate[e1[wEq, z] (Exp[-dz] - Exp[-dt])), \{z, 0, t\}, \text{PrecisionGoal} \to pr] (1 - \text{Exp}\left[-d t\right]) / (\text{rvK} NIntegrate[e1[wEq, z], \{z, 0, t\}, \text{PrecisionGoal} \to pr] + 1)^2, \{t, 0, T\}, \text{PrecisionGoal} \to pr] \)

\( B_{1} = -(J_{11} + J_{22}) \)

\( B_{2} = J_{11} J_{22} - J_{12} J_{21} \)

\( \text{Cond1} = B_{1} + B_{2} + 1 \)

\( \text{Cond2} = 1 - B_{1} + B_{2} \)

\( \text{Cond3} = 1 - B_{2} \)

(* See if we are at the boundary *)

\( \text{Cond2Prev} = \text{Cond2New} \)

\( \text{Cond2New} = \text{Cond2} \)

\( \text{If}[\text{Cond2Prev} \ast \text{Cond2New} < 0, \text{test} = 1; \text{If}[\text{test} = 1, \text{data3} = \text{Prepend}[\text{data3}, \{d, a, vEq, wEq, \text{Cond1}, \text{Cond2}, \text{Cond3}\}] ; \text{Print}["d": , d, " a": , a]; \text{Print}["Weq": , wEq, " Veq": , vEq, " Cond2": , Cond2, " Cond3": , Cond3];] ; \text{Print}[\text{data3}]; \)

(* Get the bottom of the left arm *)

\( aMin = 0; \)

\( dStep = d / 100; \)

\( aStep = a / 100; \)

(* adjust dStep and aStep to make sure that Cond2 is very close to 0 when the boundary is crossed *)

\( \text{data4} = \{\}; \)

\( \text{For}[a = aCurrent, a > aMin, a -= aStep, \text{Cond2Prev} = 1; \text{Cond2New} = 1; \text{test} = 0; d = dCurrent; \text{While}[\text{test} < 1, d += dStep; (* Equilibrium *) \text{wEq} = r d (T - (\text{Exp}[d T] - 1) / (\text{Theta a K})) / (a (1 - \text{Exp}[\text{-d T}])); \text{vEq} = K / r (e1[wEq, T] - 1) / \text{NIntegrate[e1[wEq, t], \{t, 0, T\}, \text{PrecisionGoal} \to pr]}; (* Calculate the following for faster computation *) \text{rvK} = r vEq / K; (* Explicit expressions for the Jacobian *) \text{J11} = 1 / e1[wEq, T];] ; \text{Print}[\text{data3}]; \)

\( \text{rvK} NIntegrate[e1[wEq, t], \{t, 0, T\}, \text{PrecisionGoal} \to pr] + 1)\)
J12 = a vEq / d

(Exp[-d T] - (rvK NIntegrate[el[wEq, t] Exp[-d t], {t, 0, T}, PrecisionGoal -> pr] + 1) /
 (rvK NIntegrate[el[wEq, t], {t, 0, T}, PrecisionGoal -> pr] + 1));

J21 = wEq (1 - Exp[-d T]) / vEq - Theta a wEq Exp[-d T] rvK 
NIntegrate[el[wEq, t] NIntegrate[el[wEq, z], {z, 0, t}, PrecisionGoal -> pr] /
 (rvK NIntegrate[el[wEq, z], {z, 0, t}, PrecisionGoal -> pr] + 1)^2,
{t, 0, T}, PrecisionGoal -> pr];

J22 = 1 - Exp[-d T] Theta a^2 / d vEq wEq NIntegrate[el[wEq, t] 
(rvK NIntegrate[el[wEq, z] (Exp[-d z] - Exp[-d t]), {z, 0, t}, PrecisionGoal -> pr]
 (1 - Exp[-d t])) / (rvK NIntegrate[el[wEq, z], {z, 0, t}, PrecisionGoal -> pr] +
 1) ^ 2, {t, 0, T}, PrecisionGoal -> pr];

B1 = -(J11 + J22);
B2 = J11 J22 - J12 J21;
Cond1 = B1 + B2 + 1;
Cond2 = 1 - B1 + B2;
Cond3 = 1 - B2;

(* See if we are at the boundary *)
Cond2Prev = Cond2New;
Cond2New = Cond2;
If[Cond2Prev * Cond2New < 0,
  test = 1;
  If[test == 1,
    data4 = Append[data4, {d, a, vEq, wEq, Cond1, Cond2, Cond3}];
  ];
  Print["d: ", d, " a: ", a];
  Print["Weq: ", wEq, " Veq: ", vEq, " Cond2: ", Cond2, " Cond3: ", Cond3];
]
If[d > 2.0, aMin = a;]
];
Print[data4];
Print["Get the right arm of the boundary."];

d = data4[[Dimensions[data4, 1], 1]][[1]];
daCurrent = data4[[Dimensions[data4, 1], 2]][[1]];
aStep = a/20;
dStep = d/80;
data5 = {};
For[a = aCurrent, a ≤ aMax(*(aCurrent+aStep*500)*), a += aStep, (*Print["a is ", a];*)
  Cond2Prev = -1;
  Cond2New = -1;
  test = 0;
  While[test < 1, d += dStep;
    (* Equilibrium *)
    wEq = r d (T - (Exp[d T] - 1) / (Theta a K)) / (a (1 - Exp[-d T]));
    vEq = K / r (el[wEq, T] - 1) / NIntegrate[el[wEq, t], {t, 0, T}, PrecisionGoal → pr];
    (* Calculate the following for faster computation *)
    rvK = r vEq / K;
    (* Explicit expressions for the Jacobian *)
    J11 = 1/el[wEq, T];
    J12 = a vEq / d
      (Exp[-d T] - (rvK NIntegrate[el[wEq, t] Exp[-d t], {t, 0, T}, PrecisionGoal → pr] + 1) / (rvK NIntegrate[el[wEq, t], {t, 0, T}, PrecisionGoal → pr] + 1));
    J21 = wEq (1 - Exp[-d T]) / vEq - Theta a wEq Exp[-d T] rvK
      NIntegrate[el[wEq, t] NIntegrate[el[wEq, z], {z, 0, t}, PrecisionGoal → pr] / (rvK NIntegrate[el[wEq, z], {z, 0, t}, PrecisionGoal → pr] + 1)^2,
      {t, 0, T}, PrecisionGoal → pr];
    J22 = 1 - Exp[-d T] Theta a^2 / d vEq wEq NIntegrate[el[wEq, t]
      (rvK NIntegrate[el[wEq, z] (Exp[-d z] - Exp[-d t]), {z, 0, t}, PrecisionGoal → pr]
      (1 - Exp[-d t])) / (rvK NIntegrate[el[wEq, z], {z, 0, t}, PrecisionGoal → pr] + 1)^2,
      {t, 0, T}, PrecisionGoal → pr];
    B1 = -(J11 + J22);
    B2 = J11 J22 - J12 J21;
    Cond1 = B1 + B2 + 1;
    Cond2 = 1 - B1 + B2;
    Cond3 = 1 - B2;
    (* See if we are at the boundary *)
    Cond2Prev = Cond2New;
    Cond2New = Cond2;
    (*Print["Cond 2 is ", Cond2];*)
    If[Cond2Prev ∗ Cond2New < 0,
      test = 1;
      data5 = Append[data5, {d, a, vEq, wEq, Cond1, Cond2, Cond3}];
      Print["d: ", d, " a: ", a];
      Print["W eq: ", wEq, " V eq: ", vEq, " Cond2: ", Cond2, " Cond3: ", Cond3];
    ]
  ];
]

data5
plot3 = Plot[Exp[m - 1, {m, 0, 3.5}];
plot4 = Plot[10 Exp[m - 1, {m, 0, 3.5}];
b2 = Join[data3, data4, data5];
plot5 = ListPlot[Transpose[{b2[[All, 1]], b2[[All, 2]]}],
   PlotJoined -> True, PlotRange -> All, PlotStyle -> {Thickness[0.007]}];
plot6 = ListPlot[Transpose[{data1[[All, 1]], data1[[All, 2]]}],
   PlotJoined -> True, PlotRange -> All, PlotStyle -> {Thickness[0.007]}];
plot7 = ListPlot[Transpose[{data2[[All, 1]], data2[[All, 2]]}], PlotJoined -> True,
   PlotRange -> All, PlotStyle -> {Thickness[0.007], Dashing[{0.04}]}];
Show[plot3, plot5, plot6, plot7, Frame -> True, FrameLabel -> {"μ", "α"},
   PlotRange -> {{0, 3.5}, {0, 25}}]
plot8 = ListPlot[Transpose[{data1[[All, 1]], data1[[All, 8]]}], Frame -> True,
   FrameLabel -> {"μ", "Consumer-resource period"}, PlotJoined -> True];