Competition colonization trade-off

General model

Population dynamics

$$\frac{dp_i}{dt} = \beta_i\, p_i \left(1 - \sum_{\forall j} (1 - \gamma_{j,i})\, p_j\right) - p_i \left(\sum_{\forall j} \gamma_{i,j}\, \beta_j\, p_j\right) - \delta_i\, p_i;$$

←−−−−−−−−−−−→ per offspring probability of colonization

←−−−−−→ rate of displacement from a site

with

p_i = proportion of sites occupied by ' species ' i

and

β_i = per capita birth rate for type i
δ_i = per capita death rate for type i
$\gamma_{i,j}$ = prob. of site owner of type i being displaced by intruder of type j

Rewrite as

$$\frac{dp_i}{dt} = r_i\, p_i \left(1 - \frac{\sum_{\forall j} a_{i,j}\, p_j}{K_i}\right)$$

with

$$r_i = \beta_i - \delta_i$$

$$K_i = 1 - \frac{\delta_i}{\beta_i}$$

$$a_{i,j} = 1 - \gamma_{j,i} + \frac{\beta_j}{\beta_i}\,\gamma_{i,j}$$

Strategy

x_i = body size

hence we write

$\beta_i = \beta[x_i]$
$\delta_i = \delta[x_i]$
$\gamma_{i,j} = \gamma[x_i, x_j]$

Invasion fitness (monomorphic)

$$s_x[y] = r[y]\left(1 - \frac{a[y, x]\,K[x]}{K[y]}\right)$$

Invasion fitness (dimorphic)

$$s_{x_1,x_2}[y] = r[y]\left(1 - \frac{a[y, x_1]\,n_1[x_1, x_2] + a[y, x_2]\,n_2[x_1, x_2]}{K[y]}\right)$$

with

$$n_1[x_1, x_2] = \frac{K[x_1] - a[x_1, x_2]\,K[x_2]}{1 - a[x_1, x_2]\,a[x_2, x_1]}$$

$$n_2[x_1, x_2] = \frac{K[x_2] - a[x_2, x_1]\,K[x_1]}{1 - a[x_1, x_2]\,a[x_2, x_1]}$$

Implementation

Ecology

In[1]:=
```
r[x_] = β[x] - δ[x];

K[x_] = 1 - δ[x]/β[x];

a[x_, y_] = 1 - γ[y, x] + γ[x, y] β[y]/β[x];
```

Invasion fitness, gradient, curvature, drift (*monomorphic*)

In[4]:=

```
sx_[y_] = r[y] (1 - a[y, x] n11[x] / K[y]);

n11[x_] = K[x];

grad[x_] = ∂η sξ[η] /. {ξ → x, η → x};

xCurv[x_] = ∂ξ,ξ sξ[η] /. {ξ → x, η → x};
yCurv[x_] = ∂η,η sξ[η] /. {ξ → x, η → x};

driftMonomorph[x1_] = .5 μ σ2[x] K[x] grad[x];
```

Invasion fitness, gradient, curvature, drift (*dimorphic*)

In[10]:=

```
sx1_,x2_[y_] = r[y] (1 - (a[y, x1] n21[x1, x2] + a[y, x2] n22[x1, x2]) / K[y]);

n21[x1_, x2_] = (K[x1] - a[x1, x2] K[x2]) / (1 - a[x1, x2] a[x2, x1]);

n22[x1_, x2_] = (K[x2] - a[x2, x1] K[x1]) / (1 - a[x1, x2] a[x2, x1]);

grad1[x1_, x2_] = ∂η sξ1,ξ2[η] /. {ξ1 → x1, ξ2 → x2, η → x1};
grad2[x1_, x2_] = ∂η sξ1,ξ2[η] /. {ξ1 → x1, ξ2 → x2, η → x2};

curv1[x1_, x2_] = ∂η,η sξ1,ξ2[η] /. {ξ1 → x1, ξ2 → x2, η → x1};
curv2[x1_, x2_] = ∂η,η sξ1,ξ2[η] /. {ξ1 → x1, ξ2 → x2, η → x2};

driftDimorph[{x1_, x2_}] =
  {.5 μ σ2[x1] n21[x1, x2] grad1[x1, x2],
   .5 μ σ2[x2] n22[x1, x2] grad2[x1, x2]};
```

Invasion fitness, gradient, curvature, drift (*trimorphic*)

In[18]:=
```
s_{x1_,x2_,x3_}[y_] =
  r[y]
   (1 - (a[y, x1] n31[x1, x2, x3] + a[y, x2] n32[x1, x2, x3] + a[y, x3] n33[x1, x2, x3]) / K[y]);

n31[x1_, x2_, x3_] = ((a[x1, x3] a[x3, x2] - a[x1, x2] a[x3, x3]) K[x2] + a[x2, x3]
        (-a[x3, x2] K[x1] + a[x1, x2] K[x3]) + a[x2, x2] (a[x3, x3] K[x1] - a[x1, x3] K[x3])) /
    (a[x1, x3] (-a[x2, x2] a[x3, x1] + a[x2, x1] a[x3, x2]) +
      a[x1, x2] (a[x2, x3] a[x3, x1] - a[x2, x1] a[x3, x3]) +
      a[x1, x1] (-a[x2, x3] a[x3, x2] + a[x2, x2] a[x3, x3]));

n32[x1_, x2_, x3_] = ((-a[x1, x3] a[x3, x1] + a[x1, x1] a[x3, x3]) K[x2] + a[x2, x3]
        (a[x3, x1] K[x1] - a[x1, x1] K[x3]) + a[x2, x1] (-a[x3, x3] K[x1] + a[x1, x3] K[x3])) /
    (a[x1, x3] (-a[x2, x2] a[x3, x1] + a[x2, x1] a[x3, x2]) +
      a[x1, x2] (a[x2, x3] a[x3, x1] - a[x2, x1] a[x3, x3]) +
      a[x1, x1] (-a[x2, x3] a[x3, x2] + a[x2, x2] a[x3, x3]));

n33[x1_, x2_, x3_] = ((a[x1, x2] a[x3, x1] - a[x1, x1] a[x3, x2]) K[x2] + a[x2, x2]
        (-a[x3, x1] K[x1] + a[x1, x1] K[x3]) + a[x2, x1] (a[x3, x2] K[x1] - a[x1, x2] K[x3])) /
    (a[x1, x3] (-a[x2, x2] a[x3, x1] + a[x2, x1] a[x3, x2]) +
      a[x1, x2] (a[x2, x3] a[x3, x1] - a[x2, x1] a[x3, x3]) +
      a[x1, x1] (-a[x2, x3] a[x3, x2] + a[x2, x2] a[x3, x3]));

grad31[x1_, x2_, x3_] = ∂_η s_{ξ1,ξ2,ξ3}[η] /. {ξ1 → x1, ξ2 → x2, ξ3 → x3, η → x1};
grad32[x1_, x2_, x3_] = ∂_η s_{ξ1,ξ2,ξ3}[η] /. {ξ1 → x1, ξ2 → x2, ξ3 → x3, η → x2};
grad33[x1_, x2_, x3_] = ∂_η s_{ξ1,ξ2,ξ3}[η] /. {ξ1 → x1, ξ2 → x2, ξ3 → x3, η → x3};

driftTrimorph[{x1_, x2_, x3_}] =
  {.5 μ σ2[x1] n31[x1, x2, x3] grad31[x1, x2, x3],
   .5 μ σ2[x2] n32[x1, x2, x3] grad32[x1, x2, x3],
   .5 μ σ2[x3] n33[x1, x2, x3] grad33[x1, x2, x3]};
```

Case: $\gamma[x_i, x_j] = 1 - \gamma[x_j, x_i]$ (i.e., no effect of ownership)

Functions and parameters

In[26]:=
```
β[x_] = R / x;
R = 1; (* R : per capita amount of resources *)

δ[x_] = 1;
γ[x_, y_] = c[y] / (c[x] + c[y]);

c[x_] = e^(α x); (* competitive ability; α comp asymmetry *)

μ = 1; (* mutation probability per birth event *)
σ2[x_] = .001 x (R - x); (* variance of mutation step size distribution *)
(* variance becomes zero at the boundaries of the strategy space [0,R] *)
```

Strategy range

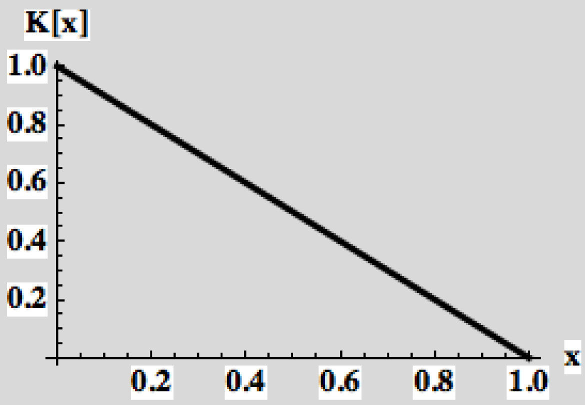
In[33]:=
```
xMin = 0;
xMax = R;
```

Carrying capacity & competition kernel

In[35]:=

```
Plot[K[x], {x, xMin, xMax}, AxesOrigin → {0, 0}, PlotStyle → {Thick, Black},
 AxesLabel → {"x", "K[x]"}, LabelStyle → {Bold}, ImageSize → {Small}]
```

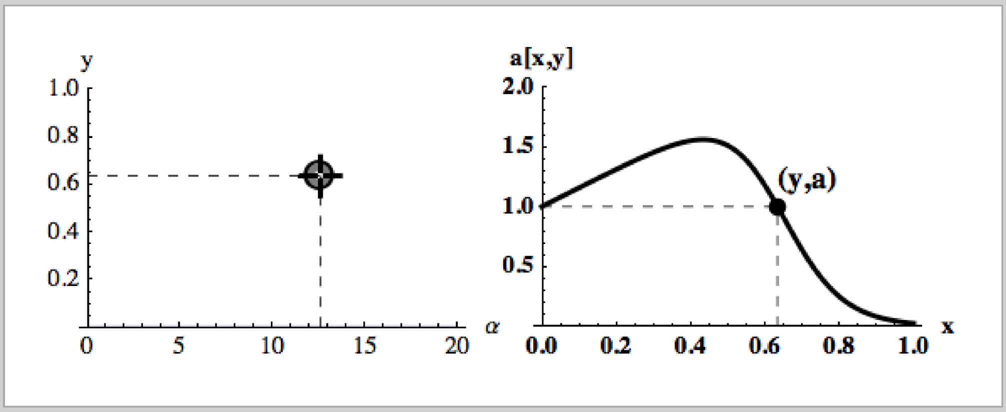
Out[35]=



In[36]:=

```
Manipulate[
 α = αy[[1]];
 y = αy[[2]];
 Row[{
   Show[
    Plot[0, {α, 0, 20}, AxesOrigin → {0, xMin}, PlotRange → {xMin, xMax},
     AxesLabel → {"α", "y"}, Frame → False, ImageSize → Small],
    Graphics[Locator[αy], PlotRange → {{0, 20}, {xMin, xMax}}],
    Graphics[{Dashed, Line[{{{α, xMin}, {α, y}}, {{0, y}, {α, y}}}]}]
   ],

   Show[
    Plot[a[x, y], {x, xMin, xMax},
     AxesOrigin → {0, 0}, AxesLabel → {"x", "a[x,y]"}, LabelStyle → {Bold},
     PlotStyle → {Black, Thick}, PlotRange → {0, 2}, ImageSize → Small],
    Graphics[{PointSize[Large], Point[{y, a[y, y]}],
      Text[Style["(y,a)", Medium, Bold], {y, 1.1}, {-1, -1}]}],
    Graphics[{Dashed, Line[{{{y, 0}, {y, 1}}, {{xMin, 1}, {y, 1}}}]}]
   ]
  }],
 {{αy, {5, .5 (xMax - xMin)}}, Locator}
]
```

Out[36]=



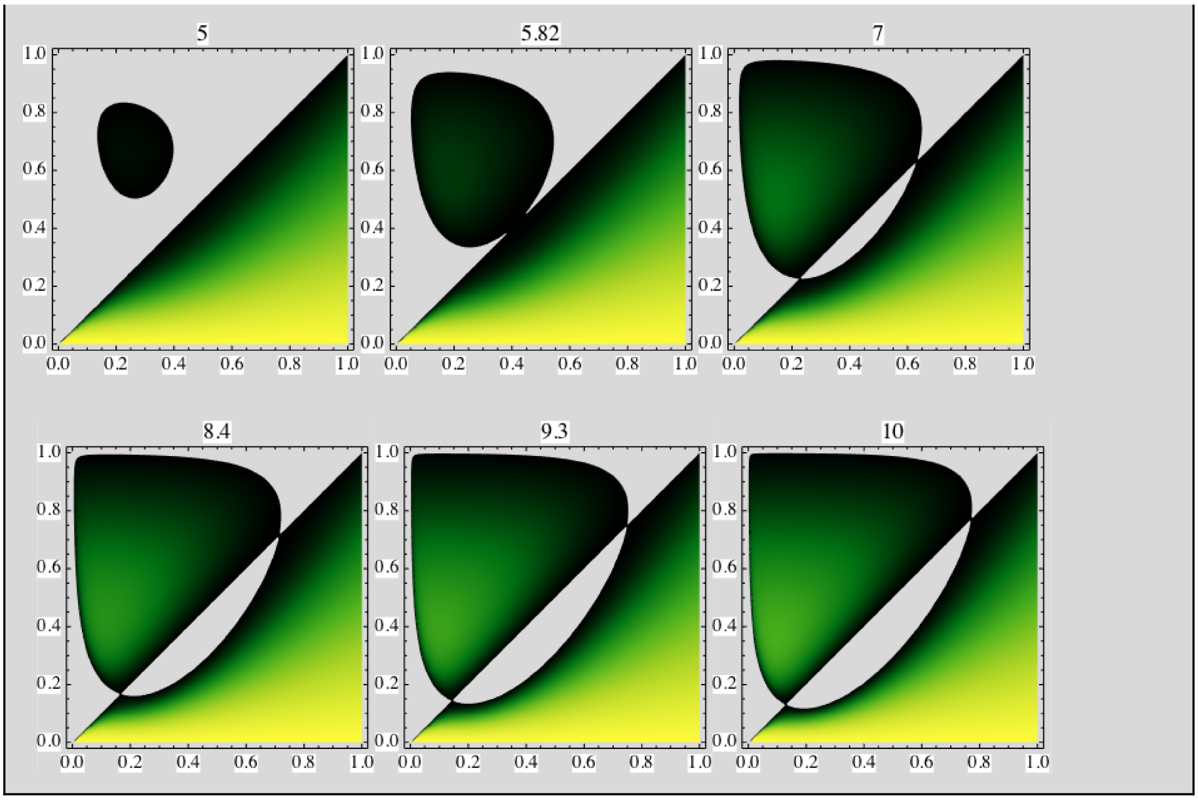
Monomorphic evolution

Pairwise invadability plots (PIP)

In[37]:=

```
Do[
  PIP[α] =
   DensityPlot[
    Block[{inv},
     inv = s_x[y];
     If[0 < inv, ArcTan[inv]]
    ],
    {x, xMin, xMax}, {y, xMin, xMax},
    PlotPoints → 50, ColorFunction → "AvocadoColors"
   ],
  {α, {5, 5.82, 7, 8.4, 9.3, 10}}
 ];

Row[
 Table[
  Show[PIP[α], PlotLabel → α, ImageSize → Small],
  {α, {5, 5.82, 7, 8.4, 9.3, 10}}]
]
```

Out[38]=

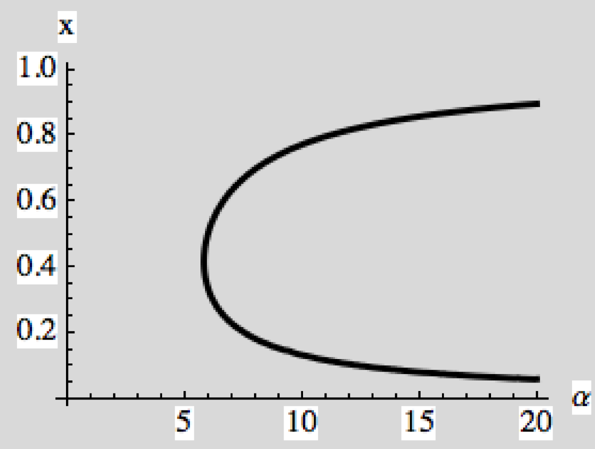


Bifurcation Plot (x vs α)

In[39]:=

```
ContourPlot[grad[x], {α, 0, 20}, {x, xMin, xMax}, Contours → {0},
  ContourStyle → {Black, Thick}, ContourShading → False, PlotPoints → 20, Frame → False,
  Axes → True, AspectRatio → .7, AxesLabel → {"α", "x"}, ImageSize → Small]
```

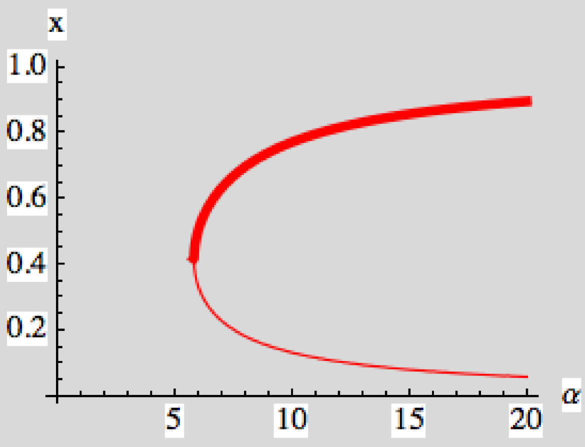
Out[39]=



In[40]:=

```
monoBifPlot =
 Show[
   (* black & thick ⟶ uninvadable attractor *)
   ContourPlot[If[yCurv[x] < Min[xCurv[x], 0], grad[x]], {α, 0, 20},
    {x, xMin, xMax}, Contours → {0}, ContourStyle → {Black, Thickness[0.02]},
    ContourShading → False, PlotPoints → 20],
   (* black & thin ⟶ uninvadable repeller *)
   ContourPlot[If[xCurv[x] < yCurv[x] < 0, grad[x]], {α, 0, 20},
    {x, xMin, xMax}, Contours → {0}, ContourStyle → {Black, Thickness[.005]},
    ContourShading → False, PlotPoints → 20],
   (* red & thin ⟶ invadable repeller *)
   ContourPlot[If[Max[xCurv[x], 0] < yCurv[x], grad[x]], {α, 0, 20},
    {x, xMin, xMax}, Contours → {0}, ContourStyle → {Red, Thickness[.005]},
    ContourShading → False, PlotPoints → 20],
   (* red & thick ⟶ invadable attractor, i.e. branching point *)
   ContourPlot[If[0 < yCurv[x] < xCurv[x], grad[x]], {α, 0, 20},
    {x, xMin, xMax}, Contours → {0}, ContourStyle → {Red, Thickness[0.02]},
    ContourShading → False, PlotPoints → 20],
   Frame → False, Axes → True, AspectRatio → .7,
   AxesLabel → {"α", "x"}, ImageSize → Small
  ] //
  Quiet
```

Out[40]=

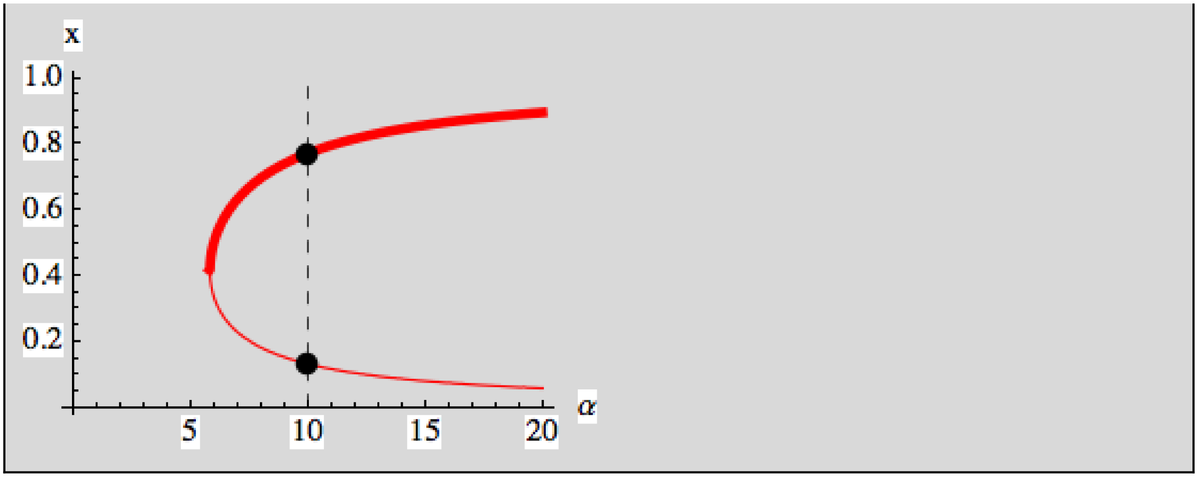


Singular strategies for α=10

In[41]:=

```
α = 10;

sing1 = ξ /. FindRoot[grad[ξ], {ξ, .1}];
sing2 = ξ /. FindRoot[grad[ξ], {ξ, .9}];

Show[
 monoBifPlot,
 Graphics[{Dashed, Line[{{10, xMin}, {10, xMax}}]}],
 Graphics[{PointSize[Large], Point[{{10, sing1}, {10, sing2}}]}],
 Frame → False, Axes → True, AspectRatio → .7,
 AxesLabel → {"α", "x"}, ImageSize → Small
]
```

Out[44]=

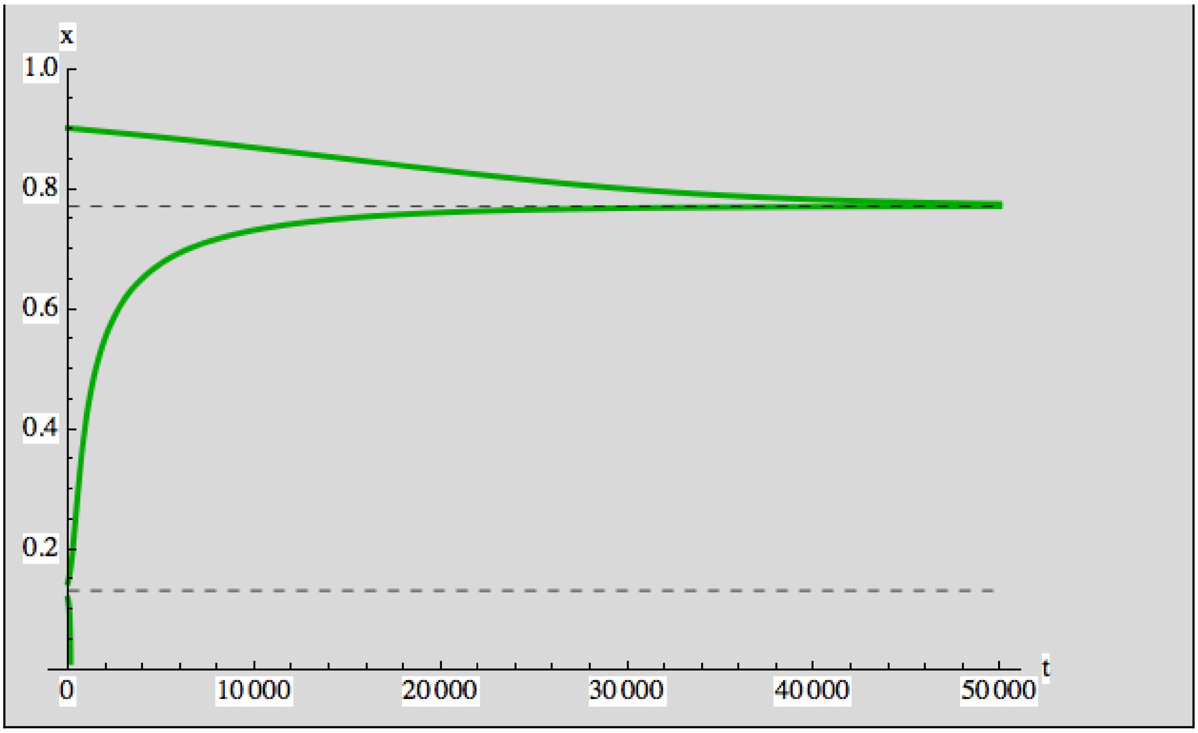


Canonical equation

In[45]:=

```
α = 10;

(* orbit 1 *)
x0 = {.9 sing1}; (* starting point *)
t0 = 0;(* start time *)
t∞ = 50000;(* stop time *)
Δt = 10;(* integration time step *)
dataMonomorph = {};
x = x0; t = t0;
While[t < t∞ ⋀ xMin < x[[1]] < xMax,
  dataMonomorph = Join[dataMonomorph, {{t, x[[1]]}}];
  x = x + Δt driftMonomorph[x];
  t = t + Δt;
 ];
CEorbitMonomorph1 = ListPlot[dataMonomorph,
   PlotStyle → {Darker[Green], Thick}, Joined → True, PlotRange → {0, 1}];

(* orbit 2 *)
x0 = {1.1 sing1}; (* starting point *)
t0 = 0;(* start time *)
t∞ = 50000;(* stop time *)
Δt = 10;(* integration time step *)
dataMonomorph = {};
x = x0; t = t0;
While[t < t∞ ⋀ xMin < x[[1]] < xMax,
  dataMonomorph = Join[dataMonomorph, {{t, x[[1]]}}];
  x = x + Δt driftMonomorph[x];
  t = t + Δt;
 ];
CEorbitMonomorph2 = ListPlot[dataMonomorph,
   PlotStyle → {Darker[Green], Thick}, Joined → True, PlotRange → {0, 1}];

(* orbit 3 *)
x0 = {.9 xMax}; (* starting point *)
t0 = 0;(* start time *)
t∞ = 50000;(* stop time *)
Δt = 10;(* integration time step *)
dataMonomorph = {};
x = x0; t = t0;
While[t < t∞ ⋀ xMin < x[[1]] < xMax,
  dataMonomorph = Join[dataMonomorph, {{t, x[[1]]}}];
  x = x + Δt driftMonomorph[x];
  t = t + Δt;
 ];
CEorbitMonomorph3 = ListPlot[dataMonomorph,
   PlotStyle → {Darker[Green], Thick}, Joined → True, PlotRange → {0, 1}];

Show[
 CEorbitMonomorph1, CEorbitMonomorph2, CEorbitMonomorph3,
 Graphics[{Dashed, Line[{{t0, sing1}, {t∞, sing1}}]}],
 Graphics[{Dashed, Line[{{t0, sing2}, {t∞, sing2}}]}],
 AxesLabel → {"t", "x"}
]
```

Out[70]=



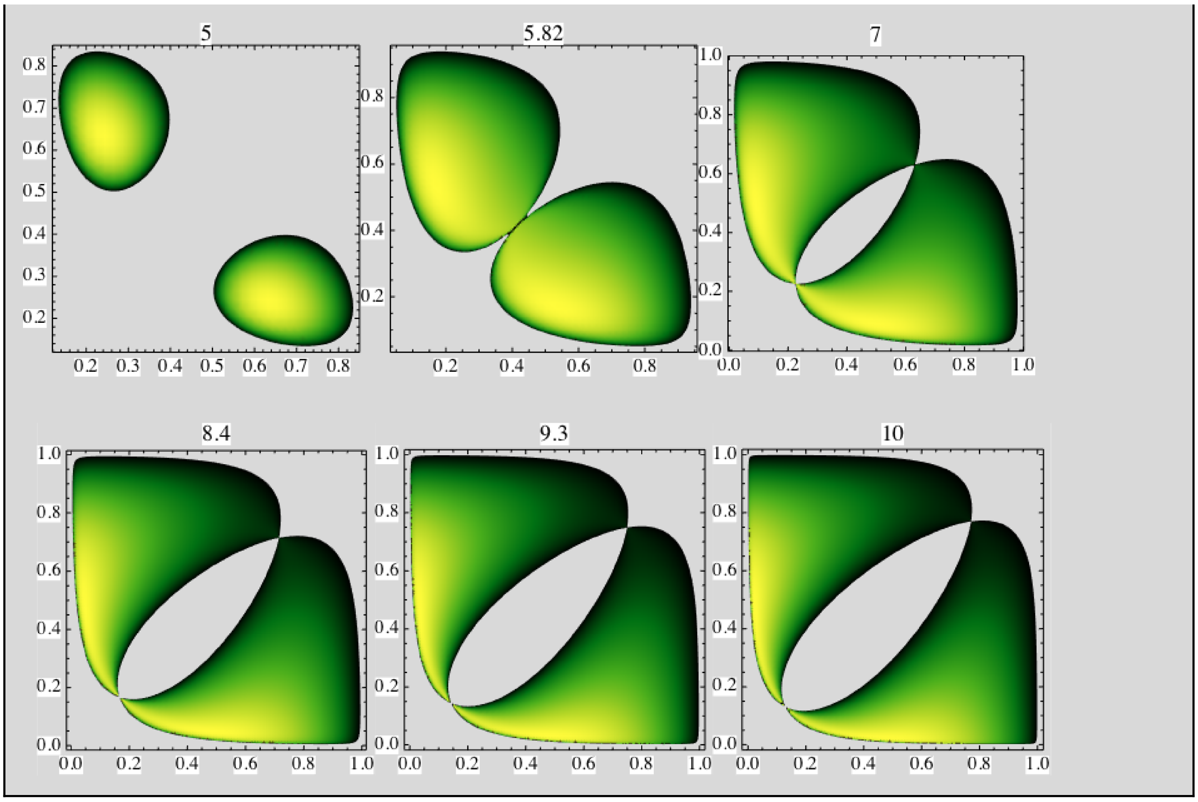
Dimorphic evolution

Mutual invadability plots (MIP)

In[71]:=

```
Do[
  MIP[α] =
   DensityPlot[Block[{inv}, inv = {s_x1[x2], s_x2[x1]};
      If[0 < inv[[1]] ∧ 0 < inv[[2]], n21[x1, x2] n22[x1, x2]]],
     {x1, xMin, xMax}, {x2, xMin, xMax},
     PlotPoints → 50,
     PlotRange → All,
     ColorFunction → "AvocadoColors"] // Quiet,
  {α, {5, 5.82, 7, 8.4, 9.3, 10}}
 ];

Row[
 Table[
  Show[MIP[α], PlotLabel → α, ImageSize → Small],
  {α, {5, 5.82, 7, 8.4, 9.3, 10}}
 ]
]
```

Out[72]=



"Anti-MIP" **(to be used later as a mask to cover what happens outside the oexistence region)**

In[73]:=

```
Do[
 antiMIP[α] =
  RegionPlot[sx[y] < 0 ∨ sy[x] < 0, {x, xMin, xMax}, {y, xMin, xMax},
    PlotPoints → 50, PlotStyle → LightGray, BoundaryStyle → None] // Quiet,
 {α, {5, 5.82, 7, 8.4, 9.3, 10}}
]
```

Evolutionary isoclines

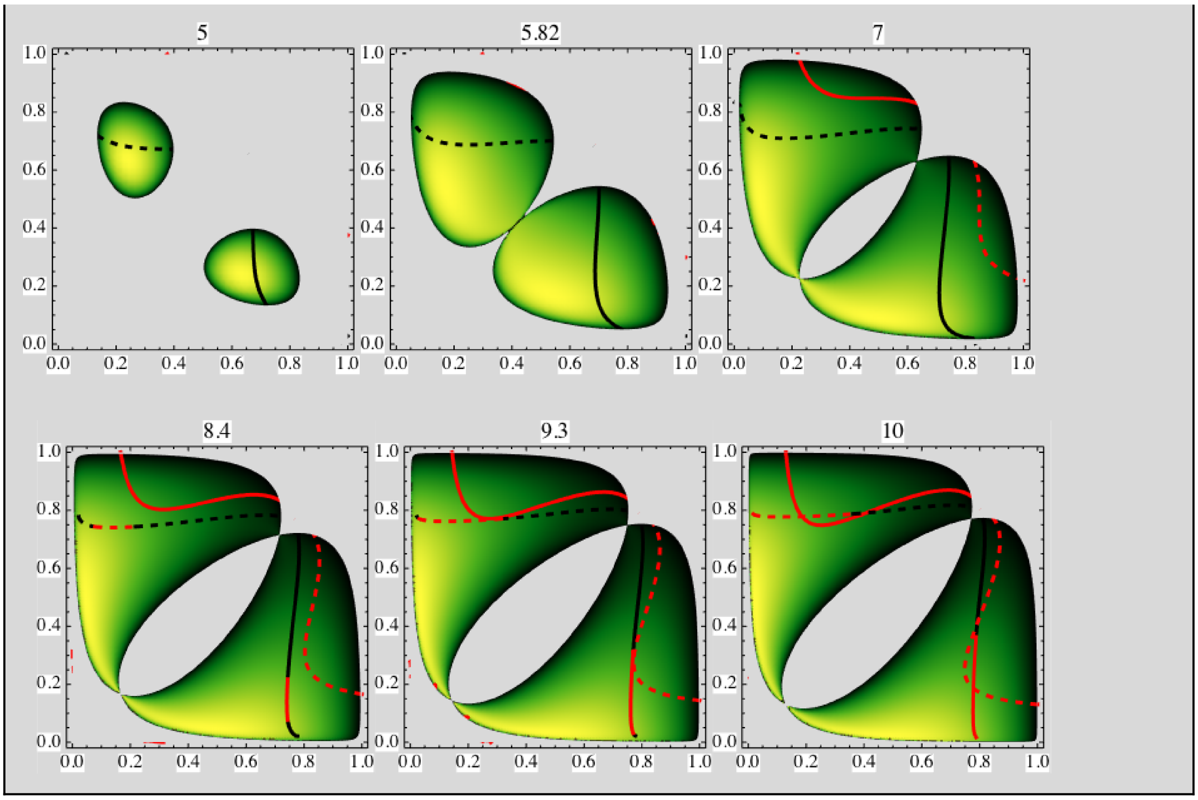
In[74]:=

```
(* VERY SLOW PROCEDURE / DO NOT RUN IN CLASSROOM*)

Do[
  IPes1[α] =
   ContourPlot[If[0 > curv1[x1, x2] ∧ (Abs[x1 - x2] > .005 (xMax - xMin)), grad1[x1, x2]],
     {x1, xMin, xMax}, {x2, xMin, xMax}, Contours → {0}, ContourStyle → {Thick, Black},
     ContourShading → False, PlotPoints → 10] // Quiet;
  IPbp1[α] = ContourPlot[If[0 < curv1[x1, x2] ∧ (Abs[x1 - x2] > .005 (xMax - xMin)),
      grad1[x1, x2]], {x1, xMin, xMax}, {x2, xMin, xMax}, Contours → {0},
     ContourStyle → {Thick, Red}, ContourShading → False, PlotPoints → 10] // Quiet;
  IPes2[α] = ContourPlot[If[0 > curv2[x1, x2] ∧ (Abs[x1 - x2] > .005 (xMax - xMin)),
      grad2[x1, x2]], {x1, xMin, xMax}, {x2, xMin, xMax}, Contours → {0}, ContourStyle →
      {Thick, Black, Dashed}, ContourShading → False, PlotPoints → 10] // Quiet;
  IPbp2[α] = ContourPlot[If[0 < curv2[x1, x2] ∧ (Abs[x1 - x2] > .005 (xMax - xMin)),
      grad2[x1, x2]], {x1, xMin, xMax}, {x2, xMin, xMax}, Contours → {0}, ContourStyle →
      {Thick, Red, Dashed}, ContourShading → False, PlotPoints → 10] // Quiet,
  {α, {5, 5.82, 7, 8.4, 9.3, 10}}
 ];
```

In[75]:=

```
Row[
 Table[
  Show[MIP[α], IPes1[α], IPbp1[α], IPes2[α], IPbp2[α], antiMIP[α],
   PlotLabel → α, ImageSize → Small],
  {α, {5, 5.82, 7, 8.4, 9.3, 10}}
 ]
]
```

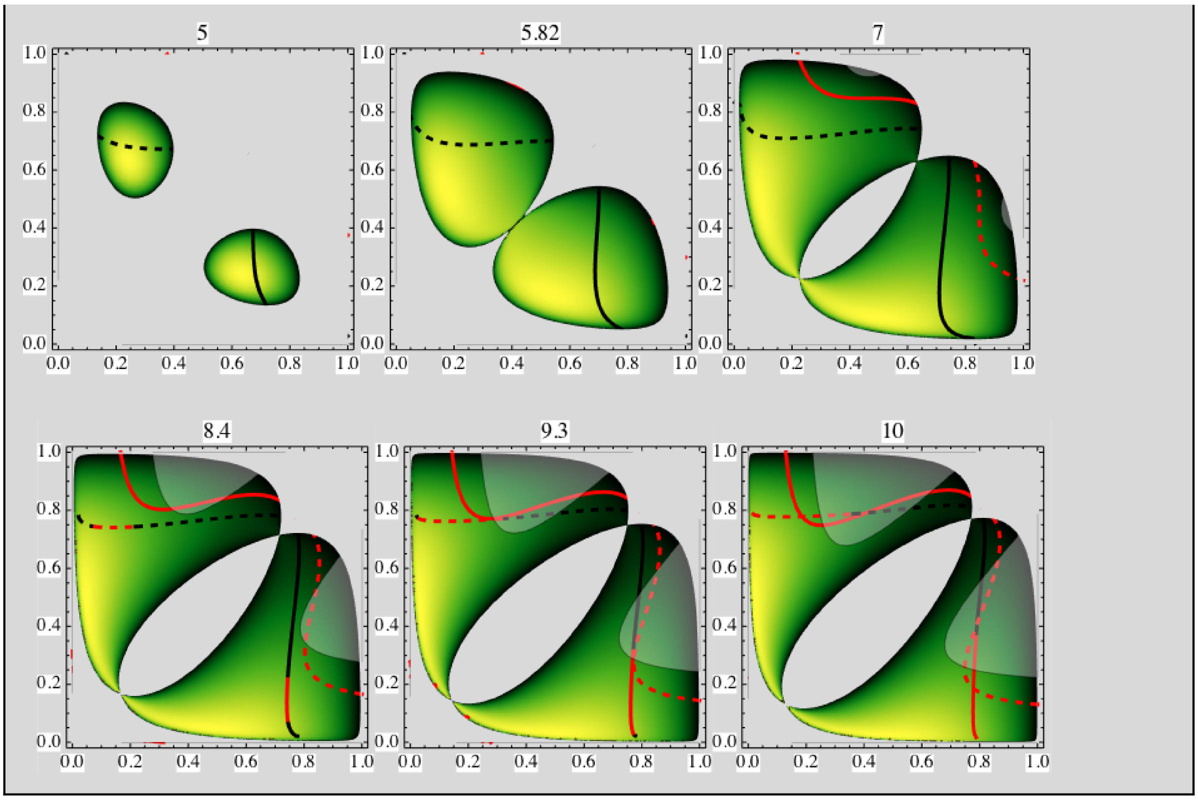
Out[75]=



Total stability

In[76]:=
```
Do[
  totStab[α] =
   Block[{A, a11, a12, a21, a22},
    A = D[{grad1[x1, x2], grad2[x1, x2]}, {{x1, x2}}];
    a11 = A[[1, 1]]; a12 = A[[1, 2]]; a21 = A[[2, 1]]; a22 = A[[2, 2]];
    RegionPlot[a11 < 0 ⋀ a22 < 0 ⋀ Abs[a12 a21] < a11 a22,
      {x1, xMin, xMax}, {x2, xMin, xMax},
      PlotStyle → {LightGray, Opacity[0.4]}] // Quiet
   ],
  {α, {5, 5.82, 7, 8.4, 9.3, 10}}
 ];

Row[
 Table[
  Show[MIP[α], IPes1[α], IPbp1[α], IPes2[α], IPbp2[α], totStab[α], antiMIP[α],
   PlotLabel → α, ImageSize → Small],
  {α, {5, 5.82, 7, 8.4, 9.3, 10}}]
]
```

Out[77]=

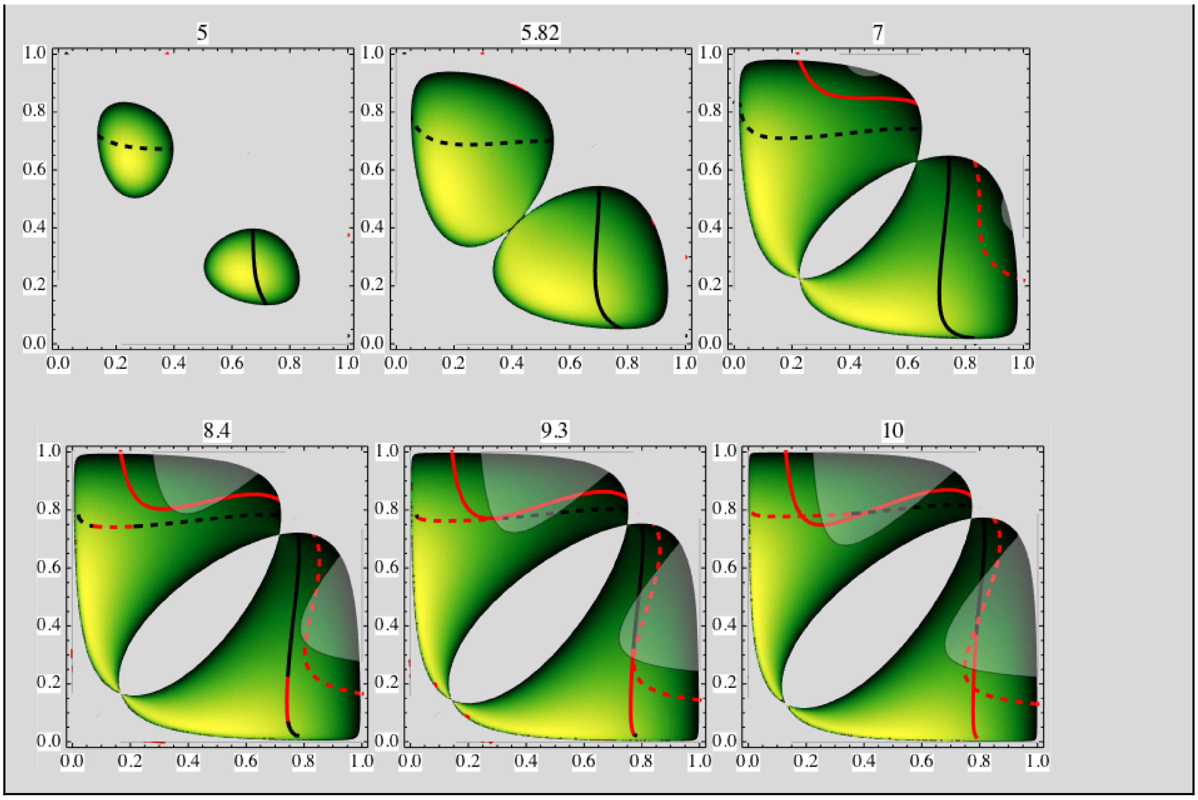


Strong stability

In[78]:=

```
Do[
  strongStab[α] =
   Block[{A, a11, a12, a21, a22},
    A = D[{grad1[x1, x2], grad2[x1, x2]}, {{x1, x2}}];
    a11 = A[[1, 1]]; a12 = A[[1, 2]]; a21 = A[[2, 1]]; a22 = A[[2, 2]];
    RegionPlot[a11 < 0 ⋀ a22 < 0 ⋀ a12 a21 < a11 a22,
      {x1, xMin, xMax}, {x2, xMin, xMax},
      PlotStyle → {LightGray, Opacity[0.4]}] // Quiet
   ],
  {α, {5, 5.82, 7, 8.4, 9.3, 10}}
 ];

Row[
 Table[
  Show[MIP[α], IPes1[α], IPbp1[α], IPes2[α], IPbp2[α], strongStab[α], antiMIP[α],
   PlotLabel → α, ImageSize → Small],
  {α, {5, 5.82, 7, 8.4, 9.3, 10}}
 ]
]
```

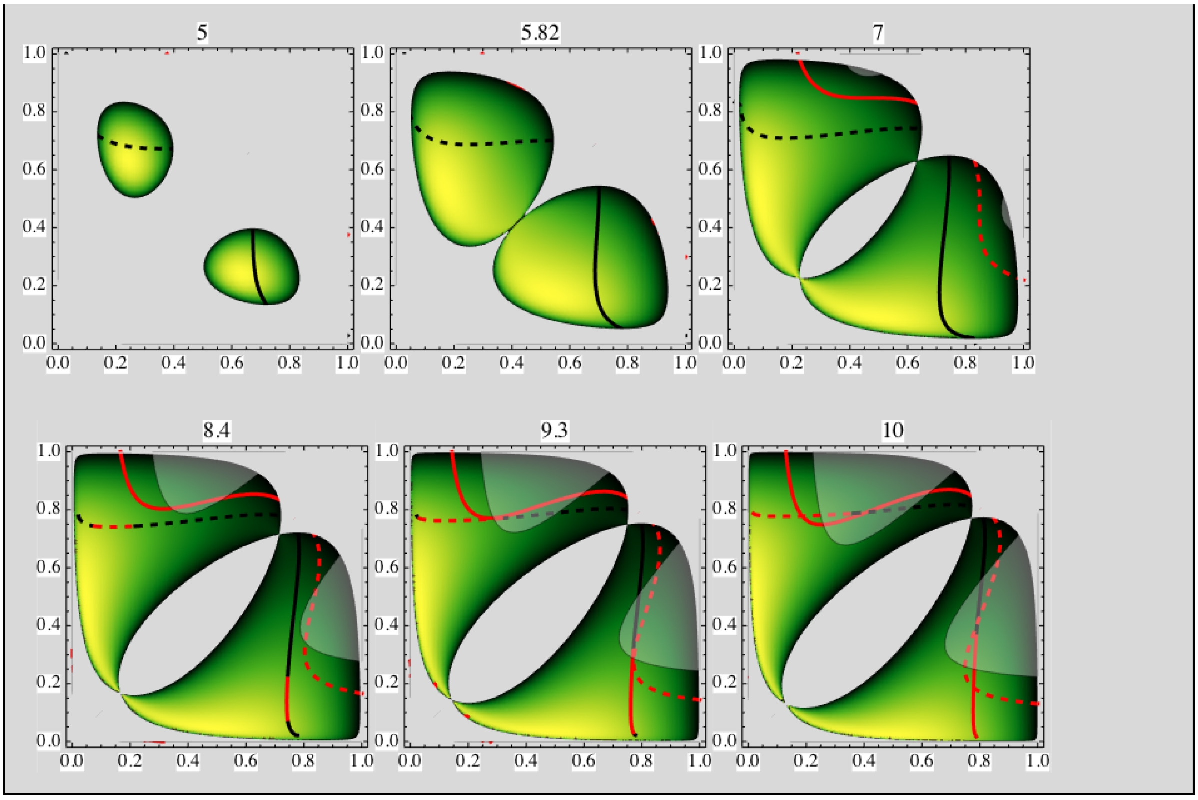
Out[79]=



Weak stability

In[80]:=
```
Do[
  weakStab[α] =
   Block[{A, a11, a12, a21, a22},
    A = D[{grad1[x1, x2], grad2[x1, x2]}, {{x1, x2}}];
    a11 = A[[1, 1]]; a12 = A[[1, 2]]; a21 = A[[2, 1]]; a22 = A[[2, 2]];
    RegionPlot[(a11 < 0 ⋁ a22 < 0) ⋀ a12 a21 < a11 a22,
      {x1, xMin, xMax}, {x2, xMin, xMax},
      PlotStyle → {LightGray, Opacity[0.4]}] // Quiet
   ],
  {α, {5, 5.82, 7, 8.4, 9.3, 10}}
 ];

Row[
 Table[
  Show[MIP[α], IPes1[α], IPbp1[α], IPes2[α], IPbp2[α], weakStab[α], antiMIP[α],
   PlotLabel → α, ImageSize → Small],
  {α, {5, 5.82, 7, 8.4, 9.3, 10}}
 ]
]
```

Out[81]=



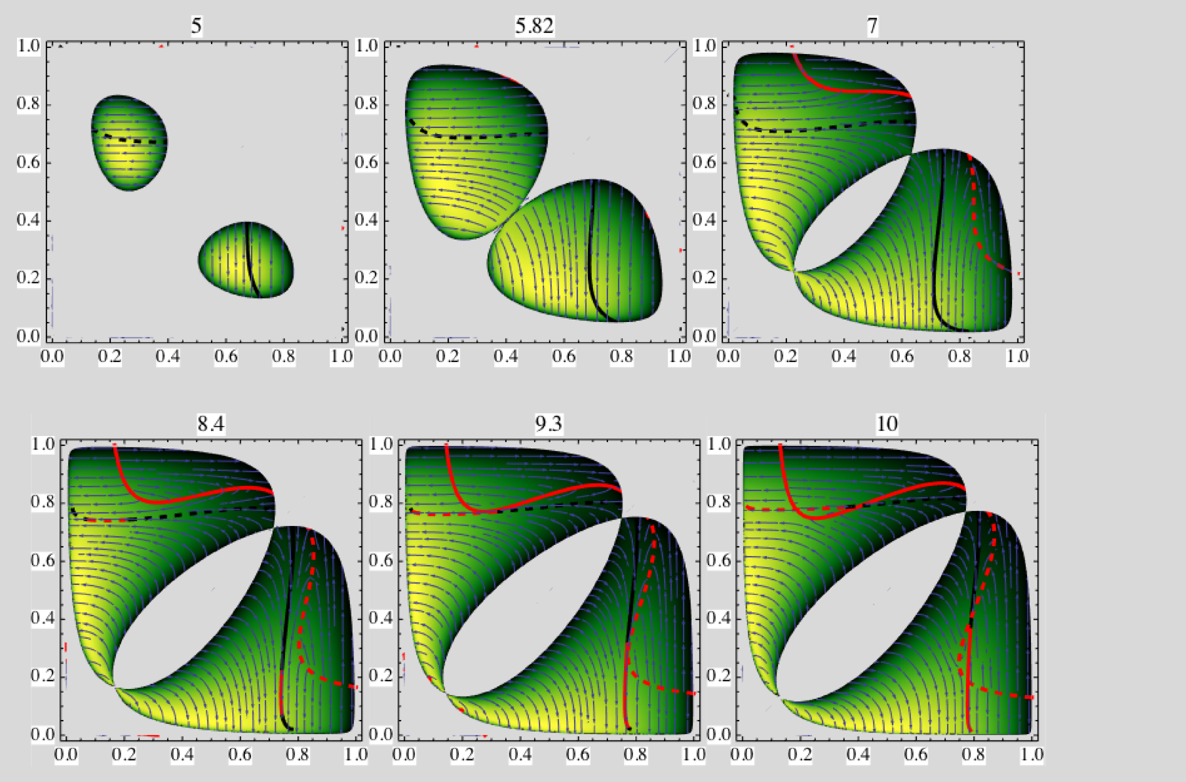
Canonical equation

In[82]:=
```
Do[
   stream[α] = StreamPlot[driftDimorph[{x1, x2}],
     {x1, xMin, xMax}, {x2, xMin, xMax}, StreamPoints → Fine],
   {α, {5, 5.82, 7, 8.4, 9.3, 10}}
  ] // Quiet;
```

In[83]:=

```
Row[
 Table[
  Show[MIP[α], IPes1[α], IPbp1[α], IPes2[α], IPbp2[α], stream[α], antiMIP[α],
   PlotLabel → α, ImageSize → Small],
  {α, {5, 5.82, 7, 8.4, 9.3, 10}}
 ]
]
```

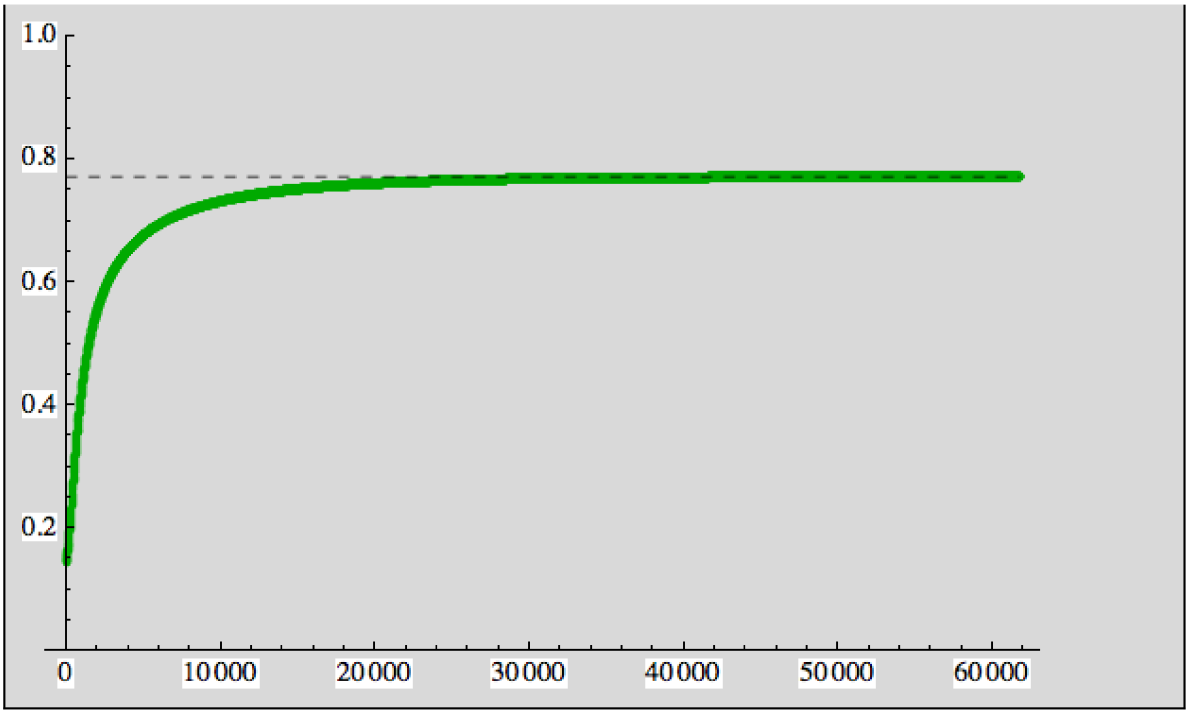
Out[83]=



Evolutionary tree

In[358]:=
```
α = 10;

sing2 = ξ /. FindRoot[grad[ξ], {ξ, .9}];

x0 = {1.1 sing1}; (* starting point *)
t0 = 0; (* start time *)
t∞ = 5 000 000; (* stop time *)
Δt = 10; (* integration time step *)
ϵ = .0001;
dataMonomorph = {};
x = x0; t = t0;
While[t < t∞ ⋀ 0 < grad[x[[1]] - ϵ] grad[x[[1]] + ϵ],
  dataMonomorph = Join[dataMonomorph, {{t, x[[1]]}}];
  x = x + Δt driftMonomorph[x];
  t = t + Δt;
 ];

CEorbitMonomorph =
 Show[
  ListPlot[dataMonomorph,
   PlotStyle → {Darker[Green], Thick}, Joined → False, PlotRange → {0, 1}],
  Graphics[{Dashed, Line[{{t0, sing2}, {t, sing2}}]}]
 ]
```

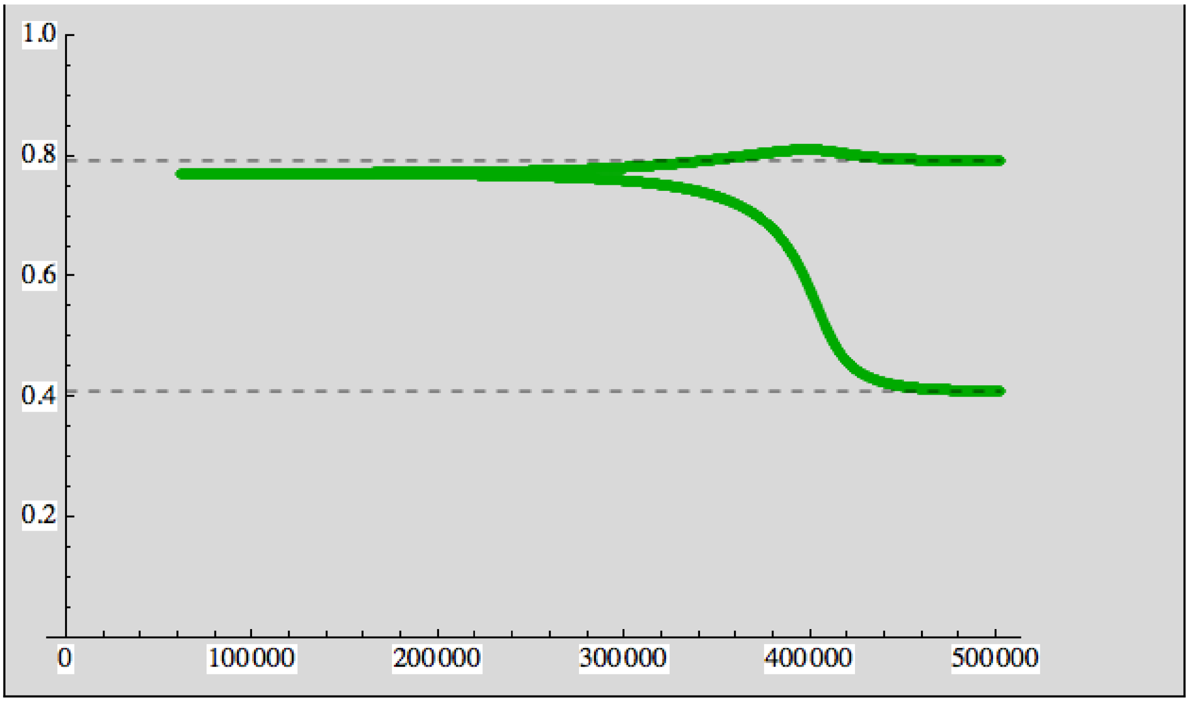
Out[365]=



In[366]:=

```
{sing21, sing22} =
  {ξ1, ξ2} /. FindRoot[{grad1[ξ1, ξ2] == 0, grad2[ξ1, ξ2] == 0}, {{ξ1, .4}, {ξ2, .8}}];

dataDimorph = {};
t = dataMonomorph[[-1]][[1]];
x = {dataMonomorph[[-1]][[2]] - ϵ, dataMonomorph[[-1]][[2]] + ϵ};
Δt = 100;
While[
  t ≤ t∞ ⋀ 0 < grad1[x[[1]] - ϵ, x[[2]]] grad1[x[[1]] + ϵ, x[[2]]],
  dataDimorph = Join[dataDimorph, {{t, x[[1]]}, {t, x[[2]]}}];
  x = x + Δt driftDimorph[x];
  t = t + Δt;
 ];

CEorbitDimorph =
 Show[
  ListPlot[dataDimorph,
   PlotStyle → {Darker[Green], Thick}, Joined → False, PlotRange → {0, 1}],
  Graphics[{Dashed, Line[{{t0, sing21}, {t, sing21}}]}],
  Graphics[{Dashed, Line[{{t0, sing22}, {t, sing22}}]}]
 ]
```

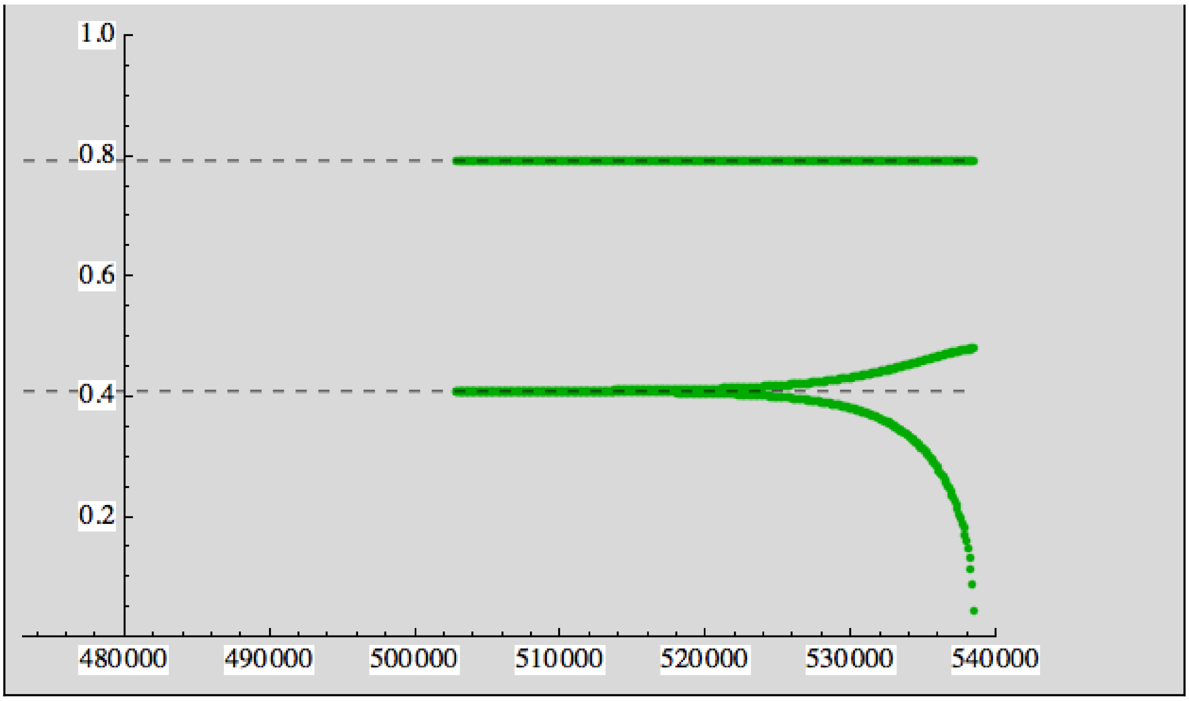
Out[372]=



In[379]:=

```
dataTrimorph = {};
x = {sing21 - ϵ, sing21 + ϵ, sing22};
t = dataDimorph[[-1]][[1]];
Δt = 100;
While[
  t ≤ t∞ ⋀ 0 < x[[1]] ⋀ 0 < x[[2]] ⋀ 0 < x[[3]],
  dataTrimorph = Join[dataTrimorph, {{t, x[[1]]}, {t, x[[2]]}, {t, x[[3]]}}];
  dat = Join[dat, {{t, n31[x[[1]], x[[2]], x[[3]]]}}];
  x = x + Δt driftTrimorph[x];
  t = t + Δt;
 ];

CEorbitTrimorph =
 Show[
  ListPlot[dataTrimorph,
   PlotStyle → {Darker[Green], Thick}, Joined → False, PlotRange → {0, 1}],
  Graphics[{Dashed, Line[{{t0, sing21}, {t, sing21}}]}],
  Graphics[{Dashed, Line[{{t0, sing22}, {t, sing22}}]}]
 ]
```

Out[384]=



```
Show[
 ListPlot[Join[dataMonomorph, dataDimorph, dataTrimorph],
  PlotStyle → {Darker[Green], Thick}, Joined → False, PlotRange → {0, 1}],
 Graphics[{Dashed, Line[{{t0, sing21}, {t, sing21}}]}],
 Graphics[{Dashed, Line[{{t0, sing22}, {t, sing22}}]}]
]
```

Out[91]=