The Lotka - Volterra cannibalism time budget model

Resident population dynamics

$$\frac{d}{dt}R = r\,R\left(1-\frac{R}{K}\right) - \alpha\,R\sum_{j=1}^{l}\left(1-x_j\right)n_j$$

$$\frac{d}{dt}n_i = \epsilon\,\alpha\,R\,(1-x_i)\,n_i - \delta\,n_i + \gamma\,\beta[x_i]\,x_i\,n_i\left(\sum_{j=1}^{l}\left(1-x_j\right)n_j\right) - (1-x_i)\,n_i\left(\sum_{j=1}^{l}\beta\left[x_j\right]x_j\,n_j\right)\ \ (i=1,\ \ldots,\ k)$$

Invader population dynamics

$$\frac{d}{dt}\mathrm{Log}[m] = \epsilon\,\alpha\,R\,(1-y) - \delta + \gamma\,\beta[y]\,y\,\left(\sum_{j=1}^{l}\left(1-x_j\right)n_j\right) - (1-y)\left(\sum_{j=1}^{l}\beta\left[x_j\right]x_j\,n_j\right)$$

$$= E_1\,(1-y) - \delta + E_2\,\beta[y]\,y - E_3\,(1-y)$$

where $E_1 = \epsilon\,\alpha\,R$ and $E_2 = \gamma\sum_{j=1}^{l}\left(1-x_j\right)n_j$ and $E_3 = \sum_{j=1}^{l}\beta\left[x_j\right]x_j\,n_j\right)$

Invasion fitness

$$s_{\underline{E}}(y) = \langle E_1\rangle\,(1-y) - \delta + \langle E_2\rangle\,\beta[y]\,y - \langle E_3\rangle\,(1-y)$$

MONOMORPHIC RESIDENT POPULATION

Monomorphic resident population dynamics:
(R = resource density; n = resident population density)

```
dLogR = r - r R / k - α (1 - x) n;

dLogn = ϵ α (1 - x) R - δ + γ β[x] x (1 - x) n - (1 - x) β[x] x n;
```

Monomorphic resident population equilibrium:

```
Solve[{0 == r - r R / k - α (1 - x) n,
  0 == ϵ α (1 - x) R - δ + γ β[x] x (1 - x) n - (1 - x) β[x] x n}, {R, n}]
```

$$\left\{\left\{R \to -\frac{k\,\alpha\,\delta + k\,r\,x\,\beta[x] - k\,r\,x\,\gamma\,\beta[x]}{-k\,\alpha^2\,\epsilon + k\,x\,\alpha^2\,\epsilon - r\,x\,\beta[x] + r\,x\,\gamma\,\beta[x]},\right.\right.$$

$$\left.\left. n \to -\frac{r\,\delta - k\,r\,\alpha\,\epsilon + k\,r\,x\,\alpha\,\epsilon}{(-1+x)\left(-k\,\alpha^2\,\epsilon + k\,x\,\alpha^2\,\epsilon - r\,x\,\beta[x] + r\,x\,\gamma\,\beta[x]\right)}\right\}\right\}$$

$$R[x\_] := -\frac{k\,(\alpha\,\delta - r\,x\,(-1+\gamma)\,\beta[x])}{k\,(-1+x)\,\alpha^2\,\epsilon + r\,x\,(-1+\gamma)\,\beta[x]};$$

$$n[x\_] := -\frac{r\,(\delta + k\,(-1+x)\,\alpha\,\epsilon)}{(-1+x)\,\left(k\,(-1+x)\,\alpha^2\,\epsilon + r\,x\,(-1+\gamma)\,\beta[x]\right)};$$

Invasion fitness and its derivatives:

```
smo[x_, y_] := ϵ α (1 - y) R[x] - δ + γ β[y] y (1 - x) n[x] -
    (1 - y) β[x] x n[x];

dsmo[x_] := (∂y smo[x, y]) /. {y → x};
ddsmo[x_] := (∂y ∂y smo[x, y]) /. {y → x};
```

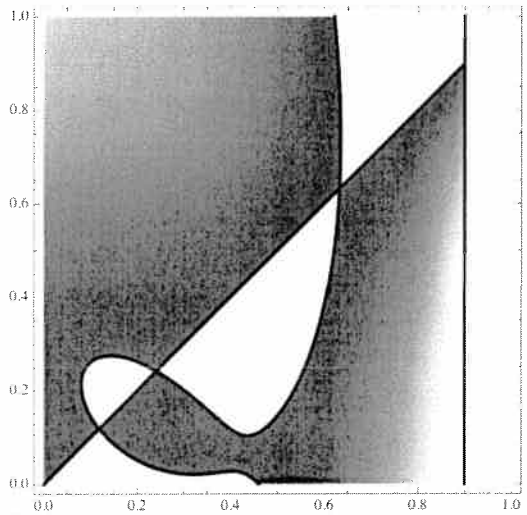
Default parameter values and functions:

```
α = 1; γ = 0.2; δ = 0.1; ϵ = 0.1; r = 1; k = 10;

β[x_] := 2 - 9 (0.03 + x)^p (1 - x)^q; p = 1;
q = 2.5;
```

Pairwise invadability plot (PIP):

```
PIPbnd = ContourPlot[If[n[x] > 0, smo[x, y]], {x, 0, 1}, {y, 0, 1},
    Contours → {0}, ContourStyle → {Black, Thick},
    ContourShading → False, PlotPoints → 100];

PIPint = DensityPlot[If[smo[x, y] > 0 && n[x] > 0, smo[x, y]],
    {x, 0, 1}, {y, 0, 1}, PlotPoints → 50];

nPos = ContourPlot[n[x], {x, 0, 1}, {y, 0, 1}, Contours → {0},
    ContourStyle → {Black, Thick}, ContourShading → False,
    PlotPoints → 10];
```

```
Show[PIPint, PIPbnd, nPos]
```



Canonical equation:

Mutation rate:

```
μ[x_] := 1;
```

Standard deviation of the mutation step distribution:

```
σ[x_] := 0.04 x (1 - x);
(* note that the mutation varance becomes zero st the
 boundary of the strategy space [0,1] *)
```

Deterministic drift:

$$\text{drift}_{mo}[x\_] := \frac{1}{2}\,\mu[x]\,\sigma[x]^2\,n[x]\,\text{ds}_{mo}[x];$$

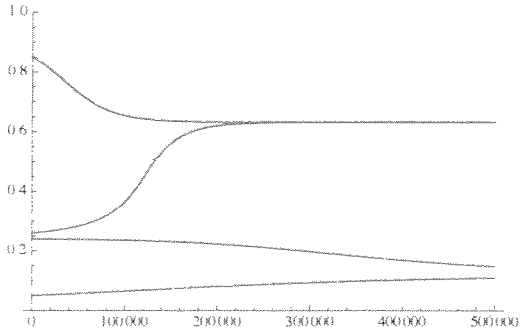
Deterministic orbit (Euler method):

```
x0 = {0.05, 0.24, 0.26, 0.85}; (* starting points *)
t0 = 0;(* start time *)
t∞ = 500 000;(* stop time *)
Δt = 100;(* integration time step *)

noofStarts = Length[x0];

data = {};
For[i = 1, i ≤ noofStarts, i++,
  x = x0[[i]];
  t = t0;
  While[t ≤ t∞ && n[x] > 0,
   data = Join[data, {{t, x}}];
   x = x + Δt drift_mo[x];
   t = t + Δt;
  ];
 ];
```

Deterministic orbits:

```
CEorbit = ListPlot[data, PlotStyle → {Red, PointSize[0.001]},
   Joined → False, AxesOrigin → {0, 0}, PlotRange → {0, 1}];
```

```
Show[CEorbit]
```



Stochastic differential equation (Ito):

Third absolute moment of the mutation step distribution, which is assumed to be Gaussian:

```
θ[x_] := 2 σ[x]^3 Sqrt[2/π];
```

Diffusion coefficient:

```
diff_mo[x_] := 1/2 μ[x] θ[x] n[x] Abs[ds_mo[x]];
```

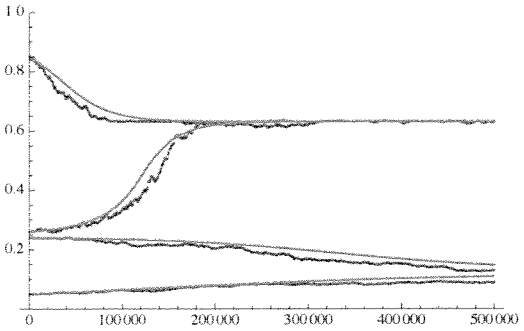
Single stochastic orbit per starting point (Euler metod):

```
x0 = {0.05, 0.24, 0.26, 0.85}; (* starting points *)
t0 = 0;(* start time *)
t∞ = 500 000;(* stop time *)
Δt = 500;(* integration time step *)

noofStarts = Length[x0];

data = {};
For[i = 1, i ≤ noofStarts, i++,
  x = x0[[i]];
  t = t0;
  While[t ≤ t∞,
   data = Join[data, {{t, x}}];
   z = RandomReal[NormalDistribution[0, 1]];
   x = x + Δt drift_mo[x] + z Sqrt[Δt diff_mo[x]];
   t = t + Δt;
  ];
 ];
```

Stochastic orbits:

```
SDEorbit = ListPlot[data, PlotStyle → PointSize[0.005],
   Joined → False, AxesOrigin → {0, 0}, PlotRange → {0, 1}];
```

```
Show[SDEorbit, CEorbit]
```



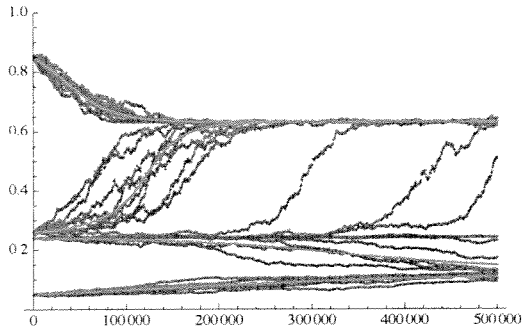
Multiple stochastic orbits for each starting point (Euler metod):

```
x0 = {0.05, 0.24, 0.26, 0.85}; (* starting points *)
t0 = 0;(* start time *)
t∞ = 500 000;(* stop time *)
Δt = 500;(* integration time step *)

noofStarts = Length[x0];
sampleSize = 10;

data = {};
For[i = 1, i ≤ noofStarts, i++,
  For[j = 1, j ≤ sampleSize, j++,
    x = x0[[i]];
    t = t0;
    While[t ≤ t∞,
     data = Join[data, {{t, x}}];
     z = RandomReal[NormalDistribution[0, 1]];
     x = x + Δt drift_mo[x] + z √(Δt diff_mo[x]) ;
     t = t + Δt;
    ];
   ];
 ];
```

Stochastic orbits (blue) and deterministic orbits (red):

```
SDEorbit = ListPlot[data, PlotStyle → PointSize[0.005],
   Joined → False, AxesOrigin → {0, 0}, PlotRange → {0, 1}];
```

```
Show[SDEorbit, CEorbit]
```



DIMORPHIC RESIDENT POPULATION

Reset:

```
Clear[α, β, γ, δ, ϵ, r, k, x];
```

Dimorphic resident population equilibrium:

```
equ =
  {0 == r - r R / k - α ((1 - x1) n1 + (1 - x2) n2),
   0 == ϵ α (1 - x1) R - δ + γ β[x1] x1 ((1 - x1) n1 + (1 - x2) n2) -
     (1 - x1) (β[x1] x1 n1 + β[x2] x2 n2),
   0 == ϵ α (1 - x2) R - δ + γ β[x2] x2 ((1 - x1) n1 + (1 - x2) n2) -
     (1 - x2) (β[x1] x1 n1 + β[x2] x2 n2)};
var = {R, n1, n2};
Solve[equ, var];
Simplify[%]
```

$$\left\{\left\{R \to \frac{k\,((x1-x2)\,\alpha\,\delta + r\,x1\,(-1+x2)\,\gamma\,\beta[x1] - r\,(-1+x1)\,x2\,\gamma\,\beta[x2])}{r\,\gamma\,(x1\,(-1+x2)\,\beta[x1] - (-1+x1)\,x2\,\beta[x2])},\ n1 \to \right.\right.$$
$$\left(k\,(x1-x2)\,(-1+x2)\,\alpha^2\,\delta\,\epsilon + r\,x1\,(-1+x2)\,\gamma\,(\delta + k\,(-1+x2)\,\alpha\,\epsilon)\,\beta[x1] + \right.$$
$$r\,x2\,(\gamma\,(\delta - k\,\alpha\,\epsilon) + x2\,(\delta - \gamma\,\delta + k\,\alpha\,\gamma\,\epsilon) - x1\,(\delta + k\,(-1+x2)\,\alpha\,\gamma\,\epsilon))$$
$$\left.\beta[x2]\right) \Big/ \left(r\,\gamma\,(x1\,(-1+x2)\,\beta[x1] - (-1+x1)\,x2\,\beta[x2])^2\right),$$
$$n2 \to \left(r\,x1\,(-x2\,\delta + \gamma\,\delta - k\,\alpha\,\gamma\,\epsilon + k\,x2\,\alpha\,\gamma\,\epsilon + x1\,(\delta - \gamma\,\delta - k\,(-1+x2)\,\alpha\,\gamma\,\epsilon))\right.$$
$$\beta[x1] -$$
$$\left.(-1+x1)\,\left(k\,(x1-x2)\,\alpha^2\,\delta\,\epsilon - r\,x2\,\gamma\,(\delta + k\,(-1+x1)\,\alpha\,\epsilon)\,\beta[x2]\right)\right) \Big/$$
$$\left.\left.\left(r\,\gamma\,(x1\,(-1+x2)\,\beta[x1] - (-1+x1)\,x2\,\beta[x2])^2\right)\right\}\right\}$$

```
R[{x1_, x2_}] :=
  (k ((x1 - x2) α δ + r x1 (-1 + x2) γ β[x1] - r (-1 + x1) x2 γ β[x2])) /
   (r γ (x1 (-1 + x2) β[x1] - (-1 + x1) x2 β[x2]));

n1[{x1_, x2_}] :=
  (k (x1 - x2) (-1 + x2) α^2 δ ϵ + r x1 (-1 + x2) γ (δ + k (-1 + x2) α ϵ) β[x1] +
     r x2 (γ (δ - k α ϵ) + x2 (δ - γ δ + k α γ ϵ) - x1 (δ + k (-1 + x2) α γ ϵ))
      β[x2]) / (r γ (x1 (-1 + x2) β[x1] - (-1 + x1) x2 β[x2])^2);

n2[{x1_, x2_}] :=
  (r x1 (-x2 δ + γ δ - k α γ ϵ + k x2 α γ ϵ + x1 (δ - γ δ - k (-1 + x2) α γ ϵ))
      β[x1] -
     (-1 + x1) (k (x1 - x2) α^2 δ ϵ - r x2 γ (δ + k (-1 + x1) α ϵ) β[x2])) /
   (r γ (x1 (-1 + x2) β[x1] - (-1 + x1) x2 β[x2])^2);
```

Dimorphic invasion fitness and derivatives:

```
s_di[{x1_, x2_}, y_] :=
  ϵ α (1 - y) R[{x1,x2}] - δ +
   γ β[y] y ((1 - x1) n1[{x1, x2}] + (1 - x2) n2[{x1, x2}]) -
   (1 - y) (β[x1] x1 n1[{x1, x2}] + β[x2] x2 n2[{x1, x2}]);

x1ds_di[{x1_, x2_}] := (∂_y s_di[{x1, x2}, y]) /. {y → x1};
x2ds_di[{x1_, x2_}] := (∂_y s_di[{x1, x2}, y]) /. {y → x2};
x1dds_di[{x1_, x2_}] := (∂_y ∂_y s_di[{x1, x2}, y]) /. {y → x1};
x2dds_di[{x1_, x2_}] := (∂_y ∂_y s_di[{x1, x2}, y]) /. {y → x2};
```

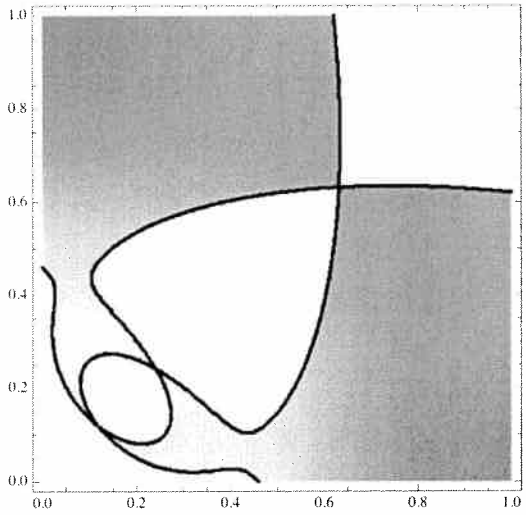
Default parameter values and functions :

```
α = 1; γ = 0.2; δ = 0.1; ϵ = 0.1; r = 1; k = 10;
p = 1; q = 2.5;
β[x_] := 2 - 9 (0.03 + x)^p (1 - x)^q;
```

Coexistence plot:

```
coexBnd = ContourPlot[If[n1[{x1, x2}] > 0 && n2[{x1, x2}] > 0, 1, -1],
   {x1, 0, 1}, {x2, 0, 1}, Contours → {0},
   ContourStyle → {Black, Thick}, ContourShading → False,
   PlotPoints → 60];

coexInt = DensityPlot[If[n1[{x1, x2}] > 0 && n2[{x1, x2}] > 0,
    n1[{x1, x2}] + n2[{x1, x2}]], {x1, 0, 1}, {x2, 0, 1},
   PlotPoints → 60];
```

```
Show[coexInt, coexBnd]
```



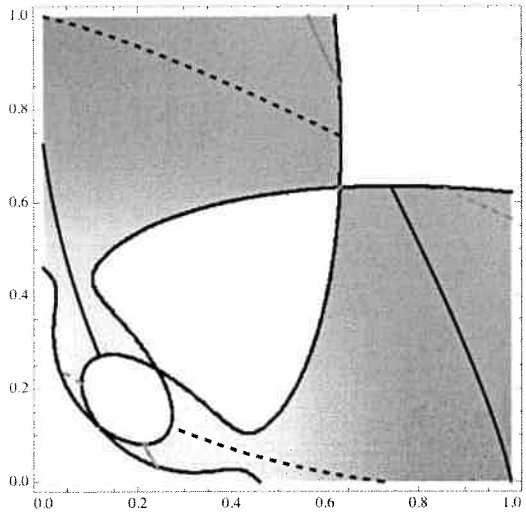
Isocline plot:

(solid = x1-isocline; dashed = x2-isocline; black = evolutionarily stable; red = not evolutionarily stable)

```
iso1ES =
  ContourPlot[
   If[n1[{x1, x2}] > 0 && n2[{x1, x2}] > 0 && x1ddsdi[{x1, x2}] ≤ 0,
    x1dsdi[{x1, x2}]], {x1, 0, 1}, {x2, 0, 1}, Contours → {0},
   ContourShading → False, ContourStyle → {Black, Thick},
   PlotPoints → 30];

iso1NES =
  ContourPlot[
   If[n1[{x1, x2}] > 0 && n2[{x1, x2}] > 0 && x1ddsdi[{x1, x2}] > 0,
    x1dsdi[{x1, x2}]], {x1, 0, 1}, {x2, 0, 1}, Contours → {0},
   ContourShading → False, ContourStyle → {Red, Thick},
   PlotPoints → 30];

iso2ES =
  ContourPlot[
   If[n1[{x1, x2}] > 0 && n2[{x1, x2}] > 0 && x2ddsdi[{x1, x2}] ≤ 0,
    x2dsdi[{x1, x2}]], {x1, 0, 1}, {x2, 0, 1}, Contours → {0},
   ContourShading → False, ContourStyle → {Black, Thick, Dashed},
   PlotPoints → 30];

iso2NES =
  ContourPlot[
   If[n1[{x1, x2}] > 0 && n2[{x1, x2}] > 0 && x2ddsdi[{x1, x2}] > 0,
    x2dsdi[{x1, x2}]], {x1, 0, 1}, {x2, 0, 1}, Contours → {0},
   ContourShading → False, ContourStyle → {Red, Thick, Dashed},
   PlotPoints → 30];
```

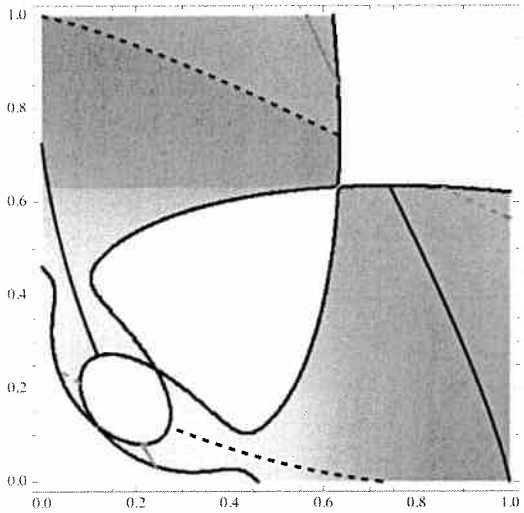
Isoclines (solid = x1-isocline; dashed = x2-isocline; black = ES; red = NES):

```
Show[coexInt, coexBnd, iso1ES, iso1NES, iso2ES, iso2NES]
```



Differential inclusion:

Reachability set:



Canonical equation:

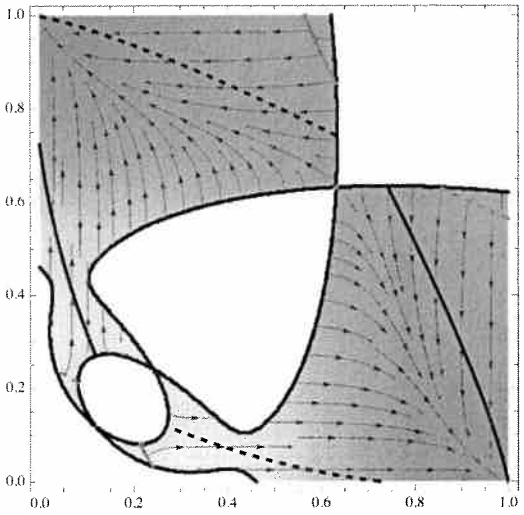
Deterministic drift:

```
drift_di[{x1_, x2_}] :=
  {1/2 μ[x1] σ[x1]^2 n1[{x1, x2}] x1ds_di[{x1, x2}],
   1/2 μ[x2] σ[x2]^2 n2[{x1, x2}] x2ds_di[{x1, x2}]};
```

Stream plot:

```
CEstream =
  StreamPlot[If[n1[{x1, x2}] > 0 && n2[{x1, x2}] > 0,
    drift_di[{x1, x2}], {0, 0}], {x1, 0, 1}, {x2, 0, 1}];
```

```
Show[coexInt, coexBnd, iso1ES, iso1NES, iso2ES, iso2NES,
 CEstream]
```



Stochastic orbits:

Diffusion coefficient:

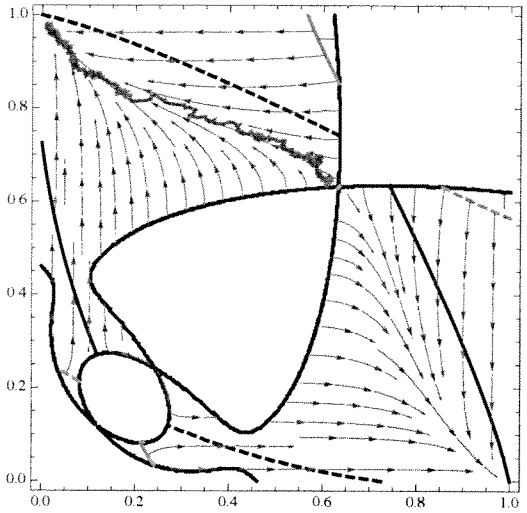
$$\text{diff}_{di}[\{x1\_, x2\_\}] :=$$
$$\left\{\frac{1}{2}\,\mu[x1]\,\theta[x1]\,n1[\{x1, x2\}]\,\text{Abs}[x1ds_{di}[\{x1, x2\}]],\right.$$
$$\left.\frac{1}{2}\,\mu[x2]\,\theta[x2]\,n2[\{x1, x2\}]\,\text{Abs}[x2ds_{di}[\{x1, x2\}]]\right\};$$

Single stochastic orbit (Euler metod):

```
x0 = {0.615, 0.645}; (* starting point *)
t0 = 0;(* start time *)
t∞ = 15 000 000;(* stop time *)
Δt = 5000;(* integration time step *)

data = {};
x = x0;
t = t0;
While[t ≤ t∞,
  data = Join[data, {Append[x, t]}];
  z = RandomReal[NormalDistribution[0, 1], 2];
  x = x + Δt driftdi[x] + z √(Δt diffdi[x]) ;
  t = t + Δt;
 ];
```

```
SDEorbit = ListPlot[data[[All, {1, 2}]], PlotStyle → {Thick},
   Joined → True];
```

```
Show[coexBnd, iso1ES, iso1NES, iso2ES, iso2NES, CEstream,
 SDEorbit]
```



EVOLUTIONARY TREE

Deterministic approximation of the evolutionary tree using the canonical equation (CE).

```
x0 = 0.26; (* starting point of tree *)
t0 = 0; (* start time *)
t∞ = 5 000 000; (* stop time *)
Δt = 500; (* integration time step *)

data = {};

(* monomorphic part of the deterministic tree;
notice the extra stopping condition *)
x = x0;
t = t0;
While[t ≤ t∞ && dsmo[x - 0.01 σ[x]] dsmo[x + 0.01 σ[x]] > 0,
  data = Join[data, {{x, t}}];
  x = x + Δt driftmo[x];
  t = t + Δt;
 ];

(* dimorphic part of the deterministic tree *)
x = {x - 0.1 σ[x], x + 0.1 σ[x]};
While[t ≤ t∞,
  data = Join[data, {{x[[1]], t}, {x[[2]], t}}];
  x = x + Δt driftdi[x];
  t = t + Δt;
 ];
```

```
CEtree = ListPlot[data, PlotStyle → {Red, PointSize[0.005]},
   Joined → False, AxesOrigin → {0, 0}, PlotRange → {{0, 1}, All}];
```

Show[CEtree]