Figures Bistab c / sigma

```
params[ss_, mm_] = {c → mm, mu → 1, β0 → 10, k → 0.01, sig → ss, eps → 1, rm → 2};

β[a_] = β0 a / (1 + a); (* parasite trade-off*)

rS[ga_] = rm / (1 + c ga) ; (*susceptible host trade-off*)

rI[ga_] = eps rS[ga];(*infected host trade-off*)

getres[aa_, gg_] =
 Solve[{0 == (rS[gg] x + rI[gg] y) (1 - k (x + y)) - (mu + (β[aa] y)) x + gg y,
     0 == β[aa] x y - (mu + aa + gg) y}, {x, y}][[4]] //
  Simplify; (* epidemiological system*)

para[a_, ga_] = Block[{}, res = getres[a, ga];
  Seq = x /. res;
  Ieq = y /. res;
  β'[a] - β[a] / (mu + a + ga + sig β[a] Ieq)]; (* parasite selection gradient*)

hote[a_, ga_] = Block[{}, res = getres[a, ga];
   Seq = x /. res;
   Ieq = y /. res;
   rS'[ga] (1 - k (Seq + Ieq)) + (rI'[ga] (1 - k (Seq + Ieq)) β[a] Ieq) / (mu + a + ga) +
    (rS[ga] (1 - k (Seq + Ieq)) - mu) / (mu + a + ga)] // Simplify;
(*host selection gradient*)

getcoess[sig_, c_] := Solve[
  {0 == para[a, ga], 0 == hote[a, ga]} /. params[sig, c], {a, ga}](*find coess*)

(*getcoess[0,0]{{a→0.0206254275068642`,ga→0.00042540825984091035`},
  {a→2.1614125639307775`,ga→4.671704271517817`},
  {a→18.692962008562358`,ga→349.4268286535557`},
  {a→30.622776601683793`,ga→937.7544467966325`}}

(*list=Table[{"sig = "<>ToString[sig],"c = "<>ToString[c],getcoess[sig,mu]},
   {sig,0,1.0,0.1},{c,0,2.0,0.1}];*)

getcoess[0.25, 0.1] // Timing

{3.712824, {{a → -0.53964, ga → 51.5116}}}

list = ParallelTable[{sig, c, getcoess[sig, c]},
   {sig, 0.0, 0.25, 0.0005}, {c, 0.0, 0.2, 0.001}];

Export["listbrutc.csv", list, "CSV"];
```

```
dataAll = {};
(*dataAll2={};*)
For[i = 1, i ≤ Dimensions[list][[2]], i++,
 For[j = 1, j ≤ Dimensions[list][[1]], j++,
  mysol = Select[{a, ga} /. list[[j, i, 3]], Element[#[[1]], Reals] &&
      #[[1]] > 0 && Element[#[[2]], Reals] && #[[2]] > 0 &];
  AppendTo[dataAll, {list[[j, i, 2]], list[[j, i, 1]], Length[mysol]}];
  (*AppendTo[dataAll2,{list[[j,i,2]],list[[j,i,1]],mysol}];*)
  Export["datacsigma.csv", dataAll, "CSV"]
  (*Export["datacsigmadat.csv",dataAll2,"CSV"];*)
 ]
]
```

Part::partd : Part specification a[[1]] is longer than depth of object. ≫

Part::partd : Part specification a[[1]] is longer than depth of object. ≫

Part::partd : Part specification a[[2]] is longer than depth of object. ≫

General::stop : Further output of Part::partd will be suppressed during this calculation. ≫

```
Export["datacsigma13.csv", dataAll, "CSV"];
```

```
dataEqIhi = {}; dataEqShi = {}; dataEqIlow = {}; dataEqSlow = {};
For[i = 1, i ≤ Dimensions[list][[2]], i++,
 For[j = 1, j ≤ Dimensions[list][[1]], j++,
  mysol = Select[{a, ga} /. list[[j, i, 3]], Element[#[[1]], Reals] &&
      #[[1]] > 0 && Element[#[[2]], Reals] && #[[2]] > 0 &];

  For[l = 1, l ≤ Length[mysol], l++,
   findEqI = Ieq /. res /. params[list[[j, i, 1]], list[[j, i, 2]]] /.
      a → mysol[[l, 1]] /. ga → mysol[[l, 2]];
   findEqS = Seq /. res /. params[list[[j, i, 1]], list[[j, i, 2]]] /.
      a → mysol[[l, 1]] /. ga → mysol[[l, 2]];
   If[findEqI > 0 && findEqI < 0.5, AppendTo[dataEqIlow,
     {list[[j, i, 2]], list[[j, i, 1]], findEqI}]];
   If[findEqI > 0 && findEqI > 0.5, AppendTo[dataEqIhi,
     {list[[j, i, 2]], list[[j, i, 1]], findEqI}]];
   If[findEqS < 100 && findEqS < 0.51, AppendTo[dataEqSlow,
     {list[[j, i, 2]], list[[j, i, 1]], findEqS}]];
   If[findEqS < 100 && findEqS > 0.51, AppendTo[dataEqShi,
     {list[[j, i, 2]], list[[j, i, 1]], findEqS}]];
  ];
 ]
]

Export["datacsigmaEqSlow.csv", dataEqSlow, "CSV"];
Export["datacsigmaEqShi.csv", dataEqShi, "CSV"];
Export["datacsigmaEqIlow.csv", dataEqIlow, "CSV"];
Export["datacsigmaEqIhi.csv", dataEqIhi, "CSV"];
```

Part::partd : Part specification a⟦1⟧ is longer than depth of object. ≫

Part::partd : Part specification a⟦1⟧ is longer than depth of object. ≫

Part::partd : Part specification a⟦2⟧ is longer than depth of object. ≫

General::stop : Further output of Part::partd will be suppressed during this calculation. ≫

```
dataEqI = {};
dataEqS = {};
dataAllEqI = {};
dataAllEqS = {};
dataEqIc = {};
dataEqSc = {};
dataAll = {};
dataAll2 = {};
For[i = 1, i ≤ Dimensions[list][[2]], i++,
 For[j = 1, j ≤ Dimensions[list][[1]], j++,
  mysol = Select[{a, ga} /. list[[j, i, 3]], Element[#[[1]], Reals] &&
      #[[1]] > 0 && Element[#[[2]], Reals] && #[[2]] > 0 &];

  For[l = 1, l ≤ Length[mysol], l++,
   findEqI = Ieq /. res /. params[list[[j, i, 1]], list[[j, i, 2]]] /.
      a → mysol[[l, 1]] /. ga → mysol[[l, 2]];
   findEqS = Seq /. res /. params[list[[j, i, 1]], list[[j, i, 2]]] /.
      a → mysol[[l, 1]] /. ga → mysol[[l, 2]];
   AppendTo[dataEqI, {list[[j, i, 2]], list[[j, i, 1]], findEqI}];
   AppendTo[dataEqS, {list[[j, i, 2]], list[[j, i, 1]], findEqS}];
  ];

  EqI = Select[dataEqI[[All, 3]], # > 0 &] ;
  EqS = Select[dataEqS[[All, 3]], # < 100 &] ;
  AppendTo[dataAllEqS, {list[[j, i, 2]], list[[j, i, 1]], EqS}];
  AppendTo[dataAllEqI, {list[[j, i, 2]], list[[j, i, 1]], EqI}];

  AppendTo[dataAll, {list[[j, i, 2]], list[[j, i, 1]], Length[mysol]}];
  AppendTo[dataAll2, {list[[j, i, 2]], list[[j, i, 1]], mysol}];

  AppendTo[dataEqSc, {list[[j, i, 2]], list[[j, i, 1]], Length[EqS]}];
  AppendTo[dataEqIc, {list[[j, i, 2]], list[[j, i, 1]], Length[EqI]}];

  dataEqI = {}; dataEqS = {};
 ]
]

Export["datacsigmaEqS.csv", dataEqSc, "CSV"];
Export["datacsigmaEqI.csv", dataEqIc, "CSV"];
Export["datacsigmaEqS-brut.csv", dataAllEqS, "CSV"];
Export["datacsigmaEqI-brut.csv", dataAllEqI, "CSV"];
```

Part::partd : Part specification a[[1]] is longer than depth of object. >>

Part::partd : Part specification a[[1]] is longer than depth of object. >>

Part::partd : Part specification a[[2]] is longer than depth of object. >>

General::stop : Further output of Part::partd will be suppressed during this calculation. >>