# Are equilibration fix points global optima? Parameter-Robustness of our simple RE model

This notebook checks whether fixed points are global optima in an ensemble where the weights of the achievemnt function have been varied randomly

In[183]:=
```
SetDirectory[$HomeDirectory];
If[! MemberQ[$Path, #], AppendTo[$Path, #]] &[
  FileNameJoin[{"git", "DialecticalStructures"}]];
If[! MemberQ[$Path, #], AppendTo[$Path, #]] &[
  FileNameJoin[{"git", "ReflectiveEquilibrium"}]];
<< DialecticalStructures`BasicTDS`;
<< DialecticalStructures`InductiveReasoning`;
<< DialecticalStructures`CoherenceMeasures`;
<< DialecticalStructures`PositionsAnalytics`;
<< ReflectiveEquilibrium`ReflectiveEquilibrium`;
```

In[ ]:=

---

## Setting up the scene

In[ ]:=
```
ensembleDir = "2020_06_25-0001";
```

Get data from first case.

In[ ]:=
```
data = Get[FileNameJoin[{
      NotebookDirectory[],
      "results",
      ensembleDir,
      ensembleDir <> "#" <> IntegerString[1, 10, 6] <> ".m"
     }]];
senIDs = Cases[data, {"senIDs", _}][[1, 2]];
tau = Cases[data, {"tau", _}][[1, 2]];
param = Cases[data, {"parameters", _}][[1, 2]];
```

Get initial commitments for all cases.

In[•]:=
```
initialComs = Module[{
   n
  },
  n = Length[FileNames[ensembleDir <> "#*.m", FileNameJoin[{
        NotebookDirectory[],
        "results",
        ensembleDir}]]];
  Table[
   Lookup[
    Cases[
      Get[FileNameJoin[{
         NotebookDirectory[],
         "results",
         ensembleDir,
         ensembleDir <> "#" <> IntegerString[i, 10, 6] <> ".m"
        }]],
      {"posEvolution", _}][[1, 2, 1]],
    "COM"],
   {i, n}
  ]
 ]
```

Out[•]= {1090, 121, 118, 1333, 121, 121, 121, 1090, 1333, 1333, 118, 1090, 118, 1333, 118,
1090, 121, 121, 118, 1333, 1333, 1333, 121, 1333, 121, 1333, 1333, 121, 1333,
1333, 1333, 118, 118, 1333, 118, 118, 118, 121, 118, 118, 1333, 118, 118, 121,
118, 1090, 1333, 121, 121, 118, 121, 1090, 1333, 118, 1090, 118, 118, 121,
118, 118, 121, 1333, 121, 1090, 1090, 1090, 1090, 1090, 121, 121, 1333, 1333,
1333, 1090, 118, 1333, 118, 118, 121, 121, 121, 1090, 118, 1090, 118, 1333,
121, 1090, 1090, 1333, 1090, 121, 1333, 1090, 121, 121, 121, 1090, 118, 118,
121, 1090, 1333, 118, 118, 1333, 121, 1090, 118, 118, 121, 1090, 121, 1333,
118, 118, 118, 118, 1333, 1333, 1090, 121, 1333, 1333, 1090, 121, 1333, 118,
1090, 118, 1090, 118, 121, 118, 1090, 1333, 118, 1333, 121, 118, 121, 1333,
118, 121, 1333, 118, 1090, 1333, 121, 1333, 118, 1333, 118, 1333, 1090, 121,
1090, 1333, 1090, 118, 1333, 121, 121, 1090, 121, 118, 1090, 1333, 1333, 121,
121, 121, 1090, 121, 1333, 1333, 1333, 1090, 1333, 1090, 1333, 121, 121, 121,
1333, 1333, 1333, 118, 118, 1333, 1333, 121, 1090, 1090, 121, 1090, 1333,
1333, 1333, 1090, 1090, 121, 121, 121, 118, 1090, 1090, 1333, 121, 118, 1333,
1333, 1333, 121, 1090, 1090, 121, 118, 1090, 1090, 121, 121, 118, 1090, 1333,
118, 118, 1090, 1333, 1090, 118, 1090, 118, 118, 1333, 118, 118, 118, 1090,
121, 1090, 118, 118, 1090, 121, 121, 1333, 118, 1090, 121, 121, 118, 1090,
1333, 1333, 1333, 1333, 121, 121, 121, 121, 1333, 1333, 118, 1333, 121, 118,
1333, 1333, 118, 118, 1090, 118, 118, 1333, 1090, 1333, 1333, 1333, 1090,
1333, 121, 118, 121, 118, 1090, 1090, 1333, 1333, 1090, 121, 121, 1333, 121,
121, 1333, 118, 118, 121, 118, 1333, 118, 1090, 1333, 1090, 1333, 118, 1090,
118, 121, 1333, 121, 1333, 1333, 1333, 121, 1333, 1333, 1333, 121, 118, 118,
1090, 1333, 1333, 121, 1090, 121, 118, 1090, 118, 1333, 118, 1333, 1090, 121,
118, 121, 121, 121, 121, 1333, 121, 118, 1333, 118, 1090, 1090, 121, 1090,
1090, 118, 1333, 1090, 121, 121, 1333, 1333, 118, 1333, 1090, 1090, 118,
1333, 121, 121, 1090, 1333, 1090, 121, 118, 1090, 1090, 121, 1333, 1090, 121,
1333, 121, 1090, 1333, 121, 1333, 1090, 1333, 1090, 1333, 118, 1090, 121,
118, 121, 1333, 118, 121, 1090, 1333, 118, 1090, 1333, 1090, 1333, 121, 1090,
1333, 121, 1333, 1333, 118, 1090, 1090, 121, 118, 118, 121, 121, 1090, 1090,
1090, 118, 118, 1333, 121, 118, 121, 1090, 1090, 1090, 1090, 1333, 118, 121,
1090, 1090, 1090, 121, 121, 1333, 1090, 1090, 1333, 121, 1333, 121, 1333,
118, 1090, 121, 1090, 1090, 1090, 1333, 121, 1333, 1333, 1090, 121, 1090,
1090, 1333, 121, 1090, 118, 1333, 118, 1090, 118, 1090, 118, 1090, 121, 121,
1090, 118, 118, 121, 121, 1090, 1090, 118, 1090, 1090, 1090, 1333, 121, 121,
121, 1090, 1090, 1090, 121, 1333, 118, 1333, 121, 121, 1333, 1333, 1090, 121}

Calculate sigma.

In[•]:=
```
PrintTemporary["Creating sigma..."];
sigma = Lookup[
  param,
  "sigma",
  Sigma[tau, True, senIDs]
  ];
PrintTemporary["...done."];
```

Calculate nPrinciples.

```
In[•]:= PrintTemporary["Creating nPrinciples..."];
        nPrinciples = Lookup[
          param,
          "nPrinciples",
          NPrinciples[sigma, senIDs]
          ];
        PrintTemporary["...done."];
```

Calculate closedPosition.

```
In[•]:= PrintTemporary["Creating closedPositions..."];
        closedPositions =
          DeleteCases[DialecticallyClosedPositions[tau, True, senIDs], 1];
        PrintTemporary["...done."];
```

We say that **P** is the set of all permissible theory-commitment pairs <T,C>, where such a pair is permissible iff C is minimally consistent and T is consistent and closed. (Claus is right in saying that we've defined the notions of theory and commitments such that <T,C>-pairs are permissible.) Furthermore, let **T** be the set of all consistent and closed theories, and **C** be the set of all minimally consistent commitments. **C** is the list **Range[3^7]** (integer-representation of partial positions). **T** is a subset of **C**. **P** is hence a list of integer-pairs.

```
In[•]:= allTCPairs = Tuples[{closedPositions, Range[3^Length[senIDs]]}];
```

We further calculate

- for every T in **T**: Simplicity[T] and store the results in SparseArray **Simp**. (Simplicity[T] is stored in **Simp** at position T.)

```
In[•]:= Simp =
          SparseArray[# -> Simplicity[#, nPrinciples[[#]], senIDs] & /@ closedPositions];
```

- for every C in **C** and for every $C_0^i$ in initialComs: Closeness[C,$C_0$] and store the results in SparseArray **Clos[[i]]**. (Closeness[C,$C_0^i$] is stored in **Clos[[i]]** at position C.)

```
In[•]:= Clos = Map[
           Function[
            initialCom,
            SparseArray[# → Closeness[#, initialCom, senIDs, KeyTake[param,
                  {"ConflictPenalty", "ContractionPenalty", "ExpansionPenalty"}]] &
              /@ Range[3^Length[senIDs]]
            ]
           ],
           initialComs
          ];
```

- for every <T,C> in **P**: Account[T,C] and store the results in SparseArray **Acco**. (Account[T,C] is stored in **Acco** at position {T,C}.)

```
In[•]:= Account = AccountFunction[param];
```

```
In[•]:= Acco = Monitor[
          SparseArray[
           Flatten[Table[
             {closedPositions[[i]], c} →
              Account[c, closedPositions[[i]], sigma, senIDs],
             {i, Length[closedPositions]}, {c, 3^Length[senIDs]}
            ],
           1
          ]
         ],
         ProgressIndicator[i, {1, Length[closedPositions]}]
        ];
```

# Evaluating fixed points

```
In[•]:= WeightAccount[alpha_, beta_] := (alpha * beta)/(alpha + beta - alpha * beta);
        WeightSystematicity[alpha_, beta_] := (beta - alpha * beta)/(alpha + beta - alpha * beta);
        WeightCloseness[alpha_, beta_] :=
          1 - (WeightAccount[alpha, beta] + WeightSystematicity[alpha, beta]);

In[•]:= {WeightAccount[#1, #2], WeightSystematicity[#1, #2], WeightCloseness[#1, #2]} &[
         0.35, 0.75]

Out[•]= {0.313433, 0.58209, 0.104478}

In[•]:= {WeightAccount[#1, #2], WeightSystematicity[#1, #2], WeightCloseness[#1, #2]} &[
         0.4, 0.8]

Out[•]= {0.363636, 0.545455, 0.0909091}

In[•]:= {WeightAccount[#1, #2], WeightSystematicity[#1, #2], WeightCloseness[#1, #2]} &[
         0.45, 0.85]

Out[•]= {0.416894, 0.509537, 0.0735695}

In[•]:=
        AchievementFunction[tcp_, alpha_, beta_, caseindex_] := Module[{wa, ws, wc},
            (*Calculate weights*)
            wa = WeightAccount[alpha, beta];
            ws = WeightSystematicity[alpha, beta];
            wc = WeightCloseness[alpha, beta];
            (*Calculate achievement value*)
            ws * Simp[[First[tcp]]] +
             wc * Clos[[caseindex, Last[tcp]]] + wa * Acco[[First[tcp], Last[tcp]]]
           ];
```

In[•]:=
```
(* TEST *)
AchievementFunction[
 {closedPositions[[4]], 68},
 0.8, 0.4,
 3
]
```

Out[•]=
```
0.888683
```

In[•]:=
```
diagnostics = Module[{
    casedata, initialCom, finalstate, n, allAchievementValues, alpha, beta
   },
   n = Length[FileNames[ensembleDir <> "#*.m", FileNameJoin[{
        NotebookDirectory[],
        "results",
        ensembleDir}]]];
   Table[
    (
     PrintTemporary["Analysing ensemble-member " <> ToString[i]];
     casedata = Get[FileNameJoin[{
         NotebookDirectory[],
         "results",
         ensembleDir,
         ensembleDir <> "#" <> IntegerString[i, 10, 6] <> ".m"
        }]];

     initialCom = Lookup[
       Cases[casedata, {"posEvolution", _}][[1, 2, 1]],
       "COM"];
     PrintTemporary["  initial commitments: " <> ToString[initialCom]];

     finalstate = Lookup[
       Cases[casedata, {"posEvolution", _}][[1, 2, -1]],
       {"THE", "COM"}];

     alpha = Lookup[
       Cases[casedata, {"parameters", _}][[1, 2]],
       "alpha"];
     PrintTemporary["  alpha: " <> ToString[alpha]];

     beta = Lookup[
       Cases[casedata, {"parameters", _}][[1, 2]],
       "beta"];
     PrintTemporary["  beta: " <> ToString[beta]];

     allAchievementValues =
      (AchievementFunction[#, alpha, beta, i] &) /@ allTCPairs;
     (*Return as entry i in diagnostics:*)
```

```
    {
     i,
     First[finalstate] == Last[finalstate],
     Max[allAchievementValues] -
      AchievementFunction[finalstate, alpha, beta, i]
    }
   ),
   {i, n}
  ]
 ];
```

In[•]:=
```
diagnostics
```

Out[•]=
```
{{1, False, 0.}, {2, False, 0.}, {3, True, 0.}, {4, True, 0.}, {5, True, 0.},
 {6, True, 0.}, {7, True, 0.}, {8, True, 0.0038267}, {9, False, 0.}, {10, True, 0.},
 {11, True, 0.}, {12, True, 0.}, {13, True, 0.}, {14, True, 0.00135638},
 {15, True, 0.}, {16, False, 0.}, {17, True, 0.}, {18, True, 0.}, {19, True, 0.},
 {20, True, 0.}, {21, True, 0.}, {22, True, 0.}, {23, True, 0.}, {24, True, 0.},
 {25, True, 0.}, {26, True, 0.}, {27, True, 0.}, {28, True, 0.}, {29, True, 0.},
 {30, True, 0.}, {31, True, 0.00115288}, {32, True, 0.}, {33, True, 0.},
 {34, True, 0.}, {35, True, 0.}, {36, True, 0.}, {37, True, 0.}, {38, False, 0.},
 {39, True, 0.}, {40, True, 0.}, {41, True, 0.}, {42, True, 0.}, {43, True, 0.},
 {44, False, 0.}, {45, True, 0.}, {46, False, 0.}, {47, True, 0.},
 {48, False, 0.}, {49, False, 0.}, {50, True, 0.}, {51, False, 0.},
 {52, False, 0.}, {53, True, 0.00287989}, {54, True, 0.}, {55, True, 0.},
 {56, True, 0.}, {57, True, 0.}, {58, False, 0.000502452}, {59, True, 0.},
 {60, True, 0.}, {61, False, 0.}, {62, True, 0.000983055}, {63, True, 0.},
 {64, False, 0.}, {65, True, 0.022734}, {66, False, 0.}, {67, True, 0.00364146},
 {68, True, 0.00750434}, {69, True, 0.}, {70, False, 0.}, {71, True, 0.},
 {72, True, 0.}, {73, True, 0.}, {74, True, 0.}, {75, True, 0.}, {76, True, 0.},
 {77, True, 0.}, {78, True, 0.}, {79, True, 0.}, {80, True, 0.}, {81, False, 0.},
 {82, True, 0.00660734}, {83, True, 0.}, {84, True, 0.0251748}, {85, True, 0.},
 {86, False, 0.}, {87, False, 0.}, {88, True, 0.00698298}, {89, False, 0.},
 {90, True, 0.}, {91, True, 0.}, {92, True, 0.}, {93, True, 0.}, {94, False, 0.},
 {95, True, 0.}, {96, True, 0.}, {97, True, 0.}, {98, False, 0.}, {99, True, 0.},
 {100, True, 0.}, {101, False, 0.}, {102, False, 0.}, {103, False, 0.},
 {104, True, 0.}, {105, True, 0.}, {106, True, 0.}, {107, True, 0.},
 {108, False, 0.}, {109, True, 0.}, {110, True, 0.}, {111, True, 0.},
 {112, False, 0.}, {113, False, 0.}, {114, True, 0.00502573}, {115, True, 0.},
 {116, True, 0.}, {117, True, 0.}, {118, True, 0.00333228}, {119, False, 0.},
 {120, True, 0.}, {121, False, 0.}, {122, True, 0.}, {123, False, 0.},
 {124, True, 0.}, {125, True, 0.00196731}, {126, True, 0.}, {127, True, 0.00395092},
 {128, True, 0.}, {129, True, 0.}, {130, True, 0.}, {131, True, 0.}, {132, True, 0.},
 {133, False, 0.}, {134, True, 0.}, {135, False, 0.}, {136, False, 0.},
 {137, True, 0.}, {138, True, 0.007204}, {139, False, 0.00402631}, {140, True, 0.},
 {141, True, 0.}, {142, True, 0.}, {143, True, 0.}, {144, True, 0.},
 {145, False, 0.}, {146, True, 0.}, {147, True, 0.0178562}, {148, False, 0.},
 {149, True, 0.}, {150, True, 0.}, {151, True, 0.}, {152, True, 0.},
```

{153, True, 0.}, {154, True, 0.00307557}, {155, False, 0.}, {156, False, 0.},
{157, True, 0.00829114}, {158, True, 0.000669348}, {159, True, 0.},
{160, True, 0.}, {161, False, 0.}, {162, True, 0.}, {163, True, 0.},
{164, True, 0.}, {165, False, 0.}, {166, True, 0.}, {167, True, 0.00181099},
{168, True, 0.}, {169, True, 0.}, {170, True, 0.}, {171, True, 0.}, {172, True, 0.},
{173, True, 0.}, {174, True, 0.}, {175, True, 0.}, {176, False, 0.},
{177, True, 0.00380572}, {178, True, 0.00231014}, {179, True, 0.},
{180, False, 0.}, {181, True, 0.}, {182, False, 0.}, {183, True, 0.},
{184, False, 0.}, {185, False, 0.}, {186, True, 0.}, {187, True, 0.},
{188, True, 0.}, {189, True, 0.}, {190, False, 0.}, {191, True, 0.},
{192, True, 0.}, {193, False, 0.}, {194, False, 0.}, {195, False, 0.000216286},
$\{196, \text{False}, 2.89592 \times 10^{-6}\}$, {197, True, 0.}, {198, False, 0.}, {199, True, 0.},
{200, False, 0.}, {201, True, 0.00711977}, {202, True, 0.00553463},
{203, True, 0.}, {204, True, 0.}, {205, True, 0.}, {206, False, 0.},
{207, True, 0.0267032}, {208, True, 0.}, {209, True, 0.}, {210, True, 0.},
{211, True, 0.}, {212, True, 0.}, {213, True, 0.}, {214, False, 0.},
{215, True, 0.}, {216, False, 0.}, {217, False, 0.}, {218, True, 0.},
{219, True, 0.}, {220, True, 0.}, {221, True, 0.}, {222, True, 0.00461674},
{223, True, 0.}, {224, False, 0.}, {225, True, 0.000857338}, {226, True, 0.},
{227, True, 0.}, {228, False, 0.}, {229, True, 0.}, {230, True, 0.00153571},
{231, True, 0.}, {232, True, 0.000515405}, {233, True, 0.}, {234, True, 0.},
{235, False, 0.}, {236, True, 0.}, {237, True, 0.000691813}, {238, True, 0.},
{239, True, 0.0167288}, {240, False, 0.}, {241, False, 0.}, {242, True, 0.},
{243, True, 0.}, {244, False, 0.}, {245, False, 0.}, {246, False, 0.},
{247, True, 0.}, {248, True, 0.}, {249, True, 0.}, {250, True, 0.},
{251, False, 0.}, {252, True, 0.}, {253, False, 0.}, {254, True, 0.},
{255, True, 0.}, {256, True, 0.}, {257, False, 0.}, {258, True, 0.00201384},
{259, True, 0.}, {260, True, 0.}, {261, False, 0.}, {262, True, 0.00275032},
{263, True, 0.}, {264, True, 0.}, {265, True, 0.}, {266, False, 0.},
{267, True, 0.}, {268, False, 0.}, {269, True, 0.00672847}, {270, True, 0.},
{271, True, 0.}, {272, True, 0.}, {273, True, 0.}, {274, True, 0.},
{275, False, 0.}, {276, True, 0.}, {277, True, 0.}, {278, True, 0.},
{279, False, 0.}, {280, True, 0.0106524}, {281, False, 0.}, {282, False, 0.},
{283, True, 0.}, {284, True, 0.}, {285, True, 0.}, {286, False, 0.},
{287, False, 0.}, {288, True, 0.00127283}, {289, True, 0.}, {290, False, 0.},
{291, True, 0.}, {292, True, 0.}, {293, True, 0.}, {294, False, 0.},
{295, False, 0.}, {296, True, 0.0015031}, {297, True, 0.}, {298, True, 0.},
{299, True, 0.}, {300, True, 0.}, {301, True, 0.}, {302, True, 0.},
{303, False, 0.}, {304, False, 0.}, {305, False, 0.}, {306, True, 0.},
{307, True, 0.}, {308, False, 0.}, {309, True, 0.}, {310, False, 0.0114888},
{311, False, 0.}, {312, False, 0.}, {313, True, 0.}, {314, False, 0.},
{315, True, 0.}, {316, True, 0.}, {317, True, 0.}, {318, True, 0.},
{319, True, 0.}, {320, False, 0.}, {321, True, 0.}, {322, True, 0.},
{323, False, 0.}, {324, True, 0.00275655}, {325, True, 0.}, {326, False, 0.},
{327, True, 0.}, {328, True, 0.}, {329, True, 0.}, {330, True, 0.0036858},
{331, True, 0.}, {332, True, 0.}, {333, True, 0.}, {334, False, 0.},
{335, False, 0.}, {336, False, 0.}, {337, True, 0.}, {338, False, 0.},

{339, False, 0.}, {340, True, 0.}, {341, False, 0.}, {342, True, 0.000649916},
{343, True, 0.}, {344, True, 0.}, {345, False, 0.}, {346, True, 0.},
{347, True, 0.000459191}, {348, False, 0.}, {349, False, 0.}, {350, False, 0.},
{351, False, 0.}, {352, True, 0.}, {353, True, 0.}, {354, True, 0.},
{355, True, 0.}, {356, False, 0.}, {357, False, 0.}, {358, False, 0.},
{359, True, 0.000593212}, {360, True, 0.000565589}, {361, True, 0.},
{362, False, 0.}, {363, True, 0.}, {364, True, 0.}, {365, False, 0.},
{366, True, 0.}, {367, True, 0.}, {368, True, 0.}, {369, True, 0.},
{370, False, 0.0039054}, {371, True, 0.}, {372, True, 0.}, {373, True, 0.0270439},
{374, True, 0.}, {375, False, 0.}, {376, False, 0.}, {377, False, 0.},
{378, True, 0.}, {379, False, 0.}, {380, True, 0.}, {381, True, 0.},
{382, True, 0.}, {383, True, 0.000376108}, {384, False, 0.}, {385, True, 0.},
{386, False, 0.}, {387, True, 0.}, {388, True, 0.}, {389, True, 0.},
{390, False, 0.}, {391, True, 0.}, {392, True, 0.}, {393, True, 0.},
{394, True, 0.}, {395, True, 0.}, {396, False, 0.}, {397, True, 0.},
{398, True, 0.}, {399, False, 0.}, {400, True, 0.00188}, {401, True, 0.},
{402, True, 0.00358088}, {403, False, 0.}, {404, True, 0.00215183},
{405, True, 0.}, {406, False, 0.}, {407, True, 0.}, {408, False, 0.},
{409, True, 0.}, {410, False, 0.}, {411, False, 0.}, {412, True, 0.},
{413, True, 0.}, {414, True, 0.}, {415, False, 0.}, {416, False, 0.00778541},
{417, False, 0.}, {418, True, 0.}, {419, False, 0.}, {420, True, 0.},
{421, True, 0.}, {422, True, 0.}, {423, False, 0.}, {424, True, 0.},
{425, True, 0.}, {426, False, 0.}, {427, False, 0.}, {428, True, 0.},
{429, False, 0.}, {430, True, 0.}, {431, True, 0.}, {432, False, 0.},
{433, False, 0.}, {434, False, 0.}, {435, True, 0.00322}, {436, False, 0.},
{437, True, 0.}, {438, True, 0.}, {439, True, 0.}, {440, True, 0.}, {441, True, 0.},
{442, True, 0.}, {443, False, 0.}, {444, True, 0.}, {445, True, 0.},
{446, True, 0.}, {447, True, 0.}, {448, False, 0.}, {449, False, 0.},
{450, False, 0.}, {451, False, 0.}, {452, True, 0.}, {453, False, 0.},
{454, False, 0.}, {455, True, 0.}, {456, True, 0.}, {457, True, 0.},
{458, False, 0.}, {459, True, 0.}, {460, True, 0.}, {461, True, 0.},
{462, True, 0.}, {463, True, 0.}, {464, True, 0.00102475}, {465, True, 0.},
{466, True, 0.}, {467, True, 0.}, {468, True, 0.}, {469, True, 0.},
{470, False, 0.}, {471, True, 0.}, {472, True, 0.}, {473, True, 0.},
{474, True, 0.}, {475, True, 0.}, {476, False, 0.}, {477, False, 0.},
{478, False, 0.}, {479, False, 0.}, {480, True, 0.}, {481, True, 0.},
{482, False, 0.}, {483, False, 0.}, {484, False, 0.}, {485, False, 0.0016429},
{486, True, 0.}, {487, True, 0.}, {488, True, 0.}, {489, False, 0.},
{490, True, 0.0033136}, {491, False, 0.}, {492, True, 0.00272551},
{493, True, 0.}, {494, True, 0.}, {495, False, 0.}, {496, True, 0.},
{497, False, 0.}, {498, False, 0.}, {499, False, 0.}, {500, False, 0.}}

# Analyse diagnostics

How many final states are global optima (absolute, ratio)?

```
In[•]:= Count[diagnostics, e_ /; (e[[3]] == 0)]
Out[•]= 439

In[•]:= N[Count[diagnostics, e_ /; (e[[3]] == 0)] / Length@diagnostics]
Out[•]= 0.878
```

How many final states are full RE states (absolute, ratio)?

```
In[•]:= Count[diagnostics, e_ /; (e[[3]] == 0) && e[[2]]]
Out[•]= 289

In[•]:= N[Count[diagnostics, e_ /; (e[[3]] == 0) && e[[2]]] / Length@diagnostics]
Out[•]= 0.578
```

How many of the global optima are full RE states?

```
In[•]:= N[Count[diagnostics, e_ /; (e[[3]] == 0) && e[[2]]] /
          Count[diagnostics, e_ /; (e[[3]] == 0)]]
Out[•]= 0.658314
```

Alpha/beta values that yield fixed points that are no optima

In[•]:=
```
(
  Lookup[
    Cases[
      Get[FileNameJoin[{
          NotebookDirectory[],
          "results",
          ensembleDir,
          ensembleDir <> "#" <> IntegerString[#, 10, 6] <> ".m"
        }]],
      {"parameters", _}][[1, 2]],
    {"alpha", "beta"}] &
  ) /@ Cases[diagnostics, e_ /; (e[[3]] != 0)][[All, 1]]
```

Out[•]=
```
{{0.611781, 0.90668}, {0.599839, 0.657184},
 {0.845875, 0.901509}, {0.814728, 0.954657},
 {0.925996, 0.025199}, {0.552157, 0.591515}, {0.908832, 0.883413},
 {0.930125, 0.557585}, {0.952517, 0.790852}, {0.54865, 0.934608},
 {0.901092, 0.889485}, {0.547763, 0.943285}, {0.744138, 0.989753},
 {0.977511, 0.587197}, {0.887016, 0.107306}, {0.645537, 0.82728},
 {0.52994, 0.859647}, {0.965531, 0.458517}, {0.9286, 0.898796},
 {0.6873, 0.829992}, {0.970634, 0.975537}, {0.586447, 0.614009},
 {0.643676, 0.881376}, {0.650105, 0.825179}, {0.896564, 0.157768},
 {0.994953, 0.170713}, {0.271374, 0.27144}, {0.894058, 0.473949},
 {0.976909, 0.957823}, {0.901346, 0.92391}, {0.981345, 0.989705},
 {0.773179, 0.81341}, {0.66264, 0.886034}, {0.924561, 0.113834},
 {0.982164, 0.152807}, {0.911913, 0.767892}, {0.986336, 0.58619},
 {0.512376, 0.622323}, {0.59321, 0.910963}, {0.901356, 0.588891},
 {0.907153, 0.969344}, {0.566474, 0.628257}, {0.900117, 0.434084},
 {0.584062, 0.702414}, {0.804093, 0.994052}, {0.687529, 0.716555},
 {0.925757, 0.111858}, {0.972607, 0.084451}, {0.707279, 0.732825},
 {0.898266, 0.140121}, {0.87715, 0.797582}, {0.62264, 0.638556},
 {0.766055, 0.855051}, {0.806607, 0.98144}, {0.947817, 0.607373},
 {0.901829, 0.295271}, {0.873147, 0.136451}, {0.893524, 0.943434},
 {0.985078, 0.436679}, {0.597048, 0.886979}, {0.737011, 0.865849}}
```