Bistable motif: detailed balancing and its consequences

Finding the condition of multistationarity

We consider the following reactions:

$K + S \leftrightharpoons KS \rightarrow K + S_p$
$K^* + S \leftrightharpoons K^* S \rightarrow K^* + S_p$
$S_p \rightarrow S$
$K \leftrightharpoons K^*$
$KS \leftrightharpoons K^* S$

The species of the system are:

$\{S, S_p, K, K^*, KS, K^* S\}$

In total, there are 11 reations and 6 species.

We firstly construct the ordinary differential equations based on mass-action kinetics. Then compute the determinant of Jacobian, using the solution at critical point (steady state) to calculate the determinant. The (necessary) condition for multistationarity is to make determinant equal to zero (non-zero determinant implys injectivity).

```
ClearAll["Global`*"];
A = Table[0, {11}, {6}];
A[[1]][[1]] = -1;
A[[1]][[3]] = -1;
A[[1]][[5]] = 1;
A[[2]] = -A[[1]];
A[[3]][[3]] = 1; A[[3]][[2]] = 1; A[[3]][[5]] = -1;
A[[4]][[1]] = -1;
A[[4]][[4]] = -1;
A[[4]][[6]] = 1;
A[[5]] = -A[[4]];
A[[6]][[4]] = 1;
A[[6]][[2]] = 1;
A[[6]][[6]] = -1;
A[[7]][[2]] = -1;
A[[7]][[1]] = 1;
A[[8]][[3]] = -1; A[[8]][[4]] = 1; A[[9]] = -A[[8]];
A[[10]][[5]] = -1; A[[10]][[6]] = 1; A[[11]] = -A[[10]];
stoiM = Transpose[A];
(* Now we construct the rate vector *)
ks = {k1 × x3 × x1, k2 × x5, k3 × x5, k4 × x4 × x1,
   k5 × x6, k6 × x6, k7 × x2, k8 × x3, k9 × x4, k10 × x5, k11 × x6};
ssEqns = stoiM.ks;
mC = RowReduce[NullSpace[A]];
subsEqns = {ssEqns[[2]], ssEqns[[4]],
   ssEqns[[5]], ssEqns[[6]], x1 + x2 + x5 + x6 - T1, x3 + x4 + x5 + x6 - T2};
jacobian = D[subsEqns, {{x1, x2, x3, x4, x5, x6}}];
detJ = Collect[Distribute[Det[jacobian]], {x1, x2, x3, x4, x5, x6}];
solution =
  Solve[{{subsEqns[[1]], subsEqns[[2]], subsEqns[[3]], subsEqns[[4]]} == 0},
   {x2, x4, x5, x6}];
detSubs = Replace[detJ, solution[[1]], {0, Infinity}];
(* Equivilant to detSubs=detJ/.solution[[1]]; *)
polSubs = Numerator[Together[detSubs]];
finalSubs = Collect[Distribute[polSubs], x_, FactorTerms]
```

```
factor = k1^2 k3 k4 k5 k9 k10 + k1^2 k3 k4 k6 k9 k10 - k1^2 k4 k5 k6 k9 k10 -
   k1^2 k4 k6^2 k9 k10 - k1^2 k4 k5 k6 k10^2 - k1^2 k4 k6^2 k10^2 - k1^2 k4 k5 k7 k10^2 - k1^2 k4 k6 k7 k10^2 -
   k1 k2 k3 k4^2 k8 k11 - k1 k3^2 k4^2 k8 k11 + k1 k2 k4^2 k6 k8 k11 + k1 k3 k4^2 k6 k8 k11 -
   k1^2 k3 k4 k5 k10 k11 - k1^2 k3 k4 k6 k10 k11 - k1 k2 k4^2 k6 k10 k11 - k1 k3 k4^2 k6 k10 k11 -
   k1 k2 k4^2 k7 k10 k11 - k1 k3 k4^2 k7 k10 k11 - k1^2 k4 k5 k7 k10 k11 - k1^2 k4 k6 k7 k10 k11 -
   k1 k2 k3 k4^2 k11^2 - k1 k3^2 k4^2 k11^2 - k1 k2 k4^2 k7 k11^2 - k1 k3 k4^2 k7 k11^2;
```

```
Factor[factor]
```

```
term = k1 k3 k5 k9 k10 + k1 k3 k6 k9 k10 - k1 k5 k6 k9 k10 - k1 k6^2 k9 k10 -
   k1 k5 k6 k10^2 - k1 k6^2 k10^2 - k1 k5 k7 k10^2 - k1 k6 k7 k10^2 - k2 k3 k4 k8 k11 - k3^2 k4 k8 k11 +
   k2 k4 k6 k8 k11 + k3 k4 k6 k8 k11 - k1 k3 k5 k10 k11 - k1 k3 k6 k10 k11 - k2 k4 k6 k10 k11 -
   k3 k4 k6 k10 k11 - k2 k4 k7 k10 k11 - k3 k4 k7 k10 k11 - k1 k5 k7 k10 k11 -
   k1 k6 k7 k10 k11 - k2 k3 k4 k11^2 - k3^2 k4 k11^2 - k2 k4 k7 k11^2 - k3 k4 k7 k11^2;
```

```
simpTerm = FullSimplify[term]
```

```
simplerTerm = Distribute[simpTerm / (k1 * k4)] /. {(k2 + k3) / k1 → M1, (k5 + k6) / k4 → M2}
```

This above term larger than 0 should be the necessary condition.

```
condition = simplerTerm > 0
```

By mannual simplying the term, we can have:

```
simpleCond = (k3 - k6) * (M2 * k9 * k10 - M1 * k8 * k11) >
   (k11 * M1 + k10 * M2) * ((k6 * k10 + k3 * k11) + k7 * (k10 + k11))
```

```
left = (k3 - k6) * (M2 * k9 * k10 - M1 * k8 * k11) /. {M1 → (k2 + k3) / k1, M2 → (k5 + k6) / k4}
```

```
right = (k11 * M1 + k10 * M2) * ((k6 * k10 + k3 * k11) + k7 * (k10 + k11)) /.
   {M1 → (k2 + k3) / k1, M2 → (k5 + k6) / k4}
```

To fullfile the assumption of thermodynamic conditions for the reversible reactions, we have the the constraint:

$$\frac{k_1\,k_{10}}{k_2\,k_{11}} = \frac{k_4\,k_8}{k_5\,k_9}.$$

This will give us a even simple condition. Then we will example how will this condition result in the parameter space for multistationarity.

```
oriCond = simpleCond /. {M1 → (k2 + k3) / k1, M2 → (k5 + k6) / k4}
```

$$(k_3 - k_6)\left(\frac{(k_5 + k_6)\,k_9\,k_{10}}{k_4} - \frac{(k_2 + k_3)\,k_8\,k_{11}}{k_1}\right) > \left(\frac{(k_5 + k_6)\,k_{10}}{k_4} + \frac{(k_2 + k_3)\,k_{11}}{k_1}\right)(k_6\,k_{10} + k_3\,k_{11} + k_7\,(k_{10} + k_{11}))$$

$$\text{Simplify}\left[\text{oriCond},\ \text{Assumptions} \to \frac{k_1\,k_{10}}{k_2\,k_{11}} == \frac{k_4\,k_8}{k_5\,k_9}\right]$$

$$\frac{(k_3 - k_6)\,(k_1\,k_6\,k_9\,k_{10} - k_3\,k_4\,k_8\,k_{11})}{k_1\,k_4} > \left(\frac{(k_5 + k_6)\,k_{10}}{k_4} + \frac{(k_2 + k_3)\,k_{11}}{k_1}\right)((k_6 + k_7)\,k_{10} + (k_3 + k_7)\,k_{11})$$

Better to do it manually, then we have the condition with thermodynamic constraint:

thermoCond =

$$(k_3 - k_6)\,(k_6\,k_2 - k_3\,k_5) > \left(\frac{k_2}{k_9} \times \frac{k_5\text{^}2 + k_6}{k_5} + \frac{k_5}{k_8} \times \frac{k_2\text{^}2 + k_3}{k_2}\right)((k_6 + k_7)\,k_{10} + (k_3 + k_7)\,k_{11})$$

$$(k_3 - k_6)\,(-k_3\,k_5 + k_2\,k_6) > \left(\frac{(k_2^2 + k_3)\,k_5}{k_2\,k_8} + \frac{k_2\,(k_5^2 + k_6)}{k_5\,k_9}\right)((k_6 + k_7)\,k_{10} + (k_3 + k_7)\,k_{11})$$

Fromt the above condition, we can get some general idea that in order to satisfy the thermodynamic condition we should have:

Necessarily:

$k_3 > k_6$ and $k_2 > k_5$

or

$k_3 < k_6$ and $k_5 > k_2$

With additional:

k_8, $k_9 \gg k_{10}$, k_{11} and k_7, k_{10}, $k_{11} \approx 0$

Sampling the parameters (without thermodynamic constraint)

Here we try to sampling the parameters by enforcing the thermodynamc constraint. The parameters are sampled in biologically meaningful ranges.

(NewKernel) In[546]:=

```
ClearAll["Global`*"];
A = Table[0, {11}, {6}];
```

```
A[[1]][[1]] = -1;
A[[1]][[3]] = -1;
A[[1]][[5]] = 1;
A[[2]] = -A[[1]];
A[[3]][[3]] = 1; A[[3]][[2]] = 1; A[[3]][[5]] = -1;
A[[4]][[1]] = -1;
A[[4]][[4]] = -1;
A[[4]][[6]] = 1;
A[[5]] = -A[[4]];
A[[6]][[4]] = 1;
A[[6]][[2]] = 1;
A[[6]][[6]] = -1;
A[[7]][[2]] = -1;
A[[7]][[1]] = 1;
A[[8]][[3]] = -1; A[[8]][[4]] = 1; A[[9]] = -A[[8]];
A[[10]][[5]] = -1; A[[10]][[6]] = 1; A[[11]] = -A[[10]];
stoiM = Transpose[A];
(* Now we construct the rate vector *)
ks = {k1 × x3 × x1, k2 × x5, k3 × x5, k4 × x4 × x1,
   k5 × x6, k6 × x6, k7 × x2, k8 × x3, k9 × x4, k10 × x5, k11 × x6};
ssEqns = stoiM.ks;
mC = RowReduce[NullSpace[A]];
subsEqns = {ssEqns[[2]], ssEqns[[4]],
   ssEqns[[5]], ssEqns[[6]], x1 + x2 + x5 + x6 - T1, x3 + x4 + x5 + x6 - T2};
jacobian = D[subsEqns, {{x1, x2, x3, x4, x5, x6}}];
detJ = Collect[Distribute[Det[jacobian]], {x1, x2, x3, x4, x5, x6}];
solution =
  Solve[{{subsEqns[[1]], subsEqns[[2]], subsEqns[[3]], subsEqns[[4]]} == 0},
   {x2, x4, x5, x6}];
detSubs = Replace[detJ, solution[[1]], {0, Infinity}];
(* Equivilant to detSubs=detJ/.solution[[1]]; *)
polSubs = Numerator[Together[detSubs]];
finalSubs = Collect[Distribute[polSubs], x_, FactorTerms];
(*The above code is the same as first section*)

bistableKs = {};
bistableParSets = {};
SeedRandom[];
Timing[
 Do[{
   rands = Exp[RandomReal[{0, 1}, 11] * 2 * Log[1000] - Log[1000]];
   k1 = rands[[1]];
   k2 = rands[[2]];
   k3 = rands[[3]];
   k4 = rands[[4]];
   k5 = rands[[5]];
   k6 = rands[[6]];
   k7 = rands[[7]];
   k8 = rands[[8]];
   k9 = rands[[9]];
   k10 = rands[[10]];
   k11 = rands[[11]];
   left = (k3 - k6) ((k5 + k6) k9 k10 / k4 - (k2 + k3) k8 k11 / k1);
   right = ((k5 + k6) k10 / k4 + (k2 + k3) k11 / k1) (k6 k10 + k3 k11 + k7 (k10 + k11));
   If[left > right, {
     AppendTo[bistableKs, {k1, k2, k3, k4, k5, k6, k7, k8, k9, k10, k11}];
```

```
      (*counter=1;hitQ=0;
      While[hitQ==0&&counter≤1000,{
        x1=RandomReal[{0,1000}];
        finalSol=NSolve[finalSubs==0/.{k_1→k1,k_2→k2,k_3→k3,k_4→k4,k_5→k5,
            k_6→k6,k_7→k7,k_8→k8,k_9→k9,k_10→k10,k_11→k11,x_1→x1},{x_3}];
        x3=x_3/.finalSol[[1]];
        realSol=solution/.{k_1→k1,k_2→k2,k_3→k3,k_4→k4,k_5→k5,
           k_6→k6,k_7→k7,k_8→k8,k_9→k9,k_10→k10,k_11→k11,x_1→x1,x_3→x3};
        T1=(x_1+x_2+x_5+x_6)/.Flatten[Append[{x_1→x1,x_3→x3},realSol[[1]]]];
        T2=(x_3+x_4+x_5+x_6)/.Flatten[Append[{x_1→x1,x_3→x3},realSol[[1]]]];
        If[T1≤1000&&T2≤1000,{
          AppendTo[bistableParSets,
           {k1,k2,k3,k4,k5,k6,k7,k8,k9,k10,k11,T1,T2,left,right}];
          hitQ=1;
         }];
        counter++;
       }];*)
     }];
   }, {i, 10000}];
 ]
```

(NewKernel) Out[568]=

```
{0.336731, Null}
```

(NewKernel) In[545]:=

```
Length[bistableKs]
```

(NewKernel) Out[545]=

```
14325
```

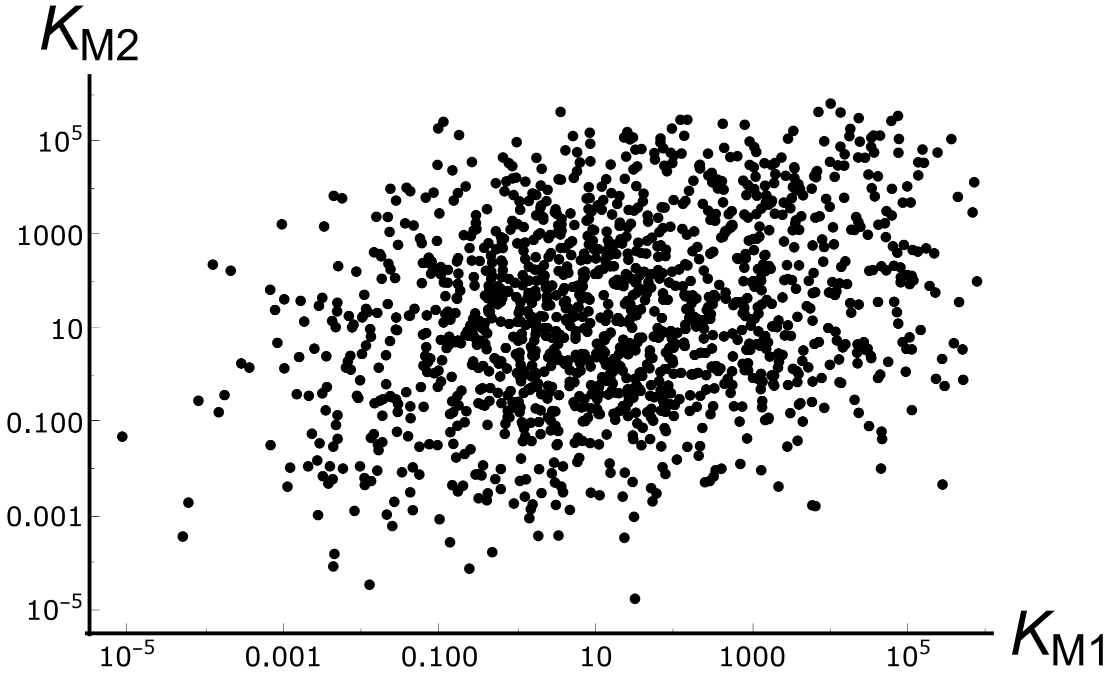
(NewKernel) In[490]:=

```
tKs = Transpose[bistableKs];
km1 = (tKs[[2]] + tKs[[3]]) / tKs[[1]];
km2 = (tKs[[5]] + tKs[[6]]) / tKs[[4]];
kd1 = tKs[[8]] / tKs[[9]];
kd2 = tKs[[10]] / tKs[[11]];
```

(NewKernel) In[425]:=

```
kmPlot = ListLogLogPlot[Transpose[{km1, km2}], ImageSize → Large, PlotRange → Full,
  PlotLabel → None, LabelStyle → {32, GrayLevel[0]}, AxesLabel → {"KM1", "KM2"},
  Ticks → Automatic, TicksStyle → Directive["Label", 14],
  AxesStyle → Thick, PlotTheme → "Monochrome"]
```

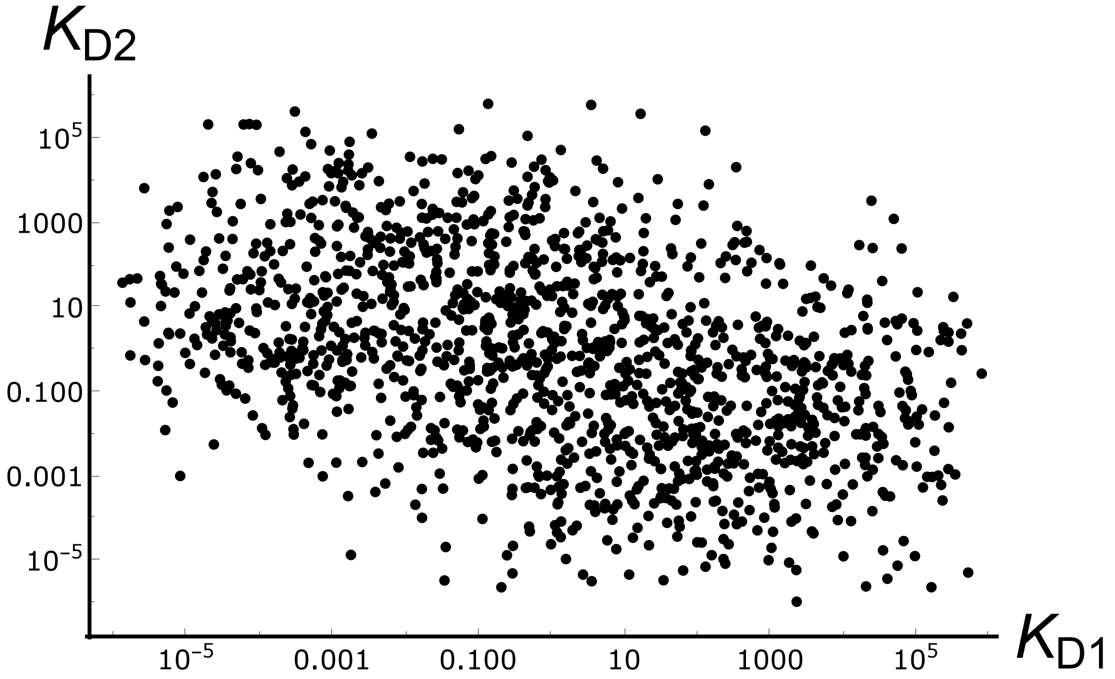
(NewKernel) Out[425]=



(NewKernel) In[426]:=

```
kdPlot = ListLogLogPlot[Transpose[{kd1, kd2}], ImageSize → Large, PlotRange → Full,
  PlotLabel → None, LabelStyle → {32, GrayLevel[0]}, AxesLabel → {"KD1", "KD2"},
  Ticks → Automatic, TicksStyle → Directive["Label", 14],
  AxesStyle → Thick, PlotTheme → "Monochrome"]
```

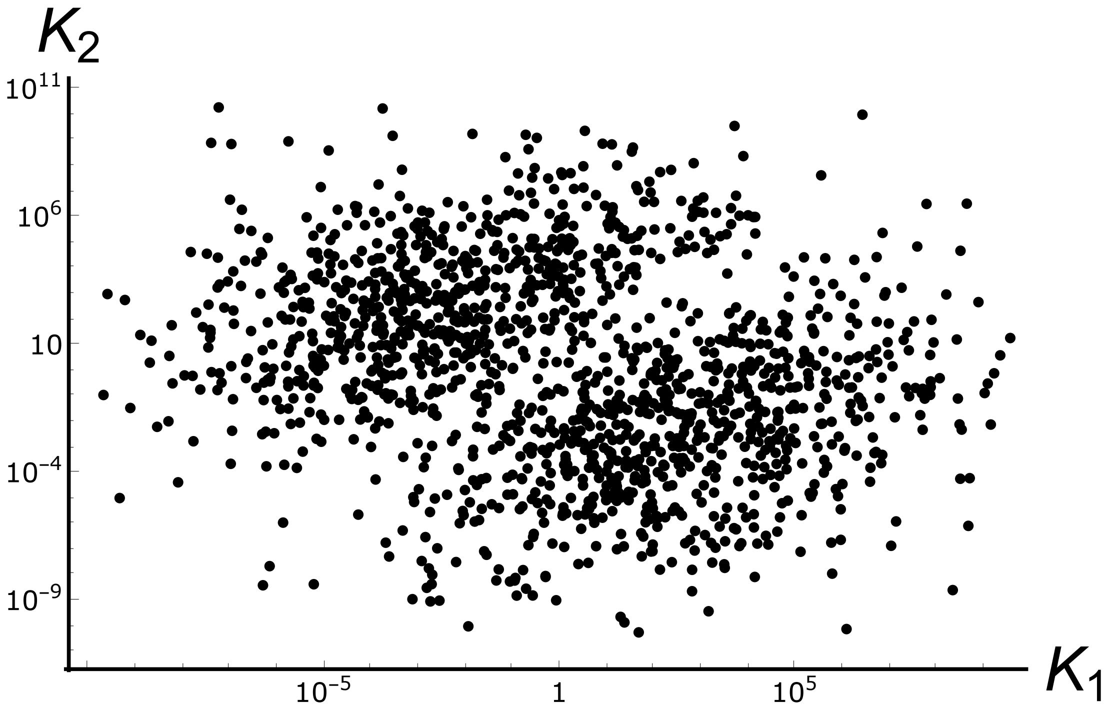
(NewKernel) Out[426]=



(NewKernel) In[497]:=

```
kdPlot = ListLogLogPlot[Transpose[{tKs[[1]] * tKs[[10]] * tKs[[5]] * tKs[[9]],
     tKs[[2]] * tKs[[11]] * tKs[[4]] * tKs[[8]]}], ImageSize → Large,
   PlotRange → Full, PlotLabel → None, LabelStyle → {32, GrayLevel[0]},
   AxesLabel → {"K₁", "K₂"}, Ticks → Automatic, TicksStyle → Directive["Label", 14],
   AxesStyle → Thick, PlotTheme → "Monochrome"]
```

(NewKernel) Out[497]=



(NewKernel) In[427]:=

```
model = LinearModelFit[Transpose[{Log10[kd1], Log10[kd2]}], x, x]
```

(NewKernel) Out[427]=

```
FittedModel[ -0.056131-0.346952x ]
```

(NewKernel) In[428]:=

```
InputForm[model["BestFit"]]
```

(NewKernel) Out[428]//InputForm=

```
-0.0561309658626419 - 0.34695151867837787*x
```

(NewKernel) In[429]:=

```
model["ParameterTable"]
```

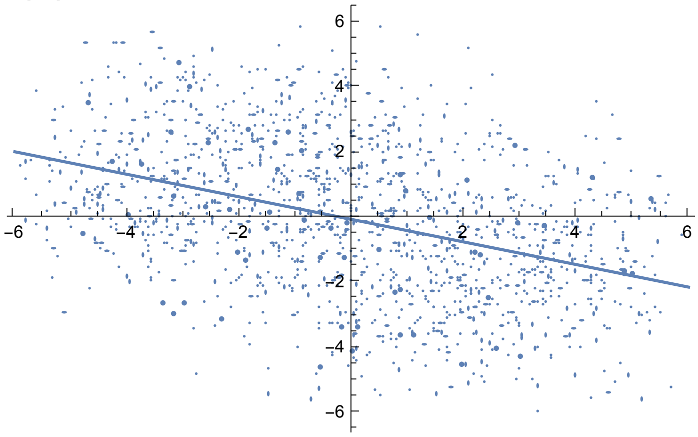
(NewKernel) Out[429]=

```
  | Estimate   StandardError  t-Statistic  P-Value
1 | -0.056131  0.05661        -0.991538    0.321594
x | -0.346952  0.0208559      -16.6356     7.66607×10^-57
```

(NewKernel) In[430]:=

```
Show[ListPlot[Transpose[{Log10[kd1], Log10[kd2]}]],
 Plot[model["BestFit"], {x, -6, 6}]]
```

(NewKernel) Out[430]=



(NewKernel) In[301]:=

```
pc = PrincipalComponents[Log10[bistableKs]];
```

(NewKernel) In[300]:=

```
pc[[Range[1, 10]]];
```

Sampling the parameters (with thermodynamic constraint)

Here we try to sampling the parameters by enforcing the thermodynamc constraint. The parameters are sampled in biologically meaningful ranges.

Sampling the parameters related to thermodynamic constraint k1, k2, k4, k5, k8, k9, k10, k11. Using Gamma distribution with $\alpha = 7$, $\beta = 2$, and then uniformly sample 4 random numbers that sum to 1, these are for k1, k5, k9, k10. Sample anther four uniform random number for k2, k4, k8, k11.

For other parameters:

(NewKernel) In[593]:=

```
ClearAll["Global`*"];
A = Table[0, {11}, {6}];
A[[1]][[1]] = -1;
A[[1]][[3]] = -1;
A[[1]][[5]] = 1;
A[[2]] = -A[[1]];
A[[3]][[3]] = 1; A[[3]][[2]] = 1; A[[3]][[5]] = -1;
A[[4]][[1]] = -1;
A[[4]][[4]] = -1;
A[[4]][[6]] = 1;
A[[5]] = -A[[4]];
A[[6]][[4]] = 1;
A[[6]][[2]] = 1;
A[[6]][[6]] = -1;
A[[7]][[2]] = -1;
A[[7]][[1]] = 1;
A[[8]][[3]] = -1; A[[8]][[4]] = 1; A[[9]] = -A[[8]];
A[[10]][[5]] = -1; A[[10]][[6]] = 1; A[[11]] = -A[[10]];
stoiM = Transpose[A];
(* Now we construct the rate vector *)
ks = {k1 × x3 × x1, k2 × x5, k3 × x5, k4 × x4 × x1,
   k5 × x6, k6 × x6, k7 × x2, k8 × x3, k9 × x4, k10 × x5, k11 × x6};
```

```
ssEqns = stoiM.ks;
mC = RowReduce[NullSpace[A]];
subsEqns = {ssEqns[[2]], ssEqns[[4]],
   ssEqns[[5]], ssEqns[[6]], x_1 + x_2 + x_5 + x_6 - T_1, x_3 + x_4 + x_5 + x_6 - T_2};
jacobian = D[subsEqns, {{x_1, x_2, x_3, x_4, x_5, x_6}}];
detJ = Collect[Distribute[Det[jacobian]], {x_1, x_2, x_3, x_4, x_5, x_6}];
solution =
  Solve[{{subsEqns[[1]], subsEqns[[2]], subsEqns[[3]], subsEqns[[4]]} == 0},
   {x_2, x_4, x_5, x_6}];
detSubs = Replace[detJ, solution[[1]], {0, Infinity}];
(* Equivilant to detSubs=detJ/.solution[[1]]; *)
polSubs = Numerator[Together[detSubs]];
finalSubs = Collect[Distribute[polSubs], x_, FactorTerms];
(*The above code is the same as first section*)


bistableKs = {};
bistableParSets = {};
SeedRandom[];
Timing[
 Do[{
    others = Exp[RandomReal[{0, 1}, 3] * 2 * Log[1000] - Log[1000]];
    While[(randls = RandomReal[{0, 1}, 4];
      sum = Total@randls;
      psum = Switch[True, 0 ≤ # ≤ 1, 1. / 4 * sum^3, 1 < # ≤ 2, 1 / 4 * (-3 * sum^3 + 12 *
             sum^2 - 12 * sum + 4), 2 < # ≤ 3, 1 / 4 * (3 * sum^3 - 24 * sum^2 + 60 * sum -
            44), 3 < # ≤ 4, 1 / 4 * (-sum^3 + 12 * sum^2 - 48 * sum + 64)] &[sum];
      RandomReal ≤ psum)];
    rand23 = RandomVariate[DirichletDistribution[{1, 1, 1, 1}]];
    rand21 = 1 - Total@rand23;
    k1 = Exp[randls[[1]] * 2 * Log[1000] - Log[1000]];
    k2 = Exp[sum * rand23[[1]] * 2 * Log[1000] - Log[1000]];
    k3 = others[[1]];
    k4 = Exp[sum * rand23[[2]] * 2 * Log[1000] - Log[1000]];
    k5 = Exp[randls[[2]] * 2 * Log[1000] - Log[1000]];
    k6 = others[[2]];
    k7 = others[[3]];
    k8 = Exp[sum * rand23[[3]] * 2 * Log[1000] - Log[1000]];
    k9 = Exp[randls[[3]] * 2 * Log[1000] - Log[1000]];
    k10 = Exp[randls[[4]] * 2 * Log[1000] - Log[1000]];
    k11 = Exp[sum * rand21 * 2 * Log[1000] - Log[1000]];
    left = (k3 - k6) ((k5 + k6) k9 k10 / k4 - (k2 + k3) k8 k11 / k1);
    right = ((k5 + k6) k10 / k4 + (k2 + k3) k11 / k1) (k6 k10 + k3 k11 + k7 (k10 + k11));
    If[left > right, {
      AppendTo[bistableKs, {k1, k2, k3, k4, k5, k6, k7, k8, k9, k10, k11}];
      (*counter=1;hitQ=0;
      While[hitQ==0&&counter≤1000,{
        x1=Exp[-RandomVariate[
             ExponentialDistribution[Log[2]/(-Log[0.0001])]]]*1000;
        finalSol=NSolve[finalSubs==0/.{k_1→k1,k_2→k2,k_3→k3,k_4→k4,k_5→k5,
             k_6→k6,k_7→k7,k_8→k8,k_9→k9,k_10→k10,k_11→k11,x_1→x1},{x_3}];
        x3=x_3/.finalSol[[1]];
        realSol=solution/.{k_1→k1,k_2→k2,k_3→k3,k_4→k4,k_5→k5,
            k_6→k6,k_7→k7,k_8→k8,k_9→k9,k_10→k10,k_11→k11,x_1→x1,x_3→x3};
        T1=(x_1+x_2+x_5+x_6)/.Flatten[Append[{x_1→x1,x_3→x3},realSol[[1]]]];
        T2=(x_3+x_4+x_5+x_6)/.Flatten[Append[{x_1→x1,x_3→x3},realSol[[1]]]];
```

```
            If[0.0001≤T1≤1000&&0.0001≤T2≤1000,{
              AppendTo[bistableParSets,
               {k1,k2,k3,k4,k5,k6,k7,k8,k9,k10,k11,T1,T2,left,right}];
              hitQ=1;
             }];
            counter++;
           }];*)
         }];
      }, {i, 100000}];
    ]
```

(NewKernel) Out[615]=

```
{23.683, Null}
```

(NewKernel) In[616]:=

```
Length[bistableKs]
```

(NewKernel) Out[616]=

```
4378
```

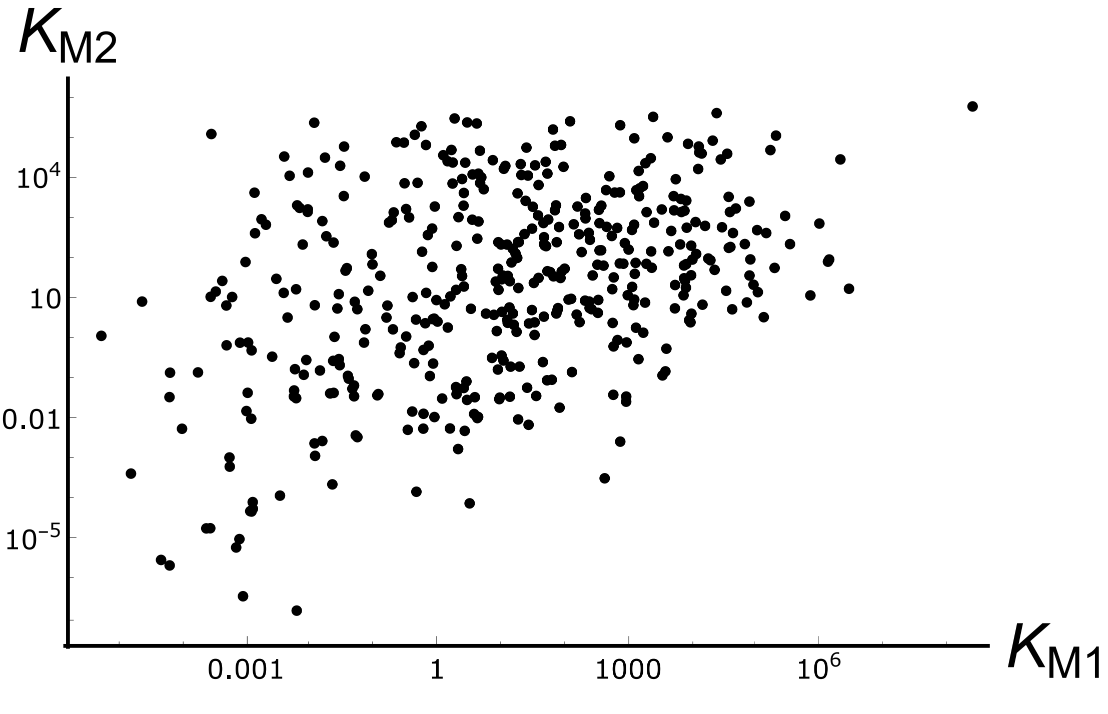
(NewKernel) In[455]:=

```
tKs = Transpose[bistableKs];
      tKs[[2]] + tKs[[3]]
km1 = ───────────────────;
           tKs[[1]]
      tKs[[5]] + tKs[[6]]
km2 = ───────────────────;
           tKs[[4]]
      tKs[[8]]
kd1 = ────────;
      tKs[[9]]
      tKs[[10]]
kd2 = ─────────;
      tKs[[11]]
```

(NewKernel) In[460]:=

```
kmPlot = ListLogLogPlot[Transpose[{km1, km2}], ImageSize → Large, PlotRange → Full,
  PlotLabel → None, LabelStyle → {32, GrayLevel[0]}, AxesLabel → {"KM1", "KM2"},
  Ticks → Automatic, TicksStyle → Directive["Label", 14],
  AxesStyle → Thick, PlotTheme → "Monochrome"]
```

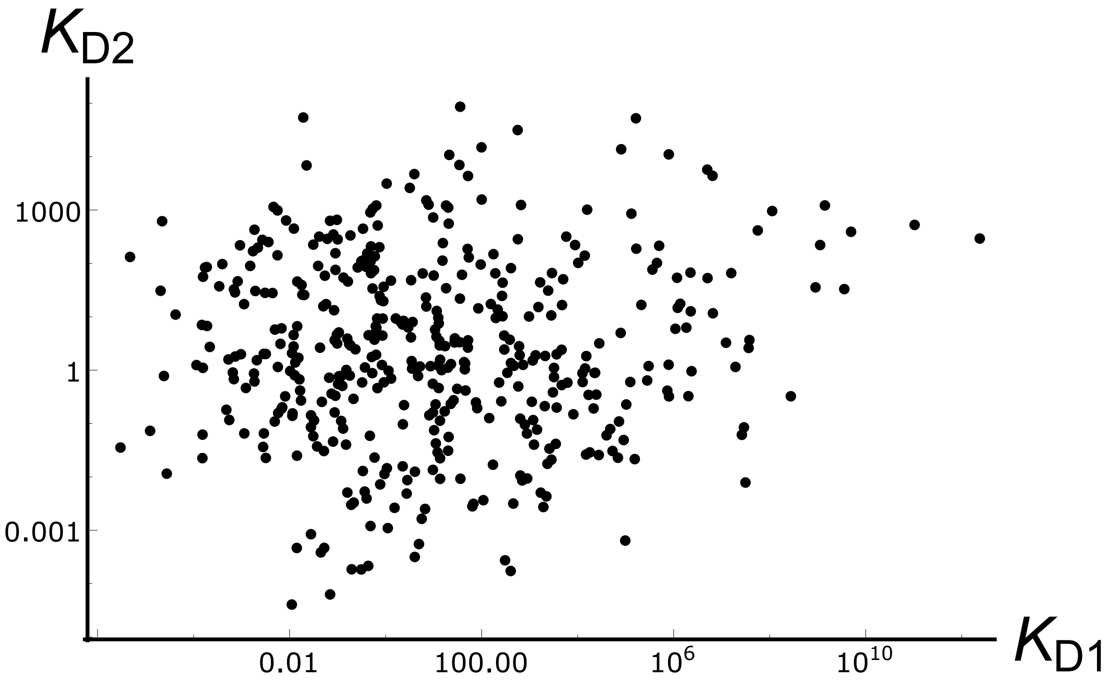
(NewKernel) Out[460]=



(NewKernel) In[461]:=

```
kdPlot = ListLogLogPlot[Transpose[{kd1, kd2}], ImageSize → Large, PlotRange → Full,
  PlotLabel → None, LabelStyle → {32, GrayLevel[0]}, AxesLabel → {"KD1", "KD2"},
  Ticks → Automatic, TicksStyle → Directive["Label", 14],
  AxesStyle → Thick, PlotTheme → "Monochrome"]
```

(NewKernel) Out[461]=



(NewKernel) In[462]:=

```
model = LinearModelFit[Transpose[{Log10[kd1], Log10[kd2]}], x, x]
```

(NewKernel) Out[462]=

```
FittedModel[ 0.372627+0.0354998x ]
```

(NewKernel) In[463]:=

```
InputForm[model["BestFit"]]
```

(NewKernel) Out[463]//InputForm=

```
0.37262682052688595 + 0.035499771883158025*x
```

(NewKernel) In[464]:=

```
model["ParameterTable"]
```

(NewKernel) Out[464]=

```
  | Estimate  StandardError t-Statistic P-Value
1 | 0.372627  0.0877545     4.24624     0.0000266342
x | 0.0354998 0.0274304     1.29417     0.196297
```

(NewKernel) In[465]:=

```
Show[ListPlot[Transpose[{Log10[kd1], Log10[kd2]}]],
 Plot[model["BestFit"], {x, -6, 6}]]
```

■ Conclusion:

These above results show that the parameter set within biological meaningful ranges can be reached by increasing the sampling size even when enforcing the thermodynamic constraint. Comparing to results from the other document (without enforcing thermodynamic constraint), the paramter space is largely reduced.

Conditions	with thermo	with thermo	without thermo	without thermo
Sampling size	only check bistability	bistability & concentrations	only check bistability	bistability & concentrations
10^4	396	N / A	1448	N / A
10^5	4378	N / A	14 325	N / A

From the sampling results, thermodynamic constraint indeed shrinks the parameter space for bistability in the motif. Although, the concentration is probably not very well sampled, the concentration is trivial for its purpose here.

Test