Kappa tutorial

Russ Harmer
LIP, CNRS & ENS Lyon

0. Introduction

Why model signaling?

- Widely believed that many diseases arise from perturbed signaling networks
  - cancer, diabetes, neurodegenerative diseases, ...
  - mutations, gene amplification / ablation, ...
- These networks are **mind-bogglingly complicated**
  - 10s or even 100s of thousands of PPIs
  - impossible to ‘figure out’ in your head

Context-dependence

- The ‘rules’ of signaling are universal ...
  - ‘Grb2 binds Sos’ is an *in vitro* fact
    - cell-centric perspective: no spatial/tissue context
- ... but perturbations affect their meaning
  - which proteins and mutants are present: *static* context that changes slowly: ‘cell type’
  - transient presence of ligands: *dynamic* context; can feed back on the static: ‘signaling’

Traditional modeling

(of dynamical systems' behavior)

- Primarily a 'mental' activity
  - specify the key variables *and* their inter-dependencies resulting from perturbations
  - this amounts to making model-level assumptions *during* model construction
- Comes from physics
  - a model is a synthesis of our understanding
  - tends to be a (small and) closed world

Traditional modeling

(of dynamical systems' behavior)

- Biology needs a different notion of model
  - we cannot hope to 'understand' signaling networks *a priori*: too big, complex, ...
  - we need models precisely *in order to* achieve this understanding!
  - models must be (large and) open worlds: extensible, modifiable and perturbable
  - do *not* want a different model for each cell type

‘Open’ modeling?

- What if ...
  - a model could be assembled from many independently-conceived elements;
    - and model-level assumptions be discovered *a posteriori* ?
- Enables incremental model development
  - ‘easy’ to modify, and add new, elements

→ rule-based modeling

Rule-based modeling

- A way of conveniently defining a large class of dynamical systems
  - based on an agent abstraction and rewriting rules
- An *agent* has an interface of sites
  - loci of binding interaction (with optional states)
- A *rule* specifies local conditions for
  - binding / unbinding and state changes

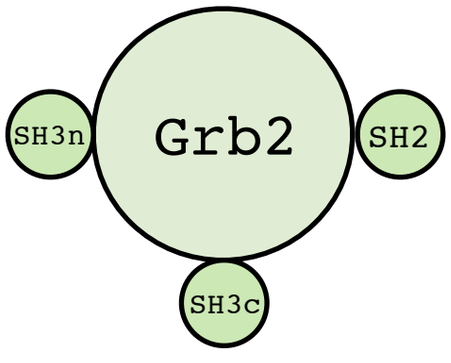
Rule-based modeling

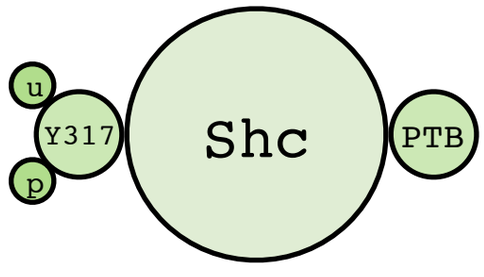
- The state of a system is a ‘site graph’
  - a graph whose *nodes* are sites
    - agents are *equivalence classes* of nodes
- Applying a rule effects a rewrite
  - induces a *state transition*
  - initial state + rules define a *transition system*
  - transitions can be *weighted*: CTMC

The agent abstraction

- Agent = formal protein
  - a name, *e.g.* `Grb2`, `Shc`
  - a set of binding sites with optional states:

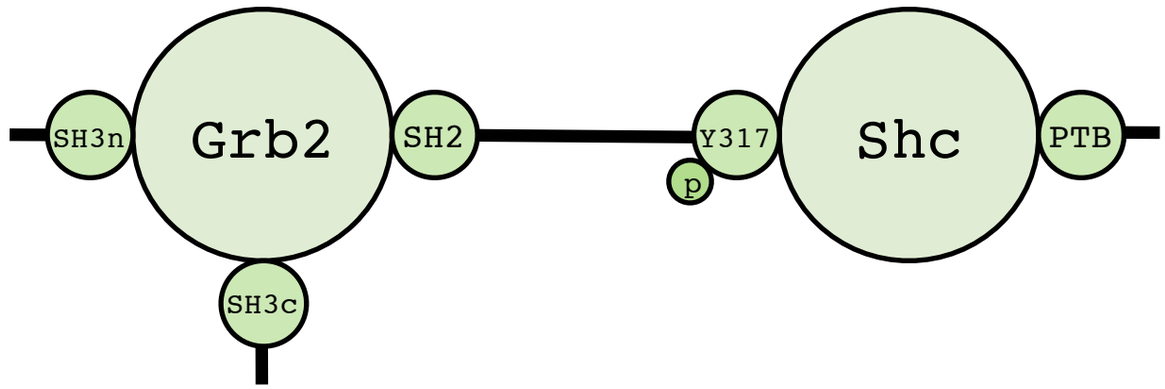
```
%agent: Grb2(SH3n, SH2, SH3c)
%agent: Shc(PTB, Y317~u~p)
```





Site graphs

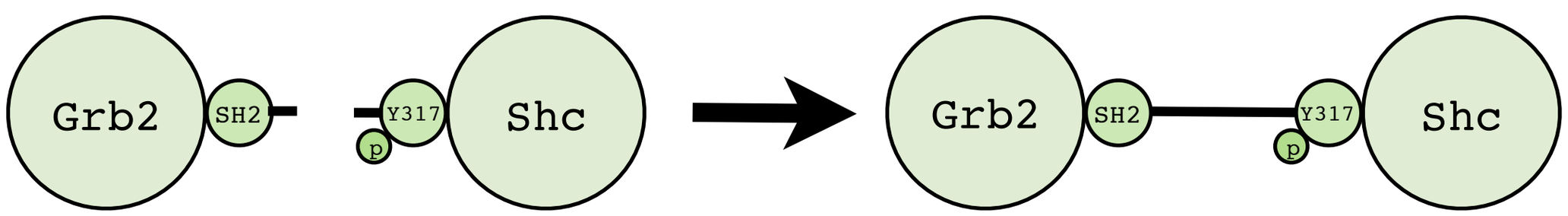
- Connected component = formal complex



- ‘Deterministic’ in the sense that
  - at most one edge/stub from any given site
  - at most one state on any given site

Rules

- Express *all and only* that which is necessary for a single interaction, *e.g.*




```
Grb2(SH2), Shc(Y317~p) -> Grb2(SH2!0), Shc(Y317~p!0)
```

- A formal expression of what biochemists say

Rule-based modeling

- A rule-based model consists of
  - a collection of rules (with rate constants)
  - initial conditions (how many of each agent)
- **Implicit-state stochastic semantics**
  - tacit assumptions, *e.g.* '`EGFR` and `Grb2` compete for `Shc`', constrain the state space
  - `KaSim` v3: developed by J. Krivine & J. Feret [available on github]

Static model-dependence

- In a given model, some rules may *never* be applicable
  - this can be detected by static analysis
    - reveals implicit constraints / invariants
    - does *not* need rate constants
- `KaSA` [in development]: J. Feret
  - previously `complx` [available on github]

Causal model-dependence

- Rules can be causally related
  - a firing of r_1 may *enable* a firing of r_2
  - a firing of r_1 may *disable* a firing of r_2
- These relations are model-independent
  - but only *some* are active in a given model
  - rules experience different causal pressure in different models: changes the variety of possible causal paths [detected by KaSim v3]

Complementary analyses

- Static analysis of a model can
  - *confirm* a negative assertion, *e.g.* ‘this complex cannot be formed’
  - *refute* a positive assertion
- Causal analysis can
  - *confirm* a positive assertion, by exhibiting the causal paths leading to it [uses simulation]
  - *refute* a negative assertion

In summary...

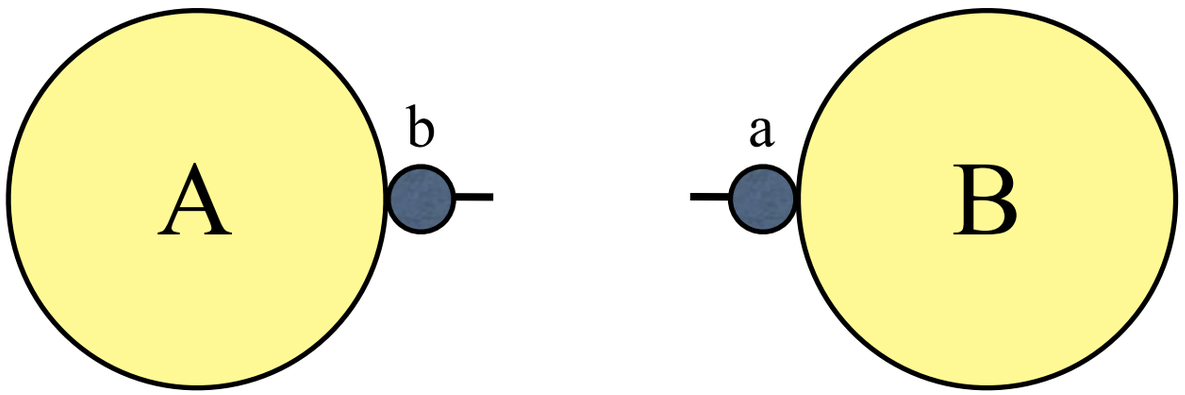
- Rules are model elements
  - document necessary conditions for individual protein-protein interactions
- Models are collections of rules
  - static and causal analyses can render explicit their implicit assumptions
  - rules *intrinsically* have model-dependent meaning [reflects context-dependence]

I. Binding equilibrium

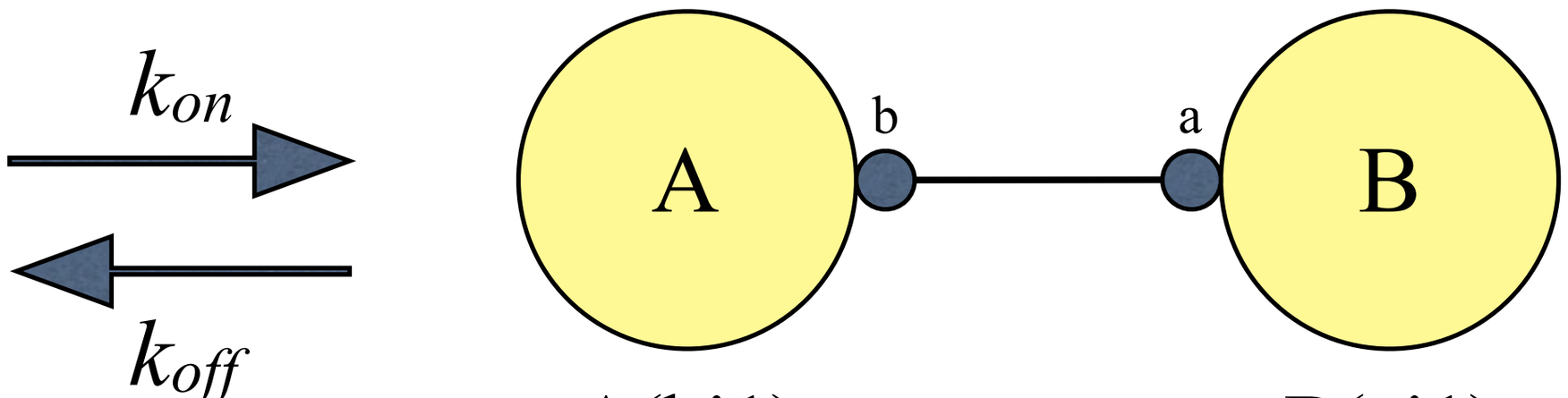
You need to be able to...

- Create and edit *plain text* files
  - TextEdit, emacs, ...
- Run the Kappa simulator from the *command line*
  - sh> **KaSim -i AB.ka -e 1000 -p 500**
- Plot the output using gnuplot (or other software)
  - gnuplot> **plot “data.out” using 1:3 with lines**

Binding equilibrium



A(b) , B(a)



A(b!1) , B(a!1)

AB.ka

http://perso.ens-lyon.fr/russell.harmer/CR11/AB.ka

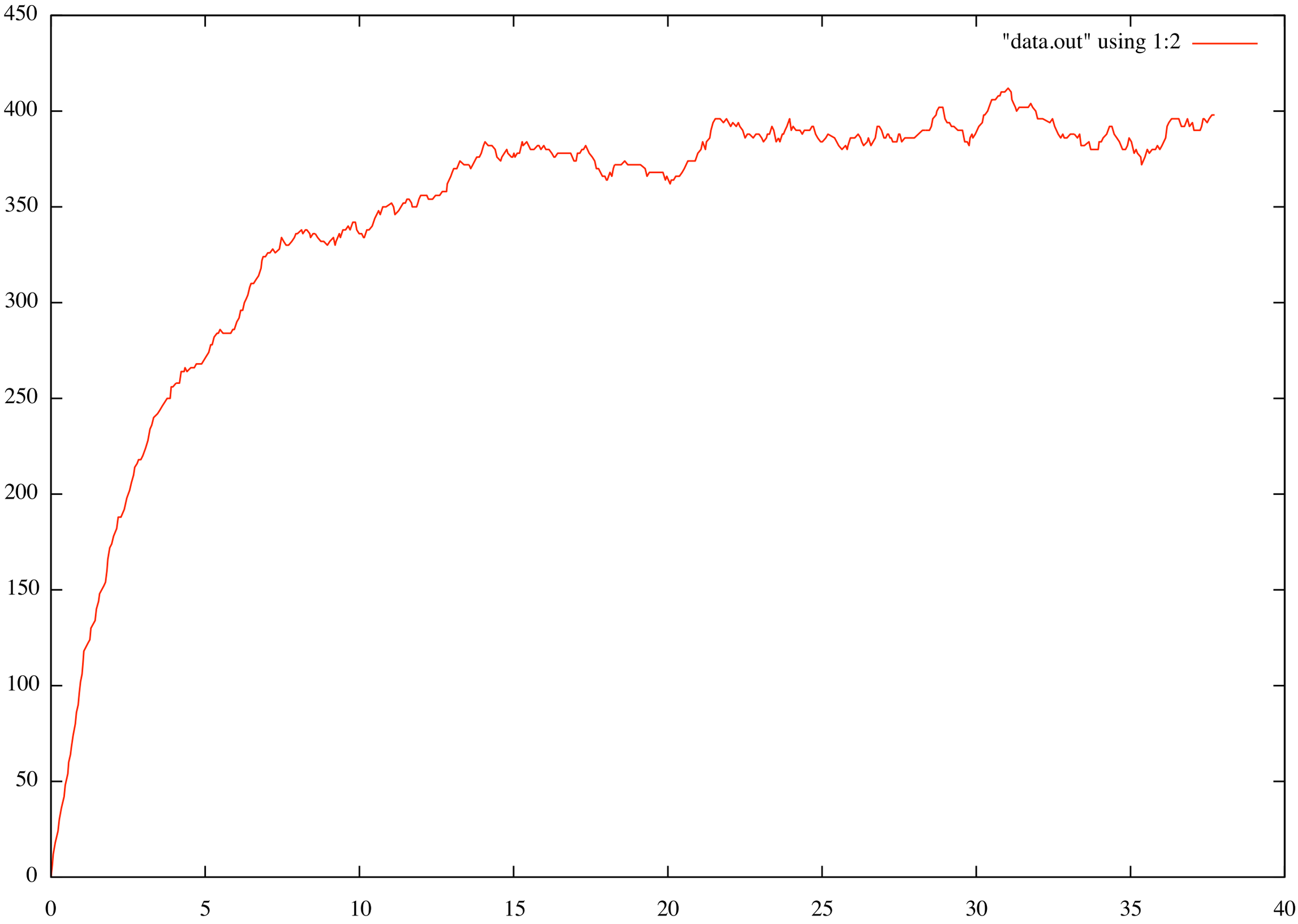
```
# agent declarations
%agent: A(b)
%agent: B(a)

# some useful variables
%var: 'fast' 10
%var: 'medium' 1
%var: 'slow' 0.1
%var: 'BND' 0.0001
%var: 'BRK' 0.1
%var: 'MOD' 0.1

# binding rule
A(b), B(a) -> A(b!0), B(a!0) @ 'BND' # * 'fast'

# unbinding rule
A(b!0), B(a!0) -> A(b), B(a) @ 'BRK' # * 'slow'

# initial state
%init: 1000 A(b)
%init: 1000 B(a)

# count the number of AB complexes
%obs: 'AB' A(b!1), B(a!1)
```

Running KaSim

- We must specify:
  - the input file: -i
  - the number of events *or* time: `-e` *or* `-t`
  - number of output time points: `-p`
- e.g.
  - sh> **KaSim -i AB.ka -e 3000 -p 500**

Displaying output

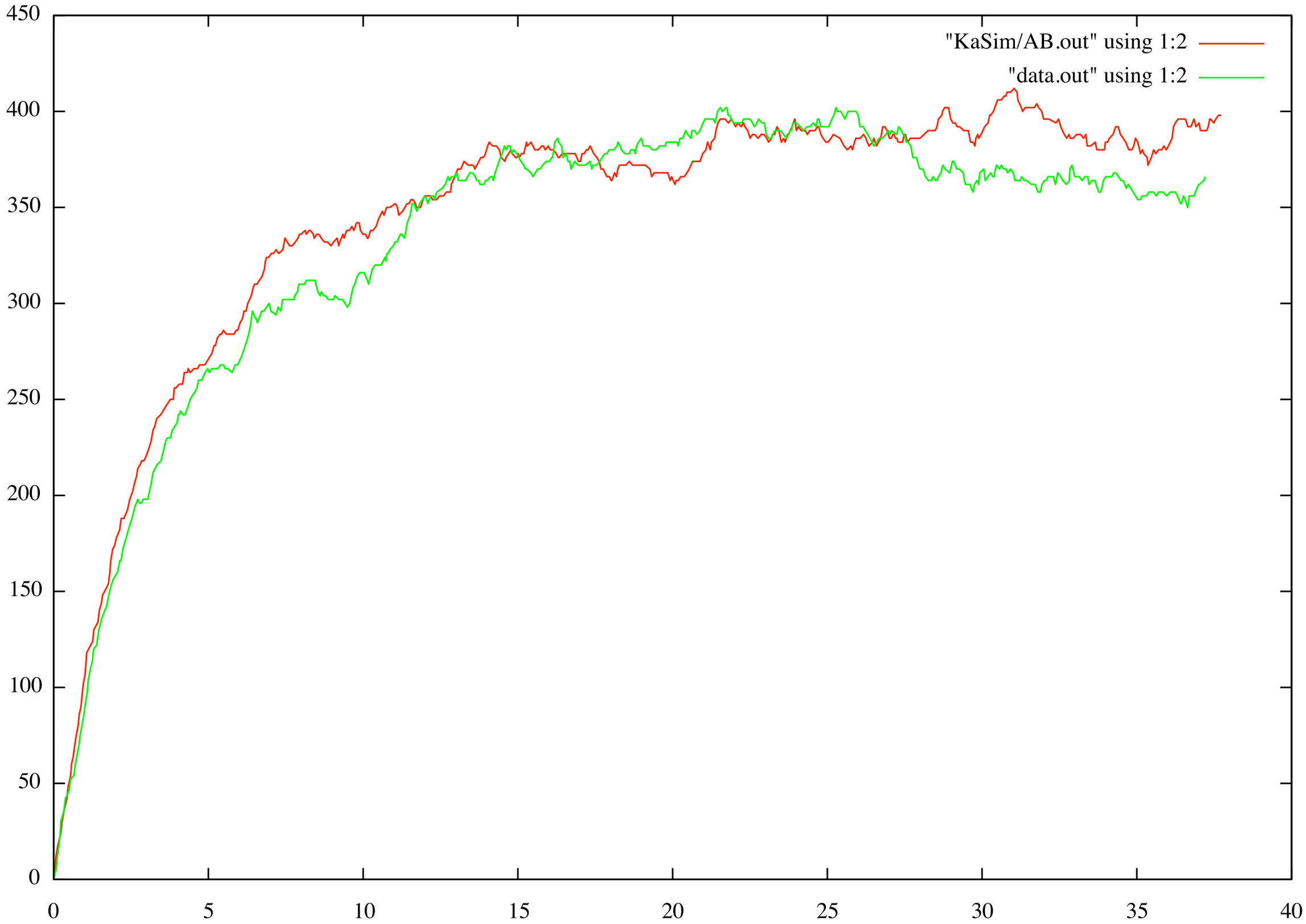
- KaSim outputs to the file data.out
  - you can change this with the -o option
- Run gnuplot from the command line
  - sh> **gnuplot**
  - gnuplot> **set term 'x11'**
- Then
  - gnuplot> **plot "data.out" using 1:2 with lines**


"data.out" using 1:2
450
400
350
300
250
200
150
100
50
0
0
5
10
15
20
25
30
35
40

Questions

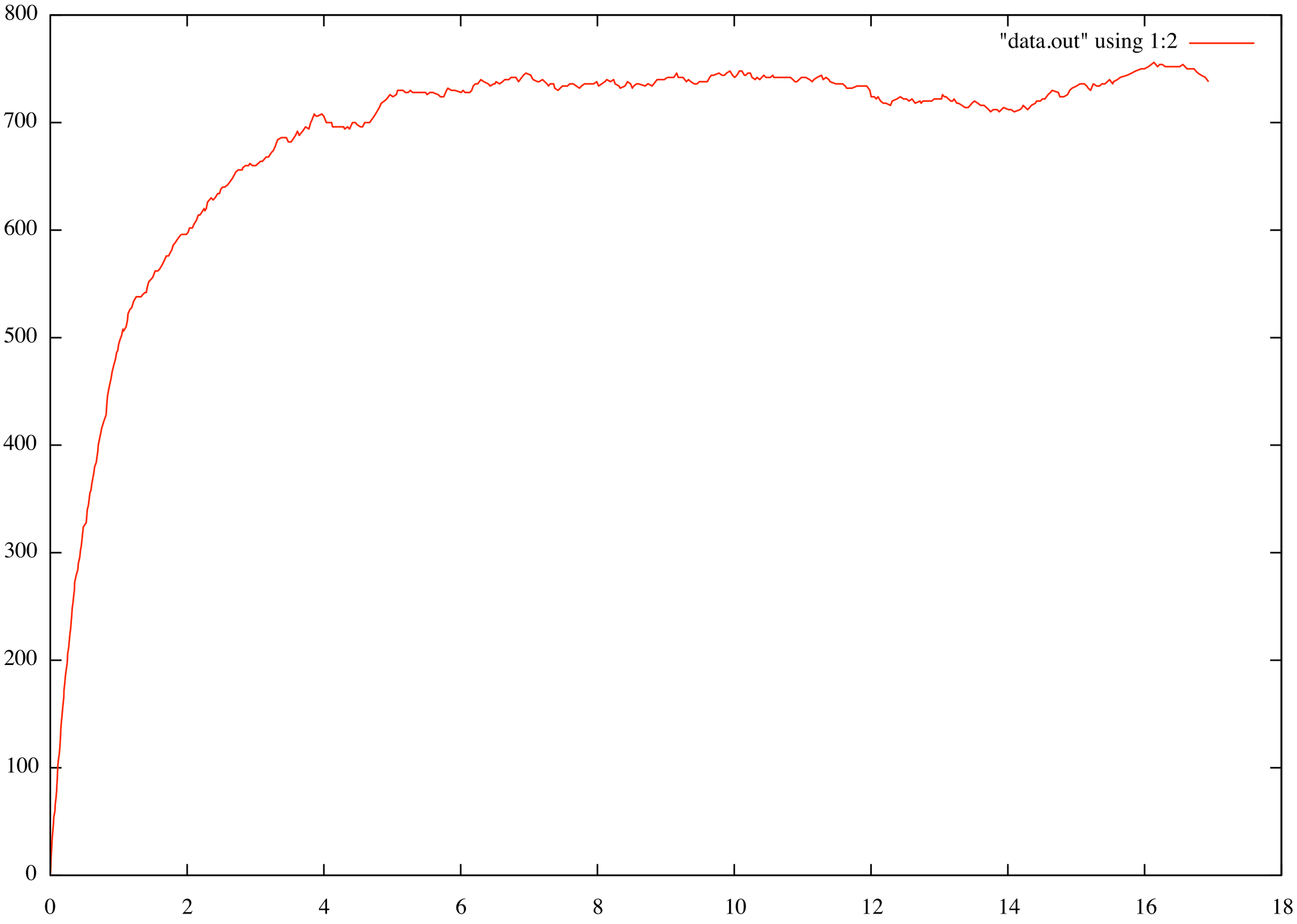
- Time to equilibrium?
- What does ‘equilibrium’ even mean in a stochastic setting like this?

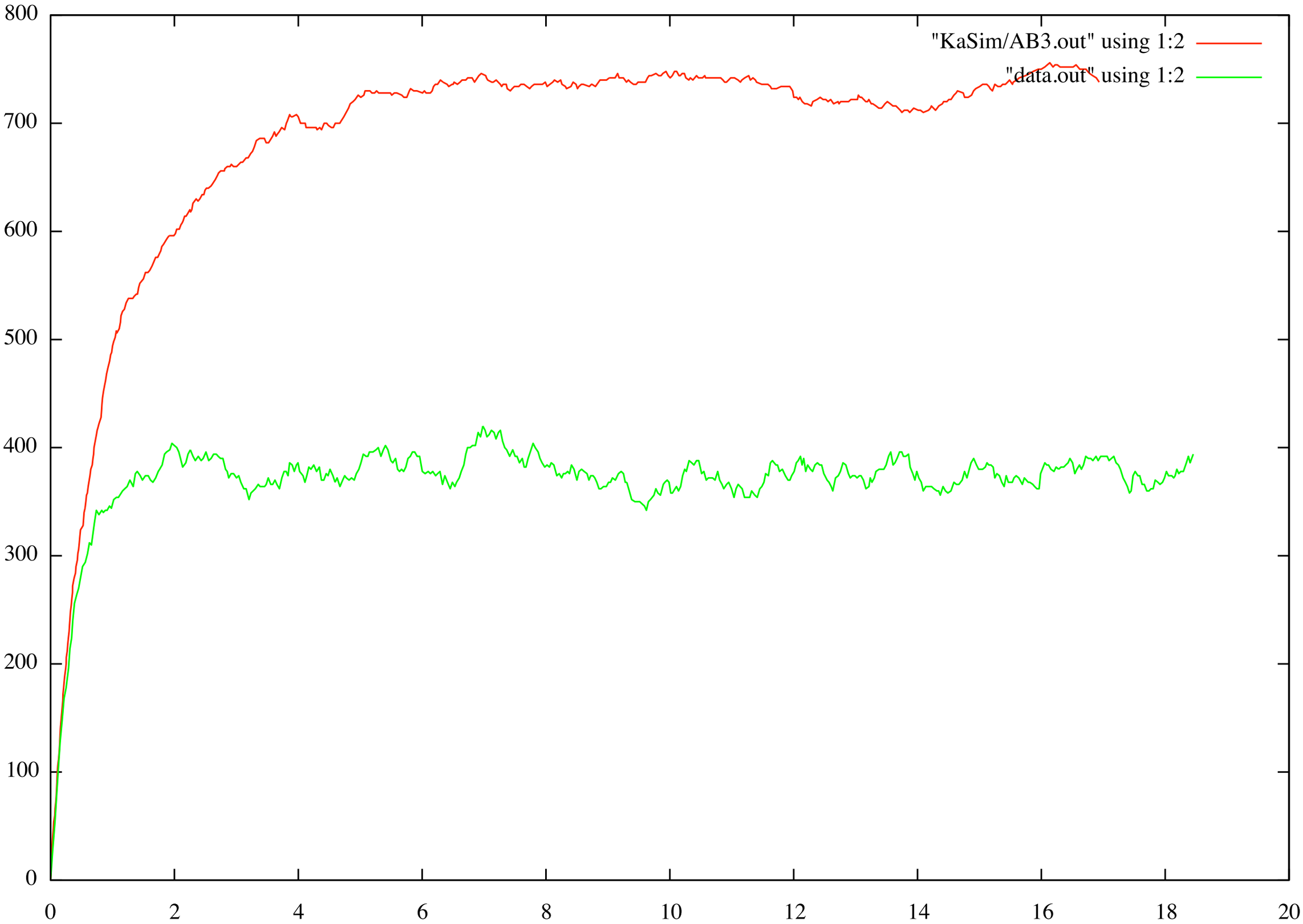
- Run the model a second time:
  - gnuplot> **replot “data.out” using 1:2 with lines**
- Run the model for longer:
  - sh> **KaSim -i AB.ka -t 100 -p 500**


"KaSim/AB.out" using 1:2
"data.out" using 1:2
450
400
350
300
250
200
150
100
50
0
0
5
10
15
20
25
30
35
40

Questions

- What happens if you
  - increase the binding rate by a factor of 10 ?
  - increase the binding *and* the unbinding rate by a factor of 10 ?


"data.out" using 1:2
800
700
600
500
400
300
200
100
0
0
2
4
6
8
10
12
14
16
18


"KaSim/AB3.out" using 1:2
"data.out" using 1:2
800
700
600
500
400
300
200
100
0
0
2
4
6
8
10
12
14
16
18
20

Briefly…

- By equilibrium thermodynamics:

$$K_d := [A]_{eq}\,[B]_{eq} / [AB]_{eq} \;\propto\; e^{-\Delta G/T}$$

- By detailed balance of the kinetic equations:

$$k_{on}\,[A]_{eq}\,[B]_{eq} = k_{off}\,[AB]_{eq}$$

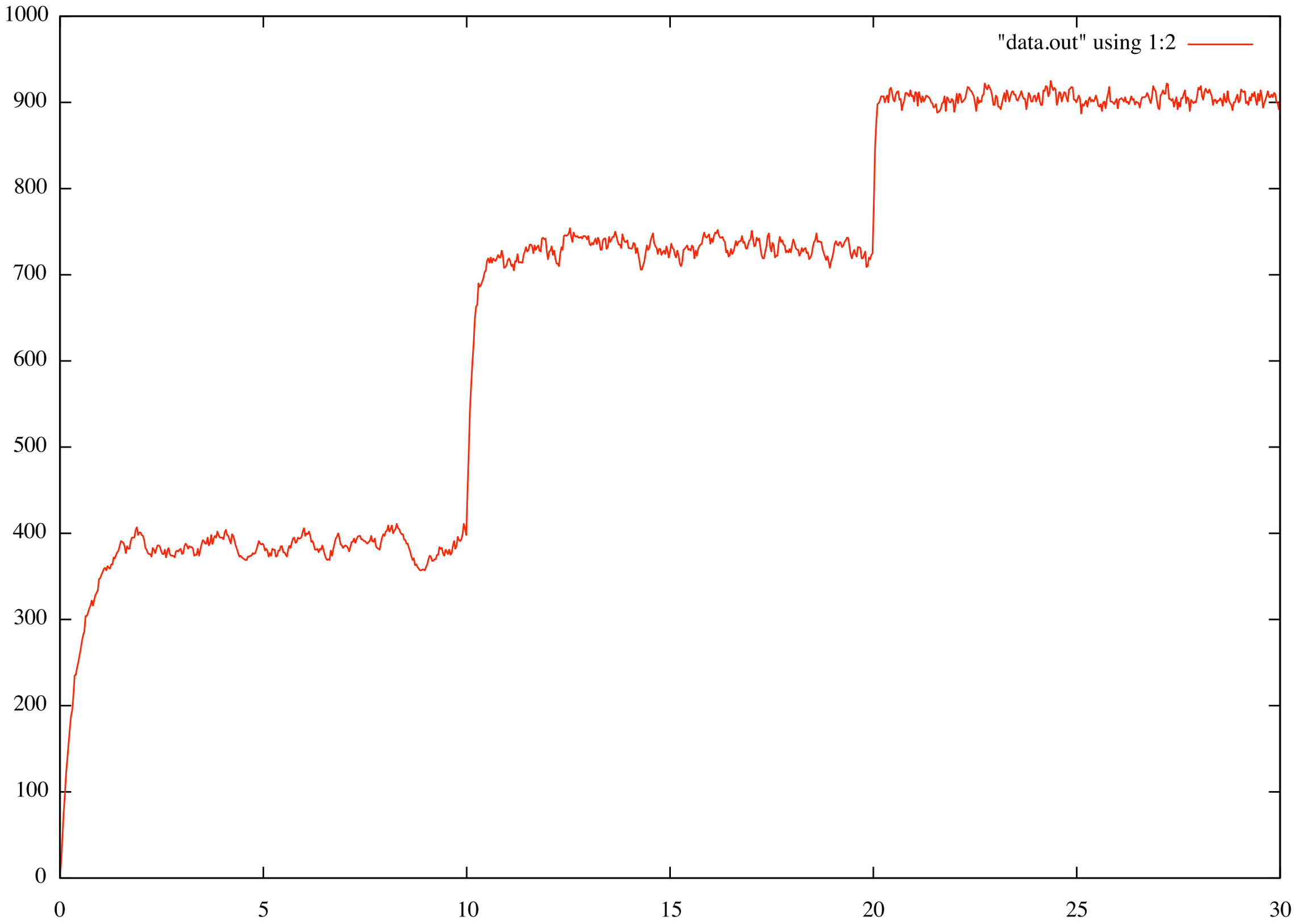
- *i.e.* $k_{off} / k_{on} = K_d \propto e^{-\Delta G/T}$

Perturbations

- Modify rate constants *during* simulation !

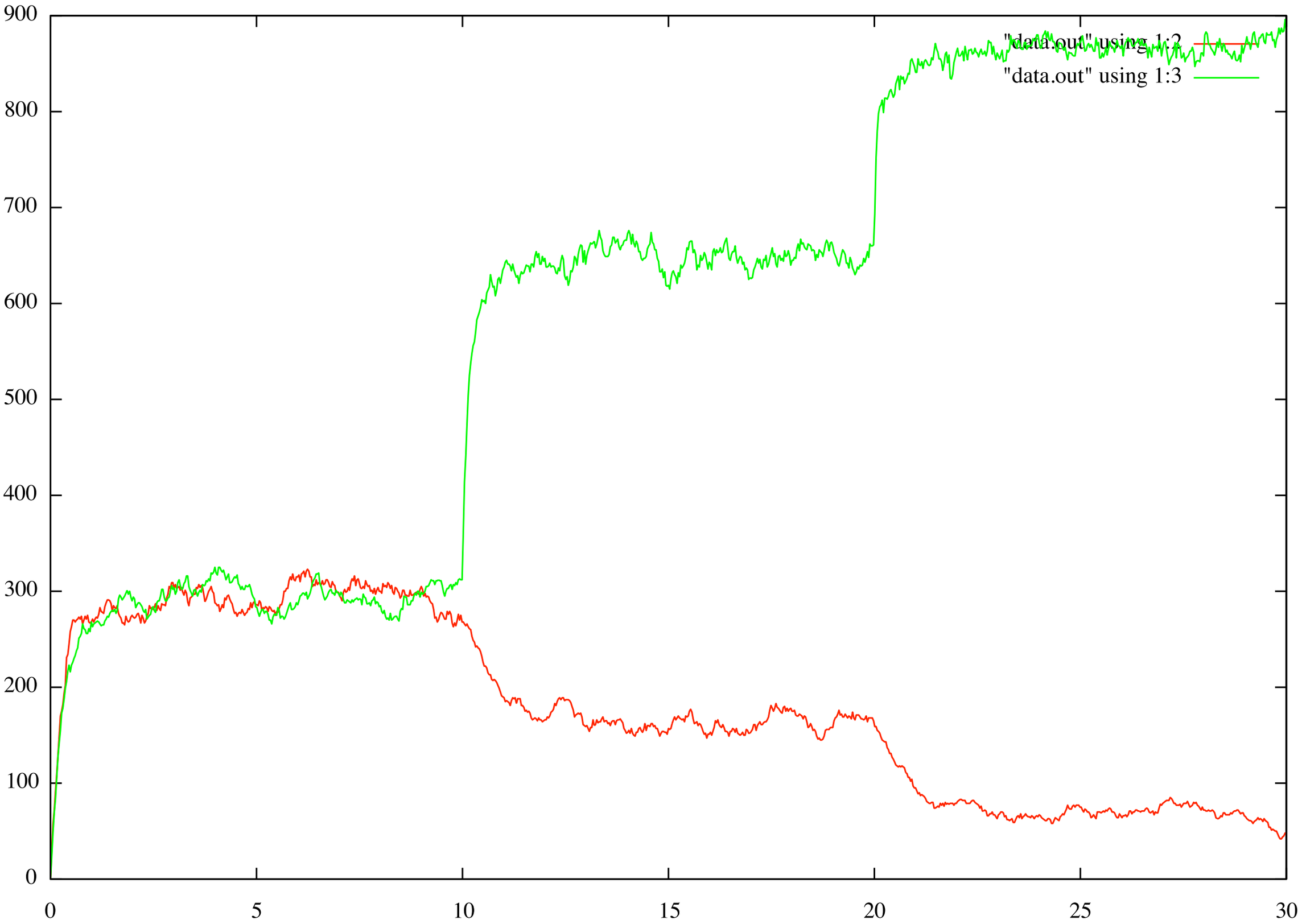
```
%mod: [T] > 10 do 'BND' := 'BND' * 10
%mod: [T] > 20 do 'BND' := 'BND' * 10
```

- sh> **KaSim -i AB.ka -t 30 -p 1000**


"data.out" using 1:2
1000
900
800
700
600
500
400
300
200
100
0
0
5
10
15
20
25
30

Questions

- What happens if you introduce a *conflict* ?
  - a new agent `C(a)`
  - rules for binding and unbinding of `C` to/from `A`


"data.out" using 1:2
"data.out" using 1:3
900
800
700
600
500
400
300
200
100
0
0
5
10
15
20
25
30

About bi-molecular rate constants

- k_{det} has dimension $conc^{-1}time^{-1}$
  - usually $M^{-1}s^{-1}$ where $M = mol / l$
  - sometimes use ‘mass concentration’, e.g. g/l
- k_{stoch} has dimension $time^{-1}$
  - k_{det} / V has units $mol^{-1}s^{-1}$
  - $k_{stoch} = k_{det} / AV$ has units $molecule^{-1}s^{-1}$

    (where V = volume in l and A = Avagadro)

About rate constants

- For eukaryotes, $AV \sim 10^{12}$
  - typical $k_{det} \sim 10^{7}$ - 10^{9} $M^{-1}s^{-1}$
  - so typical $k_{stoch} \sim 10^{-5}$ - 10^{-3} $molecule^{-1}s^{-1}$
- Unbinding is volume-independent
  - $k_{det} = k_{stoch} \sim 0.1$ s^{-1}

Rescaling
(a useful trick)

- **%var: 'vol' 1.0**
- Modify birth and binding rates and variables:

A(b),B(a) -> A(b!0),B(a!0) @ 'BND'/'vol'

-> A(b) @ 'BIRTH' * 'vol'

%init: 1000 * 'vol' A(b)

- Decreasing vol preserves system dynamics
  - increases fluctuations; speeds up simulation
- What about increasing vol ?

II. Enzyme-Substrate

Enzyme-Substrate

$$E + S \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} C \overset{k_2}{\rightarrow} E + P$$

- Agent signatures

  %agent: E(s)

  %agent: S(s~0~1)

- Observable

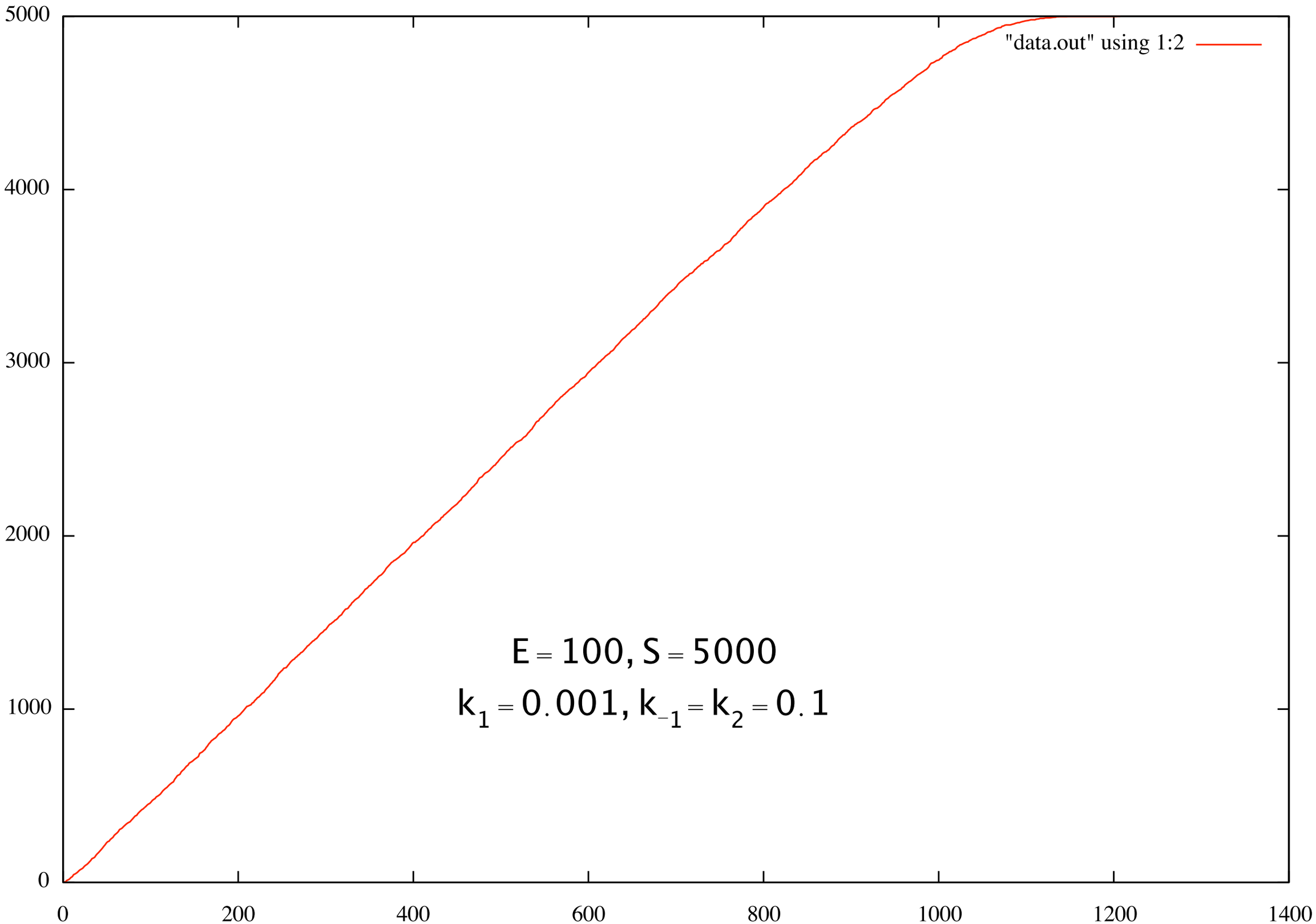
  %obs: S(s~1)

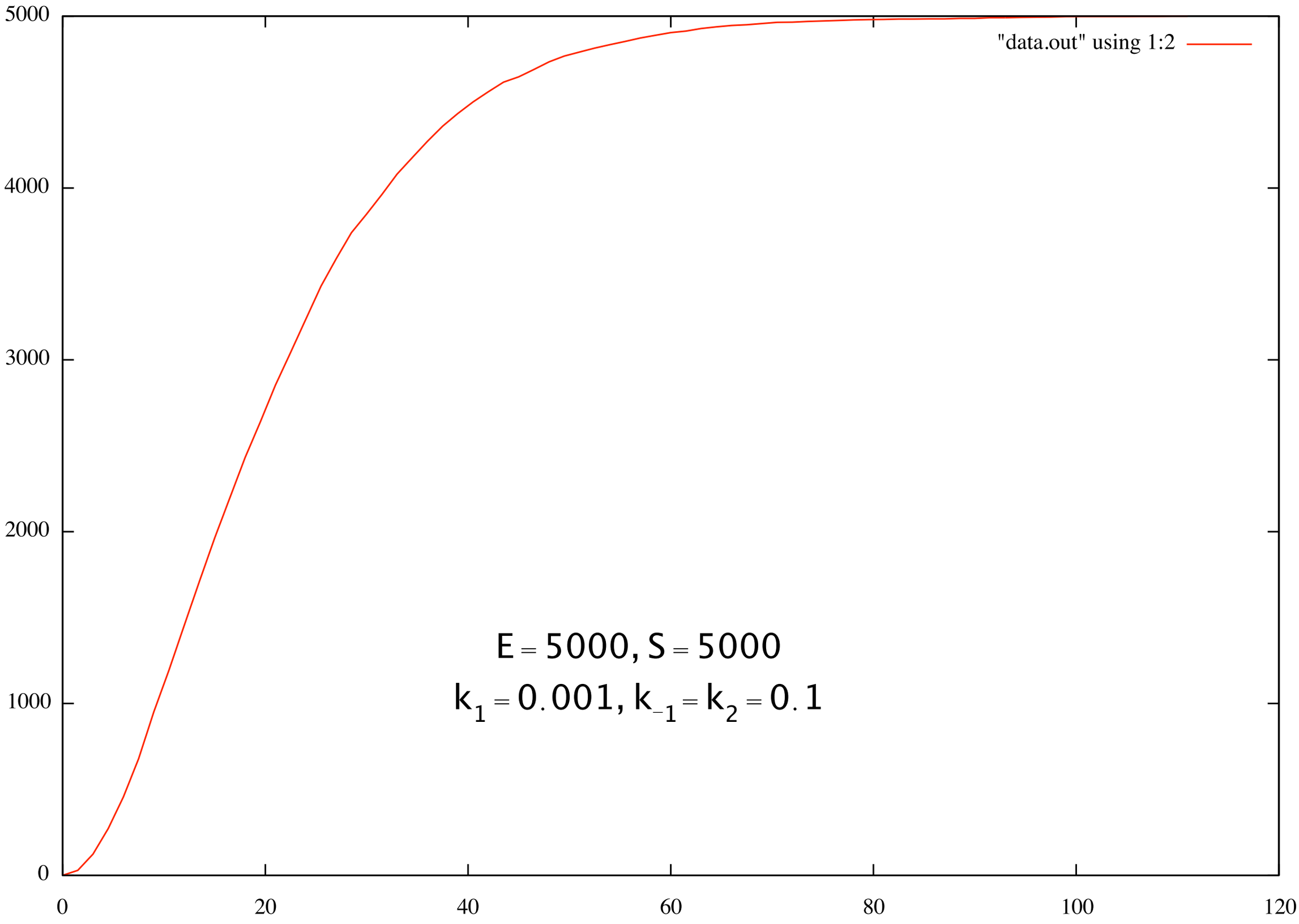
ES.ka

```
%agent: E(s)
%agent: S(s~0~1)

# E(s), S(s) -> E(s!0), S(s!0) @ 0.0001
E(s), S(s~0) -> E(s!0), S(s~0!0) @ 0.001
# E(s), S(s~1) -> E(s!0), S(s~1!0) @ 0.00001

E(s!0), S(s!0) -> E(s), S(s) @ 0.1
# E(s!0), S(s~0!0) -> E(s), S(s~0) @ 0.01
# E(s!0), S(s~1!0) -> E(s), S(s~1) @ 1.0

E(s!0), S(s~0!0) -> E(s!0), S(s~1!0) @ 0.1
# E(s!0), S(s~0!0) -> E(s), S(s~0) @ 0.1
# E(s!0), S(s~0!0) -> E(s), S(s~1) @ 0.1

%init: 5000 E(s)
%init: 5000 S(s~0)

%obs: 'free active substrate' S(s~1)
%obs: 'total active substrate' S(s~1?)
```

Questions

- Try different relative amounts of *E* and *S*
  - roughly equal
  - excess *S* (*e.g.* 50-100×*E*)
- How does this affect the rate of production of *P* ?
  - saturation ?


"data.out" using 1:2
5000
4000
3000
2000
1000
0
0
200
400
600
800
1000
1200
1400
E = 100, S = 5000
k1 = 0.001, k−1 = k2 = 0.1


"data.out" using 1:2
5000
4000
3000
2000
1000
0
0
20
40
60
80
100
120
E = 5000, S = 5000
k1 = 0.001, k−1 = k2 = 0.1

Variants

- Easy to build small variants of a model
  - rules that allow *E* to bind to *P* [*product inhibition*]
  - make unbinding sensitive to the state of *S*
  - `E(s!0), S(s~0!0) -> E(s), S(s~1)` [*MOD* + *BRK*]

QSSA

- Corresponding system of ODEs cannot be solved analytically:

$$\begin{aligned}
\frac{d}{dt}[E] &= -k_1[E][S] + (k_{-1} + k_2)[C] \\
\frac{d}{dt}[S] &= -k_1[E][S] + k_{-1}[C] \\
\frac{d}{dt}[C] &= k_1[E][S] - (k_{-1} + k_2)[C] \\
\frac{d}{dt}[P] &= k_2[C]
\end{aligned}$$

- What can we say about the rate of the *overall* reaction ?

QSSA

QSSA [quasi-steady state approximation]:

$$\frac{d}{dt}[C] \approx 0$$

after a brief transient. This remains valid *until* substrate S is significantly consumed. (Can be formalized.)

We can now express:

$$\frac{d}{dt}[P] \approx \frac{k_2 E_{tot}[S]}{K_M + [S]} \qquad \text{(why?)}$$

$$(K_M := \frac{k_{-1}+k_2}{k_1})$$

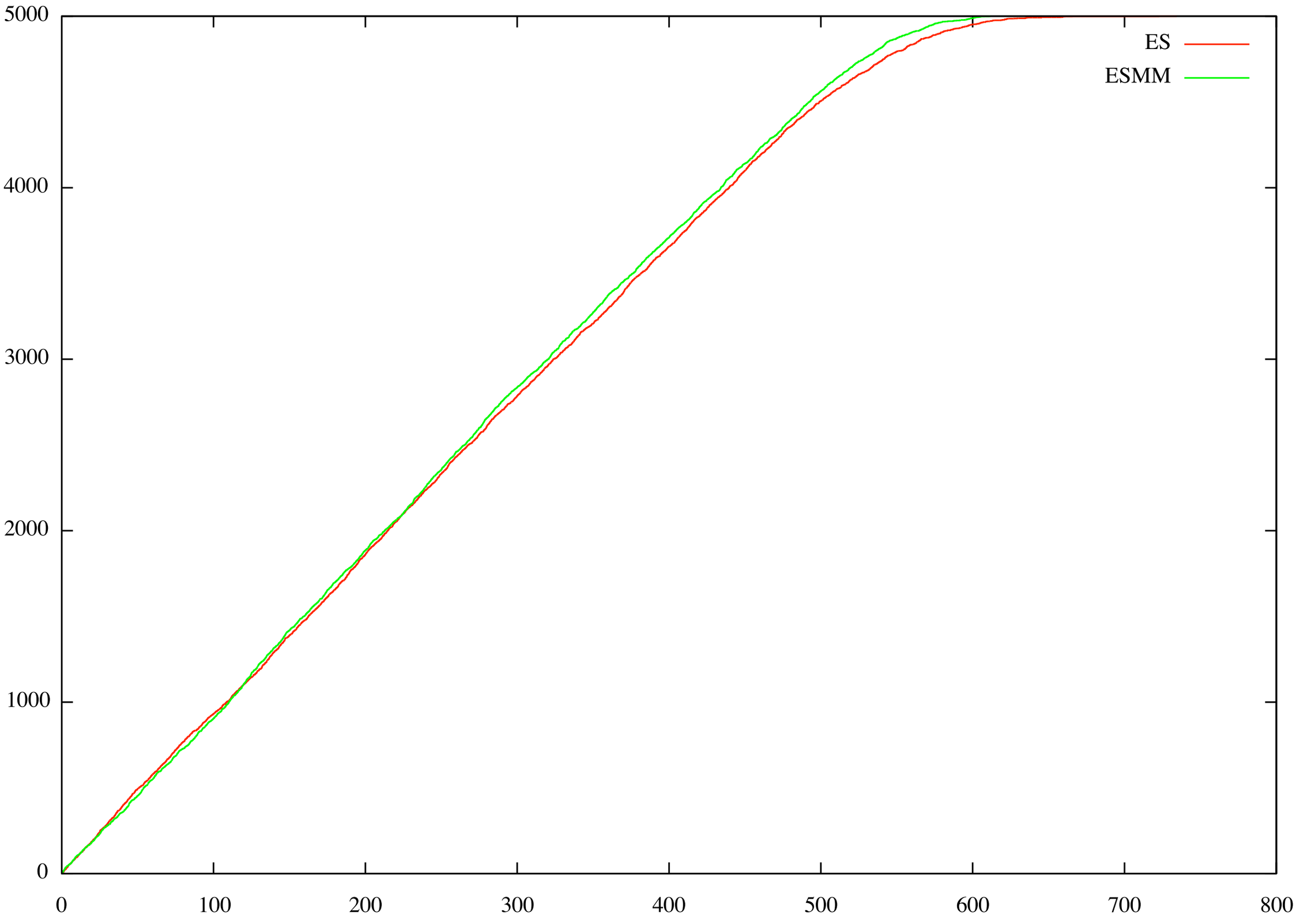
QSSA

```
E(s),S(s~0) -> E(s),S(s~1) @ ‘k2’ / (‘KM’ + ‘S0’)

%var: ‘k1’ 0.0001
%var: ‘k-1’ 0.1
%var: ‘k2’ 0.1

%var: ‘KM’ (‘k-1’ + ‘k2’) / ‘k1’
%var: ’S0’ S(s~0)
```

Rate laws in Kappa always include the LHS so the above really is Michaelis-Menten:

$$\frac{k_2 E_{tot}[S]}{K_M + [S]}$$

http://www.pps.univ-paris-diderot.fr/~russ/AIV/ESMM.ka



QSSA

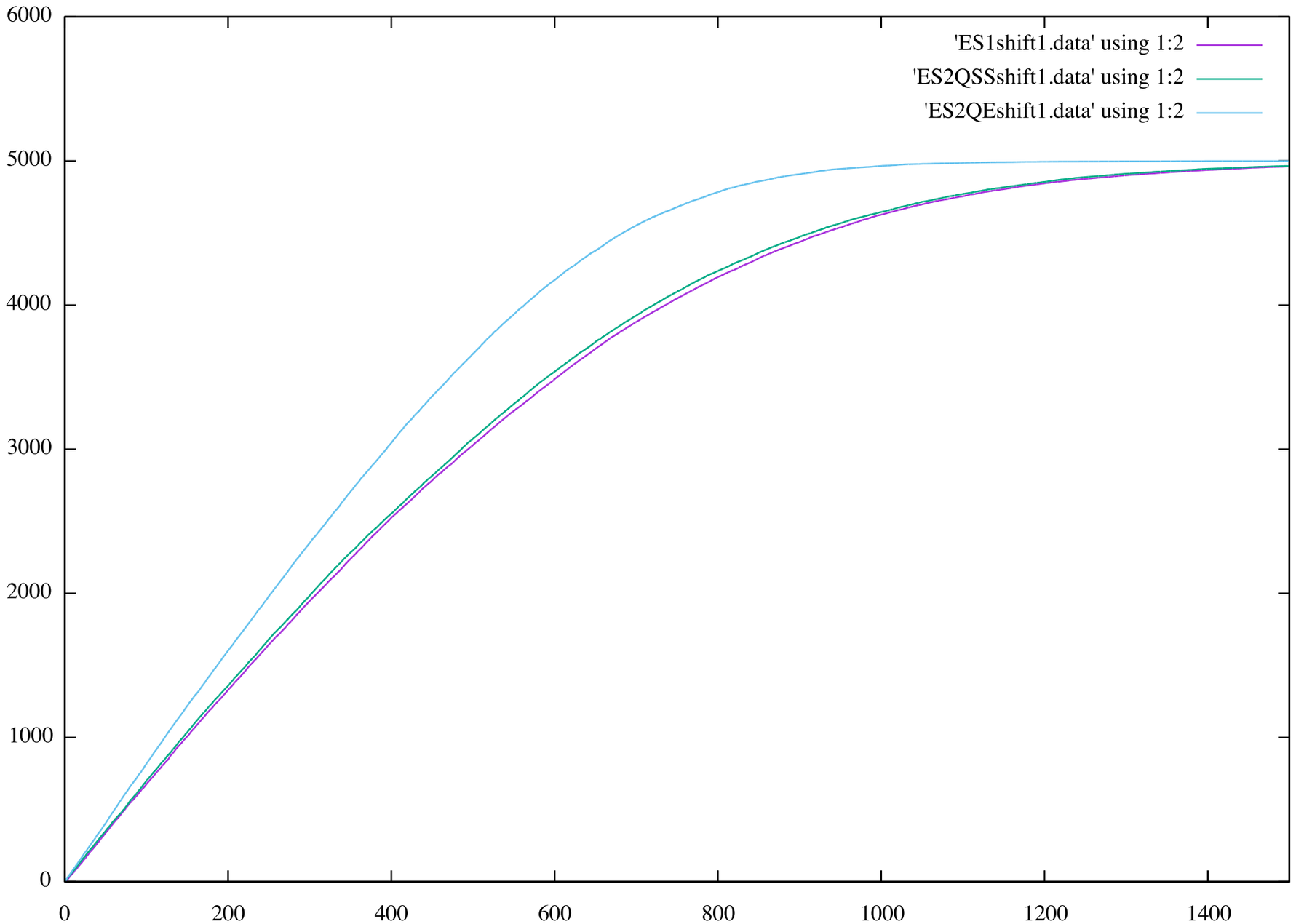
- A very good approximation under appropriate conditions; see
  - Lee Segel. On the validity of the steady state assumption of enzyme kinetics. BMB, 1988.
  - Lee Segel and Marshall Slemrod. The quasi steady state assumption: a case study in perturbation. SIAM review, 1989.

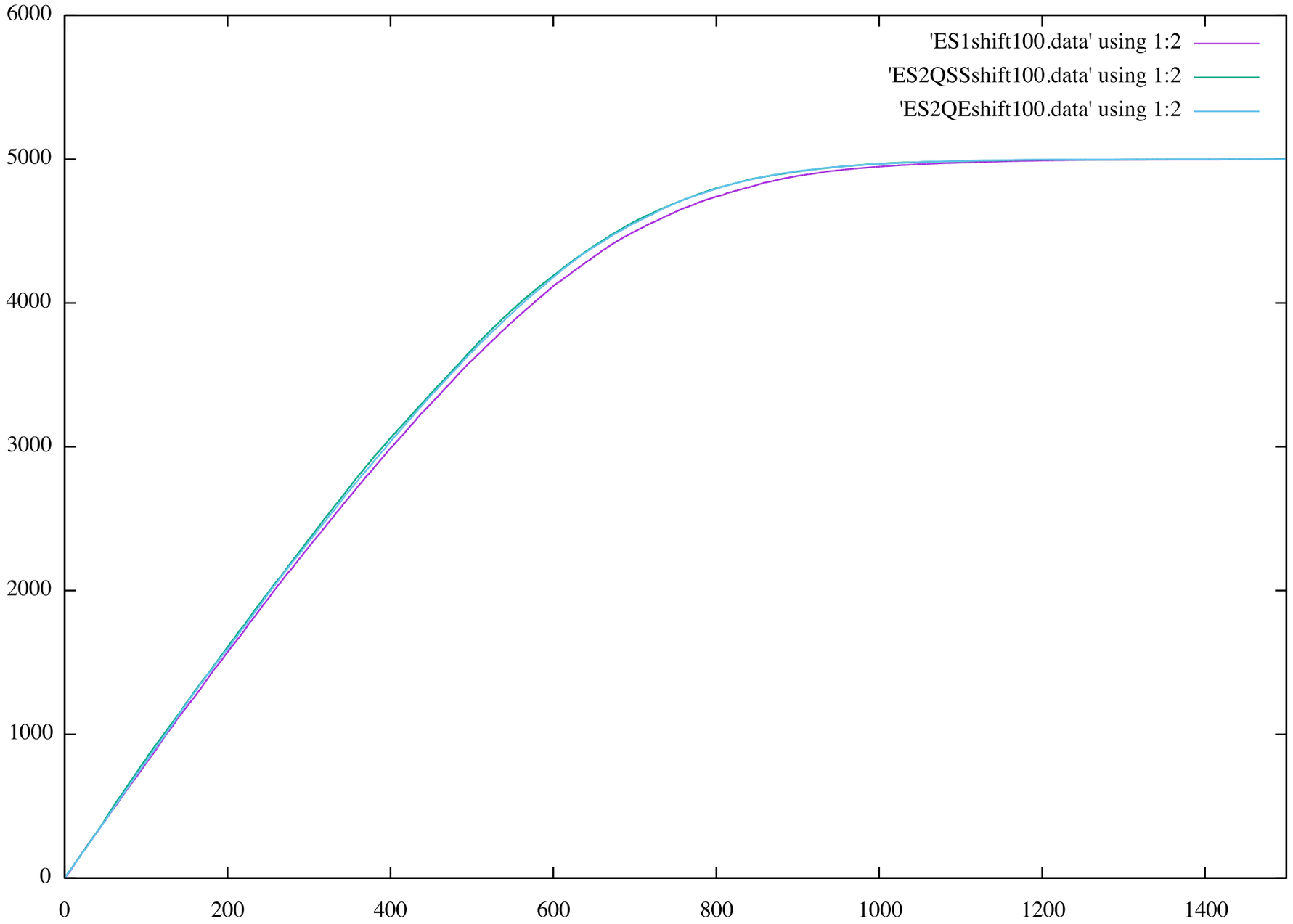
Quasi-equilibrium

- A different approximation
  - leads to the same equation for the time evolution of P
  - but with a different constant: $K_d := k_{-1} / k_1$

Quasi-equilibrium

```
# standard enzyme-substrate system: irreversible modification coupled with unbinding

%var: 'rescale' 10
%var: 'shift' 10 # speed up the equilibrium

# rate constant values
%var: 'BND' 0.0001 * 'shift'
%var: 'BRK' 0.1 * 'shift'
%var: 'MOD' 0.01

# agents
%agent: E(s)
%agent: S(s~0~1)

# binding
E(s), S(s~0) -> E(s!0), S(s~0!0) @ 'BND' / 'rescale'

# unbinding (S0 only)
E(s!0), S(s~0!0) -> E(s), S(s~0) @ 'BRK'

# modification (and S1-unbinding)
E(s!0), S(s~0!0) -> E(s), S(s~1) @ 'MOD'

# initial conditions
%var: 'nE' 100 * 'rescale'
%var: 'nS' 5000 * 'rescale'
%init: 'nE' E(s)
%init: 'nS' S(s~0)

# observables
%var: 'C' |E(s!1), S(s~0!1)|
%var: 'P' |S(s~1)|
%var: 'BNDact' 'BND' * |E(s)| * |S(s~0)| / 'rescale'
%var: 'BRKact' 'BRK' * 'C'
%var: 'MODact' 'MOD' * 'C'
%plot: 'P' / 'rescale'
%plot: 'C' / 'rescale'
%plot: 'BNDact' / 'rescale'
%plot: 'BRKact' / 'rescale'
%plot: 'MODact' / 'rescale'
```


'ES1shift1.data' using 1:2
'ES2QSSshift1.data' using 1:2
'ES2QEshift1.data' using 1:2
6000
5000
4000
3000
2000
1000
0
0
200
400
600
800
1000
1200
1400


'ES1shift100.data' using 1:2
'ES2QSSshift100.data' using 1:2
'ES2QEshift100.data' using 1:2
6000
5000
4000
3000
2000
1000
0
0
200
400
600
800
1000
1200
1400

Rule refinement

- A rule does *not* have to mention all the sites of the agents in the rule
  - many possible *contexts* may be possible
    - *e.g.* ‘E’ can unbind ’S’ in state 0 *or* 1
- Refine a rule by adding some (or all) of the ‘missing’ context
  - can evaluate the relative importance of sub-cases
  - can introduce kinetic subtleties: co-operativity...

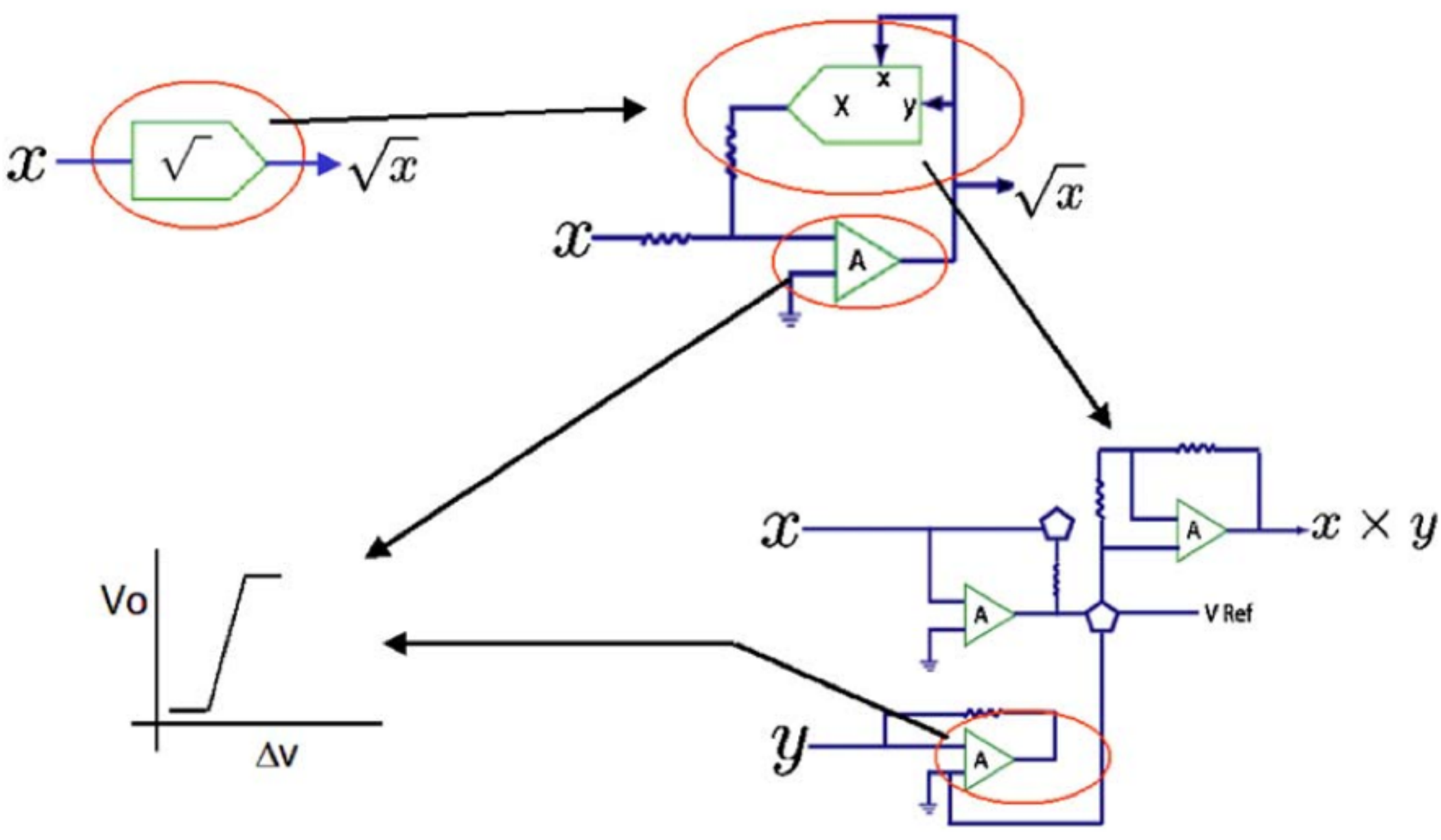
Neutral refinement

- A refinement is neutral if the refined rules collectively behave *exactly* like the original rule
  - *all* sub-cases have the *same* rate constant
- Provides a baseline against which kinetic adjustments can be made
  - *e.g.* kinases tend to have lower affinity for their products than their substrates

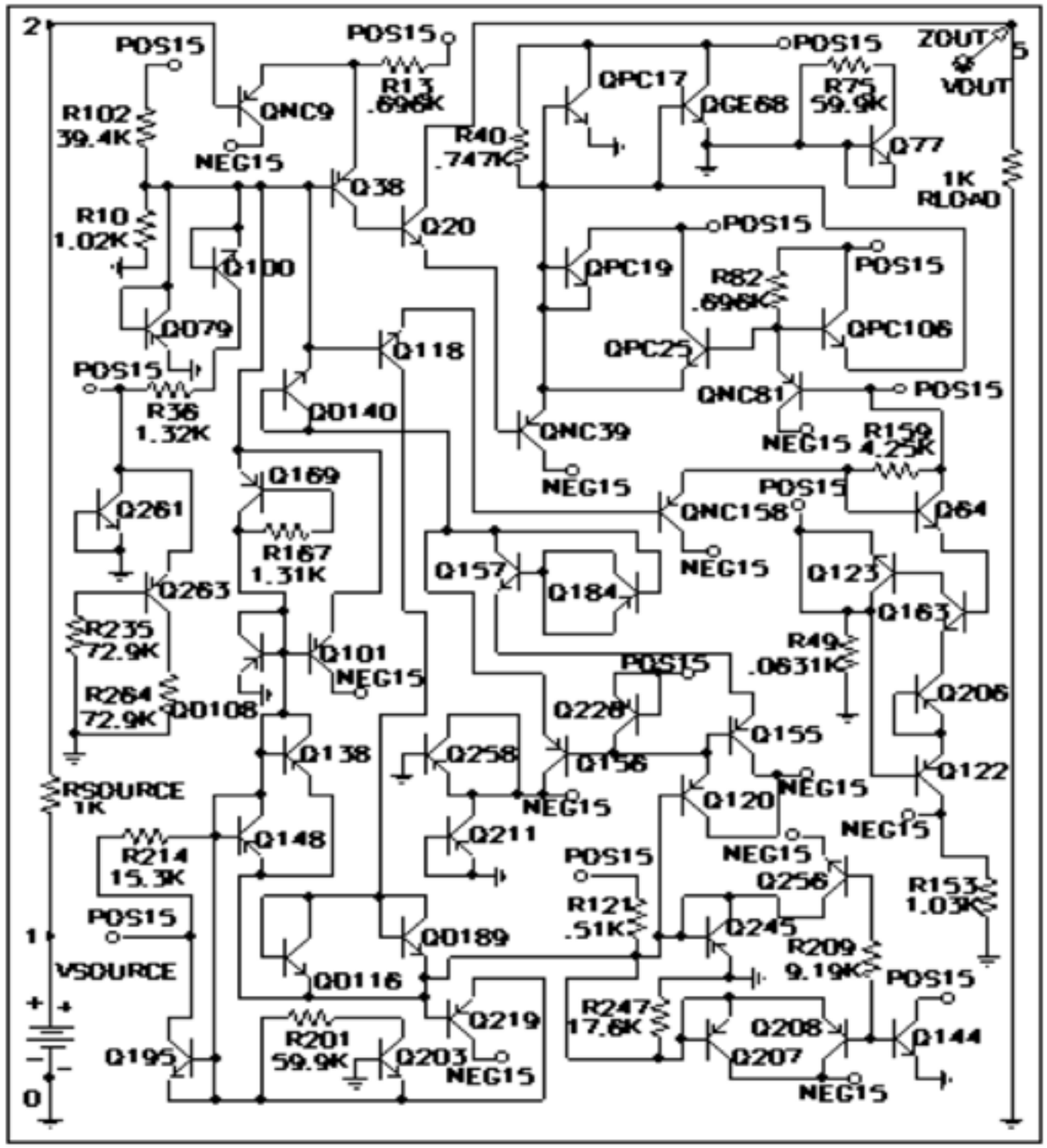
A different viewpoint

- Refinements as (small) perturbations of a rule
  - expose some previously ‘hidden’ bit of context to enable ‘kinetic adjustment’
  - can only be done in a rule-based setting
- A plausible mechanism by which a signaling network could be subject to selection
  - increases the number of ‘tunable parameters’
  - could give rise to very opaque systems…

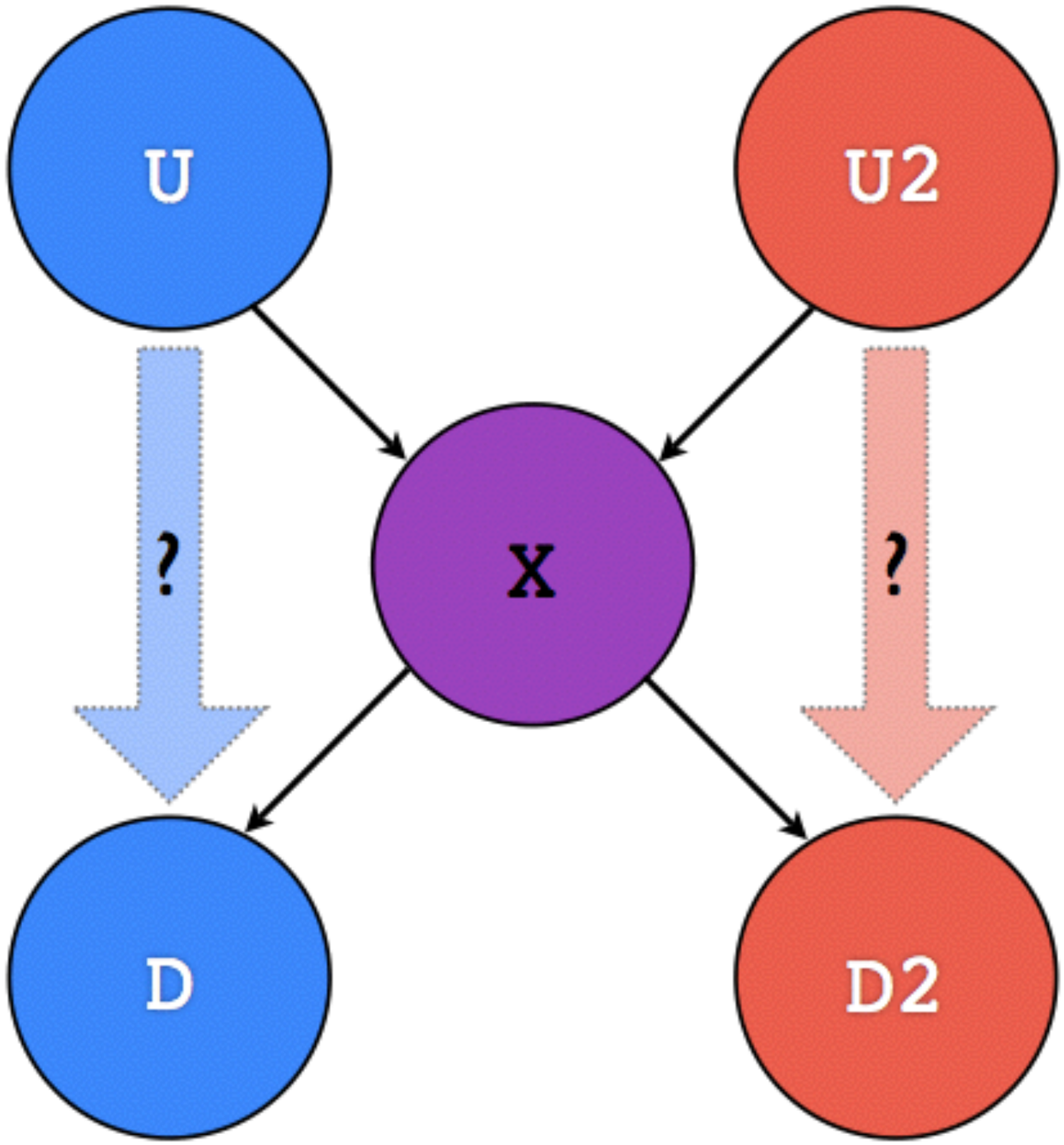
ID



Evolution


2
POS15
POS15
R13
.696K
R102
39.4K
QNC9
NEG15
Q38
R40
.747K
QPC17
POS15
QGE68
R75
59.9K
ZOUT
VOUT
5
Q77
1K
RLOAD
R10
1.02K
Q20
POS15
Q100
QPC19
R82
.696K
POS15
Q079
QPC106
QPC25
Q118
POS15
R36
1.32K
QNC81
POS15
Q0140
QNC39
NEG15
R159
4.25K
NEG15
POS15
Q169
QNC158
Q64
Q261
R167
1.31K
NEG15
Q157
Q184
Q123
Q263
Q163
R235
72.9K
Q101
NEG15
R49
.0631K
POS15
R264
72.9K
Q0108
Q228
Q206
Q155
Q258
Q156
Q138
Q122
RSOURCE
1K
NEG15
Q120
NEG15
Q211
R214
15.3K
Q148
POS15
NEG15
Q256
R153
1.03K
R121
.51K
POS15
Q0189
Q245
1
VSOURCE
R209
9.19K
POS15
Q0116
R247
17.8K
Q208
Q144
Q219
R201
59.9K
Q203
Q195
Q207
NEG15
0
NEG15

A specificity puzzle


U
U2
?
X
?
D
D2

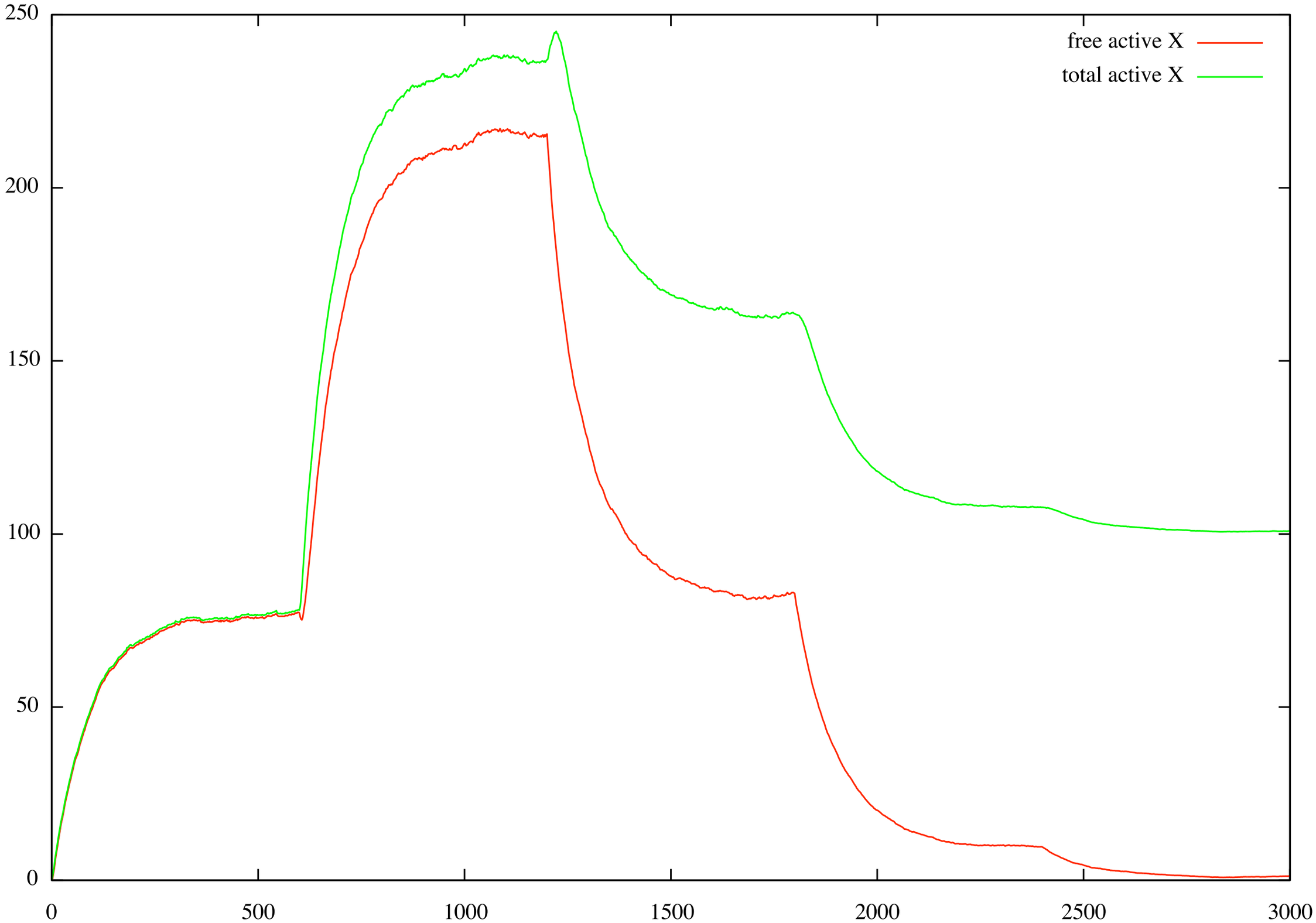
The rules!

- U1 and U2 bind the *same* site s of X
- D1 and D2 bind the same site t of X
- X *cannot* distinguish U1 and U2, nor D1 and D2
- Specificity:
  - U1 should activate D1
  - U2 should activate D2
  - some ‘leakage’ is permitted (not too much!)

```
%agent: U(s,u~1~2)
%agent: X(s~0~1,t)
%agent: D(t~0~1,d~1~2)

%var: 'vol' 10.0
%var: 'BND' 0.0001
%var: 'BRK' 0.1
%var: 'MOD' 0.1
%var: 'nU' 100*'vol'
%var: 'nX' 1000*'vol'
%var: 'nD' 1000*'vol'

U(s), X(s~0) -> U(s!0), X(s~0!0) @ 'BND'/'vol'
'U_X_op' U(s!0), X(s!0) -> U(s), X(s) @ 1.0
U(s!0), X(s~0!0) -> U(s!0), X(s~1!0) @ 'MOD'
X(s~1) -> X(s~0) @ 0.01

%init: 'nU' U #(s,u~2)
%init: 'nX' X

%var: 'X?' X(s~1?) # total active X
%var: 'X' X(s~1) # free active X
%obs: 'total active X' 'X?'/'vol'
%obs: 'free active X' 'X'/'vol'

%mod: [T]>600 do 'U_X_op':=0.1
%mod: [T]>1200 do 'U_X_op':=0.01
%mod: [T]>1800 do 'U_X_op':=0.001
%mod: [T]>2400 do 'U_X_op':=0.0001
```

What do you think?

- The perturbations gradually reduce the rate of unbinding of U and X, i.e. the system gets *stickier* over time
- How does this affect the amount of active X?
  - and what about *free* active X?

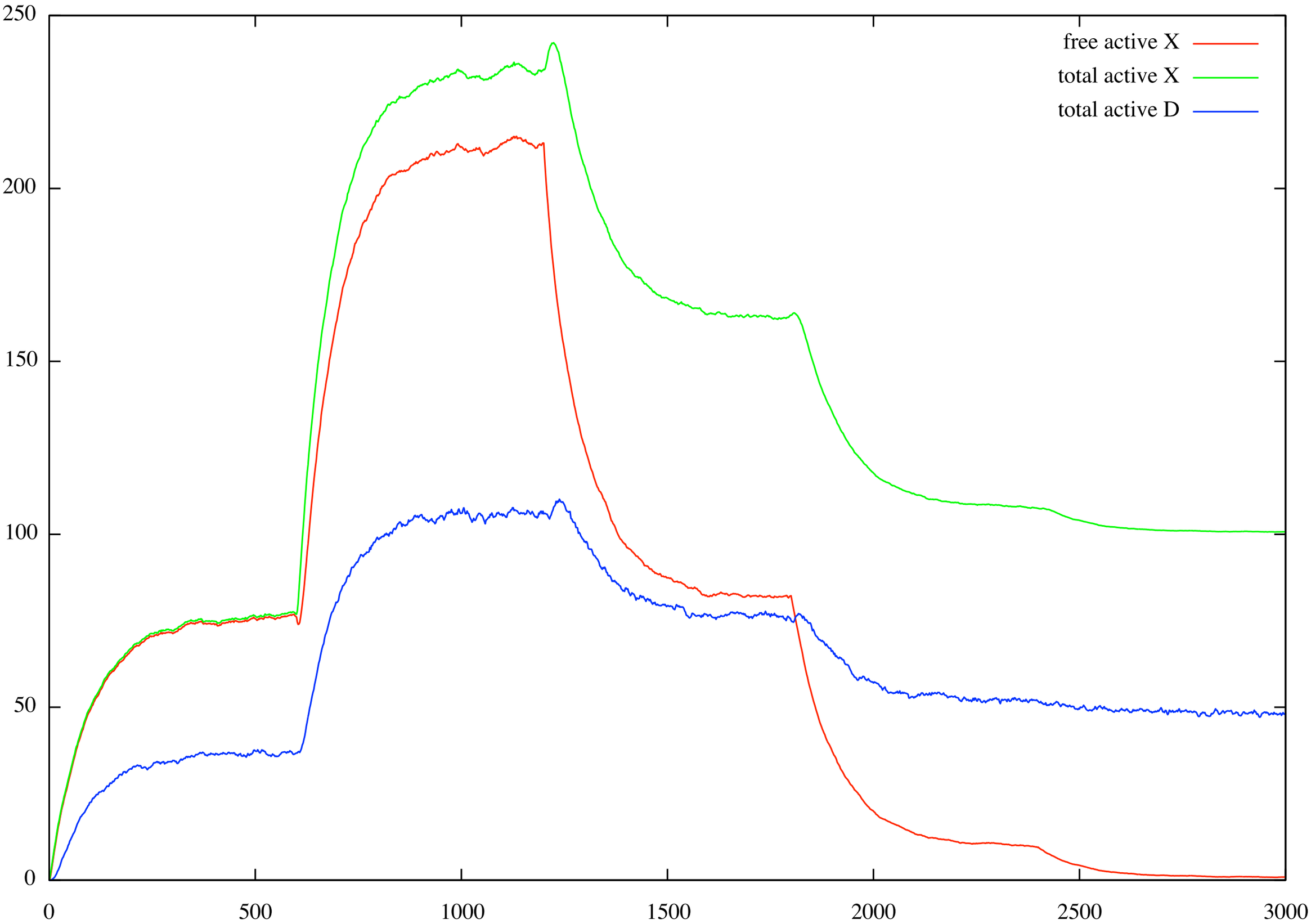

250
200
150
100
50
0
0
500
1000
1500
2000
2500
3000
free active X
total active X

Let’s add D !

- Write 3 rules expressing:
  - active X, bound or not to U, can bind inactive D
    - X and D can unbind
  - active X, bound or not to U, can activate D
- Note: `X(s~1?)` means ‘`X` with site `s` in state `1` but *unspecified* binding status
- Add a 4th rule

  `D(t~1) -> D(t~0) @ 0.1`

```
%agent: U(s,u~1~2)
%agent: X(s~0~1,t)
%agent: D(t~0~1,d~1~2)

%var: 'vol' 10.0
%var: 'BND' 0.0001
%var: 'BRK' 0.1
%var: 'MOD' 0.1
%var: 'nU' 100*'vol'
%var: 'nX' 1000*'vol'
%var: 'nD' 1000*'vol'

U(s), X(s~0) -> U(s!0), X(s~0!0) @ 'BND'/'vol'
'U_X_op' U(s!0), X(s!0) -> U(s), X(s) @ 1.0
U(s!0), X(s~0!0) -> U(s!0), X(s~1!0) @ 'MOD'

X(s~1) -> X(s~0) @ 0.01

X(s~1?,t), D(t~0) -> X(s~1?,t!0), D(t~0!0) @ 'BND'/'vol'
X(t!0), D(t!0) -> X(t), D(t) @ 'BRK'
X(s~1?,t!0), D(t~0!0) -> X(s~1?,t!0), D(t~1!0) @ 'MOD'

D(t~1) -> D(t~0) @ 0.1

%init: 'nU' U
%init: 'nX' X
%init: 'nD' D

%var: 'X?' X(s~1?) # total active X
%var: 'X' X(s~1) # free active X
%obs: 'total active X' 'X?'/'vol'
%obs: 'free active X' 'X'/'vol'
```


free active X
total active X
total active D
250
200
150
100
50
0
0
500
1000
1500
2000
2500
3000

Specificity?

- How can we obtain the desired specificity?
  - X cannot distinguish U1 from U2
  - but something has to be different...

Specificity?

- How can we obtain the desired specificity?
  - X cannot distinguish U1 from U2
  - but something has to be different...
- What about their binding affinities?

Specificity?

- How can we obtain the desired specificity?
  - X cannot distinguish U1 from U2
    - but something has to be different...
- What about their binding affinities?
  - U1 could have low affinity
  - and U2 have high affinity...

```
%agent: U(s,u~1~2)
%agent: X(s~0~1,t)
%agent: D(t~0~1,d~1~2)

%var: 'vol' 100.0
%var: 'BND' 0.0001
%var: 'BRK' 0.1
%var: 'MOD' 0.1
%var: 'nU' 100*'vol'
%var: 'nX' 1000*'vol'
%var: 'nD' 500*'vol'

U(s), X(s~0) -> U(s!0), X(s~0!0) @ 'BND'/'vol'
U(s!0,u~1), X(s!0) -> U(s,u~1), X(s) @ 1.0
U(s!0,u~2), X(s!0) -> U(s,u~2), X(s) @ 0.0001
U(s!0), X(s~0!0) -> U(s!0), X(s~1!0) @ 'MOD'

X(s~1) -> X(s~0) @ 0.01

X(s~1,t), D(t~0,d~1) -> X(s~1,t!0), D(t~0!0,d~1) @ 'BND'/'vol'
X(s~1!_,t), D(t~0,d~2) -> X(s~1!_,t!0), D(t~0!0,d~2) @ 'BND'/'vol'
X(t!0), D(t!0) -> X(t), D(t) @ 'BRK'
X(s~1?,t!0), D(t~0!0) -> X(s~1?,t!0), D(t~1!0) @ 'MOD'

D(t~1) -> D(t~0) @ 0.1

%init: 'nU' U # U1
%init: 'nU' U(s,u~2) # U2
%init: 'nX' X
%init: 'nD' D # D1
%init: 'nD' D(t~0,d~2) # D2
```

Liquid

Sticky

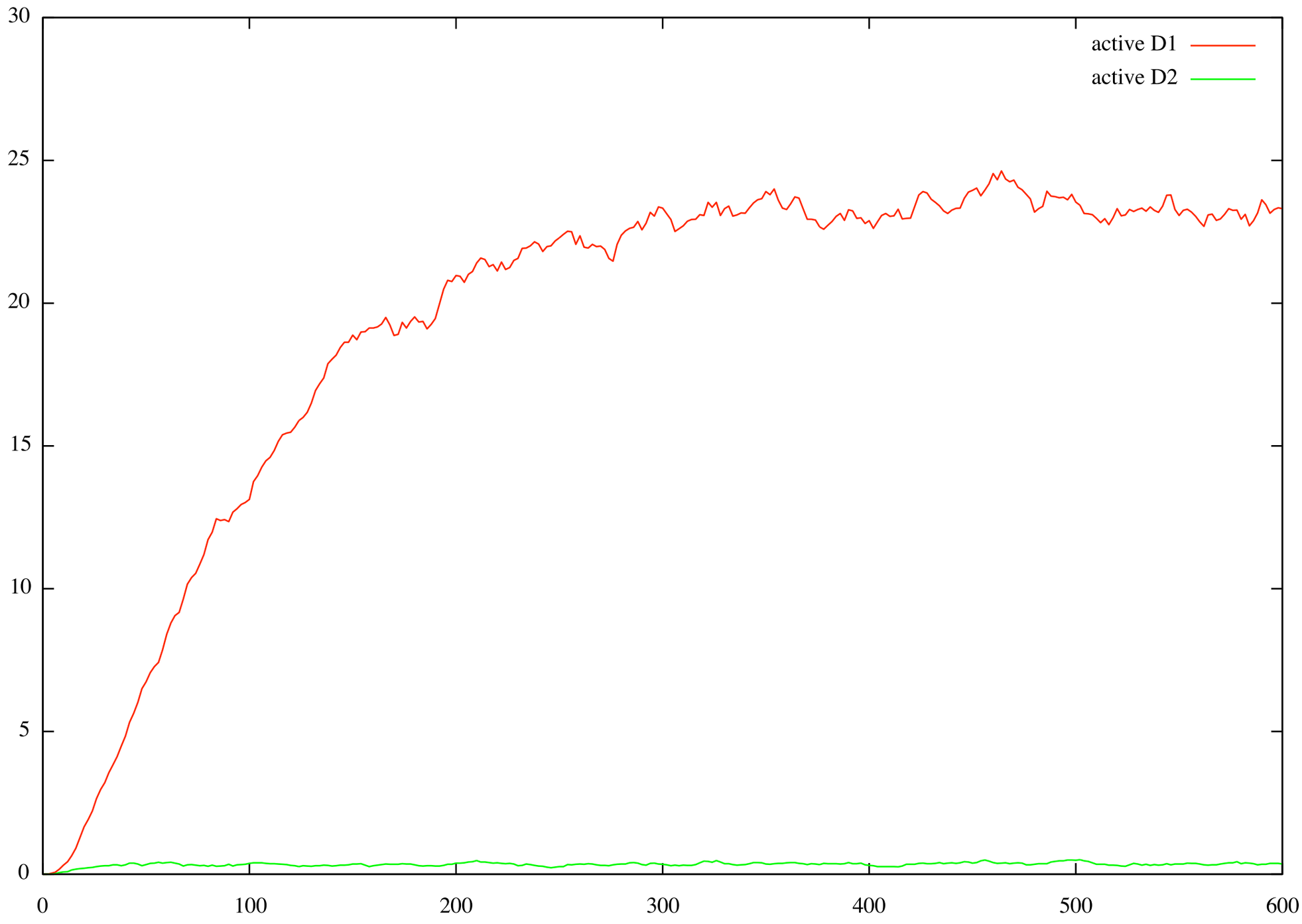
Free

Bound

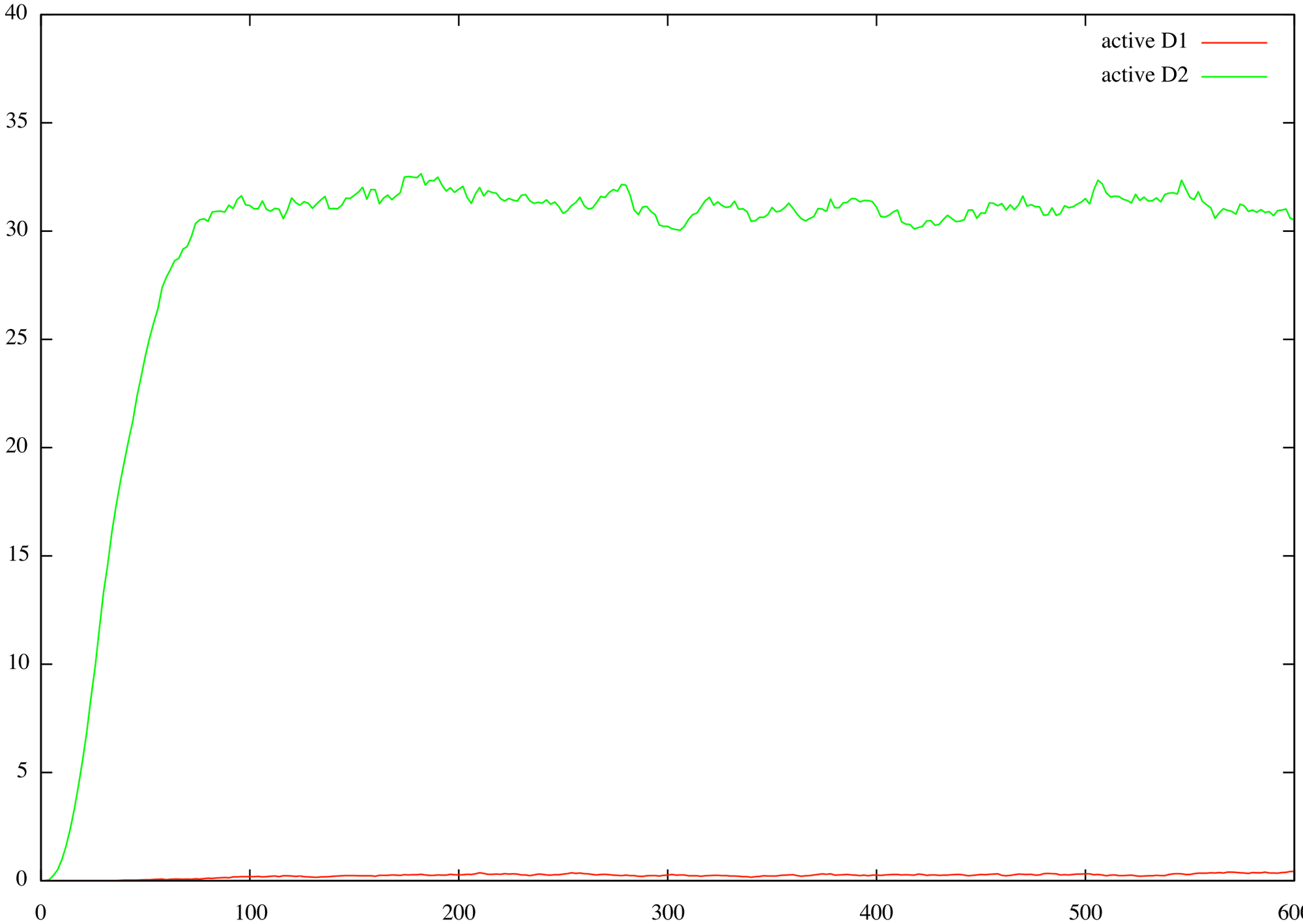
D1

D2

U1 only



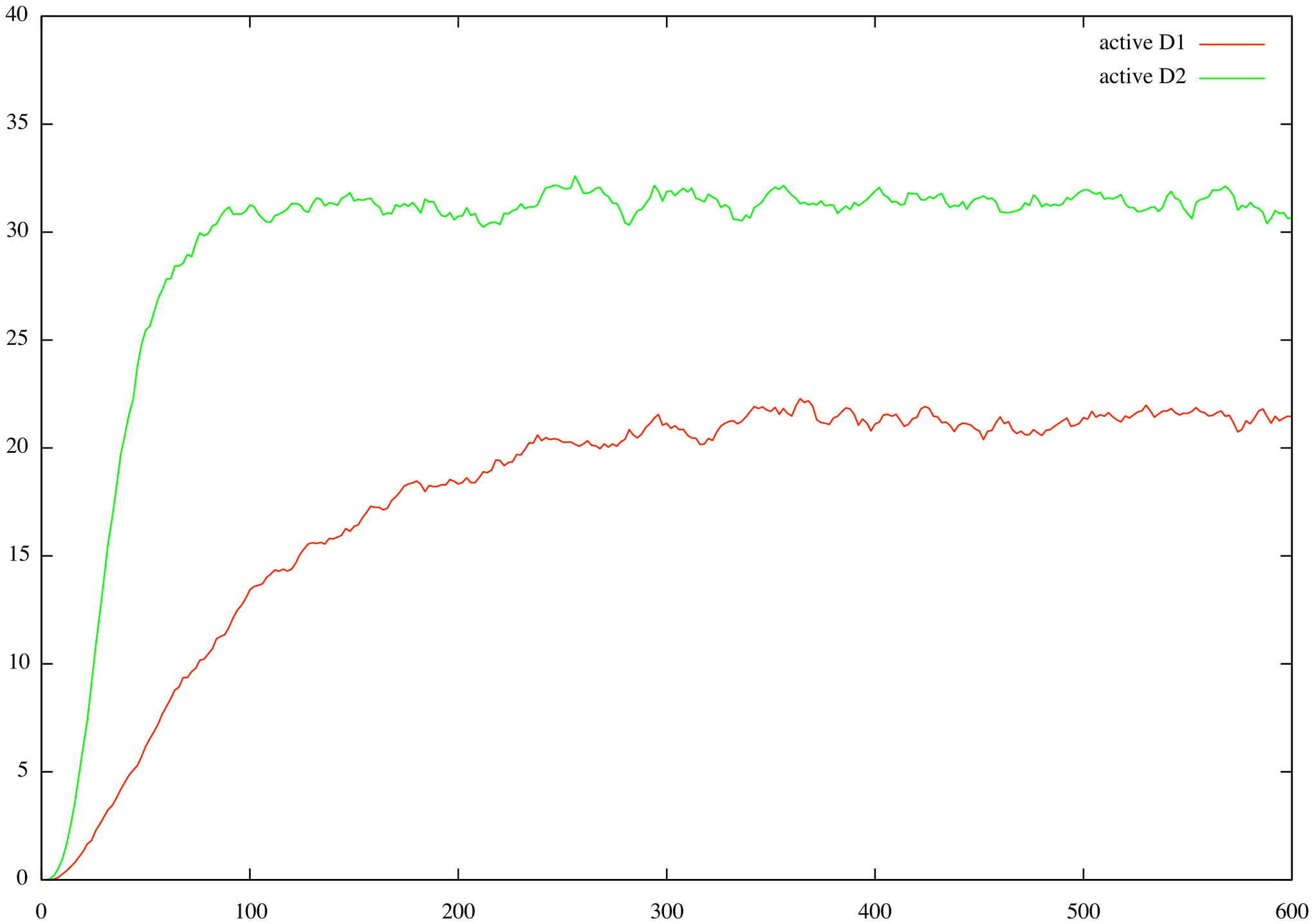
U2 only


active D1
active D2
40
35
30
25
20
15
10
5
0
0
100
200
300
400
500
600

What do you think?

- What happens if both U1 *and* U2 are present?

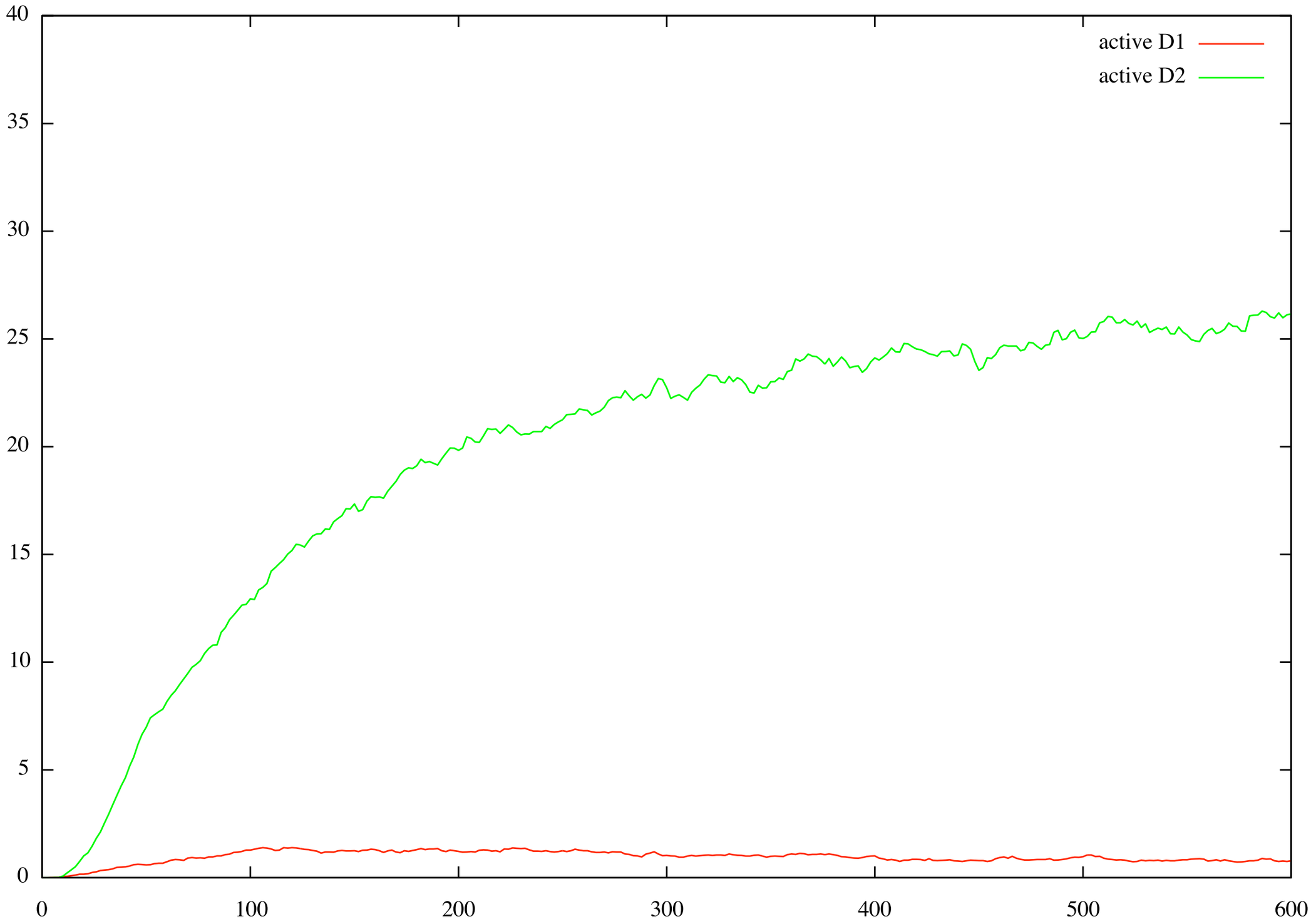
U1 + U2 + excess X



What do you think?

- What if we reduce the amount of X by 10-fold?
  - 100 instead of 1000 agents

U1 + U2 + limited X



What do you think?

- Why do U2/D2 ‘win’ over U1/D1?
  - one input trumps the other
  - like a transistor...

Conclusion?

- Cell signalling is complicated!
- Kappa provides novel tools to analyze their subtle and counter-intuitive dynamics...

END