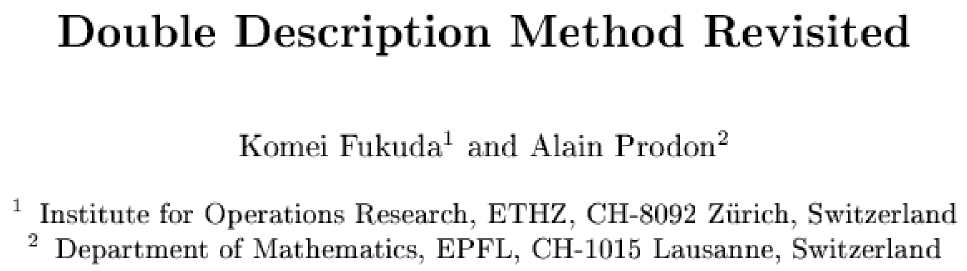
V17 The Double Description method: Theoretical framework behind EFM and EP / Integration Algorithms

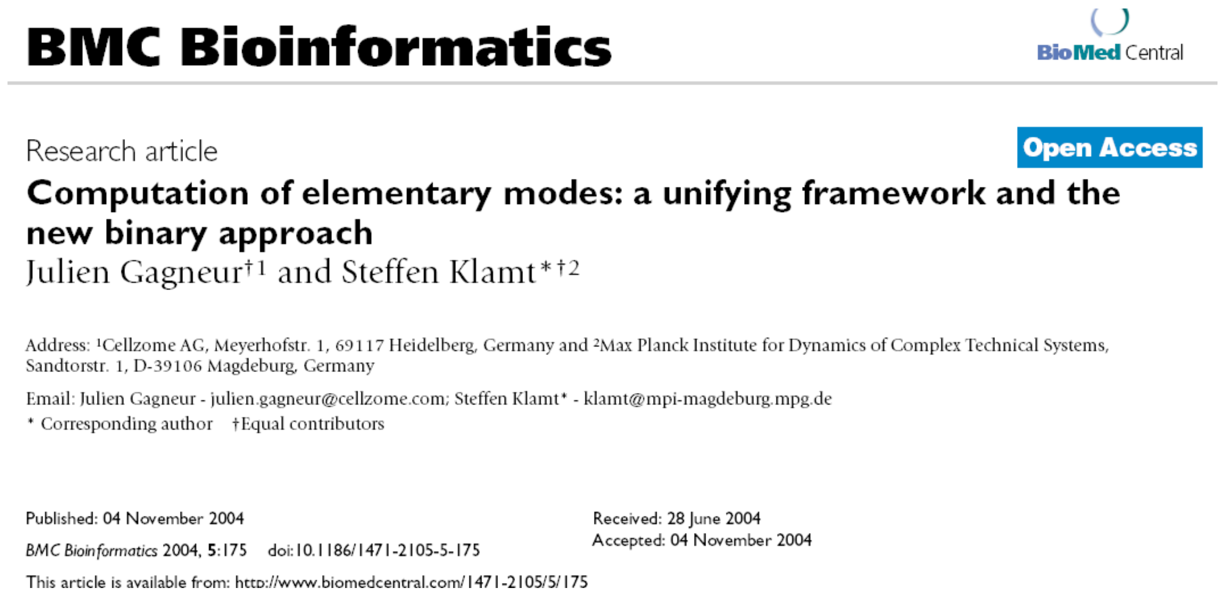


Double Description Method Revisited

Komei Fukuda[1] and Alain Prodon[2]

[1] Institute for Operations Research, ETHZ, CH-8092 Zürich, Switzerland
[2] Department of Mathematics, EPFL, CH-1015 Lausanne, Switzerland

in „Combinatorics and Computer Science Vol. 1120“ edited by Deza, Euler, Manoussakis, Springer, 1996:91



BMC Bioinformatics

BioMed Central

Research article

Open Access

Computation of elementary modes: a unifying framework and the new binary approach

Julien Gagneur†1 and Steffen Klamt*†2

Address: 1Cellzome AG, Meyerhofstr. 1, 69117 Heidelberg, Germany and 2Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstr. 1, D-39106 Magdeburg, Germany

Email: Julien Gagneur - julien.gagneur@cellzome.com; Steffen Klamt* - klamt@mpi-magdeburg.mpg.de

* Corresponding author †Equal contributors

Published: 04 November 2004

BMC Bioinformatics 2004, **5**:175 doi:10.1186/1471-2105-5-175

This article is available from: http://www.biomedcentral.com/1471-2105/5/175

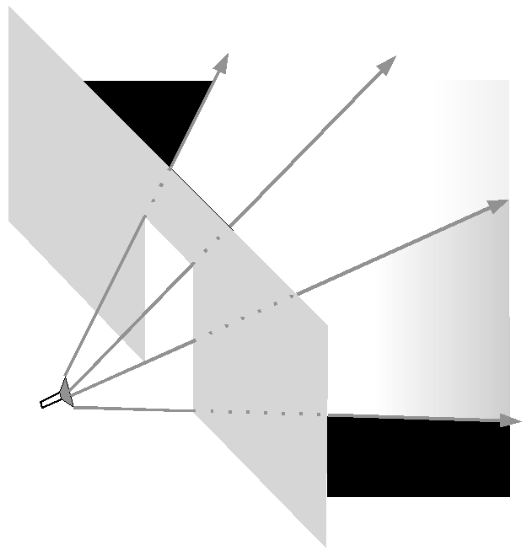
Received: 28 June 2004
Accepted: 04 November 2004

Double Description Method (1953)

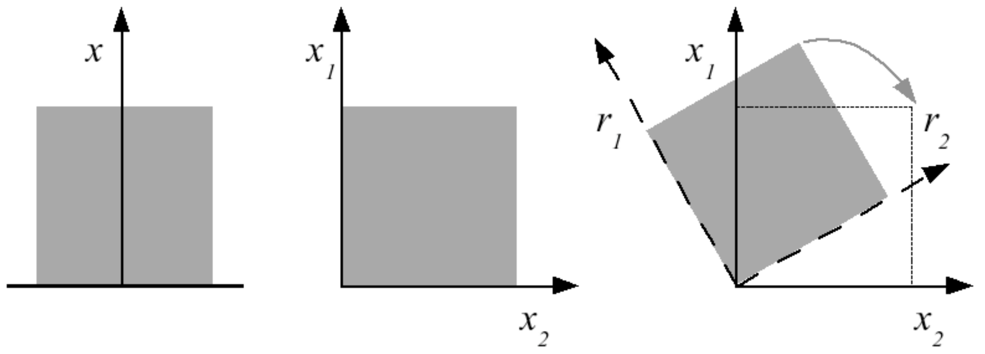
The Double Description method is the basis for simple & efficient algorithms for the task of enumerating extreme rays.

It serves as a framework for popular methods to compute elementary flux modes.

Analogy with Computer Graphics problem:
How can one efficiently describe the space in a dark room that is lighted by a torch shining through the open door?



Review: Duality of Matrices



This is the duality

Left: all points above the dividing line (the shaded area) fulfill the condition $x \geq 0$.
Middle: the points in the grey area fulfill the conditions $x_1 \geq 0$ and $x_2 \geq 0$.

But how could we describe the points in the grey area on the right side in a correspondingly simple manner?
Obviously, we could define a new coordinate system (r_1, r_2) as a new set of generating vectors.
But we could also try to transform this area back into the grey area of the middle panel and use the old axes x_1 and x_2.

In 2D, this transformation can be obviously best performed by multiplying all vectors inside the grey area by a two-dimensional **rotation matrix**.

The Double Description Method

A pair (**A**,**R**) of real matrices **A** and **R** is said to be a **double description pair** or simply a ***DD* pair** if the relationship

$$\boxed{\mathbf{A}\,\mathbf{x} \geq \mathbf{0} \quad \text{if and only if} \quad \mathbf{x} = \mathbf{R}\,\lambda \text{ for some } \lambda \geq \mathbf{0}}$$

holds. The column size of **A** has to equal the row size of **R**, say *d.*

For such a pair,

the set *P(***A***)* represented by **A** as

$$P(A) = \{x \in \Re^d : Ax \geq 0\}$$

is simultaneously represented by **R** as

$$\{x \in \Re^d : x = R\lambda \quad \text{for some } \lambda \geq 0\}$$

A subset *P* of $\Re^d$ is called **polyhedral cone** if *P = P(***A***)* for some matrix **A**, and **A** is called a **representation matrix** of the polyhedral cone *P(***A***).*

Then, we say **R** is a **generating matrix** for *P*. Clearly, each column vector of a generating matrix **R** lies in the cone *P* and every vector in *P* is a nonnegative combination of some columns of **R**.

The Double Description Method

Theorem 1 (Minkowski‘s Theorem for Polyhedral Cones)
For any $m \times n$ real matrix **A***,* there exists some $d \times m$ real matrix **R** such that (**A**,**R***)* is a *DD* pair, or in other words, the cone *P(***A***)* is generated by **R***.*

The theorem states that every polyhedral cone admits a generating matrix.

The nontriviality comes from the fact that the row size of **R** is finite.
If we allow an infinite size, there is a trivial generating matrix consisting of all vectors in the cone.

Also the converse is true:

Theorem 2 (Weyl‘s Theorem for Polyhedral Cones)
For any $d \times n$ real matrix **R***,* there exists some $m \times d$ real matrix **A** such that *(***A**,**R***)* is a *DD* pair, or in other words, the set generated by **R** is the cone *P(***A***).*

The Double Description Method

Task: how does one construct a matrix **R** from a given matrix **A***,* and the converse?

These two problems are computationally equivalent.
Farkas' Lemma shows that (**A,R**) is a *DD* pair if and only if *(*$\mathbf{R}^T$**,**$\mathbf{A}^T$*)* is a *DD* pair.

A more appropriate formulation of the problem is to require the minimality of **R***:* find a matrix **R** such that no proper submatrix is generating *P(***A***)*.
A minimal set of generators is unique up to positive scaling when we assume the regularity condition that the cone is **pointed**, i.e. the origin is an extreme point of *P(***A***).*

Geometrically, the columns of a minimal generating matrix are in 1-to-1 correspondence with the **extreme rays** of **P***.*

Thus the problem is also known as the **extreme ray enumeration problem**.

No efficient (polynomial) algorithm is known for the general problem.

Double Description Method: primitive form

Suppose that the $m \times d$ matrix **A** is given and let $P(A) = \{x : Ax \geq 0\}$

(This is equivalent to the situation at the beginning of constructing EPs or EFMs: we only know **S**.)

The *DD* method is an incremental algorithm to construct a $d \times m$ matrix **R** such that (**A**,**R**) is a *DD* pair.

Let us assume for simplicity that the cone *P*(**A**) is pointed.

Let **K** be a subset of the row indices {1,2,...,*m*} of **A** and let $\mathbf{A}_\mathbf{K}$ denote the submatrix of **A** consisting of rows indexed by **K**.
Suppose we already found a generating matrix **R** for $\mathbf{A}_\mathbf{K}$, or equivalently, ($\mathbf{A}_\mathbf{K}$,**R**) is a *DD* pair. If $\mathbf{A} = \mathbf{A}_\mathbf{K}$,we are done.

Otherwise we select any row index *i* not in **K** and try to construct a *DD* pair ($\mathbf{A}_{\mathbf{K+i}}$, **R**') using the information of the *DD* pair ($\mathbf{A}_\mathbf{K}$,**R**).

Once this basic procedure is described, we have an algorithm to construct a generating matrix **R** for *P*(**A**).

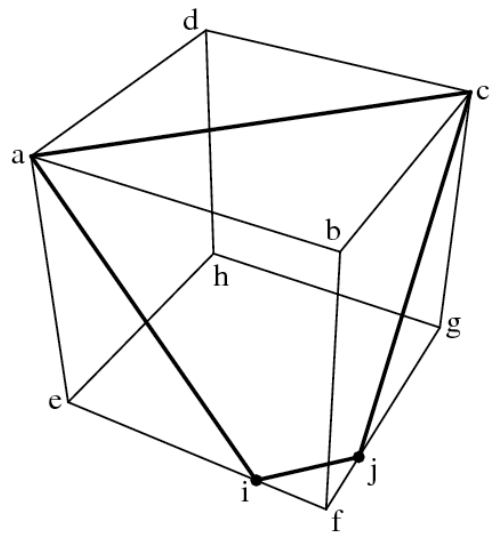
Geometric version of iteration step

The procedure can be understood geometrically by looking at the cut-section *C* of the cone *P*(**A**$_K$) with some appropriate hyperplane *h* in $\Re^d$ which intersects with every extreme ray of *P*(**A**$_K$) at a single point.

Let us assume that the cone is pointed and thus *C* is bounded.

Having a generating matrix **R** means that all extreme rays (i.e. extreme points of the cut-section) of the cone are represented by columns of **R**.

Such a cutsection is illustrated in the Fig.



Here, *C* is the cube *abcdefgh.*

Geometric version of iteration step

The newly introduced inequality $\mathbf{A_i} \cdot \mathbf{x} \geq 0$ partitions the space $\Re^d$ into three parts:

$$H_i^+ = \{\mathbf{x} \in \Re^d : \mathbf{A_i} \cdot \mathbf{x} > 0\}$$
$$H_i^0 = \{\mathbf{x} \in \Re^d : \mathbf{A_i} \cdot \mathbf{x} = 0\}$$
$$H_i^- = \{\mathbf{x} \in \Re^d : \boldsymbol{A_i} \cdot \mathbf{x} < 0\}$$

The intersection of H_i^0 with P and the new extreme points i and j in the cut-section C are shown in bold in the Fig.

Let J be the set of column indices of **R**. The rays $\mathbf{r_j}$ ($j \in J$) are then partitioned into three parts accordingly:

$$J^+ = \{j \in J : \mathbf{r_j} \in H_i^+\}$$
$$J^0 = \{j \in J : \mathbf{r_j} \in H_i^0\}$$
$$J^- = \{j \in J : \mathbf{r_j} \in H_i^-\}$$

We call the rays indexed by J^+, J^0, J^- the **positive, zero, negative** rays with respect to i, respectively.

To construct a matrix **R**‘ from **R**, we generate new $|J^+| \times |J^-|$ rays lying on the *ith* hyperplane H_i^0 by taking an appropriate positive combination of each positive ray $\mathbf{r_j}$ and each negative ray $\mathbf{r_{j'}}$ and by discarding all negative rays.

Geometric version of iteration step

The following lemma ensures that we have a *DD* pair ($\mathbf{A}_{K+i}$, $\mathbf{R}'$), and provides the key procedure for the most primitive version of the *DD* method.

Lemma 3 Let *(*$\mathbf{A}_K$,$\mathbf{R}$*)* be a *DD* pair and let *i* be a row index of **A** not in **K**.
Then the pair ($\mathbf{A}_{K+i}$, $\mathbf{R}'$) is a *DD* pair, where $\mathbf{R}'$ is the $d \times |J'|$ matrix with column vectors $\mathbf{r}_j$ $(j \in J')$ defined by

$$J' = J^+ \cup J^0 \cup (J^+ \times J^-), \text{ and}$$

$$\mathbf{r}_{jj'} = (\mathbf{A}_i \cdot \mathbf{r}_j) \cdot \mathbf{r}_{j'} - (\mathbf{A}_i \cdot \mathbf{r}_{j'}) \cdot \mathbf{r}_j \text{ for each } (j,j') \in J^+ \times J^-$$

<u>Proof</u> omitted.

Finding seed *DD* pair

It is quite simple to find a *DD* pair ($\mathbf{A_K}$,$\mathbf{R}$) when |$\mathbf{K}$| = 1, which can serve as the initial *DD* pair.

Another simple (and perhaps the most efficient) way to obtain an initial *DD* form of *P* is by selecting a maximal submatrix $\mathbf{A_K}$ of $\mathbf{A}$ consisting of linearly independent rows of $\mathbf{A}$.

The vectors $\mathbf{r_j}$'s are obtained by solving the system of equations

$$\mathbf{A_K\ R = I}$$

where $\mathbf{I}$ is the identity matrix of size |$\mathbf{K}$|, $\mathbf{R}$ is a matrix of unknown column vectors $\mathbf{r_j}$, $j \in J$.

As we have assumed rank($\mathbf{A}$) = *d,* i.e. $\mathbf{R} = \mathbf{A_K}^{-1}$, the pair ($\mathbf{A_K}$,$\mathbf{R}$) is clearly a *DD* pair, since $\mathbf{A_K} \cdot \mathbf{x} \geq \mathbf{0} \leftrightarrow \mathbf{x} = \mathbf{A_K}^{-1}\lambda,\ \lambda \geq \mathbf{0}$.

Primitive algorithm for DoubleDescriptionMethod

```
procedure DoubleDescriptionMethod($A$);
begin
    Obtain any initial DD pair $(A_K, R)$;
    while $K \neq \{1, 2, \ldots, m\}$ do
    begin
        Select any index $i$ from $\{1, 2, \ldots, m\} \setminus K$;
        Construct a DD pair $(A_{K+i}, R')$ from $(A_K, R)$;
            /* by using Lemma 3 */
        $R := R'$;    $K := K + i$;
    end;
    Output $R$;
end.
```

This algorithm is very primitive, and the straightforward implementation will be quite useless, because the size of *J* increases extremely fast.

This is because many vectors $\mathbf{r}_{jj'}$ generated by the algorithm (defined in Lemma 3) are unnessary.
We need to avoid generating redundant vectors!

To avoid generating redundant vectors, we will use the zero set or active set Z(**x**) which is the set of inequality indices satisfied by **x** in P(**A**) with equality.

Noting $\mathbf{A}_{i\bullet}$ the ith row of **A**, Z(**x**) = {i : $\mathbf{A}_{i\bullet}$ **x** = 0}

Towards the standard implementation

Two distinct extreme rays **r** and **r'** of *P* are **adjacent** if the minimal face of *P* containing both contains no other extreme rays.

Proposition 7. Let **r** and **r'** be distinct rays of *P*.
Then the following statements are equivalent
(a) **r** and **r'** are adjacent extreme rays,
(b) **r** and **r'** are extreme rays and the rank of the matrix $\mathbf{A}_{Z(\mathbf{r}) \cap Z(\mathbf{r}')}$ is $d - 2$,
(c) if **r''** is a ray with $Z(\mathbf{r}'') \supset Z(\mathbf{r}) \cap Z(\mathbf{r}')$ then either $\mathbf{r}'' \simeq \mathbf{r}$ or $\mathbf{r}'' \simeq \mathbf{r}'$.

Lemma 8. Let $(\mathbf{A}_K, \mathbf{R})$ be a *DD* pair such that rank($\mathbf{A}_K$) = d and let i be a row index of **A** not in K. Then the pair $(\mathbf{A}_{K+i}, \mathbf{R}')$ is a *DD* pair, where $\mathbf{R}'$ is the $d \times | J'|$ matrix with column vectors $\mathbf{r}_j$ $(j \in J')$ defined by

$J' = J^+ \cup J^0 \cup \text{Adj}$

$\text{Adj} = \{(j,j') \in J^+ \times J^- : \mathbf{r}_j \text{ and } \mathbf{r}_{j'} \text{ are adjacent in } P(\mathbf{A}_K)\}$ and

$\mathbf{r} = (\mathbf{A}_i \mathbf{r}_j)\, \mathbf{r}_{j'} - (\mathbf{A}_i \mathbf{r}_{j'})\, \mathbf{r}_j$ for each $(j,j') \in \text{Adj}$.

Furthermore, if **R** is a minimal generating matrix for $P(\mathbf{A}_K)$ then $\mathbf{R}'$ is a minimal generating matrix for $P(\mathbf{A}_{K+i})$.

Algorithm for standard form of double description method

This is now a straightforward variation of the *DD* method which produces a minimal generating set for P:

```
procedure : DDMethodStandard(A)
begin
    Obtain any initial DD pair $(A_K, R)$;  such that $R$ is minimal
    while $K \neq \{1, 2, \ldots, m\}$ do
    begin
        Select any index $i$ from $\{1, 2, \ldots, m\} \setminus K$;
        Construct a DD pair $(A_{K+i}, R')$ from $(A_K, R)$;
            /* by using Lemma 8 */
        $R := R'$;    $K := K + i$;
    end;
    Output $R$;
end.
```

To implement `DDMethodStandard`, we must check for each pair of extreme rays **r** and **r'** of $P(\mathbf{A}_K)$ with $\mathbf{A}_i\,\mathbf{r} > 0$ and $\mathbf{A}_i\,\mathbf{r}' < 0$ whether they are adjacent in $P(\mathbf{A}_K)$.

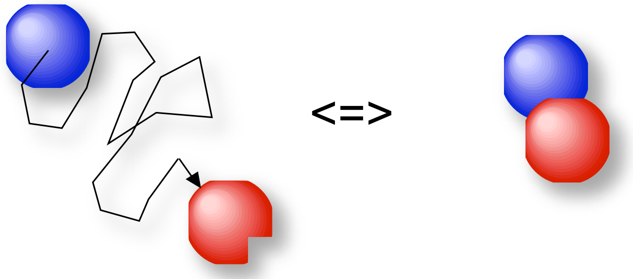
V17 – second part

Dynamic Modelling: Rate Equations + Stochastic Propagation

Mass Action Kinetics

Most simple dynamic system: inorganic chemistry

Consider reaction A + B <=> AB



Interesting quantities:
(changes of) densities of A, B, and AB

$$\textbf{density} = \frac{\text{number of particles}}{\text{unit volume}}$$

$$[A] = \frac{N_A}{V}, \quad \frac{d}{dt}[A](t)$$

1 mol = 1 Mol / Liter = $6.022 \times 10^{23} \times (0.1\ \text{m})^{-3} = 0.6\ \text{nm}^{-3}$

This means that proteins cannot reach 1 mol concentrations. Why?

Association: probability that A finds and reacts with B
=> changes proportional to densities of A *and* of B

Dissociation: probability for AB to break up
=> changes proportional to density of AB

} How to put this into formulas?

Mass Action II

Again: $A + B <=> AB$

Objective: mathematical description for the changes of [A], [B], and [AB]

Consider [A]:

Gain due to dissociation AB => A + B

Loss due to association A + B => AB

$$\frac{d}{dt}[A] = G_A - L_A$$

AB falls apart
=> G_A depends only on [AB]

A has find B
=> L_A depends on [A] *and* [B]

$$G_A = k_r\,[AB]$$

$$L_A = k_f\,[A]\,[B]$$

phenomenological proportionality constant

$$\boxed{\frac{d}{dt}[A] = k_r\,[AB] - k_f\,[A]\,[B]}$$

Mass Action !!!

A + B <=> AB

For [A]: we just found: $$\frac{d}{dt}[A] = k_r[AB] - k_f[A][B]$$

For [B]: for symmetry reasons $$\frac{d}{dt}[B] = \frac{d}{dt}[A]$$

For [AB]: exchange gain and loss $$\frac{d}{dt}[AB] = -\frac{d}{dt}[A] = k_f[A][B] - k_r[AB]$$

with $[A](t_0)$, $[B](t_0)$, and $[AB](t_0)$ => complete description of the system

time course = initial conditions + dynamics

A Second Example

Slightly more complex: $A + 2B <=> AB_2$

Association:
- one A and two B have to come together
- forming one complex AB_2 requires two units of B

$$L_A = k_f [A][B][B] = k_f [A][B]^2 \qquad L_B = 2k_f [A][B]^2$$

Dissociation: one AB_2 decays into one A and two B

$$G_A = k_r [AB_2] \qquad G_B = 2k_r [AB_2]$$

Put everything together

$$\frac{d}{dt}[A] = k_r [AB_2] - k_f [A][B]^2 \qquad \frac{d}{dt}[B] = 2\frac{d}{dt}[A] \qquad \frac{d}{dt}[AB_2] = -\frac{d}{dt}[A]$$

Some Rules of Thumb

$A + 2B \Leftrightarrow AB_2$ "A is produced when AB_2 falls apart or is consumed when AB_2 is built from one A and two B"

Sign matters: Gains with "+", losses with "–"

Logical conditions: "…from A *and* B"

"and" corresponds to " × " "or" corresponds to "+"

Stoichiometries: one factor for each educt (=> $[B]^2$)
prefactors survive

Mass conservation: terms with "–" have to show up with "+", too

$$\frac{d}{dt}[A] = k_r\,[AB_2] - k_f\,[A]\,[B]^2 \qquad \frac{d}{dt}[B] = 2\frac{d}{dt}[A] \qquad \frac{d}{dt}[AB_2] = -\frac{d}{dt}[A]$$

A Worked Example

Lotka-Volterra population model

R1:	A + X => 2X	prey X lives on A
R2:	X + Y => 2Y	predator Y lives on prey X
R3:	Y => B	predator Y dies

Rates for the reactions

$$\frac{dR_1}{dt} = k_1\,A\,X$$

$$\frac{dR_2}{dt} = k_2\,X\,Y$$

$$\frac{dR_3}{dt} = k_3\,Y$$

Changes of the metabolites

stoichiometric matrix S

	R1	R2	R3
A	–1		
X	1	–1	
Y		1	–1
B			1

=> change of X:

$$\frac{dX}{dt} = +k_1\,A\,X - k_2\,X\,Y + 0$$

Setting up the Equations

With $\vec{v} = \frac{d\vec{R}}{dt} = \begin{pmatrix} dR_1/dt \\ dR_2/dt \\ dR_3/dt \end{pmatrix}$ and $S = \begin{pmatrix} -1 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix}$

we get: $\frac{d}{dt}\vec{X} = \frac{d}{dt}\begin{pmatrix} A \\ X \\ Y \\ B \end{pmatrix} = S\,\frac{d}{dt}\vec{R}$ or $\frac{dX_i}{dt} = \sum_j S_{ij}\,\frac{dR_j}{dt}$

amounts processed per reaction

speeds of the reactions

Plug in to get:

$$\frac{dA}{dt} = -\frac{dR_1}{dt} = -k_1\,A\,X \qquad \frac{dX}{dt} = +\frac{dR_1}{dt} - \frac{dR_2}{dt} = k_1 AX - k_2 XY$$

$$\frac{dB}{dt} = +\frac{dR_3}{dt} = k_3\,Y \qquad \frac{dY}{dt} = +\frac{dR_2}{dt} - \frac{dR_3}{dt} = k_2 XY - k_3 Y$$

How Does It Look Like?

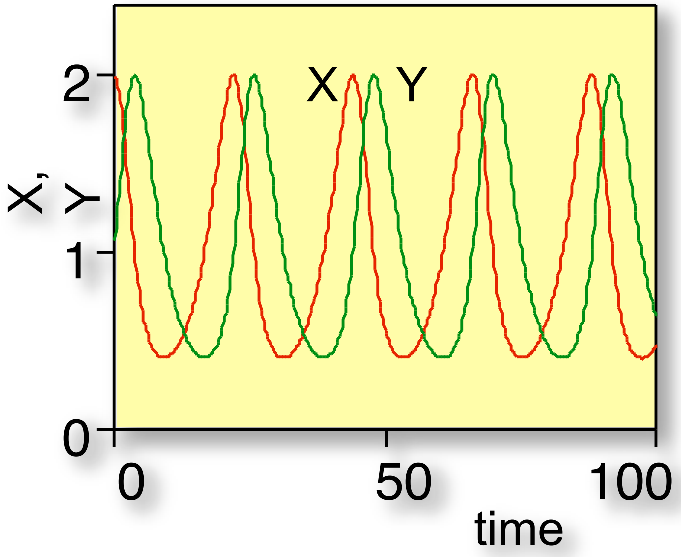
Lotka–Volterra: assume A = const, B ignored

=> cyclic population changes

$$\frac{dX}{dt} = k_1 A X - k_2 X Y$$

$$\frac{dY}{dt} = k_2 X Y - k_3 Y$$

$k_1 = k_2 = k_3 = 0.3$



Steady State: when do the populations not change?

$$\frac{dX}{dt} = \frac{dY}{dt} = 0 \quad => \quad Y = \frac{k_1}{k_2} A \quad X = \frac{k_3}{k_2}$$

Steady state = fluxes balanced

With $k_1 = k_2 = k_3 = 0.3$ and $A = 1$ => $X = Y = 1$

From rates to differences

Reaction: $A + B \Longrightarrow AB$

Rate equation: $\frac{dA}{dt} = -k \cdot A \cdot B = f(A(t), B(t))$

derivative of A(t) = some function

Taylor expansion for displacement *t* around $t_0 = 0$:

$$A(t) = A(0) + t \cdot \frac{dA}{dt}(0) + \frac{t^2}{2} \cdot \frac{d^2A}{dt^2}(0) + \ldots = \sum_{k=0} \frac{t^k}{k!} \cdot \frac{d^kA}{dt^k}(0)$$

Truncate this expansion after second term (linear approximation):

$$\begin{aligned} A(t) &\approx A(0) + \quad t \cdot \frac{dA}{dt}(0) \quad + \quad O(t^2) \\ &\approx A(0) + t \cdot f(A(0), B(0)) + O(t^2) \end{aligned}$$

From rates to differences II

Linear approximation to (true) A(t):

$$
\begin{aligned}
A(t) &\approx A(0) + \quad t \cdot \frac{dA}{dt}(0) \quad + \quad O(t^2) \\
&\approx \underbrace{A(0)}_{\text{initial condition}} + t \cdot \underbrace{f\left(A(0), B(0)\right)}_{\text{increment}} + \underbrace{O(t^2)}_{\text{error}}
\end{aligned}
$$

For $t \to 0$

$$
t \cdot \frac{dA}{dt}(0) \quad \gg \quad \frac{t^2}{2} \cdot \frac{d^2A}{dt^2}(0) \quad \gg \quad \ldots
$$

Use linear approximation for small time step Δt:

$$
A(t+\Delta t) = A(t) + \Delta t \cdot \frac{dA}{dt}(t)
$$

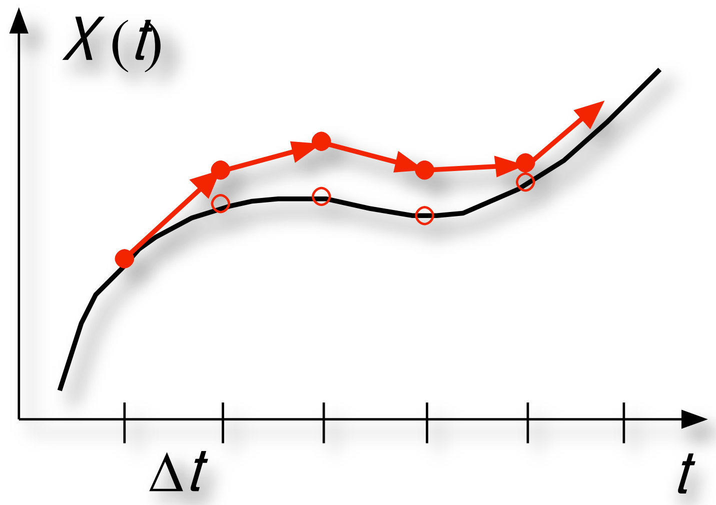
This is the so-called **"forward Euler" algorithm**

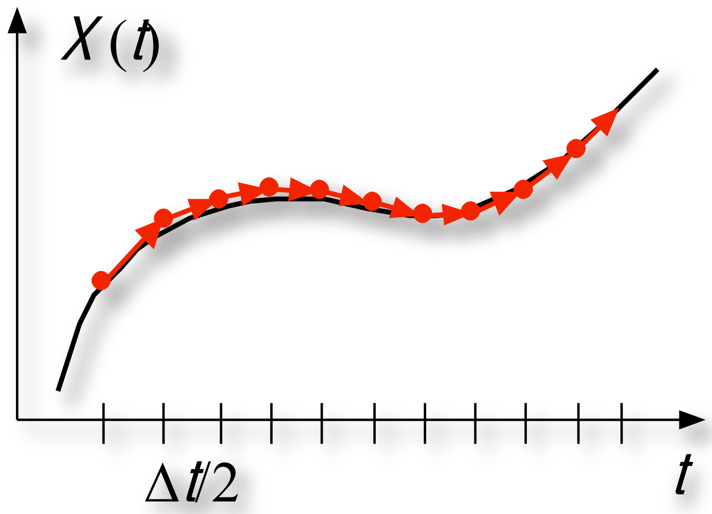
"Forward Euler" algorithm

General form: $$\vec{X}_i(t+\Delta t) = \vec{X}_i(t) + \Delta t \cdot \vec{f}(\vec{X}_j(t)) + O(\Delta t^2)$$

relative error: $$\varepsilon = \frac{\Delta t^2/2 \cdot X''}{\Delta t\, X'} \propto \Delta t$$ 1st order algorithm

relative error decreases with 1st power of step size Δt



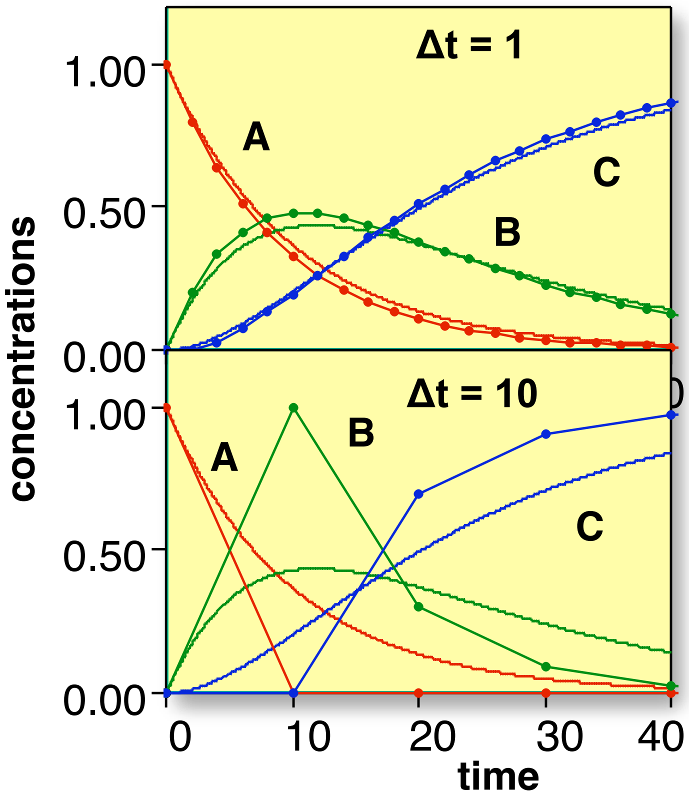


Black: ideal dynamic trajectory, **red:** dynamics integrated by forward Euler algorithm
Right side: integration time steps are half of left side -> smaller error

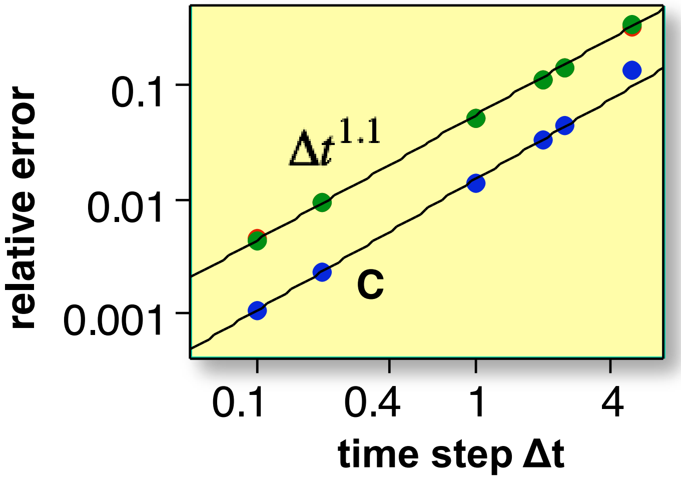
Example: chained reactions

Reaction: $A \Longrightarrow B \Longrightarrow C \qquad k_{AB} = 0.1, \quad k_{BC} = 0.07$

Time evolution:

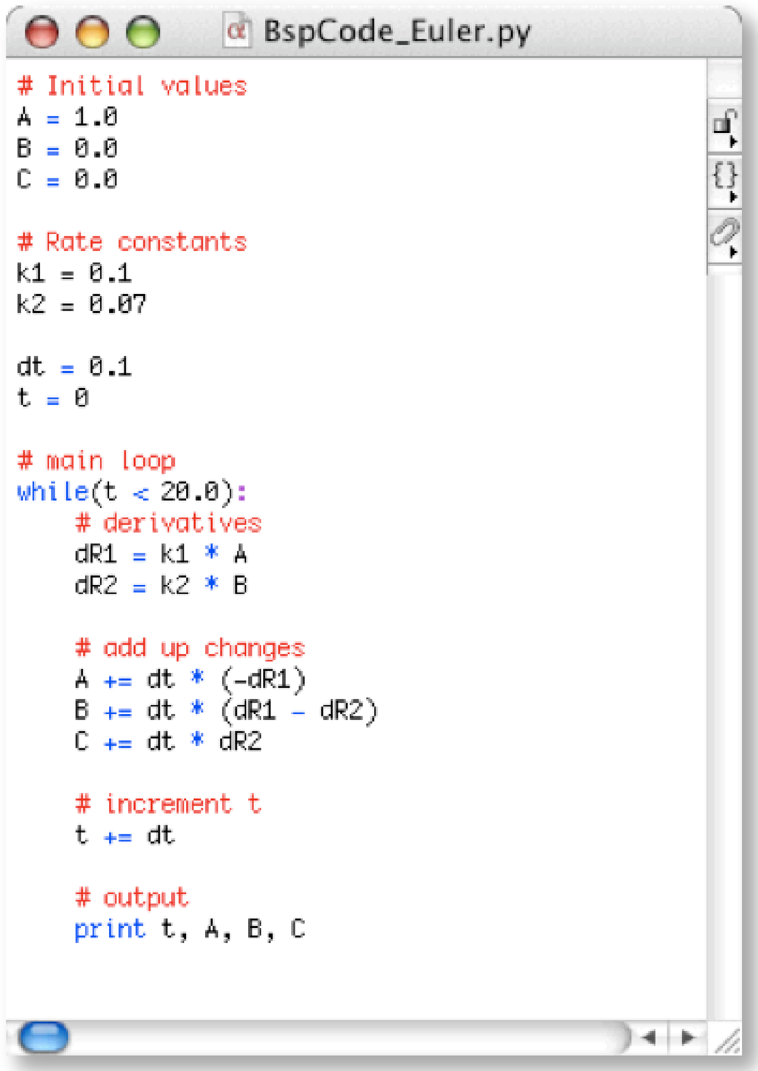


Relative error vs. Δt
at t = 10:



runtime α $(\Delta t)^{-1}$

Example Code: Forward Euler


BspCode_Euler.py

```
# Initial values
A = 1.0
B = 0.0
C = 0.0

# Rate constants
k1 = 0.1
k2 = 0.07

dt = 0.1
t = 0

# main loop
while(t < 20.0):
    # derivatives
    dR1 = k1 * A
    dR2 = k2 * B

    # add up changes
    A += dt * (-dR1)
    B += dt * (dR1 - dR2)
    C += dt * dR2

    # increment t
    t += dt

    # output
    print t, A, B, C
```

A => B => C

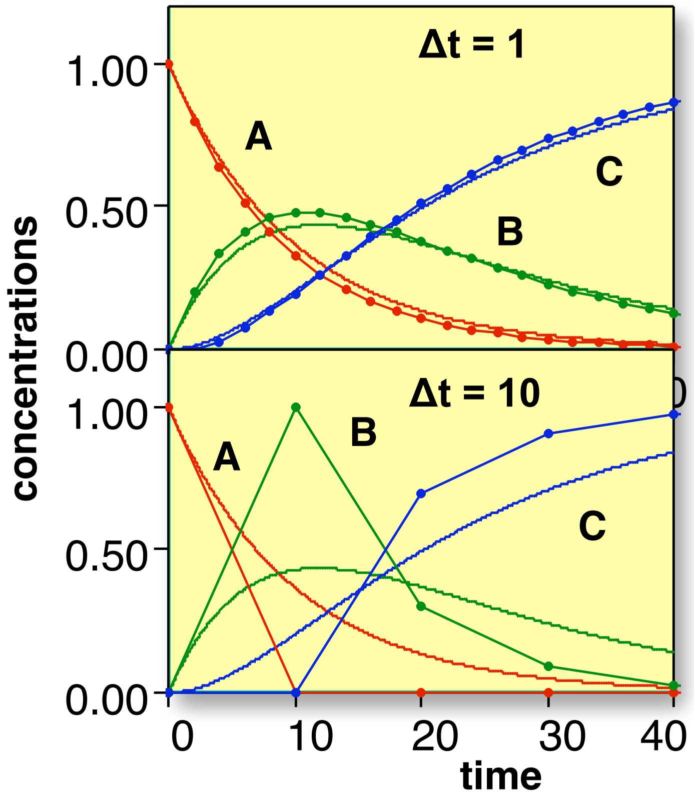
Iterate:

$$A(t + \Delta t) = A(t) + \Delta t \cdot \frac{dA}{dt}(t)$$

Important:

first calculate all derivatives,
then update densities!

What is the “correct” time step?



$$A \Longrightarrow B \Longrightarrow C$$

Approximation works for:

$$|\Delta A| = \left|\Delta t \frac{dA}{dt}\right| = \left|-k_{AB} \cdot A \cdot \Delta t\right| \ll A$$

$$\Rightarrow \quad \Delta t \ll \frac{1}{\max(k)}$$

Here: $k_{AB} = 0.1, \quad k_{BC} = 0.07$

$$\Rightarrow \quad \Delta t \ll 0.1^{-1} = 10$$

<u>Note 1:</u>
read “«” as “a few percent”

From Test Tubes to Cells

Rate equations <=> description via densities

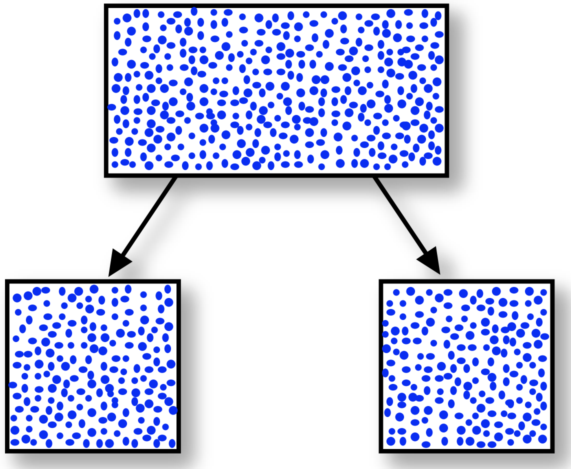
$$\text{density} = \frac{\text{indistinguishable particles}}{\text{volume element}}$$

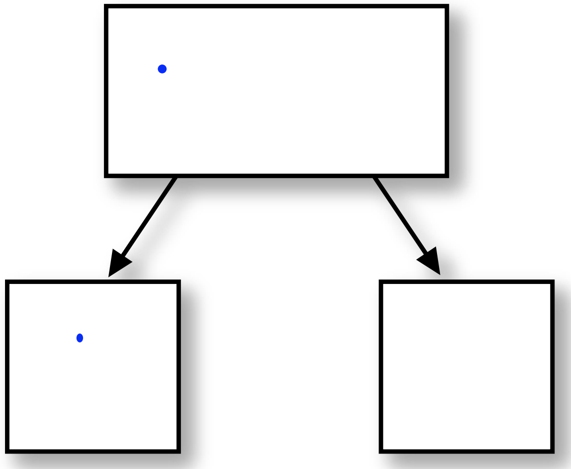
=> density is a continuum measure,
independent of the volume element

"half of the volume => half of the particles"

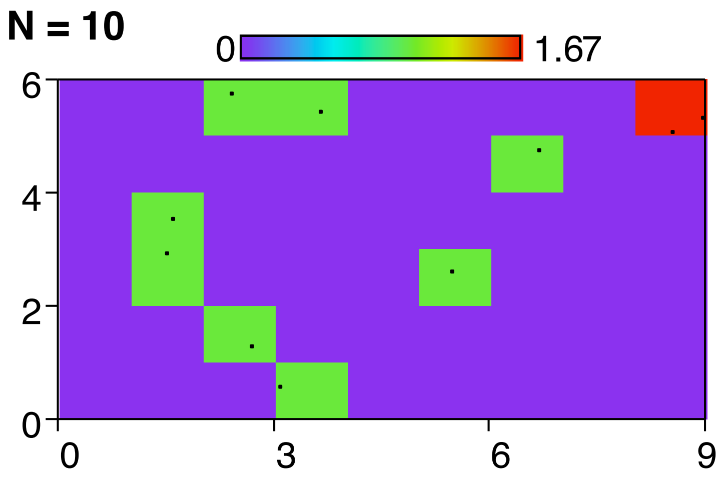
When density gets very low
=> each particle matters

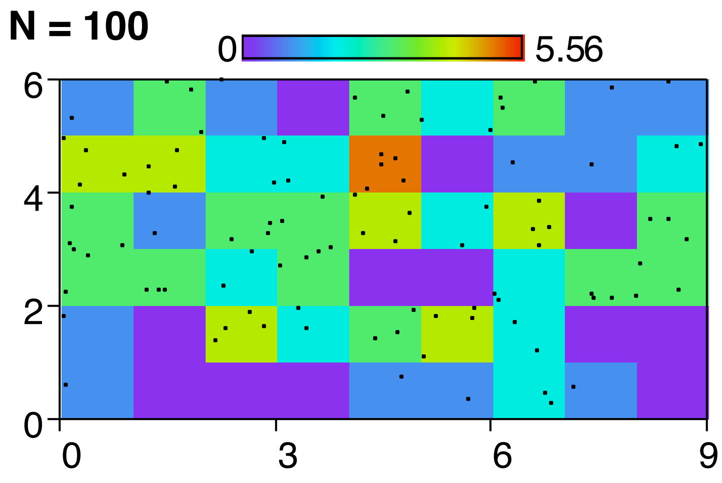
Examples:
~10 Lac repressors per cell, chemotaxis,
transcription from a single gene, …

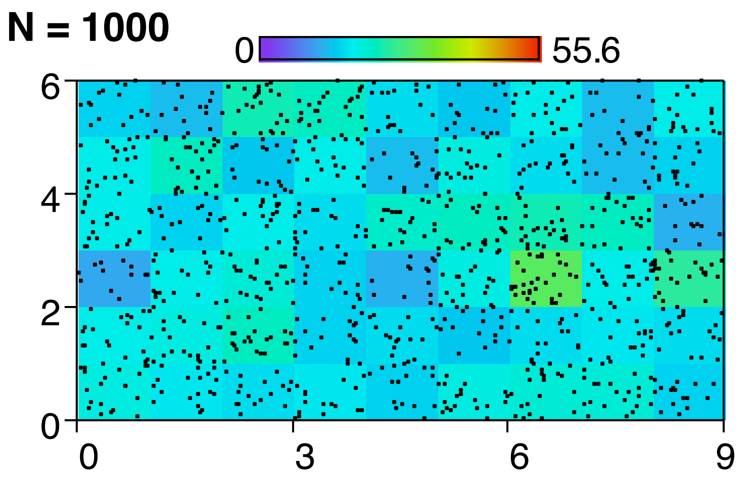


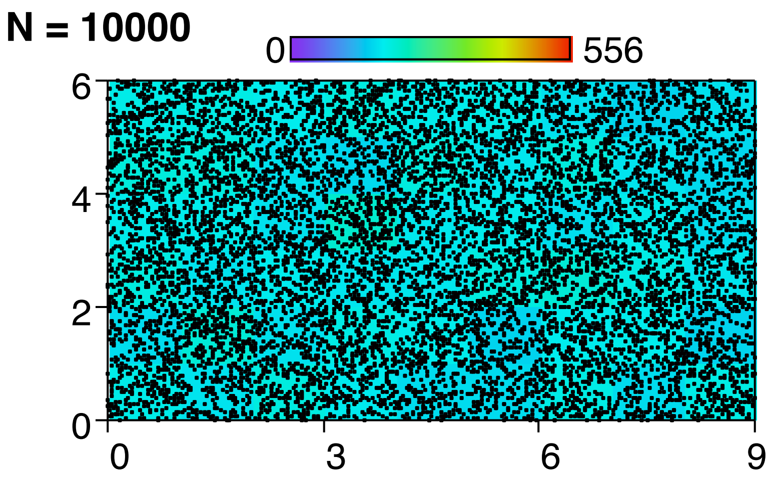


Density Fluctuations









Spread: Poisson Distribution

Stochastic probability that *k* events occur follows the Poisson distribution (here: event = "a particle is present"):

$$p_k = \frac{\lambda^k}{k!} e^{-\lambda}$$

k = 0, 1, 2, …

λ > 0 is a parameter

Average: $\langle k \rangle = \sum k \, p_k = \lambda$

Variance: $\sigma^2 = \sum p_k \, (k - \langle k \rangle)^2 = \lambda$

$$\sigma = \sqrt{\lambda}$$

Relative spread (error): $\frac{\Delta k}{k} = \frac{\sigma}{\langle k \rangle} = \frac{1}{\sqrt{\lambda}}$

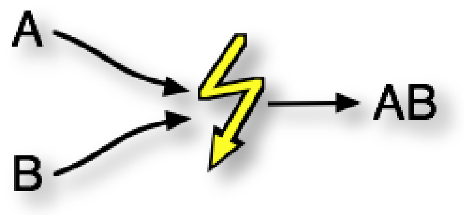
Avg. number of particles per unit volume	100	1000	1 Mol
relative uncertainty	10%	3%	1e-12

=> Fluctuations are negligible for "chemical" test tube situations

Reactions in the Particle View

Consider association:

A + B => AB



Continuous rate equation: $\frac{d[AB]}{dt} = k\,[A]\,[B]$

Number of new AB in volume V during Δt:

$$\begin{aligned}
\Delta N_{AB} &= \frac{d[AB]}{dt}\, V\, \Delta t \\
&= k_{AB}\, \frac{N_A}{V}\, \frac{N_B}{V}\, V\, \Delta t \\
&= \frac{k_{AB}\, \Delta t}{V}\, N_A\, N_B \\
&= P_{AB}\, N_A\, N_B
\end{aligned}$$

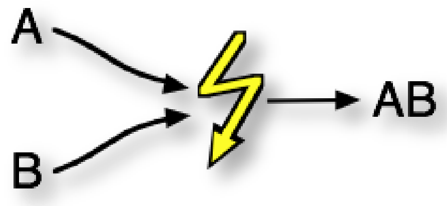
Density “picture” Particle “picture”

reaction rate k_{AB} => **reaction probability** P_{AB}

Units!

Consider:

A + B => AB



Change in the number of AB:

$$\Delta N_{AB} = P_{AB}\, N_A\, N_B$$

Association probability:

$$P_{AB} = \frac{k_{AB}\,\Delta t}{V}$$

Units: Continuous case

$$\frac{dAB}{dt} = k_{AB}\, A\, B$$

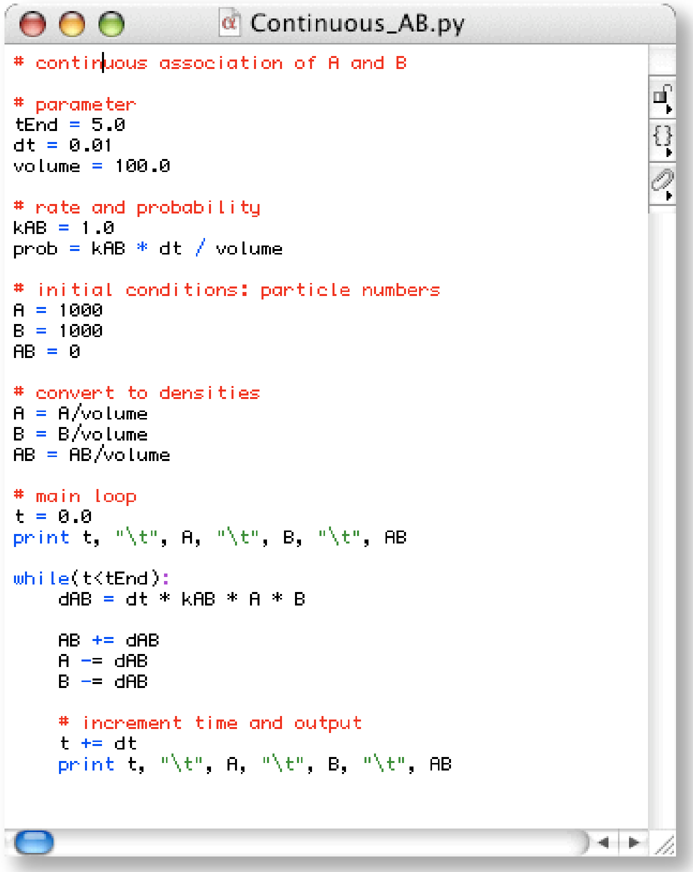
$$\left[\frac{dAB}{dt}\right] = \frac{\text{Mol}}{l\, s} \qquad [A] = [B] = \frac{\text{Mol}}{l} \qquad \texttt{<=>} \qquad [k_{AB}] = \frac{l}{\text{Mol}\, s}$$

Stochastic case

$$[N_{AB}] = [N_A] = [N_B] = 1 \qquad \texttt{<=>} \qquad [P_{AB}] = 1$$

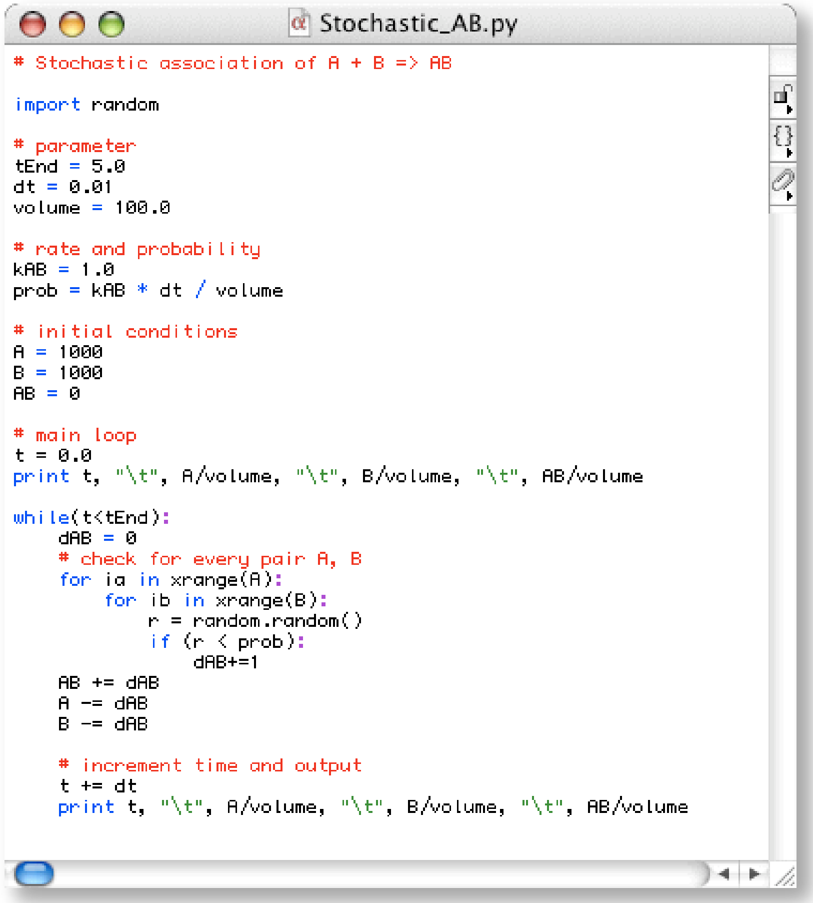
Direct Implementation

A + B => AB



Continuous_AB.py

```
# continuous association of A and B

# parameter
tEnd = 5.0
dt = 0.01
volume = 100.0

# rate and probability
kAB = 1.0
prob = kAB * dt / volume

# initial conditions: particle numbers
A = 1000
B = 1000
AB = 0

# convert to densities
A = A/volume
B = B/volume
AB = AB/volume

# main loop
t = 0.0
print t, "\t", A, "\t", B, "\t", AB

while(t<tEnd):
    dAB = dt * kAB * A * B

    AB += dAB
    A -= dAB
    B -= dAB

    # increment time and output
    t += dt
    print t, "\t", A, "\t", B, "\t", AB
```



Stochastic_AB.py

```
# Stochastic association of A + B => AB

import random

# parameter
tEnd = 5.0
dt = 0.01
volume = 100.0

# rate and probability
kAB = 1.0
prob = kAB * dt / volume

# initial conditions
A = 1000
B = 1000
AB = 0

# main loop
t = 0.0
print t, "\t", A/volume, "\t", B/volume, "\t", AB/volume

while(t<tEnd):
    dAB = 0
    # check for every pair A, B
    for ia in xrange(A):
        for ib in xrange(B):
            r = random.random()
            if (r < prob):
                dAB+=1
    AB += dAB
    A -= dAB
    B -= dAB

    # increment time and output
    t += dt
    print t, "\t", A/volume, "\t", B/volume, "\t", AB/volume
```

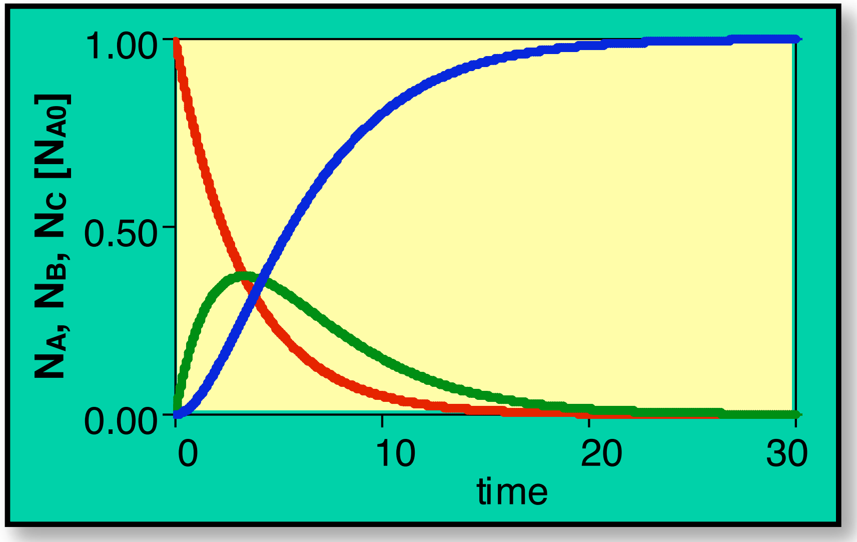
Note: both versions are didactic implementations

Example: Chained Reactions

A => B => C

Rates: $$\frac{dA}{dt} = -k_1 A \qquad \frac{dB}{dt} = k_1 A - k_2 B \qquad \frac{dC}{dt} = k_2 B$$

Time course from continuous rate equations (benchmark):

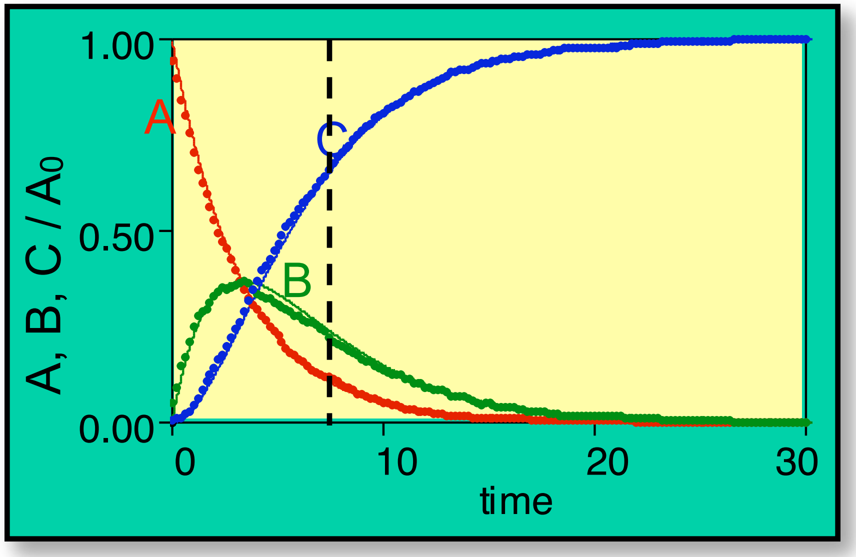


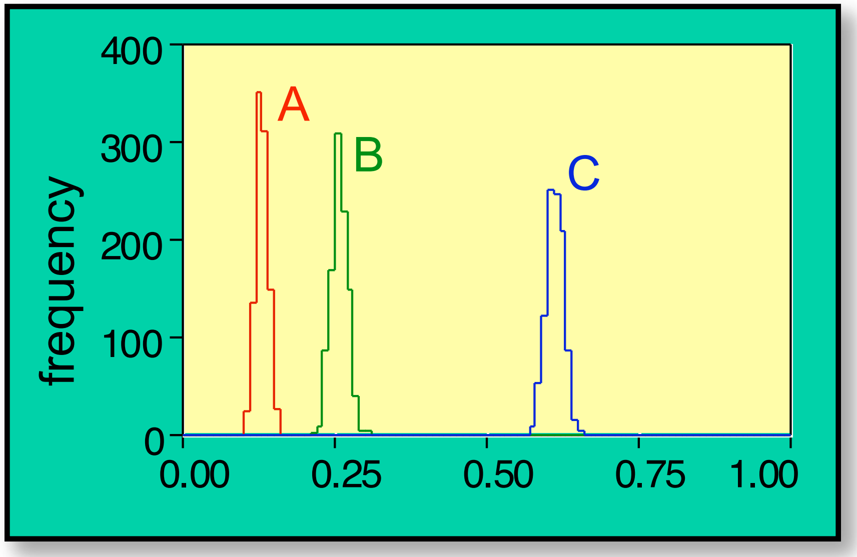
$k_1 = k_2 = 0.3$ (units?)

Stochastic Implementation

A => B => C　　　　A_0 = 1000 particles initially

t = 7



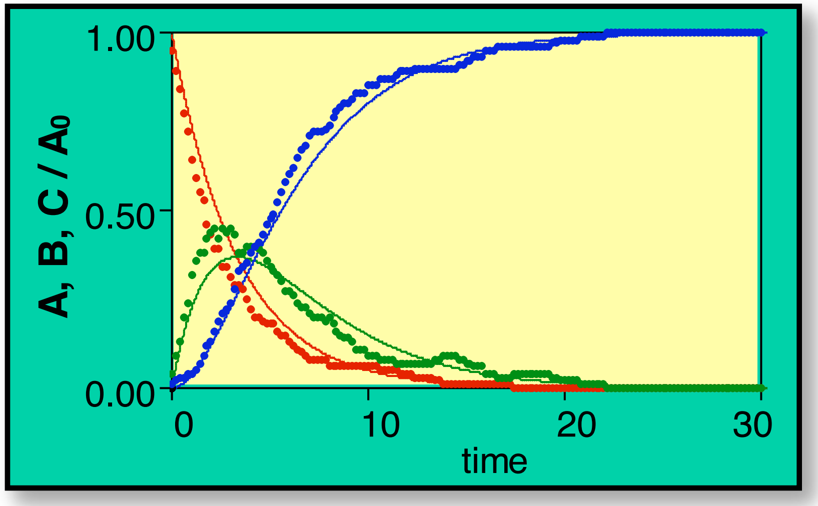


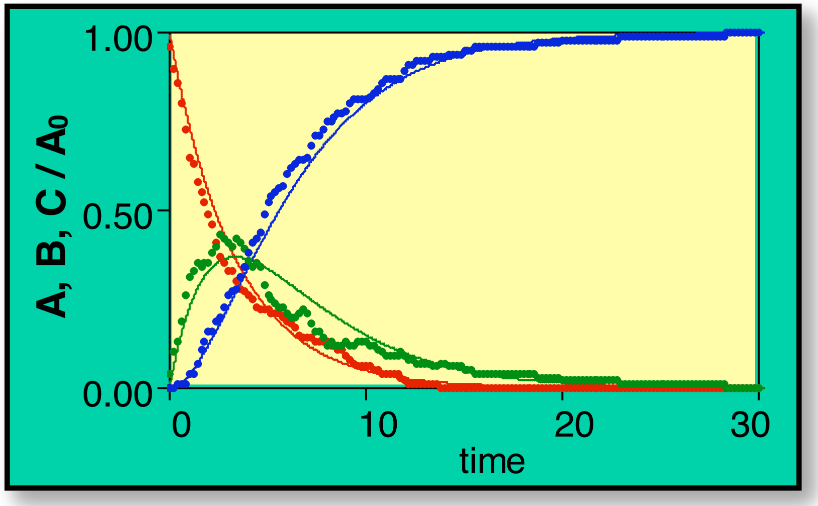
$k_1 = k_2 = 0.3$　　　　Values at t = 7 (1000 runs)

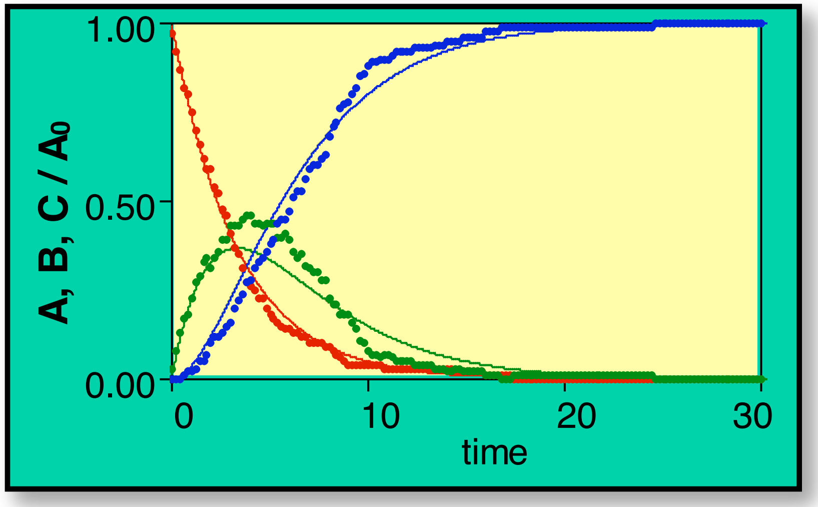
=> Stochastic version exhibits fluctuations

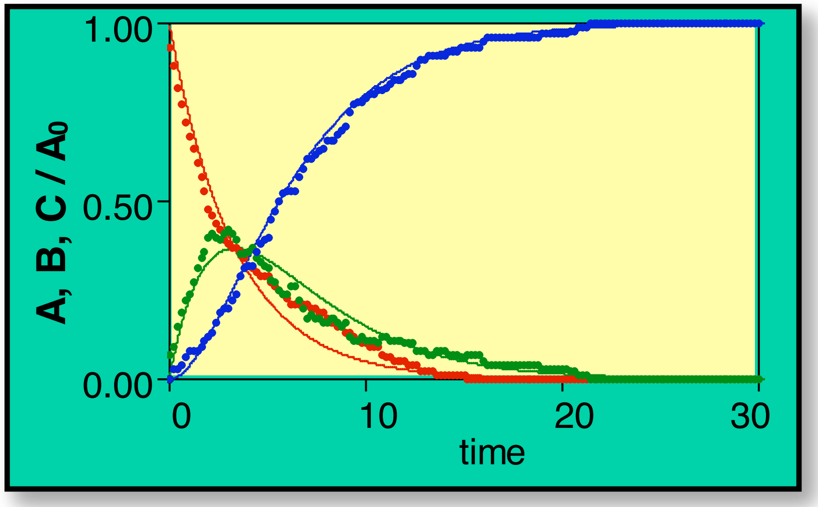
Less Particles => Larger Fluctuations

A_0 = 100 shown are 4 different runs



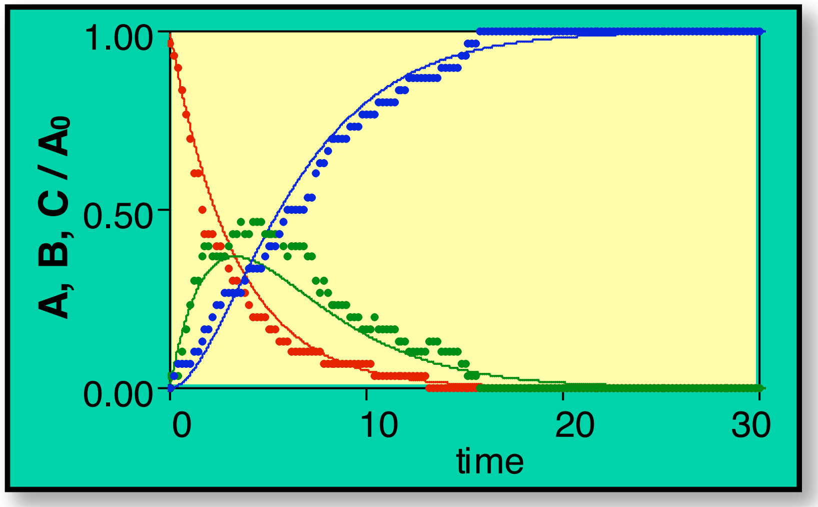


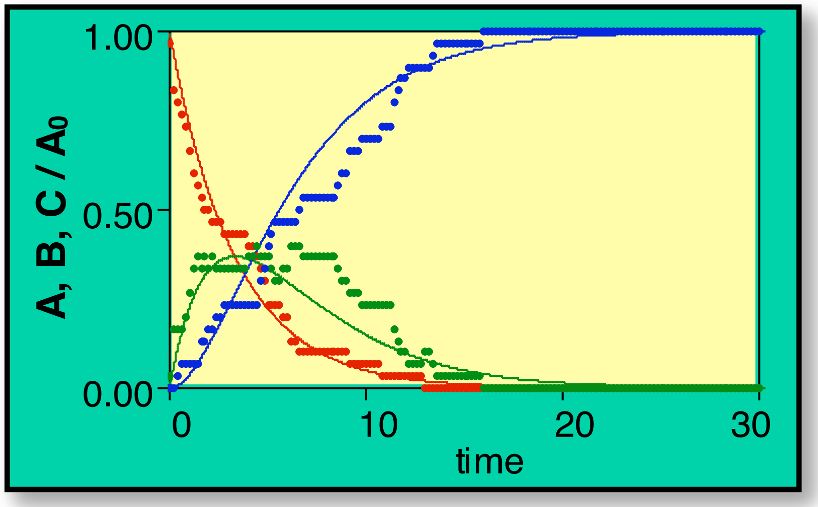


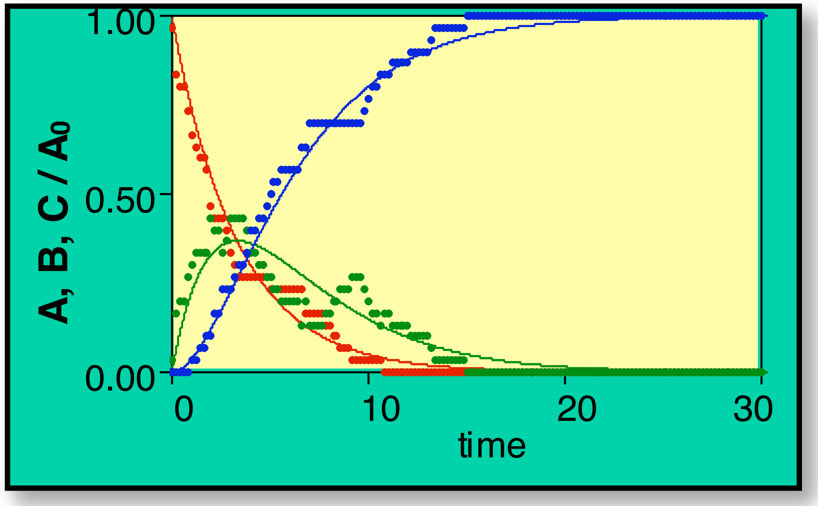


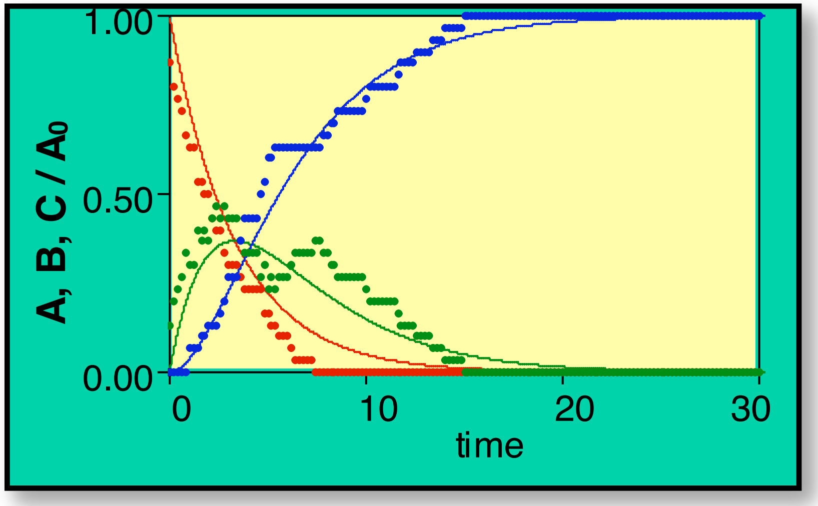
Even Less Particles

$A_0 = 30$









Spread vs. Particle Number

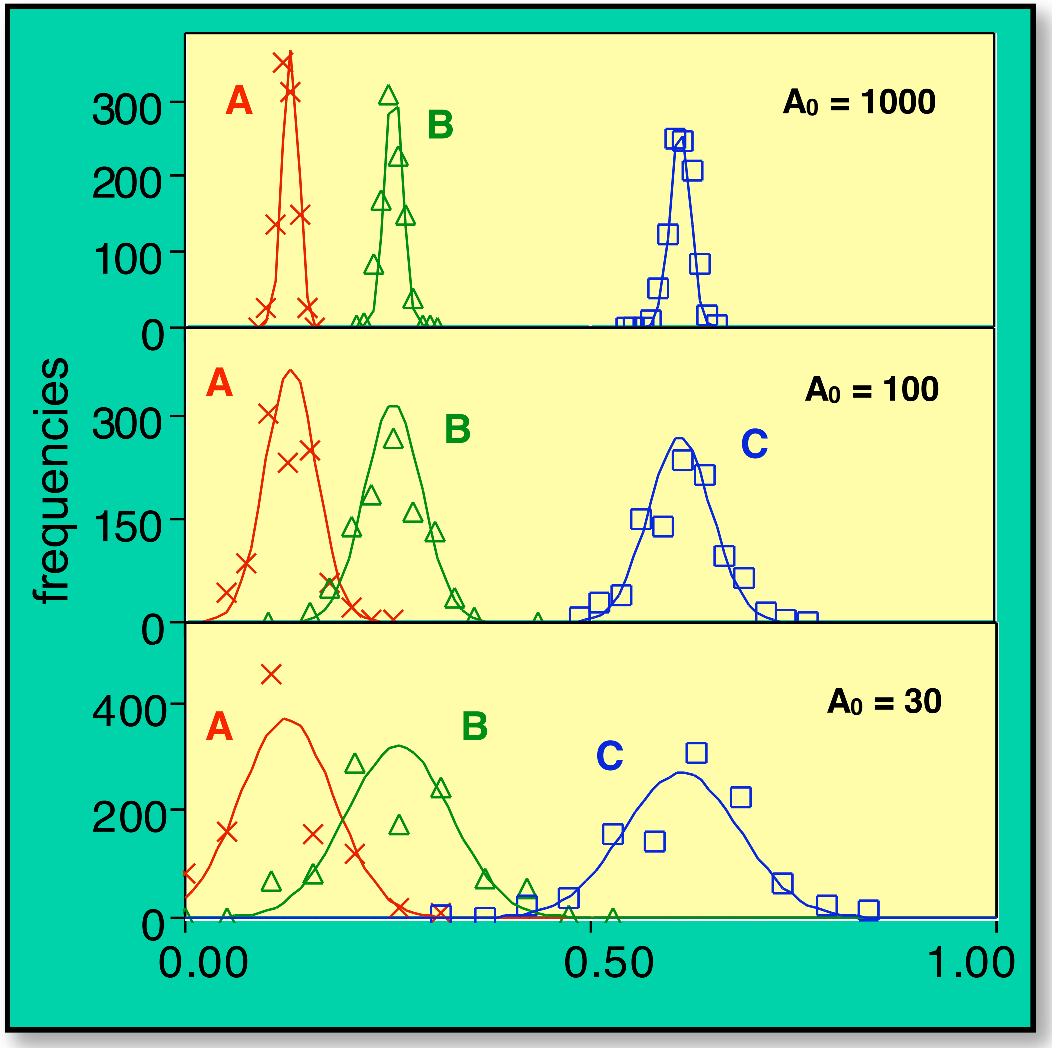
Poisson:
relative fluctuations $\propto 1/\sqrt{N}$

Repeat calculation 1000 times and record values at $t = 7$.

Fit distributions with Gaussian (Normal distribution)

$$g(x) = \exp\left[-\frac{(x - \langle x \rangle)^2}{w/\sqrt{A_0}}\right]$$

$\langle A \rangle = 0.13$,	$w_A = 0.45$
$\langle B \rangle = 0.26$,	$w_B = 0.55$
$\langle C \rangle = 0.61$,	$w_C = 0.45$



Stochastic Propagation

Naive implementation:

```
For every timestep:
        events = 0
        For every possible pair of A, B:
            get random number $r \in [0, 1)$
            if $r \leq P_{AB}$:
                events++
        AB += events
        A, B –= events
```

Features of this implementation

+ very simple
+ direct implementation of the underlying process

– costly runtime $O(N^2)$

– first order approximation

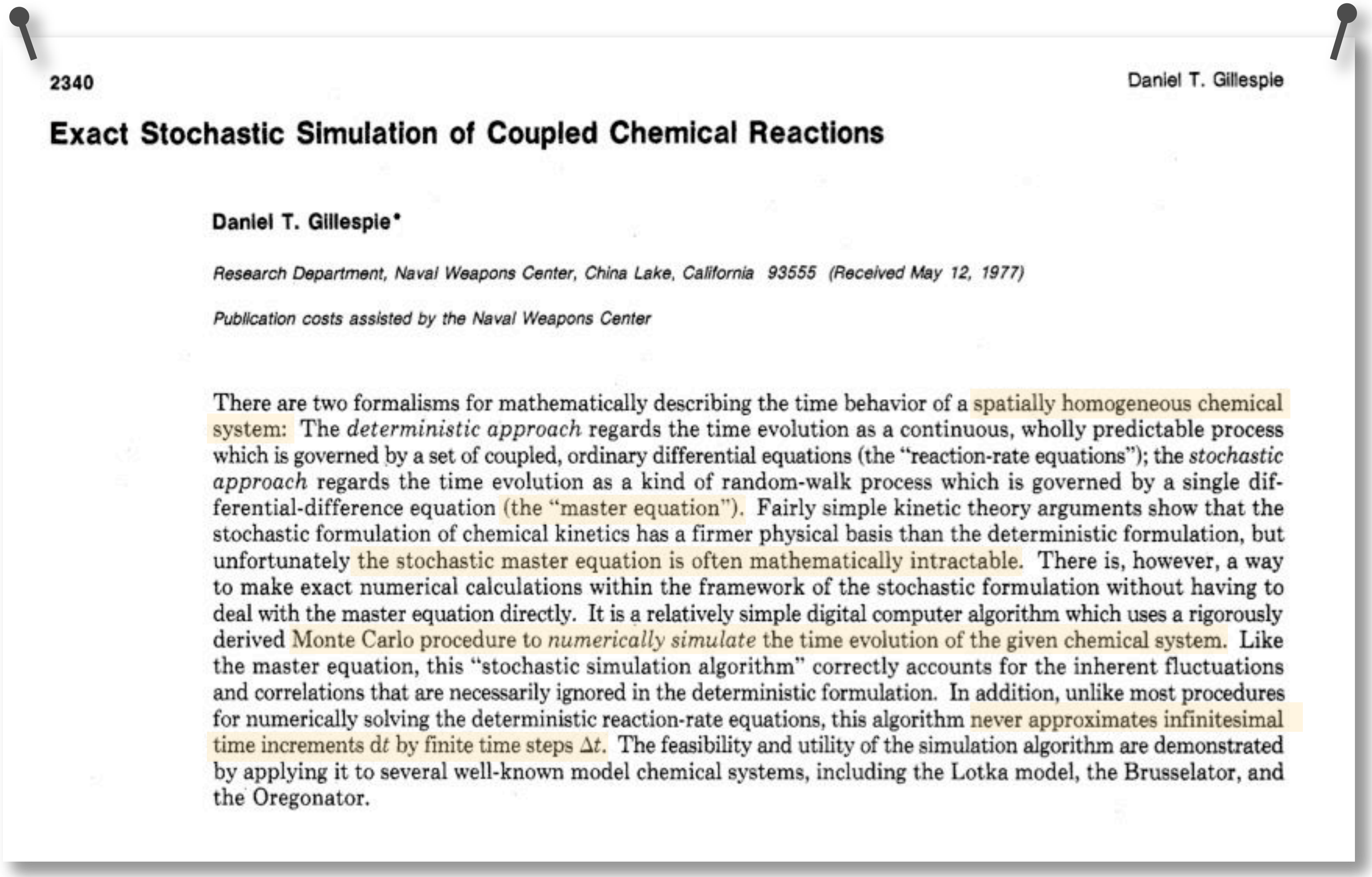
– one trajectory at a time

=> how to do better???

Determine complete probability distribution
=> **Master equation**

More efficient propagation
=> **Gillespie algorithm**

A Fast Algorithm



2340 Daniel T. Gillespie

Exact Stochastic Simulation of Coupled Chemical Reactions

Daniel T. Gillespie*

Research Department, Naval Weapons Center, China Lake, California 93555 (Received May 12, 1977)

Publication costs assisted by the Naval Weapons Center

There are two formalisms for mathematically describing the time behavior of a spatially homogeneous chemical system: The *deterministic approach* regards the time evolution as a continuous, wholly predictable process which is governed by a set of coupled, ordinary differential equations (the "reaction-rate equations"); the *stochastic approach* regards the time evolution as a kind of random-walk process which is governed by a single differential-difference equation (the "master equation"). Fairly simple kinetic theory arguments show that the stochastic formulation of chemical kinetics has a firmer physical basis than the deterministic formulation, but unfortunately the stochastic master equation is often mathematically intractable. There is, however, a way to make exact numerical calculations within the framework of the stochastic formulation without having to deal with the master equation directly. It is a relatively simple digital computer algorithm which uses a rigorously derived Monte Carlo procedure to *numerically simulate* the time evolution of the given chemical system. Like the master equation, this "stochastic simulation algorithm" correctly accounts for the inherent fluctuations and correlations that are necessarily ignored in the deterministic formulation. In addition, unlike most procedures for numerically solving the deterministic reaction-rate equations, this algorithm never approximates infinitesimal time increments dt by finite time steps Δt. The feasibility and utility of the simulation algorithm are demonstrated by applying it to several well-known model chemical systems, including the Lotka model, the Brusselator, and the Oregonator.

D. Gillespie, J. Phys. Chem. 81 (1977) 2340–2361

Gillespie – Step 0

Consider decay reation: A => Ø (this model describes e.g. the radioactive decay)

Probability for one reaction in (t, $t+\Delta t$) with $A(t)$ molecules = $A(t)\, k\, \Delta t$

Naive Algorithm:

```
A = A0
For every timestep:
        get random number r ε [0, 1)
        if r ≤ A*k*dt:
            A = A-1
```

It works, but: *A*k*dt* << 1 for reasons of (good) accuracy

=> many many steps where nothings happens

=> Use adaptive stepsize method?

Gillespie – Step 1

Idea: Figure out **when** the next reaction will take place!

(In between the discrete events nothing happens anyway … :-)

Suppose there are A(t) molecules in the system at time t

f(A(t), s) = probability that with A(t) molecules the next reaction takes place in interval (t+s, t+s+ds) with ds => 0

g(A(t), s) = probability that with A(t) molecules no reaction occurs in (t, t+s)

Then: $$f(A(t),s)\,ds = g(A(t),s)\,A(t+s)\,k\,ds$$

No reaction during (t, t+s):

$$f(A(t),s)\,ds = g(A(t),s)\,\underbrace{A(t)\,k\,ds}$$

probability for reaction in (t+s, t+s+ds)

Probability for (No Reaction)

Now we need g(A(t), s)

Extend g(A(t), s) a bit:

$$g(A(t), s+ds) = g(A(t), s)\,[1 - A(t+s)\,k\,ds]$$

Replace again *A(t+s)* by *A(t)* and rearrange:

$$\lim_{ds \to 0} \frac{g(A(t), s+ds) - g(A(t), s)}{ds} = \frac{dg(A(t), s)}{ds} = -A(t)k\, g((A(t), s)$$

With g(A, 0) = 1 ("no reaction during no time")

=> Distribution of waiting times between discrete reaction events:

$$g(A(t), s) = \exp[-A(t)\,k\,s]$$

Life time = average waiting time: $s_0 = \dfrac{1}{kA(t)}$

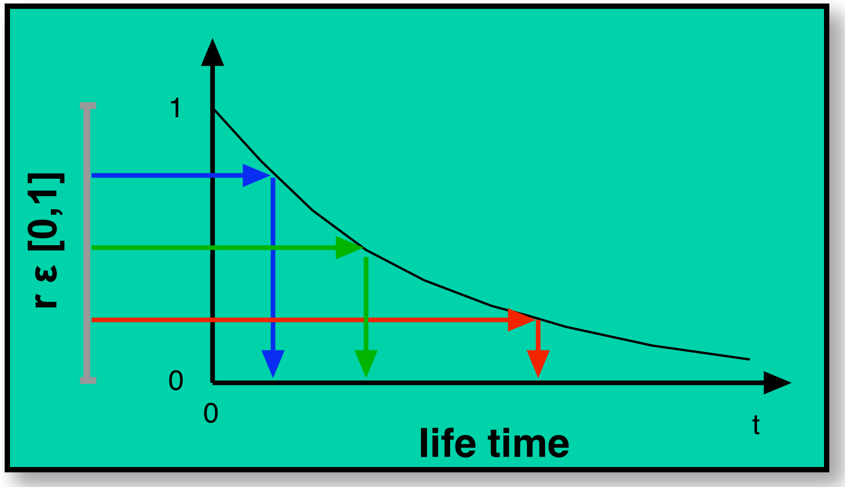
Exponentially Distributed Random Numbers

Exponential probability distribution:

$$g(A(t),s) = \exp[-A(t)ks]$$

Solve $r = \exp[-A(t)ks]$ for s:

$$s = \frac{1}{kA(t)} \ln\left[\frac{1}{r}\right] = \frac{1}{\alpha_0} \ln\left[\frac{1}{r}\right]$$



Simple Gillespie algorithm for the decay reaction A => Ø :

```
A = A0
While(A > 0):
        get random number r ε [0, 1)
        t = t + s(r)
        A = A - 1
```

Gillespie vs. Naive Algorithm

Naive:

"What is the probability that an event will occur during the next Δt?"

=> small fixed timesteps

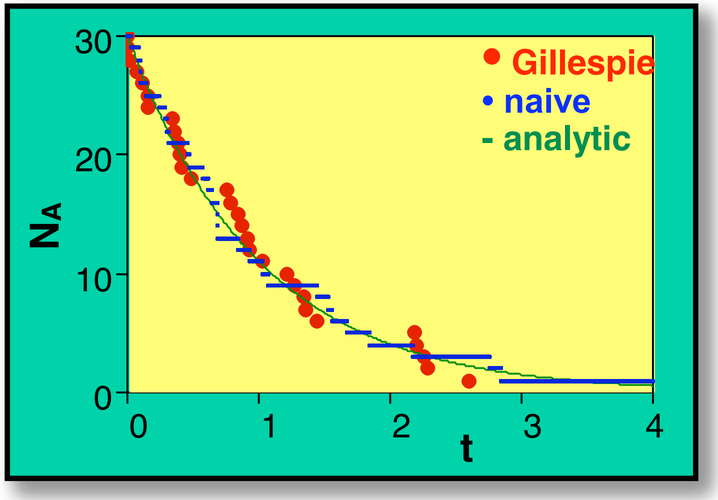
=> 1st order approximation

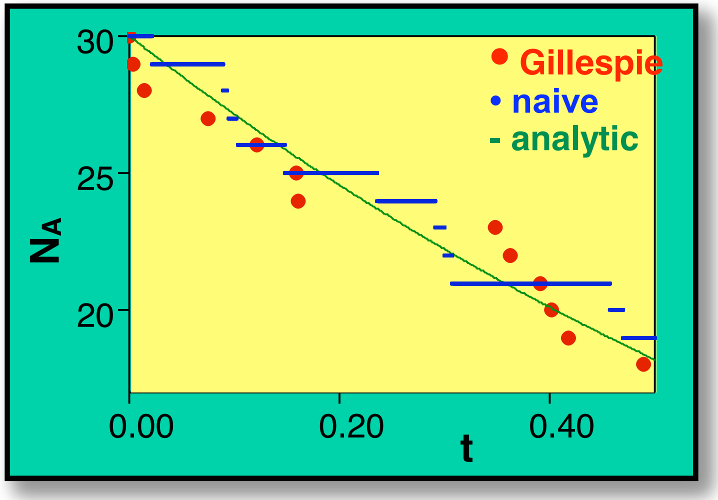
Gillespie:

"How long will it take until the next event?"

=> variable timesteps

=> exact





Gillespie – Complete

For an arbitrary number of reactions (events):

(i) determine probabilities for the individual reactions: α_i $i = 1, \ldots, N$

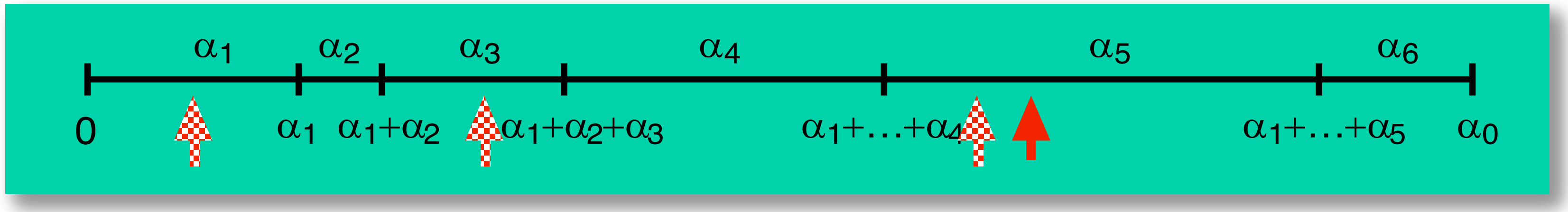
total probability $\alpha_0 = \Sigma\, \alpha_i$

(ii) get time s until next event in any of the reactions:

$$s = \frac{1}{\alpha_0} \ln\left[\frac{1}{r_1}\right]$$

(iii) Choose the next reaction j from:

$$\sum_{i=1}^{j-1} \alpha_i \leq \alpha_0 r_2 < \sum_{i=1}^{j} \alpha_i$$



(iv) update time and particle numbers

An Example with Two Species

Reactions: $A + A \overset{k_1}{\Rightarrow} Ø$ $A + B \overset{k_2}{\Rightarrow} Ø$ $Ø \overset{k_3}{\Rightarrow} A$ $Ø \overset{k_4}{\Rightarrow} B$

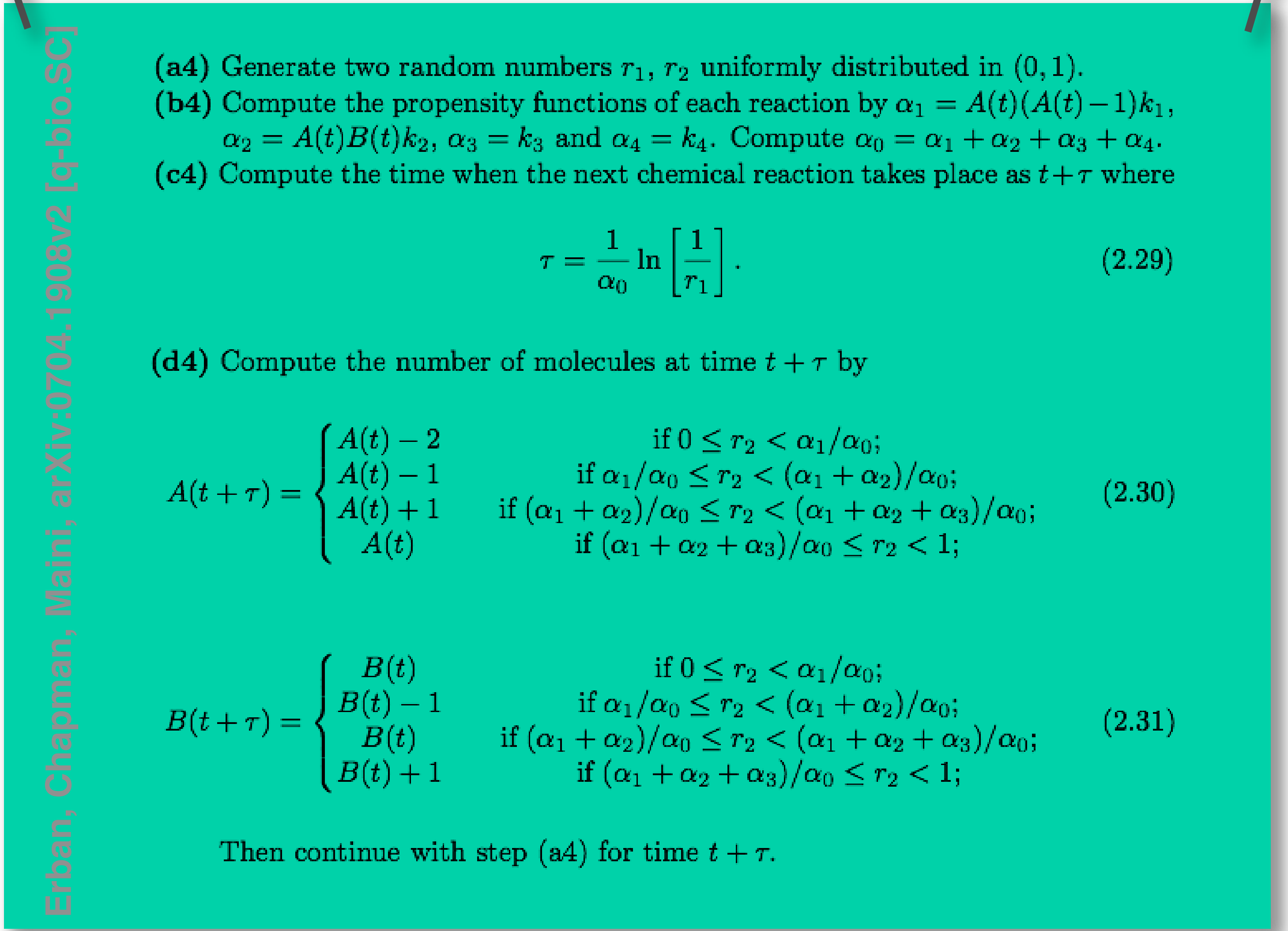
Continuous rate equations: $$\frac{dA}{dt} = k_3 - 2A^2 k_1 - ABk_2$$

Stationary state:

with $k_1 = 10^{-3}\ s^{-1}$ $k_2 = 10^{-2}\ s^{-1}$ $k_3 = 1.2\ s^{-1}$ $k_4 = 1\ s^{-1}$

$\Rightarrow A_{ss} = 10, B_{ss} = 10$

Gillespie Algorithm



(a4) Generate two random numbers r_1, r_2 uniformly distributed in $(0,1)$.

(b4) Compute the propensity functions of each reaction by $\alpha_1 = A(t)(A(t)-1)k_1$, $\alpha_2 = A(t)B(t)k_2$, $\alpha_3 = k_3$ and $\alpha_4 = k_4$. Compute $\alpha_0 = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4$.

(c4) Compute the time when the next chemical reaction takes place as $t+\tau$ where

$$\tau = \frac{1}{\alpha_0} \ln \left[\frac{1}{r_1} \right]. \tag{2.29}$$

(d4) Compute the number of molecules at time $t+\tau$ by

$$A(t+\tau) = \begin{cases} A(t)-2 & \text{if } 0 \le r_2 < \alpha_1/\alpha_0; \\ A(t)-1 & \text{if } \alpha_1/\alpha_0 \le r_2 < (\alpha_1+\alpha_2)/\alpha_0; \\ A(t)+1 & \text{if } (\alpha_1+\alpha_2)/\alpha_0 \le r_2 < (\alpha_1+\alpha_2+\alpha_3)/\alpha_0; \\ A(t) & \text{if } (\alpha_1+\alpha_2+\alpha_3)/\alpha_0 \le r_2 < 1; \end{cases} \tag{2.30}$$

$$B(t+\tau) = \begin{cases} B(t) & \text{if } 0 \le r_2 < \alpha_1/\alpha_0; \\ B(t)-1 & \text{if } \alpha_1/\alpha_0 \le r_2 < (\alpha_1+\alpha_2)/\alpha_0; \\ B(t) & \text{if } (\alpha_1+\alpha_2)/\alpha_0 \le r_2 < (\alpha_1+\alpha_2+\alpha_3)/\alpha_0; \\ B(t)+1 & \text{if } (\alpha_1+\alpha_2+\alpha_3)/\alpha_0 \le r_2 < 1; \end{cases} \tag{2.31}$$

Then continue with step (a4) for time $t+\tau$.

Stochastic Simulation

$A + A \overset{k_1}{\Rightarrow} \varnothing$ $A + B \overset{k_2}{\Rightarrow} \varnothing$ $\varnothing \overset{k_3}{\Rightarrow} A$ $\varnothing \overset{k_4}{\Rightarrow} B$

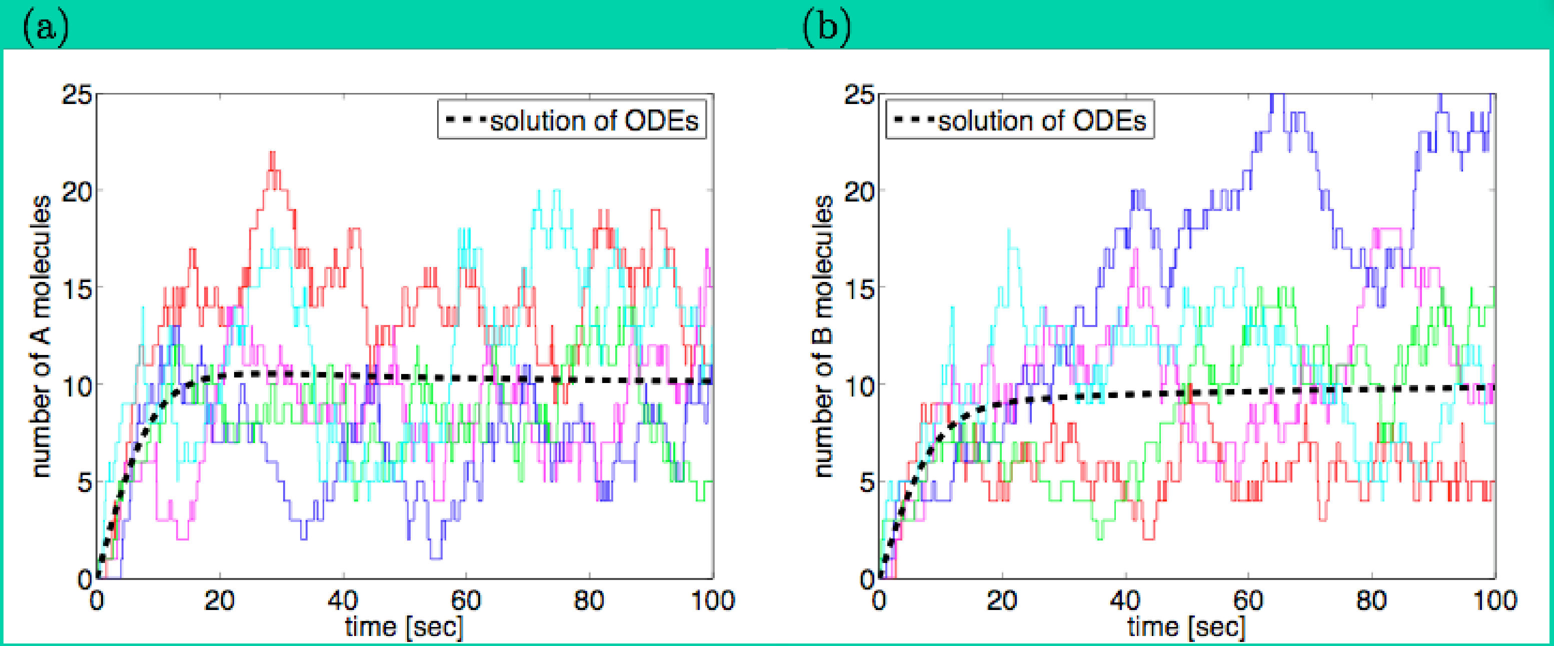


FIG. 2.3. *Five realizations of SSA (a4)–(d4). Number of molecules of chemical species A (left panel) and B (right panel) are plotted as functions of time as solid lines. Different colours correspond to different realizations. The solution of* (2.33)–(2.34) *is given by the dashed line. We use* $A(0) = 0$, $B(0) = 0$, $k_1 = 10^{-3}\ sec^{-1}$, $k_2 = 10^{-2}\ sec^{-1}$, $k_3 = 1.2\ sec^{-1}$ *and* $k_4 = 1\ sec^{-1}$.

Distribution of Stationary States

$A + A \overset{k_1}{=>} \varnothing$ $\quad$ $A + B \overset{k_2}{=>} \varnothing$ $\quad$ $\varnothing \overset{k_3}{=>} A$ $\quad$ $\varnothing \overset{k_4}{=>} B$

$\mathbf{k_1 = 10^{-3}\ s^{-1}}$ $\quad$ $\mathbf{k_2 = 10^{-2}\ s^{-1}}$ $\quad$ $\mathbf{k_3 = 1.2\ s^{-1}}$ $\quad$ $\mathbf{k_4 = 1\ s^{-1}}$

Continuous model:
$A_{ss} = 10, \quad B_{ss} = 10$

<=>

From long–time Gillespie runs:
$\langle A \rangle = 9.6, \quad \langle B \rangle = 12.2$

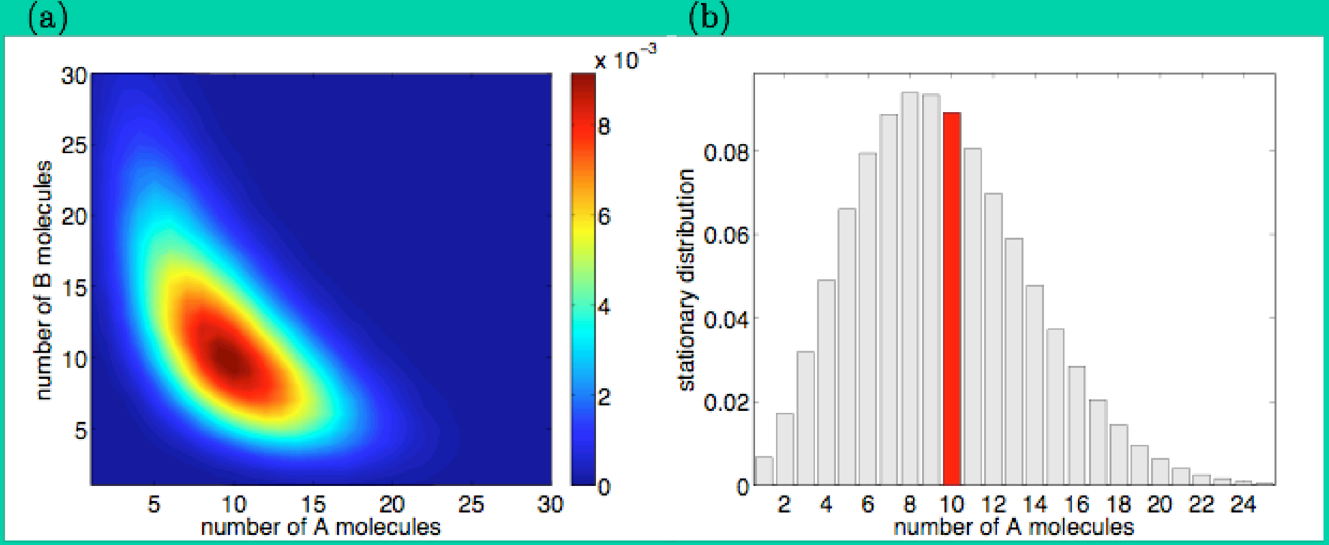


FIG. 2.4. (a) *Stationary distribution* $\phi(n, m)$ *obtained by long time simulation of (a4)–(d4) for* $k_1 = 10^{-3}$ *sec*$^{-1}$, $k_2 = 10^{-2}$ *sec*$^{-1}$, $k_3 = 1.2$ *sec*$^{-1}$ *and* $k_4 = 1$ *sec*$^{-1}$. (b) *Stationary distribution of A obtained by (2.35).*

Stochastic vs. Continuous

For many simple systems:

stochastic solution looks like noisy deterministic solution

Yet in some cases, stochastic description gives qualitatively different results

- swapping between two stationary states
- noise-induced oscillations
- Lotka-Volterra with small populations
- sensitivity in signalling

Two Stationary States

Reactions: $2A \underset{k_2}{\overset{k_1}{\rightleftarrows}} 3A, \quad \emptyset \underset{k_4}{\overset{k_3}{\rightleftarrows}} A$ F. Schlögl, Z. Physik 253 (1972) 147–162

Rate equation: $$\frac{dA}{dt} = k_1 A^2 - k_2 A^3 + k_3 - k_4 A$$

With: $k_1 = 0.18\ \text{min}^{-1}$ $k_2 = 2.5 \times 10^{-4}\ \text{min}^{-1}$ $k_3 = 2200\ \text{min}^{-1}$ $k_4 = 37.5\ \text{min}^{-1}$

Stationary states: $A_{s1} = 100$, $A_{s2} = 400$ (stable) $A_u = 220$ (unstable)

=> Depending on initial conditions (A(0) <> 220),
the deterministic system goes into A_{s1} or A_{s2} (and stays there).

Two States – Stochastic

Erban, Chapman, Maini, arXiv:0704.1908v2

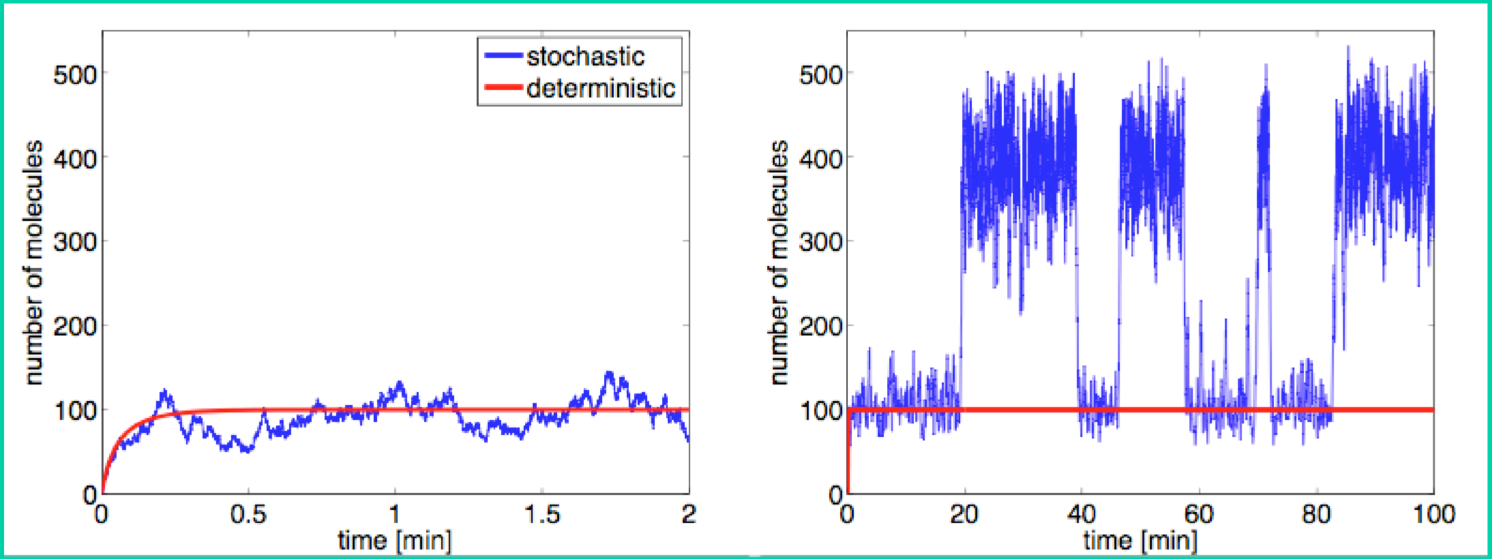


FIG. 5.1. *Simulation of (5.1). One realization of SSA (a5)–(d5) for the system of chemical reactions (5.1) (blue line) and the solution of the deterministic ODE (5.2) (red line).* (a) *The number of molecules of A as a function of time over the first two minutes of simulation.* (b) *Time evolution over 100 minutes.*

=> Fluctuations can drive the system from one stable state into another

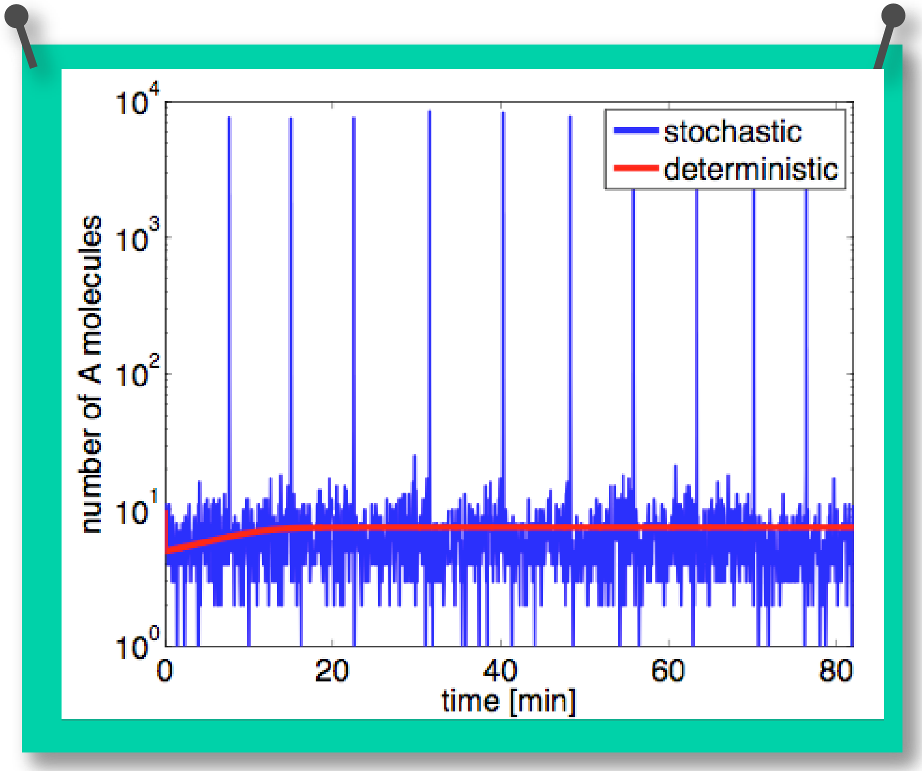
Self-Induced Stochastic Resonance

System $2A + B \xrightarrow{k_1} 3A$ $\quad \varnothing \underset{k_3}{\overset{k_2}{\Longleftrightarrow}} A$ $\quad \varnothing \xrightarrow{k_4} B$

Compare the time evolution from initial state (A, B) = (10, 10) in deterministic and stochastic simulations.

=> deterministic simulation converges to and stays at fixed point (A, B) = (10, 1.1e4)

=> periodic oscillations in the stochastic model



Summary

• Mass action kinetics
  => solving (integrating) differential equations for time-dependent behavior
  => Forward-Euler: extrapolation, time steps

• Stochastic Description
  => why stochastic?
  => Gillespie algorithm
  => different dynamic behavior