Modelling Biological Systems with Differential Equations

Rainer Breitling
(R.Breitling@bio.gla.ac.uk)
Bioinformatics Research Centre
February 2005
Outline

• Part 1: Why modelling?
• Part 2: The statistical physics of modelling: \[ A \rightarrow B \]
  (where do differential equations come from?)
• Part 3: Translating biology to mathematics
  (finding the right differential equations)
Biology = Concentrations
Humans think small-scale…

(the “7 items” rule)

• phone number length (memory constraint)
• optimal team size (manipulation constraint)
• maximum complexity for rational decision making

...but biological systems contain (at least) dozens of relevant interacting components!
Humans think linear...

...but biological systems contain:

- non-linear interaction between components
- positive and negative feedback loops
- complex cross-talk phenomena
The simplest chemical reaction

A → B

- irreversible, one-molecule reaction
- examples: all sorts of decay processes, e.g. radioactive, fluorescence, activated receptor returning to inactive state
- any metabolic pathway can be described by a combination of processes of this type (including reversible reactions and, in some respects, multi-molecule reactions)
The simplest chemical reaction

$$A \rightarrow B$$

various levels of description:

- homogeneous system, large numbers of molecules = ordinary differential equations, *kinetics*
- small numbers of molecules = probabilistic equations, *stochastics*
- spatial heterogeneity = partial differential equations, *diffusion*
- small number of heterogeneously distributed molecules = single-molecule tracking (e.g. cytoskeleton modelling)
Kinetics Description

Main idea: Molecules don’t talk

- Imagine a box containing N molecules.
  How many will decay during time t? \( k \times N \)

- Imagine two boxes containing \( N/2 \) molecules each.
  How many decay? \( k \times N \)

- Imagine two boxes containing N molecules each.
  How many decay? \( 2k \times N \)

- In general:

  \[
  - \frac{dn(t)}{dt} = \lambda \times n(t) \quad \Leftrightarrow \quad n(t) = N_0 e^{-\lambda t}
  \]

  differential equation (ordinary, linear, first-order)  

  exact solution (in more complex cases replaced by a numerical approximation)
Kinetics Description

If you know the concentration at one time, you can calculate it for any other time! (and this really works)
Probabilistic Description

Main idea: Molecules are isolated entities without memory

Probability of decay of a single molecule in some small time interval:

\[ p_1 = \lambda \Delta t \]

Probability of survival in \( \Delta t \):

\[ p_2 = 1 - p_1 = 1 - \lambda \Delta t \]

Probability of survival for some time \( t \):

\[ p = \lim_{x \to \infty} (1 - \lambda \frac{t}{x})^x = e^{-\lambda t} \]

Transition to large number of molecules:

\[ n(t) = N_0 e^{-\lambda t} \quad \text{or} \quad \frac{dn(t)}{dt} = -\lambda N_0 e^{-\lambda t} = -\lambda n(t) \]
Probabilistic Description – 2

Probability of survival of a single molecule for some time $t$:

$$p = \lim_{x \to \infty} (1 - \lambda \frac{t}{x})^x = e^{-\lambda t}$$

Probability that exactly $x$ molecules survive for some time $t$:

$$p_x = \left(e^{-\lambda t}\right)^x (1 - e^{-\lambda t})^{N_0 - x} \binom{N_0}{x}$$

Most likely number to survive to time $t$:

$$\max(x \mid p_x) = N_0 e^{-\lambda t}$$
Probabilistic Description – 3
Markov Model (pure death!)

Decay rate: 
\[ \Lambda(n, t) = n\lambda \]

Probability of decay: 
\[ p = \Lambda(n, t)dt \]

Probability distribution of \( n \) surviving molecules at time \( t \):

Description:

Time: \( t \rightarrow \text{wait } dt \rightarrow t+dt \)

Molecules:
- \( n \rightarrow \text{no decay } \rightarrow n \)
- \( n+1 \rightarrow \text{one decay } \rightarrow n \)

Final Result (after some calculating): The same as in the previous probabilistic description
Spatial heterogeneity

• concentrations are different in different places, \( n = f(t,x,y,z) \)

• diffusion superimposed on chemical reactions:

\[
\frac{\partial n(t)}{\partial t} = -\lambda n(t) \pm \text{diffusion}
\]

• partial differential equation
Spatial heterogeneity

- one-dimensional case (diffusion only, and conservation of mass)

\[
\frac{\partial n(t, x)}{\partial t} \Delta x = \text{inflow} - \text{outflow}
\]

outflow = \(-K \frac{\partial n(t, x + \Delta x)}{\partial x}\)

inflow = \(-K \frac{\partial n(t, x)}{\partial x}\)
Spatial heterogeneity – 2

\[
\frac{\partial n(t, x)}{\partial t} \Delta x = K \frac{\partial n(t, x + \Delta x)}{\partial x} - K \frac{\partial n(t, x)}{\partial x}
\]

Transition to differential equation to get diffusion equation:

\[
\frac{\partial n(t, x)}{\partial t} = K \frac{\partial^2 n(t, x)}{\partial x^2}
\]

Shorthand for three dimensions:

\[
\frac{\partial n(t, x, y, z)}{\partial t} = K \nabla^2 n(t, x, y, z)
\]

Combination with chemical reaction:

\[
\frac{\partial n(t)}{\partial t} = -\lambda n(t) + K \nabla^2 n(t)
\]
Summary of Physical Chemistry

• Simple reactions are easy to model accurately
• Kinetic, probabilistic, Markovian approaches lead to the same basic description

\[ \frac{dn(t)}{dt} = -\lambda n(t) \leftrightarrow n(t) = N_0 e^{-\lambda t} \]

• Diffusion leads only to slightly more complexity
• Next step: Everything is decay...
Some (Bio)Chemical Conventions

Concentration of Molecule A = [A], usually in units mol/litre (molar)

Rate constant = k, with indices indicating constants for various reactions (k₁, k₂...)

Therefore:

\[
A \rightarrow B
\]

\[
\frac{d[A]}{dt} = - \frac{d[B]}{dt} = -k_1[A]
\]
Description in MATLAB:
1. Simple Decay Reaction

M-file (description of the model)

```matlab
function dydt = decay(t, y)
% A -> B or y(1) -> y(2)
k = 1;
dydt = [-k*y(1)
        k*y(1)];
```

Analysis of the model

```matlab
>> [t y] = ode45(@decay, [0 10], [5 1]);
>> plot (t, y);
>> legend ('[A]', '[B]');
```
Decay Reaction in MATLAB
Reversible, Single-Molecule Reaction

A $\leftrightarrow$ B, or A $\rightarrow$ B $\parallel$ B $\rightarrow$ A, or

Differential equations:

- Forward:
  \[
  \frac{d[A]}{dt} = -k_1[A] + k_2[B]
  \]

- Reverse:
  \[
  \frac{d[B]}{dt} = k_1[A] - k_2[B]
  \]

Main principle: Partial reactions are independent!
Reversible, single-molecule reaction – 2

Differential Equation:

\[
\frac{d[A]}{dt} = -k_1[A] + k_2[B]
\]

\[
\frac{d[B]}{dt} = k_1[A] - k_2[B]
\]

Equilibrium (=steady-state):

\[
\frac{d[A]_{equi}}{dt} = \frac{d[B]_{equi}}{dt} = 0
\]

\[-k_1[A]_{equi} + k_2[B]_{equi} = 0
\]

\[
[A]_{equi} = \frac{k_2}{k_1} = K_{equi}
\]

\[
[B]_{equi}
\]
Description in MATLAB:

2. Reversible Reaction

M-file (description of the model)

```matlab
function dydt = isomerisation(t, y)
% A <-> B or y(1) <-> y(2)
k1 = 1;
k2 = 0.5;

dydt = [-k1*y(1)+k2*y(2)    % d[A]/dt
       k1*y(1)-k2*y(2)     % d[B]/dt
       ];
```

Analysis of the model

```matlab
>> [t y] = ode45(@isomerisation, [0 10], [5 1]);
>> plot (t, y);
>> legend ('[A]', '[B]');
```
Isomerization Reaction in MATLAB
Isomerization Reaction in MATLAB

If you know the concentration at one time, you can calculate it for any other time... so what would be the algorithm for that?
Euler’s method - pseudocode

\[ y_{n+1} = y_n + hf(t_n, y_n) \]

1. define \( f(t, y) \)
2. input \( t_0 \) and \( y_0 \).
3. input \( h \) and the number of steps, \( n \).
4. for \( j \) from 1 to \( n \) do
   a. \( m = f(t_0, y_0) \)
   b. \( y_1 = y_0 + h\cdot m \)
   c. \( t_1 = t_0 + h \)
   d. Print \( t_1 \) and \( y_1 \)
   e. \( t_0 = t_1 \)
   f. \( y_0 = y_1 \)
5. end
Euler’s method in Perl

\[ y_{n+1} = y_n + hf(t_n, y_n) \]

```perl
sub ode_euler {
    my ($t0, $t_end, $h, $yref, $dydt_ref) = @_;  # Euler's method in Perl
    my @y = @$yref;  # y0 = yref
    my @solution;
    for (my $t = $t0; $t < $t_end; $t += $h) {
        push @solution, [@y];  # Store the current solution y
        my @dydt = &$dydt_ref(
            \@y, $t);  # Solve the differential equation
        foreach my $i (0..$#y) {
            $y[$i] += ($h * $dydt[$i]);  # Update y for the next step
        }
    }
    push @solution, [@y];  # Store the final solution
    return @solution;
}
```

**Note:** The above code snippet implements Euler's method for solving ordinary differential equations (ODEs) in Perl. The function `ode_euler` takes the initial time `$t0`, the final time `$t_end`, the step size `$h`, the reference to the initial condition `$yref`, and the reference to the derivative function `$dydt_ref` as inputs. It calculates the solution at discrete points in time using the Euler's method formula. The result is a list of solution values at each time step.
Euler’s method in Perl

#!/usr/bin/perl -w

use strict;

my @initial_values = (5, 1);

my @result = ode_euler (0, 10, 0.01, \@initial_values, \&dydt);

foreach (@result) {
    print join " ", @$_, "\n";
}

exit;

% simple A <-> B reversible mono-molecular reaction

sub dydt {
   my $yref = shift;
   my @y = @$yref;
   my @dydt = ( -$y[0] + 0.5*$y[1],
               +$y[0] - 0.5*$y[1]);
   return @dydt;
}

....
Improving Euler’s method

\[ y_{n+1} = y_n + hf(t_n, y_n) \]

\[ y_{n+1} = y_n + hf\left(t_n + \frac{1}{2} h, y_n + \frac{1}{2} hf(t_n, y_n)\right) \]

(second-order Runge-Kutta method)
Isomerization Reaction in MATLAB
Isomerization Reaction in MATLAB
Irreversible, two-molecule reaction

The last piece of the puzzle

\[ A + B \rightarrow C \]

Differential equations:

\[
\frac{d[A]}{dt} = \frac{d[B]}{dt} = -\frac{d[C]}{dt} \\
\frac{d[A]}{dt} = -k[A][B] \quad \text{Non-linear!}
\]

Underlying idea: Reaction probability = Combined probability that both [A] and [B] are in a “reactive mood”:

\[ p(AB) = p(A)p(B) = k_1^*[A]k_2^*[B] = k[A][B] \]
A simple metabolic pathway

A $\rightarrow$ B $\leftrightarrow$ C + D

Differential equations:

<table>
<thead>
<tr>
<th>d/dt</th>
<th>decay</th>
<th>forward</th>
<th>reverse</th>
</tr>
</thead>
<tbody>
<tr>
<td>[A]=</td>
<td>$-k_1 [A]$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[B]=</td>
<td>$+k_1 [A]$</td>
<td>$-k_2 [B]$</td>
<td>$+k_3 [C] [D]$</td>
</tr>
<tr>
<td>[C]=</td>
<td></td>
<td>$+k_2 [B]$</td>
<td>$-k_3 [C] [D]$</td>
</tr>
<tr>
<td>[D]=</td>
<td></td>
<td>$+k_2 [B]$</td>
<td>$-k_3 [C] [D]$</td>
</tr>
</tbody>
</table>
Metabolic Networks as Bigraphs

A $\rightarrow$ B $\leftrightarrow$ C + D

\[
\begin{array}{c|ccc}
 & k1 & k2 & k3 \\
\hline
A & -1 & 0 & 0 \\
B & 1 & -1 & 1 \\
C & 0 & 1 & -1 \\
D & 0 & 1 & -1 \\
\end{array}
\]

\[
\begin{array}{c|cccc}
\text{d/dt} & \text{decay} & \text{forward} & \text{reverse} \\
\hline
[A] & -k1 [A] & & \\
[B] & +k1 [A] & -k2 [B] & +k3 [C] [D] \\
[C] & & +k2 [B] & -k3 [C] [D] \\
[D] & & +k2 [B] & -k3 [C] [D] \\
\end{array}
\]
Biological description → bigraph → differential equations
Biological description $\rightarrow$ bigraph $\rightarrow$ differential equations
Biological description → bigraph → differential equations
Biological description $\rightarrow$ bigraph $\rightarrow$ differential equations
Biological description → bigraph → differential equations

Fig. courtesy of W. Kolch
Biological description $\rightarrow$ bigraph $\rightarrow$ differential equations
Biological description $\rightarrow$ bigraph $\rightarrow$ differential equations

Fig. courtesy of W. Kolch
The Raf-1/RKIP/ERK pathway

Can you model it?
(11x11 table, 34 entries)
Description in MATLAB:
3. The RKIP/ERK pathway

function dydt = erk_pathway_wolkenhauer(t, y)
% from Kwang-Hyun Cho et al., Mathematical Modeling...

k1 = 0.53;
k2 = 0.0072;
k3 = 0.625;
k4 = 0.00245;
k5 = 0.0315;
k6 = 0.8;
k7 = 0.0075;
k8 = 0.071;
k9 = 0.92;
k10 = 0.00122;
k11 = 0.87;
% continued on next slide...
Description in MATLAB:
3. The RKIP/ERK pathway

dydt = [
    -k1*y(1)*y(2) + k2*y(3) + k5*y(4)
    -k1*y(1)*y(2) + k2*y(3) + k11*y(11)
    k1*y(1)*y(2) - k2*y(3) - k3*y(3)*y(9) + k4*y(4)
    k3*y(3)*y(9) - k4*y(4) - k5*y(4)
    k5*y(4) - k6*y(5)*y(7) + k7*y(8)
    k5*y(4) - k9*y(6)*y(10) + k10*y(11)
    -k6*y(5)*y(7) + k7*y(8) + k8*y(8)
    k6*y(5)*y(7) - k7*y(8) - k8*y(8)
    -k3*y(3)*y(9) + k4*y(4) + k8*y(8)
    -k9*y(6)*y(10) + k10*y(11) + k11*y(11)
    k9*y(6)*y(10) - k10*y(11) - k11*y(11)
];
Description in MATLAB:
3. The RKIP/ERK pathway

Analysis of the model:

```matlab
>> [t y] = ode45(@erk_pathway_wolkenhauer, [0 10], [2.5 2.5 0 0 0 0 2.5 0 2.5 3 0]); % (initial values!)
>> plot (t, y);
>> legend ('[Raf1*]', '[RKIP]', '[Raf1/RKIP]', '[RAF/RKIP/ERK]', '[ERK]', '[RKIP-P]', '[MEK-PP]', '[MEK-PP/ERK]', '[ERK-PP]', '[RP]', '[RKIP-P/RP]');
```
The RKIP/ERK pathway in MATLAB
Further Analyses in MATLAB et al.

All initial concentrations can be varied at will, e.g. to test a concentration series of one component (sensitivity analysis)

Effect of slightly different k-values can be tested (stability of the model with respect to measurement/estimation errors)

Effect of inhibitors of each reaction (changed k-values) can be predicted

Concentrations at each time-point are predicted exactly and can be tested experimentally
Example of Sensitivity Analysis

function [tt,yy] = sensitivity(f, range, initvec, which_stuff_vary, ep, step, which_stuff_show, timeres);

timevec = range(1):timeres:range(2);
vec = [initvec];
[tt y] = ode45(f, timevec, vec);
yy = y(:,which_stuff_show);

for i=initvec(which_stuff_vary)+step:step:ep;
    vec(which_stuff_vary) = i;
    [t y] = ode45(f, timevec, vec);
    tt = [t];
    yy = [yy y(:,which_stuff_show)];
end
Example of Sensitivity Analysis

```matlab
>> [t y] =
    sensitivity(@erk_pathway_wolkenhauer, [0 1], [2.5 2.5 0 0 0 0 2.5 0 2.5 3 0], 5, 6, 1, 8, 0.05);
```

>> surf (y);

varies concentration of m5 (ERK-PP) from 0..6, outputs concentration of m8 (ERK/MEK-PP), time range [0 1], steps of 0.05. Then plots a surface map.
Example of Sensitivity Analysis

after Cho et al. (2003) CSMB
Example of Sensitivity Analysis

(longer time course)
Conclusions and Outlook

• differential equations allow exact predictions of systems behaviour in a unified formalism
• modelling = *in silico* experimentation
• difficulties:
  – translation from biology
    • modular model building interfaces, e.g. Gepasi/COPASI, Genomic Object Net, E-cell, Ingeneue
  – managing complexity explosion
    • pathway visualization and construction software
    • standardized description language, e.g. Systems Biology Markup Language (SBML)
  – lack of biological data
    • perturbation-based parameter estimation, e.g. metabolic control analysis (MCA)
    • constraints-based modelling, e.g. flux balance analysis (FBA)
    • semi-quantitative differential equations for inexact knowledge