Deterministic modelling and stochastic simulation of biochemical pathways using MATLAB

M. Ullah, H. Schmidt, K.-H. Cho and O. Wolkenhauer

Abstract: The analysis of complex biochemical networks is conducted in two popular conceptual frameworks for modelling. The deterministic approach requires the solution of ordinary differential equations (ODEs, reaction rate equations) with concentrations as continuous state variables. The stochastic approach involves the simulation of differential-difference equations (chemical master equations, CMEs) with probabilities as variables. This is to generate counts of molecules for chemical species as realisations of random variables drawn from the probability distribution described by the CMEs. Although there are numerous tools available, many of them free, the modelling and simulation environment MATLAB is widely used in the physical and engineering sciences. We describe a collection of MATLAB functions to construct and solve ODEs for deterministic simulation and to implement realisations of CMEs for stochastic simulation using advanced MATLAB coding (Release 14). The program was successfully applied to pathway models from the literature for both cases. The results were compared to implementations using alternative tools for dynamic modelling and simulation of biochemical networks. The aim is to provide a concise set of MATLAB functions that encourage the experimentation with systems biology models. All the script files are available from www.sbi.uni-rostock.de/publications_matlab-paper.html.

1 Introduction

Modelling and simulation of pathways as networks of biochemical reactions have received increased interest in the context of systems biology [1]. There are two popular frameworks for modelling intracellular dynamics. The deterministic modelling is based on the construction of a set of rate equations to describe the reactions in the biochemical pathways of interest. These rate equations are in fact non-linear ordinary differential equations (ODEs) with concentrations of chemical species as variables [2–5]. Deterministic simulation produces concentrations by solving the ODEs. The stochastic modelling involves the formation of a set of chemical master equations (CMEs) with probabilities as variables [6]. Stochastic simulation, however, produces counts of molecules of some chemical species as realisations of random variables drawn from the probability distribution described by the CMEs. Which framework is appropriate for a given biological system is not only a question of what biological phenomena is investigated but also influenced by assumptions one makes to simplify the analysis [7]. In most cases, neither ODEs nor CMEs have a simple closed form solution, in which case one resorts to numerical simulations.

A large number of software tools are available for both deterministic and stochastic simulations (see www.sbml.org). For a survey of tools, we will refer the readers Huang and Ferrell [8] and a collection of models [9]. The aim of this article is not to argue for one or the other tool, but to provide a small collection of files that will allow MATLAB [10] users to model and simulate biochemical pathways without resorting to specialist tools or toolboxes and to encourage experimentation with models. MATLAB is a commercial tool that is widely used in the physical and engineering sciences and frequently available at universities. The main advantage of MATLAB over C/C^{++}/Java and so on is that MATLAB is a scripting language, allowing the user to easily edit available functions according to his needs. MATLAB has other functionalities including ready access to data analysis and optimisation tools and advanced visualisation techniques and a programmable graphical user interface. Moreover, MATLAB has built-in vector/matrix representations and manipulations, which can be exploited for speedy simulation and comes with a handful of toolboxes that simplifies coding. The structure of these scripts is compact, and they are easy to understand and can be modified to one's own particular needs.

2 ODE modelling

In our ODE framework, modelling a pathway involves two steps. First, the pathway is decomposed into a set of elementary reactions for which the information is encoded in three matrices, related to the stoichiometry and rate coefficients of the biochemical system. This information is used to construct and solve the ODEs. It is assumed that a pathway can be decomposed into unidirectional elementary

IEE Proceedings online no. 20050064

doi:10.1049/ip-syb:20050064

Paper first received 5th September and in revised form 23rd November 2005

M. Ullah and O. Wolkenhauer are with the Systems Biology and Bioinformatics Group, University of Rostock, Albert Einstein Street 21, Rostock 18055, Germany

H. Schmidt is with the Fraunhofer-Chalmers Research Centre for Industrial Mathematics, Chalmers Science Park, Göteborg SE-412 88, Sweden

K.-H. Cho is with the College of Medicine and Korea Bio-MAX Institute, Seoul National University, Seoul 151-818, Korea

E-mail: ow@informatik.uni-rostock.de

reactions. Then law of mass action can be applied to each elementary reaction to obtain ODEs. We shall discuss other approaches to construct the right-hand side of the following differential equations. The flexibility of MATLAB documents also allows other approaches (e.g. power law representations or S-systems) which are not discussed here. Most of the elementary reactions can be grouped into three categories. Association: where an enzyme binds to a substrate to form a complex. Dissociation: where the complex dissociates back into the substrate and enzyme [11]. Product formation: where the enzyme converts the substrate into the product. To set up a general mathematical framework, consider a biochemical reaction network or pathway involving N molecular species. To apply the law of mass action, the network is decomposed into M elementary reactions (also called reaction channels). An elementary reaction will be written as

$$R_j\colon \sum_{i=1}^{N} l_{ij}\mathrm{X}_i \xrightarrow{k_j} \sum_{i=1}^{N} r_{ij}\mathrm{X}_i, \quad j = 1, \ldots, M \tag{1}$$

where X_i is ith chemical species; R_j, jth elementary reaction; l_{ij}, r_{ij}, stoichiometric coefficients and k_j, rate coefficient for R_j.

The stoichiometric coefficients are non-negative integers. If a species X_i is not a reactant in reaction R_j, l_{ij} is zero. Similarly $r_{ij} = 0$, if X_i is not a product in R_j. We will be using the simplified notation x_i for the molar concentration $[\mathrm{X}_i]$ and $\boldsymbol{x} = [x_i]_{N\times 1}$ for the vector of concentrations. As R_j is an elementary reaction, one of l_{ij} and r_{ij} must always be zero. This means that the step change in x_i, after completion of R_j denoted $d_{ij} = r_{ij} - l_{ij}$, is given by

$$d_{ij} = \begin{cases} -l_{ij} & \text{if } \mathrm{X}_i \text{ is a reactant in } R_j \\ +r_{ij} & \text{if } \mathrm{X}_i \text{ is a product in } R_j \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

We can thus write for the reaction rate v_j of R_j

$$v_j(x) = k_j \prod_{i=1}^{N} x_i^{l_{ij}}(t) \tag{3}$$

and for the system of ODEs describing how each concentration changes over time

$$\dot{x}_i(t) = \sum_{j=1}^{M} d_{ij} v_j(x) \tag{4}$$

For a more formal development of (1)–(4), we will refer the Heinrich and Schuster and Wolkenhauer *et al.* [12–14]. Substituting (3) into (4) and writing $p_j(x) = \prod_{i=1}^{N} x_i^{l_{ij}}(t)$ we obtain a more practical form

$$\dot{x}_i(t) = \sum_{j=1}^{M} d_{ij}\, k_j\, p_j(x) \tag{5}$$

Defining matrices $\boldsymbol{k} = [k_j]_{1\times M}$, $\boldsymbol{L} = [l_{ij}]_{N\times M}$, $\boldsymbol{D} = [d_{ij}]_{N\times M}$ and $\boldsymbol{p}(x) = [p_j(x)]_{1\times M}$, the previous equations can be written in the vector form

$$\dot{\boldsymbol{x}} = \boldsymbol{V}(x) = \boldsymbol{D} * (\boldsymbol{k} \cdot * \boldsymbol{p}(x))' \tag{6}$$

where $*$ denotes matrix multiplication, $\cdot*$ the entry-wise (Hadamard) multiplication, $'$ the transpose of a matrix and $\boldsymbol{V}(x)$ the vector field. Notation for the binary operations in the last equation has been chosen to relate easily to MATLAB. In a more realistic framework, known as generalised mass action (GMA) modelling, the exponents l_{ij} are replaced by non-negative real numbers (kinetic orders) [4].

The modelling approach described earlier can be illustrated by an enzyme-kinetic reaction, as a simple example

$$\mathrm{E} + \mathrm{S} \underset{k_2}{\overset{k_1}{\rightleftharpoons}} \mathrm{ES} \xrightarrow{k_3} \mathrm{E} + \mathrm{P} \tag{7}$$

which can be decomposed into the following three elementary reactions

$$R_1\colon \mathrm{E} + \mathrm{S} \xrightarrow{k_1} \mathrm{ES}$$

$$R_2\colon \mathrm{ES} \xrightarrow{k_2} \mathrm{E} + \mathrm{S}$$

$$R_3\colon \mathrm{ES} \xrightarrow{k_3} \mathrm{E} + \mathrm{P}$$

where E, S, ES and P are enzyme, substrate, complex and product, respectively. The matrices $\boldsymbol{x}$, $\boldsymbol{k}$, $\boldsymbol{L}$ and $\boldsymbol{D}$ will assume the following values for this example

$$\boldsymbol{x} = \begin{bmatrix} [\mathrm{E}] \\ [\mathrm{S}] \\ [\mathrm{ES}] \\ [\mathrm{P}] \end{bmatrix}, \quad \boldsymbol{k} = \begin{bmatrix} k_1 & k_2 & k_3 \end{bmatrix}$$

$$\boldsymbol{L} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{D} = \begin{bmatrix} -1 & 1 & 1 \\ -1 & 1 & 0 \\ 1 & -1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

Substituting these values in (6), we arrive at four ODEs.

Biochemists are commonly interested in the rate or velocity of the enzymatic reaction, denoted V_3 and assume that $\dot{x}_3 = 0$ (quasi-steady state). This gives us the familiar Michaelis–Menten equation

$$V_3 = \dot{x}_4 = -\dot{x}_2 = \frac{k_3 x_1 x_2}{K_m + x_2} \tag{8}$$

where $K_m \equiv (k_2 + k_3)/k_1$ is called Michaelis–Menten constant.

Next, we consider an important aspect of modelling signal transduction pathways, the activation and deactivation through phosphorylation/dephosphorylation, modelled as two enzymatic reactions

$$R_{j1}\colon \mathrm{S}_j + \mathrm{E}_j \underset{d_{j1}}{\overset{a_{j1}}{\rightleftharpoons}} \mathrm{S}_j\mathrm{E}_j \xrightarrow{k_{j1}} X_j$$

$$R_{j2}\colon \mathrm{X}_j + \mathrm{P}_j \underset{d_{j2}}{\overset{a_{j2}}{\rightleftharpoons}} \mathrm{X}_j\mathrm{P}_j \xrightarrow{k_{j2}} \mathrm{S}_j$$

where R_{j1} is jth phosphorylation; R_{j2}, jth dephosphorylation; S_j, unphosphorylated protein for R_j; X_j, phosphorylated protein R_j; E_j, kinase for R_j; P_j, phosphatase for R_j; a_{j1}, d_{j1}, k_{j1}, rate coefficients for R_{j1} and a_{j2}, d_{j2}, k_{j2}, rate coefficients for R_{j2}.

In the literature, not all these rate constants are available at all the time. If only k_{j1}, k_{j2}, K_{j1}, K_{j2} are available, where K_{j1}, K_{j2} are Michaelis–Menten constants, the reaction velocities for each step can be written using (8)

$$\underbrace{V_{j1} = \frac{k_{j1} e_j s_j}{K_{j1} + s_j}}_{\text{phosphorylation}}, \quad \underbrace{V_{j2} = \frac{k_{j2} p_j x_j}{K_{j2} + x_j}}_{\text{dephosphorylation}} \tag{9}$$

with the rate equation

$$\dot{x}_j(t) = V_{j1} - V_{j2} \tag{10}$$

Any coupling between R_j with other steps can be incorporated into the last equation with an additive term as needed. The introduction of such feedback loops and experimentation with their effect can easily be done in these MATLAB functions. In most cases, the phosphatase p_j is assumed constant and lumped into the rate constant k_{j2}.

We thus have two ways of ODE modelling for pathways. Decomposition into elementary reactions leads to (6), whereas decomposition into phosphorylation/dephosphorylation steps leads to (9). Both of these sets of equations have a form, which can be implemented in MATLAB very efficiently by utilising the vectorised code. Examples will be given in the following sections.

3 Stochastic modelling

The stochastic framework involves the same decomposition of a pathway into elementary reactions. However, here we are looking at populations or counts of molecules, $\#X_i$, for each species X_i. The goal is to determine the probability distribution which is defined as

$$P_{\boldsymbol{n}}(t) \triangleq \text{Prob}\{\#X(t) = \boldsymbol{n}\} \tag{11}$$

where $\#X(t) = [\#X_i]_{N\times 1}$ and $\boldsymbol{n} \in \mathbb{Z}_+^N$ denotes the vector of counts of molecules for all molecular species involved. The vector of step changes after the completion of elementary reaction R_j will be denoted by $\boldsymbol{D}_j \in \mathbb{Z}^N$. The chemical master equation for this system takes the form

$$\frac{dP_{\boldsymbol{n}}(t)}{dt} = \sum_{j=1}^{M} [a_j(\boldsymbol{n} - \boldsymbol{D}_j)P_{\boldsymbol{n}-\boldsymbol{D}_j}(t) - a_j(\boldsymbol{n})P_{\boldsymbol{n}}(t)] \tag{12}$$

where $a_j(\boldsymbol{n})$ is referred to as the propensity of reaction R_j, defined such that $a_j(\boldsymbol{n})\,dt$ is the probability of reaction R_j occurring in the time interval $[t, t+dt]$, given that $\#X(t) = \boldsymbol{n}$. For intuitive proofs of this equation, we will refer the readers to van Kampen and Wolkenhauer *et al.* [6, 7].

As a simple example, consider the stochastic gating of an ensemble of ion channels. Each channel has two states, one closed (C) and the other open (O) [5], which is equivalent to an isomerisation reaction where a molecule changes its conformation

$$C \underset{\beta}{\overset{\alpha}{\rightleftharpoons}} O \quad \text{alternatively } X_c \underset{\beta}{\overset{\alpha}{\rightleftharpoons}} X_o \tag{13}$$

Let us write $\boldsymbol{n}_T$ for the number of channels, and let $P_C(\boldsymbol{n}, t)$ and $P_O(\boldsymbol{n}, t)$ be the probabilities of having n channels in state O and C, respectively. As the presence of $\boldsymbol{n}$ open channels implies $\boldsymbol{n}_T - \boldsymbol{n}$ closed channels, we have

$$P_C(\boldsymbol{n}, t) = P_O(\boldsymbol{n}_T - \boldsymbol{n}, t), \quad 0 \le \boldsymbol{n} \le \boldsymbol{n}_T$$

We therefore need only one differential equation. Writing $P_{\boldsymbol{n}}(t) = P_O(\boldsymbol{n}, t)$, we arrive at the CME [5]

$$\begin{aligned}\frac{dP_{\boldsymbol{n}}(t)}{dt} &= (\boldsymbol{n}_T - \boldsymbol{n} + 1)\alpha P_{\boldsymbol{n}-1}(t) - (\boldsymbol{n}_T - \boldsymbol{n})\alpha P_{\boldsymbol{n}}(t) \\ &\quad + (\boldsymbol{n} + 1)\beta P_{\boldsymbol{n}+1}(t) - \boldsymbol{n}\beta P_{\boldsymbol{n}}(t)\end{aligned} \tag{14}$$

Formal analysis of even simple CMEs like (14) is difficult. One possibility is to approximate the CME by a truncated Taylor expansion [6, 15, 16]. However, it is often more practical to simulate the CME directly. Each of the elementary reactions of the system (1) can be modelled as a continuous Markov process [5]. Thus, Monte Carlo methods [5] can be employed to simulate such systems as cascades of Markov processes. For a specific class of Markov processes, Gillespie [17] developed an efficient algorithm which has been frequently used [18–21]. A comparison of these approaches and biological examples can easily be realised with the MATLAB functions provided here.

4 Methods

The script files are written and optimised for MATLAB 7 (Release 14). In the following two sections, we will explain the MATLAB scripts developed to realise deterministic and stochastic simulations. The programs are written in a way to facilitate entering information about the pathway to be analysed. It will turn out that entering pathway models only involves a few matrices. Moreover, the vectorised features of MATLAB ensure that each line of code corresponds intuitively to its mathematical counterpart in the model. The purpose of describing the MATLAB code here is not only to encourage the reader to experiment with models but also to show how advanced MATLAB functions can be used to obtain compact programs.

4.1 Deterministic simulation

4.1.1 Mass action kinetics: For the first part of our toolbox, our goal is to encode the vector field $\boldsymbol{V}(x)$. Towards this end, the idea of a function handle in MATLAB is used. A function handle stores the information about a function or expression and can be used in other MATLAB scripts/functions, as if it was a function per se. The following two lines of code will return a function handle to compute $\boldsymbol{V}(x)$

```
M1s=ones(size(k));
V=@(t, x) D*(k.*prod(x(:,M1s).^L))';
```

where `D`, `k`, `L` and `x` are MATLAB variables representing, respectively, $\boldsymbol{D}$, $\boldsymbol{k}$, $\boldsymbol{L}$ and $\boldsymbol{x}$; and the function handle `V` represents the vector field $\boldsymbol{V}(x)$. Note the simplicity of this code and its intuitive resemblance to (6), its mathematical counterpart. As far as biochemical pathways are concerned, most of the elements in the matrices `D` and `L` are zeros, a fact not exploited by the earlier-mentioned code. To exploit this, an optimised form to realise $\boldsymbol{V}(x)$ is the following function `gma`:

```
function V=gma(D,k,L)
if nargin<3
  L=-D.*(D<0);
  i2=L>1;
else
  i2=L>0 & L~=1;
end
i0=L==0;
M1s=ones(size(k));
V=@Vfn;
  function xd=Vfn(t,x)
  X=x(:,M1s);
  X(i0)=1;
  X(i2)=X(i2).^L(i2);
  xd=D*(k.*prod(X))';
  end
end.
```

It can be seen that computationally intensive operations like products and raising powers have been significantly reduced in this new function. This function will be subsequently used to construct the differential equations from information encoded in $\boldsymbol{D}$, $\boldsymbol{k}$ and $\boldsymbol{L}$. The use of the nested function `Vfn` in the earlier-mentioned code is a useful feature of MATLAB 7. For a specific pathway, the function `gma`

has to be called only once to generate V. Subsequently, whenever V is called as a function in simulations, only the nested function will be executed. This improves the speed of simulations, particularly when V is used many times, for example, in bifurcation analysis [22] and stimulus-response plots [11, 23]. Note that this code can be used for two different modelling frameworks. For mass action models, the function gma can be called with just the first two input arguments because the last argument, L, can be computed from the matrix D, according to (2). For the case of GMA models, additionally the matrix L has to be provided.

The enzyme-kinetic reaction (7) corresponds to only a single phosphorylation or dephosphorylation step in a signalling cascade. For more complex pathways, the size of these matrices will naturally grow. For example, the well-known mitogen-activated protein kinase (MAPK) signalling cascade [11], in its simplest form (without feedback), comprises 30 elementary reactions, involving 22 chemical species. The respective sizes of L,D and k will be 30 × 30, 22 × 30 and 30 × 1. In this case, it may not be practical to input these matrices during the MATLAB user interface. An intuitive approach is to encode these matrices in Microsoft Excel spreadsheets by the inspection of the pathway. These spreadsheets can be imported into the MATLAB very easily by calling the MATLAB function xlsread. This is described in further details subsequently. A more practical approach is to compute the matrices from an SBML model. For the convenience of the user, we have included a function importSBML. The calling syntax for the function is

```
[D,k,x0]=importSBML
```

where D and k are as defined previously and x0 is the vector of initial conditions (concentrations). The conclusion is that the function gma presented here is general and can be used to compute reaction rates for larger pathways.

The second part of our small toolbox calls a MATLAB ODE solver for a numerical simulation of the model. The common syntax of calling a solver in MATLAB looks like this

```
[t,x]=ode15s(V,tspan,x0);
```

where V is the vector of function handles returned by the function gma, tspan a vector specifying the time span of simulation and ode15s the MATLAB solver chosen. Such solvers are standard MATLAB functions. A range of other ODE solvers are offered if there are particular numerical requirements.

The following MATLAB code illustrates the simulation of the enzyme-kinetic example (7).

```
x0=[0.5; 1; 0; 0];
D=[-1 1 1;-1 1 0;1 -1 -1;0 0 1];
L=-D.*(D<0);
k=60*[10 2 0.02];
M1s=ones(size(k));
V=@(t,x) D*(k.*prod(x(:,M1s).^L)).';
[t,x]=ode15s(V, [0 10], x0);
```

Once the information about the pathway is available, only two lines of code are required to construct and simulate the system.

4.1.2 Michaelis—Menten kinetics: Next we consider the implementation of (9) and (10). The form of these equations lends itself naturally to a vectorised code. Assume the variables s,e,k,K and V are defined in the MATLAB workspace, corresponding to the variables in (9) and (10),

s : vector of all s_j

e : vector of all e_j

k : matrix { 1st column: vector of all k_{j1}; 2nd column: vector of all k_{j2}

K : matrix { 1st column: vector of all K_{j1}; 2nd column: vector of all K_{j2}

V : matrix { 1st column: vector of all V_{j1}; 2nd column: vector of all V_{j2}

xd : vector of all $\dot{x}_j$

Note that the vector of phosphatases p has been assumed constant and lumped into the vector of rate constants k(:,2). The following lines of the MATLAB code are needed to construct ODEs for a pathway, on the basis of the Michaelis–Menten kinetics.

```
V(:,1)=k(:,1).*e.*s./(K(:,1)+s);
V(:,2)=k(:,2).*x./(K(:,2)+x);
xd=V(:,1) - V(:,2);
```

4.2 Stochastic simulation

Two widely used techniques for stochastic simulations are Monte Carlo simulation [5] and Gillespie's method [17]. Monte Carlo simulation involves generating uniform random numbers. For the two-state ion channel model (13), we would choose a random number Y uniformly distributed on the interval [0, 1] and make a transition (or not) in a time step of duration Δt based on the subinterval into which Y falls. An open channel will close if $0 \leq Y \leq \beta\Delta t$ and a closed channel will open if $1 - \alpha\Delta t \leq Y \leq 1$. Gillespie's method exploits the fact that the time to the next transition (or next reaction) is an exponentially distributed random variable with the reciprocal of transition rate (or rate coefficient) as the mean. Thus we can simulate the two state channel by alternatively choosing open and closed dwell times consistent with these distributions. The method is exact because there is no time step involved. However, it assumes that transition rates (α, β) are functions of the state n only. When transition rates vary with time through variables other than n, Gillespie's method cannot be applied.

In complex systems, like (1), more than one possible transition (reaction) may contribute to the dwell time for a given state. Then Gillespie's method becomes involved, requiring the computation of stochastic rate constants c_j given by

$$c_j = \frac{k_j}{(N_A V)^{K_j - 1}} \prod_{i=1}^{N} l_{ij}! \tag{15}$$

for each reaction R_j, where V is the volume, N_A the Avogadro's constant and $K_j = \sum_{i=1}^{N} l_{ij}$. The other important quantity required by the algorithm is prospensity a_j given by

$$a_j = c_j \prod_{i=1}^{N} \binom{n_i}{l_{ij}} \tag{16}$$

where $\binom{n_i}{l_{ij}}$ is the binomial coefficient. For proofs of (15) and (16) we will refer the readers Wolkenhauer *et al.* and

Gillespie [7, 17]. In light of the terminology in (1), a brief outline of Gillespie's algorithm is

- Initialisation:
  - load reactions and compute c_j
  - t, $\boldsymbol{n}$, random number generator
- Iteration:
  - compute a_j for each reaction R_j
  - $a^* = \sum_{j=1}^{M} a_j$
  - generate uniform random numbers r_1, r_2
  - compute $\tau = -(1/a^*)\ln r_1$
  - μ such that $\sum_{j=1}^{\mu-1} a_j \leq r_2 a^* < \sum_{j=1}^{\mu} a_j$
  - set $\boldsymbol{n} = \boldsymbol{n} + \boldsymbol{D}_\mu$
  - set $t = t + \tau$
- Termination:
  - terminate when $t \geq t_{\max}$ or $a^* = 0$

Our implementation of Gillespie's algorithm in MATLAB looks intuitively similar to this pseudocode. The following MATLAB code implements Gillespie's algorithm, provided the MATLAB variables `n0,L,D,c,tmax` are available in the workspace (which is included in the initialisation of the algorithm).

```
[t,n,i1,i2,M1s]=...
 deal(0,n0,L==1,L==2,ones(size(c)));
rand('state', sum(100*clock));
while t <= tmax
   m=n(:,M1s);
   b=double(~L);
   b(i1)=m(i1);
   b(i2)=m(i2).*(m(i2)-1)/2;
   a=c.*prod(b);
   astr = sum(a);
   if ~astr, break, end
   tau=-1/astr*log(rand);
   u = find(cumsum(a)>astr*rand,1);
   n=n+D(:,u);
   t=t+ tau;
end
```

The MATLAB variable `n0` corresponds to the initial population vector n_0, `c` to the vector of stochastic rate constants c, and `tmax` to the final time of simulation. It can be seen that all the computations, which are independent of the time-varying state of the system n, are done outside the `while` loop. In multiple runs of simulation, the `while` loop will be made into a nested function to speed-up simulations, so that the pre-computed quantities are visible to it.

In the following, all the deterministic and stochastic simulations were performed by using simple programs like those explained so far.

5 Results

5.1 ODEs (mass action)

The first example for our simulations is the enzyme-kinetic reaction (7), for which the results are shown in Fig. 1. As a second example, we consider the MAPK pathway model published by Huang and Ferrell [11]. The pathway is composed of 30 elementary reactions, which gives rise to 18 rate equations with 30 parameters. Owing to the large size of the matrices `L,D` and `k` for the considered pathway, we use an Excel spreadsheet to store these

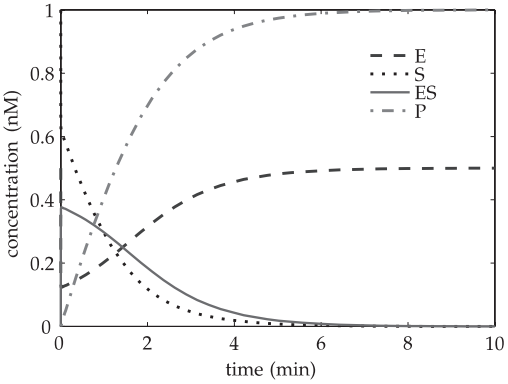

Fig. 1 *Simulation of enzyme-kinetic reaction*

matrices. The data in the spreadsheet need to be structured in the following way:

rows	columns	fill
1	all	comments
all	1	comments
2	3:end	`k`
3:end	2	`x0`
3:end	3:end	`D`

The MATLAB function `xlsread` can be used to import an Excel spreadsheet into MATLAB. The following piece of code imports the spreadsheet `MAPK.xls` into a matrix, extracts the required matrices (`x0,k,D`) and saves them in a compressed format (`.mat`), so that they can be loaded efficiently into MATLAB.

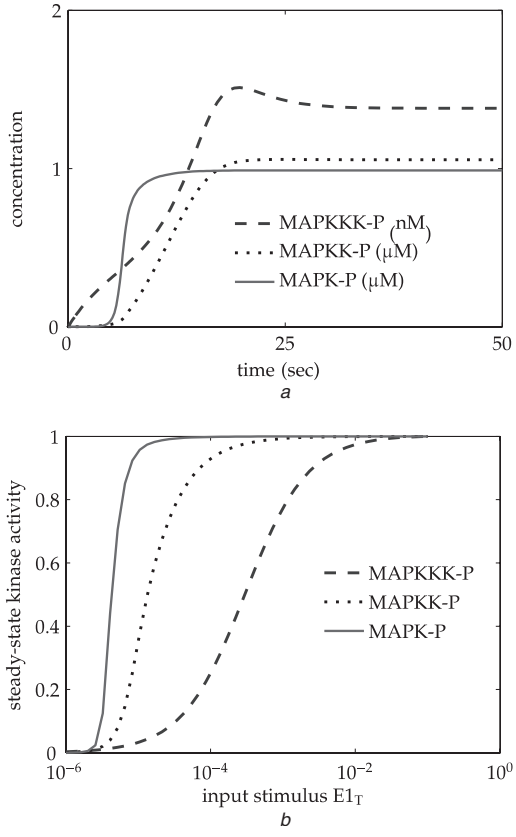

Fig. 2 *Simulation of MAPK pathway*

a Time history
b Stimulus–response curve
For parameters, we refer the readers to Huang and Ferrell [11]

```
T=sparse(xlsread('MAPK.xls'));
x0=T(2:end,1);
k=T(1,2:end);
D=T(2:end, 2:end);
save mapk x0 k D
```

The last line of the previous code generates the data file `mapk.mat` which stores the matrices `x0`, `k` and `D` in the aforementioned form. When these matrices are needed in a simulation script, the MATLAB function `load` is used to import them from `mapk.mat`. This is more efficient than calling `xlsread` repeatedly. If one wishes to change parameters, say for bifurcation analysis, this can be done in the script file, because one needs to change only few parameters. For example, to generate a stimulus–response curve, only one of the initial conditions (the stimulus) is changed in each simulation run. The following piece of code generates the temporal evolution of concentrations for some key variables

```
load mapk
[t,x]=ode15s(gma(D,k),[0 50],x0);
plotyy(t,1e3*x(:,4),t,x(:,[13 21]));
```

which is shown in Fig. 2*a*. The following piece of code generates the stimulus–response curve shown in Fig. 2*b*.

```
load mapk
V=gma(d,k);
tspan=[0 50 100];
n=50;
E1=logspace(-6,-1,n);
xx=x0(:,ones(1,n));
for i=1:n
   x0(2)=E1(i);
   [t,x]=ode15s(V,tspan,x0);
   x0=x(end,:);
   xx(:,i)=x0;
end
xxm=max(xx,[],2);
xx=xx./xxm(:,ones(1,n));
semilogx(E1, xx([4 13 21],:));
```

The computation times to generate these two plots were approximately 1.5 and 1.95 s, respectively, on a Pentium 4 system with a 3 GHz CPU. These two plots for MAPK cascade can be compared with the ones in Huang and Ferrell [11]. It should be noted that stimulus–response curves require multiple simulation runs (50 here) of the system and demonstrate the effectiveness of the previously mentioned code in these situations. To appreciate the simplicity of the MATLAB code for these simulations, a comparison should be made with the Mathematica code used in Huang and Ferrell [11].

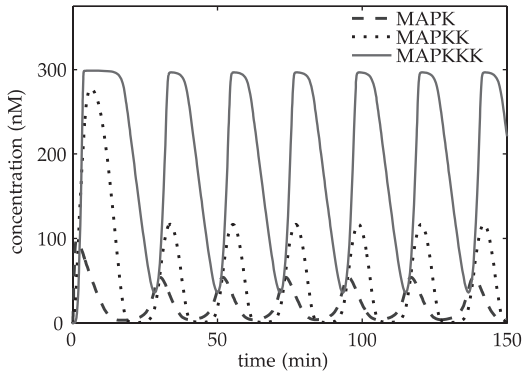

Fig. 3 *Simulation of MAPK pathway with feedback* [24]

5.2 ODEs (Michaelis–Menten)

As an illustration for implementing the Michaelis–Menten kinetics, we consider the MAPK pathway with feedback [24]. The following MATLAB code was written to generate the plot in Fig. 3 which shows that ultrasensitivity combined with feedback gives rise to oscillations.

```
function pathway
k=[2.5.25;.025.75;.025.75; ...
   .025.5;.025.5];
Km=[10 8;15 15;15 15;15 15;15 15];
[xT,x0,KI,n,u]=deal([100;300;300],...
                zeros(5,1),9,1,1);
function xd=rate(t,x)
  s=[xT(1)-x(1);xT(2)-x(2)-x(3); ...
     x(2);xT(3)-x(4)-x(5);x(4)];
  e=[u/(1+(x(5)/KI)^n);x([1;1;3;3])];
  V=[k(:,1).*e.*s./(Km(:,1)+s) ...
     k(:,2).*x./(Km(:,2)+x)];
  xd=V(:,1) - V(:,2);
  xd([2 4])=xd([2 4])-xd([3 5]);
end
[t,x]=ode15s(@rate, [0 150*60], x0);
plot(t/60,x(:,[1 3 5]))
end
```

5.3 CMEs

As an illustration, we have simulated the enzyme-kinetic reaction (7) by employing Gillespie's algorithm for which we gave a MATLAB implementation in Section 4.2. The simulation results, together with deterministic results for comparison, are given in Fig. 4 that uses the parameters: $V = 1$ pL, $[E]_0 = 0.01$, $[S]_0 = 0.1$, $[ES]_0 = [P]_0 = 0$, $k_1 = 10$ (nM s)$^{-1}$, $k_2 = 2$ s^{-1}, $k_3 = 0.02$ s^{-1}.

For a further example, we consider the Li-Rinzil Markov model for stochastic Ca^{2+} release from small clusters of inositol 1,4,5-triphosphate receptors (IP_3Rs). For details and parameters of the model, we will refer the readers Shuai *et al.* [25]. A cluster consists of n_T IP_3 R_s with three stochastic h gates each. The channel will be open only if all the three h gates are open. As specified by Shuai and Jung, α is a function of [IP_3R] treated as stimulus, and hence α is fixed for a fixed stimulus. However, β is a function of [Ca^{2+}] whose release is stochastic because of stochastic opening of IP_3 channels, leading to a variation of β with time, and hence we cannot use Gillespie's algorithm in this case. To determine the channel opening pattern, that is, calcium release, over time, uniform random numbers are generated. For each time step s*t*, we need to generate $3n_T$ random numbers. Again, the

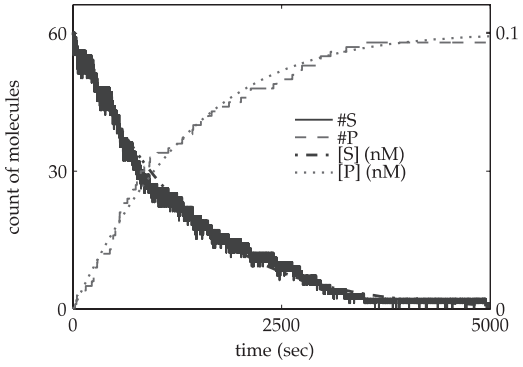

Fig. 4 *Stochastic simulation using Gillespie's algorithm for enzyme-kinetic reaction*

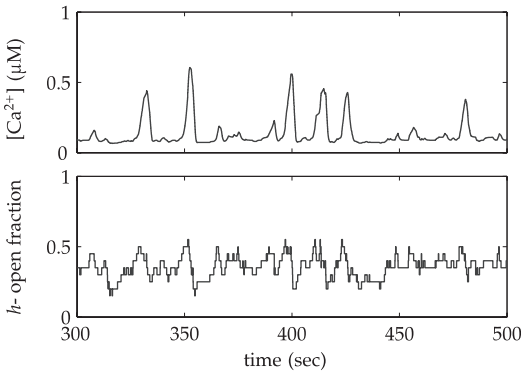

Fig. 5 *Stochastic oscillation of calcium release compared with the fraction of h-open channels for $n_T = 20$, $[IP_3] = 0.3\ \mu M$*

vectorised features of MATLAB can be employed. The following piece of code illustrates the implementation of this alternative method:

```
% random configuration of channels
isop=unidrnd(2,nT,3) - 1;
% generate 3*nT random numbers
y=rand(nT,3);
% find indices for transitions
id=(isop & y<bhdt) | (~isop & y<ahdt);
% make transition based on y
isop(id)=~isop(id);
% find the number of h-open channels
hop(i+1)=numel(nonzeros(all(isop,2)));
```

Note the compact way of generating random numbers for all time points by a single call to MATLAB function `rand` and the resulting ease by which the state of all gates is calculated. These matrix oriented features can be appreciated by comparing this with programs written in other languages. The result of the simulation is shown in Fig. 5 where a calcium trace is compared with the associated trace of the fraction of `h`-open channels. This plot corresponds to Fig. 2 given by Shuai *et al.* [25].

6 Discussion

Systems biology is an art of making appropriate assumptions. It is for this reason that experimentation with models is an important aspect of the modelling cycle. MATLAB is a modelling and simulation language and environment that encourages such experimentation, not only with parameter values but also with model structures. We describe a collection of MATLAB functions to construct and solve ODEs for deterministic simulation and to implement realisations of CMEs for stochastic simulation using advanced MATLAB coding (Release 14). The program was successfully applied to pathway models from the literature for both cases. In deterministic simulations, we included mass action and Michaelis–Menten kinetic models, providing tutorial examples for enzyme-kinetic reaction and (de)activation step in a signal transduction pathway (MAPK model with/without feedback). In stochastic simulation, we covered Markov processes (illustrated with calcium channeling example) and Gillespie's algorithm (illustrated with enzyme-kinetic reaction). The results were compared to implementations using alternative tools for dynamic modelling and simulation of biochemical networks. We observed a style of programming in MATLAB which results in compact code that has a straightforward resemblance to its mathematical counterpart in the model. These programs are compact, easy to modify without compromising on computational power. MATLAB is available in most universities and we hope that our small toolbox and tutorial encourage experimentation with models in systems biology.

We emphasize that the purpose of this article is not to provide a comprehensive toolbox but a small collection of files that invites to experimentation. A fully fledged systems biology toolbox for MATLAB is available from www.sbtoolbox.org.

7 Acknowledgments

O. Wolkenhauer and M. Ullah acknowledge the support of the UK Department for the Environment, Food and Rural Affairs (DEFRA) and the collaboration with the Veterinary Laboratories Agency (VLA) Weybridge, UK. O. Wolkenhauer was further supported through the EU FP6 funded STREP COSBICS. K.-H Cho acknowledges the support received from the Korea Ministry of Science and Technology (Korean Systems Biology Research Grant, M10503010001-05N030100111), the 21C Frontier Microbial Genomics and Application Center Program, Ministry of Science and Technology (Grant MG05-0204-3-0), Republic of Korea, and from the Korea Bio-Hub programme of the Korean Ministry of Commerce, Industry and Energy under grant no. 2005-B0000002. H. Schmidt acknowledges the support by the Swedish Foundation for Strategic Research.

8 References

1 Wolkenhauer, O., and Mesarovic, M.: 'Feedback dynamics and cell function: Why systems biology is called systems biology', *Mol. BioSyst., Royal Soc. Chem.*, 2005, **1**, (1), pp. 14–16
2 Rubinow, S.I.: 'Introduction to mathematical biology' (John Wiley, 1975)
3 Murray, J.D.: 'Mathematical biology' (Springer Verlag, 2002, 3rd Edn.)
4 Voit, E.O.: 'Computational analysis of biochemical systems' (Cambrigde University Press, 2000)
5 Fall, C.P., *et al.*: 'Computational cell biology' (Springer Verlag, 2002)
6 van Kampen, N.G.: 'Stochastic processes in physics and chemistry' (North-Holland, 1992)
7 Wolkenhauer, O., Ullah, M., Kolch, W., and Cho, K.-H.: 'Modelling and simulation of intra cellular dynamics: Choosing an appropriate framework', *IEEE Trans. NanoBioSci.*, 2004, **3**, (3), pp. 200–207
8 SBML, 2004. www.sbml.org
9 Olivier, B.G., and Snoep, J.L.: 'Web-based kinetic modelling using JWS Online', *Bioinformatics*, 2004, **3**, p. 318
10 The MathWorks Inc. Matlab 7 R14, 2004. www.mathworks.com
11 Huang, C-Y.F., and Ferrell, J.E. Jr.: 'Ultrasensitivity in the mitogen-activated protein kinase cascade', *Proc. Natl. Acad. Sci. USA*, 1996, **93**, pp. 10 078–10 083
12 Heinrich, R., and Schuster, S.: 'The regulation of cellular systems' (Chapmann and Hall, 1996)
13 Wolkenhauer, O., Ullah, M., Wellstead, P., and Cho, K.-H.: 'The dynamic systems approach to control and regulation of intracellular networks', *FEBS Lett.*, 2005, **579**, (29331), pp. 1846–1853
14 Wolkenhauer, O., Sreenath, S.N., Wellstead, P., Ullah, M., and Cho, K.-H.: 'A systems- and signal-oriented approach to intracellular dynamics', *Biochem. Soc. Trans.*, 2005, **33**, (3), pp. 507–515
15 Elf, J., Lötstedt, P., and Sjöberg, P.: 'Problems of high dimension in molecular biology', '17th GAMM-Seminar' (Leipzig, 2001), pp. 1–10
16 Gardiner, C.W.: 'Handbook of stochastic models' (Springer, 1985, 2nd Edn.)
17 Gillespie, D.T.: 'Exact stochastic simulation of coupled chemical reactions', *J. Phys. Chem.*, 1977, **81**, (25), pp. 2340–2361
18 McAdams, H.H., and Arkin, A.P.: 'Stochastic mechanisms in gene expression', *Proc. Natl. Acad. Sci. USA*, 1997, **94**, pp. 814–819

19 Rao, C.V., Wolf, D.M., and Arkin, A.P.: 'Control, exploitation and tolerance of intracellular noise', *Nature*, 2002, **420**, pp. 231–237
20 van Gend, C., and Kummer, U.: 'STODE – automatic stochastic simulation of systems described by differential equations', in Yi and Hucka (Eds.): 'Proceedings of the 2nd International Conference on Systems Biology', (Omnipress, Pasadena, 2001), pp. 326–333
21 Kierzek, A.M.: 'STOCKS STOchastic Kinetic Simulations of biochemical systems with Gillespie algorithm', *Bioinformatics*, 2002, **18**, pp. 470–481
22 Kaplan, D., and Glass, L.: 'Understanding nonlinear dynamics' (Springer Verlag, 1995)
23 Goldbeter, A., and Koshland, D.E. Jr.: 'An amplified sensitivity arising from covalent modification in biological systems', *Proc. Natl. Acad. Sci. USA*, 1981, **78**, (11), pp. 6840–6844
24 Kholodenko, B.N.: 'Negative feedback and ultrasensitivity can bring about oscillations in the mitogen-activated protein kinase cascade', *Eur. J. Biochem.*, 2000, **267**, pp. 1583–1588
25 Shuai, J.W.: 'Stochastic properties of Ca^{2+} release of inositol 1,4,5-triphosphate receptor clusters', *Biophys. J.*, 2002