Simulating biochemical systems with input in Matlab

Author: N.A.W. van Riel, TU/e
Date: June 22, 2017

Dynamic models of biochemical networks can be developed by writing down mass balance equations for each species, resulting in a system of coupled first order differential equations (state-space models).
Material is also available on GitHub: https://github.com/nvanriel/sbs .

Autonomous system
As an example, an implementation of a model of the following irreversible enzymatic reaction is shown

$$A + E \underset{k_{-1}}{\overset{k_{+1}}{\rightleftharpoons}} C \xrightarrow{k_{+2}} B + E \qquad (1)$$

where *A* is the substrate, *B* is the product, *E* is free enzyme and *C* is a complex of substrate-bounded enzyme. Parameter values: k_{+1} = 1000 $M^{-1}s^{-1}$, k_{-1} = 1 s^{-1}, k_{+2} = 0.1 s^{-1} and E_0 = 0.1 mM, with E_0 the total enzyme concentration (free *E* and bound in complex *C*). Initial conditions: [*A*(0)] = 1 mM, [*B*(0)] = 0 and [*C*(0)] = 0.

In chemistry such a system is called 'closed' (no in- or outflow of material). In system theory it is called an autonomous system as there is no external input which drives the system. The resulting dynamic behavior is entirely due to the 'energy' stored inside the system, i.e., the initial conditions. The system dynamics converge towards a steady-state, which is equal to the chemical equilibrium (according to thermodynamics).
In Biochemistry, Eq(1) is a very well-known reaction mechanism, forming the basis for so-called Michaelis-Menten kinetics. (See textbooks on Biochemistry for details.)

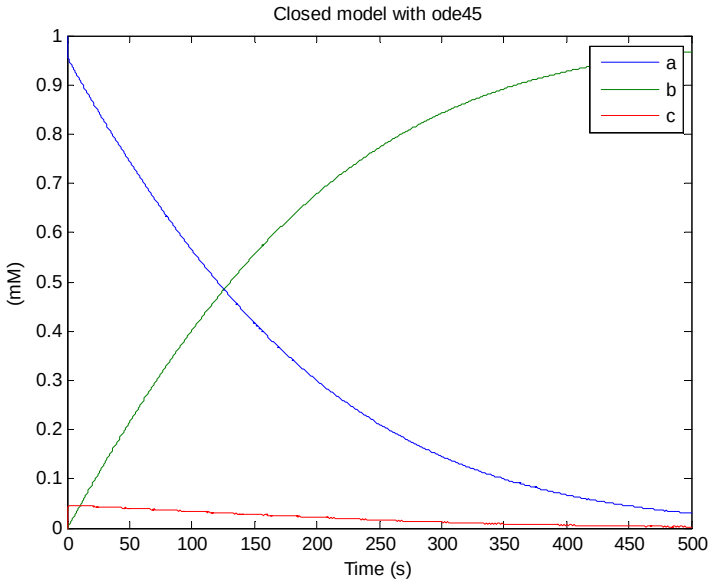
Simulation 1 Initial value problem with Runge-Kutta integration method
First we will implement and simulate the model in Matlab (The MathWorks) using a so-called Runge-Kutta integration method (ode45 in Matlab). See help ode45 and Matlab documentation for an explanation.

```
%MM_script1

% Initial Conditions:
x0 = [0.001 0 0];   %(M)
% Integrate ODEs:
tspan = [0 500];    %(s)
[t,x] = ode45(@MM_ode1,tspan,x0);   %Runge-Kutta
% Plot results:
figure; plot(t,x(:,1)*1e3,t,x(:,2)*1e3,t,x(:,3)*1e3);
```

This script calls the function @MM_ode1, which contains the differential equations:

```
function [f] = MM_ode1(t,x)

%State variables
a = x(1);   %substrate
b = x(2);   %product
c = x(3);   %enzyme complex

%Parameters
kp1 = 1000; %forward rate constant (M^{-1} sec^{-1})
km1 = 1.0;  %reverse rate constant (sec^{-1})
kp2 = 0.1;  %forward rate constant (sec^{-1})
E0  = 1e-4; %total enzyme concentration (M)

%ODEs
f(1) = (km1 + kp1*a)*c - kp1*E0*a;
f(2) = kp2*c;
f(3) = kp1*a*(E0 - c) - (km1 + kp2)*c;

f = f(:);   %Output needs to be a column vector
```

This function should be saved as MM_ode1.m before the script can be run (save in the same directory as the script). The result is shown in the figure below.



Be careful (consistent) with the units of all parameters and variables in the implementation!

Question 1
What are the units of the matrix X resulting from the simulation with ode45?

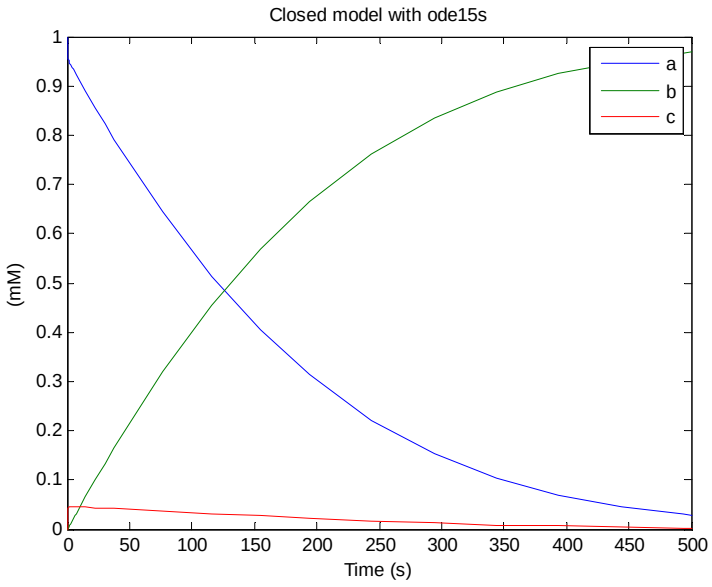
Simulation 2 Variable step solver
Secondly, exactly the same simulation is repeated, but using the ode15s method, which is a so-called variable step method.

Question 2

a) For which class of problems is ode15s suited according to the Matlab help information?
b) Explain in your own words what type of problems these are.
c) Does the irreversible reaction studied here belong to this class?

In the above script (MM_script1.m) only one line needs to be changed to replace ode45.
It is also interesting to compare the computational time required by both methods. You can use Matlab's stopwatch timer (tic, toc) to measure computational time.

```
tic
[t,x] = ode15s(@MM_ode1,tspan,x0);
toc
```



Question 3
Do you see any difference between the result with ode45 and ode15s? (Zoom in on the responses.)

Question 4
Which solvers is faster ode45 or ide15s?

Question 5 (OPTIONAL)
Try to implement the model using a stoichiometric matrix for the network and a vector for the rate equations.

System with input
Next, the reaction is linked to a metabolic pathway

$$\xrightarrow{v(t)} A + E \underset{k_{-1}}{\overset{k_{+1}}{\rightleftharpoons}} C \xrightarrow{k_{+2}} B + E \xrightarrow{k_{+3}} \qquad (2)$$

which supplies *A* with an input flux *v* and B disappears with a rate constant k_{+3} = 0.01 s^{-1}.

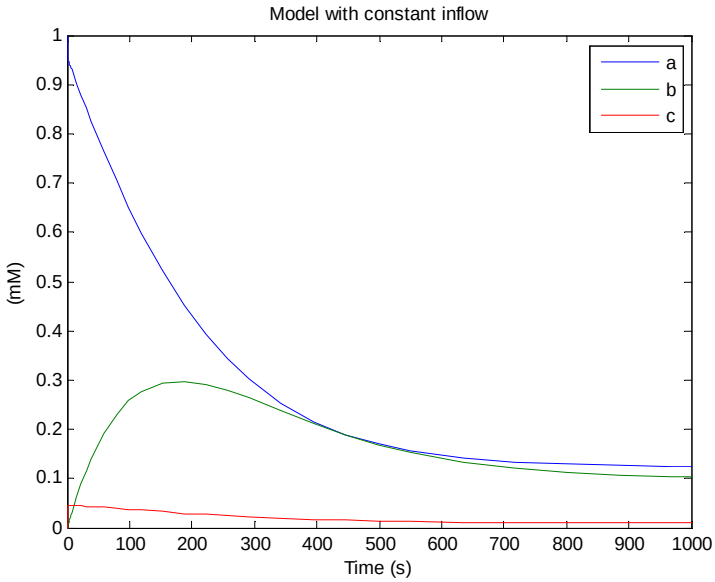
Chemically this is an open system, which is not in thermodynamic equilibrium.
Mathematically, a steady-state still exists if the input is constant.

Simulation 3 System with a constant input
First a simulation is performed with a constant input flux $v(t)$ = 1 μM·s^{-1}.

```
%MM_script2

% Initial Conditions:
x0 = [0.001 0 0];   %[a,b,c] (M)
% Integrate ODE:
tspan = [0 1000];    %(s)
[t,x] = ode15s(@MM_ode2,tspan,x0);
% Plot results:
figure; plot(t,x(:,1)*1e3,t,x(:,2)*1e3,t,x(:,3)*1e3);
```

Script MM_script2 is exactly the same as MM_script1, except that a different ODE file is called (@MM_ode2) and for the result shown below the simulation time was increased to 1000 s (tspan = [0, 1000]).

```
function [f] = MM_ode2(t,x)

%State variables
a = x(1);   %substrate
b = x(2);   %product
c = x(3);   %enzyme complex

%Parameters
kp1 = 1000; %forward rate constant (M^{-1} sec^{-1})
km1 = 1.0;  %reverse rate constant (sec^{-1})
kp2 = 0.1;  %forward rate constant (sec^{-1})
E0  = 1e-4; %total enzyme concentration (M)
kp3 = 0.01; %rate constant product outflux (sec^{-1})

%Input
u = 1e-6;  %influx (M/sec)

%ODE's
f(1) = u + (km1 + kp1*a)*c - kp1*E0*a;
f(2) = kp2*c - kp3*b;
f(3) = kp1*a*(E0 - c) - (km1 + kp2)*c;

f = f(:);   %Output needs to be a column vector
```



Question 6

a) Calculate the steady-state solution of this system analytically.

b) Does the numerical solution agree?

Simulation 4 System with pulsed input superimposed on a constant inflow

Next, the response to an input according to:

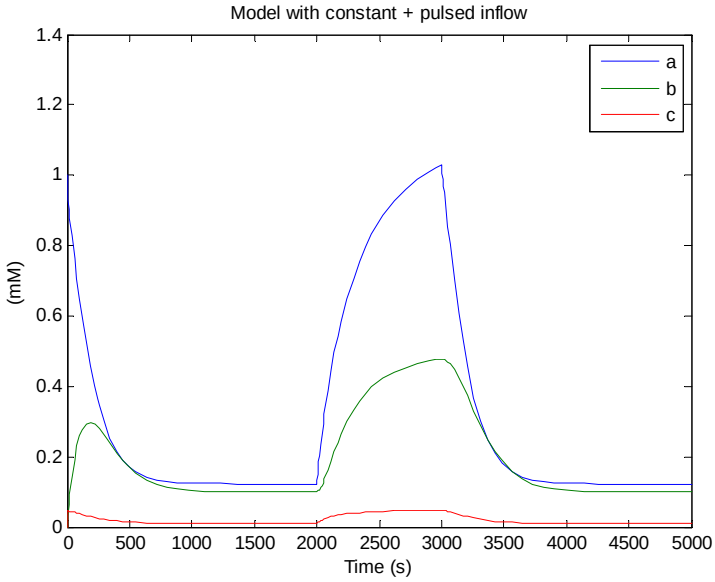
$$v(t) = \begin{cases} 1 & t < 2000 \wedge t > 3000 \\ 5 & 2000 \le t \le 3000 \end{cases} \text{ in } \mu M \cdot s^{-1} \qquad (3)$$

is simulated.

Again, the only changes to be made to the script is to call a different ODE file in which the above input has been implemented (@MM_ode3), and increase the simulation time span to include all interesting dynamics (tspan = [0, 5000]).

```
function [f] = MM_ode3(t,x)

%State variables
a = x(1);   %substrate
b = x(2);   %product
c = x(3);   %enzyme complex

%Parameters
kp1 = 1000; %forward rate constant (M^{-1} sec^{-1})
km1 = 1.0;  %reverse rate constant (sec^{-1})
kp2 = 0.1;  %forward rate constant (sec^{-1})
E0  = 1e-4; %total enzyme concentration (M)
kp3 = 0.01; %rate constant product outflux (sec^{-1})

%Input
if t<2000 | t>3000
    u = 1e-6;  %influx (M/sec)
else
    u = 5e-6;  %influx (M/sec)
```

```
end

%ODE's
f(1) = u + (km1 + kp1*a)*c - kp1*E0*a;
f(2) = kp2*c - kp3*b;
f(3) = kp1*a*(E0 - c) - (km1 + kp2)*c;

f = f(:);   %Output needs to be a column vector
```



The first part of the simulation is the same as in simulation 3. The system is simulated with a constant input until it reaches steady-state. Then the influx is 5-fold increased for 1000 sec, after which the input is reset to its reference value and the system goes back to steady-state.

Question 7
Is this final steady-state different from the initial steady-state (the steady-state from 1500-2000 sec)? Why?

Simulation 5 More flexibility: defining the input in a separate function
In the implementations above, parameter values and system input were hard-coded in the file containing the ODE's. Often it is useful to be able to modify those *outside* the ODE file, which is shown here for system Eq(2) with input Eq(3).

For the input a separate m-file, MM_pulse.m, is created:

```
function u = MM_pulse(t)
%MM_pulse Pulse inflow for model of irreversible enzyme reaction.
%
%   u = MM_pulse(t)
%

if t<2000 | t>3000
    u = 1e-6;  %influx (M/sec)
else
```

```
    u = 5e-6;  %influx (M/sec)
end
```

You can test and plot the function:

```
t = 0:5000;    %(s)
for k=1:length(t)
    u(k) = MM_pulse(t(k));
end
figure; plot(t,u)
```

The script:

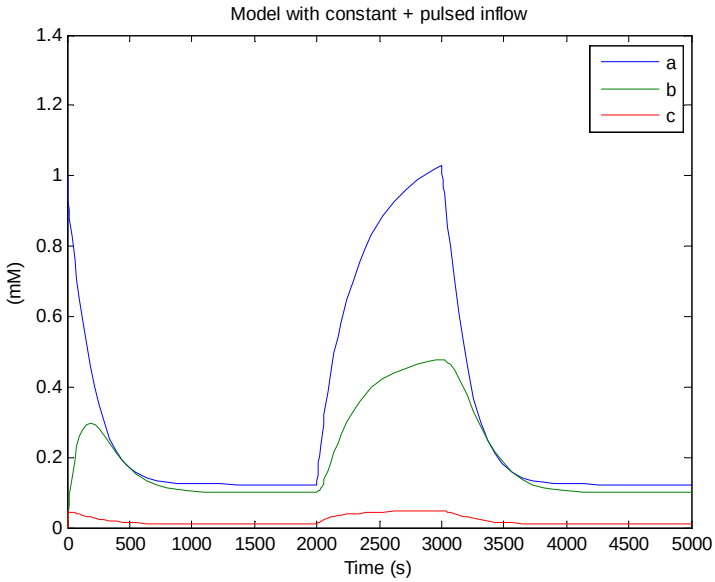
```
%MM_script4

% Parameter values:
kp1 = 1000; %forward rate constant (M^{-1} sec^{-1})
km1 = 1.0;  %reverse rate constant (sec^{-1})
kp2 = 0.1;  %forward rate constant (sec^{-1})
E0  = 1e-4; %total enzyme concentration (M)
kp3 = 0.01; %rate constant product outflux (sec^{-1})

par = [kp1, km1, kp2, E0, kp3];

%Input
inputfile = @MM_pulse;

% Initial Conditions:
x0 = [0.001 0 0];

% Integrate ODE:
tspan = [0 5000];    %(s)
odeoptions = [];    %use defaults
[t,x] = ode15s(@MM_ode4,tspan,x0,odeoptions, par,inputfile);

% Plot results:
figure; plot(t,x(:,1)*1e3,t,x(:,2)*1e3,t,x(:,3)*1e3);
```

For the Matlab ODE-solvers, the first 4 input arguments are reserved (see Matlab help for an explanation). Fifth and further arguments can be defined by the user and will be passed forward by Matlab to the ODE-function. Here, these are the parameter vector (par) and the name of the input function (inputfile). The ODE-function (MM_ode4.m) should be modified to accept these additional input arguments:

```
function [f] = MM_ode4(t,x,par,inputfile);

%State variables
a = x(1);   %substrate
b = x(2);   %product
c = x(3);   %enzyme complex

%Parameters
kp1 = par(1);   %forward rate constant (M^{-1} sec^{-1})
km1 = par(2);   %reverse rate constant (sec^{-1})
kp2 = par(3);   %forward rate constant (sec^{-1})
E0  = par(4);   %total enzyme concentration (M)
```

```
kp3 = par(5);   %rate constant product outflux (sec^{-1})

%Input
u = inputfile(t);

%ODE's
f(1) = u + (km1 + kp1*a)*c - kp1*E0*a;
f(2) = kp2*c - kp3*b;
f(3) = kp1*a*(E0 - c) - (km1 + kp2)*c;

f = f(:);   %Output needs to be a column vector
```

As expected / desired, the result is exactly the same as for simulation 4.



Question 8

a) Change above simulation such that the 1000 sec, 5 $\mu M \cdot s^{-1}$ pulse is added at t = 300 sec., and explain the result.

Assume the pathway of Eq(2) is part of a cellular process which in the basal, inactive state carries a flux of 1 $\mu M \cdot s^{-1}$. In an experiment to reveal the dynamic behavior the process is perturbed with the 1000 sec, 5 $\mu M \cdot s^{-1}$ pulse.

b) Assuming that the model of the pathway is correct, which of the above simulations (pulse at 2000 sec or pulse at 300 sec) will most likely be in agreement with the experimental data?

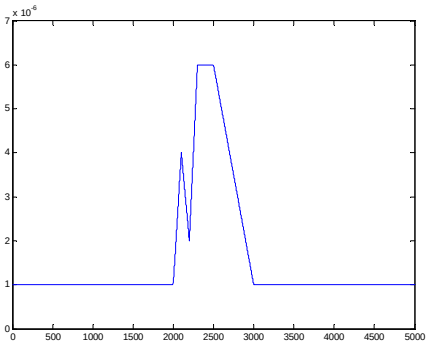
Simulation 6 Measured data as input

The flexibility obtained by not hard-coding any values in the ODE-function can be used to easily change parameter values, or, for example, use a measured signal / profile as input, which is the final example shown here. In this case the sampled data is stored in a *lookup table* and *interpolation* is used to generate an input sample for any desired time point t. (Remember, a variable-step integration method is used, so the integration time steps are determined by the solver in Matlab and are a priori not known.)

Input file

```
function u = MM_data(t)

%first column: Time (s), second column Influx (M/sec)
inputdata = [0     1e-6;...
             2000  1e-6;...
             2100  4e-6;...
             2200  2e-6;...
             2300  6e-6;...
             2500  6e-6;...
             3000  1e-6;...
             5000  1e-6];

u=interp1(inputdata(:,1),inputdata(:,2),t); %linear interpolation
                %see help interp1
```

Plotting the input data gives

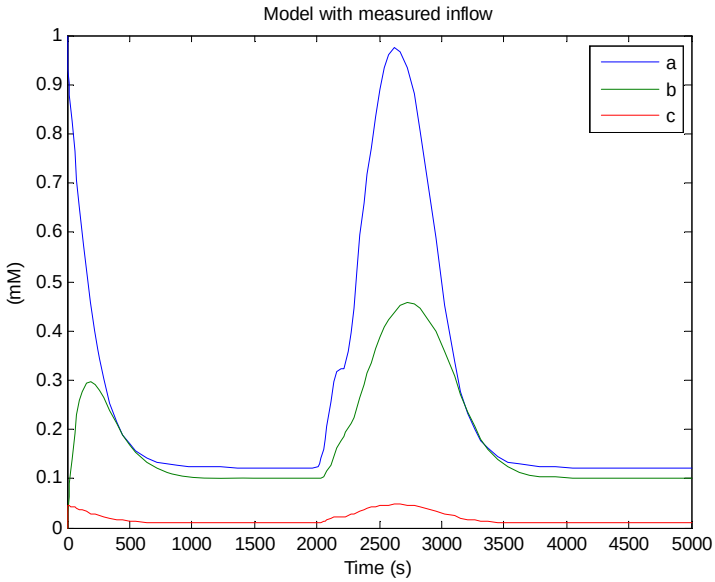


The first part is again the basal influx to allow the model to reach its reference steady-state. At t = 2000 sec the measurements are introduced.

In the previous script (MM_script4.m) only the name of the input file needs to be changed:

```
%Input
inputfile = @MM_data;
```

and the same MM_ode4.m can be used.



Simulation 7 Measured data as input

Three other syntaxes (implementations) that can be used to pass-forward the input data and perform the interpolation.

I) Using a time and input data vector

In the main script:

```
% Time (s),
tvec = [0 2000 2100 2200 2300 2500 3000 5000]';
Second column Influx (M/sec)
uvec = [1e-6 1e-6 4e-6 2e-6 6e-6 6e-6 1e-6 1e-6]';
```

In the ode file:

```
function [f] = MM_ode5(t,x,par,tvec,uvec);

%Input
u=interp1(tvec,uvec,t);

%State variables
a = x(1);   %substrate
b = x(2);   %product
c = x(3);   %enzyme complex
…etc.
```

II) Using a matrix

In the main script:

```
%first column: Time (s), second column Influx (M/sec)
tu = [0     1e-6;...
      2000  1e-6;...
      2100  4e-6;...
      2200  2e-6;...
      2300  6e-6;...
      2500  6e-6;...
      3000  1e-6;...
      5000  1e-6];
```

In the ode file:

```
function [f] = MM_ode5(t,x,par,tu);

%Input
u=interp1(tu(:,1),tu(:,2),t);

%State variables
a = x(1);   %substrate
b = x(2);   %product
c = x(3);   %enzyme complex
…etc.
```

III) Using a struct

In the main script:

In the ode file:

```
function [f] = MM_ode5(t,x,par,tu);

%Input
u=interp1(tu(:,1),tu(:,2),t);

%State variables
a = x(1);   %substrate
b = x(2);   %product
c = x(3);   %enzyme complex
…etc.
```