MATLAB Ordinary Differential Equation (ODE) solver for a simple example

1. Introduction

Differential equations are a convenient way to express mathematically a *change* of a *dependent* variable (e.g. concentration of species A) with respect to an *independent* variable (e.g. time). When writing a differential equation, one operate on the *rates of change* of quantities rather than the quantities themselves. This simplifies greatly the derivation of important relationships used by engineers around the world. However, switching from the rates of change to the values themselves (i.e. solving a differential equation) can be troublesome, especially when one deals with coupled (multiple) differential equations which have to be solved simultaneously. In many cases, an analytical solution does not exist and engineers have to rely on numerical (approximate) solutions.

A special and very abundant group of differential equations is called ordinary differential equations (ODEs). In these equations, dependent variables are functions of **one** independent variable only, for example:

$$-\frac{dc_A}{dt} = kc_A = f(c_A)$$

where the rate of change of the concentration of species A over time is directly proportional to the concentration of species A. One may also think of a system of ODEs, where two or more dependent variables exist, each of them expressed as a separate ODE, for example:

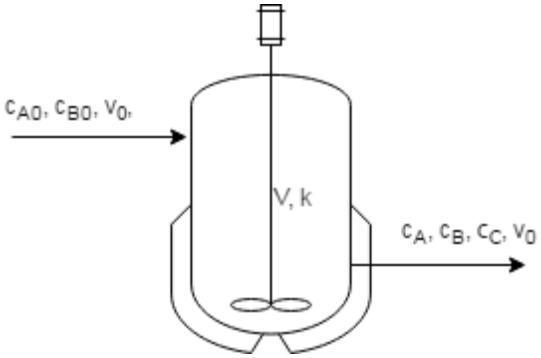
$$\begin{cases} -\frac{dc_A}{dt} = f_1(c_A, c_B) \\ -\frac{dc_B}{dt} = f_2(c_A, c_B) \end{cases}$$

Systems of equations similar to these shown above are very common in CRE problems, therefore it is advisable to learn how to solve them in order to predict the evolution of variables in time or space (e.g. length of the reactor). In this tutorial we will solve a simple ODE and compare the result with analytical solution. In another tutorial (see **Ordinary Differential Equation (ODE) solver for Example 12-1** in MATLAB tutorials on the CRE website) we tackle a system of ODEs where more than one dependent variable changes with time.

2. Developing a simple model with ODE to solve

We will now develop a simple model of an isothermal CSTR (see Figure to the right) operated at non-steady state where a second-order, irreversible elementary reaction takes place:

$$A + B \rightarrow C$$



The rate constant is $k = 0.003 \frac{m^3}{mol \cdot s}$,

the initial concentration of A, and B are:

$c_{A0} = 10\, mol/m^3$, $c_{B0} = 15\, mol/m^3$.

The volumetric flow of reactants is $v = v_0 = 0.5\frac{m^3}{s}$ and the active volume of the reactor is $V = 1000m^3$.

The CSTR initially (t=0) contains reactants A and B at concentrations specified before and the volumetric flowrate is constant. Concentration of product C is equal to zero at t=0.

The task is to simulate the system and obtain concentration-time trajectories for all species: A, B, C. The exercise is to be solved in terms of concentration of the limiting reactant.

We begin with the **mole balance** over species A which is the limiting reactant in this case:

$$[IN] - [OUT] + [GENERATION] = [ACCUMULATION]$$

$$F_{A0} - F_A + (r_A)V = \frac{dN_A}{dt}$$

Furthermore, the **rate law** reads as:

$$-r_A = kc_Ac_B$$

From the reaction stoichiometry, we know that: $c_B = c_{B0} - (c_{A0} - c_A)$, thus we can write the rate law having only c_A as an *independent* variable:

$$-r_A = kc_A[c_{B0} - (c_{A0} - c_A)]$$

For the liquid phase, isothermal conditions and constant volumetric flowrate we can write $F_i = c_i \cdot v_0$ and $N_i = c_i \cdot V$. Combining yields:

$$v_0(c_{A0} - cA) - kc_A[c_{B0} - (c_{A0} - c_A)]V = V\frac{dc_A}{dt}$$

Letting $\tau = V/v_0$ and rearranging gives:

$$\frac{dc_A}{dt} = \frac{c_{A0} - c_A}{\tau} - kc_A(c_{B0} - c_{A0} + c_A)$$

We can clean up the equation even further:

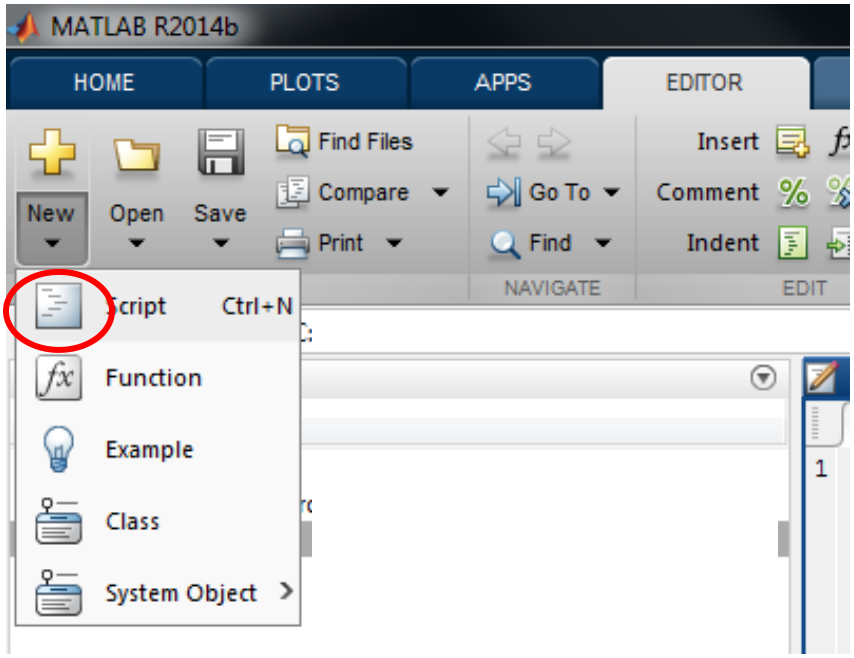
$$\frac{dc_A}{dt} = -kc_A^2 + \left(kc_{A0} - kc_{B0} - \frac{1}{\tau}\right)c_A + \frac{c_{A0}}{\tau}$$

We ended up with quite a long ODE where c_A is the dependent variable and t is the independent variable. Analytical solution to this ODE exists, but we will first solve it with the use of MATLAB numerical method.

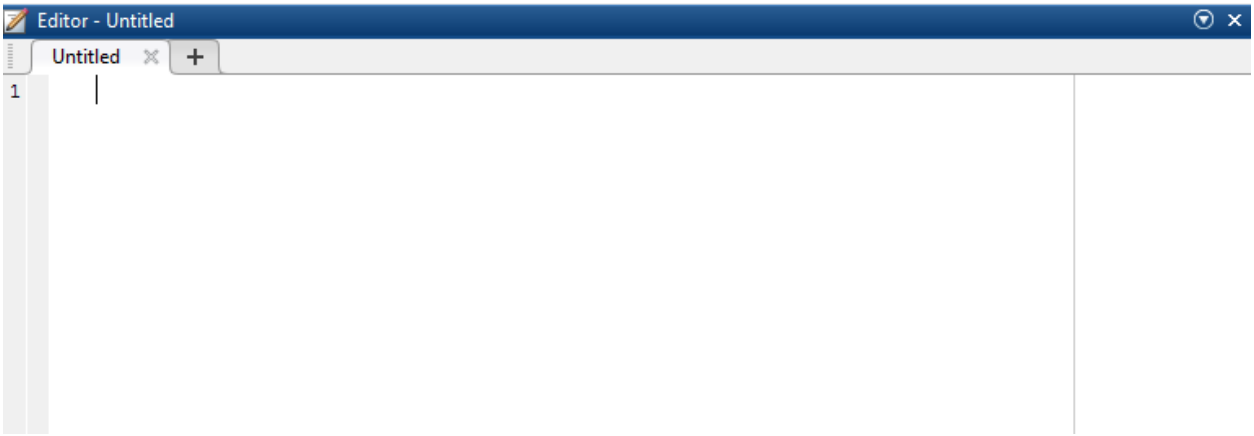
3. Writing MATLAB code

We will treat this point in steps for convenience.

STEP 1. Open MATLAB console and click “New” and then “Script” under “Editor” bookmark:



The following window will open:



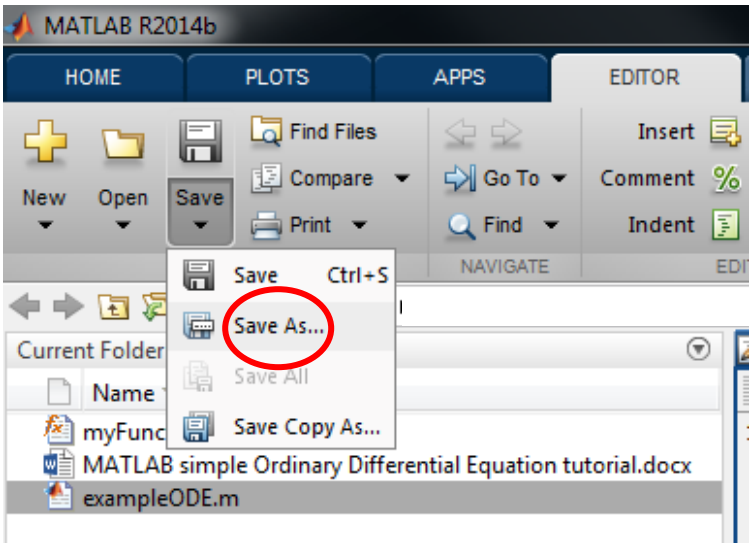
STEP 2: Create a new function by typing the following command:

```
1  function f = myFunction(t, cA)
2
3
4
5  end
```

Here **f** is the variable to which the function returns its value(s). myFunction is an arbitrary name of the function which we specify. The terms in brackets are all the dependent and independent variables which

our function deals with. In our case the variables are: time (t) and concentration of species A (cA). The variables must be listed in the order: (independent, dependent1, dependent2, …).

STEP 3. Save the script as "myFunction.m" by selecting "Save" and then "Save As…". Note that the name of the file is automatically typed in by the software since all MATLAB functions names must be in accordance with the filename of the function.



STEP 3. List the constant parameters and all explicit equations which exist in our model. The code must be placed between **function** and **end** keywords. Semicolons suppress displaying the values of the variables on the screen.

```
1   function f = myFunction(t, cA)
2
3   % parameters
4   cA0 = 10; %mol/m3
5   cB0 = 15; %mol/m3
6   V = 1000; %m^3
7   v0 = 0.5; %m^3/s
8   k = 0.003; %m^3/(mol*s)
9   %explicit equations
10  tau = V/v0;
11
12  end
```

STEP 4. Write the ODE, first setting a separate variable for the sign of the derivative $\frac{dc_A}{dt}$, for example "dcAdt". The choice of the name is arbitrary, following the rules for naming the variables in MATLAB. Write the right hand side of the ODE. Next, for the sake of brevity, assign the "derivative variable" to the variable which we created when building our function, f. The function returns the values of the derivative at given t and cA and saves it in the variable f. This method proves handy when we have to work with many ODEs (a system of ODEs). An example is provided in **Ordinary Differential Equation (ODE) solver for Example 12-1** in MATLAB tutorials section on the CRE website.

```
function f = myFunction(t, cA)

% parameters
cA0 = 10; %mol/m3
cB0 = 15; %mol/m3
V = 1000; %m^3
v0 = 0.5; %m^3/s
k = 0.003; %m^3/(mol*s)
%explicit equations
tau = V/v0;

%differential equation(s)
dcAdt = (cA0 - cA)/tau - k*cA*(cB0-cA0+cA);
f = dcAdt;
end
```

Save the file by pressing Ctrl+S on Windows or Command+S on Mac.

STEP 5. Create a new script (STEP 1) and save it (STEP3) as "exampleODE". Type **clear all** and **close all** commands to prepare a new workspace each time we run the code. Next, type the time span for numerical integration (let's assume times from 0 to 400 s) and enter the initial condition(s) (initial concentration of A in the reactor at time t=0 is equal to 10 mol/m^3). If there are more than one dependent variable, the initial conditions have to be listed in an array (use square brackets), e.g. initial = [IC1, IC2, …];. An example of this is shown in the second MATLAB tutorial on ODEs.

```
clear all; close all
%specification of time span for numerical integration
timeStart = 0; %s
timeEnd = 400; %s
timespan = [timeStart, timeEnd];

%specification of initial condition(s):
initCA = 10; %mol/m^3 at timeStart
```

STEP 6. Initialize the solver and specify the solution method by typing:

```
%ODE solver initialization:
[timeOut, cA] = ode45(@myFunction, timespan, initCA);
```

timeOut is a variable where the time steps will be stored and cA is the variable where the concentration of species A will be saved. timeOut and cA will correspond with each other row-by-row, i.e. the first row of timeOut (set to zero by us) will correspond with the initial concentration of A, c_{A0}.

You can choose between many different solution algorithms (see the second MATAB tutorial for the table of possible options). The most versatile is **ode45.** After typing the name of the solver, we open parenthesis and specify the name of the function with our ODE with the "@" at the beginning to create a function handle. Next, we specify our timespan and initial conditions, separating them with commas. The order of listing them is important.

STEP 7. Prepare data plotting

The solution (a pair of vectors timeOut and cA) can be plotted with the use of the following commands (type them after the solver line):

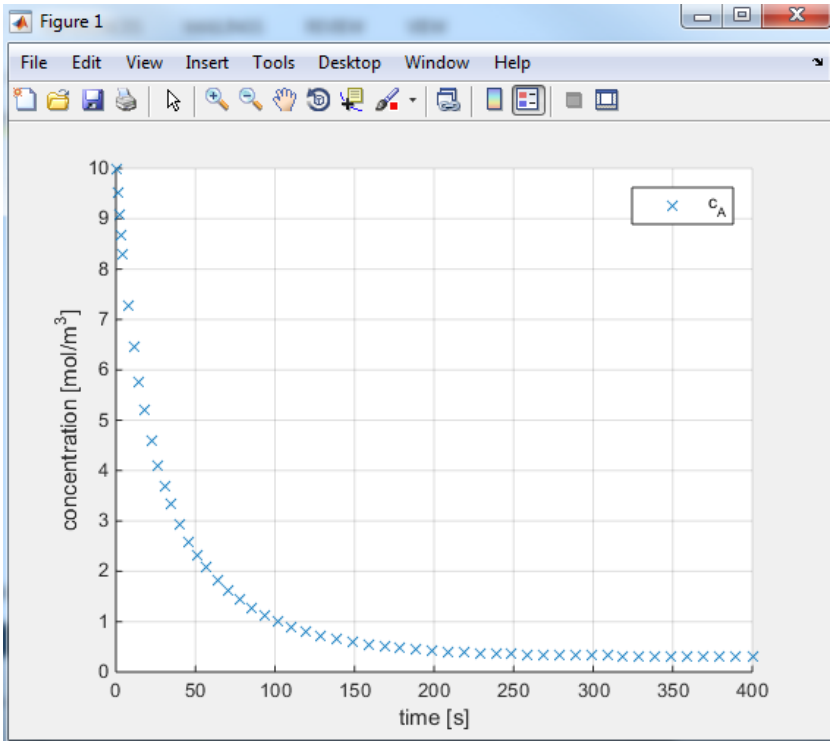
```
%plotting data


plot(timeOut, cA, 'x')


%Adding grid, labels and legend
grid on
xlabel('time [s]')
ylabel('concentration [mol/m^3]')
legend('c_A')
```

STEP 7. Run the script by clicking “Run” button under “Editor” bookmark.

The following plot appears:



We analyze the solution: at time 0 the concentration of Species A is equal to 10 mol/m^3 and gradually decays until the steady state is achieved (around 250 seconds after the onset of the reaction). You can play with parameters (especially k, V and v_0) to see which of them has the largest impact on the shape of the curve. What is the conversion of A (complete/incomplete) at the steady state for different set of parameters?

4. Analytical solution and comparison with numerical approximation

We have to be conscious that the solution achieved in part 3 is only an approximation of the real (analytical) solution. Sometimes analytical solution does not exist whatsoever owing to the complexity of the problem. In such cases, numerical solution is the only one we can afford.

In our case, however, the equation

$$\frac{dc_A}{dt} = -kc_A^2 + \left(kc_{A0} - kc_{B0} - \frac{1}{\tau}\right)c_A + \frac{c_{A0}}{\tau}$$

Is not that complicated and the analytical solution is readily available. By letting

$$A = c_{A0}/\tau,\ B = -1/\tau - kc_{B0} + kc_{A0}\ ,\ C = k \text{ and } D = -4 \cdot A \cdot C - B^2$$

the solution for the same initial conditions as before reads as:

$$t(C_A) = E - \frac{2\tan^{-1}\left(\frac{2C \cdot c_A - B}{\sqrt{D}}\right)}{\sqrt{D}}$$

where E is the integration constant:

$$E = \frac{2\tan^{-1}\left(\frac{2C \cdot c_{A0} - B}{\sqrt{D}}\right)}{\sqrt{D}}$$

As we can see, even for a relatively simple kinetics, the analytical solution may look arduous. This example is not meant to frighten students, but to give an insight on the difference between numerical and analytical methods. We save the convoluted math problems for the related course during studies.

Concentrations of species B and C can be computed from the stoichiometry having species A as the key component:

$$c_B = c_{B0} - (c_{A0} - c_A)$$

$$c_C = c_{A0} - c_A$$

When we plot all the concentrations against time and compare with the numerical solution, we obtain the figure presented below. As we can see, for this simple problem the approximate numerical solution introduces very little error. One has to have always in mind that all numerical methods treat the problem

with certain doze of inaccuracy and the magnitude of error depends on the complexity of the problem, the shape of the curve (“stiffness of the ODE”) and the computational resources.