A First Course on Kinetics and Reaction Engineering
Supplemental Unit S5. Solving Initial Value Differential Equations

Defining the Problem

This supplemental unit describes how to solve a set of initial value ordinary differential equations (ODEs) numerically. The equations must be of the form shown in equation (1), or using matrix notation as in equation (2). Notice, in particular that there are no partial derivatives in the equations, only ordinary first derivatives. All of the derivatives are taken with respect to the same independent variable, represented here as t. It is essential that the boundary conditions are specified at the same value of the independent variable, denoted here as t_0. The latter requirement is what distinguishes these as *initial value* ODEs, and in this situation, the boundary conditions are usually referred to as initial conditions. Often the value of t_0 is zero, but this is not required.

$$\begin{aligned}
\frac{dz_1}{dt} &= f_1\left(t, z_1, z_2, \cdots, z_n\right); \quad z_1\left(t_0\right) = z_1^0 \\
\frac{dz_2}{dt} &= f_2\left(t, z_1, z_2, \cdots, z_n\right); \quad z_2\left(t_0\right) = z_2^0 \\
&\vdots \\
\frac{dz_n}{dt} &= f_n\left(t, z_1, z_2, \cdots, z_n\right); \quad z_n\left(t_0\right) = z_n^0
\end{aligned} \tag{1}$$

$$\frac{d\underline{z}}{dt} = \underline{f}\left(t, \underline{z}\right) \tag{2}$$

Information and Data Required for Numerical Solution

To numerically solve a system of initial value ODEs like those in equation (1) or (2), you will need to use mathematics software that provides an appropriate ODE solver. Irrespective of the particular brand of software you use, you typically will need to provide three things as input to that software:

- the initial values of the independent and dependent variables: t_0 and $\underline{z}(t_0)$
- the final value of either the independent variable or one of the dependent variables
- code that evaluates each of the derivatives given a value for the independent variable and values for each of the dependent variables

At the minimum, the software will return the final values of the independent and dependent variables that were not specified. In many cases the software will return sets of values of the independent and dependent variables spanning the range from the initial values to the final values. The latter information is quite useful if you want to make a plot, for example of one or more dependent variables as a function of the independent variable.

Overview of the Numerical Method

Recall that when an ODE is solved analytically, the solution is an expression for the functional dependence of the dependent variable upon the independent variable. The analytical solution to a first order ODE always contains an unknown constant. For example, consider equation (3) where k is a known constant. The solution is given in equation (4), and equation (4) contains the additional constant C which did not appear in the original ODE. The value of this constant is determined using the initial condition. When a set of ODEs is solved numerically, the solution usually takes the form of a table of values of the independent variable and the corresponding values of the dependent variables. Thus, a numerical solution of equation (3) would consist of pairs of values of t and the corresponding value of z.

$$\frac{dz}{dt} = -kz \tag{3}$$

$$z = C\exp(-kt) \tag{4}$$

ODEs like those considered here are solved numerically by starting at the initial value of the independent t_0 because at this point the values of the dependent variables are known. A very small incremental change of t from t_0 to $t_0 + h$ is then considered, where h is a small incremental change in the independent variable sometimes referred to as the step size. I will illustrate here using a single ODE, equation (5), but extending the approach to multiple ODEs is straightforward. Within the interval from t_0 to $t_0 + h$, the unknown z is approximated as some convenient mathematical function of t. Here I will illustrate by approximating z as a linear function of t, but other convenient mathematical forms can also be used.

Approximating z as a linear function of t within the interval from t_0 to $t_0 + h$ gives equation (5). The objective is to solve equation (6) to find the value of z at $t = t_0 + h$. The value of z at $t = t_0$ is known and the size of the incremental change, h, can be chosen (as long as it is small). However, a value for the slope, m, is also needed. If z truly was a linear function of t, the slope would be equal to the derivative of z with respect to t, equation (6), and that derivative would be a constant. In reality, however, the derivative of z with respect to t is not constant across the interval h, but it varies according to equation (7), which is the ODE being solved. In order to approximate z as a linear function of t in the interval of t from t_0 to $t_0 + h$ a single value has to be chosen for the slope, m.

$$z(t_0 + h) = z(t_0) + mh \tag{5}$$

$$m = \frac{dz}{dt} \tag{6}$$

$$\frac{dz}{dt} = f(t, z) \tag{7}$$

The easiest, and perhaps most obvious, choice for a value of m is to evaluate the derivative at $t = t_0$ using equation (5). Since the value of z is known at $t = t_0$, the value of the derivative can be computed

from equation (7) and substituted into equation (5) for the slope, m. Equation (5) can then be solved directly for the value of z at $t = t_0 + h$. That approach is sometimes used; it is known as the *explicit* Euler method. The problem with the explicit Euler method is that it isn't very accurate unless the increment h is very, very small. Making the increment very, very small can lead to other problems. For that reason, it is preferable to approximate the slope m in a different way.

An alternative choice for a value of m is to evaluate the derivative at $t = t_0 + h$ as shown in equation (8). Equation (8) cannot be solved directly for the value of the slope, m, however because the value of z at $t = t_0 + h$ is not known. Instead, it is necessary to solve equations (5) and (8) simultaneously. This approach is known as the *implicit* Euler method. It is more accurate than the explicit Euler method and eliminates many of its problems. There are situations, however, where the implicit Euler method is not accurate. These will be discussed later. The key point, for present purposes, is that the implicit Euler process allows solution of the ODEs numerically. Once the value of z at $t = t_0 + h$ has been found, the resulting point becomes the new initial condition. Thus, the process can be repeated over and over, producing pairs of values of t and z over any desired range. That is, the process can be repeated over and over until either *t* or *z* reaches some desired final value.

$$m = \left.\frac{dz}{dt}\right|_{t=t_0+h} = f\left(z\left(t_0+h\right), t_0+h\right) \tag{8}$$

There are many different variations of the approach described here. They differ with respect to the mathematical form of the function that is used to approximate z in the interval interval of t from t_0 to $t_0 + h$ and in the way that the slope (or its equivalent when non-linear approximating functions are used) is estimated. A very popular method is the Runge-Kutta method. It gives an accuracy that is comparable to a fourth order polynomial approximating function by using a weighted average of several different approximations of the slope. Those who are interested in other numerical approaches are encouraged to take a course on the topic or consult a good textbook. The purpose here is just to provide a basic understanding of how these methods work.

It was noted earlier that sometimes the implicit Euler method is not sufficiently accurate. The same can be true of the Runge-Kutta method. The problem arises when solving sets of ODEs. In essence it occurs when one of the functions changes very abruptly over a very small range of the independent variable and the changes in that function significantly affect the changes in other functions over a much larger range of the independent variable. Equations like this are called *stiff* ODEs, and they require special treatment of the interval size, h. A full discussion goes beyond the intended scope of this supplemental unit. When solving sets of ODEs, one should pay attention to whether any of the dependent variables change very abruptly as the independent variable changes, and if they do, it is probably advisable to check the solution obtained by repeating the solution using a different numerical method that is specifically tailored to *stiff* ODEs.

MATLAB Template Files

MATLAB includes a built in function named `ode45` that implements the Runge-Kutta version of the approach described above, and it also includes a built in function named `ode15s` that is designed to handle stiff sets of ODEs. In both cases, the built in functions must be provided with the name of a subroutine (written by the user) that takes t and a vector $\underline{z}$ and uses them to compute and return the functions, $\underline{f}$, in equation (1) as a column vector. The built in ODE solvers also must be given initial and final values of the independent variable and initial values of the dependent variables. There are many options that can be set when using these ODE solvers, and you should read the MATLAB documentation to learn what is available.

One option that is commonly needed is to set a stopping criterion other than reaching the final value of the independent variable provided as an argument to the ODE solver. It is quite common to need to solve a set of initial value ODEs under conditions where you know the initial values of the independent and dependent variables, but instead of knowing the final value of the independent variable, you know the final value of one of the dependent variables and you want to calculate the corresponding value of the independent variable. For example, consider equation (3). Given the initial values of z and t, one might know the final value of t and be asked to calculate the corresponding final value of z, or, one might know the final value of z and be asked to calculate the value of t.

When using the MATLAB ODE solvers, you always need to pass initial and final values for the independent variable, even if the final value is one of the quantities you are trying to solve for. What one does in this case is to provide a very large number for the final value of the independent variable, and then set an option to use a second termination criterion. The second termination criterion is set using a built in function named `odeset`, and when you do so, you must provide the name of a function (again written by the user) that takes t and $\underline{z}$ as arguments. I'm not going to go into the details here, but one quantity that this function must return is a scalar, and when that scalar reaches zero, the ODE solver terminates. Thus, if you want the ODE solver to stop when z_3 reaches some known final value, z_{3f}, you would set this scalar equal to $z_3 - z_{3f}$. A critical aspect of this is that z_3 must reach z_{3f} before t reaches the final value passed to the ODE solver. That is why you give a very large number as the final value of the independent variable. When the ODE solver returns the results, you always need to check to see whether t reached the final value passed to the ODE solver. If it did, then you need to repeat the solution using a higher number for the final value of the independent variable until the solver stops without reaching that value.

Using the MATLAB ODE solvers is not particularly difficult, but it does require care to set all the arguments and function calls properly. This can be distracting if you are required to do it every time you solve a set of initial value ODEs. Therefore, two MATLAB template files named SolvIVDifII.m and SolvIVDifD.m are provided with this supplemental unit. The names are intended as mnemonic for **Solv**e **I**nitial **V**alue **Dif**ferential equations. SolvIVDifI.m is used when you know the final value of the **i**ndependent

(hence the final I in the name) variable while SolvIVDifD.m is used when you know the final value of one of the **d**ependent variables (and hence the final D in the name). These template files handle most of the programming details so that you can focus on the kinetics and reaction engineering aspects of the problems you solve, and not on computer programming.

The template files do require some modifications each time they are used. Set of step-by-step instructions for making these modifications for each of the template files are provided with this supplemental unit. Example S5.1 illustrates how to modify and use the SolvIVDifI.m to solve a problem where the final value of the independent variable is known, and Example S5.2 illustrates how to modify and use SolvIVDifD.m when the final value of one of the dependent variables is known.