A First Course on Kinetics and Reaction Engineering

Example S5.2

Problem Purpose

This example illustrates the use of the MATLAB template file SolvIVDifD.m to solve a set of initial value ordinary differential equations where the final value of one of the dependent variable is known.

Problem Statement

In equations (1) and (2), A is a constant with a value of 0.1597. Calculate the value of t at which z_1 equals half of its initial value and the ratio of z_2 to z_1 at this t.

$$\frac{dz_1}{dt} = -A\frac{z_1}{z_1+z_2} \quad ; \qquad z_1(0) = 0.103 \tag{1}$$

$$\frac{dz_2}{dt} = 0.5A\frac{z_1}{z_1+z_2} \quad ; \qquad z_2(0) = 0.0 \tag{2}$$

Problem Analysis

Equations (1) and (2) are ordinary differential equations (ODEs), and both of the boundary conditions are applied at the same boundary, namely $t = 0$. This means that they are initial value ODEs. The problem gives us the final value of z_1, and z_1 is one of the dependent variables. That means that the MATLAB template file SolvIVDifD.m can be used to solve the problem. Here I will follow the step-by-step instructions for using SolvIVDifD.m that are provided with this supplemental unit. Note also that the equations are written in the form shown in equation (3).

$$\begin{aligned}
\frac{dz_1}{dt} &= f_1(t,z_1,z_2,\cdots,z_n); \qquad z_1(t_0) = z_1^0 \\
\frac{dz_2}{dt} &= f_2(t,z_1,z_2,\cdots,z_n); \qquad z_2(t_0) = z_2^0 \\
&\vdots \\
\frac{dz_n}{dt} &= f_n(t,z_1,z_2,\cdots,z_n); \qquad z_n(t_0) = z_n^0
\end{aligned} \tag{3}$$

You may have noticed that the equations and the constant in this problem are exactly the same as in Example S5.1. The only difference is found in what we are asked to calculate. In Example S5.1 we were given a final value of t and asked to find the corresponding value of $\underline{z}$. Here, we are given the final value of z_1 and asked to calculate the corresponding values of t and z_2. The primary purpose of this example is to illustrate the difference in the required modifications when you know the final value of one of the dependent variables instead of knowing the final value of the independent variable. Therefore, large parts of this solution are word-for-word the same as Example S5.1. If that bothers you, sorry!

Problem Solution

A copy of SolvIVDifD.m was saved in the working directory as S5_Example_2.m. The function declaration statement was changed so that the function name was the same as the filename without the ".m" filetype. At the same time, the long initial comment was changed to a shorter comment indicating the purpose of the modified template file. The template file must be modified in six places each time it is used to solve a new problem. The locations where the modifications are required are indicated by comments that begin "% EDIT HERE." The first required modification involves declaring variables for each constant that appears in the problem and assigning values to those variables in consistent units. Here there is only one constant, A, and it does not have units. All of these changes can be seen in Listing 1.

```
% Modified version of the MATLAB template file SolvIVDifD.m used to solve
% Example 2 of Supplemental Unit S5 of "A First Course on Kinetics and
% Reaction Engineering."
%
function [t_f,z] = S5_Example_2
    % Known quantities and constants (in consistent units)
    A = 0.1597;
```

Listing 1. Modified version of SolvIVDifD.m after renaming and making the first required modification.

The second required modification involves entering the code to evaluate the derivatives, that is, the right hand sides of equations (1) and (2) when they are written in the form of equation (3). This modification appears within the internal function named `odeqns`. The resulting, modified version of `odeqns` is shown in Listing 2.

```
    % Function that evaluates the ODEs
    function dzdt = odeqns(t,z)
        dzdt = [
            -A*z(1)/(z(1) + z(2))
            0.5*A*z(1)/(z(1) + z(2))
        ];
    end % of internal function odeqns
```

Listing 2. Internal function odeqns after making the second required modification.

The third required modification involves entering the initial value of the independent variable, t, and the initial values of the dependent variables, $\underline{z}$. Equations (1) and (2) show that for this problem, the initial value of the independent variable is $t = 0$, and the corresponding initial values of the dependent variables are $z_1(0) = 0.103$ and $z_2(0) = 0.0$. These values must be assigned to the variables `t0`, `z0(1)` and `z0(2)`, respectively, in the template file at the location indicated by the "%EDIT HERE" comment for the third required modification. Listing 3 shows the section of the template file where this information was entered.

```
% Initial values
t0 = 0.0;
z0 = [
    0.103
    0.0
];
```

Listing 3. Initial conditions added as the third required modification of the template file.

The way MATLAB solves the ODEs is to incrementally increase t, calculate the corresponding values of $\underline{z}$, check to see whether a stopping criterion has been met and then either stop (if a criterion is met) or repeat. A final value for t always has to be specified, and it is always one of the stopping criteria. When we know the final value of one of the dependent variables, as we do in this problem (and every problem where we use SolvIVDifD.m), we don't ever want to reach this stopping criterion based on t. That is, in this particular problem, we want z_1 to reach half its initial value ***before*** t reaches the value we specify as `tf`. Therefore in the fourth required modification, we must set `tf` equal to a large value; here I used 1000, as you can see in Listing 4. I have no way of knowing how large to set `tf`, and it is possible that t will reach 1000 before z_1 reaches half its initial value. Therefore, when I execute the modified template file, I'll need to check the final value of t and make sure it is less than 1000. If it is equal to 1000, that means that the solution stopped because the `tf` stopping criterion had been satisfied and not the criterion on z_1. In that case, I'll need to re-edit the template file, setting `tf` to equal a larger number and re-execute the file.

```
tf = 1000.0;
options = odeset('Events',@stop);
[t, zz, te, ze, ie] = ode45(@odeqns,[t0, tf],z0,options);

% Function that provides the integration stopping criterion
function [stop_when, isterminal, direction] = stop(t,z)
    isterminal = 1;
    direction = 0;

    % The variable stop_when should equal zero when the desired
    % stopping criterion is reached
    stop_when = z(1)-z0(1)/2;
end % of internal function stop
```

Listing 4. Contents of the second block comment after it has been "uncommented" and all required modifications to it completed.

To complete the fifth required modification, the stopping criterion for z_1 is set. This modification is located within an internal function named `stop`. Within that internal function there is a variable named `stop_when`. That variable, `stop_when`, should evaluate to zero when z_1 equals its final value. If stop_when is defined as being equal to the current value of z_1 minus it's desired final value (in this case one half of $z_1(0)$), then `stop_when` will equal zero when z_1 equals one half of $z_1(0)$. Thus, the template file is modified so that `stop_when` is set accordingly, as shown in Listing 4.

The sixth and final modification involves adding code to calculate any additional quantities using the results from solving the ODEs. In this problem we are asked to calculate the ratio of z_2 to z_1. That is easily done, a can be seen in Listing 5 which shows the last few lines of the modified template file where this modification takes place. Notice that I did not put a semicolon at the end of the line that calculates the ratio; if I did, the ratio would be calculated, but it would not be listed in the output when the file is executed, and I'd never know its value. Any code you enter here should not have a semicolon at the end of the line if you want to see the result.

```
    % calculate the ratio of z2 to z1
    ratio = z(2)/z(1)

end % of S5_Example_2.m
```

Listing 5. Bottom section of the template file after all required modifications have been completed.

That completes the required modifications, so once the file is saved to make the changes permanent, it can be executed by typing the first line shown in Listing 6. Doing so produces the remainder of Listing 6. Looking at the output, you can see that the final value of t was 0.3848. Thus, t never reached 1000 (the value I specified as `tf` when modifying the code), so the results should be valid (i. e. the solution must have terminated because the other stopping criterion I specified was met). Checking we see that the final value of z_1 is 0.0515 which is, indeed, one half of its initial value, 0.103. Therefore, z_1 reaches one half of its initial value at t = 0.3848 at which point z_2 is equal to 0.0258 and the ratio of z_2 to z_1 is 0.5.

```
>> [t_f,z] = S5_Example_2

ratio =

    0.5000

t_f =

    0.3848

z =

    0.0515    0.0258
```

Listing 6. Output generated upon execution of the modified template file, S5_Example_2.m