A First Course on Kinetics and Reaction Engineering

Example S4.3

Problem Purpose

The purpose of this example is to illustrate how to use the MATLAB template file FitNumDifSR.m to perform a regression analysis for single response data with a model that consists of a set of coupled initial-value ordinary differential equations that must be solved numerically.

Problem Statement

Suppose that 12 experiments were performed wherein the values of three variables, x_1, x_2 and x_3, were set and then the value of variable y was measured. The results of the experiments are shown in Table 1. There are 12 data points; for each data point there are three set variables, x_1, x_2 and x_3, and one response variable, $\hat{y}$. Differential equations (1) through (3) are the experimental model being analyzed. The dependent variables in the model equations are u_1, u_2 and u_3, and the independent variable is v. The response variable can be calculated from the solution of the model equations using equation (4). The constant, A, that appears in the model equations has a value of 10.

Table 1. Experimental Data.

x_1	x_2	x_3	$\hat{y}$
2	0.2	0.2	0.208
2	0.6	0.6	0.491
2	0.2	1	0.911
2	1	0.2	0.091
1	0.2	0.2	0.112
1	0.6	0.6	0.333
1	0.2	1	0.694
1	1	0.2	0.060
0.67	0.2	0.2	0.083
0.67	0.6	0.6	0.274
0.67	0.2	1	0.545
0.67	1	0.2	0.042

$$\frac{du_1}{dv} = -A\frac{\theta_1 u_1 u_2^2}{1+\theta_2 u_1+\theta_3 u_2}; \qquad u_1(0) = x_2 \tag{1}$$

$$\frac{du_2}{dv} = -A\frac{\theta_1 u_1 u_2^2}{1+\theta_2 u_1 + \theta_3 u_2}; \qquad u_2(0) = x_3 \tag{2}$$

$$\frac{du_3}{dv} = 2A\frac{\theta_1 u_1 u_2^2}{1+\theta_2 u_1 + \theta_3 u_2}; \qquad u_3(0) = 0 \tag{3}$$

$$y = \frac{x_2 - u_1(v_f)}{x_2}; \qquad v_f = x_1 \tag{4}$$

Problem Analysis

This problem is nearly the same as the problem presented in Example S4.2; the difference is that here the model equations are coupled initial value ordinary differential equations while in Example S4.2 they were non-linear equations. Consequently, the proper template file to use is FitNumDifSR.m. Step-b-step instructions for using FitNumDifSR.m are provided with this supplemental unit, and they will be illustrated in the solution of this example problem.

Problem Solution

A copy FitNumDifSR.m was saved as S4_Example_3.m, and because the filename had been changed, the function statement also had to be changed to match, as required by MATLAB. At the same time, the introductory comment was changed to indicate the purpose of the modified file. These changes are shown in Listing 1.

```
% Modified version of the MATLAB template file FitNumDifSR.m that was
% modified for the solution of Example 3 of Supplemental Unit S4 of "A
% First Course on Kinetics and Reaction Engineering."
%
function S4_Example_3(p_guess)
```

Listing 1. Modified introductory comment and changed function name.

There are six required modifications that must be made to the template file each time it is used. They are indicated in the code by a comment that begins “% EDIT HERE”. The first involves entering all constants given in the problem statement or needed for fitting the model to the data. In this problem, there is one constant, A, so its value is entered at this point. The second required modification involves entering the set variables as a matrix (it must be named x) with a separate column for each different response variable and one row per experiment. Here there are two set variables, x_1 and x_2, so the matrix has two columns. There are eleven rows in the matrix, one for each data point. The third required modification involves entering the experimentally measured values of the response variable as a column vector named y_hat. Again, there is one row per data point in this column vector. Listing 2 shows the code after these modifications were completed.

```
% Known quantities and constants (in consistent units)
A = 10.0;

% Set variables
x = [2  0.2   0.2
2 0.6   0.6
2 0.2   1
2 1     0.2
1 0.2   0.2
1 0.6   0.6
1 0.2   1
1 1     0.2
0.67    0.2   0.2
0.67    0.6   0.6
0.67    0.2   1
0.67    1     0.2];

% Experimental response variables
y_hat = [0.208
0.491
0.911
0.091
0.112
0.333
0.694
0.060
0.083
0.274
0.545
0.042];
```

Listing 2. Entry of constants, set variables and response variables.

The fourth required modification is located in the internal function odeqns. It involves evaluating the derivatives in equations (1) through (3), given values of v and $\underline{u}$. In this problem, the evaluation of the derivatives also requires values for the parameters, θ_1, θ_2 and θ_3. These are not passed to odeqns as an argument, but they are available in the global variable p. Hence, θ_1 is stored in p(1), θ_2 in p(2) and θ_3 in p(3). The values of the set variables aren't needed for this problem, but if they had been, they similarly would have been available in the column vector x_set. All the quantities needed for the evaluation of the derivatives are available, so it is easy to write the code to do so as seen in Listing 3.

```
    % Function that evaluates the ODEs
  function dudv = odeqns(v,u)
      % Current parameter values are available in column vector p
      % Current set variable values are available in column vector x_set
      term = A*p(1)*u(1)*u(2)^2/(1.0 + p(2)*u(1) + p(3)*u(2));
      dudv = [
          -term
          -term
          2*term
      ];
  end % of internal function odeqns
```

Listing 3. S4_Example_3.m after performing the fourth required modification.

The last two required modifications take place within the internal function, `nlmodel`. They occur within a loop through each of the experimental data points, i. The first is to provide the initial values of the independent variable, v and the dependent variables, $\underline{u}$, and the final value of the independent variable for data point i. Equations (1) through (3) indicate that the initial value of the independent variable, v, is 0 and the initial values of the dependent variables are $u_1(0) = x_2$, $u_2(0) = x_3$ and $u_3(0) = 0$. Looking at equation (4), we see that we need the value of u_1 at $v_f = x_1$, so the final value of the independent variable is x_1. The final required modification to the template file appears a few lines later after the model equations have been solved, but still within the loop through the data points. The modification involves using the results of solving the model equations, stored in the column vector `z`, to calculate the response using equation (4). Listing 4 shows the internal function `nlmodel` after these modifications have been completed.

```
    function y = nlmodel(p_current,x)
        % Declare y
        y = zeros(n_data,1);
        % Make the current parameters available to the model equations
        p = p_current;

        for i = 1:n_data
            % Make the set variables available to the model equations
            for j = 1:n_set
                x_set(j) = x(i,j);
            end

            % Initial and final values
            v0 = 0;
            u0 = [
                x_set(2)
                x_set(3)
                0.0
            ];
            vf = x_set(1);

            % Solve the set of ordinary differential model equations
            [v,uu] = ode45(@odeqns,[v0 vf],u0);
            last_value = length(v);
            u = uu(last_value,:);

            % calculate the response, y, by evaluating
            % f(u(1),...,u(n),x_set(1),...,x_set(n_set))
            y(i) = (x_set(2)-u(1))/x_set(2);
        end
    end % of internal function nlmodel
```

Listing 4. Internal function nlmodel after completing the final two required modifications.

That completes the required modification of S4_Example_3.m, and it can be saved in the current MATLAB working directory or in a directory that is part of the MATLAB search path. When S4_Example_3 is executed, it must be passed a column vector that contains guesses for each of the three parameters in the model. This column vector was named `p_guess` and was set up from the MATLAB command line; as Listing 5 indicates, I simply guessed a value of 1.0 for each of the three parameters. After that, the file was executed by typing S4_Example_3(p_guess), as also shown in Listing 5, which also shows the output that is produced.

```
>> p_guess = [1
1
1];
>> S4_Example_3(p_guess)

r_squared =
   9.9926e-01

Best Values for the Parameters:

pf =
   8.8542e-01
   2.7199e+00
   5.4315e+00

95% Confidence Intervals for the Parameters:

pf_u =
   3.6195e-01
   1.8676e+00
   2.6105e+00
```

Listing 5. Creation of the input column vector, p_guess, execution of S4_Example_3, and the resulting output.

Looking at the output, it can be seen that the correlation coefficient is very close to 1.0. The magnitude of the scatter of the data from the line in the parity plot (Figure 1) is very small, and there are no trends in the deviation in any of the residuals plots (Figures 2 through 4), so it can be concluded that the model does a very good job of fitting the experimental data. The best values of the parameters are θ_1 = 0.89 ± 0.36, θ_2 = 2.72 ± 1.87 and θ_3 = 5.43 ± 2.61 (95% confidence interval based upon 12 data). If these results are used as an initial guess to S4_Example_3, the same result is obtained indicating that this is a converged solution.

Suppose, however, that the fit was OK, but not excellent. In that case, the parameter values that were found by the process might correspond to a local minimum of the objective function and not to the global minimum. It would then be advisable to repeat the entire fitting process, but with a starting guess that was very different from the one used here (each of the three parameters equal to 1.0). It is possible that doing so would lead to a converged solution where the fit was considerably better.

Alternatively, suppose that this was a kinetics problem and the parameters corresponded to rate coefficients. Further suppose that even though the fit was very good, the values of the parameters didn't make sense as rate coefficients for the problem being solved. In that case, even though the fit is quite good, it would again be advisable to repeat the entire fitting process with a very different initial guess. In this case you'd be looking for a fit that was as good as the present fit, but where the resulting parameter values did make sense physically.

When you reach the point where the fit is acceptably good and the parameters make sense on physical grounds, you can probably stop looking for better sets of parameters. In contrast, if you never reach the point where the fit is acceptably good and the parameters make sense on physical grounds, you might want to consider the possibility that the model you are using simply isn't appropriate.

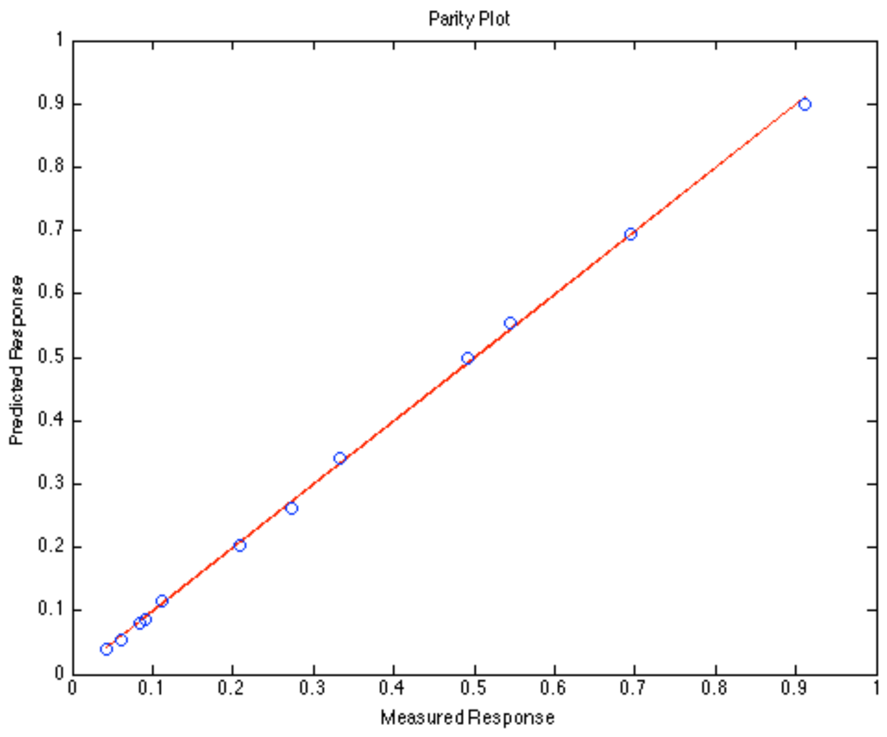

Figure 1. A parity plot for the fit of the model in equations (1) through (4) to the data in Table 1.

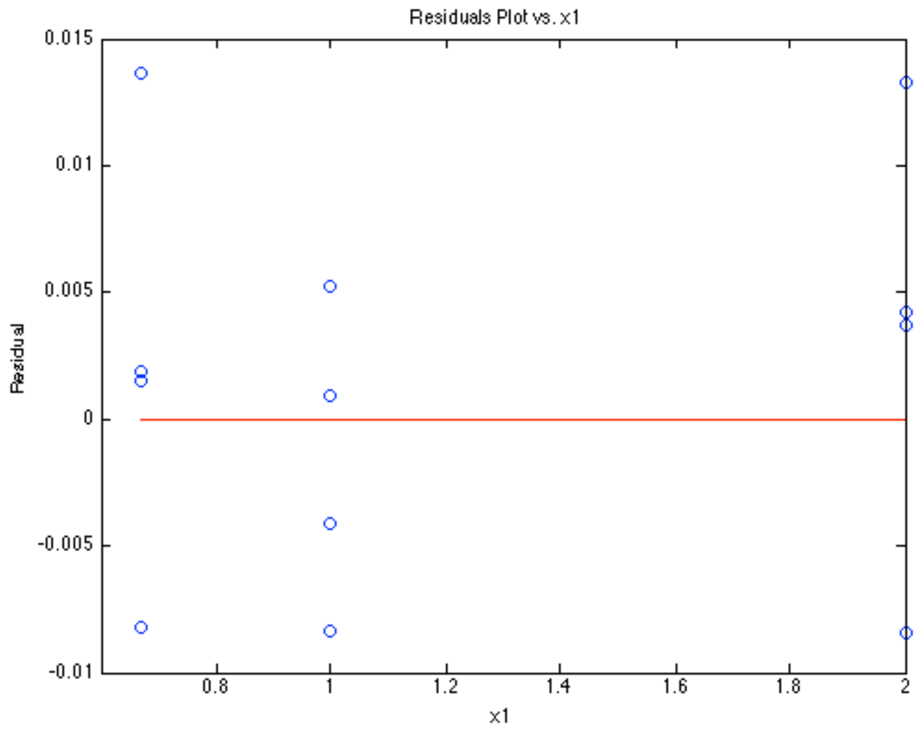

Figure 2. Residuals plot against x_1.

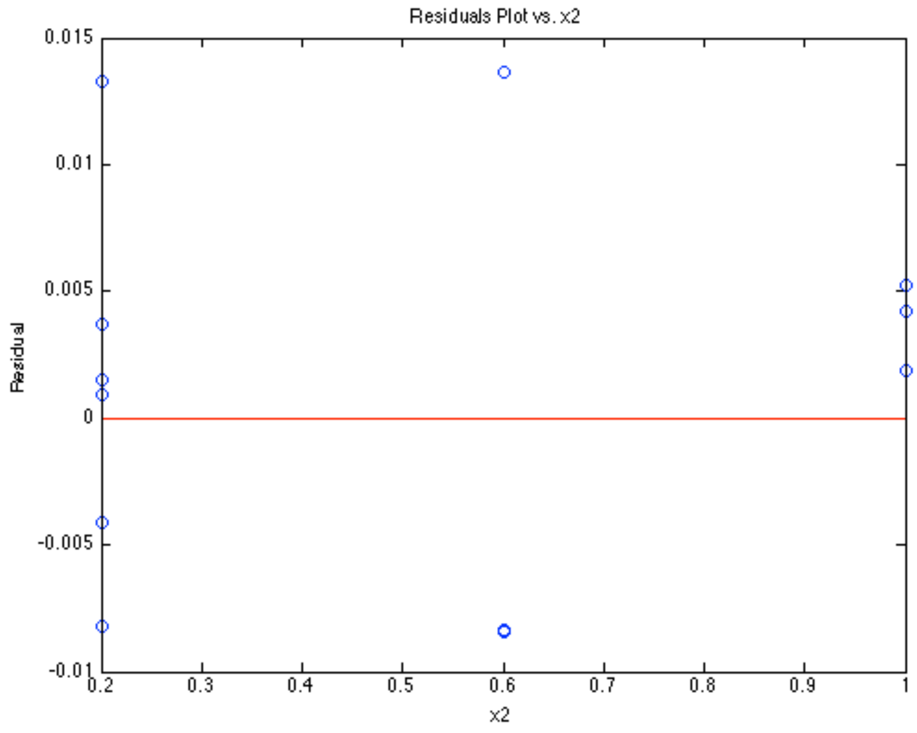

Figure 3. Residuals plot against x_2.

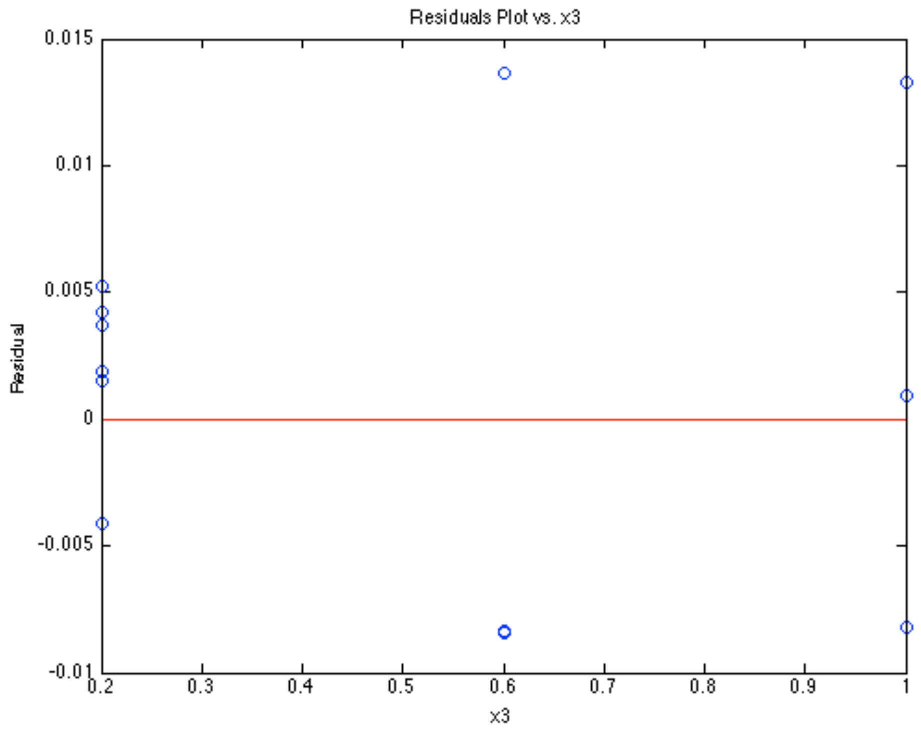

Figure 4. Residuals plot against x_3.