A First Course on Kinetics and Reaction Engineering

Example S4.2

Problem Purpose

The purpose of this example is to illustrate how to use the MATLAB template file FitNumAlgSR.m to perform a regression analysis for single response data with a model that consists of a set of non-linear algebraic 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. Additionally suppose that equations (1) through (3) constitute a model of the experiments with unknowns, u_1, u_2 and u_3, and that the response variable can be calculated from the solution of the model equations using equation (4). In these equations, A represents an experimental constant that has a value of 10.

Table 1. Data for Example 1.

x_1	x_2	x_3	$\hat{y}$
5	0.2	0.2	0.200
5	0.6	0.6	0.422
5	0.2	1	0.234
5	1	0.2	0.128
10	0.2	0.2	0.119
10	0.6	0.6	0.312
10	0.2	1	0.191
10	1	0.2	0.092
15	0.2	0.2	0.090
15	0.6	0.6	0.239
15	0.2	1	0.170
15	1	0.2	0.073

$$x_1x_2 - A\frac{\theta_1 u_1 u_2^2}{1+\theta_2 u_1 + \theta_3 u_2} - x_1 u_1 = 0 \tag{1}$$

$$x_1x_3 - A\frac{\theta_1 u_1 u_2^2}{1+\theta_2 u_1 + \theta_3 u_2} - x_1 u_2 = 0 \tag{2}$$

$$u_3 - 2\left(x_2 - u_1\right) = 0 \tag{3}$$

$$y = \frac{u_3}{u_1 + u_2 + u_3} \tag{4}$$

Problem Analysis

The model for the experiment is a set of three non-linear equations and the data have a single response, so the MATLAB template file, FitNumAlgSR.m can be used to fit the model to the data. The solution presented here follows the step-by-step instructions for using FitNumAlgSR.m that accompany this supplemental unit.

Problem Solution

A copy FitNumAlgSR.m was saved as S4_Example_2.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 FitNumAlgSR.m that was
% modified for the solution of Example 2 of Supplemental Unit S4 of "A
% First Course on Kinetics and Reaction Engineering."
%
function S4_Example_2(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 corresponding to the eleven data points. 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
    A = 10.0;

    % Enter the set variables
    x = [5  0.2   0.2
    5 0.6   0.6
    5 0.2   1
    5 1     0.2
    10      0.2   0.2
    10      0.6   0.6
    10      0.2   1
    10      1     0.2
    15      0.2   0.2
    15      0.6   0.6
    15      0.2   1
    15      1     0.2];

    % Enter the experimentally measured responses
    y_hat = [0.200
    0.422
    0.234
    0.128
    0.119
    0.312
    0.191
    0.092
    0.090
    0.239
    0.170
    0.073];
```

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

The fourth required modification appears within the internal function, `nlaeqns`. It involves evaluating the three functions that constitute the model, namely equations (1) through (3), and returning the results as a column vector, `g`. You can see in Listing 3 that the only argument received by `nlaeqns` is a column vector named u containing the values of u_1, u_2 and u_3. Looking at the model equations (1) through (3), you can see that they also contain the model parameters, θ_1, θ_2 and θ_3, and the set variables, x_1, x_2 and x_3. As the comment within the internal function indicates, these quantities are available in the global column vector variables `p`, and `x_set`. That is, θ_1 is stored in `p(1)`, θ_2 in `p(2)`, θ_3 in `p(3)`, x_1 in `x_set(1)`, x_2 in `x_set(2)`, and x_3 in `x_set(3)`. With this knowledge, the required modification is straightforward, as shown in Listing 3.

```
      function g = nlaeqns(u)
        % Current parameter values are available in column vector p
        % Current set variable values are available in column vector x_set
        term2 = A*p(1)*u(1)*u(2)^2/(1.0 + p(2)*u(1) + p(3)*u(2));
            g = [
                  x_set(1)*x_set(2) - term2 - x_set(1)*u(1);
                  x_set(1)*x_set(3) - term2 - x_set(1)*u(2);
                  u(3) - 2*(x_set(2) - u(1));
            ];
    end % of internal function nlaeqns
```

Listing 3. Modification for the evaluation of the non-linear equation set.

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. First, as indicated in the comments, guesses for the model unknowns, u_1, u_2 and u_3, for data point i need to be entered. Often, if you know the physics of the problem, you can make reasonable guesses based upon that knowledge. Alternatively, it is sometimes possible to generate guesses for the unknowns in the model equations using the experimentally measured response variables. Here, however, we don't have any other problem details to guide us, so I arbitrarily guessed that the three unknowns would each have a value of 0.5. Just a few lines after that modification, still within the loop through the data points, the final required modification must be made. This modification is located just after the model equations have been solved for the values of $\underline{u}$, and as the comment indicates, the modification involves calculating the value of the response variable using those results. In this problem, the response variable is calculated using equation (4), so this modification simply involves adding the code to calculate `y(i)` using equation (4). These final two required modifications are shown in Listing 4.

That completes the required modification of S4_Example_2.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_2 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_2(p_guess), as also shown in Listing 5.

Executing the file produces a long, long list of comments like the one shown in Listing 6. Each time `fsolve` finds a solution to the model equations, it prints a comment like this. The printing of all these comments can be suppressed as explained in a comment within the template file, but it's not a bad idea to let them print out at first, until you are sure the code is functioning properly. That way, if the equation solver failed for any reason, you would know from the message.

```
    % Function that calculates the responses using the model solution
    function y = nlmodel(p_current,x)
        % Make the current parameters available to the model equations
        p = p_current;
        y = zeros(n_data,1);

        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

            % Guesses
            % The measured response is available here as y_hat(i)
            u_guess = [
                0.5
                0.5
                0.5
            ];

            % Solve the set of nonlinear modle equations
            u = fsolve(@nlaeqns, u_guess);

            % Calculate the response
            y(i) = u(3)/(u(1) + u(2) + u(3));
        end
    end % of internal function nlmodel
```

Listing 4. Modifications within the internal function nlmodel.

```
>> p_guess = [1
1
1];
>> S4_Example_2(p_guess)
```

Listing 5. Creation of the input column vector, p_guess and execution of S4_Example_2.

```
Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the default value of the function tolerance, and
the problem appears regular as measured by the gradient.

<stopping criteria details>
```

Listing 6. Message that appears each time fsolve solves the model equations.

Eventually, after repeating this "equation solved" message many, many times, the desired output appeared in the MATLAB command window, as shown in Listing 7. In addition to the text output, the plots shown here as Figures 1 through 4 were generated and displayed. The fitted parameters displayed as the column vector `pf` were copied to the column vector `p_guess` and the modified template file was executed again with this new guess. The resulting output was essentially identical to that shown in Listing 10, indicating that the minimization had converged. (The only differences were in the last significant digit of two of the parameter uncertainty values).

```
      ⋮
Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the default value of the function tolerance, and
the problem appears regular as measured by the gradient.

<stopping criteria details>

r_squared =
  9.9734e-01

Best Values for the Parameters:

pf =
  1.0804e+01
  2.2078e+00
  5.5261e+00

95% Confidence Intervals for the Parameters:

pf_u =
  3.3373e+00
  1.1961e+00
  2.2772e+00
```

Listing 7. Results of executing the modified template file (after all the fsolve messages).

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 deviations 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 model parameters are θ_1 = 10.8 ± 3.3, θ_2 = 2.2 ± 1.2 and θ_3 = 5.5 ± 2.3 (95% confidence intervals based upon 12 data).

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.

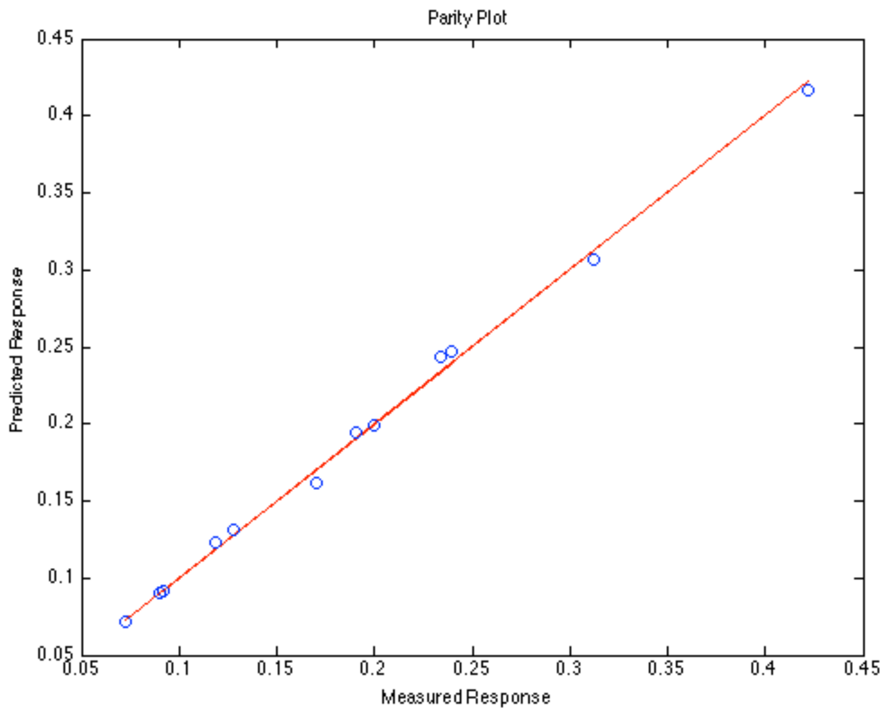

Figure 1. Parity plot showing the predicted responses versus the measured responses.

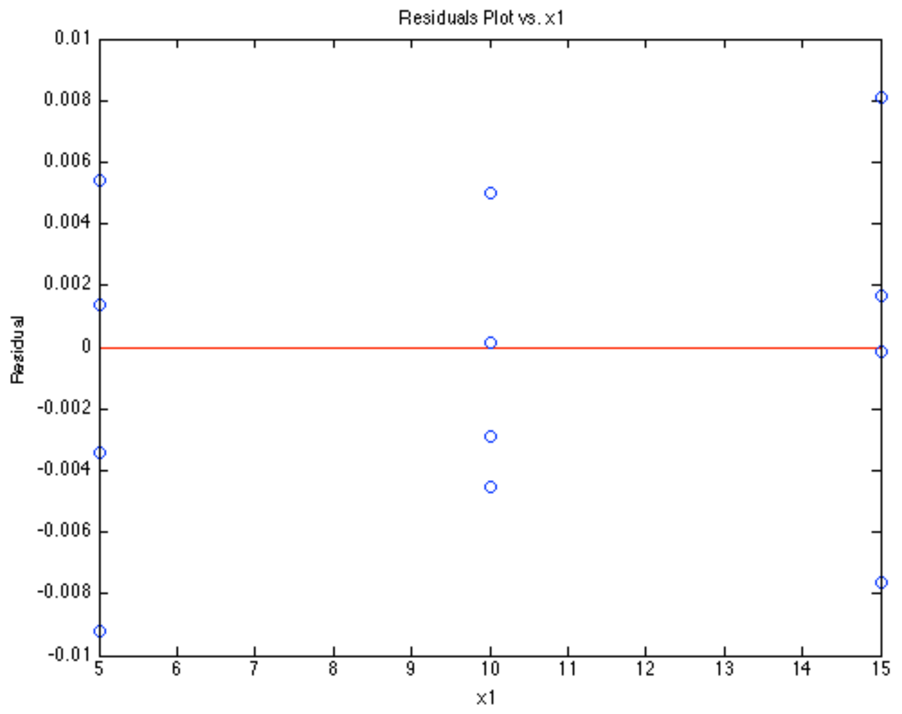

Figure 2. Residuals plot versus set variable x_1.

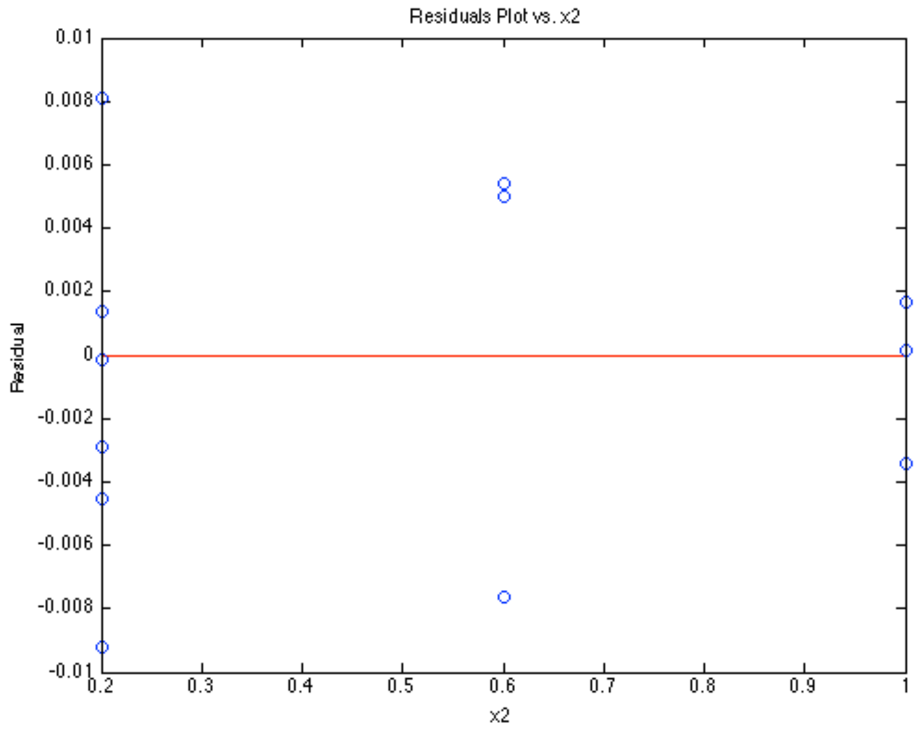

Figure 3. Residuals plot versus set variable x_2.

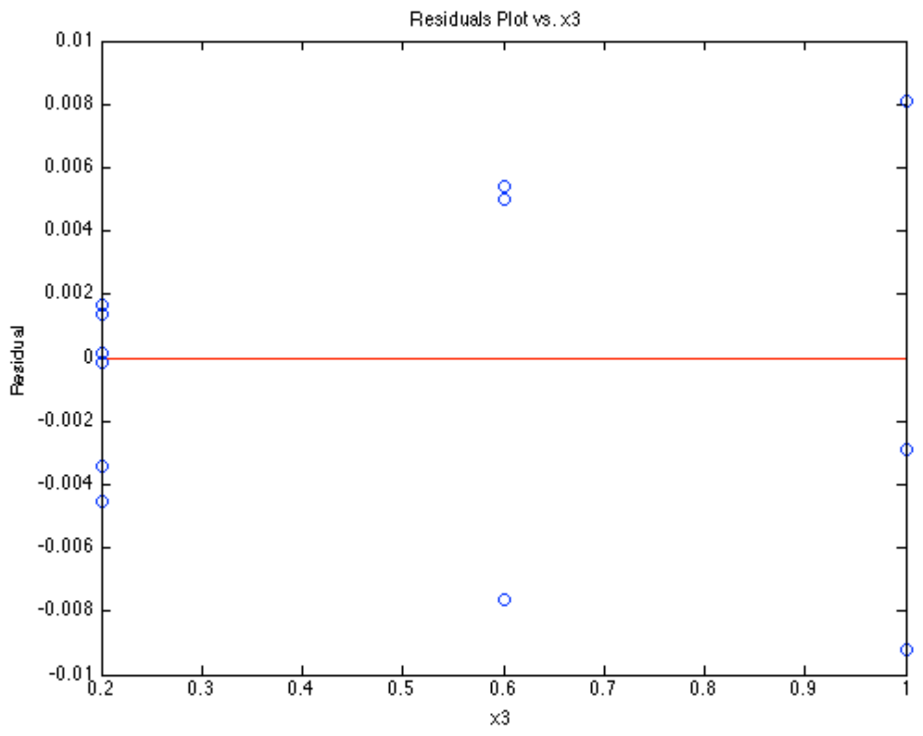

Figure 4. Residuals plot versus set variable x_3.