A First Course on Kinetics and Reaction Engineering

Example S4.4

Problem Purpose

The purpose of this example is to illustrate how to use the MATLAB template file FitNumAlgMR to perform a regression analysis for multiple response data with a model that consists of a set of non-linear algebraic equations that must be solved numerically.

Problem Statement

Suppose that 8 experiments were performed wherein the values of two variables, x_1 and x_2 were set and then the values of variables y_1, y_2 and y_3 were measured. The results of the experiments are shown in Table 1. Additionally suppose that equations (1) through (3) constitute a model of the experiments. The goal here is to fit the model to the data and assess the quality of the fit.

Table 1. Data for Example 1.

x_1	x_2	$\hat{y}_1$	$\hat{y}_2$	$\hat{y}_3$
0.4809	36.0	0.4188	0.0063	0.0298
0.4809	143.0	0.2882	0.0046	0.1200
0.4809	296.0	0.2188	0.0027	0.1682
0.4809	561.0	0.0868	0.0011	0.2578
0.4725	25.0	0.4163	0.0067	0.0279
0.4725	185.0	0.2629	0.0043	0.1334
0.4725	335.5	0.1598	0.0021	0.2056
0.4725	462.0	0.1074	0.0014	0.2326

$$y_1 - x_1 \exp\left(-\theta_1 x_2\right) = 0 \tag{1}$$

$$y_2 - \theta_1 x_1 \left(\frac{\exp\left(\left(-\theta_2 - \theta_3\right) x_2\right) - \exp\left(-\theta_1 x_2\right)}{\theta_1 - \theta_2 - \theta_3} \right) = 0 \tag{2}$$

$$y_3 - \theta_2 x_1 \left(\frac{1}{\theta_2 + \theta_3} + \frac{\exp\left(-\theta_1 x_2\right)}{\theta_1 - \theta_2 - \theta_3} - \frac{\theta_1 \exp\left(\left(-\theta_2 - \theta_3\right) x_2\right)}{\left(\theta_1 - \theta_2 - \theta_3\right)\left(\theta_2 + \theta_3\right)} \right) = 0 \tag{3}$$

Problem Analysis

The data points in this problem have three response variables, and the value of all three responses was measured for every data point. In other words, the data set in this problem is a complete, multiple response data set. The model consists of a set of algebraic equations. These facts indicate that the model can be fit to the data using the MATLAB template file FitNumAlgMR.m.

Problem Solution

Notice, the model equations do not contain unknowns represented by $\underline{u}$. That is because in this problem, the responses are equal to the unknowns in the model equations. You may recall that the template file is set up to solve the model equations for $\underline{u}$ and then use the result to calculate $\underline{y}$. Here it is trivial to recast the equations in that form, as shown in equations (4) through (9).

$$u_1 - x_1 \exp\left(-\theta_1 x_2\right) = 0 \tag{4}$$

$$u_2 - \theta_1 x_1 \left(\frac{\exp\left(\left(-\theta_2 - \theta_3\right)x_2\right) - \exp\left(-\theta_1 x_2\right)}{\theta_1 - \theta_2 - \theta_3} \right) = 0 \tag{5}$$

$$u_3 - \theta_2 x_1 \left(\frac{1}{\theta_2 + \theta_3} + \frac{\exp\left(-\theta_1 x_2\right)}{\theta_1 - \theta_2 - \theta_3} - \frac{\theta_1 \exp\left(\left(-\theta_2 - \theta_3\right)x_2\right)}{\left(\theta_1 - \theta_2 - \theta_3\right)\left(\theta_2 + \theta_3\right)} \right) = 0 \tag{6}$$

$$y_1 = u_1 \tag{7}$$

$$y_2 = u_2 \tag{8}$$

$$y_3 = u_3 \tag{9}$$

(In fact, looking at equations (1) through (3), you can see that they can be solved analytically for the values of the response variables. In other words, it isn't necessary to solve them numerically. However, the template file is set up to do so, a numerical solution will be performed. Therefore, from this point forward, equations (4) through (6) will be taken to be the model equations and equations (7) through (9) will be taken to represent the functions f referred to in the informational reading.)

While the main purpose here is to illustrate how to use FitNumAlgMR.m, in the course of doing so, I will also illustrate two "tricks" that can sometimes help in getting the code to converge to a solution. First, let's suppose that the values of the parameters, θ_1, θ_2 and θ_3, might fall anywhere in a range from 10^{-10} to 10^{10}. (This might be the case if the parameters represent rate coefficients.) It will be difficult to provide guesses that are close to the best values, and as the code attempts to find the best values of the parameters, the routine will have to vary the guesses for the parameters over that very large span. Therefore, instead of using the parameters directly, I'll use their base-10 logarithms. In that way, my guess only has to be in the range from -10 to +10, and the fitting routine only has to vary my guesses over that same range. The only thing I need to remember when I do this, is that when I enter the code for equations (4) through (6) in the internal function `nlaeqns`, I need to use 10^{θ_1} instead of θ_1; 10^{θ_2} instead of θ_2 and 10^{θ_3} instead of θ_3.

The other trick is related to guessing values for the unknowns in the model equations. Recall, in the internal function `mrmodel`, one of the required modifications is to provides guesses for u_1, u_2 and u_3 just before `fsolve` is called to solve equations (1) through (3) for a given data point. The trick I'm going to use is to calculate these guesses using the ***measured*** responses, $\hat{y}_1$, $\hat{y}_2$ and $\hat{y}_3$. For this problem that is trivial because the unknowns are equal to the responses. Therefore, I can use $\hat{y}_1$ (the measured response) as a guess for y_1 (the predicted response) which in this case equals the unknown u_1. The same can be done to generate guesses for u_2 and u_3.

Following the step-by-step instructions for the use of FitNumAlgMR.m, a copy of that file was saved as S4_Example_4.m and is provided with this supplemental unit. The introductory comment and the function name were then changed. There are six modifications that must be made each time this template file is used. They are indicated in the code by a comment that begins "% EDIT HERE". The first required modification involves defining any experimental and universal constants, but in this problem there aren't any experimental or universal constants in the model. Consequently, the first "% EDIT HERE:" comment was simply deleted. 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 eight rows in the matrix, corresponding to the eight 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 1 shows the code after these modifications were completed.

The fourth required modification appears within the internal function, `nlaeqns`. It involves evaluating the three functions that constitute the model, namely equations (4) through (6), and returning the results as a column vector, `g`. You can see that the only argument received by `nlaeqns` is the values of u_1, u_2 and u_3, in the form of a column vector named `u`. Looking at the model equations (4) through (6), you can see that they also contain the model parameters, θ_1, θ_2 and θ_3, and the set variables, x_1 and x_2. The parameters and set variables are available in the column vectors `p` and `x_set`. Recalling that the guesses I'll provide for the parameters are the base 10 logs of the actual parameters, θ_1 is equal to `10^p(1)`, θ_2 is `10^p(2)`, θ_3 is `10^p(3)`, x_1 is `x_set(1)` and x_2 is `x_set(2)`. With this knowledge, the modification of internal function nlaeqns is straightforward, as shown In Listing 2.

The other two required modifications takes place within the internal function, `mrmodel`. They occur within a loop through each of the experimental data points, i. The first is where guesses for the model unknowns, u_1, u_2 and u_3, for data point i need to be entered. The guesses were generated from the measured responses for data point i as described earlier and shown in Listing 3. Just a few lines after that modification, still within the loop through the data points within the internal function, `mrmodel`, 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 the comment indicates that the values of the response variables need to be calculated. In this problem, the response variables are calculated using equations (7) through (9), so this modification simply involves adding the code to calculate y_1, y_2 and y_3 using those equations, as also shown in Listing 3.

```
% Modified version of the MATLAB template file FitNumAlgMR.m that was
% modified for the solution of Example 4 of Supplemental Unit S4 of "A
% First Course on Kinetics and Reaction Engineering."
%
function S4_Example_4(p_guess)
    % Set variables
    x = [0.4809   36.0
    0.4809  143.0
    0.4809  296.0
    0.4809  561.0
    0.4725  25.0
    0.4725  185.0
    0.4725  335.5
    0.4725  462.0];

    % Experimental response variables
    y_hat = [0.4188     0.0063      0.0298
    0.2882  0.0046      0.1200
    0.2188  0.0027      0.1682
    0.0868  0.0011      0.2578
    0.4163  0.0067      0.0279
    0.2629  0.0043      0.1334
    0.1598  0.0021      0.2056
    0.1074  0.0014      0.2326];
```

Listing 1. Results of modifications to change the file name and enter the set and response variable values.

```
    function g = nlaeqns(u)
        % Current parameter values are available in column vector p
        % Current set variables are available in column vector x_set
        x1 = x_set(1);
        x2 = x_set(2);
        p1 = 10^p(1);
        p2 = 10^p(2);
        p3 = 10^p(3);
        s1 = p2 + p3;
        s2 = p1-p2-p3;
            g = [
                  u(1) - x1*exp(-p1*x2);
                  u(2) - p1*x1*((exp(-s1*x2)-exp(-p1*x2))/s2);
                  u(3) - p2*x1*(1/s1+exp(-p1*x2)/s2-p1*exp(-s1*x2)/s1/s2);
            ];
    end % of internal function nlaeqns
```

Listing 2. Internal function nlaeqns after modification.

```
    function y = mrmodel(p_current)
        % Make the current parameters available to the model equations
        p = p_current;
        y = zeros(n_data,n_resp);

        % loop through the data points
        for i = 1:n_data
            % Get the set variables
            for j = 1:n_set
                x_set(j) = x(i,j);
            end

            % provide guesses for the solution to the model equations
            u_guess = [
                y_hat(k,1);
                y_hat(k,2);
                y_hat(k,3);
            ];

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

            % Use the solution to calculate the responses
            y(i,:) = [
                u(1);
                u(2);
                u(3);
            ];
        end
    end % of internal function mrmodel
```

Listing 3. Internal function mrmodel after modification.

That completes the required modification of S4_Example_4.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_4 is executed, it must be passed a column vector that contains guesses for each of the base-ten logs of the three parameters in the model, as discussed previously. My first few guesses were so far off that the function made essentially no progress toward minimizing the objective function. When I used guesses of -4, -4 and -6, the code ran, but the final fit did not look very good, as evidenced, for example, by the deviation of the data from the diagonal in the parity plot for y_3, Figure 1. (It also produced the fsolve "equation solved" message many, many times, see Supplemental Unit S4. Here I will not show that message; the modified template file provided with this unit has the call that generates the messages commented out and replaced by two lines that suppress the messages. I don't recommend that you do this until you are sure that fsolve is converging properly.)

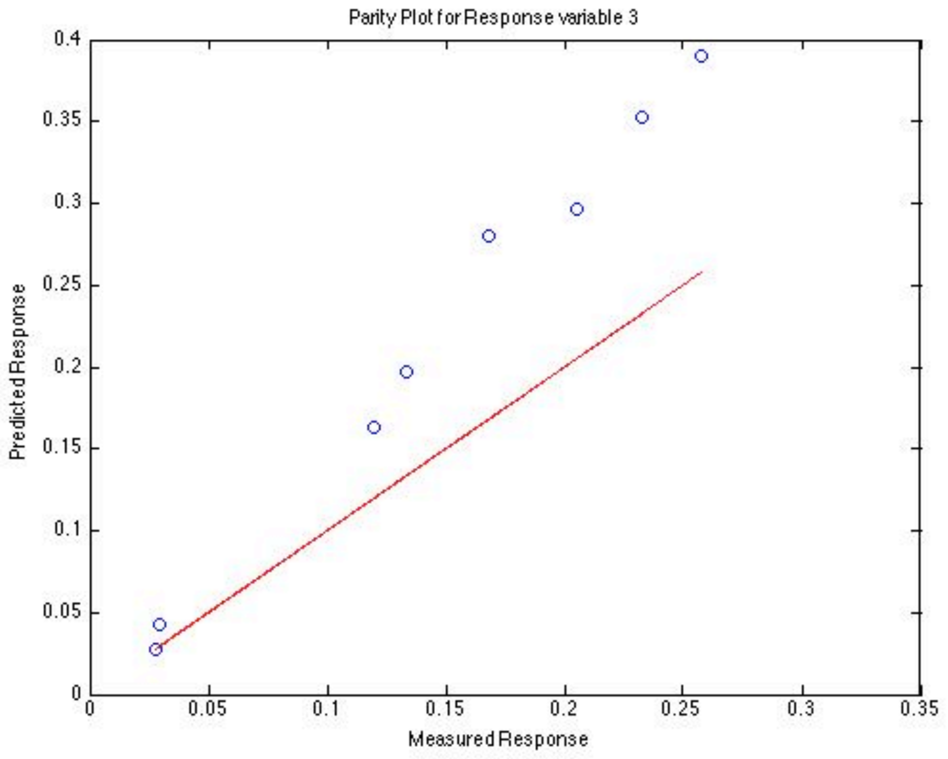

Figure 1. Parity plot showing the predicted versus measured values of response variable y_1.

At that point, I copied the fitted parameters displayed as the column vector `pf` into the column vector `p_guess` and re-executed S4_Example_4.m. Doing so returned different parameter values, indicating that the minimization had not converged. Therefore, I keep repeating this process of copying `pf` into `p_guess` and re-executing S4_Example_4.m. Eventually I reached the point where the returned parameter values were the same as the values I had guessed, indicating that the minimization had converged. This is shown in Listing 4, which shows the final parameter copy and the text output from running S4_Example_4.m. In addition to the text output, nine figures were generated, including Figures 2 through 5 included here.

```
>> p_guess = [
  -2.4941
  -0.8519
  -1.1161
];
>> S4_Example_4(p_guess)
Best Values for the Parameters:

pf =
  -2.4941
  -0.8519
  -1.1161
```

Listing 4. Output from the final re-execution of S4_Example_4.m.

Unlike the other curve fitting template files, `FitNumAlgMR` does not report a correlation coefficient nor 95% uncertainty limits for the fitted parameters. (For single response data, these quantities can be computed using the built-in MATLAB functions `nlinfit` and `nlparci`; I am not aware of equivalent built-in functions that can be used with multiple response data.) It does generate parity plots and residuals plots, however. Here the three parity plots are presented in Figures 2 through 4. In all three cases, the data points lie close to the diagonal line, indicating a reasonably good fit. Six residuals plots are also generated, but here I've only shown Figure 5 as a representative example. Figure 5 shows the residuals for y_2 as a function of the set variable x_2. You might argue that the residuals are all positive at higher values of x_2, but there really are too few data points to draw any conclusions here. If one were fitting this model for anything other than illustrational purposes, then there should be a lot more data than just 8 data points.

It is also important to remember that the parameters being fit here using FitNumAlgMR are the base 10 logarithms of the actual parameters that appear in the equations. Therefore, the fitted values of the model parameters are given in equations (10) through (12).

$$\theta_1 = 10^{-2.4941} = 0.0032 \tag{10}$$

$$\theta_2 = 10^{-0.8519} = 0.141 \tag{11}$$

$$\theta_3 = 10^{-1.1161} = 0.0765 \tag{12}$$

This might represent a local minimum, and not the global minimum of the objective function. However, with so few data, it probably does not make sense to repeat the whole process using a different initial guess to see if a different minimum can be found. In a situation where it was important that the model be accurate and correct, this would be a required step, but in that case there would also (hopefully) be many, many more data points.

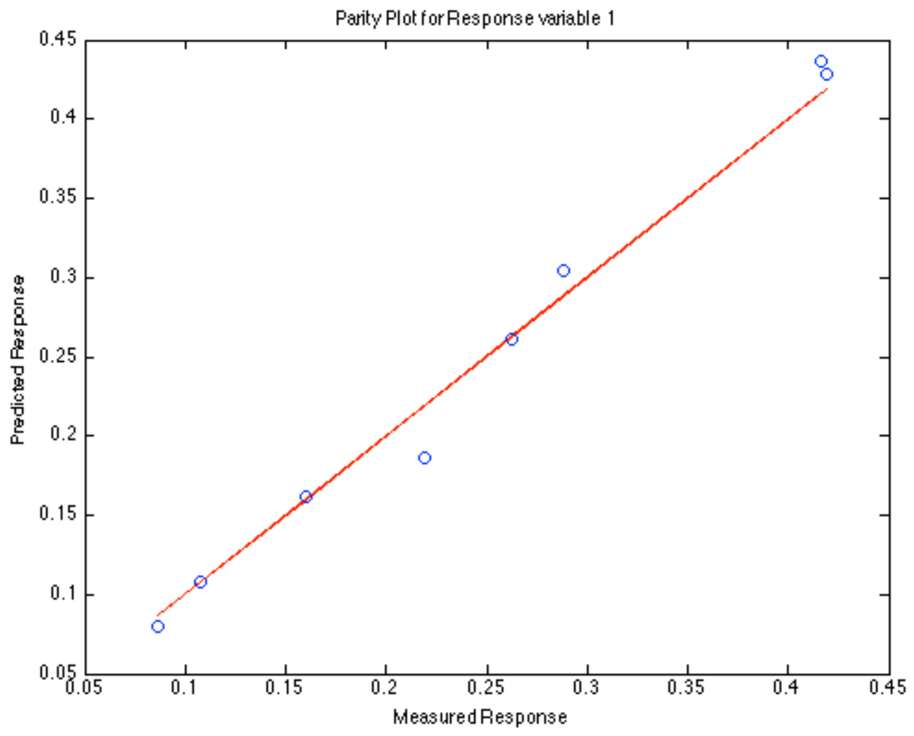

Figure 1. Parity plot showing the predicted versus measured values of response variable y_1.

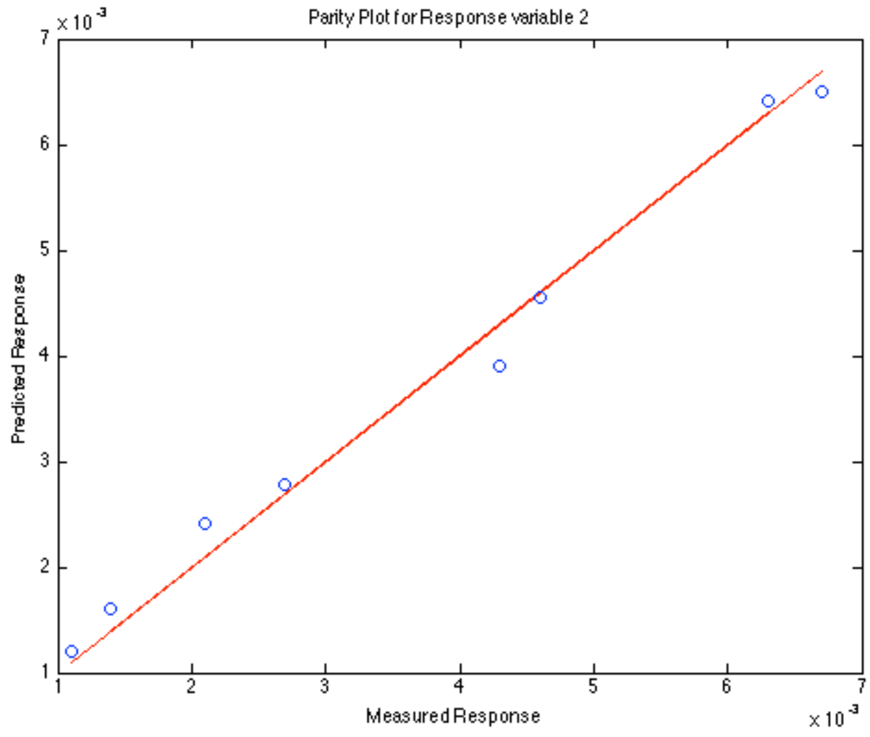

Figure 2. Parity plot showing the predicted versus measured values of response variable y_2.

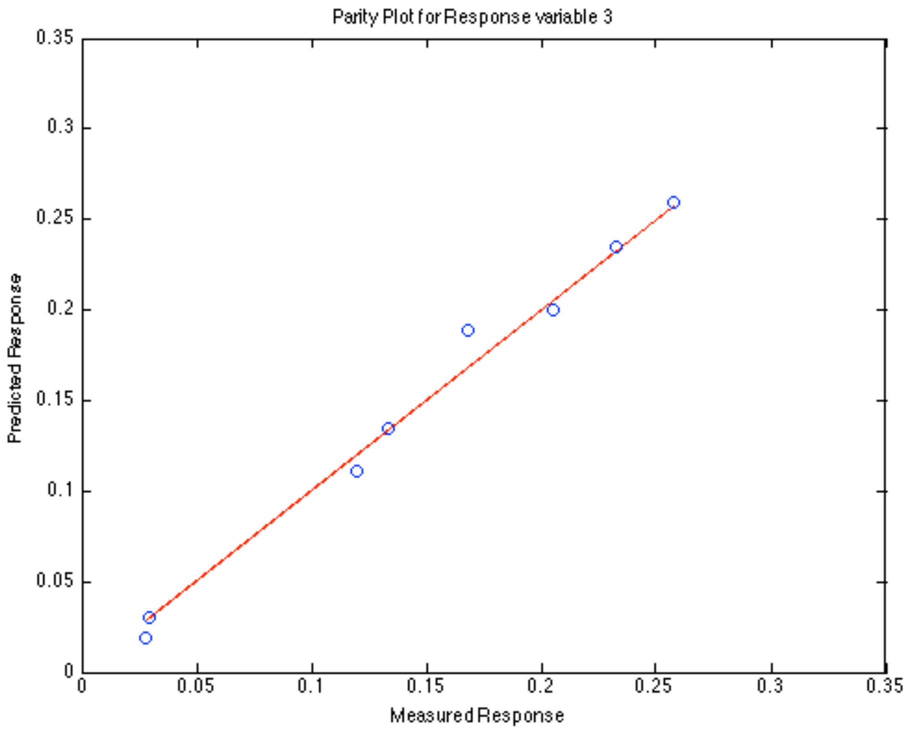

Figure 3. Parity plot showing the predicted versus measured values of response variable y_3.

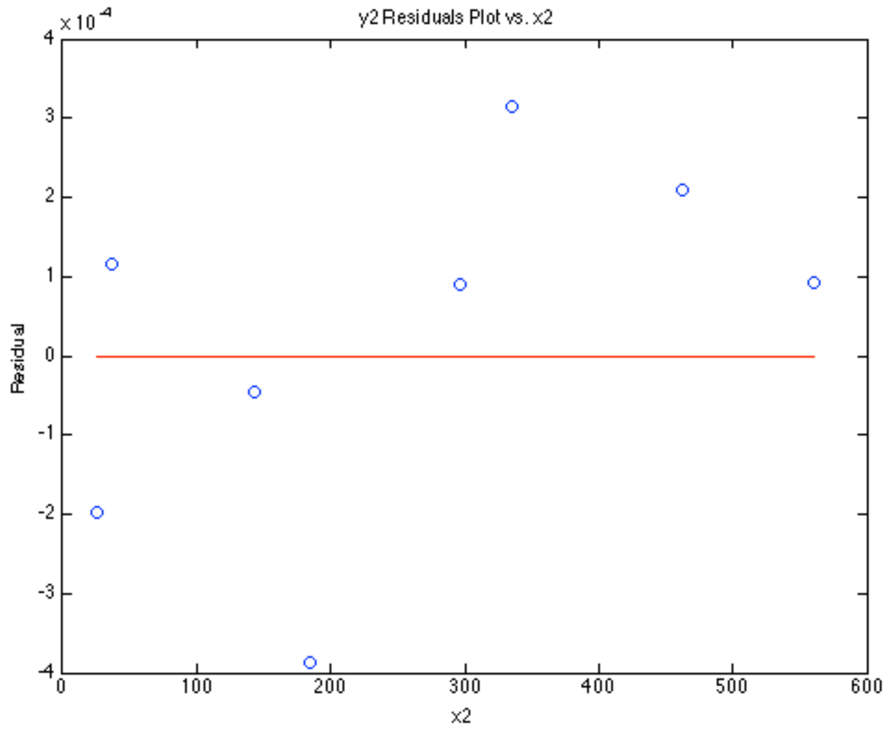

Figure 4. Residuals plot for response variable y_2 versus set variable x_2.