A First Course on Kinetics and Reaction Engineering

Example S4.1

Problem Purpose

The purpose of this example is to illustrate how to fit an explicit nonlinear model to single response experimental data using the MATLAB template file FitNonlinSR.m.

Problem Statement

The data set shown in Table 1 includes the values of the two set variables, x_1 and x_2 and the measured response variable, $\hat{y}$, for each of 11 experimental data points. Suppose equation (1) needs to be fit to these data, it must be decided whether the fit is acceptable, and if it is acceptable, the best values for the parameters θ_1 through θ_3 must be reported.

Table 1

x_1	x_2	$\hat{y}$
1	0	7.8
1.1	1	10.3
1.2	2	12.1
1.3	3	15.0
1.4	4	16.4
1.5	5	18.2
1.6	6	20.0
1.7	7	23.7
1.8	8	24.6
1.9	9	28.3
2	10	29.6

$$y = \left(\theta_1 x_1 + 1\right)\sqrt{\theta_2 x_2 + \theta_3} \tag{1}$$

Problem Analysis

Equation (1) gives an explicit equation for the response that is non-linear and that cannot be easily re-written in a linear form. In addition, there is single response variable in the data set. The MATLAB template file FitNonlinSR.m can be used for fitting an explicit non-linear model to single response data.

Problem Solution

To begin, a copy FitNonlinSR.m was saved as S4_Example_1.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 FitNonlinSR.m used to solve
% Example 1 of Supplemental Unit 4 of "A First Course on Kinetics and
% Reaction Engineering."
%
function S4_Example_1(p_guess)
```

Listing 1. Function statement after changing the filename.

There are four 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 entering any constants from the problem statement. This problem does not involve any constants other than the unknown model parameters, so no modification is necessary. 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 experimental 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.

The final required modification entails entering the code to calculate the response, y, for data point i using the model, equation (1). It appears within an internal function named nlmodel. As the comments in the template file indicate, the values of the parameters and the set variables for data point i are available at the point of the modification in the form of the column vector p and in row i of the matrix x. Hence, the relationship between the quantities in equation (1) and the corresponding quantities in the code are given in equations (2) through (6). Using those relationships, equation (1) is coded as shown in equation (7) and Listing 3.

$$x_1 = \texttt{x(i,1)} \qquad (2)$$

$$x_2 = \texttt{x(i,2)} \qquad (3)$$

$$\theta_1 = \texttt{p(1)} \qquad (4)$$

$$\theta_2 = \texttt{p(2)} \qquad (5)$$

$$\theta_3 = \texttt{p(3)} \qquad (6)$$

$$y = \left(\theta_1 x_1 + 1\right)\sqrt{\theta_2 x_2 + \theta_3} = \texttt{(p(1)*x(i,1)+1)*sqrt(p(2)*x(i,2)+p(3))} \qquad (7)$$

```
% Set variables
x = [1  0
1.1     1
1.2     2
1.3     3
1.4     4
1.5     5
1.6     6
1.7     7
1.8     8
1.9     9
2 10];
% response variables
y_hat = [7.8
10.3
12.1
15.0
16.4
18.2
20.0
23.7
24.6
28.3
29.6];
```

Listing 2. Results of modifying the template file by entering the set variables and the experimentally measured response variables.

```
% Function that calculates the responses using the model
function y = nlmodel(p,x)
    n_data = size(x,1);
    y = zeros(n_data,1);
    for i = 1:n_data
        % p(k) is the value of parameter k
        % x(i,j) is the value of set variable xj for data point i
        y(i) = (p(1)*x(i,1)+1)*sqrt(p(2)*x(i,2)+p(3));
    end
end % of internal function nlmodel
```

Listing 3. Internal function nlmodel after completing the required modification.

That completes the required modification of S4_Example_1.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_1 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 4 indicates, I simply guessed a value of 1.0 for each of the three parameters. After that, the file was executed as shown in Listing 4, which also shows the output that was generated.

```
>> p_guess = [1
1
1];
>> S4_Example_1(p_guess)

r_squared =

   9.9458e-01

Best Values for the Parameters:

pf =

   1.7203e+00
   3.6636e+00
   8.5972e+00

95% Confidence Intervals for the Parameters:

pf_u =

   3.1264e+00
   1.0742e+01
   1.8206e+01
```

Listing 4. Creation of the input column vector, p_guess, execution of S4_Example_1, and the resulting output.

In addition to the text output, the plots shown here as Figures 1 through 3 were generated and displayed. In order to check that the minimization has converged, the fitted parameters should be saved as `p_guess` and the modified template file should be executed again with this new guess. In this case, doing so will generate results that are almost identical to those shown in Listing 4. The only differences appear in the fourth decimal place of the uncertainties in the parameters. A change of that magnitude is negligible, and it can be concluded that the minimization process has converged.

It is possible that the minimum found here is a local minimum, and not the global minimum of the objective function. In this case there isn't any physical basis to the model; it's just an equation. As such, I have no basis for judging whether the parameter values are reasonable, but the fit is excellent, as discussed below. Therefore I will accept this result as representing the best values of the parameters in the model.

The output shows that the correlation coefficient is 0.9946, which is very close to 1.0 and indicates that the fit is very good. Still, the graphical output should be examined to see if the data display any trends that aren't being captured by the model. Here we have three variables, x_1, x_2 and y. In this case, a plot of the model would be a three-dimensional surface, while the data would still be points, but located in three dimensional space. As such, it could be very hard to gauge the magnitude and randomness of the deviations of the data from the model. For this reason, the template file does not generate a 3 dimensional model plot, instead, it generates the parity plot shown here as Figure 1 wherein the values of the response predicted by the model are plotted against the experimentally measured values of the response. If the model and the data were perfect, every point would fall on a diagonal line like the red line in the figure. For real data we expect some random scatter of the data about the line. If the fit is good, the deviations of the data points from the line should be small. Examination of Figure 1 reveals this to be true for this problem, as would be expected since the reported correlation coefficient is very nearly equal to 1.0.

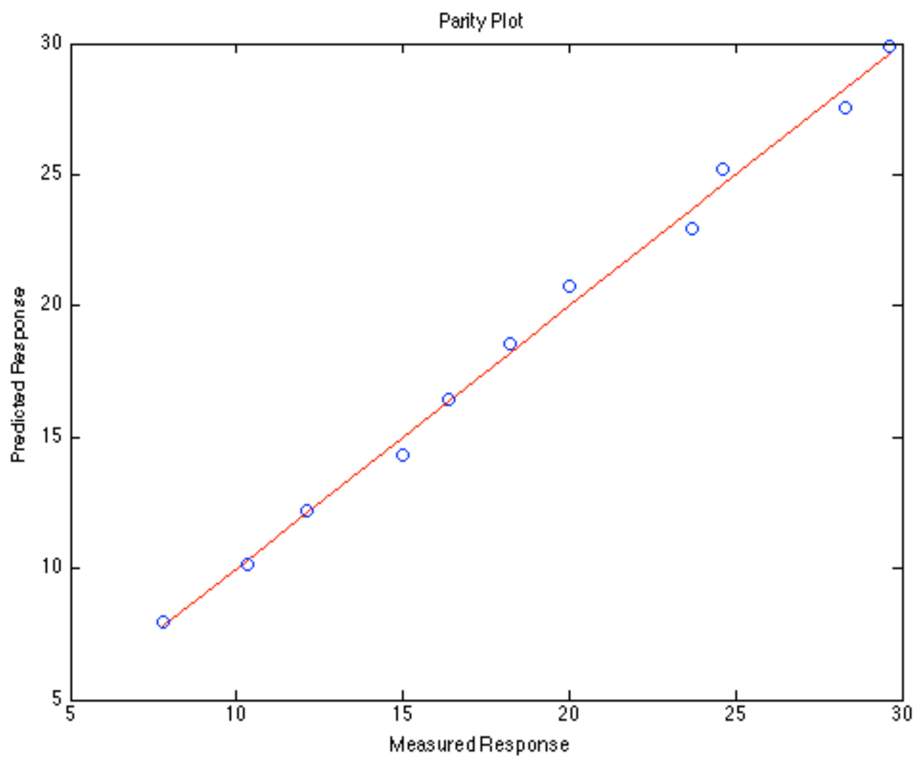

Figure 1. Parity plot showing the predicted responses versus the measured responses.

A parity plot such as Figure 1 may not reveal all *systematic* deviations of the data relative to the model predictions, so the MATLAB template file also produces a residuals plot for each set variable. Here the residuals plots are shown as Figures 2 and 3. The deviations from expected behavior often look very large in a residuals plot because the scale of the plot is typically highly magnified. Truly large deviations would be apparent in the parity plot, Figure 1. For this reason, residuals plots are more important for identification of trends in the data that are not fully captured by the model. That is, they can reveal situations where, for example, the errors are always positive at high values of one of the set variables. In the present case, the residuals plots do not reveal any systematic deviations of the data from the model.

Considering the correlation coefficient, parity plot and residuals plots, we can conclude that equation (1) fits the data very well. The best values of the parameters are θ_1 = 1.72 ± 3.13, θ_2 = 3.66 ± 10.74 and θ_3 = 8.60 ± 18.21 (95% confidence range based on 11 data).

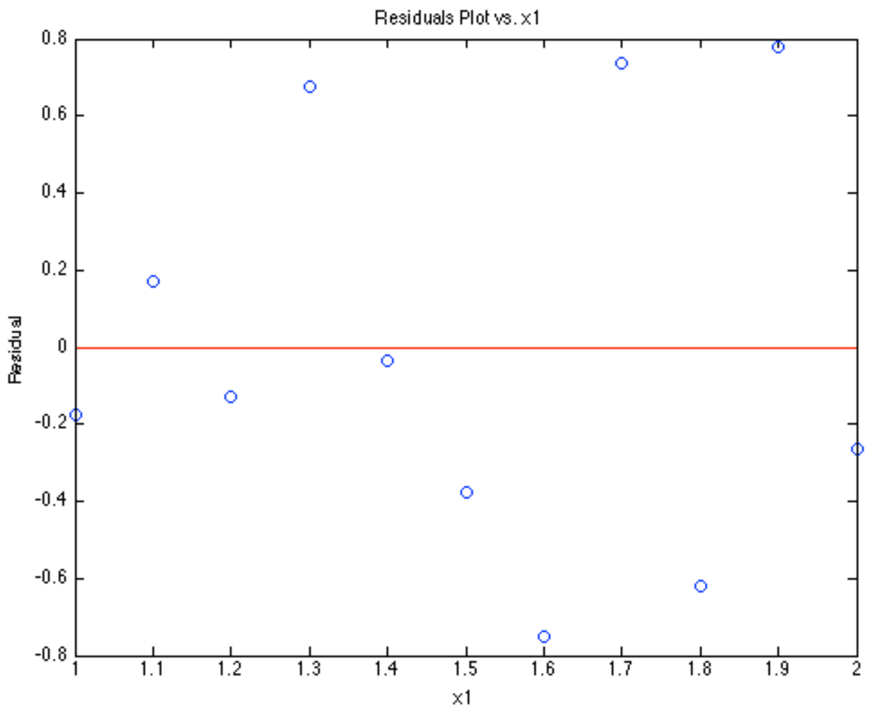

Figure 2. Residuals plot versus set variable x_1.

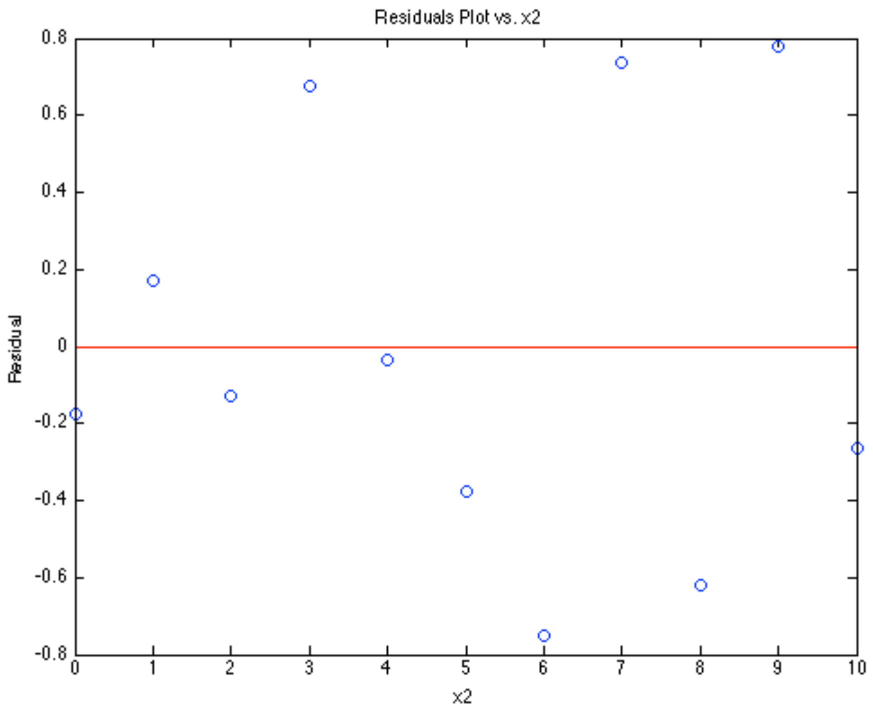

Figure 3. Residuals plot versus set variable x_2.