A First Course on Kinetics and Reaction Engineering

Example 14.2

Problem Purpose

This problem illustrates the use of finite differences and polynomial fitting to estimate the value of a derivative and its use in the differential analysis of batch reactor kinetics data.

Problem Statement

The kinetics of reaction (1) were studied using a 2.3 L batch reactor. The reaction takes place in the gas phase at a constant temperature of 848K. An experiment began with pure reactant at a pressure of 1036 Torr. The partial pressure of I_2 was measured as a function of time. The data are given in the table below.

$$2\ C_3H_5I \rightleftarrows C_6H_{10} + I_2 \qquad (1)$$

a. Using the differential method, test a first order rate expression, $r_{C_3H_5I} = -kC_{C_3H_5I}$, using these data.

b. Using the differential method, test a second order rate expression, $r_{C_3H_5I} = -kC^2_{C_3H_5I}$, using these data.

Table 1.

Time (ms)	P_{I_2} ***(Torr)***
0	0
0.1	121
0.2	202
0.3	275
0.4	336
0.5	375
0.6	417
0.7	416
0.8	433
0.9	457
1.0	476

Problem Analysis

This problem gives batch reactor kinetics data and asks us to test two rate expressions using differential data analysis. To do this, we will write the mole balance design equation, substitute the rate expression and fit the result to the experimental data, treating the derivative in the design equation as if it is an experimentally measured variable.

Problem Solution

First, to simplify the notation, we will let A denote C_3H_5I , Y denote C_6H_{10} and Z denote I_2. Equation (1) can be re-written using this notation.

$$2\,A \rightleftarrows Y + Z \tag{1}$$

(a) For an isothermal batch reactor with a single reaction taking place, all that is needed to model the reactor is a mole balance design equation written for one of the reactants or products of the reaction. Since we are given rate expressions with respect to the reactant A, a mole balance on A, equation (2), will be used here, though a balance on Y or Z would work equally well. The rate expression to be tested in part (a) of the problem is given in equation (3). Substitution of the rate expression into the design equation leads to equation (4).

$$\frac{dn_A}{dt} = Vr_{A,1} \tag{2}$$

$$r_{A,1} = -kC_A \tag{3}$$

$$\frac{dn_A}{dt} = -kVC_A \tag{4}$$

Since we are told to use the differential method, the derivative appearing in equation (4) will be treated like an experimental variable, and equation (4) will be fit directly to the experimental data. Equation (4) is a linear equation, as can easily be seen if x, y and m are defined as in equations (5), (6) and (7). Upon substitution of these definitions, equation (4) becomes equation (8), which is clearly a straight line through the origin.

$$x = -VC_A \tag{5}$$

$$y = \frac{dn_A}{dt} \tag{6}$$

$$m = k \tag{7}$$

$$y = mx \tag{8}$$

Since the reactor model, equation (8), is linear, it can be fit to the experimental data using linear least squares (see Supplemental Unit S3). Linear least squares fitting can be performed manually, using a

calculator, using a spreadsheet or using mathematics software. No matter which tool one chooses to use, it will be necessary to provide the following information and input data:

- the number of independent (x) variables
- whether or not the model includes an intercept (b)
- a set of experimental data points, each of which consists of a value for the dependent variable (y) and corresponding values for each of the independent variables (x_i)

Thus we need to calculate values of x and y for each experimental data point. To calculate x, we begin by substitution of the definition of concentration into equation (5), leading to equation (9). The problem statement does not give the value of n_A for each experimental data point; instead, it gives values of the partial pressure of Z. Since the volume and the temperature are also known, the moles of Z can be calculated for each experiment using equation (10). The moles of Z can be related to the moles of A by use of a mole table or simply by application of the definition of the extent of reaction (see Unit 1), equation (11). The reactor initially contains pure A, so the initial moles of A can be found using equation (12). Combining equations (9) through (12) leads to an expression for x in terms of experimentally measured quantities, equation (13).

$$x = -VC_A = -V\left(\frac{n_A}{V}\right) = -n_A \tag{9}$$

$$n_Z = \frac{P_Z V}{RT} \tag{10}$$

$$n_A = n_A^0 - 2n_Z \tag{11}$$

$$n_A^0 = \frac{P^0 V}{RT} \tag{12}$$

$$x = \frac{V}{RT}\left(2P_Z - P^0\right) \tag{13}$$

We also need to calculate the value of y for each experimental data point. This requires that we estimate the value of $\frac{dn_A}{dt}$ for each experimental data point. For illustrative purposes, two different ways of doing this will be presented here. One way to estimate the value of the derivative for a particular data point, j, involves using finite differences. We can choose to use forward differences, backward differences or central differences. Here we will use forward differences as expressed in equation (14). Note that because we are using forward differences, it will not be possible to estimate a value for $\frac{dn_A}{dt}$ for the last data point because we don't have a value for $(n_A)_{j+1}$ or $(t)_{j+1}$. By combining equations (10) through (12), as in equation (15), the value of n_A can be calculated for each data point, and the corresponding elapsed

times, t, are given in the data table, hence $\frac{dn_A}{dt}$, which equals y according to equation (6), can be calculated for each data point using equation (14).

$$\left.\frac{dn_A}{dt}\right|_j \approx \frac{(n_A)_{j+1} - (n_A)_j}{(t)_{j+1} - (t)_j} \tag{14}$$

$$n_A = \frac{V}{RT}\left(P^0 - 2P_Z\right) \tag{15}$$

An alternative way to estimate the value of $\frac{dn_A}{dt}$ for each data point involves fitting a polynomial to the n_A (from equation (15)) versus t data with the aim of producing a smooth curve through the data. The derivative of that polynomial with respect to t can then be used to estimate the value of $\frac{dn_A}{dt}$. We have no way of knowing what order polynomial to use, a priori, so we'll begin by fitting a quadratic, cubic and fourth order polynomial to the data and plotting the results as shown in Figure 1.

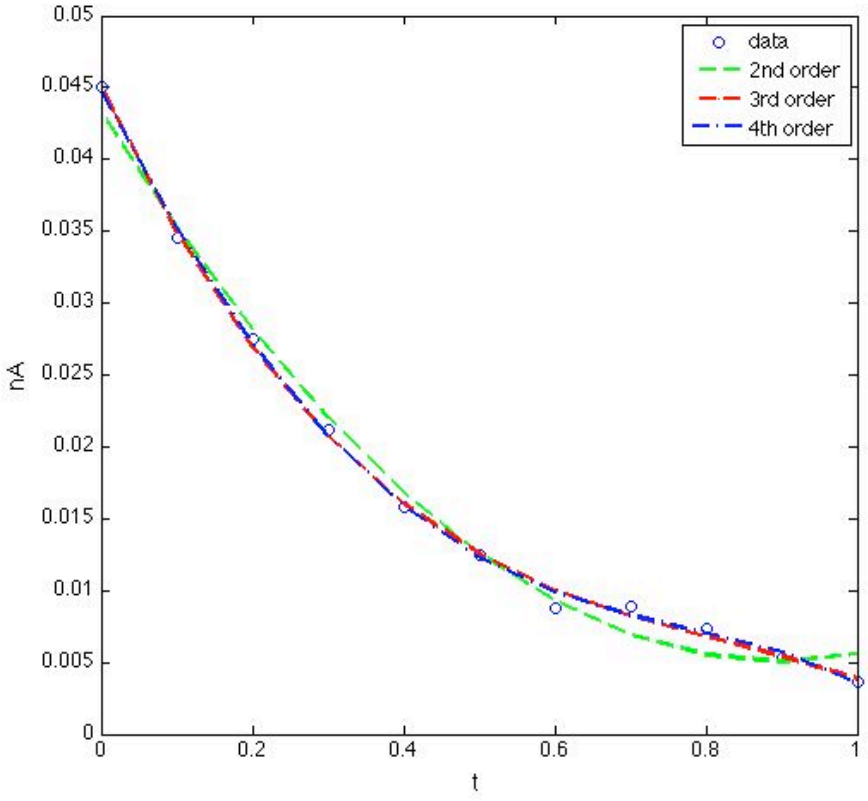

Figure 1. Second, third and fourth order polynomials fit to the n_A vs. t data

The second order polynomial deviates from the data slightly more than the third and fourth order polynomials, and it also appears to pass through a minimum around t = 0.8 while the data do not show a

minimum, so the second order polynomial will not be used. The third and fourth order polynomials give nearly identical fits, and they both fit the data well. Since the two are nearly identical, the third order polynomial will be used. The fitting process yielded the coefficients of the third order polynomial, as shown here in equation (16). Taking the derivative gives equation (17) which can then be used to calculate $\frac{dn_A}{dt}$, which equals y according to equation (6), for each data point.

$$n_A = -0.0474t^3 + 0.1187t^2 - 0.1126t + 0.0451 \tag{16}$$

$$\frac{dn_A}{dt} = -3(0.0474)t^2 + 2(0.1187)t - 0.1126 \tag{17}$$

Thus, we can generate a set of (x,y) data using finite differences, equations (13) and (14), or using polynomial approximation, equations (13) and (17). The model has only one independent variable, x, and it does not include an intercept. With this information and input data we have everything needed to fit the model to the data. Table 2 shows the values of the correlation coefficient, slope and slope uncertainty that result from fitting equation (8) to the (x,y) data generated using forward differences and the polynomial method. Figures 2 and 3 show the corresponding model plots for part (a).

Table 2. Fitting Results.

Method	***r²***	***m***
Part (a)		
Forward Differences	0.93	2.25 ± 0.27 ms^{-1}
Polynomial	0.99	2.53 ± 0.12 ms^{-1}
Part (b)		
Forward Differences	0.67	138 ± 38 L mol^{-1} ms^{-1}
Polynomial	0.71	155 ± 38 L mol^{-1} ms^{-1}

The polynomial method yielded a better fit, as evidenced by both the correlation coefficient being closer to 1.0 and by the magnitude of the deviations of the data from the line representing the model in the model plots. The fit from the polynomial method is acceptably accurate, given the small number of data used, so equation (3) can be accepted as the rate expression. Normally it would be necessary to calculate k and its uncertainty from m and its uncertainty, but in this problem, they are equal, so the best value of k is taken to be 2.53 ± 0.12 ms^{-1}.

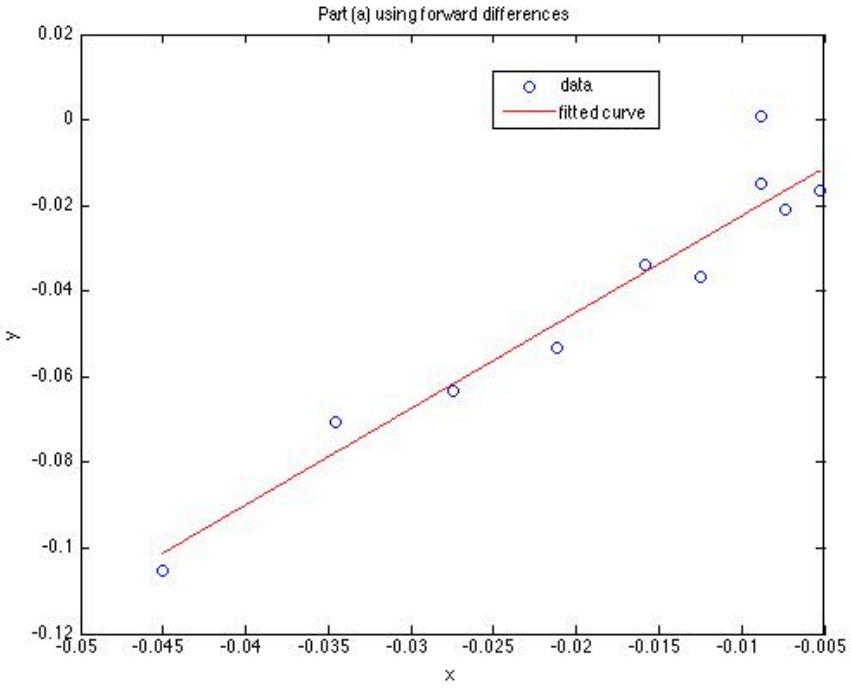

Figure 2. Model plot showing the experimental data as points and the model's predictions as a line for part (a) when the derivative was approximated using forward differences.

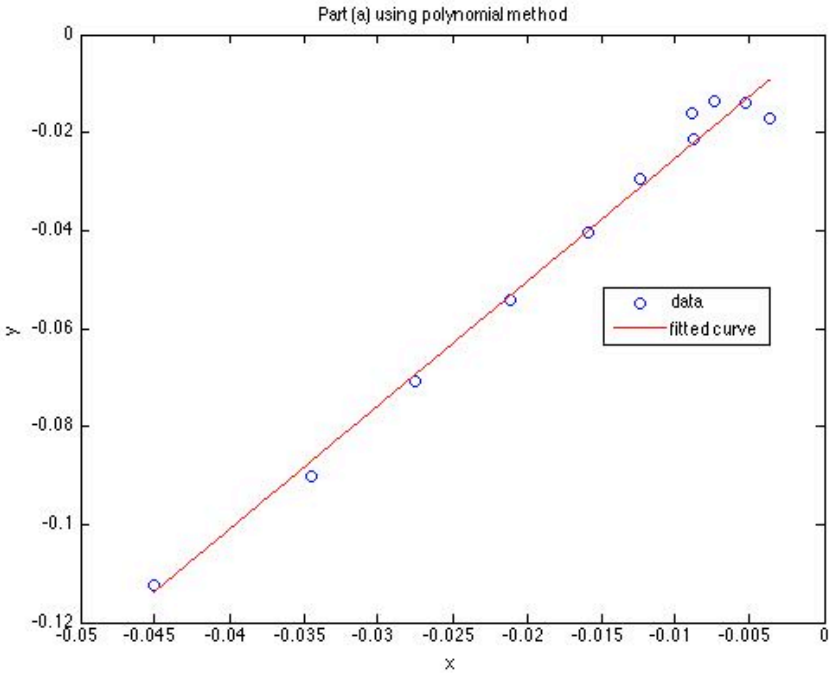

Figure 3. Model plot showing the experimental data as points and the model's predictions as a line for part (a) when the derivative was approximated using the polynomial method.

(b) The only difference in part (b) of the problem is that the rate expression being tested is equation (18), which leads to the mole balance design equation given in equation (19). Equation (19) is still linear with a y intercept equal to zero, and y and m are still defined as in equations (6) and (7). The only difference is that x is now defined as in equation (20). The estimated derivatives remain the same as in part (a), as well.

$$r_{A,1} = -kC_A^2 \tag{18}$$

$$\frac{dn_A}{dt} = -kVC_A^2 \tag{19}$$

$$x = -VC_A^2 = -V\left(\frac{n_A}{V}\right)^2 = \frac{-n_A^2}{V} \tag{20}$$

Hence, the fitting process is the same as above, except using equation (20) for the calculation of x. The correlation coefficient, slope and uncertainty in the slope that result from fitting using forward differences and the polynomial method for part (b) are also reported in Table 2. Given that the first order rate expression was found acceptable, it isn't very surprising that the correlation coefficients here are not close to 1.0. The model plots for part (b) are shown as Figures 4 and 5. They, too, show that the fit is poor, and consequently the rate expression given in equation (18) is not acceptable. One last point to note, in discussing the assessment of the quality of the fit, I have noted that there should not be systematic deviations of the data from the model. Figures 4 and 5 shows an extreme example of such systematic deviations: the data clearly curve away from the model line.

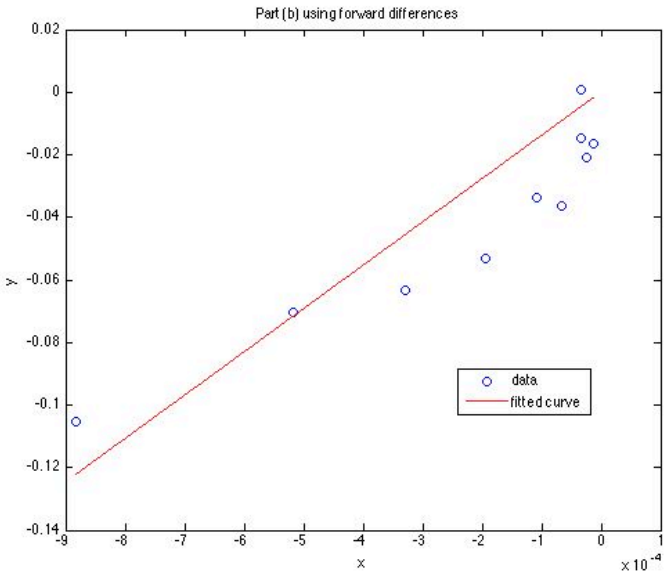

Figure 4. Model plot showing the experimental data as points and the model's predictions as a line for part (b) when the derivative was approximated using forward differences.

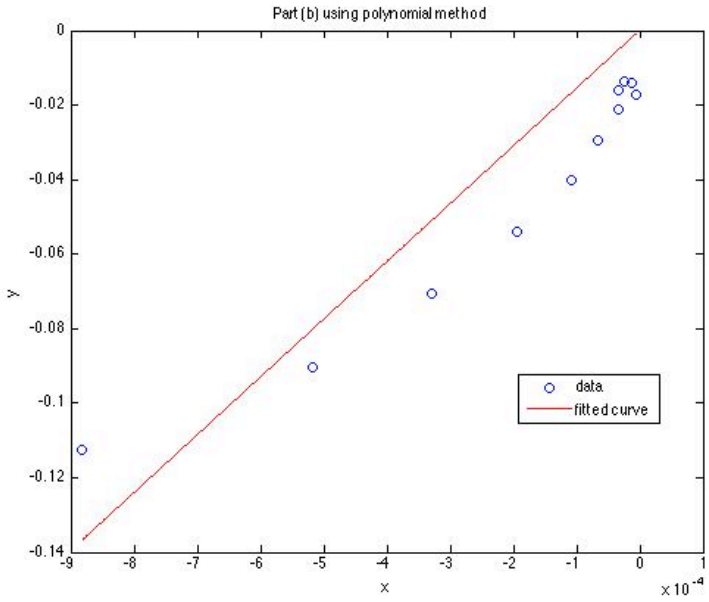

Figure 5. Model plot showing the experimental data as points and the model's predictions as a line for part (b) when the derivative was approximated using the polynomial method.

Calculation Details Using MATLAB

The MATLAB calculations for this problem were performed in two stages. The first stage involved fitting the three polynomials to the n_A vs. t data and plotting the results. To do this, I used two built-in MATLAB functions: `polyfit` and `polyval`. The former does the fitting while the latter makes it easy to plot the results; I will not describe these functions, but instead will leave it to you to consult the MATLAB documentation. The code for this task accompanies this solution as the file Example_14_2_polyfit.m; it is reproduced in Listing 1. The code is straightforward; first the constants given in the problem statement are entered (in consistent units), next the value of n_A is calculated for every experiment using equation (15), and finally the fitting and plotting are performed. The plot generated by the code was shown here as Figure 1.

Listing 2 shows the output from Example_14_2_polyfit.m. It consists of three row vectors that contain the coefficients for the three polynomial fits. You can see that the values returned for p3 are polynomial coefficients in equation (16).

```
% MATLAB file used in the solution of Example 14.2 of
% "A First Course on Kinetics and Reaction Engineering."

% Data provided in the problem statement, in consistent units
V=2.3; % L
T=848.0; % K
P0=1036.0; % Torr
Rgas=62.366; % L-Torr/mol/K
t = [0; 0.1; 0.2; 0.3; 0.4; 0.5; 0.6; 0.7; 0.8; 0.9; 1.0]; % ms
Pz = [0; 121; 202; 275; 336; 375; 417; 416; 433; 457; 476]; % Torr

% calculate nA using equation (15) from the solution.
nA = V/Rgas/T*(P0 - 2*Pz);

% fit 2nd, third and 4th order polynomials to the nA vs t data
p2 = polyfit(t,nA,2)
f2 = polyval(p2,t);
p3 = polyfit(t,nA,3)
f3 = polyval(p3,t);
p4 = polyfit(t,nA,4)
f4 = polyval(p4,t);

% plot the results
figure
plot(t,nA,'o',t,f2,'--g',t,f3,':r',t,f4,'-.b')
xlabel('t');
ylabel('nA');
legend('data','2nd order','3rd order','4th order');
```

Listing 1. MATLAB code used to generate Figure 1.

```
>> Example_14_2_polyfit

p2 =

    0.0476   -0.0854    0.0434

p3 =

   -0.0474    0.1187   -0.1126    0.0451

p4 =

   -0.0378    0.0281    0.0715   -0.1031    0.0448
```

Listing 2. Output generated by the MATLAB file Example_14_2_polyfit.m.

The fitting of the model to the experimental data involved linear least squares. Three MATLAB script files are provided with Supplemental Unit S3. The file names indicate the number of independent variables and whether or not the model has an intercept. The script named FitLinmSR is used when the model has one independent variable (x) and ***does not*** include the intercept (b). FitLinmbSR is used when the model has one independent variable and ***does*** include the intercept, and FitLinSR is used when the model has two or more independent variables. (With MATLAB, when the model has two or more

independent variables, it must have an intercept; Supplemental Unit S3 describes how to convert a model without an intercept into a model that has an intercept.) In this problem the model has one independent variable, but no intercept, so the script file named FitLinmSR was used. To do so, the script file must be located in the current MATLAB working directory or in the MATLAB search path.

Before executing FitLinmSR, the experimental values of x must be stored in a vector named x, and the experimental values of y must be stored in a vector named y_hat. Upon execution of the script, it will return the correlation coefficient, r^2, as r_squared, the slope, m, as m, and the 95% confidence limits on the slope, λ_m, as m_u. It will also generate a model plot. In this problem, the kinetic parameter, k, is equal to the fitted slope, m, so no additional calculations are needed to compute k and its uncertainty. The commands for performing these tasks can be entered at the MATLAB command prompt, but here they have been recorded in the MATLAB file named Example_14_2.m which accompanies this solution. That file includes the fitting code for all four cases encountered in this problem.

Example_14_2.m begins with the entry of the data from the problem statement. The value of n_A is then calculated for each data point using equation (15). A third order polynomial is then fit to the n_A vs. t data. The derivative of the polynomial, equation (17) is then used to calculate a value of $\frac{dn_A}{dt}$ for each experimental data point. Following that, the derivative is calculated for every data point except the last using forward differences, equation (14). This part of the code is shown in Listing 3.

```
% MATLAB file used in the solution of Example 14.2 of
% "A First Course on Kinetics and Reaction Engineering."

% Data provided in the problem statement, in consistent units
V=2.3; % L
T=848.0; % K
P0=1036.0; % Torr
Rgas=62.366; % L-Torr/mol/K
t = [0; 0.1; 0.2; 0.3; 0.4; 0.5; 0.6; 0.7; 0.8; 0.9; 1.0]; % ms
Pz = [0; 121; 202; 275; 336; 375; 417; 416; 433; 457; 476]; % Torr

% calculate nA using equation (15) from the solution.
nA = V/Rgas/T*(P0 - 2*Pz);

% fit third order polynomials to the nA vs t data
p = polyfit(t,nA,3);

% use its derivative to estimate dnA/dt
dnAdt_poly = 3*p(1)*t.*t + 2*p(2)*t + p(3);

% use forward differences to estimate dnA/dt for all but the last point
dnAdt_fd =zeros(10,1);
for i = 1:10
    dnAdt_fd(i) = (nA(i+1)-nA(i))/(t(i+1)-t(i));
end
```

Listing 3. Portion of the file Example_14_2.m showing the entry of the data from the problem statement and the calculation of the value of the derivative using the polynomial method and forward differences.

The next section of the code, shown in Listing 4, performs the calculations for parts (a) and (b) of the problem. For each case, the values of x and y are calculated and stored in the arrays x and y_hat. FitLinmSR is then called to perform the fitting. This generates the model plots shown previously. The output, which was used to fill in Table 2, is shown in Listing 5.

```
% solve part (a) of the problem
disp('part (a)');

%fit using forward differences (there's one less data point)
disp('forward differences results');
x = -nA(1:10);
y_hat = dnAdt_fd;
figure;
% Use the MATLAB script file "FitLinmSR.m" from "A First Course on
% Kinetics and Reaction Engineering" to fit a straight line through the
% origin to the data here and below
FitLinmSR;
title('Part (a) using forward differences');

% fit using the polynomial method
disp('polynomial method results');
x = -nA;
y_hat = dnAdt_poly;
figure;
FitLinmSR;
title('Part (a) using polynomial method');

% solve part (b) of the problem
disp('part (b)');

%fit using forward differences (there's one less data point)
disp('forward differences results');
x = -(nA(1:10).^2)/V;
y_hat = dnAdt_fd;
figure;
FitLinmSR;
title('Part (b) using forward differences');

% fit using the polynomial method
disp('polynomial method results');
x = -(nA.^2)/V;
y_hat = dnAdt_poly;
figure;
FitLinmSR;
title('Part (b) using polynomial method');
```

Listing 4. Portion of the file showing the use of FitLinmSR.m to fit the model to the data sets.

```
>> Example_14_2
part (a)
forward differences results

r_squared =

    0.9271

m =

    2.2488

m_u =

    0.2737

polynomial method results

r_squared =

    0.9872

m =

    2.5274

m_u =

    0.1211

part (b)
forward differences results

r_squared =

    0.6665

m =

  138.4473

m_u =

   37.8446

polynomial method results

r_squared =

    0.7061

m =

  155.0799

m_u =

   37.5565
```

Listing 5. Output from FitLinmSR.m.