Cross-Product Heteronomial Model Selection

by
Namir Clement Shammas
THE HERETIC EMPIRICIST

Contents

1/Introduction

Polynomials are very popular constructs for math, calculus, and curve fitting. A polynomial has one or more terms with the independent variable raised to an integer power. Parallel to polynomials are loosely named multivariable constructs that involve different independent variables. I will rename these constructs ***heteronomials***. An example of a simple linear heteronomial, often used in simple multiple linear regression is:

$$Y = a + b_1X_1 + b_2X_2 \qquad (1)$$

Where X_1 and X_2 are the independent variables and Y is the dependent variable. I call equation (1) a linear heteronomial because each variable is linear. A simple example of a non-linear heteronomial is:

$$Y = a + b_1/X_1 + b_2X_2^2 \qquad (2)$$

In equation (2) variable X_1 is raised to the power -1 and variable X_2 is raised to the power 2. More advanced heteronomials involve more independent variables and different powers like -3, -2, 3, 0.5, 1.5, and so on. Power zero is translated to the ln(x) function while the other powers are taken at face values. In this study, I limit the powers of the heteronomials to negative, zero (to use function ln(x)), and positive numbers (both integers and reals). The general form of a heteronomial is:

$$Y^{py} = a + b_1 X_1^{px1} + b_2X_2^{px2} + \ldots + b_nX_n^{pxn} \qquad (3)$$

Notice that each variable in equation (3) **can have** a non-unity power!

☛ Heteronomials are useful models for empirical study. They can find relations between variables that are not easily constructed or derived from mathematical analysis.

A **cross-product heteronomial** model has the general formula:

$$Y = a + \sum_{i=1}^{n} b_i\, X_i^{px(i)} + \sum_{j,k}^{m} c_{jk}\, X_j^{qx(j)}\, op(j,k)\, X_k^{qx(k)} \qquad (4)$$

The op(j,k) is the * operator if it is equal to 1, and is the / operator if it is not equal to 1. The independent variable X_i comes from a set of n variables. The first summation can include some or all the n variables. Likewise, the second summation includes a more specific number of cross-product terms with any of the n variables. I define a limited and special version of the cross-product heteronomial model as one where each cross-product term MUST use two different variables. Also notice that the dependent variable appears as linear values.

Here are examples of cross-product heteronomials:

$$Y = a + b_1*X_1^{px1} + b_2*X_2^{px2} + b_3*X_1^{px3}*X_2^{px4} \quad (5)$$
$$Y = a + b_1*X_1^{px1} + b_2*X_2^{px2} + b_3*X_1^{px3}/X_3^{px4} \quad (6)$$
$$Y = a + b_1*X_1^{px1} + b_2*X_2^{px2} + b_3*X_3^{px3} + b_4*X_1^{px4}*X_2^{px5} + b_5*X_1^{px6}/X_3^{px7} \quad (7)$$
$$Y = a + b_1*X_1^{px1} + b_2*X_2^{px2} + b_3*X_1^{px3}*X_2^{px4} + b_4*X_1^{px5}/X_3^{px6} \quad (8)$$
$$Y = a + b_1*X_2^{px1} + b_2*X_1^{px2}/X_2^{px3} + b_3*X_1^{px4}/X_3^{px5} \quad (9)$$

Equations (8) and (9) show that the single-variable terms can exclude one or more of the three variables involved in the model.

Selecting the best cross-product heteronomial is a bit more complicated because the optimization includes selecting whether to multiply or divide variables in the cross-product terms. I started testing the best model selection using evolutionary optimization algorithms. I used the approach where all the optimized powers in the cross-product terms are selected by integer coded values that are then translated to actual powers. Also selecting whether to multiply or divide the variables in a cross-product term involves random selection between 1 (integer code for multiplying) and 2 (integer code for diving). So, all the optimized variables had to be integers. The optimization process yielded correct answers only in the case of the simpler models that have one and two cross-product terms. The optimization process attempts to zigzag its way to the best values that optimize the target function.

I decided to also use a second different strategy. I applied a virtual nested loop programming trick to emulate a set of nested loops that tap into the ranges of the various optimized variables. These virtual nested loops would calculate the adjusted R-square statistic in hope of getting a value as close as or equal to 1. Using nested virtual loops is basically a brute force approach.

One design iteration for using the virtual nested loops was to optimize the powers for ALL the terms in the heteronomial. Unfortunately, the total number of iterations became unreasonably high! So, I made the following design decisions:

1. Allow the user to select the variables and select their **fixed** powers in the single-variable terms. These powers can be integers or floating-point values.

The user is responsible for choosing these powers and the variables. No iterations are involved for the powers and selection of variables in the single-variable terms.

2. Optimize the powers of cross-product terms and determine if dividing or multiplying these cross-product terms yield a better model.

☛What is unique about the optimization of cross-products in this study is that it includes selecting the best of multiplication and division (in other words, selecting between multiplying two variables or between multiplying one variable by the reciprocal of another variable).

Unlike my other heteronomial studies, I use integer codes to select the actual powers for ALL terms. The MATLB code uses a function that maps these integer codes into actual transformation. Such a function uses a series of if statements to transform the arrays of a dependent variable. For example, a zero-integer code leads to logarithmic transformations. Integer codes of 1, 2, 3, lead to linear, squared, and cubic transformations. The integer code of 4 leads to square root transformations. You can edit the code for this function to change, add, or delete transformations.

Bother approaches (using evolutionary optimization and virtual nested loops) work with Excel files of the same formats. The models with different number of cross-products use Excel files with specific formats.

2/The Excel Files

The model optimization process uses Excel files to supply input and output (the latter is also echoed to a diary text file). The Excel file has the sheet **Data** to store the values for the dependent variable and all the independent variables. The sheet **Transf** contains the data that guides the calculations and transformations

The Sheet **Data**

The sheet **Data** contains the input data and the following columns (see Figure 2.1):

1. The first row has the headers that **you must enter**.
2. Cell A has the header Y and the data for the dependent variable starting in row 2.
3. Columns B, C, and beyond have the header X1, X2, and so on. Rows 2 and down have the values for the various independent variables.

☛ In this study the dependent variable is calculated **without injecting any errors**. The reason for this is to test if the various methods can find the correct best model. Injecting errors in the dependent variable would easily steer the calculations away from the best correct model.

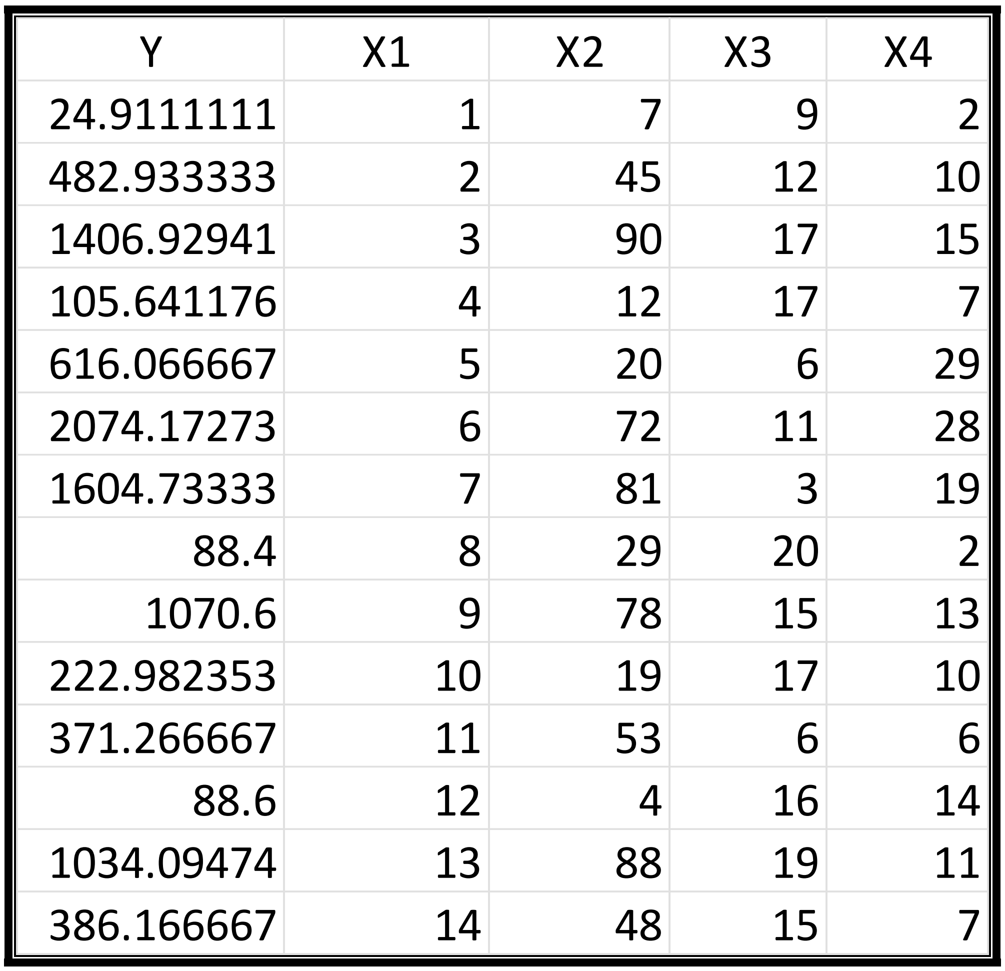
The Sheet **Transf**

The sheet **Transf** contains the indices for selecting variables and their transformations and the following columns and rows (see Figure 2.2):

1. The first row contains groups of headers **that you must enter**. The first group of headers is px1, px2, and so on. The second group of headers is i1, i2, and so on. The third group of headers is op1, op2, and so on.
2. The second row has the tag **Lb** in column A to indicate that the cells to its right side contain lower bound values. These values are the low powers of the cross-product terms (under the headers px1, px2, and so on), the low indices for selecting variables in cross-product terms (under the headers i1, i2, and so on), and values of 1 for the operator code (under the headers op1 and so on).
3. The third row has the tag **Ub** in column A to indicate that the cells to its right side contain upper bound values. These values are the high powers of the cross-product terms (under the headers px1, px2, and so on), the high indices for selecting variables in cross-product terms (under the headers i1, i2, and so on), and values of 2 for the operator code (under the headers op1 and so on).
4. The fourth row has the tag **Num Indepnt Variables** in column A. Cell B4 contains the number of independent variables in single-variable terms. Cells C4 and on contain the indices for these independent variables.
5. The fifth row has the tag **Powers of Xi** in column A. Cells C2 and C3 are empty. Cells C4 and on contain the **fixed powers that you select** for the independent variables in single-variable terms. Most common values are 1. I am allowing you to select values other than 1 in special cases for your study.

Please note that the values under the headers px1, px2 and so on, and under headers i1, i2, and so on (in rows 2 and 3) are meant to be *paired powers* and *paired indices* for cross-product terms. So, for example, the independent variables in the first cross-product term are selected by the values under headers i1 and i2 in rows 2 and 3.

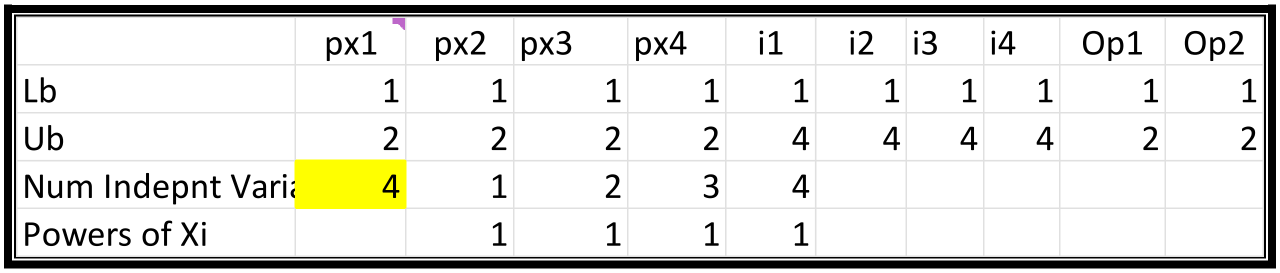
Note that you should enter all the headers in the Excel sheets. The names that you use are your choice and do not affect the calculations. The headers help you easily identify the input data and results.



Y	X1	X2	X3	X4
24.9111111	1	7	9	2
482.933333	2	45	12	10
1406.92941	3	90	17	15
105.641176	4	12	17	7
616.066667	5	20	6	29
2074.17273	6	72	11	28
1604.73333	7	81	3	19
88.4	8	29	20	2
1070.6	9	78	15	13
222.982353	10	19	17	10
371.266667	11	53	6	6
88.6	12	4	16	14
1034.09474	13	88	19	11
386.166667	14	48	15	7

Figure 2.1. Sample sheet ***Data*** *for 4 independent variables in file data2.xlsx.*

Figure 2.2 shows a sample sheet **Transf**. Based on the data in that sheet, the set of models examined have four single-variables with X_1, X_2, X_3, and X_4. Each of these variables has the power of 1 (i.e. no transformations). The model has two cross-product terms--X_i^{px1} with X_j^{px2} and X_m^{px3} with X_n^{px4}. The powers px1, px2, px3, and px4 are either 1 or 2 (since their ranges are (1,2)). The indices i, j, m, and n for the variables are all in the range 1 to 4. The MATLAB code skips regression calculations if a cross-product term (a) has the same two variables (i.e. i=j and/or m=n), and (b) the two cross-product terms refer to the same variables raised to the same powers. These two tests are extended in the case of three cross-product terms to detect duplications.



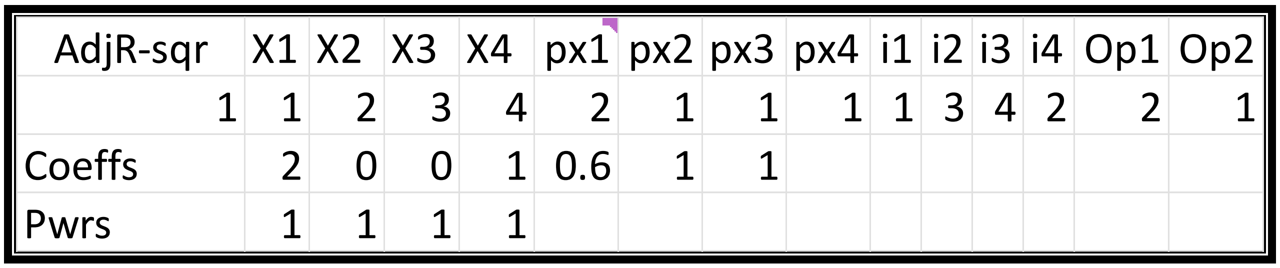
	px1	px2	px3	px4	i1	i2	i3	i4	Op1	Op2
Lb	1	1	1	1	1	1	1	1	1	1
Ub	2	2	2	2	4	4	4	4	2	2
Num Indepnt Vari	4	1	2	3	4					
Powers of Xi		1	1	1	1					

*Figure 2.2. Sample sheet **Transf** for 4 independent variables with two cross-product terms in file data2.xlsx.*

The Sheet **Results**

The sheet **Results** is one of the output sheets. Figure (2.3) shows a sample sheet for two cross-product terms. The sheet has the following rows:

1. Row has the headers that **you need to enter**!
2. Row 2 has the following groups of values:
   a. Cell A2 has the adjusted R0square values.
   b. Cells B2 to E2 contain the indices for the single-variable terms. This range varies with the number of independent variables that are used in the single-variable terms.
   c. Cells F2 to I2 is the set of four powers used in the two cross-product terms (each terms uses two variables, each raised to a power).
   d. Cells J2 to M2 is a set of four indices used to select the variables in the two cross-product terms.
   e. Cells N2 and O2 store the integer-code used to select whether to multiply or divide the variables in the cross-product terms.
3. Row 3 contains the regression coefficients:
   a. Cell B3 contains the intercept.
   b. Cells C3 to E3 contain the coefficients of the single-variable terms.
   c. Cells G3 and H3 contain the coefficients of the cross-product terms.
4. Row 4 has the user-selected powers of the single-variable terms in cells B4 to E4. The values are copied from the sheet **Transf**.

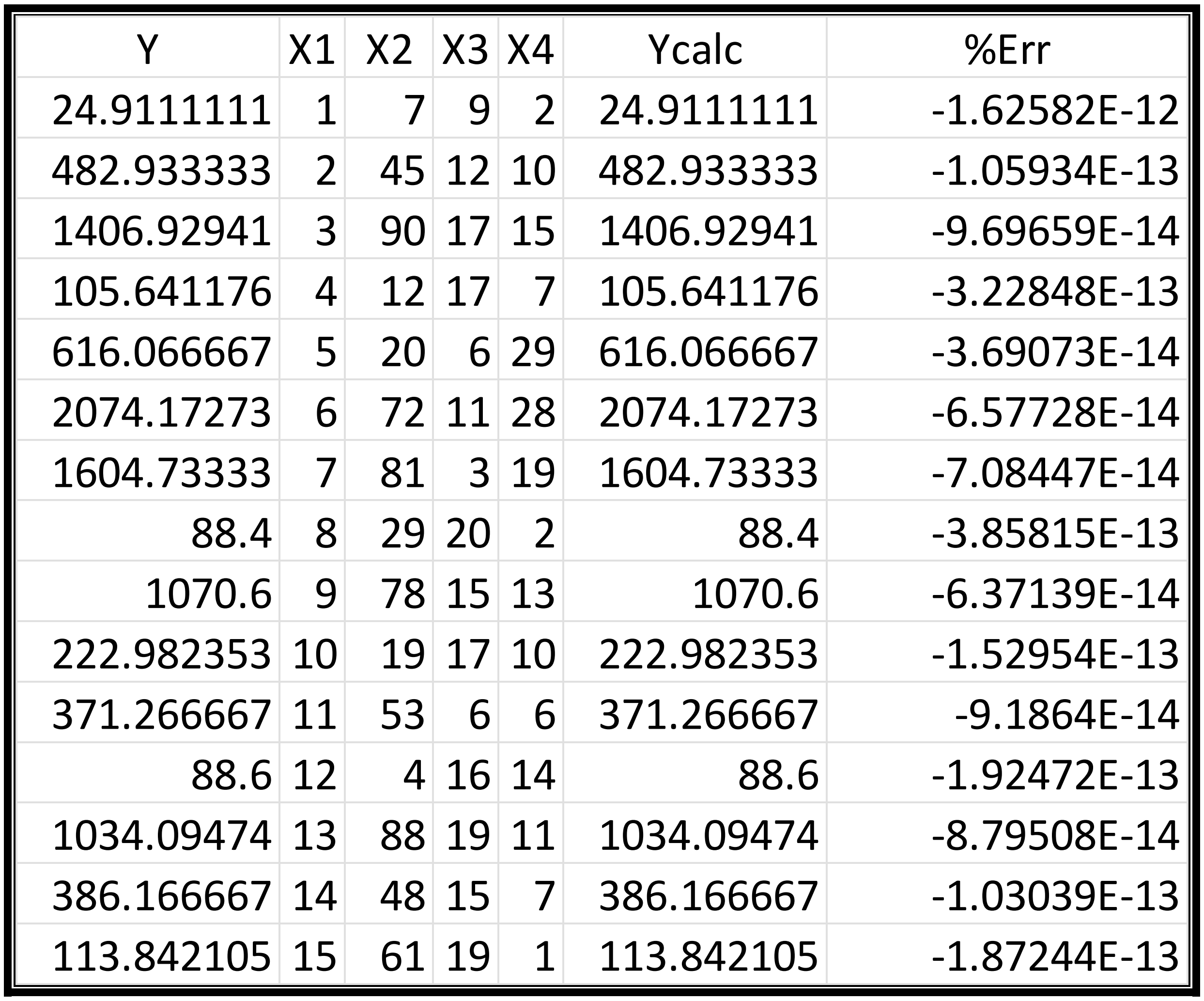


AdjR-sqr	X1	X2	X3	X4	px1	px2	px3	px4	i1	i2	i3	i4	Op1	Op2
1	1	2	3	4	2	1	1	1	1	3	4	2	2	1
Coeffs	2	0	0	1	0.6	1	1							
Pwrs	1	1	1	1										

*Figure 2.3. Sample sheet **Results** for 4 independent variables with two cross-product terms in file data2.xlsx.*

The Sheet **Project**

The sheet Project copies the input data from the sheet Data and inserts two new columns. The first one is for the calculated values of the dependent variable using the best model. The second new column is the percentage error in calculating the values of the dependent variable. Figure (2.4) shows a partial sample view of sheet **Project**.



Y	X1	X2	X3	X4	Ycalc	%Err
24.9111111	1	7	9	2	24.9111111	-1.62582E-12
482.933333	2	45	12	10	482.933333	-1.05934E-13
1406.92941	3	90	17	15	1406.92941	-9.69659E-14
105.641176	4	12	17	7	105.641176	-3.22848E-13
616.066667	5	20	6	29	616.066667	-3.69073E-14
2074.17273	6	72	11	28	2074.17273	-6.57728E-14
1604.73333	7	81	3	19	1604.73333	-7.08447E-14
88.4	8	29	20	2	88.4	-3.85815E-13
1070.6	9	78	15	13	1070.6	-6.37139E-14
222.982353	10	19	17	10	222.982353	-1.52954E-13
371.266667	11	53	6	6	371.266667	-9.1864E-14
88.6	12	4	16	14	88.6	-1.92472E-13
1034.09474	13	88	19	11	1034.09474	-8.79508E-14
386.166667	14	48	15	7	386.166667	-1.03039E-13
113.842105	15	61	19	1	113.842105	-1.87244E-13

*Figure 2.4. Sample sheet **Project** for 4 independent variables with two cross-product terms in file data2.xlsx.*

3/Optimizing Models with Single Cross-Product Terms

Let's start with the simplest models with cross-products—ones with one cross-product terms. Here are examples of such models:

$$Y = a + b_1*X_1^{px1} + b_2*X_2^{px2} + b_3*X_1^{px3}*X_2^{px4} \quad (10)$$
$$Y = a + b_1*X_1^{px1} + b_2*X_2^{px2} + b_3*X_1^{px3}/X_3^{px4} \quad (11)$$
$$Y = a + b_1*X_1^{px1} + b_2*X_1^{px2}*X_2^{px3} \quad (12)$$
$$Y = a + b_1*X_2^{px1} + b_2*X_1^{px2}*X_2^{px3} \quad (13)$$

The above examples indicate that the model must have at least one single-variable term and one cross-product term. The number of single-variable terms can exceed two and may also include other independent variables that do not appear in the cross-product terms. Here are some examples:

$$Y = a + b_1*X_1^{px1} + b_2*X_3^{px2} + b_3*X_1^{px3}*X_2^{px4} \quad (14)$$
$$Y = a + b_1*X_3^{px1} + b_2*X_4^{px2} + b_3*X_1^{px3}*X_2^{px4} \quad (15)$$

Equation (14) shows two single-variable terms, one of which also appears in the cross-product term. The variable X_3 does not! Equation (15) shows that the two independent variables in the single-variable terms do not appear in the cross-product terms. This scheme gives you more freedom in building the empirical models with cross-product terms,

The optimization process relies on the MATLAB function pso() that implements a special version of the particle swarm optimization. This version handles optimized variables that are strictly integers! Here is the listing for function pso():

```
function [gBestX,gBestCost,bestMdl] =
pso(fx,Lb,Ub,MaxPop,MaxIters,itersToReset)
% PSO implements particle swarm optimization.
%
% INPUT
% ======
% fx - handle of optimized function.
% Lb - array of low bound values.
% Ub - array of upper bound values.
% MaxPop - maximum population of swarm.
% MaxIters - maximum number of iterations
% itersToReset - number of stalled iterations that lead to data reset.
%
% OUTPUT
% ======
% gBestX - array of best solutions.
% gBestCost - best optimized function value.
% bestMdl – best regression model object
%
```

```
% %
  global mdl

  nVar = length(Lb);
  template.position = zeros(1,nVar);
  template.velocity = zeros(1,nVar);
  template.cost =  0;
  template.best_position = zeros(1,nVar);
  template.best_cost = 1e+99;

  c1 = 1.5;
  c2 = 2;
  w = 1;
  wdamp = 0.995;
  gBestCost = 1;
  gBestX = zeros(1,nVar);

  % Initizialize population
  for i=1:MaxPop
    pop(i) = template;
    for j=1:nVar
      pop(i).position(j) = fix(Lb(j) + (Ub(j) - Lb(j))*rand);
      pop(i).velocity(j) = 0;
    end

    pop(i).cost = fx(pop(i).position);

    if pop(i).cost < pop(i).best_cost
      pop(i).best_cost = pop(i).cost;
      pop(i).best_position = pop(i).position;

    end

    if pop(i).best_cost < gBestCost
      bestMdl = mdl;
      gBestCost = pop(i).best_cost;
      gBestX = pop(i).best_position;
    end
  end

  fprintf('Best Fx = %e, ', gBestCost);
  fprintf('Best X =[');
  fprintf('%d, ', gBestX(1:nVar-1));
  fprintf(' %d]\n', gBestX(nVar));
  lastIter = 0;
  % pso loop
  for iter = 1:MaxIters
    if gBestCost == 0,return; end
    if (iter - lastIter) >= itersToReset
      fprintf("----------------------- Reset population ---------------------
-- \n")
      fprintf("AT iteration %d\n", iter);
      lastIter = iter;
      for i=1:MaxPop
        pop(i) = template;
        for j=1:nVar
          pop(i).position(j) = fix(Lb(j) + (Ub(j) - Lb(j))*rand);
          pop(i).velocity(j) = 0;
        end

        pop(i).cost = fx(pop(i).position);

        if pop(i).cost < pop(i).best_cost
          pop(i).best_cost = pop(i).cost;
          pop(i).best_position = pop(i).position;
        end

        if pop(i).best_cost < gBestCost
          lastIter = iter;
          bestMdl = mdl;
          gBestCost = pop(i).best_cost;
          gBestX = pop(i).best_position;
          fprintf('Iter = %i, ', iter);
          fprintf('Best Fx = %e, ', gBestCost);
          fprintf('Best X = [');
          fprintf('%d, ', gBestX(1:nVar-1));
          fprintf(' %d]\n', gBestX(nVar));
        end
      end
    end

    for i = 1:MaxPop
      for j = 1:nVar
        pop(i).velocity(j) = w * pop(i).velocity(j) + ...
          c1 * rand * (pop(i).best_position(j) - pop(i).position(j)) + ...
          c2 * rand * (gBestX(j) - pop(i).position(j));
        pop(i).position(j) = fix(pop(i).position(j) + pop(i).velocity(j));
        if pop(i).position(j) < Lb(j) || pop(i).position(j) > Ub(j)
          pop(i).position(j) = fix(Lb(j) + (Ub(j) - Lb(j)) * rand);
        end
      end

      pop(i).cost = fx(pop(i).position);

      if pop(i).cost < pop(i).best_cost
        pop(i).best_cost = pop(i).cost;
      end

      if pop(i).best_cost < gBestCost
        lastIter = iter;
        bestMdl = mdl;
        gBestCost = pop(i).best_cost;
        gBestX = pop(i).best_position;
        fprintf('Iter = %i, ', iter);
        fprintf('Best Fx = %e, ', gBestCost);
        fprintf('Best X = [');
        fprintf('%d, ', gBestX(1:nVar-1));
        fprintf(' %d]\n', gBestX(nVar));
      end
    end
    w = w * wdamp;
  end
end
```

The function pso() has the following input parameters:

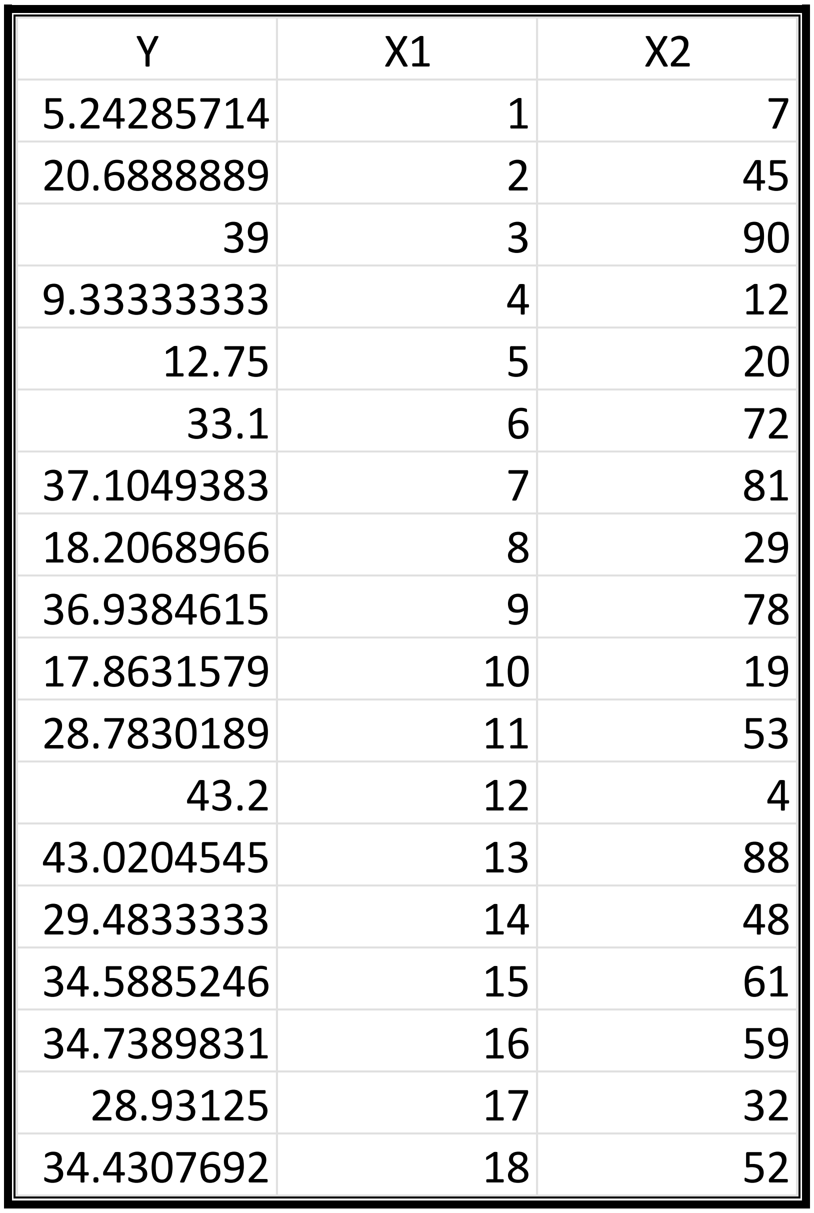
- Parameter fx is the handle of optimized function.
- Parameter Lb is the array of low bound values.
- Parameter Ub is the array of upper bound values.
- Parameter MaxPop is the maximum population of swarm.
- Parameter MaxIters is the maximum number of iterations
- Parameter itersToReset is the number of stalled iterations that lead to data reset.

The function has the following output parameters:

- Parameter gBestX is the array of best solutions.
- Parameter gBestCost is the best optimized function value.
- Parameter bestMdl is the best regression model object

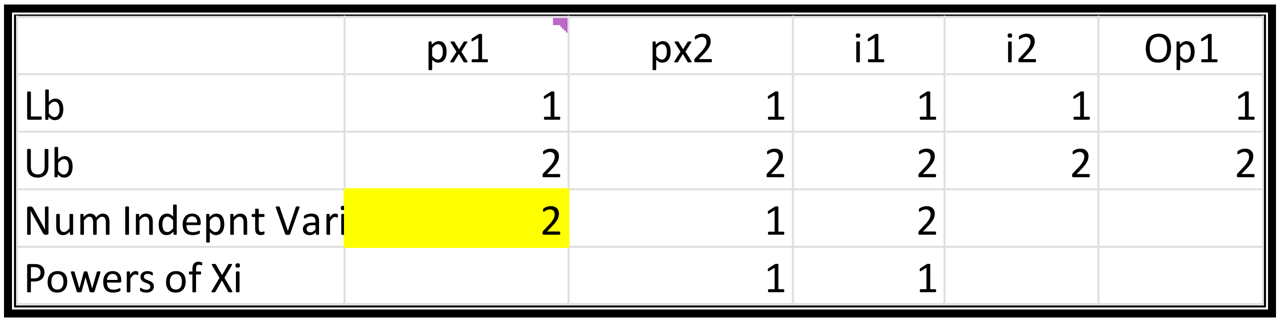
Notice that the special implementation of function pso() uses the MATLAB function fix() to extract integer values from floating-point expressions.

Figures (3.1), (3.2), (3.3), and (3.4) show the Excel sheets **Data**, **Transf**, **Results**, and **Project** (partial view), respectively. These sheets are in file data1.xlsx.



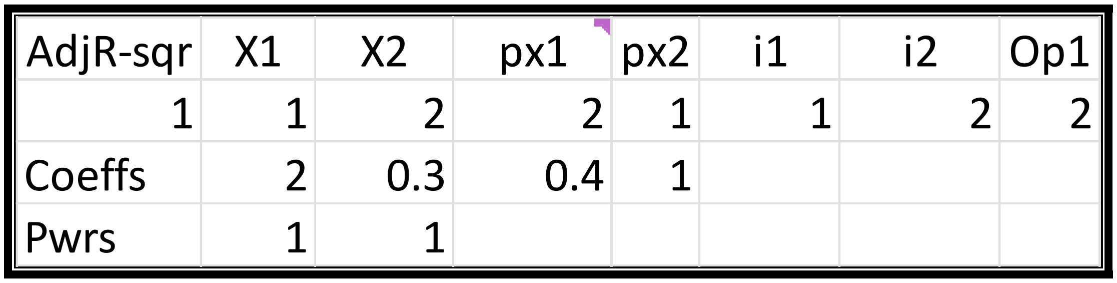
Y	X1	X2
5.24285714	1	7
20.6888889	2	45
39	3	90
9.33333333	4	12
12.75	5	20
33.1	6	72
37.1049383	7	81
18.2068966	8	29
36.9384615	9	78
17.8631579	10	19
28.7830189	11	53
43.2	12	4
43.0204545	13	88
29.4833333	14	48
34.5885246	15	61
34.7389831	16	59
28.93125	17	32
34.4307692	18	52

Figure 3.1. The partial view of sheet ***Data*** *for the single cross-product example, in file data1.xlsx.*



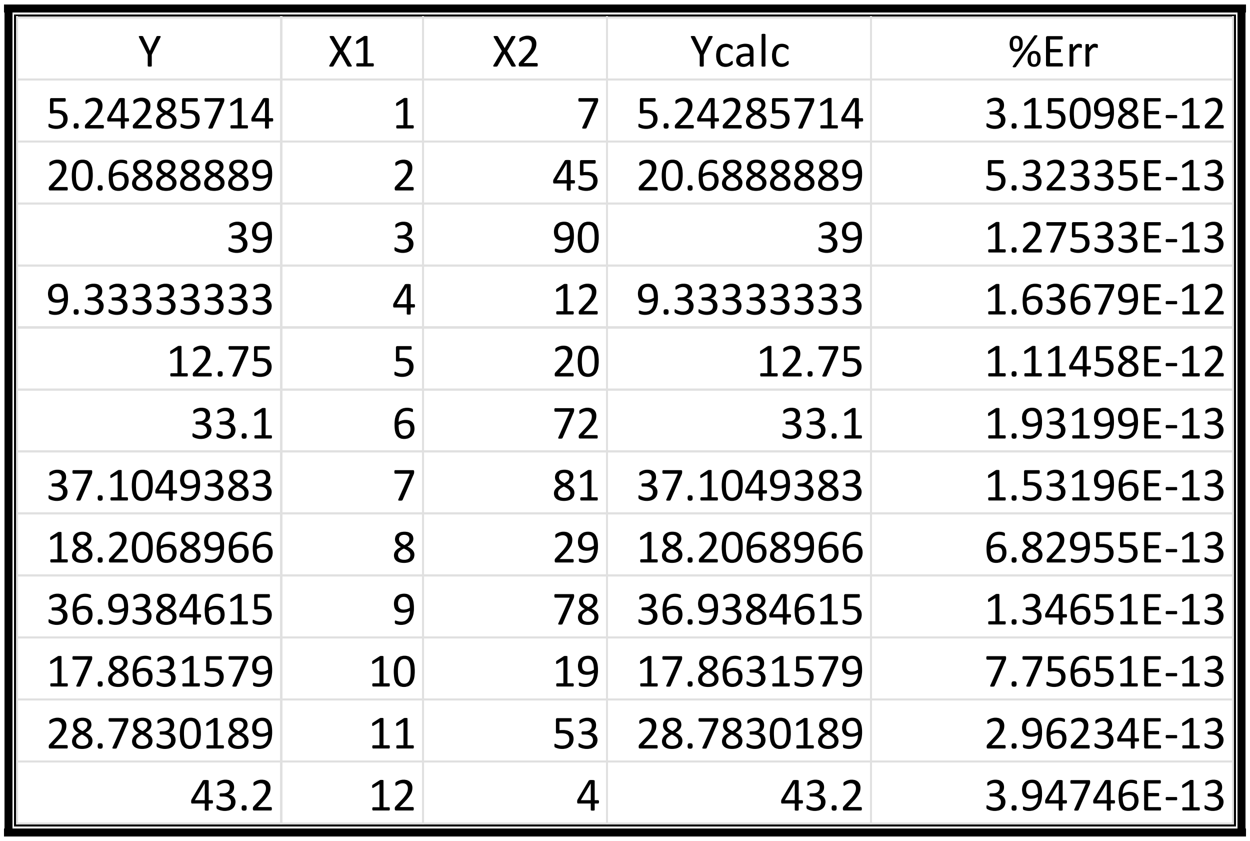
	px1	px2	i1	i2	Op1
Lb	1	1	1	1	1
Ub	2	2	2	2	2
Num Indepnt Vari	2	1	2		
Powers of Xi		1	1		

Figure 3.1. The sheet ***Transf*** *for the single cross-product example, in file data1.xlsx.*



AdjR-sqr	X1	X2	px1	px2	i1	i2	Op1
1	1	2	2	1	1	2	2
Coeffs	2	0.3	0.4	1			
Pwrs	1	1					

*Figure 3.1. The sheet **Results** for the single cross-product example, in file data1.xlsx.*



Y	X1	X2	Ycalc	%Err
5.24285714	1	7	5.24285714	3.15098E-12
20.6888889	2	45	20.6888889	5.32335E-13
39	3	90	39	1.27533E-13
9.33333333	4	12	9.33333333	1.63679E-12
12.75	5	20	12.75	1.11458E-12
33.1	6	72	33.1	1.93199E-13
37.1049383	7	81	37.1049383	1.53196E-13
18.2068966	8	29	18.2068966	6.82955E-13
36.9384615	9	78	36.9384615	1.34651E-13
17.8631579	10	19	17.8631579	7.75651E-13
28.7830189	11	53	28.7830189	2.96234E-13
43.2	12	4	43.2	3.94746E-13

*Figure 3.1. The partial view of sheet **Project** for the single cross-product example, in file data1.xlsx.*

File clem1.m has the program that calculates the optimum parameters for a single cross-product model with two single-term variables. The code for this program is:

```
clc
close all
clear all

global mdl
global xdata
global ydata
global idxX
global pwrs

warning("off")
dt = datetime('now','TimeZone','local','Format','y-MMM-d HH_mm_ss');
dtstr = string(dt);
filename = strcat(pwd,'\data1.xlsx');
delete("data1_*.txt");
outFile = strcat("data1_",dtstr,".txt");
diary(outFile)

fprintf("Program  clem1\n\n");
fprintf("%s\n", dtstr);
fprintf("Please wait ...\n\n");
fprintf("Excel file used is %s\n", filename);
fprintf("Diary file used is %s\n", outFile);
xyData = readmatrix(filename,'Sheet','Data');
[maxrows,~] = size(xyData);
ydata = xyData(:,1);
xdata = xyData(:,2:end);
xyTransf = readmatrix(filename,'Sheet','Transf');
xyTransf = xyTransf(:,2:end);
[n,nXVars] = size(xdata);
numIndpX = xyTransf(3,1);
idxX = zeros(1,numIndpX);
pwrs = zeros(1,numIndpX);
for i=1:numIndpX
  idxX(i) = xyTransf(3,1+i);
  pwrs(i) = xyTransf(4,1+i);
  if isnan(pwrs(i)), pwrs(i) = 1; end
end

Lb = xyTransf(1,:);
Ub = 1+xyTransf(2,:);
nVars = length(Ub);
MaxIters = 2000;
MaxPop= 250;
itersToReset = 75;
[bestX,gBestCost,bestMdl] = pso(@myFit,Lb,Ub,MaxPop,MaxIters,itersToReset);
mdl = bestMdl;
AdjR2 = mdl.Rsquared.Adjusted;
fprintf("Regression model\n")
fprintf("Adjusted R-square = %14.10f\n", AdjR2);
format long
mdl
bx = bestX;
bestX = [AdjR2];
for i = 1:numIndpX
    bestX = [bestX idxX(i)];
end
bestX = [bestX bx];
k =length(bestX);
res = zeros(3,k);
res(1,:) = bestX;
k2 = 1+numIndpX;
res(2,2:k2+2) = mdl.Coefficients{:,1}';
res(3,3:2+numIndpX) = pwrs;
writematrix(res(1,:), filename, 'Sheet', 'Results', 'Range', 'A2:Z2');
writematrix(res(2,2:k2+1), filename, 'Sheet', 'Results', 'Range', 'B3:Z3');
writematrix(res(3,3:2+numIndpX), filename, 'Sheet', 'Results', 'Range',
'B4:Z4');
%
% now do the % projections
ycalc = project(res);
percErr = 100.*(ycalc-ydata)./ydata;
ProjMat = [ydata xdata ycalc percErr];
writematrix(ProjMat, filename, 'Sheet', 'Project', 'Range', strcat("A2:Z",
num2str(1+length(ydata))));

i =2+numIndpX;
j=i+numIndpX;
sJ = num2str(bestX(j));
if bestX(i) == 0
  sx1 = strcat("ln(X", sJ,")");
elseif bestX(i) == 1
  sx1 = strcat("X", sJ);
elseif bestX(i) == 4
  sx1 = strcat("sqrt(X", sJ,")");
elseif bestX(i) > 1
  sx1 = strcat("X", sJ,"^", num2str(bestX(i)));
end
ix1 = sJ;
k2=1+numIndpX+2;
if bestX(k2+1) == 1
  sOp1 = "*";
else
  sOp1 = "/";
end
if bestX(k2+2) == 1
  sOp2 = "*";
else
  sOp2 = "/";
end

i=i+1;
j=j+1;
sJ = num2str(bestX(j));
if bestX(i) == 1
  sx2 = strcat("X", sJ);
elseif bestX(i) == 0
  sx2 = strcat("ln(X", sJ,")");
elseif bestX(i) == 4
  sx2 = strcat("sqrt(X", sJ,")");
elseif bestX(i) > 1
  sx2 = strcat("X", sJ,"^", num2str(bestX(i)));
end
ix2 = sJ;
s = strcat("Y = (", num2str(mdl.Coefficients{1,1}),")");
fprintf("\n\nBest model is:\n%s\n", s)
for i=1:numIndpX
  j = idxX(i);
  sJ= num2str(j);
  if pwrs(i)==1
    s = strcat("   + (", num2str(mdl.Coefficients{1+i,1}),")*X", sJ);
  elseif pwrs(i)==0
```

```
    s = strcat("   + (", num2str(mdl.Coefficients{1+i,1}),")*ln(X", sJ,")");
  else
    sp = num2str(pwrs(i));
    s = strcat("   + (", num2str(mdl.Coefficients{1+i,1}),")*X", sJ,"^",sp);
  end
  fprintf("%s\n", s)
end
s = strcat("   + (",num2str(mdl.Coefficients{2+numIndpX,1}), ")*", sx1, sOp1,
sx2);
fprintf("%s\n", s)


format short
fprintf("Done!\n");
diary off
beep on
for i=1:3
  beep;
  pause(1);
end

function x = myfx(x,p)
  if p==0
    x = log(x);
  elseif p==2
    x = x.^2;
  elseif p==3
    x = x.^3;
  elseif p==4
    x = sqrt(x);
  end
end

function r = myFit(coeff)
  global mdl
  global xdata
  global ydata
  global idxX
  global pwrs

  numIndpX = length(idxX);
  y = ydata;
  px1 = coeff(1);
  px2 = coeff(2);
  i1 = coeff(3);
  i2 = coeff(4);
  op1 = coeff(5);

  if i1==i2
    r = 1;
    return;
  end

  x1 = myfx(xdata(:,i1),px1);
  x2 = myfx(xdata(:,i2),px2);
  if op1==1
    x12 = x1.*x2;
  else
    x12 = x1./x2;
  end
  X = [];
  for i=1:numIndpX
    X = [X myfx(xdata(:,idxX(i)),pwrs(i))];
  end
  X = [X x12];
  mdl = fitlm(X,y);
  r = 1 - mdl.Rsquared.Adjusted;
end

function ycalc = project(res)
  global xdata
  global idxX
  global pwrs

  numIndpX = length(idxX);
  shift = numIndpX;
  px1 = res(1,2+shift);
  px2 = res(1,3+shift);
  i1 = res(1,4+shift);
  i2 = res(1,5+shift);
  op1 = res(1,6+shift);

  if px1 == 0
    x1= log(xdata(:,i1));
  elseif px1 == 1
    x1 = xdata(:,i1);
  elseif px1 == 2
    x1 = xdata(:,i1).^2;
  elseif px1 == 3
    x1 = xdata(:,i1).^3;
  elseif px1 == 4
    x1 = sqrt(xdata(:,i1));
  end

  if px2 == 0
    x2= log(xdata(:,i2));
  elseif px2 == 1
    x2 = xdata(:,i2);
  elseif px2 == 2
    x2 = xdata(:,i2).^2;
  elseif px2 == 3
    x2 = xdata(:,i2).^3;
  elseif px2 == 4
    x2 = sqrt(xdata(:,i2));
  end

  ycalc = res(2,2);
  for i=1:numIndpX
    ycalc = ycalc + res(2,2+i) * myfx(xdata(:,idxX(i)),pwrs(i));
  end
  k = 1 + numIndpX;
  if op1==1
    ycalc = ycalc + res(2,k+2) * x1.*x2;
  else
```

```
    ycalc = ycalc + res(2,k+2) * x1./x2;
  end
end
```

Program clem1 finds the best cross-product model using optimization. The program calls function pso() and supplies the needed parameters. Notice that the program adds 1 to each element of the array of upper ranges values. This step is needed because function pso() uses the MATLAB function fix(). The program has three helper functions (used in all the programs in this study):

- The function fx() transforms the values of the array x using the integer code in parameter p.
- The function myFit() performs the regression calculations and returns 1-AdjustedR-Square value. The function pso() minimizes the result of function myFit(). The function uses variable op1 to decide whether to multiply or divide the variables in a cross-product term.
- The function project() calculates the projected values of the dependent variable based on the best model. The function uses variable op1 to decide whether to multiply or divide the variables in a cross-product term.

☛ All the programs in this study rearrange the contents of the array bestX. The program first copies the array bestX into the copy array bx. Then, the programs reinitializes array bestX by placing the adjusted R-square value as the new first element in the array bestX. Next, the programs append the indices of the single-variable terms. Finally, the programs append the values in array bx. The programs use this approach whether they are optimizing the model parameters or searching for the best model using virtual nested loops. The programs write array bestX in the second row of the sheet **Results** in the various Excel files used.

The console output from running program clem1 is shown below:

```
Program  clem1

2024-Dec-30 02_26_40
Please wait ...

Excel file used is C:\Users\nsham\Dropbox\MATLAB\Clement3\data1.xlsx
Diary file used is data1_2024-Dec-30 02_26_40.txt
Best Fx = 0.000000e+00, Best X =[2.000000, 1.000000, 1.000000, 2.000000, 
2.000000]
Regression model
Adjusted R-square =  1.0000000000

mdl =


Linear regression model:
    y ~ 1 + x1 + x2 + x3

Estimated Coefficients:
                       Estimate          SE    tStat    pValue
                   _________________     __    _____    ______

    (Intercept)      2.00000000000018    0      Inf       0   
    x1             0.299999999999999     0      Inf       0   
    x2             0.399999999999999     0      Inf       0   
    x3                             1     0      Inf       0   


Number of observations: 100, Error degrees of freedom: 96
R-squared: 1,  Adjusted R-Squared: 1
F-statistic vs. constant model: 6.13e+31, p-value = 0


Best model is:
Y = (2)
   + (0.3)*X1
   + (0.4)*X2
   + (1)*X1^2*X2
Done!
```

The above output shows that the optimization process succeeded in finding the model used to create the values of the dependent variable. Figures (3.3) and (3.4) show the output as it appears in sheets **Results** and **Project** of the Excel file data1.xlsx.

4/Using Virtual Nested Loop to Find the Best Models with Single Cross-Product Terms

This section uses the virtual nested loops to find the best model with a single cross-product term. The Excel input and output file is the same as in the last section. The program clem1b determines the best model. Here is the listing for clem1b:

```
clc
close all
clear all

global mdl
global xdata
global ydata
global idxX
global pwrs

warning("off")
dt = datetime('now','TimeZone','local','Format','y-MMM-d HH_mm_ss');
```

```
dtstr = string(dt);
filename = strcat(pwd,'\data1.xlsx');
delete("data1b_*.txt");
outFile = strcat("data1b_",dtstr,".txt");
diary(outFile)

fprintf("Program  clem1b\n\n");
fprintf("%s\n", dtstr);
fprintf("Please wait ...\n\n");
fprintf("Excel file used is %s\n", filename);
fprintf("Diary file used is %s\n", outFile);
xyData = readmatrix(filename,'Sheet','Data');
[maxrows,~] = size(xyData);
ydata = xyData(:,1);
xdata = xyData(:,2:end);
xyTransf = readmatrix(filename,'Sheet','Transf');
xyTransf = xyTransf(:,2:end);
[n,nXVars] = size(xdata);
numIndpX = xyTransf(3,1);
idxX = zeros(1,numIndpX);
pwrs = zeros(1,numIndpX);
for i=1:numIndpX
  idxX(i) = xyTransf(3,1+i);
  pwrs(i) = xyTransf(4,1+i);
  if isnan(pwrs(i)), pwrs(i) = 1; end
end

Lb = xyTransf(1,:);
Ub = xyTransf(2,:);
nVars = length(Ub);
iFrom = Lb;
iTo = Ub;
iStep = 1 + zeros(1,nVars);
idx= Lb;
bStop = false;
iter = 0;
fprintf("------------- Calculating Max Iters ----------------\n");
while ~bStop
  iter = iter +1;
  % Start implementing the quasi-nested loops
  idx(1) = idx(1) + iStep(1);
  if idx(1) > iTo(1)
    idx(1) = iFrom(1);
    for i = 2:nVars
      idx(i) = idx(i) + iStep(i);
      if idx(i) > iTo(i) && i < nVars
        idx(i) = iFrom(i);
      elseif idx(i) > iTo(i) && i == nVars
        bStop = true;
      else
        break
      end
    end
  end
end
str = num2str(iter);
str = fliplr(regexprep(fliplr(str), '(\d+\.)?(\d{3})(?=\S+)', '$1$2,'));
```

```
fprintf("Max iters = %s\n", str);
MaxIters = iter;
FivePercent = fix(iter/20);

iFrom = Lb;
iTo = Ub;
iStep = 1 + zeros(1,nVars);
idx= Lb;
bestX = zeros(1,nVars);
bestFx = 0;
% --------------------------------- start main loop
bStop = false;
iter = 0;
fprintf("------------- Maximizing Fx ----------------\n");
while ~bStop
  iter = iter +1;
  if mod(iter, FivePercent) == 0
    percent = 100*iter/MaxIters;
    fprintf("Progress %5.2f%%\n", percent);
  end

  r2adj = myFit(idx);
  if r2adj > bestFx
    bestFx = r2adj;
    bestX = idx;
    bestMdl = mdl;
    fprintf("Iter %d, Fx = %e, X=[", iter, bestFx)
    fprintf("%d, ", bestX);
    fprintf("]\n");
  end
  % Start implementing the quasi-nested loops
  idx(1) = idx(1) + iStep(1);
  if idx(1) > iTo(1)
    idx(1) = iFrom(1);
    for i = 2:nVars
      idx(i) = idx(i) + iStep(i);
      if idx(i) > iTo(i) && i < nVars
        idx(i) = iFrom(i);
      elseif idx(i) > iTo(i) && i == nVars
        bStop = true;
      else
        break
      end
    end
  end
end
fprintf("Regression model\n")
format long
AdjR2 = bestMdl.Rsquared.Adjusted;
fprintf("Adjusted R-square = %14.12f\n", AdjR2);
mdl = bestMdl
bx = bestX;
bestX = [AdjR2];
for i = 1:numIndpX
    bestX = [bestX idxX(i)];
end
bestX = [bestX bx];
```

```
k =length(bestX);
res = zeros(3,k);
res(1,:) = bestX;
k2 = 1+numIndpX;
res(2,2:k2+2) = mdl.Coefficients{:,1}';
res(3,3:2+numIndpX) = pwrs;
writematrix(res(1,:), filename, 'Sheet', 'Results', 'Range', 'A2:Z2');
writematrix(res(2,2:k2+1), filename, 'Sheet', 'Results', 'Range', 'B3:Z3');
writematrix(res(3,3:2+numIndpX), filename, 'Sheet', 'Results', 'Range',
'B4:Z4');
%
% now do the % projections
ycalc = project(res);
percErr = 100.*(ycalc-ydata)./ydata;
ProjMat = [ydata xdata ycalc percErr];
writematrix(ProjMat, filename, 'Sheet', 'Project', 'Range', strcat("A2:Z",
num2str(1+length(ydata))));

i =2+numIndpX;
j=i+numIndpX;
sJ = num2str(bestX(j));
if bestX(i) == 0
  sx1 = strcat("ln(X", sJ,")");
elseif bestX(i) == 1
  sx1 = strcat("X", sJ);
elseif bestX(i) == 4
  sx1 = strcat("sqrt(X", sJ,")");
elseif bestX(i) > 1
  sx1 = strcat("X", sJ,"^", num2str(bestX(i)));
end
ix1 = sJ;
k2=1+numIndpX+2;
if bestX(k2+1) == 1
  sOp1 = "*";
else
  sOp1 = "/";
end
if bestX(k2+2) == 1
  sOp2 = "*";
else
  sOp2 = "/";
end

i=i+1;
j=j+1;
sJ = num2str(bestX(j));
if bestX(i) == 1
  sx2 = strcat("X", sJ);
elseif bestX(i) == 0
  sx2 = strcat("ln(X", sJ,")");
elseif bestX(i) == 4
  sx2 = strcat("sqrt(X", sJ,")");
elseif bestX(i) > 1
  sx2 = strcat("X", sJ,"^", num2str(bestX(i)));
end
ix2 = sJ;
s = strcat("Y = (", num2str(mdl.Coefficients{1,1}),")");
```

```
fprintf("\n\nBest model is:\n%s\n", s)
for i=1:numIndpX
  j = idxX(i);
  sJ= num2str(j);
  if pwrs(i)==1
    s = strcat("   + (", num2str(mdl.Coefficients{1+i,1}),")*X", sJ);
  elseif pwrs(i)==0
    s = strcat("   + (", num2str(mdl.Coefficients{1+i,1}),")*ln(X", sJ,")");
  else
    sp = num2str(pwrs(i));
    s = strcat("   + (", num2str(mdl.Coefficients{1+i,1}),")*X", sJ,"^",sp);
  end
  fprintf("%s\n", s)
end
s = strcat("   + (",num2str(mdl.Coefficients{2+numIndpX,1}), ")*", sx1, sOp1,
sx2);
fprintf("%s\n", s)


format short
fprintf("Done!\n");
diary off
beep on
for i=1:3
  beep;
  pause(1);
end

function x = myfx(x,p)
  if p==0
    x = log(x);
  elseif p==2
    x = x.^2;
  elseif p==3
    x = x.^3;
  elseif p==4
    x = sqrt(x);
  end
end

function r = myFit(coeff)
  global mdl
  global xdata
  global ydata
  global idxX
  global pwrs

  numIndpX = length(idxX);
  y = ydata;
  px1 = coeff(1);
  px2 = coeff(2);
  i1 = coeff(3);
  i2 = coeff(4);
  op1 = coeff(5);

  if i1==i2
    r = 0;
    return;
  end

  x1 = myfx(xdata(:,i1),px1);
  x2 = myfx(xdata(:,i2),px2);
  if op1==1
    x12 = x1.*x2;
  else
    x12 = x1./x2;
  end
  X = [];
  for i=1:numIndpX
    X = [X myfx(xdata(:,idxX(i)),pwrs(i))];
  end
  X = [X x12];
  mdl = fitlm(X,y);
  r = mdl.Rsquared.Adjusted;
end

function ycalc = project(res)
  global xdata
  global idxX
  global pwrs

  numIndpX = length(idxX);
  shift = numIndpX;
  px1 = res(1,2+shift);
  px2 = res(1,3+shift);
  i1 = res(1,4+shift);
  i2 = res(1,5+shift);
  op1 = res(1,6+shift);

  if px1 == 0
    x1= log(xdata(:,i1));
  elseif px1 == 1
    x1 = xdata(:,i1);
  elseif px1 == 2
    x1 = xdata(:,i1).^2;
  elseif px1 == 3
    x1 = xdata(:,i1).^3;
  elseif px1 == 4
    x1 = sqrt(xdata(:,i1));
  end

  if px2 == 0
    x2= log(xdata(:,i2));
  elseif px2 == 1
    x2 = xdata(:,i2);
  elseif px2 == 2
    x2 = xdata(:,i2).^2;
  elseif px2 == 3
    x2 = xdata(:,i2).^3;
  elseif px2 == 4
    x2 = sqrt(xdata(:,i2));
  end

  ycalc = res(2,2);
  for i=1:numIndpX
    ycalc = ycalc + res(2,2+i) * myfx(xdata(:,idxX(i)),pwrs(i));
  end
  k = 1 + numIndpX;
  if op1==1
    ycalc = ycalc + res(2,k+2) * x1.*x2;
  else
    ycalc = ycalc + res(2,k+2) * x1./x2;
  end
end
```

The above program uses the virtual nested loops (in red text) twice. The first time, these loops ⦿serve to calculate the maximum number of iterations needed to test all the combinations of powers and selections of variables. The second time, the nested loops work at finding the best model by varying the powers and indices of the variables. The loops use arrays iFrom, iTo, iStep, and idx to store the initial values for the various ranges, final values for the ranges, increment values for the ranges, and the current values, respectively. Like the previous program, clem1b uses the helper functions fx(), myFit(), and project().

The console output from running program clem1b is shown below:

```
Program  clem1b

2025-Jan-2 20_06_31
Please wait ...

Excel file used is I:\DropBox\Dropbox\MATLAB\Clement3\data1.xlsx
Diary file used is data1b_2025-Jan-2 20_06_31.txt
------------- Calculating Max Iters ----------------
Max iters = 32
------------- Maximizing Fx ----------------
Progress  3.12%
Progress  6.25%
Progress  9.38%
Progress 12.50%
Progress 15.62%
Iter 5, Fx = 3.584175e-01, X=[1, 1, 2, 1, 1, ]
Progress 18.75%
Progress 21.88%
Progress 25.00%
Progress 28.12%
Progress 31.25%
Progress 34.38%
Progress 37.50%
Progress 40.62%
Progress 43.75%
Progress 46.88%
Progress 50.00%
Progress 53.12%
Progress 56.25%
Progress 59.38%
Progress 62.50%
Progress 65.62%
Progress 68.75%
Progress 71.88%
Progress 75.00%
Progress 78.12%
Iter 25, Fx = 8.966799e-01, X=[1, 1, 1, 2, 2, ]
Progress 81.25%
Iter 26, Fx = 1.000000e+00, X=[2, 1, 1, 2, 2, ]
Progress 84.38%
Progress 87.50%
Progress 90.62%
Progress 93.75%
Progress 96.88%
Progress 100.00%
Regression model
Adjusted R-square = 1.000000000000

mdl = 


Linear regression model:
    y ~ 1 + x1 + x2 + x3

Estimated Coefficients:
                       Estimate          SE    tStat    pValue
                   _________________    __    _____    ______

    (Intercept)      2.00000000000018    0      Inf       0   
    x1              0.299999999999999    0      Inf       0   
    x2              0.399999999999999    0      Inf       0   
    x3                              1    0      Inf       0   


Number of observations: 100, Error degrees of freedom: 96
R-squared: 1,  Adjusted R-Squared: 1
F-statistic vs. constant model: 6.13e+31, p-value = 0


Best model is:
Y = (2)
   + (0.3)*X1
   + (0.4)*X2
   + (1)*X1^2*X2
Done!
```

The above results show that the nested loops find the best model. Figures (3.3) and (3.4) show the output as it appears in the sheets **Results** and **Project** of the Excel file data1.xlsx.

5/Optimizing Models with Two Cross-Product Terms

This section looks at models with two cross-product terms. The model used is:

$$Y = a + \sum_{i=1}^{4} b_i\, X_i^{px(i)} + \sum_{j,k}^{4} c_{jk}\, X_j^{qx(j)}\; op(j,k)\; X_k^{qx(k)} \qquad (16)$$

The above model has four single-variable terms and two cross-product terms.

Figure (2.1), (2.2), (2.3), and (2.4) show the sheets **Data**, **Transf**, **Results**, and **Project** that are in file data1.xlsx. The program clem2 determines the best model. Here is the listing:

```
clc
close all
clear all

global mdl
global xdata
global ydata
global idxX
 global pwrs

warning("off")
dt = datetime('now','TimeZone','local','Format','y-MMM-d HH_mm_ss');
dtstr = string(dt);
filename = strcat(pwd,'\data2.xlsx');
delete("data2_*.txt");
outFile = strcat("data2_",dtstr,".txt");
diary(outFile)

fprintf("Program  clem2\n\n");
fprintf("%s\n", dtstr);
fprintf("Please wait ...\n\n");
fprintf("Excel file used is %s\n", filename);
fprintf("Diary file used is %s\n", outFile);
xyData = readmatrix(filename,'Sheet','Data');
[maxrows,~] = size(xyData);
ydata = xyData(:,1);
xdata = xyData(:,2:end);
xyTransf = readmatrix(filename,'Sheet','Transf');
xyTransf = xyTransf(:,2:end);
[n,nXVars] = size(xdata);
numIndpX = xyTransf(3,1);
idxX = zeros(1,numIndpX);
pwrs = zeros(1,numIndpX);
for i=1:numIndpX
  idxX(i) = xyTransf(3,1+i);
  pwrs(i) = xyTransf(4,1+i);
  if isnan(pwrs(i)), pwrs(i) = 1; end
end

Lb = xyTransf(1,:);
Ub = 1+xyTransf(2,:);
nVars = length(Ub);
MaxIters = 3000;
MaxPop= 300;
itersToReset = 100;
[bestX,gBestCost,bestMdl] = pso(@myFit,Lb,Ub,MaxPop,MaxIters,itersToReset);
```

```
mdl = bestMdl;
AdjR2 = mdl.Rsquared.Adjusted;
fprintf("Regression model\n")
fprintf("Adjusted R-square = %14.10f\n", AdjR2);
format long
mdl
bx = bestX;
bestX = [AdjR2];
for i = 1:numIndpX
    bestX = [bestX idxX(i)];
end
bestX = [bestX bx];
k =length(bestX);
res = zeros(3,k);
res(1,:) = bestX;
k2 = 3+numIndpX;
res(2,2:k2+1) = mdl.Coefficients{:,1}';
res(3,3:2+numIndpX) = pwrs;
writematrix(res(1,:), filename, 'Sheet', 'Results', 'Range', 'A2:Z2');
writematrix(res(2,2:k2+1), filename, 'Sheet', 'Results', 'Range', 'B3:Z3');
writematrix(res(3,3:2+numIndpX), filename, 'Sheet', 'Results', 'Range',
'B4:Z4');
%
% now do the % projections
ycalc = project(res);
percErr = 100.*(ycalc-ydata)./ydata;
ProjMat = [ydata xdata ycalc percErr];
writematrix(ProjMat, filename, 'Sheet', 'Project', 'Range', strcat("A2:Z",
num2str(1+length(ydata))));

i=2+numIndpX;
j=i+numIndpX;
sJ = num2str(bestX(j));
if bestX(i) == 0
  sx1 = strcat("ln(X", sJ,")");
elseif bestX(i) == 1
  sx1 = strcat("X", sJ);
elseif bestX(i) == 4
  sx1 = strcat("sqrt(X", sJ,")");
elseif bestX(i) > 1
  sx1 = strcat("X", sJ,"^", num2str(bestX(i)));
end
ix1 = sJ;
k2=1+numIndpX+2;
if bestX(k2+1) == 1
  sOp1 = "*";
else
  sOp1 = "/";
end
if bestX(k2+2) == 1
  sOp2 = "*";
else
  sOp2 = "/";
end

i=i+1;
j=j+1;
```

```
sJ = num2str(bestX(j));
if bestX(i) == 1
  sx2 = strcat("X", sJ);
elseif bestX(i) == 0
  sx2 = strcat("ln(X", sJ,")");
elseif bestX(i) == 4
  sx2 = strcat("sqrt(X", sJ,")");
elseif bestX(i) > 1
  sx2 = strcat("X", sJ,"^", num2str(bestX(i)));
end
ix2 = sJ;

i=i+1;
j=j+1;
sJ = num2str(bestX(j));
if bestX(i) == 1
  sx3 = strcat("X", sJ);
elseif bestX(i) == 0
  sx3 = strcat("ln(X", sJ,")");
elseif bestX(i) == 4
  sx3 = strcat("sqrt(X", sJ,")");
elseif bestX(i) > 1
  sx3 = strcat("X", sJ,"^", num2str(bestX(i)));
end
ix3 = sJ;

i=i+1;
j=j+1;
sJ = num2str(bestX(j));
if bestX(i) == 1
  sx4 = strcat("X", sJ);
elseif bestX(i) == 0
  sx4 = strcat("ln(X", sJ,")");
elseif bestX(i) == 4
  sx4 = strcat("sqrt(X", sJ,")");
elseif bestX(i) > 1
  sx4 = strcat("X", sJ,"^", num2str(bestX(i)));
end
ix4 = sJ;

s = strcat("Y = (", num2str(mdl.Coefficients{1,1}),")");
fprintf("\n\nBest model is:\n%s\n", s)
for i=1:numIndpX
  j = idxX(i);
  sJ= num2str(j);
  if pwrs(i)==1
    s = strcat("   + (", num2str(mdl.Coefficients{1+i,1}),")*X", sJ);
  elseif pwrs(i)==0
    s = strcat("   + (", num2str(mdl.Coefficients{1+i,1}),")*ln(X", sJ,")");
  else
    sp = num2str(pwrs(i));
    s = strcat("   + (", num2str(mdl.Coefficients{1+i,1}),")*X", sJ,"^",sp);
  end
  fprintf("%s\n", s)
end
s = strcat("   + (",num2str(mdl.Coefficients{2+numIndpX,1}), ")*", sx1, sOp1,
sx2);
```

```
fprintf("%s\n", s)
s = strcat("   + (",num2str(mdl.Coefficients{2+numIndpX,1}), ")*", sx3, sOp2,
sx4);
fprintf("%s\n", s)


format short
fprintf("Done!\n");
diary off
beep on
for i=1:3
  beep;
  pause(1);
end

function x = myfx(x,p)
  if p==0
    x = log(x);
  elseif p==2
    x = x.^2;
  elseif p==3
    x = x.^3;
  elseif p==4
    x = sqrt(x);
  end
end

function r = myFit(coeff)
  global mdl
  global xdata
  global ydata
  global idxX
  global pwrs

  [~,nXVars] = size(xdata);
  numIndpX = length(idxX);
  y = ydata;
  px1 = coeff(1);
  px2 = coeff(2);
  px3 = coeff(3);
  px4 = coeff(4);
  i1 = coeff(5);
  i2 = coeff(6);
  i3= coeff(7);
  i4 = coeff(8);
  op1 = coeff(9);
  op2 = coeff(10);

  if i1 > nXVars || i2 > nXVars || i3 > nXVars || i4 > nXVars
    r = 1;
    return
  end
  if (i1==i2) || (i3==i4)
    r = 1;
    return;
  end
  % compare X1,X2 with X3,X4
  if ((i1==i3) && (px1==px3)) || ((i2==i4) && (px2==px4))
    r = 1;
    return;
  end
  if ((i1==i4) && (px1==px4)) || ((i2==i3) && (px2==px3))
    r = 1;
    return;
  end

  x1 = myfx(xdata(:,i1),px1);
  x2 = myfx(xdata(:,i2),px2);
  x3 = myfx(xdata(:,i3),px3);
  x4 = myfx(xdata(:,i4),px4);
  if op1==1
    x12 = x1.*x2;
  else
    x12 = x1./x2;
  end
  if op2==1
    x34 = x3.*x4;
  else
    x34  = x3./x4;
  end
  X = [];
  for i=1:numIndpX
    X = [X myfx(xdata(:,idxX(i)),pwrs(i))];
  end
  X = [X x12 x34];
  mdl = fitlm(X,y);
  r = 1 - mdl.Rsquared.Adjusted;
  if r==0
    zz=1;
  end
end

function ycalc = project(res)
  global xdata
  global idxX
  global pwrs

  numIndpX = length(idxX);
  shift = numIndpX;
  px1 = res(1,2+shift);
  px2 = res(1,3+shift);
  px3 = res(1,4+shift);
  px4 = res(1,5+shift);
  i1 = res(1,6+shift);
  i2 = res(1,7+shift);
  i3 = res(1,8+shift);
  i4 = res(1,9+shift);
  op1 = res(1,10+shift);
  op2 = res(1,11+shift);
  if px1 == 0
    x1= log(xdata(:,i1));
  elseif px1 == 1
    x1 = xdata(:,i1);
  elseif px1 == 2
```

```
    x1 = xdata(:,i1).^2;
  elseif px1 == 3
    x1 = xdata(:,i1).^3;
  elseif px1 == 4
    x1 = sqrt(xdata(:,i1));
  end

  if px2 == 0
    x2= log(xdata(:,i2));
  elseif px2 == 1
    x2 = xdata(:,i2);
  elseif px2 == 2
    x2 = xdata(:,i2).^2;
  elseif px2 == 3
    x2 = xdata(:,i2).^3;
  elseif px2 == 4
    x2 = sqrt(xdata(:,i2));
  end

  if px3 == 0
    x3= log(xdata(:,i3));
  elseif px3 == 1
    x3 = xdata(:,i3);
  elseif px3 == 2
    x3 = xdata(:,i3).^2;
  elseif px3 == 3
    x3 = xdata(:,i3).^3;
  elseif px3 == 4
    x3 = sqrt(xdata(:,i3));
  end

  if px4 == 0
    x4= log(xdata(:,i4));
  elseif px4 == 1
    x4 = xdata(:,i4);
  elseif px4 == 2
    x4 = xdata(:,i4).^2;
  elseif px4 == 3
    x4 = xdata(:,i4).^3;
  elseif px4 == 4
    x4 = sqrt(xdata(:,i4));
  end

  ycalc = res(2,2);
  for i=1:numIndpX
    ycalc = ycalc + res(2,2+i) * myfx(xdata(:,idxX(i)),pwrs(i));
  end
  k = 1 + numIndpX;
  if op1==1
    ycalc = ycalc + res(2,k+2) * x1.*x2;
  else
    ycalc = ycalc + res(2,k+2) * x1./x2;
  end

  if op2==1
    ycalc = ycalc + res(2,k+3) * x3.*x4;
  else
```

```
    ycalc = ycalc + res(2,k+3) * x3./x4;
  end
end
```

Program clem2 is am expanded version of clem1. The additional code handles the two cross-product terms used in the model.

The console output from running program clem2 is shown below:

```
Program  clem2

2024-Dec-30 02_26_53
Please wait ...

Excel file used is C:\Users\nsham\Dropbox\MATLAB\Clement3\data2.xlsx
Diary file used is data2_2024-Dec-30 02_26_53.txt
Best Fx = 9.242867e-02, Best X =[1.000000, 1.000000, 1.000000, 1.000000,
2.000000, 4.000000, 1.000000, 3.000000, 2.000000,  2.000000]
Iter = 1, Best Fx = 2.851843e-02, Best X = [1, 2, 2, 1, 2, 4, 3, 2, 2,  1]
Iter = 4, Best Fx = 2.597043e-02, Best X = [2, 1, 2, 1, 2, 2, 2, 4, 1,  2]
Iter = 4, Best Fx = 0.000000e+00, Best X = [1, 1, 2, 1, 4, 3, 2, 2, 1,  2]
Regression model
Adjusted R-square = Adj Rsqr =   1.0000000000

mdl = 


Linear regression model:
    y ~ 1 + x1 + x2 + x3 + x4 + x5 + x6

Estimated Coefficients:
                       Estimate                 SE                  tStat          pValue
                   ________________    ___________________    ________________    ______

    (Intercept)     1.99999999999955    1.47491324763634e-05    135601.195745221      0
    x1             0.300000000000004    1.08591279299046e-07    2762652.78332199      0
    x2             0.400000000000003    2.11545763592126e-07    1890843.82125104      0
    x3                           0.5    4.95989239781328e-07    1008086.38554425      0
    x4             0.600000000000018    6.72073636871872e-07    892759.315471268      0
    x5             0.999999999999999    1.10726119279027e-08    90312927.6553099      0
    x6                             1    2.95655307659211e-09    338231709.052441      0


Number of observations: 100, Error degrees of freedom: 93
Root Mean Squared Error: 2.53e-05
R-squared: 1,  Adjusted R-Squared: 1
F-statistic vs. constant model: 1.75e+31, p-value = 0


Best model is:
Y = (2)
   + (0.3)*X1
   + (0.4)*X2
   + (0.5)*X3
   + (0.6)*X4
   + (1)*X4/X3
   + (1)*X2^2*X2
```

```
Done!
```

Program clem2 succeeds in finding the best model.

6/Using Virtual Nested Loop to Find the Best Models with Two Cross-Product Terms

This section uses the virtual nested loops to find the best model with a single cross-product term. The Excel input and output file is the same as in the last section. The program clem2b determines the best model. Here is the listing for clem2b:

```
clc
close all
clear all

global mdl
global xdata
global ydata
global idxX
global pwrs

warning("off")
dt = datetime('now','TimeZone','local','Format','y-MMM-d HH_mm_ss');
dtstr = string(dt);
filename = strcat(pwd,'\data2.xlsx');
delete("data2b_*.txt");
outFile = strcat("data2b_",dtstr,".txt");
diary(outFile)

fprintf("Program  clem2b\n\n");
fprintf("%s\n", dtstr);
fprintf("Please wait ...\n\n");
fprintf("Excel file used is %s\n", filename);
fprintf("Diary file used is %s\n", outFile);
xyData = readmatrix(filename,'Sheet','Data');
[maxrows,~] = size(xyData);
ydata = xyData(:,1);
xdata = xyData(:,2:end);
xyTransf = readmatrix(filename,'Sheet','Transf');
xyTransf = xyTransf(:,2:end);
[n,nXVars] = size(xdata);
numIndpX = xyTransf(3,1);
idxX = zeros(1,numIndpX);
pwrs = zeros(1,numIndpX);
for i=1:numIndpX
  idxX(i) = xyTransf(3,1+i);
  pwrs(i) = xyTransf(4,1+i);
  if isnan(pwrs(i)), pwrs(i) = 1; end
end

Lb = xyTransf(1,:);
Ub = xyTransf(2,:);
nVars = length(Ub);
```

```
iFrom = Lb;
iTo = Ub;
iStep = 1 + zeros(1,nVars);
idx= Lb;
bStop = false;
iter = 0;
fprintf("------------- Calculating Max Iters ----------------\n");
while ~bStop
  iter = iter +1;
  % Start implementing the quasi-nested loops
  idx(1) = idx(1) + iStep(1);
  if idx(1) > iTo(1)
    idx(1) = iFrom(1);
    for i = 2:nVars
      idx(i) = idx(i) + iStep(i);
      if idx(i) > iTo(i) && i < nVars
        idx(i) = iFrom(i);
      elseif idx(i) > iTo(i) && i == nVars
        bStop = true;
      else
        break
      end
    end
  end
end
str = num2str(iter);
str = fliplr(regexprep(fliplr(str), '(\d+\.)?(\d{3})(?=\S+)', '$1$2,'));
fprintf("Max iters = %s\n", str);
MaxIters = iter;
FivePercent = fix(iter/20);

iFrom = Lb;
iTo = Ub;
iStep = 1 + zeros(1,nVars);
idx= Lb;
bestX = zeros(1,nVars);
bestFx = 0;
% --------------------------------- start main loop
bStop = false;
iter = 0;
fprintf("------------- Maximizing Fx ----------------\n");
while ~bStop
  iter = iter +1;
  if mod(iter, FivePercent) == 0
    percent = 100*iter/MaxIters;
    fprintf("Progress %5.2f%%\n", percent);
  end

  r2adj = myFit(idx);
  if r2adj > bestFx
    bestFx = r2adj;
    bestX = idx;
    bestMdl = mdl;
    fprintf("Iter %d, Fx = %e, X=[", iter, bestFx)
    fprintf("%d, ", bestX);
    fprintf("]\n");
  end
  % Start implementing the quasi-nested loops
  idx(1) = idx(1) + iStep(1);
  if idx(1) > iTo(1)
    idx(1) = iFrom(1);
    for i = 2:nVars
      idx(i) = idx(i) + iStep(i);
      if idx(i) > iTo(i) && i < nVars
        idx(i) = iFrom(i);
      elseif idx(i) > iTo(i) && i == nVars
        bStop = true;
      else
        break
      end
    end
  end
end
fprintf("Regression model\n")
format long
AdjR2 = bestMdl.Rsquared.Adjusted;
fprintf("Adjusted R-square = %14.12f\n", AdjR2);
% writematrix(AdjR2, filename, 'Sheet', 'R2', 'Range', 'A2:A2');
mdl = bestMdl
bx = bestX;
bestX = [AdjR2];
for i = 1:numIndpX
    bestX = [bestX idxX(i)];
end
bestX = [bestX bx];
k =length(bestX);
res = zeros(3,k);
res(1,:) = bestX;
k2 = 3+numIndpX;
res(2,2:k2+1) = mdl.Coefficients{:,1}';
res(3,3:2+numIndpX) = pwrs;
writematrix(res(1,:), filename, 'Sheet', 'Results', 'Range', 'A2:Z2');
writematrix(res(2,2:k2+1), filename, 'Sheet', 'Results', 'Range', 'B3:Z3');
writematrix(res(3,3:2+numIndpX), filename, 'Sheet', 'Results', 'Range',
'B4:Z4');
%
% now do the % projections
ycalc = project(res);
percErr = 100.*(ycalc-ydata)./ydata;
ProjMat = [ydata xdata ycalc percErr];
writematrix(ProjMat, filename, 'Sheet', 'Project', 'Range', strcat("A2:Z",
num2str(1+length(ydata))));

i =2+numIndpX;
j=i+numIndpX;
sJ = num2str(bestX(j));
if bestX(i) == 0
  sx1 = strcat("ln(X", sJ,")");
elseif bestX(i) == 1
  sx1 = strcat("X", sJ);
elseif bestX(i) == 4
  sx1 = strcat("sqrt(X", sJ,")");
elseif bestX(i) > 1
  sx1 = strcat("X", sJ,"^", num2str(bestX(i)));
```

```
end
ix1 = sJ;
k2=1+numIndpX+8;
if bestX(k2+1) == 1
  sOp1 = "*";
else
  sOp1 = "/";
end
if bestX(k2+2) == 1
  sOp2 = "*";
else
  sOp2 = "/";
end

i=i+1;
j=j+1;
sJ = num2str(bestX(j));
if bestX(i) == 1
  sx2 = strcat("X", sJ);
elseif bestX(i) == 0
  sx2 = strcat("ln(X", sJ,")");
elseif bestX(i) == 4
  sx2 = strcat("sqrt(X", sJ,")");
elseif bestX(i) > 1
  sx2 = strcat("X", sJ,"^", num2str(bestX(i)));
end
ix2 = sJ;
i=i+1;
j=j+1;
sJ = num2str(bestX(j));
if bestX(i) == 1
  sx3 = strcat("X", sJ);
elseif bestX(i) == 0
  sx3 = strcat("ln(X", sJ,")");
elseif bestX(i) == 4
  sx3 = strcat("sqrt(X", sJ,")");
elseif bestX(i) > 1
  sx3 = strcat("X", sJ,"^", num2str(bestX(i)));
end
ix3 = sJ;
i=i+1;
j=j+1;
sJ = num2str(bestX(j));
if bestX(i) == 1
  sx4 = strcat("X", sJ);
elseif bestX(i) == 0
  sx4 = strcat("ln(X", sJ,")");
elseif bestX(i) == 4
  sx4 = strcat("sqrt(X", sJ,")");
elseif bestX(i) > 1
  sx4 = strcat("X", sJ,"^", num2str(bestX(i)));
end
ix4 = sJ;
s = strcat("Y = (", num2str(mdl.Coefficients{1,1}),")");
fprintf("\n\nBest model is:\n%s\n", s)
for i=1:numIndpX
  j = idxX(i);
  sJ= num2str(j);
  if pwrs(i)==1
    s = strcat("   + (", num2str(mdl.Coefficients{1+i,1}),")*X", sJ);
  elseif pwrs(i)==0
    s = strcat("   + (", num2str(mdl.Coefficients{1+i,1}),")*ln(X", sJ,")");
  else
    sp = num2str(pwrs(i));
    s = strcat("   + (", num2str(mdl.Coefficients{1+i,1}),")*X", sJ,"^",sp);
  end
  fprintf("%s\n", s)
end
s = strcat("   + (",num2str(mdl.Coefficients{2+numIndpX,1}), ")*", sx1, sOp1,
sx2);
fprintf("%s\n", s)
s = strcat("   + (", num2str(mdl.Coefficients{3+numIndpX,1}), ")*", sx3,
sOp2, sx4);
fprintf("%s\n", s)

format short
fprintf("Done!\n");
diary off
beep on
for i=1:3
  beep;
  pause(1);
end

function x = myfx(x,p)
  if p==0
    x = log(x);
  elseif p==2
    x = x.^2;
  elseif p==3
    x = x.^3;
  elseif p==4
    x = sqrt(x);
  end
end

function r = myFit(coeff)
  global mdl
  global xdata
  global ydata
  global idxX
  global pwrs

  numIndpX = length(idxX);
  y = ydata;
  px1 = coeff(1);
  px2 = coeff(2);
  px3 = coeff(3);
  px4 = coeff(4);

  i1 = coeff(5);
  i2 = coeff(6);
  i3 = coeff(7);
  i4 = coeff(8);
  op1 = coeff(9);
  op2 = coeff(10);

  if (i1==i2) || (i3==i4)
    r = 0;
    return;
  end
  % compare X1,X2 with X3,X4
  if ((i1==i3) && (px1==px3)) || ((i2==i4) && (px2==px4))
    r = 0;
    return;
  end
  if ((i1==i4) && (px1==px4)) || ((i2==i3) && (px2==px3))
    r = 0;
    return;
  end

  x1 = myfx(xdata(:,i1),px1);
  x2 = myfx(xdata(:,i2),px2);
  x3 = myfx(xdata(:,i3),px3);
  x4 = myfx(xdata(:,i4),px4);
  if op1==1
    x12 = x1.*x2;
  else
    x12 = x1./x2;
  end
  if op2==1
    x34 = x3.*x4;
  else
    x34  = x3./x4;
  end
  X = [];
  for i=1:numIndpX
    X = [X myfx(xdata(:,idxX(i)),pwrs(i))];
  end
  X = [X x12 x34];
  mdl = fitlm(X,y);
  r = mdl.Rsquared.Adjusted;
end

function ycalc = project(res)
  global xdata
  global idxX
  global pwrs

  numIndpX = length(idxX);
  shift = numIndpX;
  px1 = res(1,2+shift);
  px2 = res(1,3+shift);
  px3 = res(1,4+shift);
  px4 = res(1,5+shift);
  i1 = res(1,6+shift);
  i2 = res(1,7+shift);
  i3 = res(1,8+shift);
  i4 = res(1,9+shift);
  op1 = res(1,10+shift);
  op2 = res(1,11+shift);
  if px1 == 0
    x1= log(xdata(:,i1));
  elseif px1 == 1
    x1 = xdata(:,i1);
  elseif px1 == 2
    x1 = xdata(:,i1).^2;
  elseif px1 == 3
    x1 = xdata(:,i1).^3;
  elseif px1 == 4
    x1 = sqrt(xdata(:,i1));
  end

  if px2 == 0
    x2= log(xdata(:,i2));
  elseif px2 == 1
    x2 = xdata(:,i2);
  elseif px2 == 2
    x2 = xdata(:,i2).^2;
  elseif px2 == 3
    x2 = xdata(:,i2).^3;
  elseif px2 == 4
    x2 = sqrt(xdata(:,i2));
  end

  if px3 == 0
    x3= log(xdata(:,i3));
  elseif px3 == 1
    x3 = xdata(:,i3);
  elseif px3 == 2
    x3 = xdata(:,i3).^2;
  elseif px3 == 3
    x3 = xdata(:,i3).^3;
  elseif px3 == 4
    x3 = sqrt(xdata(:,i3));
  end

  if px4 == 0
    x4= log(xdata(:,i4));
  elseif px4 == 1
    x4 = xdata(:,i4);
  elseif px4 == 2
    x4 = xdata(:,i4).^2;
  elseif px4 == 3
    x4 = xdata(:,i4).^3;
  elseif px4 == 4
    x4 = sqrt(xdata(:,i4));
  end

  ycalc = res(2,2);
  for i=1:numIndpX
    ycalc = ycalc + res(2,2+i) * myfx(xdata(:,idxX(i)),pwrs(i));
  end
  k = 1 + numIndpX;
  if op1==1
    ycalc = ycalc + res(2,k+2) * x1.*x2;
  else
    ycalc = ycalc + res(2,k+2) * x1./x2;
  end

  if op2==1
    ycalc = ycalc + res(2,k+3) * x3.*x4;
  else
    ycalc = ycalc + res(2,k+3) * x3./x4;
  end
end
```

Program clem2b is an expanded version of clem1b. The additional code serves to handle more single-variable terms and the two cross-product terms.

The console output from running program clem2b is shown below:

```
Program  clem2b

2025-Jan-2 20_09_45
Please wait ...

Excel file used is I:\DropBox\Dropbox\MATLAB\Clement3\data2.xlsx
Diary file used is data2b_2025-Jan-2 20_09_45.txt
------------- Calculating Max Iters ----------------
Max iters = 16,384
------------- Maximizing Fx ----------------
Iter 276, Fx = 5.205733e-01, X=[2, 2, 1, 1, 2, 1, 2, 1, 1, 1, ]
Iter 291, Fx = 5.927248e-01, X=[1, 2, 1, 1, 3, 1, 2, 1, 1, 1, ]
Iter 297, Fx = 6.247216e-01, X=[1, 1, 1, 2, 3, 1, 2, 1, 1, 1, ]
Iter 548, Fx = 6.931868e-01, X=[2, 2, 1, 1, 3, 1, 3, 1, 1, 1, ]
Progress  5.00%
Progress 10.00%
Progress 15.00%
Progress 20.00%
Progress 24.99%
Iter 4482, Fx = 9.710166e-01, X=[2, 1, 1, 1, 1, 3, 2, 1, 2, 1, ]
Progress 29.99%
Progress 34.99%
Iter 6018, Fx = 1.000000e+00, X=[2, 1, 1, 1, 1, 3, 4, 2, 2, 1, ]
Regression model
Adjusted R-square = 1.000000000000

mdl =


Linear regression model:
    y ~ 1 + x1 + x2 + x3 + x4 + x5 + x6

Estimated Coefficients:
                        Estimate                 SE                 tStat          pValue
                    ________________    ____________________    ________________    ______

    (Intercept)     1.99999999999955    1.47491324763634e-05    135601.195745221      0
    x1             0.300000000000004    1.08591279299046e-07    2762652.78332199      0
    x2             0.400000000000003    2.11545763592126e-07    1890843.82125104      0
    x3                           0.5    4.95989239781328e-07    1008086.38554425      0
    x4             0.600000000000018    6.72073636871872e-07    892759.315471268      0
    x5                             1    2.95655307659211e-09    338231709.052441      0
    x6             0.999999999999999    1.10726119279027e-08    90312927.6553099      0
```

```
Number of observations: 100, Error degrees of freedom: 93
Root Mean Squared Error: 2.53e-05
R-squared: 1,  Adjusted R-Squared: 1
F-statistic vs. constant model: 1.75e+31, p-value = 0


Best model is:
Y = (2)
   + (0.3)*X1
   + (0.4)*X2
   + (0.5)*X3
   + (0.6)*X4
   + (1)*X1^2/X3
   + (1)*X4*X2
Done!
```

As expected, program clem2b finds the correct best model. Figures (2.3) and (2.4) show the output as it appears in sheets **Results** and **Project** of the Excel file data2.xlsx.

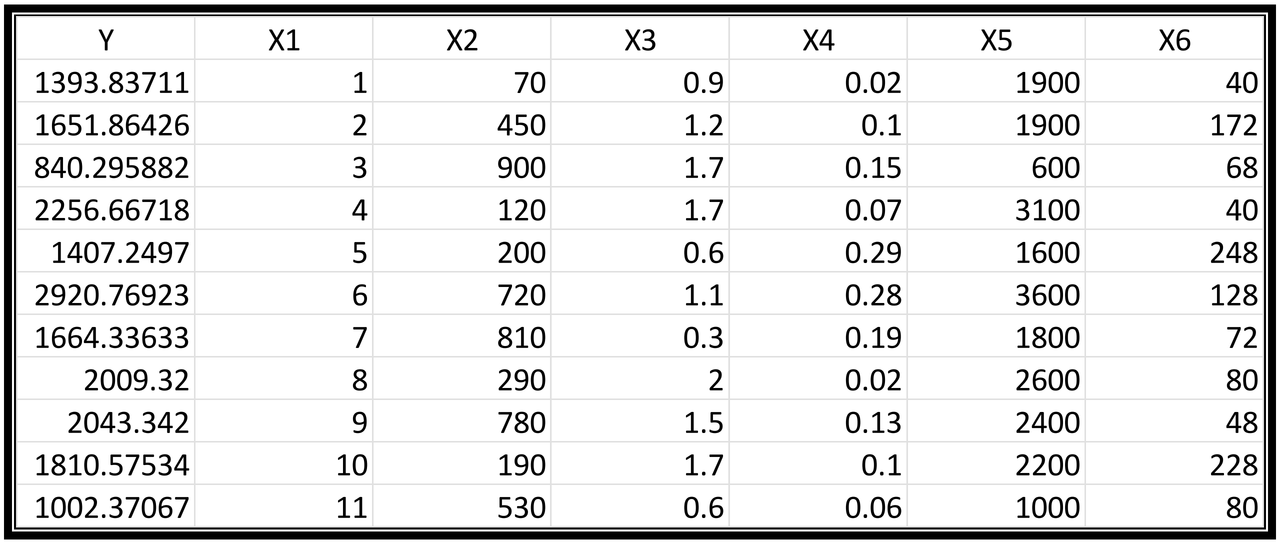
7/Optimizing Models with Three Cross-Product Terms

This section looks at models with two cross-product terms. The model used is:

$$Y = a + \sum_{i=1}^{6} b_i\, X_i^{px(i)} + \sum_{j,k}^{6} c_{jk}\, X_j^{qx(j)}\; op(j,k) X_k^{qx(k)} \qquad (17)$$

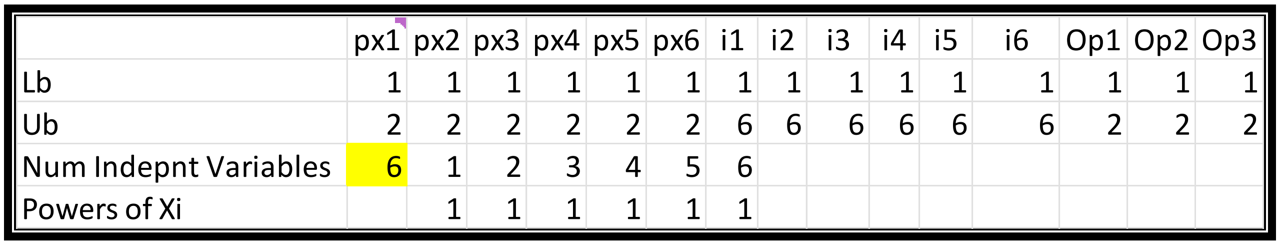
The above model has six single-variable terms and three cross-product terms.

Figure (7.1), (7.2), (7.3), and (7.4) show the sheets **Data**, **Transf**, **Results**, and **Project** that are in file data3.xlsx.



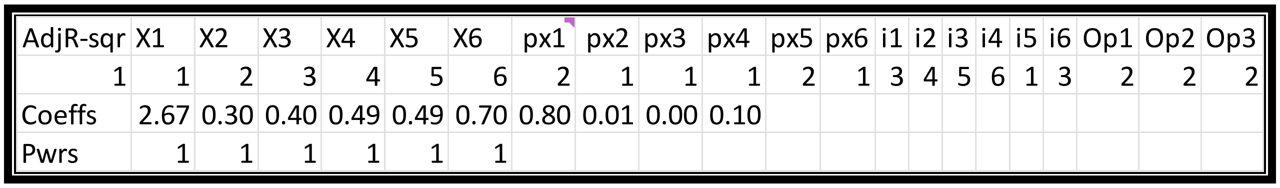
Y	X1	X2	X3	X4	X5	X6
1393.83711	1	70	0.9	0.02	1900	40
1651.86426	2	450	1.2	0.1	1900	172
840.295882	3	900	1.7	0.15	600	68
2256.66718	4	120	1.7	0.07	3100	40
1407.2497	5	200	0.6	0.29	1600	248
2920.76923	6	720	1.1	0.28	3600	128
1664.33633	7	810	0.3	0.19	1800	72
2009.32	8	290	2	0.02	2600	80
2043.342	9	780	1.5	0.13	2400	48
1810.57534	10	190	1.7	0.1	2200	228
1002.37067	11	530	0.6	0.06	1000	80

Figure 7.1. The partial view of sheet ***Data*** *in file data3.xlsx.*



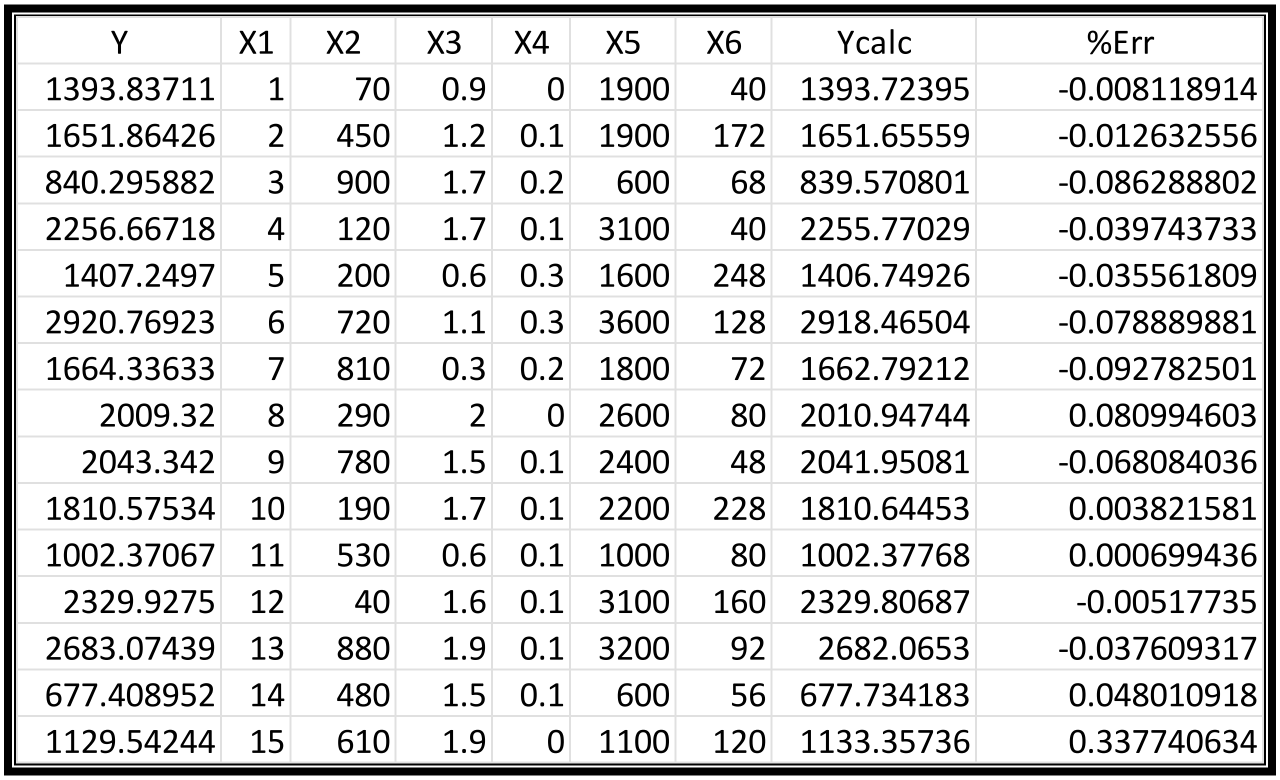
	px1	px2	px3	px4	px5	px6	i1	i2	i3	i4	i5	i6	Op1	Op2	Op3
Lb	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Ub	2	2	2	2	2	2	6	6	6	6	6	6	2	2	2
Num Indepnt Variables	6	1	2	3	4	5	6								
Powers of Xi		1	1	1	1	1	1								

Figure 7.2. The sheet ***Transf*** *in file data3.xlsx.*



AdjR-sqr	X1	X2	X3	X4	X5	X6	px1	px2	px3	px4	px5	px6	i1	i2	i3	i4	i5	i6	Op1	Op2	Op3
1	1	2	3	4	5	6	2	1	1	1	2	1	3	4	5	6	1	3	2	2	2
Coeffs	2.67	0.30	0.40	0.49	0.49	0.70	0.80	0.01	0.00	0.10											
Pwrs	1	1	1	1	1	1															

Figure 7.3. The sheet ***Results*** *in file data3.xlsx.*



Y	X1	X2	X3	X4	X5	X6	Ycalc	%Err
1393.83711	1	70	0.9	0	1900	40	1393.72395	-0.008118914
1651.86426	2	450	1.2	0.1	1900	172	1651.65559	-0.012632556
840.295882	3	900	1.7	0.2	600	68	839.570801	-0.086288802
2256.66718	4	120	1.7	0.1	3100	40	2255.77029	-0.039743733
1407.2497	5	200	0.6	0.3	1600	248	1406.74926	-0.035561809
2920.76923	6	720	1.1	0.3	3600	128	2918.46504	-0.078889881
1664.33633	7	810	0.3	0.2	1800	72	1662.79212	-0.092782501
2009.32	8	290	2	0	2600	80	2010.94744	0.080994603
2043.342	9	780	1.5	0.1	2400	48	2041.95081	-0.068084036
1810.57534	10	190	1.7	0.1	2200	228	1810.64453	0.003821581
1002.37067	11	530	0.6	0.1	1000	80	1002.37768	0.000699436
2329.9275	12	40	1.6	0.1	3100	160	2329.80687	-0.00517735
2683.07439	13	880	1.9	0.1	3200	92	2682.0653	-0.037609317
677.408952	14	480	1.5	0.1	600	56	677.734183	0.048010918
1129.54244	15	610	1.9	0	1100	120	1133.35736	0.337740634

Figure 7.4. The partial view of sheet ***Project*** *in file data3.xlsx.*

The program clem3 attempts to determine the best model. Here is the listing for clem3:

```
clc
close allfor clem3
clear all

global mdl
global xdata
global ydata
global idxX
global pwrs

warning("off")
dt = datetime('now','TimeZone','local','Format','y-MMM-d HH_mm_ss');
dtstr = string(dt);
filename = strcat(pwd,'\data3.xlsx');
delete("data3_*.txt");
outFile = strcat("data3_",dtstr,".txt");
diary(outFile)

fprintf("Program  clem3\n\n");
fprintf("%s\n", dtstr);
fprintf("Please wait ...\n\n");
fprintf("Excel file used is %s\n", filename);
fprintf("Diary file used is %s\n", outFile);
xyData = readmatrix(filename,'Sheet','Data');
[maxrows,~] = size(xyData);
ydata = xyData(:,1);
xdata = xyData(:,2:end);
xyTransf = readmatrix(filename,'Sheet','Transf');
xyTransf = xyTransf(:,2:end);
[n,nXVars] = size(xdata);
numIndpX = xyTransf(3,1);
idxX = zeros(1,numIndpX);
pwrs = zeros(1,numIndpX);
for i=1:numIndpX
  idxX(i) = xyTransf(3,1+i);
  pwrs(i) = xyTransf(4,1+i);
  if isnan(pwrs(i)), pwrs(i) = 1; end
end

Lb = xyTransf(1,:);
Ub = 1+xyTransf(2,:);
nVars = length(Ub);
MaxIters = 4000;
MaxPop= 400;
itersToReset = 400;
MaxTries = 5;
numTry = 0;
while true
  numTry = numTry + 1;
  if numTry > MaxTries
    fprintf("Optimization failed\n");
    return;
  end
  if numTry > 1
    fprintf("Optimization attempt # %d\n", numTry)
  end
  [bestX,gBestCost,bestMdl] = pso(@myFit,Lb,Ub,MaxPop,MaxIters,itersToReset);
  px1 = bestX(1);
  px2 = bestX(2);
  px3 = bestX(3);
  px4 = bestX(4);
  px5 = bestX(5);
  px6 = bestX(6);

  i1= bestX(1+numIndpX);
  i2= bestX(2+numIndpX);
  i3= bestX(3+numIndpX);
  i4= bestX(4+numIndpX);
  i5= bestX(5+numIndpX);
  i6= bestX(6+numIndpX);

  if (i1 == i2) || (i3 == i4) || (i5 == i6), continue, end
  % compare X1,X2 with X3,X4
  if ((i1==i3) && (px1==px3)) || ((i2==i4) && (px2==px4)), continue, end
  if ((i1==i4) && (px1==px4)) || ((i2==i3) && (px2==px3)), continue, end
  % compare X1,X2 with X5,X6
  if ((i1==i5) && (px1==px5)) || ((i2==i6) && (px2==px6)), continue, end
  if ((i1==i6) && (px1==px6)) || ((i2==i5) && (px2==px5)), continue, end
  % compare X3,X4 with X5,X6
  if ((i3==i5) && (px3==px5)) || ((i4==i6) && (px4==px6)), continue, end
  if ((i3==i6) && (px3==px6)) || ((i4==i5) && (px4==px5)), continue, end
  break;
end
mdl = bestMdl;
AdjR2 = mdl.Rsquared.Adjusted;
fprintf("Regression model\n")
fprintf("Adjusted R-square = %14.10f\n", AdjR2);
format long
mdl
bx = bestX;
bestX = [AdjR2];
for i = 1:numIndpX
    bestX = [bestX idxX(i)];
end
bestX = [bestX bx];
k =length(bestX);
res = zeros(3,k);
res(1,:) = bestX;
k2 = 4+numIndpX;
res(2,2:k2+1) = mdl.Coefficients{:,1}';
res(3,3:2+numIndpX) = pwrs;
writematrix(res(1,:), filename, 'Sheet', 'Results', 'Range', 'A2:Z2');
writematrix(res(2,2:k2+1), filename, 'Sheet', 'Results', 'Range', 'B3:Z3');
writematrix(res(3,3:2+numIndpX), filename, 'Sheet', 'Results', 'Range',
'B4:Z4');
%
% now do the % projections
ycalc = project(res);
percErr = 100.*(ycalc-ydata)./ydata;
ProjMat = [ydata xdata ycalc percErr];
writematrix(ProjMat, filename, 'Sheet', 'Project', 'Range', strcat("A2:Z",
num2str(1+length(ydata))));

k2=length(bestX);
if bestX(k2-2) == 1
  sOp1 = "*";
else
  sOp1 = "/";
end
```

```
if bestX(k2-1) == 1
  sOp2 = "*";
else
  sOp2 = "/";
end
if bestX(k2) == 1
  sOp3 = "*";
else
  sOp3 = "/";
end

i=2+numIndpX;
j=i+numIndpX;
sJ = num2str(bestX(j));
if bestX(i) == 0
  sx1 = strcat("ln(X", sJ,")");
elseif bestX(i) == 1
  sx1 = strcat("X", sJ);
elseif bestX(i) == 4
  sx1 = strcat("sqrt(X", sJ,")");
elseif bestX(i) > 1
  sx1 = strcat("X", sJ,"^", num2str(bestX(i)));
end


i=i+1;
j=j+1;
sJ = num2str(bestX(j));
if bestX(i) == 1
  sx2 = strcat("X", sJ);
elseif bestX(i) == 0
  sx2 = strcat("ln(X", sJ,")");
elseif bestX(i) == 4
  sx2 = strcat("sqrt(X", sJ,")");
elseif bestX(i) > 1
  sx2 = strcat("X", sJ,"^", num2str(bestX(i)));
end

i=i+1;
j=j+1;
sJ = num2str(bestX(j));
if bestX(i) == 1
  sx3 = strcat("X", sJ);
elseif bestX(i) == 0
  sx3 = strcat("ln(X", sJ,")");
elseif bestX(i) == 4
  sx3 = strcat("sqrt(X", sJ,")");
elseif bestX(i) > 1
  sx3 = strcat("X", sJ,"^", num2str(bestX(i)));
end

i=i+1;
j=j+1;
sJ = num2str(bestX(j));
if bestX(i) == 1
  sx4 = strcat("X", sJ);
elseif bestX(i) == 0
```

```
  sx4 = strcat("ln(X", sJ,")");
elseif bestX(i) == 4
  sx4 = strcat("sqrt(X", sJ,")");
elseif bestX(i) > 1
  sx4 = strcat("X", sJ,"^", num2str(bestX(i)));
end

i=i+1;
j=j+1;
sJ = num2str(bestX(j));
if bestX(i) == 1
  sx5 = strcat("X", sJ);
elseif bestX(i) == 0
  sx5 = strcat("ln(X", sJ,")");
elseif bestX(i) == 4
  sx5 = strcat("sqrt(X", sJ,")");
elseif bestX(i) > 1
  sx5 = strcat("X", sJ,"^", num2str(bestX(i)));
end

i=i+1;
j=j+1;
sJ = num2str(bestX(j));
if bestX(i) == 1
  sx6 = strcat("X", sJ);
elseif bestX(i) == 0
  sx6 = strcat("ln(X", sJ,")");
elseif bestX(i) == 4
  sx6 = strcat("sqrt(X", sJ,")");
elseif bestX(i) > 1
  sx6 = strcat("X", sJ,"^", num2str(bestX(i)));
end

s = strcat("Y = (", num2str(mdl.Coefficients{1,1}),")");
fprintf("\n\nBest model is:\n%s\n", s)
for i=1:numIndpX
  j = idxX(i);
  sJ= num2str(j);
  if pwrs(i)==1
    s = strcat("   + (", num2str(mdl.Coefficients{1+i,1}),")*X", sJ);
  elseif pwrs(i)==0
    s = strcat("   + (", num2str(mdl.Coefficients{1+i,1}),")*ln(X", sJ,")");
  else
    sp = num2str(pwrs(i));
    s = strcat("   + (", num2str(mdl.Coefficients{1+i,1}),")*X", sJ,"^",sp);
  end
  fprintf("%s\n", s)
end
s = strcat("   + (",num2str(mdl.Coefficients{2+numIndpX,1}), ")*", sx1, sOp1,
sx2);
fprintf("%s\n", s)
s = strcat("   + (",num2str(mdl.Coefficients{3+numIndpX,1}), ")*", sx3, sOp2,
sx4);
fprintf("%s\n", s)
s = strcat("   + (", num2str(mdl.Coefficients{4+numIndpX,1}), ")*", sx5,
sOp3, sx6);
fprintf("%s\n", s)
```

```
format short
fprintf("Done!\n");
diary off
beep on
for i=1:3
  beep;
  pause(1);
end

function x = myfx(x,p)
  if p==0
    x = log(x);
  elseif p==2
    x = x.^2;
  elseif p==3
    x = x.^3;
  elseif p==4
    x = sqrt(x);
  end
end

function r = myFit(coeff)
  global mdl
  global xdata
  global ydata
  global idxX
  global pwrs

  [~,nXVars] = size(xdata);
  numIndpX = length(idxX);
  y = ydata;
  px1 = coeff(1);
  px2 = coeff(2);
  px3 = coeff(3);
  px4 = coeff(4);
  px5 = coeff(5);
  px6 = coeff(6);
  i1 = coeff(7);
  i2 = coeff(8);
  i3 = coeff(9);
  i4 = coeff(10);
  i5 = coeff(11);
  i6 = coeff(12);
  op1 = coeff(13);
  op2 = coeff(14);
  op3 = coeff(15);
  if i1 > nXVars || i2 > nXVars || i3 > nXVars || ...
      i4 > nXVars || i5 > nXVars || i6 > nXVars
    r = 1;
    return
  end

  if (i1==i2) || (i3==i4) || (i5==i6)
    r = 1;
    return;
  end
  % compare X1,X2 with X3,X4
  if ((i1==i3) && (px1==px3)) || ((i2==i4) && (px2==px4))
    r = 1;
    return;
  end
  if ((i1==i4) && (px1==px4)) || ((i2==i3) && (px2==px3))
    r = 1;
    return;
  end
  % compare X1,X2 with X5,X6
  if ((i1==i5) && (px1==px5)) || ((i2==i6) && (px2==px6))
    r = 1;
    return;
  end
  if ((i1==i6) && (px1==px6)) || ((i2==i5) && (px2==px5))
    r = 1;
    return;
  end
  % compare X3,X4 with X5,X6
  if ((i3==i5) && (px3==px5)) || ((i4==i6) && (px4==px6))
    r = 1;
    return;
  end
  if ((i3==i6) && (px3==px6)) || ((i4==i5) && (px4==px5))
    r = 1;
    return;
  end

  x1 = myfx(xdata(:,i1),px1);
  x2 = myfx(xdata(:,i2),px2);
  x3 = myfx(xdata(:,i3),px3);
  x4 = myfx(xdata(:,i4),px4);
  x5 = myfx(xdata(:,i5),px5);
  x6 = myfx(xdata(:,i6),px6);
  if op1==1
    x12 = x1.*x2;
  else
    x12 = x1./x2;
  end
  if op2==1
    x34 = x3.*x4;
  else
    x34  = x3./x4;
  end
  if op3==1
    x56 = x5.*x6;
  else
    x56  = x5./x6;
  end
  X = [];
  for i=1:numIndpX
    X = [X myfx(xdata(:,idxX(i)),pwrs(i))];
  end
  X = [X x12 x34 x56];
  mdl = fitlm(X,y);
  r = 1 - mdl.Rsquared.Adjusted;
end
```

```
function ycalc = project(res)
  global xdata
  global idxX
  global pwrs

  numIndpX = length(idxX);
  shift = 1 + numIndpX;
  px1 = res(1,1+shift);
  px2 = res(1,2+shift);
  px3 = res(1,3+shift);
  px4 = res(1,4+shift);
  px5 = res(1,5+shift);
  px6 = res(1,6+shift);
  i1 = res(1,7+shift);
  i2 = res(1,8+shift);
  i3 = res(1,9+shift);
  i4 = res(1,10+shift);
  i5 = res(1,11+shift);
  i6 = res(1,12+shift);
  op1 = res(1,13+shift);
  op2 = res(1,14+shift);
  op3 = res(1,15+shift);
  if px1 == 0
    x1= log(xdata(:,i1));
  elseif px1 == 1
    x1 = xdata(:,i1);
  elseif px1 == 2
    x1 = xdata(:,i1).^2;
  elseif px1 == 3
    x1 = xdata(:,i1).^3;
  elseif px1 == 4
    x1 = sqrt(xdata(:,i1));
  end

  if px2 == 0
    x2= log(xdata(:,i2));
  elseif px2 == 1
    x2 = xdata(:,i2);
  elseif px2 == 2
    x2 = xdata(:,i2).^2;
  elseif px2 == 3
    x2 = xdata(:,i2).^3;
  elseif px2 == 4
    x2 = sqrt(xdata(:,i2));
  end

  if px3 == 0
    x3= log(xdata(:,i3));
  elseif px3 == 1
    x3 = xdata(:,i3);
  elseif px3 == 2
    x3 = xdata(:,i3).^2;
  elseif px3 == 3
    x3 = xdata(:,i3).^3;
  elseif px3 == 4
    x3 = sqrt(xdata(:,i3));
  end

  if px4 == 0
    x4= log(xdata(:,i4));
  elseif px4 == 1
    x4 = xdata(:,i4);
  elseif px4 == 2
    x4 = xdata(:,i4).^2;
  elseif px4 == 3
    x4 = xdata(:,i4).^3;
  elseif px4 == 4
    x4 = sqrt(xdata(:,i4));
  end

  if px5 == 0
    x5= log(xdata(:,i5));
  elseif px5 == 1
    x5 = xdata(:,i5);
  elseif px5 == 2
    x5 = xdata(:,i5).^2;
  elseif px5 == 3
    x5 = xdata(:,i5).^3;
  elseif px5 == 4
    x5 = sqrt(xdata(:,i5));
  end

  if px6 == 0
    x6= log(xdata(:,i6));
  elseif px6 == 1
    x6 = xdata(:,i6);
  elseif px6 == 2
    x6 = xdata(:,i6).^2;
  elseif px6 == 3
    x6 = xdata(:,i6).^3;
  elseif px6 == 4
    x6 = sqrt(xdata(:,i6));
  end

  ycalc = res(2,2);
  for i=1:numIndpX
    ycalc = ycalc + res(2,2+i) * myfx(xdata(:,idxX(i)),pwrs(i));
  end
  k = 1 + numIndpX;
  if op1==1
    ycalc = ycalc + res(2,k+2) * x1.*x2;
  else
    ycalc = ycalc + res(2,k+2) * x1./x2;
  end
  if op2==1
    ycalc = ycalc + res(2,k+3) * x3.*x4;
  else
    ycalc = ycalc + res(2,k+3) * x3./x4;
  end
  if op3==1
    ycalc = ycalc + res(2,k+4) * x5.*x6;
  else
    ycalc = ycalc + res(2,k+4) * x5./x6;
  end
end
```

Program clem3 is an extended version of program clem2. In addition, the calls to function pso() fall inside a while loop. The loop tests the indices and powers of the variables in the cross-product terms to check for invalid models that have the same two variables in a cross-product term or two cross-product terms that have the same variables raised to the same powers. The while loop has a maximum of five iterations (which you can easily change) before it declares the optimization process a failure. I inserted the while loop as the latest modification to the code of program clem3. This modification helps avoid invalid models that obtain high values for the adjusted R-square statistic but also yield high values for the percentage errors! Using the while loop made a significant difference in yielding good (albeit not perfect) results.

The console output from running program clem3 is shown below:

```
Program  clem3

2025-Jan-1 10_52_07
Please wait ...

Excel file used is I:\DropBox\Dropbox\MATLAB\Clement3\data3.xlsx
Diary file used is data3_2025-Jan-1 10_52_07.txt
Best Fx = 2.233392e-04, Best X =[2, 1, 1, 1, 2, 2, 5, 3, 1, 2, 1, 3, 1, 1,  2]
Iter = 1, Best Fx = 3.801587e-05, Best X = [1, 2, 2, 2, 1, 1, 6, 4, 3, 4, 6, 6, 2, 2,  2]
Iter = 5, Best Fx = 1.295143e-05, Best X = [1, 1, 1, 1, 1, 2, 4, 2, 6, 3, 1, 4, 2, 2,  2]
Iter = 8, Best Fx = 1.045274e-06, Best X = [1, 1, 2, 2, 2, 2, 4, 6, 1, 6, 2, 5, 2, 1,  1]
Iter = 12, Best Fx = 1.774874e-08, Best X = [2, 2, 1, 2, 1, 2, 4, 4, 2, 2, 2, 1, 2, 2,  2]
Iter = 27, Best Fx = 7.452536e-10, Best X = [2, 1, 2, 2, 1, 1, 2, 4, 1, 3, 2, 6, 1, 1,  2]
Iter = 157, Best Fx = 5.971167e-10, Best X = [1, 2, 2, 1, 2, 2, 2, 4, 2, 3, 5, 2, 1, 1,  1]
Iter = 385, Best Fx = 5.867705e-10, Best X = [2, 1, 2, 2, 1, 1, 2, 4, 1, 3, 2, 6, 1, 1,  2]
Iter = 514, Best Fx = 1.814912e-10, Best X = [1, 1, 2, 1, 2, 1, 2, 5, 6, 6, 4, 6, 1, 2,  1]
----------------------- Reset population -----------------------
AT iteration 914
----------------------- Reset population -----------------------
AT iteration 1314
----------------------- Reset population -----------------------
AT iteration 1714
Iter = 1929, Best Fx = 0.000000e+00, Best X = [1, 1, 1, 1, 1, 1, 1, 3, 1, 4, 2, 6, 2, 2,  2]
Optimization attempt # 2
Best Fx = 6.429451e-10, Best X =[1, 1, 2, 1, 2, 2, 4, 2, 1, 3, 4, 2, 1, 2,  2]
Iter = 1, Best Fx = 6.406290e-10, Best X = [2, 2, 2, 1, 1, 1, 3, 2, 1, 5, 1, 4, 2, 2,  1]
Iter = 15, Best Fx = 6.029685e-10, Best X = [1, 1, 2, 1, 1, 2, 2, 6, 6, 4, 3, 6, 1, 2,  1]
Iter = 52, Best Fx = 5.971167e-10, Best X = [1, 2, 2, 2, 2, 1, 2, 2, 3, 2, 4, 6, 2, 1,  2]
Iter = 150, Best Fx = 4.586145e-10, Best X = [1, 1, 2, 1, 2, 2, 4, 2, 1, 3, 4, 2, 1, 2,  2]
Iter = 208, Best Fx = 1.312398e-10, Best X = [1, 2, 2, 2, 1, 1, 4, 6, 2, 3, 3, 6, 1, 2,  2]
----------------------- Reset population -----------------------
AT iteration 608
Iter = 929, Best Fx = 7.892764e-11, Best X = [1, 2, 2, 1, 1, 1, 5, 2, 3, 3, 5, 3, 2, 2,  1]
----------------------- Reset population -----------------------
AT iteration 1329
----------------------- Reset population -----------------------
AT iteration 1729
Iter = 1803, Best Fx = 6.949508e-11, Best X = [2, 2, 1, 1, 2, 1, 4, 1, 1, 2, 1, 5, 2, 2,  2]
----------------------- Reset population -----------------------
AT iteration 2203
```

```
----------------------- Reset population -----------------------
AT iteration 2603
----------------------- Reset population -----------------------
AT iteration 3003
----------------------- Reset population -----------------------
AT iteration 3403
----------------------- Reset population -----------------------
AT iteration 3803
Optimization attempt # 3
Best Fx = 2.063360e-04, Best X =[1, 2, 1, 1, 2, 2, 6, 2, 1, 3, 4, 6, 1, 2,  1]
Iter = 1, Best Fx = 3.571826e-06, Best X = [1, 1, 2, 1, 2, 2, 3, 6, 4, 3, 2, 6, 2, 1,  1]
Iter = 4, Best Fx = 1.041233e-06, Best X = [1, 2, 2, 1, 2, 1, 4, 2, 2, 3, 6, 6, 1, 2,  1]
Iter = 5, Best Fx = 6.385601e-10, Best X = [1, 1, 2, 1, 1, 1, 5, 1, 3, 3, 1, 3, 1, 2,  1]
Iter = 68, Best Fx = 6.331676e-10, Best X = [2, 2, 2, 1, 2, 2, 4, 1, 1, 3, 1, 2, 1, 2,  2]
Iter = 99, Best Fx = 6.210499e-10, Best X = [1, 2, 1, 1, 1, 1, 2, 2, 6, 2, 3, 4, 2, 2,  1]
Iter = 362, Best Fx = 5.969997e-10, Best X = [1, 1, 1, 2, 1, 2, 3, 1, 2, 6, 5, 6, 1, 1,  2]
----------------------- Reset population -----------------------
AT iteration 762
----------------------- Reset population -----------------------
AT iteration 1162
----------------------- Reset population -----------------------
AT iteration 1562
----------------------- Reset population -----------------------
AT iteration 1962
----------------------- Reset population -----------------------
AT iteration 2362
----------------------- Reset population -----------------------
AT iteration 2762
Iter = 2784, Best Fx = 2.289500e-10, Best X = [2, 2, 1, 1, 2, 2, 2, 6, 1, 1, 1, 6, 2, 1,  1]
----------------------- Reset population -----------------------
AT iteration 3184
----------------------- Reset population -----------------------
AT iteration 3584
Iter = 3746, Best Fx = 1.814912e-10, Best X = [1, 2, 2, 1, 2, 1, 5, 3, 6, 6, 2, 4, 1, 1,  2]
Iter = 3969, Best Fx = 1.312398e-10, Best X = [2, 1, 1, 1, 2, 1, 3, 4, 5, 6, 1, 3, 2, 2,  2]
Regression model
Adjusted R-square =   0.9999999999

mdl = 


Linear regression model:
    y ~ 1 + x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9

Estimated Coefficients:
                        Estimate                  SE                 tStat                pValue       
                   ____________________    ____________________    ________________    ____________________

    (Intercept)      2.66929253710782        0.0474702968209636    56.2307951680853    3.51447802105453e-140
    x1               0.297741956002809     0.000630359868309793     472.33647154784                        0
    x2               0.400135021765105       5.033987252107e-05    7948.66974678226                        0
    x3               0.487505520803658      0.00687965389454851    70.8619253637106     6.5238146816822e-163
    x4               0.487403793634596        0.195892273721856    2.48812158016325       0.0135215701752775
    x5               0.699911636298057     6.75416878994054e-06    103626.613143054                        0
    x6               0.798760184141476     0.000214791373996569    3718.77217077737                        0
    x7              0.0100051556625112     1.19512004512548e-06    8371.67421240993                        0
    x8             2.46986589770104e-06    8.06170831946928e-08    30.6370039677104     6.61595433933027e-85
    x9               0.100003644511782     1.58005036935661e-06    63291.4282045976                        0


Number of observations: 250, Error degrees of freedom: 240
Root Mean Squared Error: 0.168
R-squared: 1,  Adjusted R-Squared: 1
F-statistic vs. constant model: 2.11e+11, p-value = 0


Best model is:
Y = (2.6693)
```

```
   + (0.29774)*X1
   + (0.40014)*X2
   + (0.48751)*X3
   + (0.4874)*X4
   + (0.69991)*X5
   + (0.79876)*X6
   + (0.010005)*X3^2/X4
   + (2.4699e-06)*X5/X6
   + (0.1)*X1^2/X3
Done!
```

Program clem3 fails to find the correct model but finds one that is acceptably close. The adjusted R-square value is very close to 1 and the projected values for the dependent variables are close to the actual ones! The minimum and maximum values for the absolute percent error are 0.000357232 and 7.63, respectively. The mean value for the absolute percent error is 2.1873. The standard deviation for the absolute percent error is 2.6108.

8/Using Virtual Nested Loop to Find the Best Models with Three Cross-Product Terms

This section uses the virtual nested loops to find the best model with a single cross-product term. The Excel input and output file is the same as in the last section. The program clem3b determines the best model. Here is the listing for clem3b:

```
clc
close all
clear all

global mdl
global xdata
global ydata
global idxX
global pwrs

warning("off")
dt = datetime('now','TimeZone','local','Format','y-MMM-d HH_mm_ss');
dtstr = string(dt);
filename = strcat(pwd,'\data3b.xlsx');
delete("data3b_*.txt");
outFile = strcat("data3b_",dtstr,".txt");
diary(outFile)

fprintf("Program  clem3b\n\n");
fprintf("%s\n", dtstr);
fprintf("Please wait ...\n\n");
fprintf("Excel file used is %s\n", filename);
fprintf("Diary file used is %s\n", outFile);
xyData = readmatrix(filename,'Sheet','Data');
[maxrows,~] = size(xyData);
```

```
ydata = xyData(:,1);
xdata = xyData(:,2:end);
xyTransf = readmatrix(filename,'Sheet','Transf');
xyTransf = xyTransf(:,2:end);
[n,nXVars] = size(xdata);
numIndpX = xyTransf(3,1);
idxX = zeros(1,numIndpX);
pwrs = zeros(1,numIndpX);
for i=1:numIndpX
  idxX(i) = xyTransf(3,1+i);
  pwrs(i) = xyTransf(4,1+i);
  if isnan(pwrs(i)), pwrs(i) = 1; end
end

Lb = xyTransf(1,:);
Ub = xyTransf(2,:);
nVars = length(Ub);
iFrom = Lb;
iTo = Ub;
iStep = 1 + zeros(1,nVars);
idx= Lb;
bStop = false;
iter = 0;
fprintf("------------- Calculating Max Iters ----------------\n");
while ~bStop
  iter = iter +1;
  % Start implementing the quasiâ€"nested loops
  idx(1) = idx(1) + iStep(1);
  if idx(1) > iTo(1)
    idx(1) = iFrom(1);
    for i = 2:nVars
      idx(i) = idx(i) + iStep(i);
      if idx(i) > iTo(i) && i < nVars
        idx(i) = iFrom(i);
      elseif idx(i) > iTo(i) && i == nVars
        bStop = true;
      else
        break
      end
    end
  end
end
str = num2str(iter);
str = fliplr(regexprep(fliplr(str), '(\d+\.)?(\d{3})(?=\S+)', '$1$2,'));
fprintf("Max iters = %s\n", str);
MaxIters = iter;
FivePercent = fix(iter/20);

iFrom = Lb;
iTo = Ub;
iStep = 1 + zeros(1,nVars);
idx= Lb;
bestX = zeros(1,nVars);
bestFx = 0;
% --------------------------------- start main loop
bStop = false;
iter = 0;
```

```
fprintf("------------- Maximizing Fx ----------------\n");
while ~bStop
  iter = iter +1;
  if mod(iter, FivePercent) == 0
    percent = 100*iter/MaxIters;
    fprintf("Progress %5.2f%%\n", percent);
  end

  r2adj = myFit(idx);
  if r2adj > bestFx
    bestFx = r2adj;
    bestX = idx;
    bestMdl = mdl;
    fprintf("Iter %d, Fx = %e, X=[", iter, bestFx)
    fprintf("%d, ", bestX);
    fprintf("]\n");
  end
  % Start implementing the quasiâ€"nested loops
  idx(1) = idx(1) + iStep(1);
  if idx(1) > iTo(1)
    idx(1) = iFrom(1);
    for i = 2:nVars
      idx(i) = idx(i) + iStep(i);
      if idx(i) > iTo(i) && i < nVars
        idx(i) = iFrom(i);
      elseif idx(i) > iTo(i) && i == nVars
        bStop = true;
      else
        break
      end
    end
  end
end
fprintf("Regression model\n")
format long
AdjR2 = bestMdl.Rsquared.Adjusted;
fprintf("Adjusted R-square = %14.12f\n", AdjR2);
mdl = bestMdl
bx = bestX;
bestX = [AdjR2];
for i = 1:numIndpX
    bestX = [bestX idxX(i)];
end
bestX = [bestX bx];
k =length(bestX);
res = zeros(3,k);
res(1,:) = bestX;
k2 = 4+numIndpX;
res(2,2:k2+1) = mdl.Coefficients{:,1}';
res(3,3:2+numIndpX) = pwrs;
writematrix(res(1,:), filename, 'Sheet', 'Results', 'Range', 'A2:Z2');
writematrix(res(2,2:k2+1), filename, 'Sheet', 'Results', 'Range', 'B3:Z3');
writematrix(res(3,3:2+numIndpX), filename, 'Sheet', 'Results', 'Range',
'B4:Z4');
%
% now do the % projections
ycalc = project(res);
```

```
percErr = 100.*(ycalc-ydata)./ydata;
ProjMat = [ydata xdata ycalc percErr];
writematrix(ProjMat, filename, 'Sheet', 'Project', 'Range', strcat("A2:Z",
num2str(1+length(ydata))));

i =2+numIndpX;
j=i+numIndpX;
sJ = num2str(bestX(j));
if bestX(i) == 0
  sx1 = strcat("ln(X", sJ,")");
elseif bestX(i) == 1
  sx1 = strcat("X", sJ);
elseif bestX(i) == 4
  sx1 = strcat("sqrt(X", sJ,")");
elseif bestX(i) > 1
  sx1 = strcat("X", sJ,"^", num2str(bestX(i)));
end
ix1 = sJ;
k2=1+numIndpX+12;
if bestX(k2+1) == 1
  sOp1 = "*";
else
  sOp1 = "/";
end
if bestX(k2+2) == 1
  sOp2 = "*";
else
  sOp2 = "/";
end
if bestX(k2+3) == 1
  sOp3 = "*";
else
  sOp3 = "/";
end

i=i+1;
j=j+1;
sJ = num2str(bestX(j));
if bestX(i) == 1
  sx2 = strcat("X", sJ);
elseif bestX(i) == 0
  sx2 = strcat("ln(X", sJ,")");
elseif bestX(i) == 4
  sx2 = strcat("sqrt(X", sJ,")");
elseif bestX(i) > 1
  sx2 = strcat("X", sJ,"^", num2str(bestX(i)));
end
ix2 = sJ;
i=i+1;
j=j+1;
sJ = num2str(bestX(j));
if bestX(i) == 1
  sx3 = strcat("X", sJ);
elseif bestX(i) == 0
  sx3 = strcat("ln(X", sJ,")");
elseif bestX(i) == 4
  sx3 = strcat("sqrt(X", sJ,")");
```

```
elseif bestX(i) > 1
  sx3 = strcat("X", sJ,"^", num2str(bestX(i)));
end
ix3 = sJ;
i=i+1;
j=j+1;
sJ = num2str(bestX(j));
if bestX(i) == 1
  sx4 = strcat("X", sJ);
elseif bestX(i) == 0
  sx4 = strcat("ln(X", sJ,")");
elseif bestX(i) == 4
  sx4 = strcat("sqrt(X", sJ,")");
elseif bestX(i) > 1
  sx4 = strcat("X", sJ,"^", num2str(bestX(i)));
end
ix4 = sJ;
i=i+1;
j=j+1;
sJ = num2str(bestX(j));
if bestX(i) == 1
  sx5 = strcat("X", sJ);
elseif bestX(i) == 0
  sx5 = strcat("ln(X", sJ,")");
elseif bestX(i) == 4
  sx5 = strcat("sqrt(X", sJ,")");
elseif bestX(i) > 1
  sx5 = strcat("X", sJ,"^", num2str(bestX(i)));
end
ix5 = sJ;
i=i+1;
j=j+1;
sJ = num2str(bestX(j));
if bestX(i) == 1
  sx6 = strcat("X", sJ);
elseif bestX(i) == 0
  sx6 = strcat("ln(X", sJ,")");
elseif bestX(i) == 4
  sx6 = strcat("sqrt(X", sJ,")");
elseif bestX(i) > 1
  sx6 = strcat("X", sJ,"^", num2str(bestX(i)));
end
ix6 = sJ;
s = strcat("Y = (", num2str(mdl.Coefficients{1,1}),")");
fprintf("\n\nBest model is:\n%s\n", s)
for i=1:numIndpX
  j = idxX(i);
  sJ= num2str(j);
  if pwrs(i)==1
    s = strcat("   + (", num2str(mdl.Coefficients{1+i,1}),")*X", sJ);
  elseif pwrs(i)==0
    s = strcat("   + (", num2str(mdl.Coefficients{1+i,1}),")*ln(X", sJ,")");
  else
    sp = num2str(pwrs(i));
    s = strcat("   + (", num2str(mdl.Coefficients{1+i,1}),")*X", sJ,"^",sp);
  end
  fprintf("%s\n", s)
```

```
end
s = strcat("   + (",num2str(mdl.Coefficients{2+numIndpX,1}), ")*", sx1, sOp1,
sx2);
fprintf("%s\n", s)
s = strcat("   + (", num2str(mdl.Coefficients{3+numIndpX,1}), ")*", sx3,
sOp2, sx4);
fprintf("%s\n", s)
s = strcat("   + (", num2str(mdl.Coefficients{4+numIndpX,1}), ")*", sx5,
sOp3, sx6);
fprintf("%s\n", s)

format short
fprintf("Done!\n");
diary off
beep on
for i=1:3
  beep;
  pause(1);
end

function x = myfx(x,p)
  if p==0
    x = log(x);
  elseif p==2
    x = x.^2;
  elseif p==3
    x = x.^3;
  elseif p==4
    x = sqrt(x);
  end
end

function r = myFit(coeff)
  global mdl
  global xdata
  global ydata
  global idxX
  global pwrs

  numIndpX = length(idxX);
  y = ydata;
  px1 = coeff(1);
  px2 = coeff(2);
  px3 = coeff(3);
  px4 = coeff(4);
  px5 = coeff(5);
  px6 = coeff(6);
  i1 = coeff(7);
  i2 = coeff(8);
  i3 = coeff(9);
  i4 = coeff(10);
  i5 = coeff(11);
  i6 = coeff(12);
  op1 = coeff(13);
  op2 = coeff(14);
  op3 = coeff(15);
  if (i1==i2) || (i3==i4) || (i5==i6)
```

```
    r = 0;
    return;
  end
  % compare X1,X2 with X3,X4
  if ((i1==i3) && (px1==px3)) || ((i2==i4) && (px2==px4))
    r = 0;
    return;
  end
  if ((i1==i4) && (px1==px4)) || ((i2==i3) && (px2==px3))
    r = 0;
    return;
  end
  % compare X1,X2 with X5,X6
  if ((i1==i5) && (px1==px5)) || ((i2==i6) && (px2==px6))
    r = 0;
    return;
  end
  if ((i1==i6) && (px1==px6)) || ((i2==i5) && (px2==px5))
    r = 0;
    return;
  end
  % compare X3,X4 with X5,X6
  if ((i3==i5) && (px3==px5)) || ((i4==i6) && (px4==px6))
    r = 0;
    return;
  end
  if ((i3==i6) && (px3==px6)) || ((i4==i5) && (px4==px5))
    r = 0;
    return;
  end

  x1 = myfx(xdata(:,i1),px1);
  x2 = myfx(xdata(:,i2),px2);
  x3 = myfx(xdata(:,i3),px3);
  x4 = myfx(xdata(:,i4),px4);
  x5 = myfx(xdata(:,i5),px5);
  x6 = myfx(xdata(:,i6),px6);
  if op1==1
    x12 = x1.*x2;
  else
    x12 = x1./x2;
  end
  if op2==1
    x34 = x3.*x4;
  else
    x34  = x3./x4;
  end
  if op3==1
    x56 = x5.*x6;
  else
    x56  = x5./x6;
  end
  X = [];
  for i=1:numIndpX
    X = [X myfx(xdata(:,idxX(i)),pwrs(i))];
  end
  X = [X x12 x34 x56];
  mdl = fitlm(X,y);
  r = mdl.Rsquared.Adjusted;
end

function ycalc = project(res)
  global xdata
  global idxX
  global pwrs

  numIndpX = length(idxX);
  shift = 0 + numIndpX;
  px1 = res(1,2+shift);
  px2 = res(1,3+shift);
  px3 = res(1,4+shift);
  px4 = res(1,5+shift);
  px5 = res(1,6+shift);
  px6 = res(1,7+shift);
  i1 = res(1,8+shift);
  i2 = res(1,9+shift);
  i3 = res(1,10+shift);
  i4 = res(1,11+shift);
  i5 = res(1,12+shift);
  i6 = res(1,13+shift);
  op1 = res(1,14+shift);
  op2 = res(1,15+shift);
  op3 = res(1,16+shift);
  if px1 == 0
    x1= log(xdata(:,i1));
  elseif px1 == 1
    x1 = xdata(:,i1);
  elseif px1 == 2
    x1 = xdata(:,i1).^2;
  elseif px1 == 3
    x1 = xdata(:,i1).^3;
  elseif px1 == 4
    x1 = sqrt(xdata(:,i1));
  end

  if px2 == 0
    x2= log(xdata(:,i2));
  elseif px2 == 1
    x2 = xdata(:,i2);
  elseif px2 == 2
    x2 = xdata(:,i2).^2;
  elseif px2 == 3
    x2 = xdata(:,i2).^3;
  elseif px2 == 4
    x2 = sqrt(xdata(:,i2));
  end

  if px3 == 0
    x3= log(xdata(:,i3));
  elseif px3 == 1
    x3 = xdata(:,i3);
  elseif px3 == 2
    x3 = xdata(:,i3).^2;
  elseif px3 == 3
```

```
    x3 = xdata(:,i3).^3;
  elseif px3 == 4
    x3 = sqrt(xdata(:,i3));
  end

  if px4 == 0
    x4= log(xdata(:,i4));
  elseif px4 == 1
    x4 = xdata(:,i4);
  elseif px4 == 2
    x4 = xdata(:,i4).^2;
  elseif px4 == 3
    x4 = xdata(:,i4).^3;
  elseif px4 == 4
    x4 = sqrt(xdata(:,i4));
  end

  if px5 == 0
    x5= log(xdata(:,i5));
  elseif px5 == 1
    x5 = xdata(:,i5);
  elseif px5 == 2
    x5 = xdata(:,i5).^2;
  elseif px5 == 3
    x5 = xdata(:,i5).^3;
  elseif px5 == 4
    x5 = sqrt(xdata(:,i5));
  end

  if px6 == 0
    x6= log(xdata(:,i6));
  elseif px6 == 1
    x6 = xdata(:,i6);
  elseif px6 == 2
    x6 = xdata(:,i6).^2;
  elseif px6 == 3
    x6 = xdata(:,i6).^3;
  elseif px6 == 4
    x6 = sqrt(xdata(:,i6));
  end

  ycalc = res(2,2);
  for i=1:numIndpX
    ycalc = ycalc + res(2,2+i) * myfx(xdata(:,idxX(i)),pwrs(i));
  end
  k = 1 + numIndpX;
  if op1==1
    ycalc = ycalc + res(2,k+2) * x1.*x2;
  else
    ycalc = ycalc + res(2,k+2) * x1./x2;
  end

  if op2==1
    ycalc = ycalc + res(2,k+3) * x3.*x4;
  else
    ycalc = ycalc + res(2,k+3) * x3./x4;
  end
```

```
    if op3==1
    ycalc = ycalc + res(2,k+4) * x5.*x6;
  else
    ycalc = ycalc + res(2,k+4) * x5./x6;
  end
end
```

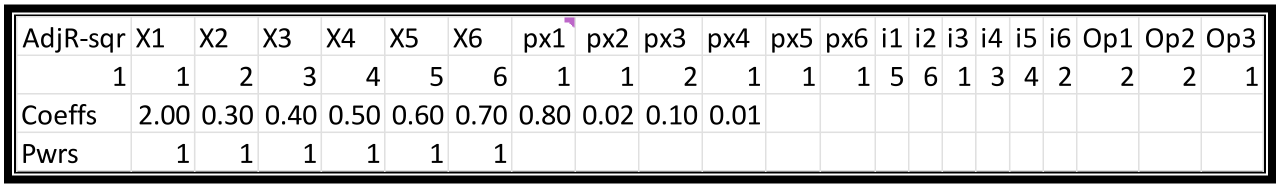
Program clem3b works with Excel file data3b.xlsx instead of data3.xlsx, because these two files have different results.

The console output from running program clem3b is shown below:

```
Program  clem3b

2025-Jan-2 20_12_49
Please wait ...

Excel file used is I:\DropBox\Dropbox\MATLAB\Clement3\data3b.xlsx
Diary file used is data3b_2025-Jan-2 20_12_49.txt
------------- Calculating Max Iters ----------------
Max iters = 23,887,872
------------- Maximizing Fx ----------------
Iter 86221, Fx = 9.978373e-01, X=[1, 1, 2, 2, 1, 1, 4, 3, 2, 1, 2, 1, 1, 1, 1, ]
Iter 86222, Fx = 9.978815e-01, X=[2, 1, 2, 2, 1, 1, 4, 3, 2, 1, 2, 1, 1, 1, 1, ]
Iter 86224, Fx = 9.978886e-01, X=[2, 2, 2, 2, 1, 1, 4, 3, 2, 1, 2, 1, 1, 1, 1, ]
Iter 86233, Fx = 9.979356e-01, X=[1, 1, 1, 2, 2, 1, 4, 3, 2, 1, 2, 1, 1, 1, 1, ]
Iter 86234, Fx = 9.979711e-01, X=[2, 1, 1, 2, 2, 1, 4, 3, 2, 1, 2, 1, 1, 1, 1, ]
Iter 88090, Fx = 9.980283e-01, X=[2, 1, 1, 2, 2, 1, 3, 2, 3, 1, 2, 1, 1, 1, 1, ]
Iter 88154, Fx = 9.980385e-01, X=[2, 1, 1, 2, 2, 1, 4, 2, 3, 1, 2, 1, 1, 1, 1, ]
Iter 102355, Fx = 9.981703e-01, X=[1, 2, 1, 1, 2, 1, 4, 3, 3, 2, 2, 1, 1, 1, 1, ]
Iter 102357, Fx = 9.984029e-01, X=[1, 1, 2, 1, 2, 1, 4, 3, 3, 2, 2, 1, 1, 1, 1, ]
Iter 102358, Fx = 9.984039e-01, X=[2, 1, 2, 1, 2, 1, 4, 3, 3, 2, 2, 1, 1, 1, 1, ]
Iter 104661, Fx = 9.984430e-01, X=[1, 1, 2, 1, 2, 1, 4, 3, 4, 2, 2, 1, 1, 1, 1, ]
Progress  5.00%
Progress 10.00%
Iter 3072281, Fx = 9.986657e-01, X=[1, 1, 1, 2, 2, 1, 5, 3, 2, 1, 2, 1, 2, 1, 1, ]
Iter 3076901, Fx = 9.986664e-01, X=[1, 1, 2, 1, 1, 2, 5, 3, 4, 1, 2, 1, 2, 1, 1, ]
Iter 3081481, Fx = 9.986678e-01, X=[1, 1, 1, 2, 1, 1, 5, 3, 6, 1, 2, 1, 2, 1, 1, ]
Iter 3081497, Fx = 9.986868e-01, X=[1, 1, 1, 2, 2, 1, 5, 3, 6, 1, 2, 1, 2, 1, 1, ]
Iter 3081505, Fx = 9.987025e-01, X=[1, 1, 1, 1, 1, 2, 5, 3, 6, 1, 2, 1, 2, 1, 1, ]
Iter 3088140, Fx = 9.997840e-01, X=[2, 2, 1, 2, 1, 1, 1, 3, 3, 2, 2, 1, 2, 1, 1, ]
Iter 3088142, Fx = 9.999999e-01, X=[2, 1, 2, 2, 1, 1, 1, 3, 3, 2, 2, 1, 2, 1, 1, ]
Iter 3088150, Fx = 1.000000e+00, X=[2, 1, 2, 1, 2, 1, 1, 3, 3, 2, 2, 1, 2, 1, 1, ]
Iter 3090450, Fx = 1.000000e+00, X=[2, 1, 1, 1, 2, 1, 1, 3, 4, 2, 2, 1, 2, 1, 1, ]
Iter 3173394, Fx = 1.000000e+00, X=[2, 1, 1, 1, 2, 1, 1, 3, 4, 2, 3, 1, 2, 1, 1, ]
Progress 15.00%
Progress 20.00%
Progress 25.00%
Progress 30.00%
Progress 35.00%
Progress 40.00%
Iter 9709830, Fx = 1.000000e+00, X=[2, 1, 2, 1, 1, 1, 1, 3, 3, 1, 4, 2, 2, 2, 1, ]
Iter 9734021, Fx = 1.000000e+00, X=[1, 1, 2, 1, 1, 1, 1, 6, 1, 3, 4, 2, 2, 2, 1, ]
Iter 9734023, Fx = 1.000000e+00, X=[1, 2, 2, 1, 1, 1, 1, 6, 1, 3, 4, 2, 2, 2, 1, ]
Iter 9734277, Fx = 1.000000e+00, X=[1, 1, 2, 1, 1, 1, 5, 6, 1, 3, 4, 2, 2, 2, 1, ]
Progress 45.00%
Progress 50.00%
Progress 55.00%
Progress 60.00%
Progress 65.00%
Progress 70.00%
Progress 75.00%
Progress 80.00%
Progress 85.00%
Progress 90.00%
Progress 95.00%
Progress 100.00%
Regression model
Adjusted R-square = 1.000000000000

mdl = 


Linear regression model:
    y ~ 1 + x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9

Estimated Coefficients:
                       Estimate          SE    tStat    pValue
                   _________________    __    _____    ______

    (Intercept)      2.00000000000167    0      Inf       0   
    x1               0.29999999999997    0      Inf       0   
    x2              0.399999999999999    0      Inf       0   
    x3              0.500000000000057    0      Inf       0   
    x4              0.600000000002467    0      Inf       0   
    x5                            0.7    0      Inf       0   
    x6              0.800000000000004    0      Inf       0   
    x7              0.019999999999992    0      Inf       0   
    x8                            0.1    0      Inf       0   
    x9                           0.01    0      Inf       0   


Number of observations: 250, Error degrees of freedom: 240
R-squared: 1,  Adjusted R-Squared: 1
F-statistic vs. constant model: 1.35e+32, p-value = 0


Best model is:
Y = (2)
   + (0.3)*X1
   + (0.4)*X2
   + (0.5)*X3
   + (0.6)*X4
   + (0.7)*X5
   + (0.8)*X6
   + (0.02)*X5/X6
   + (0.1)*X1^2/X3
   + (0.01)*X4*X2
Done!
```

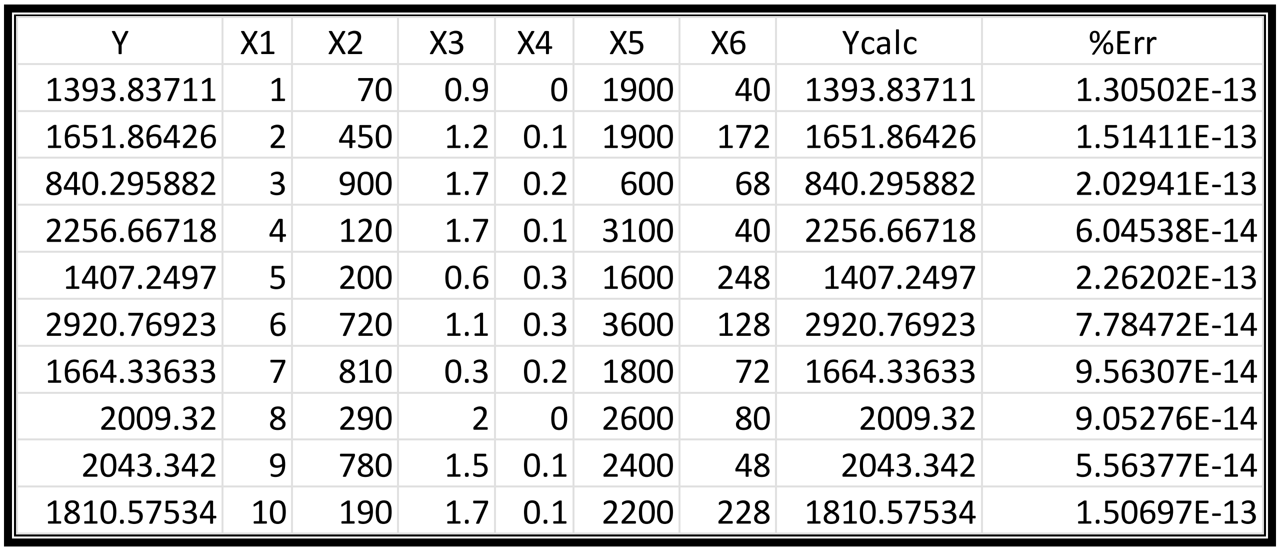
Program clem3b was able to find the correct best model, thanks to the virtual nested loops.

Figures (8.1) and (8.2) show the output as it appears in sheets **Results** and **Project** of the Excel file data3b.xlsx. The very small percent errors in Figure (8.2) confirm that the program clem3b obtained the correct best model.



AdjR-sqr	X1	X2	X3	X4	X5	X6	px1	px2	px3	px4	px5	px6	i1	i2	i3	i4	i5	i6	Op1	Op2	Op3
1	1	2	3	4	5	6	1	1	2	1	1	1	5	6	1	3	4	2	2	2	1
Coeffs	2.00	0.30	0.40	0.50	0.60	0.70	0.80	0.02	0.10	0.01											
Pwrs	1	1	1	1	1	1															

*Figure 8.1. The sheet **Results** in file data3b.xlsx.*



Y	X1	X2	X3	X4	X5	X6	Ycalc	%Err
1393.83711	1	70	0.9	0	1900	40	1393.83711	1.30502E-13
1651.86426	2	450	1.2	0.1	1900	172	1651.86426	1.51411E-13
840.295882	3	900	1.7	0.2	600	68	840.295882	2.02941E-13
2256.66718	4	120	1.7	0.1	3100	40	2256.66718	6.04538E-14
1407.2497	5	200	0.6	0.3	1600	248	1407.2497	2.26202E-13
2920.76923	6	720	1.1	0.3	3600	128	2920.76923	7.78472E-14
1664.33633	7	810	0.3	0.2	1800	72	1664.33633	9.56307E-14
2009.32	8	290	2	0	2600	80	2009.32	9.05276E-14
2043.342	9	780	1.5	0.1	2400	48	2043.342	5.56377E-14
1810.57534	10	190	1.7	0.1	2200	228	1810.57534	1.50697E-13

*Figure 8.2. The sheet **Project** in file data3b.xlsx.*

9/Optimizing Models with Triple Cross-Product Terms

This section, and the next one, work with simple models containing a single triple cross-product term, defined as:

$$Y = a + \sum_{i=1}^{n} b_i\, X_i^{px(i)} + \; X_m^{qx1} op1\, X_j^{qx2} op2\, X_k^{qx3} \qquad (18)$$

The identifiers op1 and op2 are the * operator if they are equal to 1 and are the / operator if they are not equal to 1. The independent variables X_m, X_j, and X_k come from a set of n variables. The first summation can include some or all the n variables. The triple cross-product term has three variables selected from the n variables. I define a limited version of the triple cross-product heteronomial model to reduce the number of optimized variables.

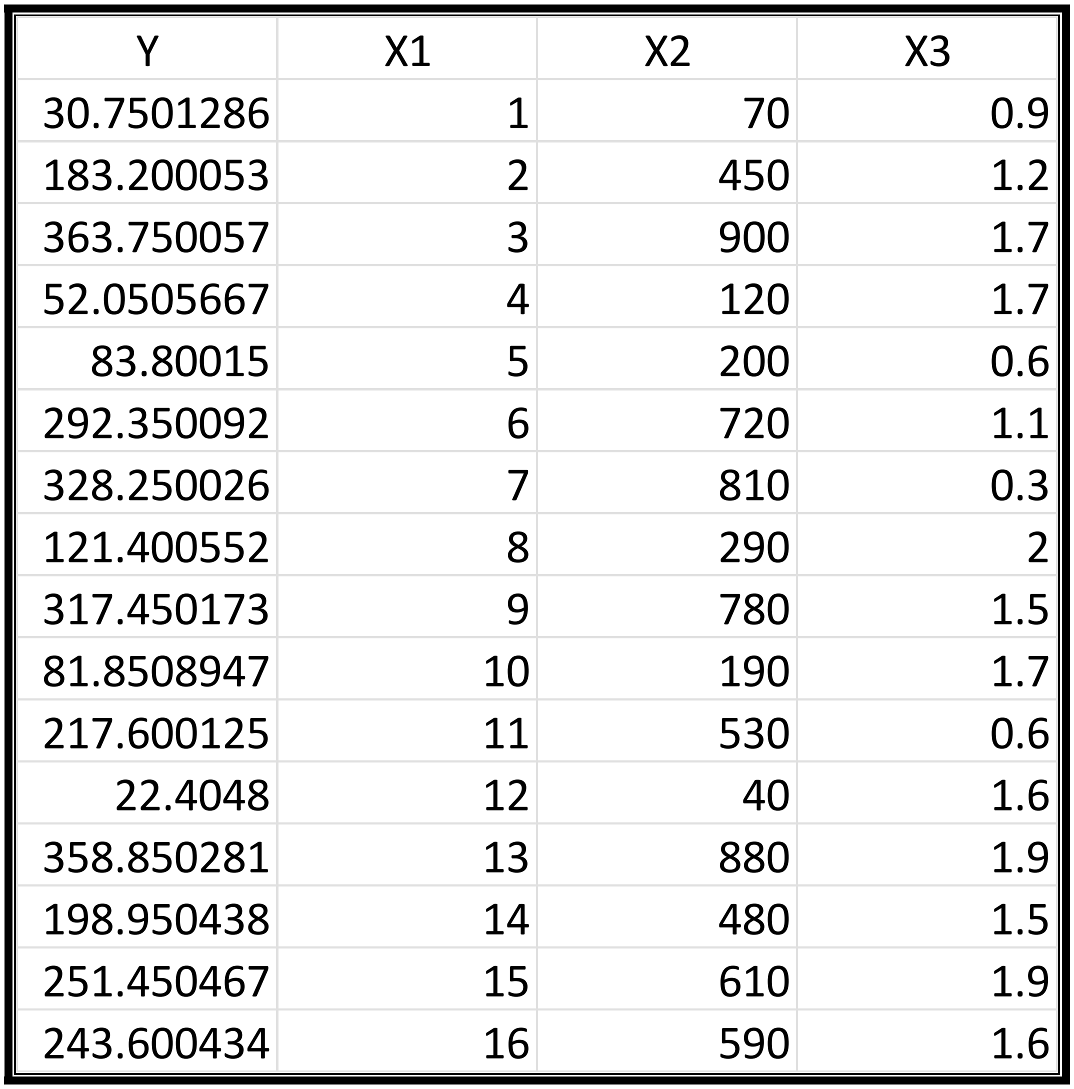
The actual model studied here is:

$$Y = a + \sum_{i=1}^{n3} b_i\, X_i^{px(i)} + X_1^{qx1} op1\, X_2^{qx2} op2\, X_3^{qx3} \qquad (19)$$

In keeping with the previous models in this study, I will optimize the powers, selected variables, and operations in the cross-product term only. You get to select the powers of the single-variable terms. The actual model is:

$$Y = 2 + 0.3{*}X_1 + 0.4{*}X_2 + 0.5{*}X_3 + 0.01{*}X_1/X_2{*}X_3 \qquad (20)$$

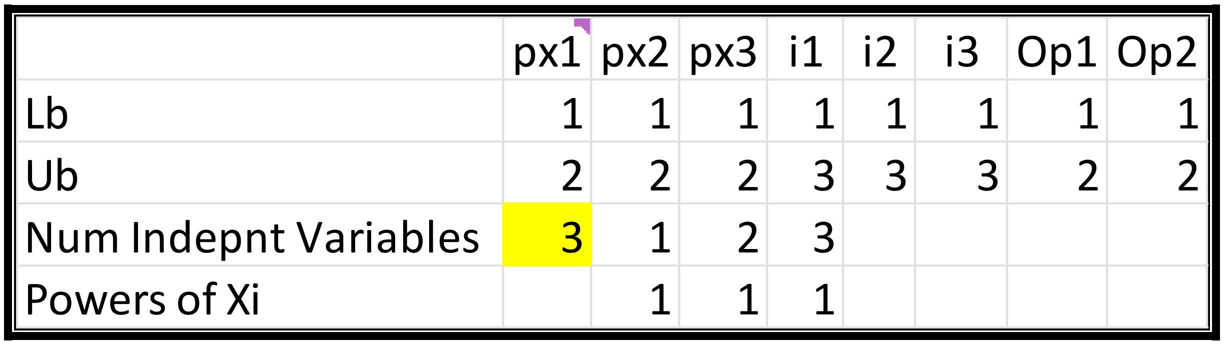
Figure 9.1 shows a partial view of the sheet **Data** in file data4.xlsx.



Y	X1	X2	X3
30.7501286	1	70	0.9
183.200053	2	450	1.2
363.750057	3	900	1.7
52.0505667	4	120	1.7
83.80015	5	200	0.6
292.350092	6	720	1.1
328.250026	7	810	0.3
121.400552	8	290	2
317.450173	9	780	1.5
81.8508947	10	190	1.7
217.600125	11	530	0.6
22.4048	12	40	1.6
358.850281	13	880	1.9
198.950438	14	480	1.5
251.450467	15	610	1.9
243.600434	16	590	1.6

*Figure 9.1. A partial view of the sheet **Data** in file data4.xlsx.*

Figure 9.2 shows the sheet **Transf** in file data4.xlsx. This sheet shows that the model has three single-term variables and one cross-product term with powers px1, px2, and px3 all in the range of 1 to 2.



	px1	px2	px3	i1	i2	i3	Op1	Op2
Lb	1	1	1	1	1	1	1	1
Ub	2	2	2	3	3	3	2	2
Num Indepnt Variables	3	1	2	3				
Powers of Xi		1	1	1				

*Figure 9.2. The sheet **Transf** in file data4.xlsx.*

Here is the source code for program clem4:

```
clc
close all
clear all

global mdl
global xdata
global ydata
global idxX
global pwrs

warning("off")
dt = datetime('now','TimeZone','local','Format','y-MMM-d HH_mm_ss');
dtstr = string(dt);
filename = strcat(pwd,'\data4.xlsx');
delete("data4_*.txt");
outFile = strcat("data4_",dtstr,".txt");
diary(outFile)

fprintf("Program  clem4\n\n");
fprintf("%s\n", dtstr);
fprintf("Please wait ...\n\n");
fprintf("Excel file used is %s\n", filename);
fprintf("Diary file used is %s\n", outFile);
xyData = readmatrix(filename,'Sheet','Data');
[maxrows,~] = size(xyData);
ydata = xyData(:,1);
xdata = xyData(:,2:end);
xyTransf = readmatrix(filename,'Sheet','Transf');
xyTransf = xyTransf(:,2:end);
[n,nXVars] = size(xdata);
numIndpX = xyTransf(3,1);
idxX = zeros(1,numIndpX);
pwrs = zeros(1,numIndpX);
for i=1:numIndpX
  idxX(i) = xyTransf(3,1+i);
  pwrs(i) = xyTransf(4,1+i);
  if isnan(pwrs(i)), pwrs(i) = 1; end
end

Lb = xyTransf(1,:);
```

```
Ub = 1+xyTransf(2,:);
nVars = length(Ub);
MaxIters = 3000;
MaxPop= 300;
itersToReset = 300;
MaxTries = 5;
numTry = 0;
while true
  numTry = numTry + 1;
  if numTry > MaxTries
    fprintf("Optimization failed\n");
    return;
  end
  if numTry > 1
    fprintf("Optimization attempt # %d\n", numTry)
  end
  [bestX,gBestCost,bestMdl] = pso(@myFit,Lb,Ub,MaxPop,MaxIters,itersToReset);
  px1 = bestX(1);
  px2 = bestX(2);
  px3 = bestX(3);

  i1= bestX(1+numIndpX);
  i2= bestX(2+numIndpX);
  i3= bestX(3+numIndpX);


  if (i1 == i2) || (i2 == i3), continue, end
  % compare X1,X2 with X3,X4
  if ((i1==i3) && (px1==px3)) || ((i2==i3) && (px2==px3)), continue, end
  break;
end
mdl = bestMdl;
AdjR2 = mdl.Rsquared.Adjusted;
fprintf("Regression model\n")
fprintf("Adjusted R-square = %14.10f\n", AdjR2);
format long
mdl
bx = bestX;
bestX = [AdjR2];
for i = 1:numIndpX
    bestX = [bestX idxX(i)];
end
bestX = [bestX bx];
k =length(bestX);
res = zeros(3,k);
res(1,:) = bestX;
k2 = 2+numIndpX;
res(2,2:k2+1) = mdl.Coefficients{:,1}';
res(3,3:2+numIndpX) = pwrs;
writematrix(res(1,:), filename, 'Sheet', 'Results', 'Range', 'A2:Z2');
writematrix(res(2,2:k2+1), filename, 'Sheet', 'Results', 'Range', 'B3:Z3');
writematrix(res(3,3:2+numIndpX), filename, 'Sheet', 'Results', 'Range',
'B4:Z4');
%
% now do the % projections
ycalc = project(res);
percErr = 100.*(ycalc-ydata)./ydata;
```

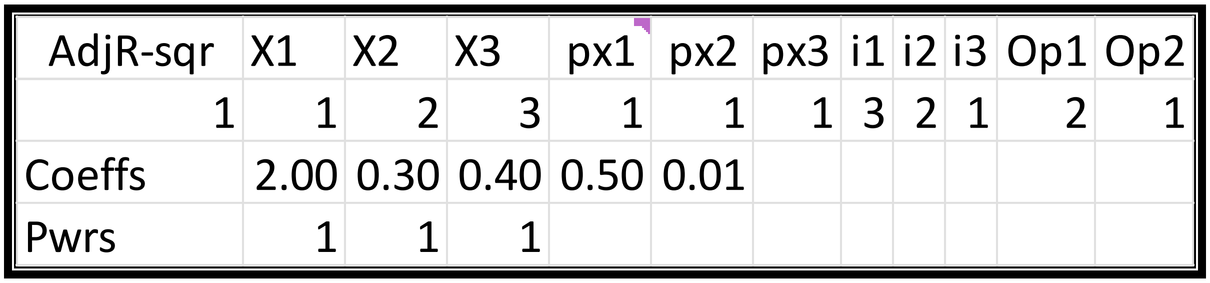
```
ProjMat = [ydata xdata ycalc percErr];
writematrix(ProjMat, filename, 'Sheet', 'Project', 'Range', strcat("A2:Z",
num2str(1+length(ydata))));

k2=length(bestX);
if bestX(k2-1) == 1
  sOp1 = "*";
else
  sOp1 = "/";
end
if bestX(k2) == 1
  sOp2 = "*";
else
  sOp2 = "/";
end

i=2+numIndpX;
j=i+numIndpX;
sJ = num2str(bestX(j));
if bestX(i) == 0
  sx1 = strcat("ln(X", sJ,")");
elseif bestX(i) == 1
  sx1 = strcat("X", sJ);
elseif bestX(i) == 4
  sx1 = strcat("sqrt(X", sJ,")");
elseif bestX(i) > 1
  sx1 = strcat("X", sJ,"^", num2str(bestX(i)));
end


i=i+1;
j=j+1;
sJ = num2str(bestX(j));
if bestX(i) == 1
  sx2 = strcat("X", sJ);
elseif bestX(i) == 0
  sx2 = strcat("ln(X", sJ,")");
elseif bestX(i) == 4
  sx2 = strcat("sqrt(X", sJ,")");
elseif bestX(i) > 1
  sx2 = strcat("X", sJ,"^", num2str(bestX(i)));
end

i=i+1;
j=j+1;
sJ = num2str(bestX(j));
if bestX(i) == 1
  sx3 = strcat("X", sJ);
elseif bestX(i) == 0
  sx3 = strcat("ln(X", sJ,")");
elseif bestX(i) == 4
  sx3 = strcat("sqrt(X", sJ,")");
elseif bestX(i) > 1
  sx3 = strcat("X", sJ,"^", num2str(bestX(i)));
end
```

```
s = strcat("Y = (", num2str(mdl.Coefficients{1,1}),")");
fprintf("\n\nBest model is:\n%s\n", s)
for i=1:numIndpX
  j = idxX(i);
  sJ= num2str(j);
  if pwrs(i)==1
    s = strcat("   + (", num2str(mdl.Coefficients{1+i,1}),")*X", sJ);
  elseif pwrs(i)==0
    s = strcat("   + (", num2str(mdl.Coefficients{1+i,1}),")*ln(X", sJ,")");
  else
    sp = num2str(pwrs(i));
    s = strcat("   + (", num2str(mdl.Coefficients{1+i,1}),")*X", sJ,"^",sp);
  end
  fprintf("%s\n", s)
end
s = strcat("   + (",num2str(mdl.Coefficients{2+numIndpX,1}), ")*", sx1, sOp1,
sx2, sOp2, sx3);
fprintf("%s\n", s)

format short
fprintf("Done!\n");
diary off
beep on
for i=1:3
  beep;
  pause(1);
end

function x = myfx(x,p)
  if p==0
    x = log(x);
  elseif p==2
    x = x.^2;
  elseif p==3
    x = x.^3;
  elseif p==4
    x = sqrt(x);
  end
end

function r = myFit(coeff)
  global mdl
  global xdata
  global ydata
  global idxX
  global pwrs

  [~,nXVars] = size(xdata);
  numIndpX = length(idxX);
  y = ydata;
  px1 = coeff(1);
  px2 = coeff(2);
  px3 = coeff(3);
  i1 = coeff(4);
  i2 = coeff(5);
  i3 = coeff(6);
  op1 = coeff(7);
  op2 = coeff(8);
  if i1 > nXVars || i2 > nXVars || i3 > nXVars
    r = 1;
    return
  end

  if (i1==i2) || (i1==i3) || (i2==i3)
    r = 1;
    return;
  end
  % compare X1,X2 with X3,X4
  if ((i1==i2) && (px1==px2)) || ((i1==i3) && (px1==px3)) || ((i2==i3) &&
(px2==px3))
    r = 1;
    return;
  end

  x1 = myfx(xdata(:,i1),px1);
  x2 = myfx(xdata(:,i2),px2);
  x3 = myfx(xdata(:,i3),px3);
  if op1==1
    x12 = x1.*x2;
  else
    x12 = x1./x2;
  end
  if op2==1
    x123 = x12.*x3;
  else
    x123  = x12./x3;
  end
  X = [];
  for i=1:numIndpX
    X = [X myfx(xdata(:,idxX(i)),pwrs(i))];
  end
  X = [X x123 ];
  mdl = fitlm(X,y);
  r = 1 - mdl.Rsquared.Adjusted;
end

function ycalc = project(res)
  global xdata
  global idxX
  global pwrs

  numIndpX = length(idxX);
  shift = 1 + numIndpX;
  px1 = res(1,1+shift);
  px2 = res(1,2+shift);
  px3 = res(1,3+shift);
  i1 = res(1,4+shift);
  i2 = res(1,5+shift);
  i3 = res(1,5+shift);
  op1 = res(1,7+shift);
  op2 = res(1,8+shift);

  if px1 == 0
    x1= log(xdata(:,i1));
  elseif px1 == 1
    x1 = xdata(:,i1);
  elseif px1 == 2
    x1 = xdata(:,i1).^2;
  elseif px1 == 3
    x1 = xdata(:,i1).^3;
  elseif px1 == 4
    x1 = sqrt(xdata(:,i1));
  end

  if px2 == 0
    x2= log(xdata(:,i2));
  elseif px2 == 1
    x2 = xdata(:,i2);
  elseif px2 == 2
    x2 = xdata(:,i2).^2;
  elseif px2 == 3
    x2 = xdata(:,i2).^3;
  elseif px2 == 4
    x2 = sqrt(xdata(:,i2));
  end

  if px3 == 0
    x3= log(xdata(:,i3));
  elseif px3 == 1
    x3 = xdata(:,i3);
  elseif px3 == 2
    x3 = xdata(:,i3).^2;
  elseif px3 == 3
    x3 = xdata(:,i3).^3;
  elseif px3 == 4
    x3 = sqrt(xdata(:,i3));
  end

  if op1==1
    x12 = x1.*x2;
  else
    x12 = x1./x2;
  end
  if op2==1
    x123 = x12.*x3;
  else
    x123  = x12./x3;
  end


  ycalc = res(2,2);
  for i=1:numIndpX
    ycalc = ycalc + res(2,2+i) * myfx(xdata(:,idxX(i)),pwrs(i));
  end
  k = 1 + numIndpX;
  ycalc = ycalc + res(2,end) * x123;
end
```

The program clem4 is a variation of clem3 that accommodates the triple cross-product term. The console output for running the program is:

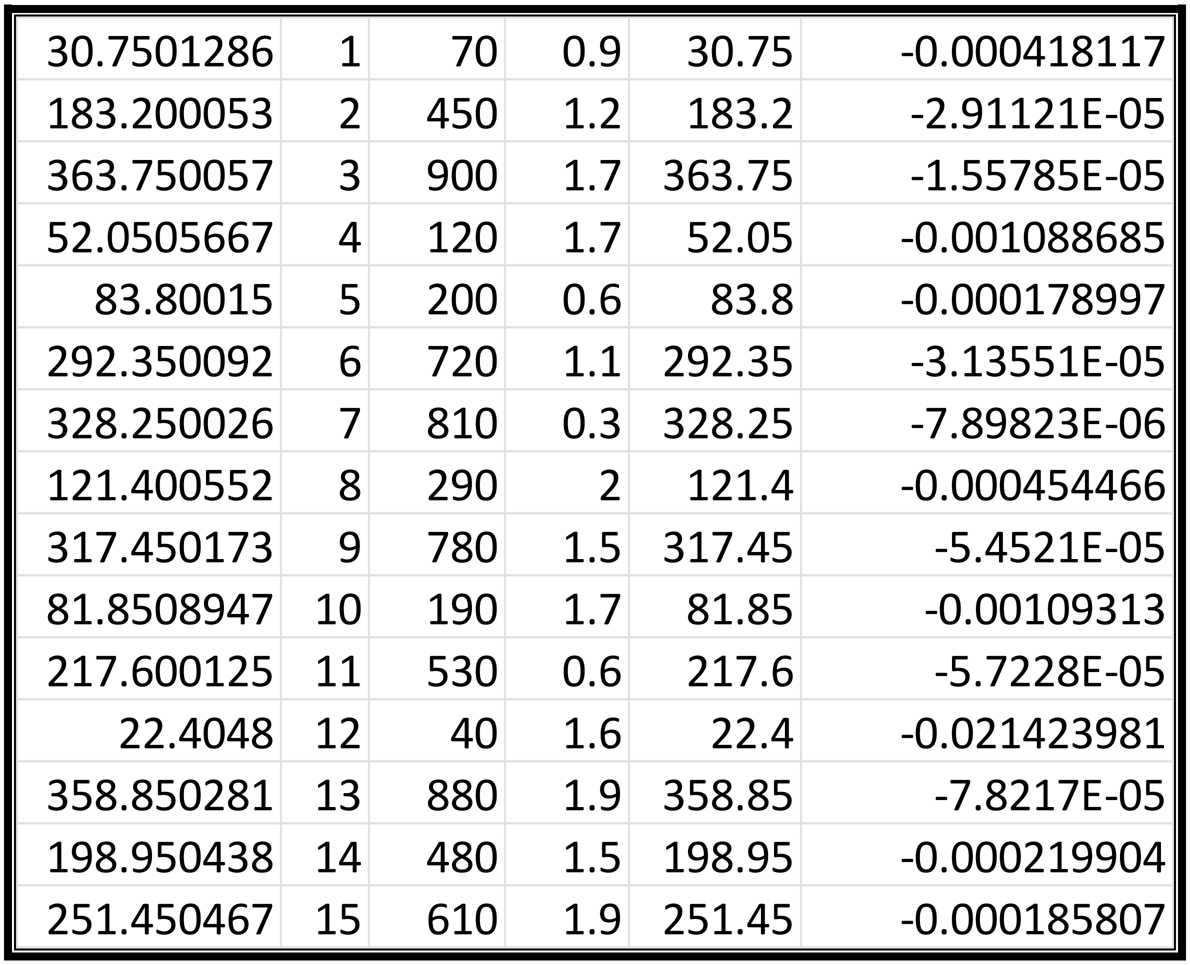
```
Program  clem4

2025-Jan-2 13_57_08
Please wait ...

Excel file used is C:\Users\nsham\Desktop\Clement4\data4.xlsx
Diary file used is data4_2025-Jan-2 13_57_08.txt
Best Fx = 0.000000e+00, Best X =[1, 1, 1, 3, 2, 1, 2,  1]
Regression model
Adjusted R-square =   1.0000000000

mdl =


Linear regression model:
    y ~ 1 + x1 + x2 + x3 + x4

Estimated Coefficients:
                         Estimate                SE                  tStat          pValue
                    ________________    ___________________    _______________    ______

    (Intercept)      1.99999999999843    2.81895802662465e-05    70948.2007574687      0
    x1              0.300000000000015      3.742249229393e-07    801656.922376265      0
    x2              0.400000000000001    2.87675921868015e-08    13904535.2632439      0
    x3              0.499999999999857    3.72566105922319e-06    134204.371265085      0
    x4             0.0100000000001017    9.80756494366592e-06    1019.62108408572      0


Number of observations: 250, Error degrees of freedom: 245
Root Mean Squared Error: 0.000125
R-squared: 1,  Adjusted R-Squared: 1
F-statistic vs. constant model: 1.87e+32, p-value = 0


Best model is:
Y = (2)
   + (0.3)*X1
   + (0.4)*X2
   + (0.5)*X3
   + (0.01)*X3/X2*X1
Done!
```

The above output shows that program clem4 was able to find the correct model. Figures 9.3 and 9.4 show the sheets **Results** and **Project** (partial view). The small percentage errors in Figure 9.4 confirm that the model obtained is the correct one.



AdjR-sqr	X1	X2	X3	px1	px2	px3	i1	i2	i3	Op1	Op2
1	1	2	3	1	1	1	3	2	1	2	1
Coeffs	2.00	0.30	0.40	0.50	0.01						
Pwrs	1	1	1								

*Figure 9.3. The sheet **Results** in file data4.xlsx.*



30.7501286	1	70	0.9	30.75	-0.000418117
183.200053	2	450	1.2	183.2	-2.91121E-05
363.750057	3	900	1.7	363.75	-1.55785E-05
52.0505667	4	120	1.7	52.05	-0.001088685
83.80015	5	200	0.6	83.8	-0.000178997
292.350092	6	720	1.1	292.35	-3.13551E-05
328.250026	7	810	0.3	328.25	-7.89823E-06
121.400552	8	290	2	121.4	-0.000454466
317.450173	9	780	1.5	317.45	-5.4521E-05
81.8508947	10	190	1.7	81.85	-0.00109313
217.600125	11	530	0.6	217.6	-5.7228E-05
22.4048	12	40	1.6	22.4	-0.021423981
358.850281	13	880	1.9	358.85	-7.8217E-05
198.950438	14	480	1.5	198.95	-0.000219904
251.450467	15	610	1.9	251.45	-0.000185807

*Figure 9.4. A partial view of sheet **Project** in file data4.xlsx.*

10/Using Virtual Nested Loop to Find the Best Models with Triple Cross-Product Terms

In this section I present the calculations for the triple cross-product model using virtual nested loops to search for the best model parameters. Program clewm4b uses Excel file data4b.xlsx. It has the same sheets **Data** and **Transf** as in Figures 9.1 and 9.2. Here is the source code for program clem4b:

```
clc
close all
clear all

global mdl
global xdata
global ydata
global idxX
global pwrs
```

```
warning("off")
dt = datetime('now','TimeZone','local','Format','y-MMM-d HH_mm_ss');
dtstr = string(dt);
filename = strcat(pwd,'\data4b.xlsx');
delete("data4b_*.txt");
outFile = strcat("data4b_",dtstr,".txt");
diary(outFile)

fprintf("Program  clem4b\n\n");
fprintf("%s\n", dtstr);
fprintf("Please wait ...\n\n");
fprintf("Excel file used is %s\n", filename);
fprintf("Diary file used is %s\n", outFile);
xyData = readmatrix(filename,'Sheet','Data');
[maxrows,~] = size(xyData);
ydata = xyData(:,1);
xdata = xyData(:,2:end);
xyTransf = readmatrix(filename,'Sheet','Transf');
xyTransf = xyTransf(:,2:end);
[n,nXVars] = size(xdata);
numIndpX = xyTransf(3,1);
idxX = zeros(1,numIndpX);
pwrs = zeros(1,numIndpX);
for i=1:numIndpX
  idxX(i) = xyTransf(3,1+i);
  pwrs(i) = xyTransf(4,1+i);
  if isnan(pwrs(i)), pwrs(i) = 1; end
end

Lb = xyTransf(1,:);
Ub = xyTransf(2,:);
nVars = length(Ub);
iFrom = Lb;
iTo = Ub;
iStep = 1 + zeros(1,nVars);
idx= Lb;
bStop = false;
iter = 0;
fprintf("------------- Calculating Max Iters ----------------\n");
while ~bStop
  iter = iter +1;
  % Start implementing the quasiâ€"nested loops
  idx(1) = idx(1) + iStep(1);
  if idx(1) > iTo(1)
    idx(1) = iFrom(1);
    for i = 2:nVars
      idx(i) = idx(i) + iStep(i);
      if idx(i) > iTo(i) && i < nVars
        idx(i) = iFrom(i);
      elseif idx(i) > iTo(i) && i == nVars
        bStop = true;
      else
        break
      end
    end
  end
```

```
end
str = num2str(iter);
str = fliplr(regexprep(fliplr(str), '(\d+\.)?(\d{3})(?=\S+)', '$1$2,'));
fprintf("Max iters = %s\n", str);
MaxIters = iter;
FivePercent = fix(iter/20);

iFrom = Lb;
iTo = Ub;
iStep = 1 + zeros(1,nVars);
idx= Lb;
bestX = zeros(1,nVars);
bestFx = 0;
% -------------------------------- start main loop
bStop = false;
iter = 0;
fprintf("------------- Maximizing Fx ----------------\n");
while ~bStop
  iter = iter +1;
  if mod(iter, FivePercent) == 0
    percent = 100*iter/MaxIters;
    fprintf("Progress %5.2f%%\n", percent);
  end

  r2adj = myFit(idx);
  if r2adj > bestFx
    bestFx = r2adj;
    bestX = idx;
    bestMdl = mdl;
    fprintf("Iter %d, Fx = %e, X=[", iter, bestFx)
    fprintf("%d, ", bestX);
    fprintf("]\n");
  end
  % Start implementing the quasiâ€"nested loops
  idx(1) = idx(1) + iStep(1);
  if idx(1) > iTo(1)
    idx(1) = iFrom(1);
    for i = 2:nVars
      idx(i) = idx(i) + iStep(i);
      if idx(i) > iTo(i) && i < nVars
        idx(i) = iFrom(i);
      elseif idx(i) > iTo(i) && i == nVars
        bStop = true;
      else
        break
      end
    end
  end
end
fprintf("Regression model\n")
format long
AdjR2 = bestMdl.Rsquared.Adjusted;
fprintf("Adjusted R-square = %14.12f\n", AdjR2);
mdl = bestMdl
bx = bestX;
bestX = [AdjR2];
for i = 1:numIndpX
```

```
    bestX = [bestX idxX(i)];
end
bestX = [bestX bx];
k =length(bestX);
res = zeros(3,k);
res(1,:) = bestX;
k2 = 2+numIndpX;
res(2,2:k2+1) = mdl.Coefficients{:,1}';
res(3,3:2+numIndpX) = pwrs;
writematrix(res(1,:), filename, 'Sheet', 'Results', 'Range', 'A2:Z2');
writematrix(res(2,2:k2+1), filename, 'Sheet', 'Results', 'Range', 'B3:Z3');
writematrix(res(3,3:2+numIndpX), filename, 'Sheet', 'Results', 'Range',
'B4:Z4');
%
% now do the % projections
ycalc = project(res);
percErr = 100.*(ycalc-ydata)./ydata;
ProjMat = [ydata xdata ycalc percErr];
writematrix(ProjMat, filename, 'Sheet', 'Project', 'Range', strcat("A2:Z",
num2str(1+length(ydata))));

i =2+numIndpX;
j=i+numIndpX;
sJ = num2str(bestX(j));
if bestX(i) == 0
  sx1 = strcat("ln(X", sJ,")");
elseif bestX(i) == 1
  sx1 = strcat("X", sJ);
elseif bestX(i) == 4
  sx1 = strcat("sqrt(X", sJ,")");
elseif bestX(i) > 1
  sx1 = strcat("X", sJ,"^", num2str(bestX(i)));
end
ix1 = sJ;
k2=length(bestX);
if bestX(k2-1) == 1
  sOp1 = "*";
else
  sOp1 = "/";
end
if bestX(k2) == 1
  sOp2 = "*";
else
  sOp2 = "/";
end

i=i+1;
j=j+1;
sJ = num2str(bestX(j));
if bestX(i) == 1
  sx2 = strcat("X", sJ);
elseif bestX(i) == 0
  sx2 = strcat("ln(X", sJ,")");
elseif bestX(i) == 4
  sx2 = strcat("sqrt(X", sJ,")");
elseif bestX(i) > 1
  sx2 = strcat("X", sJ,"^", num2str(bestX(i)));
```

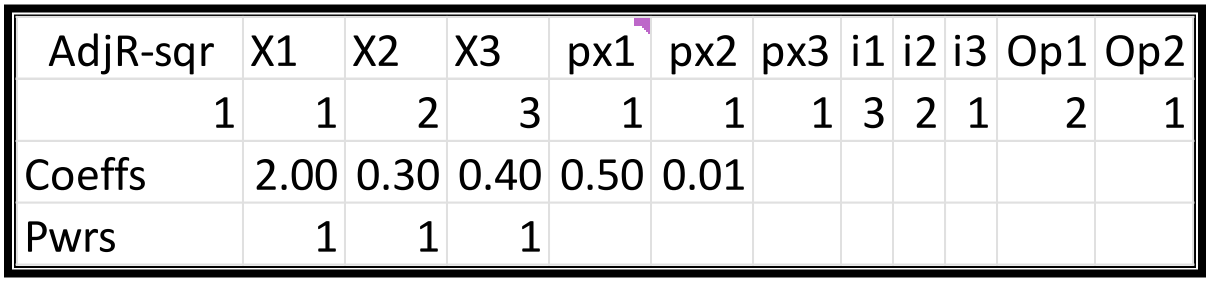
```
end
ix2 = sJ;
i=i+1;
j=j+1;
sJ = num2str(bestX(j));
if bestX(i) == 1
  sx3 = strcat("X", sJ);
elseif bestX(i) == 0
  sx3 = strcat("ln(X", sJ,")");
elseif bestX(i) == 4
  sx3 = strcat("sqrt(X", sJ,")");
elseif bestX(i) > 1
  sx3 = strcat("X", sJ,"^", num2str(bestX(i)));
end

s = strcat("Y = (", num2str(mdl.Coefficients{1,1}),")");
fprintf("\n\nBest model is:\n%s\n", s)
for i=1:numIndpX
  j = idxX(i);
  sJ= num2str(j);
  if pwrs(i)==1
    s = strcat("   + (", num2str(mdl.Coefficients{1+i,1}),")*X", sJ);
  elseif pwrs(i)==0
    s = strcat("   + (", num2str(mdl.Coefficients{1+i,1}),")*ln(X", sJ,")");
  else
    sp = num2str(pwrs(i));
    s = strcat("   + (", num2str(mdl.Coefficients{1+i,1}),")*X", sJ,"^",sp);
  end
  fprintf("%s\n", s)
end

s = strcat("   + (",num2str(mdl.Coefficients{end,1}), ")*", sx1, sOp1, sx2,
sOp2, sx3);
fprintf("%s\n", s)

format short
fprintf("Done!\n");
diary off
beep on
for i=1:3
  beep;
  pause(1);
end

function x = myfx(x,p)
  if p==0
    x = log(x);
  elseif p==2
    x = x.^2;
  elseif p==3
    x = x.^3;
  elseif p==4
    x = sqrt(x);
  end
end

function r = myFit(coeff)
```

```
  global mdl
  global xdata
  global ydata
  global idxX
  global pwrs

  [~,nXVars] = size(xdata);
  numIndpX = length(idxX);
  y = ydata;
  px1 = coeff(1);
  px2 = coeff(2);
  px3 = coeff(3);
  i1 = coeff(4);
  i2 = coeff(5);
  i3 = coeff(6);
  op1 = coeff(7);
  op2 = coeff(8);
  if i1 > nXVars || i2 > nXVars || i3 > nXVars
    r = 1;
    return
  end

  if (i1==i2) || (i1==i3) || (i2==i3)
    r = 0;
    return;
  end
  % compare X1,X2 with X3,X4
  if ((i1==i2) && (px1==px2)) || ((i1==i3) && (px1==px3)) || ((i2==i3) &&
(px2==px3))
    r = 0;
    return;
  end

  x1 = myfx(xdata(:,i1),px1);
  x2 = myfx(xdata(:,i2),px2);
  x3 = myfx(xdata(:,i3),px3);
  if op1==1
    x12 = x1.*x2;
  else
    x12 = x1./x2;
  end
  if op2==1
    x123 = x12.*x3;
  else
    x123  = x12./x3;
  end
  X = [];
  for i=1:numIndpX
    X = [X myfx(xdata(:,idxX(i)),pwrs(i))];
  end
  X = [X x123];
  mdl = fitlm(X,y);
  r = mdl.Rsquared.Adjusted;
end

function ycalc = project(res)
  global xdata
  global idxX
  global pwrs

  numIndpX = length(idxX);
  shift = 1 + numIndpX;
  px1 = res(1,1+shift);
  px2 = res(1,2+shift);
  px3 = res(1,3+shift);
  i1 = res(1,4+shift);
  i2 = res(1,5+shift);
  i3 = res(1,6+shift);
  op1 = res(1,7+shift);
  op2 = res(1,8+shift);

  if px1 == 0
    x1= log(xdata(:,i1));
  elseif px1 == 1
    x1 = xdata(:,i1);
  elseif px1 == 2
    x1 = xdata(:,i1).^2;
  elseif px1 == 3
    x1 = xdata(:,i1).^3;
  elseif px1 == 4
    x1 = sqrt(xdata(:,i1));
  end

  if px2 == 0
    x2= log(xdata(:,i2));
  elseif px2 == 1
    x2 = xdata(:,i2);
  elseif px2 == 2
    x2 = xdata(:,i2).^2;
  elseif px2 == 3
    x2 = xdata(:,i2).^3;
  elseif px2 == 4
    x2 = sqrt(xdata(:,i2));
  end

  if px3 == 0
    x3= log(xdata(:,i3));
  elseif px3 == 1
    x3 = xdata(:,i3);
  elseif px3 == 2
    x3 = xdata(:,i3).^2;
  elseif px3 == 3
    x3 = xdata(:,i3).^3;
  elseif px3 == 4
    x3 = sqrt(xdata(:,i3));
  end

  if op1==1
    x12 = x1.*x2;
  else
    x12 = x1./x2;
  end
  if op2==1
    x123 = x12.*x3;
  else
    x123  = x12./x3;
  end

  ycalc = res(2,2);
  for i=1:numIndpX
    ycalc = ycalc + res(2,2+i) * myfx(xdata(:,idxX(i)),pwrs(i));
  end
  k = 1 + numIndpX;
  ycalc = ycalc + res(2,end) * x123;
end
```

Program clem4b is a version of program clem3b that is adapted to handle the triple cross-product term. The console output for running program clem4b is:

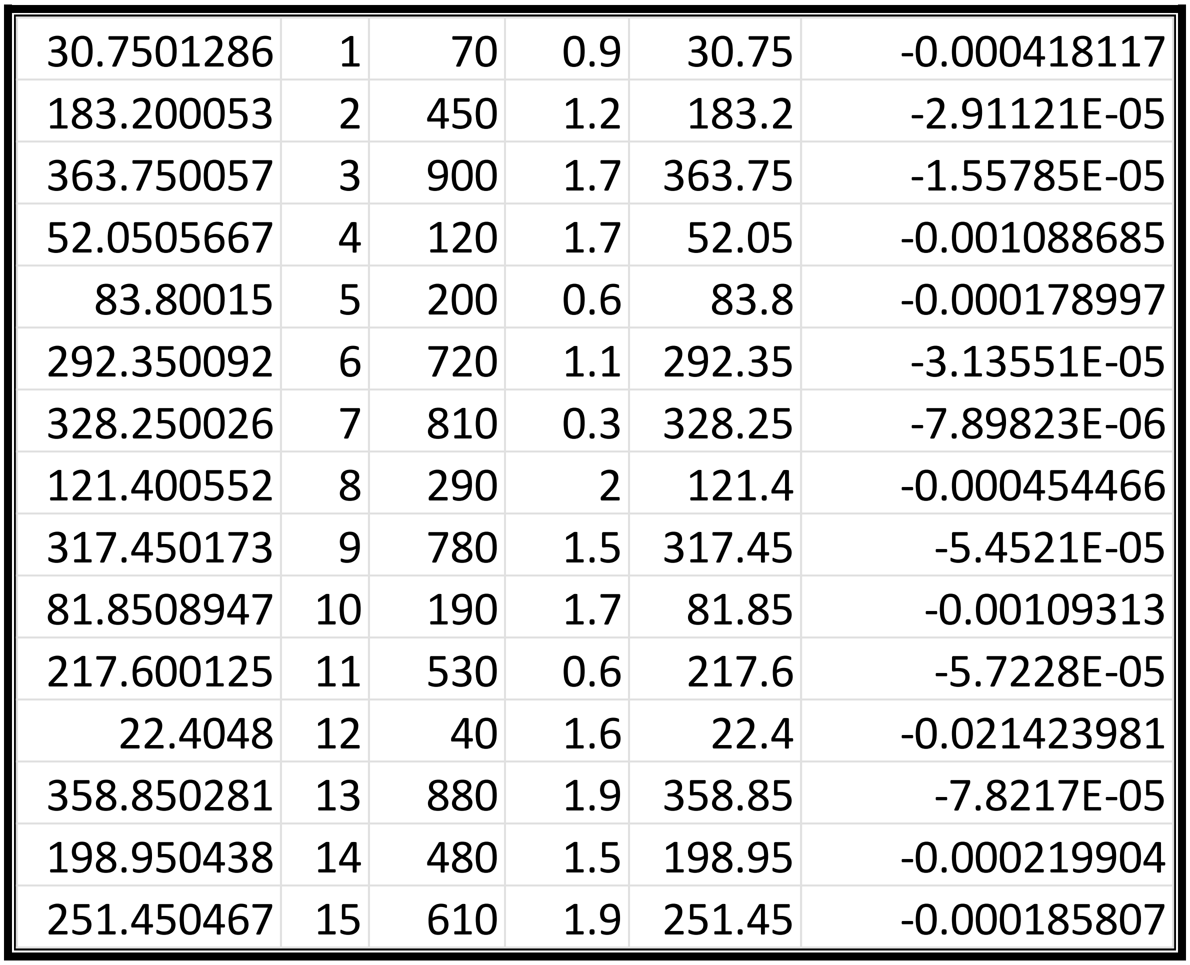
```
Program  clem4b

2025-Jan-2 16_01_13
Please wait ...

Excel file used is C:\Users\nsham\Desktop\Clement4\data4b.xlsx
Diary file used is data4b_2025-Jan-2 16_01_13.txt
------------- Calculating Max Iters ----------------
Max iters = 864
------------- Maximizing Fx ----------------
Iter 41, Fx = 1.000000e+00, X=[1, 1, 1, 3, 2, 1, 1, 1, ]
Progress  4.98%
Progress  9.95%
Progress 14.93%
Progress 19.91%
Progress 24.88%
Iter 257, Fx = 1.000000e+00, X=[1, 1, 1, 3, 2, 1, 2, 1, ]
Progress 29.86%
Progress 34.84%
Progress 39.81%
Progress 44.79%
Progress 49.77%
Progress 54.75%
Progress 59.72%
Progress 64.70%
Progress 69.68%
Progress 74.65%
Progress 79.63%
Progress 84.61%
Progress 89.58%
Progress 94.56%
Progress 99.54%
Regression model
Adjusted R-square =
AdjR2 =

     1


mdl =
Linear regression model:
    y ~ 1 + x1 + x2 + x3 + x4

Estimated Coefficients:
                       Estimate                 SE                  tStat          pValue
                   ________________    ___________________    _______________    ______

    (Intercept)      1.99999999999843    2.81895802662465e-05    70948.2007574687      0
    x1             0.300000000000015      3.742249229393e-07    801656.922376265      0
    x2             0.400000000000001    2.87675921868015e-08    13904535.2632439      0
    x3             0.499999999999857    3.72566105922319e-06    134204.371265085      0
    x4            0.0100000000001017    9.80756494366592e-06    1019.62108408572      0


Number of observations: 250, Error degrees of freedom: 245
Root Mean Squared Error: 0.000125
R-squared: 1,  Adjusted R-Squared: 1
F-statistic vs. constant model: 1.87e+32, p-value = 0


Best model is:
Y = (2)
   + (0.3)*X1
   + (0.4)*X2
   + (0.5)*X3
   + (0.01)*X3/X2*X1
Done!
```

Figures 10.3 and 10.4 show sheets **Results** and **Project** (partial view). The small percentage errors in Figure 10.4 confirm that the model obtained is the correct one.



AdjR-sqr	X1	X2	X3	px1	px2	px3	i1	i2	i3	Op1	Op2
1	1	2	3	1	1	1	3	2	1	2	1
Coeffs	2.00	0.30	0.40	0.50	0.01						
Pwrs	1	1	1								

*Figure 10.3. The sheet **Results** in file data4b.xlsx.*



30.7501286	1	70	0.9	30.75	-0.000418117
183.200053	2	450	1.2	183.2	-2.91121E-05
363.750057	3	900	1.7	363.75	-1.55785E-05
52.0505667	4	120	1.7	52.05	-0.001088685
83.80015	5	200	0.6	83.8	-0.000178997
292.350092	6	720	1.1	292.35	-3.13551E-05
328.250026	7	810	0.3	328.25	-7.89823E-06
121.400552	8	290	2	121.4	-0.000454466
317.450173	9	780	1.5	317.45	-5.4521E-05
81.8508947	10	190	1.7	81.85	-0.00109313
217.600125	11	530	0.6	217.6	-5.7228E-05
22.4048	12	40	1.6	22.4	-0.021423981
358.850281	13	880	1.9	358.85	-7.8217E-05
198.950438	14	480	1.5	198.95	-0.000219904
251.450467	15	610	1.9	251.45	-0.000185807

Figure 10.4. A partial view of sheet ***Project*** *in file data4b.xlsx.*

11/Conclusion

This study shows that using optimization works well but not perfectly. The case of using a model with three cross-product terms shows that optimization fell short in finding the best correct-product model but yielded a close approximation to it. The case of using a single triple cross-product produced good results for both computational approaches. The brute-force approach using virtual nested loops works well at the cost of computation time. More complex models favor the virtual nested loops method.