Flexible 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 very 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. The power of 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:

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

Notice that each variable in equation (3) **can have** a power other than one!

This study looks at a special type of heteronomials that can have combinations of single-variable terms and cross-product terms. The two extreme cases is having all single-variable terms or having all cross-product terms. The general model for such a heteronomial is:

$$Y = a + \sum_{i,j,k=1}^{n+1} b_k\, X_i^{p(i)}\ \ X_j^{p(j)} \qquad (4)$$

Equation (4) needs an explanation. It represents a flexible model where each term has a single variable or is a cross-product of two different variables. Equation (4) uses the special X_{n+1} as a ghost/virtual unity column vector. The cross product of any variable with X_{n+1} practically yields a single-variable term. Another aspect worth pointing out is that if $p(i)^* \ p(j) < 0$ then a term represents a ratio of two variables. Of course, when both p(i) and p(j) are negative you have a term with a reciprocal cross-product.

☛ I highly recommend that you perform the calculations for flexible heteronomials on a **dedicated fast computer**. Computing time increases significantly with the number of data points, candidate independent variables, number of selected variables, and power ranges.

2/Case of One Cross-Product Term

Now that I presented equation (4), let's look at the first case in this study that selects the best flexible cross-product heteronomial model, based on:

- The calculations involve N *candidate* independent variables (where N > 0). Some of these variables may not be selected as part of the best heteronomial model. The program injects a ghost/virtual variable at index N+1 represents the unity column vector.
- The study picks one or two independent variables (i and j from the set of N+1 variables) that best fit either one of the two models:

$$Y = a + b*X_i^{pi} * X_j^{pj} \qquad (5a)$$
$$Y = a + b*X_i^{pi} \qquad (5b)$$

- Where a, b, pi, and pj are to be determined in the calculations and yield the highest adjusted R-square statistic. The calculations obtain values for X_i^{pi} by conceptually multiplying X_i^{pi} by the unity column vector X_{n+1}^{pn1}.

The optimization for equations (5a) and (5b) involves virtual nested loops that select the combinations of the various variables and the various powers. The powers should range from negative to positives values to test products and ratios. In using equation (5a) the indices i and j for selecting the variables must be different.

3/Case of Two Cross-Product Terms

The second case in this study focuses on selecting the best rational heteronomial model, based on:

- The calculations involve N *candidate* independent variables (where N > 1). Some of these variables may not be selected as part of the best heteronomial model. The program injects a ghost/virtual variable at index N+1 represents the unity column vector.
- The study picks the four independent variables (i, j, and k from the set of N variables) that best fit one of the following two models:

$$Y = a + b*X_i^{pi} * X_j^{pj} + c*X_k^{pk} * X_m^{pm} \quad (6a)$$
$$Y = a + b*X_i^{pi} + c*X_k^{pk} * X_m^{pm} \quad (6b)$$
$$Y = a + b*X_i^{pi} + c*X_k^{pk} \quad (6c)$$

- Where the regression coefficients and powers are to be determined in the calculations and yield the highest adjusted R-square statistic to choose either model 6a, 6b, or 6c. The choices are between a model with two cross-product terms, one cross-product term and one single-variable term, and two single-variable terms.

The optimization for equations (6a) to (6c) involves a **single** set of (virtual) nested loops that select the combinations of the various variables, and the various powers.

4/Case of Three Cross-Product Terms

The third case in this study focuses on selecting the best rational heteronomial model, based on:

- The calculations involve N *candidate* independent variables (where N > 2).
- The study picks three to six independent variables that best fit a model. Here is what the two extreme models look like (one made up of purely single-variable terms and the other made up of purely cross-product terms. The program injects a ghost/virtual variable at index N+1 represents the unity column vector. This ghost variable helps to obtain single-variable terms. Other models have terms that are combinations of single-variable terms and cross-product terms):

$$Y = a + b_1*X_i^{p1} + b_2*X_j^{p2} + b_3*X_k^{p3} \quad (7a)$$
$$\begin{aligned} Y = a + {} & b_1*X_{i1}^{p11} * X_{i2}^{p12} + \\ & b_2*X_{j1}^{p21} *X_{j2}^{p22} + \\ & b_3*X_{k1}^{p31} *X_{k2}^{p32} \end{aligned} \quad (7b)$$

- Where the regression coefficients and powers are to be determined in the calculations and yield the highest adjusted R-square statistic in order to choose the best in the above models.

The optimization for equations (7a), (7b) and all in-between combinations involves a **single** set of (virtual) nested loops that select the combinations of the various variables and the various powers.

5/Case of Four Cross-Product Terms

The case of selecting the best five independent variables is an extension of the case of selecting the best four to eight independent variables.

The optimization for equations used involves a single set of nested loops that select the combinations of the various variables and the various powers

6/What are Heteronomials Good For?

Aside from commonly used heteronomials in equations (1) and (2), what are other heteronomials good for? They help us search for **good new empirical relationships** between variables—ones that theoretical analysis would not easily deduce or derive.

Readers may ask about using optimization to zoom in on the best powers of heteronomials. While optimization is a good alternative, it does not easily lend itself to working with the ln(x) transformations. In addition, the best powers calculated using optimization would lack simpler power patterns that I use in this entire study. And finally, using optimization may not give you the exact same results every time—it is not deterministic, while the approach in this study is!

7/The Excel Files for the Case of One Cross-Product Term

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 7.1):

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 7.2):

1. The first row contains groups of headers **that you must enter**. The first group of headers is ix1 and ix2. The second group of headers is px1 and px2.
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 lower limits of the indices for the candidate independent variables, and also the power values.
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 higher limits for the indices of the candidate independent variables, as well as the power values.

Please note that the values for rows 2 and 3 define the ranges for candidate variables and the ranges of powers used.

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. I use the headers X_n to indicate the candidate independent variables. I use the headers ix_n to indicate the indices to the selected candidate independent variables.

Y	X1	X2	X3	X4
4.1	1	7	51	20
8.75	2	45	87	21
11	3	90	7	20
2.9	4	12	90	22
3.2	5	20	91	20
5.6	6	72	45	21
5.47142857	7	81	3	21
3.0875	8	29	60	22
4.6	9	78	19	21
2.57	10	19	38	22
3.44545455	11	53	86	21
2.1	12	4	27	20
4.03076923	13	88	72	20
3.02857143	14	48	84	20
3.22	15	61	93	20
3.10625	16	59	71	21

*Figure 7.1. Sample sheet **Data** for 4 independent variables in file data1.xlsx.*

Figure 7.2 shows a sample sheet **Transf**. Based on the data in that sheet, the set of models examined have four candidate variables with X_1, X_2, X_3, and X_4. So, the values under columns ix1 and ix2 range from 1 to 4. The powers range from –2 to 2 and appear as such under columns px1 and px2. The negative powers allow for terms that have ratios of variables.

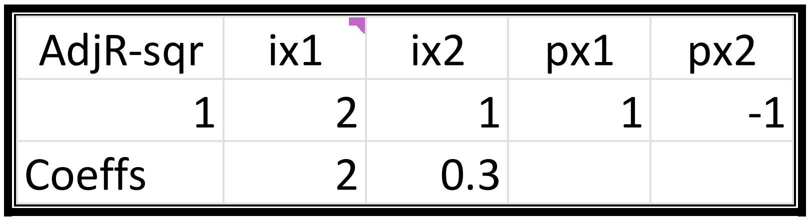
	ix1	ix2	px1	px2
Lb	1	1	-2	-2
Ub	4	4	2	2

*Figure 7.2. Sample sheet **Transf** for 2 independent variables with one cross-product term in file data1.xlsx.*

The Sheet **Results**

The sheet **Results** is one of the output sheets. Figure (7.3) 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 R-square values.
    b. Cells B2 and C2 contain the indices for the selected variables.
    c. Cells D2 and E2 contain the values for the best powers used with the selected variables.
    d. Cell B3 contains the intercept.
    e. Cell C3 contains the slope of the cross-product term.

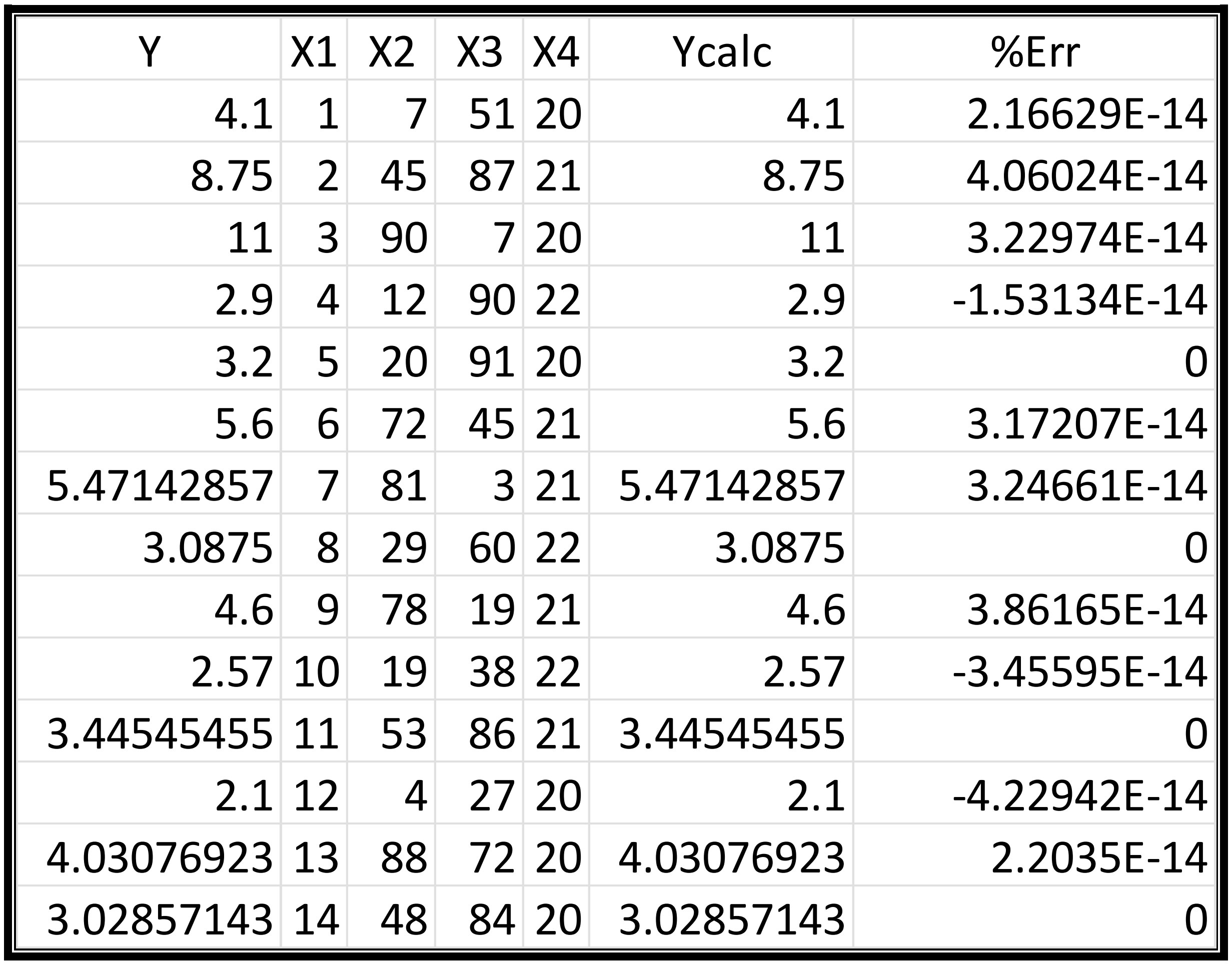


AdjR-sqr	ix1	ix2	px1	px2
1	2	1	1	-1
Coeffs	2	0.3		

*Figure 7.3. Sample sheet **Results** for 2 selected independent variables 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 (7.4) shows a partial sample view of sheet **Project**.



Y	X1	X2	X3	X4	Ycalc	%Err
4.1	1	7	51	20	4.1	2.16629E-14
8.75	2	45	87	21	8.75	4.06024E-14
11	3	90	7	20	11	3.22974E-14
2.9	4	12	90	22	2.9	-1.53134E-14
3.2	5	20	91	20	3.2	0
5.6	6	72	45	21	5.6	3.17207E-14
5.47142857	7	81	3	21	5.47142857	3.24661E-14
3.0875	8	29	60	22	3.0875	0
4.6	9	78	19	21	4.6	3.86165E-14
2.57	10	19	38	22	2.57	-3.45595E-14
3.44545455	11	53	86	21	3.44545455	0
2.1	12	4	27	20	2.1	-4.22942E-14
4.03076923	13	88	72	20	4.03076923	2.2035E-14
3.02857143	14	48	84	20	3.02857143	0

*Figure 7.4. Sample sheet **Project** for 2 selected independent variables with one cross-product term in file data1.xlsx.*

8/Code for the Case of One Cross-Product Term

Here is the MATLAB code for file clemTerm1.m that yie4lds the results shown in the above figures.

```
clc
close all
clear all

global mdl
global xdata
global ydata
global maxVarIdx

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  clemTerm1\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');

ydata = xyData(:,1);
xdata = xyData(:,2:end);
[maxrows,maxcols] = size(xdata);
xyTransf = readmatrix(filename,'Sheet','Transf');
xyTransf = xyTransf(:,2:end);
numX = 2;
Lb = xyTransf(1,:);
Ub = xyTransf(2,:);
% augment indices for variables to include the index of the dummy unity
% variable
Ub(1,1:numX) = Ub(1,1:numX)+1;
maxVarIdx = Ub(1,1:numX);
nVars = length(Lb);
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);

idx = Lb;
iter = 0;
```

```
bestX = zeros(1,length(Lb));
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");
    if r2adj==1, break; end
  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
bestX = [AdjR2 bestX];

res = zeros(2,length(bestX));
res(1,:) = bestX;
res(2,2) = mdl.Coefficients{1,1}';
res(2,3) = mdl.Coefficients{2,1}';
writematrix(res(1,:), filename, 'Sheet', 'Results', 'Range', 'A2:Z2');
writematrix(res(2,2:3), filename, 'Sheet', 'Results', 'Range', 'B3:Z3');

% 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))));

smodel = sprintf("Y = %e + (%e)*", res(2,2), res(2,3));
i1 = res(1,2);
i2 = res(1,3);
px1 = res(1,4);
px2 = res(1,5);

if px1==0
  sX1 = sprintf("log(X%d)", i1);
elseif px1==4
  sX1 = sprintf("sqrt(X%d)", i1);
elseif px1==-4
  sX1 = sprintf("1/log(X%d)", i1);
elseif px1==1
  sX1 = sprintf("X%d", i1);
elseif px1>1
  sX1 = sprintf("X%d^%d", i1, px1);
elseif px1==-1
  sX1 = sprintf("1/X%d", i1);
elseif px1<-1
  sX1 = sprintf("1/X%d^%d", i1, abs(px1));
else
  sX1 = sprintf("X%d", i1);
end

if px2==0
  sX2 = sprintf("log(X%d)", i2);
elseif px2==4
  sX2 = sprintf("sqrt(X%d)", i2);
elseif px2==-4
  sX2 = sprintf("log(X%d)", i2);
elseif px2==1
  sX2 = sprintf("X%d", i2);
elseif px2>1
  sX2 = sprintf("X%d^%d", i2, px2);
elseif px2==-1
  sX2 = sprintf("1/X%d", i2);
elseif px2<-1
  sX2 = sprintf("1/X%d^%d", i2, abs(px2));
else
  sX2 = sprintf("X%d", i2);
end

if i1<maxVarIdx(1) && i2<maxVarIdx(2)
  s = sprintf("%s * %s", sX1, sX2);
  smodel = strcat(smodel, s);
elseif i1<maxVarIdx(1)
  smodel = strcat(smodel, sX1);
elseif i2<maxVarIdx(2)
  smodel = strcat(smodel, sX2);
end

fprintf("%s\n", smodel)

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==4
    x = sqrt(x);
  elseif p==-4
    x = 1./log(x);
  elseif p > 1
    x = x.^p;
  elseif p < 0
    x = 1./x.^abs(p);
  end
end

function r = myFit(coeff)
  global mdl
  global xdata
  global ydata
  global maxVarIdx

  numX = length(maxVarIdx);
  [maxrows, ~] = size(xdata);
  y = ydata;
  i1 = coeff(1);
  i2 = coeff(2);
  px1 = coeff(3);
  px2 = coeff(4);

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

  if i1<maxVarIdx(1)
    x1 = myfx(xdata(:,i1),px1);
  else
    x1 = 1+zeros(maxrows,1); % 1+zeros(maxrows,1);
  end
  if i2<maxVarIdx(2)
    x2 = myfx(xdata(:,i2),px2);
  else
    x2 = 1+zeros(maxrows,1); % 1+zeros(maxrows,1);
  end
  if i1<maxVarIdx(1) && i2<maxVarIdx(2)
    X = x1.*x2;
  elseif i2<maxVarIdx(2)
    X = x2;
  elseif i1<maxVarIdx(1)
    X = x1;
  end
  mdl = fitlm(X,y);
  r = mdl.Rsquared.Adjusted;
end

function ycalc = project(res)
  global xdata
  global maxVarIdx

  [maxrows, maxcols] = size(xdata);

  i1 = res(1,2);
  i2 = res(1,3);
  px1 = res(1,4);
  px2 = res(1,5);

  if i1<maxVarIdx(1)
    x1 = myfx(xdata(:,i1),px1);
  else
    x1 = 1 + zeros(maxrows,1);
  end

  if i2<maxVarIdx(2)
    x2 = myfx(xdata(:,i2),px2);
  else
    x2 = 1 + zeros(maxrows,1);
  end

  ycalc = res(2,2) + res(2,3) * x1.*x2;
end
```

The above code performs the following general tasks:

- Initializes the names of the Excel file and the diary file.
- Opens the diary file to echo console output.
- Reads the input data from sheet **Data**. The program then copies the values for the dependent variable into column vector ydata. It also copies the values for the independent variables into matrix xdata.
- Initializes the row vectors Lb and Ub to contain the ranges for the indices of the independent variables and their powers.
- Initializes the arrays iFrom, iTo, iStep, and idx that are used to control the virtual nested loops. The text for this task appears in red color.
- Uses the first virtual nested loops (coded in red), the program calculates the maximum number of iterations that will be needed to optimize the selected rational heteronomial. The program displays the result.
- Re-initializes the array idx (coded in red) and implement the second virtual loop to MAXIMIZE the values of the adjusted R-square static and determine the best rational heteronomial. The loop performs the following tasks.

  - Periodically display the current best results.
  - Invokes the function myFit() to calculate the adjusted R-square static for the current model. This function returns 0 when the the variables have the same selection index.
  - Test if the calculated adjusted R-square static is greater than the value in variable bestFx. If it is, then store and display the updated best adjusted R-square, the indices of the two variables, and the powers used with these two variables.
- Displays the results and write results in the sheets **Result**.
- Calculates the array of projected values for the dependent variable (by calling function project()) and their percentage errors. Write these results to sheet **Project**.
- Builds and displays a string that shows the best fitted model.

The MATLAB code has the function myFx() that transforms the values of a variable using the specified power p. The function has a set of if statements that map the values of p to different transformations. For example, when p is zero, the function returns the natural logarithm values. Also, when p is 4, the function returns the square root values. This is of course just a choice. If you eliminate the elseif clause that tests p for equality with 4, then the function returns values raised to the power 4.

Here is the output generated by program clemTerm1.

```
Program  clemTerm1

2025-Jan-23 15_48_35
Please wait ...

Excel file used is C:\Users\nsham\Dropbox\MATLAB\Clement5\data1.xlsx
Diary file used is data1_2025-Jan-23 15_48_35.txt
------------- Calculating Max Iters ----------------
Max iters = 625
------------- Maximizing Fx ----------------
Iter 2, Fx = 4.683650e-03, X=[2, 1, -2, -2, ]
Iter 3, Fx = 3.529944e-01, X=[3, 1, -2, -2, ]
Iter 28, Fx = 4.021262e-01, X=[3, 1, -1, -2, ]
Progress  4.96%
Progress  9.92%
Iter 66, Fx = 4.062721e-01, X=[1, 4, 0, -2, ]
Iter 71, Fx = 4.754388e-01, X=[1, 5, 0, -2, ]
Iter 77, Fx = 7.829923e-01, X=[2, 1, 1, -2, ]
Progress 14.88%
Iter 102, Fx = 8.711593e-01, X=[2, 1, 2, -2, ]
Progress 19.84%
Progress 24.80%
Progress 29.76%
Iter 202, Fx = 1.000000e+00, X=[2, 1, 1, -1, ]
Regression model
Adjusted R-square = 1.000000000000

mdl =


Linear regression model:
    y ~ 1 + x1

Estimated Coefficients:
                   Estimate    SE    tStat    pValue
                   ________    __    _____    ______

    (Intercept)        2       0      Inf       0
    x1               0.3       0      Inf       0


Number of observations: 100, Error degrees of freedom: 98
R-squared: 1,  Adjusted R-Squared: 1
F-statistic vs. constant model: 1.69e+32, p-value = 0
Y = 2.000000e+00 + (3.000000e-01)*X2 * 1/X1
Done!
```

The output shows that the program can identify the correct model used to calculate the values of the dependent variable.

9/ The Excel Files for the Case of Two Cross-Product Terms

In this section, I focus on the Excel sheets **Transf** and **Results** for two terms and how they all differ from the same sheets for the case of one term.

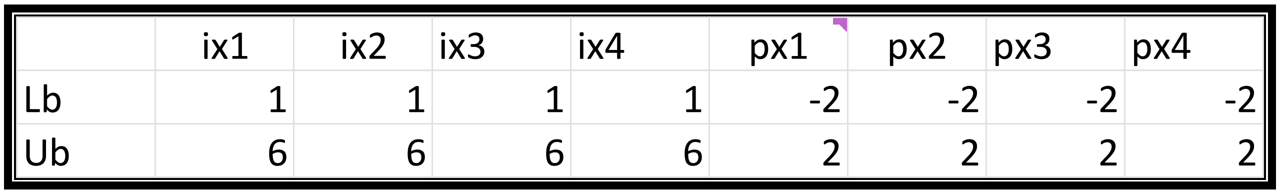
Sheet **Transf**

The sheet **Transf** contains the indices for selecting variables and their transformations and the following columns and rows (see Figure 9.1):

1. The first row contains groups of headers **that you must enter**. The first group of headers are ix1, ix2, ix3, and ix4. The second group of headers is px1, px2, px3, and px4.
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 lower limits of the indices for the candidate independent variables and the power values.

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 higher limits of the indices for the candidate independent variables and the power values.

Figure 9.1 shows a sample sheet **Transf**. Based on the data in that sheet, the set of models examined have six candidate variables with four selected variables. So, the values under columns ix1 to ix4 range from 1 to 6. The powers range from –2 to 2.



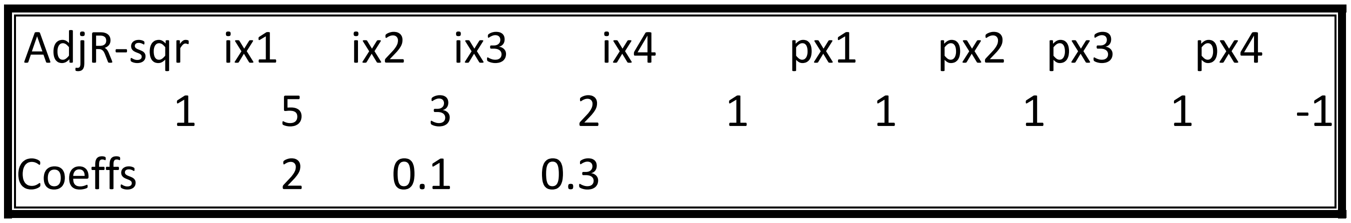
	ix1	ix2	ix3	ix4	px1	px2	px3	px4
Lb	1	1	1	1	-2	-2	-2	-2
Ub	6	6	6	6	2	2	2	2

*Figure 9.1. 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 (7.3) 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 R-square values.
   b. Cells B2 to D2 contain the indices for the selected variables.
   c. Cells E2 to G2 contain the values for the best powers used with the selected variables.
   d. Cells B3 to D3 contain the intercept and the two slopes.



AdjR-sqr	ix1	ix2	ix3	ix4	px1	px2	px3	px4
1	5	3	2	1	1	1	1	-1
Coeffs	2	0.1	0.3					

*Figure 9.2. Sample sheet **Results** for 4 selected independent variables in file data2.xlsx.*

10/ The MATLAB File for the Case of Two Terms

Here is th source code for the MATLAB file clemTerm2 which obtains the best heteronomial with two terms (either single-variable or cross-product):

```
clc
close all
clear all

global mdl
global xdata
global ydata
global maxVarIdx

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  clemTerm2\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');

ydata = xyData(:,1);
xdata = xyData(:,2:end);
[maxrows,maxcols] = size(xdata);
xyTransf = readmatrix(filename,'Sheet','Transf');
xyTransf = xyTransf(:,2:end);

% Important assignment - must match the number of X columns in sheet
% Transf of the Excel file
numX = 4;
Lb = xyTransf(1,:);
Ub = xyTransf(2,:);
% augment indices for variables to include the index of the dummy unity
% variable
Ub(1,1:numX) = Ub(1,1:numX)+1;
maxVarIdx = Ub(1,1:numX);
nVars = length(Lb);
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;
iter = 0;

bestX = zeros(1,length(Lb));
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");
    if r2adj == 1, break; end
  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
bestX = [AdjR2 bestX];

res = zeros(2,length(bestX));
res(1,:) = bestX;
res(2,2) = mdl.Coefficients{1,1}';
res(2,3) = mdl.Coefficients{2,1}';
res(2,4) = mdl.Coefficients{3,1}';
writematrix(res(1,:), filename, 'Sheet', 'Results', 'Range', 'A2:Z2');
writematrix(res(2,2:4), filename, 'Sheet', 'Results', 'Range', 'B3:Z3');

% 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))));

smodel = sprintf("Y = %e + (%e)*", res(2,2), res(2,3));
idx = res(1,2:numX+1);
px = res(1,numX+2:2*numX+1);

if px(1)==0
  sX1 = sprintf("log(X%d)", idx(1));
elseif px(1)==4
  sX1 = sprintf("sqrt(X%d)", idx(1));
elseif px(1)==-4
  sX1 = sprintf("1/log(X%d)", idx(1));
elseif px(1)==1
  sX1 = sprintf("X%d", idx(1));
elseif px(1)>1
  sX1 = sprintf("X%d^%d", idx(1), px(1));
elseif px(1)==-1
  sX1 = sprintf("1/X%d", idx(1));
elseif px(1)<-1
  sX1 = sprintf("1/X%d^%d", idx(1), abs(px(1)));
else
  sX1 = sprintf("X%d", idx(1));
end

if px(2)==0
  sX2 = sprintf("log(X%d)", idx(2));
elseif px(2)==4
  sX2 = sprintf("sqrt(X%d)", idx(2));
```

```
elseif px(2)==-4
  sX2 = sprintf("log(X%d)", idx(2));
elseif px(2)==1
  sX2 = sprintf("X%d", idx(2));
elseif px(2)>1
  sX2 = sprintf("X%d^%d", idx(2), px(2));
elseif px(2)==-1
  sX2 = sprintf("1/X%d", idx(2));
elseif px(2)<-1
  sX2 = sprintf("1/X%d^%d", idx(2), abs(px(2)));
else
  sX2 = sprintf("X%d", idx(2));
end

if px(3)==0
  sX3 = sprintf("log(X%d)", idx(3));
elseif px(3)==4
  sX3 = sprintf("sqrt(X%d)", idx(3));
elseif px(3)==-4
  sX3 = sprintf("log(X%d)", idx(3));
elseif px(3)==1
  sX3 = sprintf("X%d", idx(3));
elseif px(3)>1
  sX3 = sprintf("X%d^%d", idx(2), px(3));
elseif px(3)==-1
  sX3 = sprintf("1/X%d", idx(3));
elseif px(3)<-1
  sX3 = sprintf("1/X%d^%d", idx(2), abs(px(3)));
else
  sX3 = sprintf("X%d", idx(3));
end

if px(4)==0
  sX4 = sprintf("log(X%d)", idx(4));
elseif px(4)==4
  sX4 = sprintf("sqrt(X%d)", idx(4));
elseif px(4)==-4
  sX4 = sprintf("log(X%d)", idx(4));
elseif px(4)==1
  sX4 = sprintf("X%d", idx(4));
elseif px(4)>1
  sX4 = sprintf("X%d^%d", idx(4), px(4));
elseif px(4)==-1
  sX4 = sprintf("1/X%d", idx(4));
elseif px(4)<-1
  sX4 = sprintf("1/X%d^%d", idx(4), abs(px(4)));
else
  sX4 = sprintf("X%d", idx(4));
end
if idx(1)<maxVarIdx(1) && idx(2)<maxVarIdx(2)
  s = sprintf("%s * %s", sX1, sX2);
  smodel = strcat(smodel, s);
elseif idx(1)<maxVarIdx(1)
  smodel = strcat(smodel, sX1);
elseif idx(2)<maxVarIdx(2)
  smodel = strcat(smodel, sX2);
end
```

```
smodel = sprintf("%s + (%e)*", smodel, res(2,4));

if idx(3)<maxVarIdx(3) && idx(4)<maxVarIdx(4)
  s = sprintf("%s * %s", sX3, sX4);
  smodel = strcat(smodel, s);
elseif idx(3)<maxVarIdx(3)
  smodel = strcat(smodel, sX3);
elseif idx(4)<maxVarIdx(4)
  smodel = strcat(smodel, sX4);
end
fprintf("%s\n", smodel)

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==4
    x = sqrt(x);
  elseif p==-4
    x = 1./log(x);
  elseif p > 1
    x = x.^p;
  elseif p < 0
    x = 1./x.^abs(p);
  end
end

function r = myFit(coeff)
  global mdl
  global xdata
  global ydata
  global maxVarIdx

  numX = length(maxVarIdx);
  [maxrows, maxcols] = size(xdata);
  y = ydata;
  idx = coeff(1:numX);
  px= coeff(numX+1:2*numX);
  x = zeros(maxrows,numX);
  x2 = zeros(maxrows,fix(numX/2));

  % check for redundant vars in a term
  if idx(1)==idx(2) || idx(3)==idx(4)
    r = 0;
    return;
  end

  if idx(1)==idx(3) && px(1)==px(3) && ...
```

```
     idx(2)==idx(4) && px(2)==px(4)
    r = 0;
    return;
  end

  if idx(1)==idx(4) && px(1)==px(4) && ...
     idx(2)==idx(2) && px(2)==px(3)
    r = 0;
    return;
  end

  for i=1:numX
    if idx(i)<maxVarIdx(i)
      x(:,i) = myfx(xdata(:,idx(i)),px(i));
    else
      x(:,i) = 1+zeros(maxrows,1);
    end
  end

  i1=-1;
  i2=0;
  for i=1:fix(numX/2)
    i1=i1+2;
    i2=i2+2;
    if idx(i1)<maxVarIdx(i1) && idx(i2)<maxVarIdx(i2)
      x2(:,i) = x(:,i1).*x(:,i2);
    elseif idx(i2)<maxVarIdx(i2)
      x2(:,i) = x(:,i2);
    elseif idx(i1)<maxVarIdx(i1)
      x2(:,i) = x(:,i1);
    end
  end
  X = [x2(:,1) x2(:,2)];
  mdl = fitlm(X,y);
  r = mdl.Rsquared.Adjusted;
end

function ycalc = project(res)
  global xdata
  global maxVarIdx

  numX = length(maxVarIdx);
  [maxrows, maxcols] = size(xdata);

  idx = res(1,2:numX+1);
  px = res(1,numX+2:2*numX+1);
  x = zeros(maxrows,4);

  for i=1:numX
    if idx(i)<maxVarIdx(i)
      x(:,i) = myfx(xdata(:,idx(i)),px(i));
    else
      x(:,i) = 1 + zeros(maxrows,1);
    end
  end

    ycalc = res(2,2) + res(2,3) * x(:,1).*x(:,2);
    ycalc = ycalc + res(2,4) * x(:,3).*x(:,4);
end
```

The above code is very similar to that of clemTerm1.m. Here is the output for running program clemtTerm2.

```
Program  clemTerm2

2025-Jan-23 16_48_44
Please wait ...

Excel file used is C:\DropBox\Dropbox\MATLAB\Clement5\data2.xlsx
Diary file used is data2_2025-Jan-23 16_48_44.txt
------------- Calculating Max Iters ----------------
Max iters = 1,500,625
------------- Maximizing Fx ----------------
Iter 52, Fx = 2.194497e-02, X=[3, 1, 2, 1, -2, -2, -2, -2, ]
Iter 68, Fx = 3.212773e-02, X=[5, 3, 2, 1, -2, -2, -2, -2, ]
Iter 103, Fx = 3.711370e-02, X=[5, 1, 3, 1, -2, -2, -2, -2, ]
Iter 108, Fx = 4.641412e-02, X=[3, 2, 3, 1, -2, -2, -2, -2, ]
Iter 117, Fx = 6.552382e-02, X=[5, 3, 3, 1, -2, -2, -2, -2, ]
Iter 902, Fx = 7.176095e-02, X=[6, 3, 5, 3, -2, -2, -2, -2, ]
Iter 1260, Fx = 8.740633e-02, X=[7, 5, 5, 4, -2, -2, -2, -2, ]
Iter 2476, Fx = 8.829601e-02, X=[5, 4, 2, 1, -1, -2, -2, -2, ]
Iter 2525, Fx = 1.395030e-01, X=[5, 4, 3, 1, -1, -2, -2, -2, ]
Iter 3258, Fx = 1.582839e-01, X=[3, 4, 4, 3, -1, -2, -2, -2, ]
Iter 3279, Fx = 1.619105e-01, X=[3, 7, 4, 3, -1, -2, -2, -2, ]
Iter 3652, Fx = 2.240745e-01, X=[5, 4, 5, 4, -1, -2, -2, -2, ]
Iter 4875, Fx = 2.653914e-01, X=[3, 4, 2, 1, 0, -2, -2, -2, ]
Iter 4877, Fx = 2.929547e-01, X=[5, 4, 2, 1, 0, -2, -2, -2, ]
Iter 4898, Fx = 2.999586e-01, X=[5, 7, 2, 1, 0, -2, -2, -2, ]
Iter 4926, Fx = 3.752729e-01, X=[5, 4, 3, 1, 0, -2, -2, -2, ]
Iter 4947, Fx = 3.843021e-01, X=[5, 7, 3, 1, 0, -2, -2, -2, ]
Iter 5759, Fx = 3.874345e-01, X=[5, 4, 6, 3, 0, -2, -2, -2, ]
Iter 5780, Fx = 3.936924e-01, X=[5, 7, 6, 3, 0, -2, -2, -2, ]
Iter 7297, Fx = 3.962686e-01, X=[3, 7, 2, 1, 1, -2, -2, -2, ]
Iter 7327, Fx = 4.749511e-01, X=[5, 4, 3, 1, 1, -2, -2, -2, ]
Iter 8160, Fx = 4.932850e-01, X=[5, 4, 6, 3, 1, -2, -2, -2, ]
Iter 28880, Fx = 7.499146e-01, X=[5, 3, 2, 1, 0, 0, -2, -2, ]
Iter 28929, Fx = 7.573607e-01, X=[5, 3, 3, 1, 0, 0, -2, -2, ]
Iter 29272, Fx = 7.635166e-01, X=[5, 3, 3, 2, 0, 0, -2, -2, ]
Iter 29664, Fx = 7.855352e-01, X=[5, 3, 4, 3, 0, 0, -2, -2, ]
Iter 29713, Fx = 7.915769e-01, X=[5, 3, 5, 3, 0, 0, -2, -2, ]
Iter 31281, Fx = 8.006457e-01, X=[5, 3, 2, 1, 1, 0, -2, -2, ]
Iter 31293, Fx = 8.197783e-01, X=[3, 5, 2, 1, 1, 0, -2, -2, ]
Iter 31342, Fx = 8.215232e-01, X=[3, 5, 3, 1, 1, 0, -2, -2, ]
Iter 31440, Fx = 8.245007e-01, X=[3, 5, 5, 1, 1, 0, -2, -2, ]
Iter 43286, Fx = 9.999734e-01, X=[5, 3, 2, 1, 1, 1, -2, -2, ]
Iter 43335, Fx = 9.999829e-01, X=[5, 3, 3, 1, 1, 1, -2, -2, ]
Progress  5.00%
Iter 103360, Fx = 9.999840e-01, X=[5, 3, 3, 1, 1, 1, -1, -2, ]
Iter 103507, Fx = 9.999848e-01, X=[5, 3, 6, 1, 1, 1, -1, -2, ]
Progress 10.00%
Iter 165345, Fx = 9.999860e-01, X=[5, 3, 1, 7, 1, 1, 0, -2, ]
Iter 223361, Fx = 9.999942e-01, X=[5, 3, 2, 1, 1, 1, 1, -2, ]
Progress 15.00%
Iter 283386, Fx = 9.999966e-01, X=[5, 3, 2, 1, 1, 1, 2, -2, ]
```

```
Progress 20.00%
Progress 25.00%
Progress 30.00%
Iter 523486, Fx = 1.000000e+00, X=[5, 3, 2, 1, 1, 1, 1, -1, ]
Regression model
Adjusted R-square = 1.000000000000

mdl =


Linear regression model:
    y ~ 1 + x1 + x2

Estimated Coefficients:
                         Estimate                  SE                  tStat          pValue
                     ________________    ____________________    ________________    ______

    (Intercept)      1.99999999999999    1.23212029515539e-06    1623218.12883356      0
    x1                            0.1    3.22858766610029e-10    309732955.527229      0
    x2              0.299999999999998    1.88219959279805e-07    1593879.84753531      0


Number of observations: 100, Error degrees of freedom: 97
Root Mean Squared Error: 7.59e-06
R-squared: 1,  Adjusted R-Squared: 1
F-statistic vs. constant model: 8.71e+32, p-value = 0
Y = 2.000000e+00 + (1.000000e-01)*X5 * X3 + (3.000000e-01)*X2 * 1/X1
Done!
```

The output shows that the program can identify the correct rational heteronomial.

The ZIP file for this study also includes MATLAB files, Excel files, and text diary files for the cases of three and four terms (to model up to 6 to 8 variables, respectively).