Flexible Rational 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 appears in a linear form. 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 power -1 and variable X_2 is raised to power 2. More advanced heteronomials involve more independent variables and different powers like –3, –2.3, –2, 3, 0.5, 0.63, 1.37, 1.5, and so on—the set of powers is virtually infinite. Power zero is *strategically* translated to the ln(x) function (why waste raising to power zero which always yields 1?) while the other powers are typically 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 power other than one!

Padé polynomials take the ratio of two polynomials. The general form of a Padé polynomial is:

$$Y = (a + b_1{*}X + b_2{*}X^2 + \ldots + b_m{*}X^m) / (1 + c_1{*}X + c_2{*}X^2 + \ldots + c_n{*}X^n) \qquad (4)$$

Notice that the intercept of the denominator polynomial is always 1. The Padé polynomials have orders of m and n which may or may not be equal. Moreover, n may be greater than m, or vice versa. The linearized regression form of the Padé polynomials is:

$$Y = a + b_1{*}X + b_2{*}X^2 + \ldots + b_m{*}X^m – c1{*}Y{*}X – c_2{*}Y{*}X^2 – \ldots – c_n{*}Y{*}X^n \qquad (4b)$$

Notice that the terms that were in the denominator of equation (4) have been multiplied by –Y and added to the terms in the numerator.

Now let me introduce you to ***rational heteronomials***. They are to heteronomials what Padé polynomials are to regular polynomials. The general form of rational heteronomials (with no cross-product terms) is:

$$Y^{py} = (a + b_1 * X_1^{px11} + b_2 * X_2^{px12} + \ldots + b_n * X_n^{px1n}) / (1 + c_1 * X_4^{px21} + c_2 * X_5^{px22} + \ldots + c_m * X_n^{px2m}) \quad (5)$$

Each variable in equation (5) has its own power. The minimal condition for rational heteronomials is that **at least one independent variable must appear in both numerator and denominator**. Also, the numerator must have terms with unique variables. The same rule is held for the denominator. Any variable can appear in both the numerator and denominator, but only in one term.

The linearized multiple regression model based on equation (5) is as follows.

$$Y^{py} = a + b_1 * X_1^{px11} + b_2 * X_2^{px12} + \ldots + b_n * X_n^{px1n} - - c_1 * Y^{py} * X_4^{px21} - c_2 * Y^{py} * X_5^{px22} - \ldots - c_n * Y^{py} * X_n^{px2n}) \quad (5b)$$

Notice that the terms in the denominator of equation (5) are now multiplied by minus Y^{py}.

In this study I will use linear values for the dependent variable Y. This scheme will reduce attempts to take roots of negative numbers, which takes us into complex math territory.

2/Case of Two Independent Variables

Now that I have introduced you to rational heteronomials, you may ask about how this study will work with them. Notice that the title of the study includes the word *flexible*, and for a very good reason. The first case in this study focuses on selecting the best rational heteronomial model, based on:

- The calculations involve N *candidate* independent variables (where N > 1). I use the term candidate because some variables may not end up in the best model.
- The study picks the two independent variables (i and j from the set of N variables) that best fit the model:

$$Y = (a + b * X_i^{pi})/(1 + c * X_j^{pj}) \quad (6)$$

- Where a, b, c, pi, and pj are to be determined in the calculations and yield the highest adjusted R-square statistic.

Equation (6) shows that the regression model has one variable in the numerator and one in the denominator. The calculations seek to determine which variables are selected and what powers they are raised to. The indices i and j can be the equal.

The optimization for equation (6) involves nested loops that select the combinations of the various variables and the various powers.

3/Case of Three Independent Variables

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 > 2). Note that some of these candidate variables may not appear in the best rational heteronomial model.
- The study picks the three independent variables (i, j, and k from the set of N variables) that best fit one of the following two models:

$$Y = (a + b_1*X_i^{p1} + b_2*X_j^{p2})/(1 + c_1*X_k^{p3}) \qquad (7a)$$
$$Y = (a + b_1*X_i^{p1})/(1+ c_1*X_j^{p2} + c_2*X_k^{p3}) \qquad (7b)$$

- 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 7a or 7b. In the case of the model in 7a, the values for i and j must be different. In the case of the model in 7b, the values for j and k must be different.

Notice that in equations (7a) and (7b) *some* variable Xj will appear in either the numerator or denominator to yield the best model. To implement this feature, the calculations include an array of orientation flags (with values of 1=for the numerator or 2=for the denominator). Since we are working with three variables, the array of orientation flags has three elements. The first element is always 1 to signal that the selected variable X_i is always in the numerator. The second element can vary between 1 and 2 since the calculations can place variable X_j in either the numerator or in the denominator. The third element is always 2, since X_k always appears in the denominator.

The optimization for equations (7a) and (7b) involves a **single** set of (virtual) nested loops that select the combinations of the various variables, the various powers, and the various orientation flags. The code contains several if statements that guide the calculations to work with different models based on the values of the orientation flags.

4/Case of Four Independent Variables

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 > 3).
- The study picks the four independent variables (i, j, k, and m from the set of N variables) that best fit one of the following three models:

$$Y = (a + b_1*X_i^{p1} + b_2*X_j^{p2} + b_3*X_k^{p3})/(1 + c_1*X_m^{p4}) \tag{8a}$$
$$Y = (a + b_1*X_i^{p1} + b_2*X_j^{p2})/(1+ c_1*X_k^{p3} + c_2*X_m^{p4}) \tag{8b}$$
$$Y = (a + b_1*X_i^{p1})/(1+ c_1*X_j^{p2}+ c_2*X_k^{p3} + c_3*X_m^{p4}) \tag{8c}$$

- Where the regression coefficients and powers are to be determined in the calculations and yield the highest adjusted R-square statistic to choose the best in the above models. In the case of the model in 8a, the values for i, j, and k must be different. In the case of the model in 8b, the values for i and j AND k and m must be different. In the case of the model in 8c, the values for j, k, and m must be different.

Notice that in equations (8a), (8b), and (8c) require that there is always at least one independent variable in the numerator and one in the denominator. The other two variables can appear either in the numerator or in the denominator. Again, the orientation flags help select which model to process.

The optimization for equations (8a), (8b) and (8c) involves a **single** set of (virtual) nested loops that select the combinations of the various variables, the various powers, and the various orientation flags. The code contains several if statements that guide the calculations to work with different models based on the values of the orientation flags.

5/Case of Five Independent Variables

The case of selecting the best five independent variables is an extension of the case of selecting the best four independent variables. As in the last section we require that there is always at least one independent variable in the numerator and one in the denominator. The other three variables can appear either in the numerator or in the denominator. Again, orientation flags are used to test the various models that can be derived from selecting five variables.

The optimization for equations used involves a single set of nested loops that select the combinations of the various variables, the various powers, and the various orientation flags. The code contains several if statements that guide the calculations to work with different models based on the values of the orientation flags.

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 brute-force approach used in this study is!

7/The Excel Files for the Case of Two Independent Variables

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 candidate independent variables. The minimum number of independent variables is 2.

☛ In this study the dependent variable is calculated **without injecting any errors**. The reason for this is to test if the various methods can correctly find the 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 upper limits for the indices for the candidate independent variables, and also for 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.

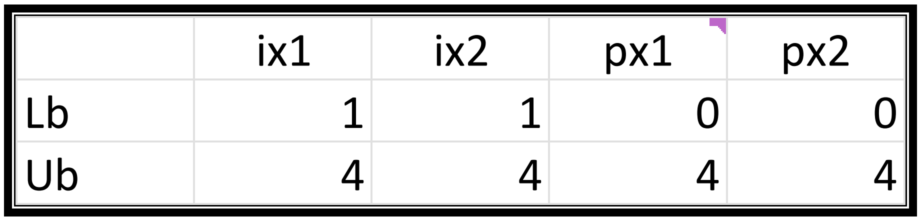
Note that you should enter all the headers in the Excel sheets. The names that you use are your choice and do not affect the calculations. The headers help you easily identify the input data and results. I use the headers X_n to indicate the **candidate** independent variables. I use the headers ix_n to indicate the **indices** to the **selected** candidate independent variables

Y	X1	X2	X3	X4
7.66666667	1	7	51	20
44.1935484	2	45	87	21
90.6666667	3	90	7	20

11.875	4	12	90	22
20.6666667	5	20	91	20
70.3225806	6	72	45	21
79.0322581	7	81	3	21
27.8125	8	29	60	22
76.1290323	9	78	19	21
18.4375	10	19	38	22
51.9354839	11	53	86	21
4.66666667	12	4	27	20
88.6666667	13	88	72	20
48.6666667	14	48	84	20

Figure 7.1. Sample sheet ***Data*** *for 4 independent variables in file data2.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. I could have used ranges 2 to 4 to exclude variable X_1. Likewise, I could have used ranges 1 to 3 to exclude variable X_4. I suggest that you place candidate variables that may be excluded in the rightmst trailing columns of sheet **Data**. The powers range from 0 (for ln(x)) to 4 and appear as such under columns px1 and px2.



	ix1	ix2	px1	px2
Lb	1	1	0	0
Ub	4	4	4	4

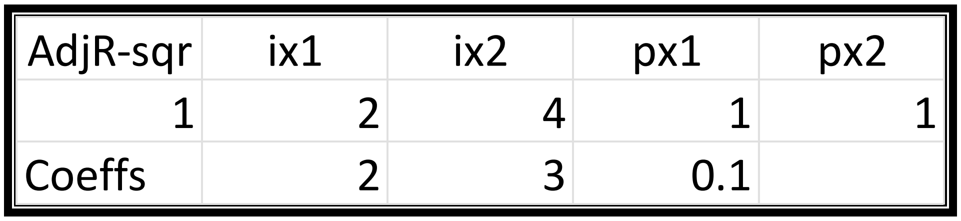
Figure 7.2. Sample sheet ***Transf*** *for 2 selected independent variables with two cross-product terms in file data2.xlsx.*

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 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 values for the intercept.
e. Cells C3 and D3 contain the slopes for the two terms.

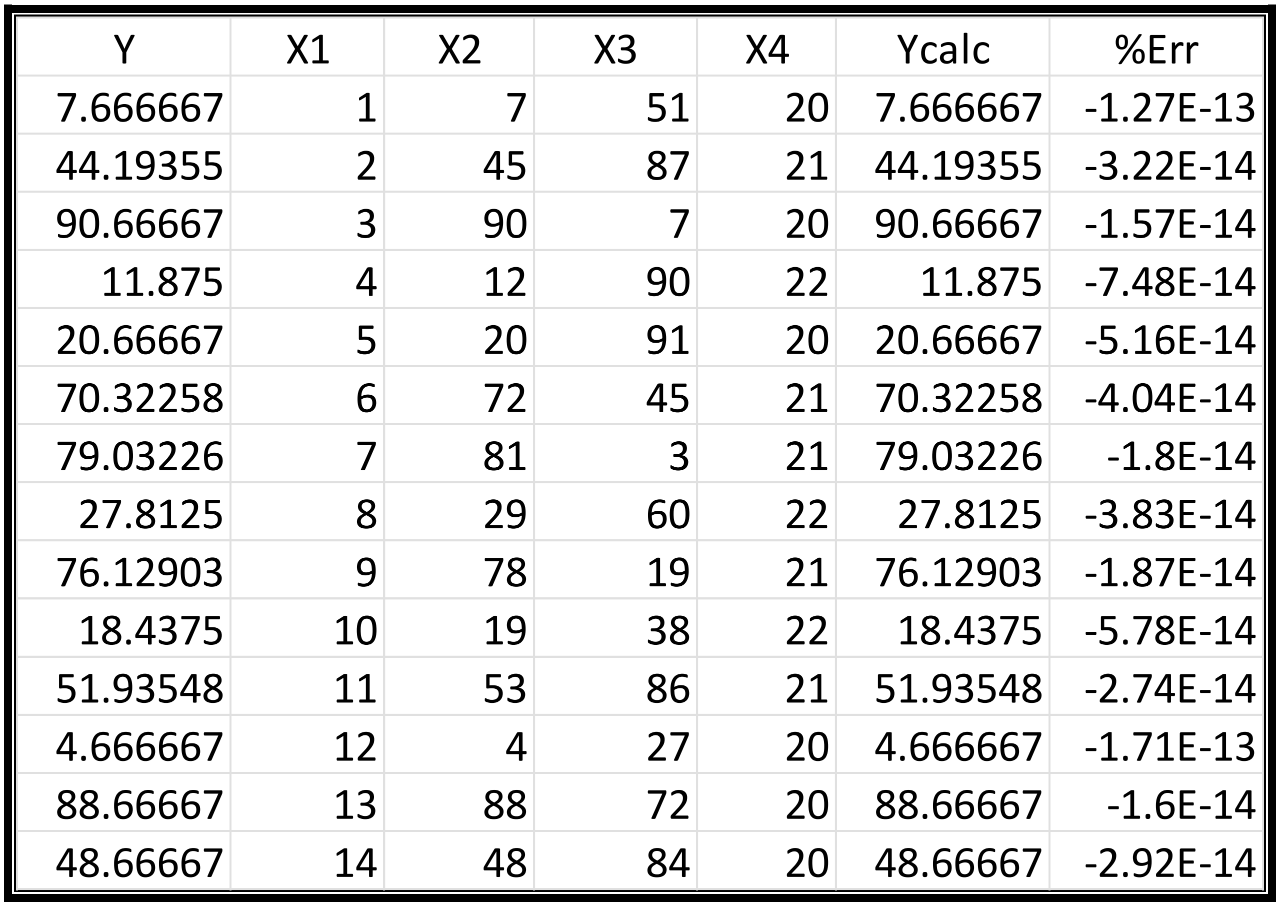


AdjR-sqr	ix1	ix2	px1	px2
1	2	4	1	1
Coeffs	2	3	0.1	

*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
7.666667	1	7	51	20	7.666667	-1.27E-13
44.19355	2	45	87	21	44.19355	-3.22E-14
90.66667	3	90	7	20	90.66667	-1.57E-14
11.875	4	12	90	22	11.875	-7.48E-14
20.66667	5	20	91	20	20.66667	-5.16E-14
70.32258	6	72	45	21	70.32258	-4.04E-14
79.03226	7	81	3	21	79.03226	-1.8E-14
27.8125	8	29	60	22	27.8125	-3.83E-14
76.12903	9	78	19	21	76.12903	-1.87E-14
18.4375	10	19	38	22	18.4375	-5.78E-14
51.93548	11	53	86	21	51.93548	-2.74E-14
4.666667	12	4	27	20	4.666667	-1.71E-13
88.66667	13	88	72	20	88.66667	-1.6E-14
48.66667	14	48	84	20	48.66667	-2.92E-14

Figure 7.4. Sample sheet ***Project*** *for 4 independent variables with two cross-product terms in file data2.xlsx.*

8/Code for the Case of Two Independent Variables

Here is the MATLAB code for file clem2.m that yie4lds the results shown in the above figures.

```
clc
close all
clear all

global mdl
global xdata
global ydata

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');

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

Lb = xyTransf(1,:);
Ub = xyTransf(2,:);
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 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);

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");
  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}';
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))));


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

```
px2 = res(1,5);
smodel = sprintf("Y = (%e + (%e)*", res(2,2), res(2,3));

if px1==0
  sX1 = sprintf("log(X%d)", i1);
elseif px1==4
  sX1 = sprintf("sqrt(X%d)", i1);
elseif px1==1
  sX1 = sprintf("X%d", i1);
elseif px1>1
  sX1 = sprintf("X%d^%d", i1, px1);
elseif px1<0
  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==1
  sX2 = sprintf("X%d", i2);
elseif px2>1
  sX2 = sprintf("X%d^%d", i2, px2);
elseif px2<0
  sX2 = sprintf("1/X%d^%d", i2, abs(px2));
else
  sX2 = sprintf("X%d", i2);
end

s = sprintf("%s)/(1 + (%e)* %s %s", sX1, res(2,4), sX2);
smodel = strcat(smodel, s, ")");
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 > 1
    x = x.^p;
  elseif p < 0
    x = 1./x.^abs(p);
  end
end

function r = myFit(coeff)
```

```
  global mdl
  global xdata
  global ydata


  y = ydata;
  i1 = coeff(1);
  i2 = coeff(2);
  px1 = coeff(3);
  px2 = coeff(4);

  x1 = myfx(xdata(:,i1),px1);
  x2 = -ydata.*myfx(xdata(:,i2),px2);
  X=[x1 x2];
  mdl = fitlm(X,y);
  r = mdl.Rsquared.Adjusted;
end

function ycalc = project(res)
  global xdata

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

  x1 = myfx(xdata(:,i1),px1);
  x2 = myfx(xdata(:,i2),px2);
  ycalc = (res(2,2) + res(2,3) * x1)./(1 + res(2,4)*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 numerator (and denominator) has multiple variables with 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 se 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 (instead of raising values to the power 4).
Here is the output generated by program clem2:

```
Program  clem2

2025-Jan-23 17_05_58
Please wait ...

Excel file used is C:\Users\nsham\Dropbox\MATLAB\Clement4\data2.xlsx
Diary file used is data2_2025-Jan-23 17_05_58.txt
------------- Calculating Max Iters ----------------
Max iters = 400
------------- Maximizing Fx ----------------
Iter 1, Fx = 9.290578e-01, X=[1, 1, 0, 0, ]
Iter 5, Fx = 9.954420e-01, X=[1, 2, 0, 0, ]
Iter 6, Fx = 9.992839e-01, X=[2, 2, 0, 0, ]
Iter 13, Fx = 9.993787e-01, X=[1, 4, 0, 0, ]
Iter 14, Fx = 9.993844e-01, X=[2, 4, 0, 0, ]
Iter 16, Fx = 9.998406e-01, X=[4, 4, 0, 0, ]
Progress  5.00%
Iter 30, Fx = 9.999994e-01, X=[2, 4, 1, 0, ]
Progress 10.00%
Progress 15.00%
Progress 20.00%
Progress 25.00%
Iter 110, Fx = 1.000000e+00, X=[2, 4, 1, 1, ]
Progress 30.00%
Progress 35.00%
Progress 40.00%
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

Estimated Coefficients:
                       Estimate       SE   tStat   pValue</strong>
                   ________________   __   _____   ______</strong>

    (Intercept)   1.99999999999998    0      Inf       0
    x1                           3    0      Inf       0
    x2                         0.1    0      Inf       0


Number of observations: 100, Error degrees of freedom: 97
R-squared: 1,  Adjusted R-Squared: 1
F-statistic vs. constant model: 1.73e+31, p-value = 0
Y = (2.000000e+00 + (3.000000e+00)*X2)/(1 + (1.000000e-01)* X4)
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 Three Independent Variables

In this section, I focus on the Excel sheets **Transf** and **Results** for three (or more) independent variables and how they all differ from the same sheets for the case of two independent variables.

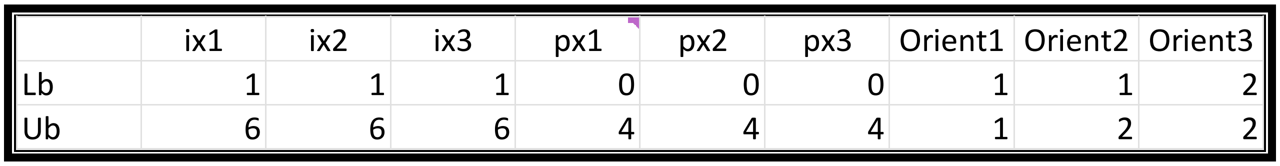
Sheet **Transf**

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 is ix1, ix2, and ix3. The second group of headers is px1, px2, and px3. The third group is the orientation flags Orient1, Orient2, and Orient3.
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, the power values, and the orientation flag values. Notice that the orientation flag for the first variable is AWLAYS 1. The orientation flag for the second variable is 1. The orientation flag for the last variable is ALWAYS 2.
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, the power values, and the orientation flag values. Notice that the orientation flag for the first variable is AWLAYS 1. The orientation flag for the second variable is 2. The orientation flag for the last variable is ALWAYS 2.

☛ Please note that the lower and upper limits for the orientation flag for the first variable are ALWAYS 1. Likewise, lower and upper limits for the orientation flag for the last variable are ALWAYS 2. The lower and upper limits for the orientation flag of any other variable are 1 and 2, respectively.

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 X_1 through X_6. So, the values under columns ix1, ix2, and ix3 range from 1 to 6. The powers range from 0 (for ln(x)) to 4 and appear as such under columns px1, px2, and px3. The values of the orientation flags are as discussed above.



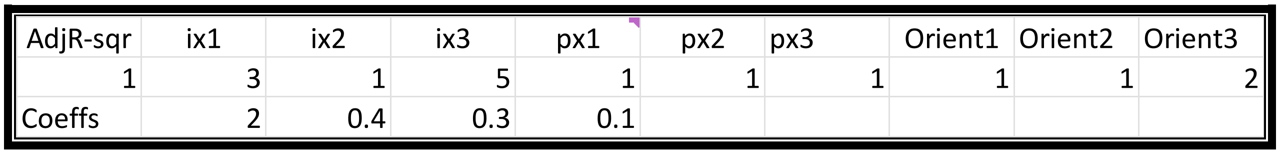
	ix1	ix2	ix3	px1	px2	px3	Orient1	Orient2	Orient3
Lb	1	1	1	0	0	0	1	1	2
Ub	6	6	6	4	4	4	1	2	2

*Figure 9.1. Sample sheet **Transf** for 3 independent variables with two cross-product terms in file data3.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 H2 to J2 contain the orientation flags.
    e. Cell B3 contains the intercept.
    f. Cells C3 to E3 contain the slopes of the three terms.



AdjR-sqr	ix1	ix2	ix3	px1	px2	px3	Orient1	Orient2	Orient3
1	3	1	5	1	1	1	1	1	2
Coeffs	2	0.4	0.3	0.1					

*Figure 9.2. Sample sheet **Results** for 3 selected independent variables in file*

10/ The Code for the Case of Three Independent Variables

Here is the code for file clem3.m. It is very similar to code of clem3.m. The additional statements examine the values of the orientation flags to determine the flow of program execution—whether the variable Xj appears in the numerator or denominator of the rational heteronomial.

```
clc
close all
clear all

global mdl
global xdata
global ydata

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');

ydata = xyData(:,1);
```

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

Lb = xyTransf(1,:);
Ub = xyTransf(2,:);
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}';
res(2,5) = mdl.Coefficients{4,1}';
writematrix(res(1,:), filename, 'Sheet', 'Results', 'Range', 'A2:Z2');
writematrix(res(2,2:5), 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))));


i1 = res(1,2);
i2 = res(1,3);
i3 = res(1,4);
px1 = res(1,5);
px2 = res(1,6);
px3 = res(1,7);
```

```
orient = res(1,9);


if px1==0
  sX1 = sprintf("log(X%d)", i1);
elseif px1==4
  sX1 = sprintf("sqrt(X%d)", i1);
elseif px1==1
  sX1 = sprintf("X%d", i1);
elseif px1>1
  sX1 = sprintf("X%d^%d", i1, px1);
elseif px1<0
  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==1
  sX2 = sprintf("X%d", i2);
elseif px2>1
  sX2 = sprintf("X%d^%d", i2, px2);
elseif px2<0
  sX2 = sprintf("1/X%d^%d", i2, abs(px2));
else
  sX2 = sprintf("X%d", i2);
end

if px3==0
  sX3 = sprintf("log(X%d)", i3);
elseif px3==4
  sX3 = sprintf("sqrt(X%d)", i3);
elseif px3==1
  sX3 = sprintf("X%d", i3);
elseif px3>1
  sX3 = sprintf("X%d^%d", i3, px3);
elseif px3<0
  sX3 = sprintf("1/X%d^%d", i3, abs(px3));
else
  sX3 = sprintf("X%d", i3);
end

if orient==1
  smodel = sprintf("Y = (%e + (%e)*%s + (%e)*%s)/\n(1 + (%e)*%s)", ...
    res(2,2), res(2,3), sX1, res(2,4), sX2, res(2,5), sX3);

else
  smodel = sprintf("Y = (%e + (%e)*%s)/\n(1 + (%e)*%s + (%e)*%s)", ...
    res(2,2), res(2,3), sX1, res(2,4), sX2, res(2,5), sX3);

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

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


  y = ydata;
  idx = coeff(1:3);
  px = coeff(4:6);
  orient = coeff(7:9);

  if idx(1)==idx(2) && orient(1) == orient(2)
    r = 0;
    return;
  end

  if idx(2)==idx(3) && orient(2) == orient(3)
    r = 0;
    return;
  end

  x1 = myfx(xdata(:,idx(1)),px(1));
  if orient(2)==1
    x2 = myfx(xdata(:,idx(2)),px(2));
  else
    x2 = -ydata.*myfx(xdata(:,idx(2)),px(2));
  end
  x3 = -ydata.*myfx(xdata(:,idx(3)),px(3));
  X=[x1 x2 x3];
  mdl = fitlm(X,y);
  r = mdl.Rsquared.Adjusted;
end

function ycalc = project(res)
  global xdata
  idx = res(1,2:4);
  px = res(1,5:7);
  orient = res(1,8:10);

  x1 = myfx(xdata(:,idx(1)),px(1));
  x2 = myfx(xdata(:,idx(2)),px(2));
  x3 = myfx(xdata(:,idx(3)),px(3));
  if orient(2) == 1
    ycalc = (res(2,2) + res(2,3) * x1 + res(2,4)*x2)./(1 + res(2,5)*x3);
  else
    ycalc = (res(2,2) + res(2,3) * x1 )./(1 + res(2,4)*x2 + res(2,5)*x3);
  end

end
```

The local variables orient handle the orientation flags and determine if a variable is part of the numerator or denominator.

The output of the above program is:

```
Program  clem3

2025-Jan-23 17_06_12
Please wait ...

Excel file used is C:\Users\nsham\Dropbox\MATLAB\Clement4\data3.xlsx
Diary file used is data3_2025-Jan-23 17_06_12.txt
------------- Calculating Max Iters ----------------
Max iters = 54,000
------------- Maximizing Fx ----------------
Iter 2, Fx = 9.788294e-01, X=[2, 1, 1, 0, 0, 0, 1, 1, 2, ]
Iter 3, Fx = 9.805682e-01, X=[3, 1, 1, 0, 0, 0, 1, 1, 2, ]
Iter 5, Fx = 9.870975e-01, X=[5, 1, 1, 0, 0, 0, 1, 1, 2, ]
Iter 77, Fx = 9.978101e-01, X=[5, 1, 3, 0, 0, 0, 1, 1, 2, ]
Iter 110, Fx = 9.996015e-01, X=[2, 1, 4, 0, 0, 0, 1, 1, 2, ]
Iter 111, Fx = 9.996074e-01, X=[3, 1, 4, 0, 0, 0, 1, 1, 2, ]
Iter 112, Fx = 9.997950e-01, X=[4, 1, 4, 0, 0, 0, 1, 1, 2, ]
Iter 132, Fx = 9.997991e-01, X=[6, 4, 4, 0, 0, 0, 1, 1, 2, ]
Progress  5.00%
Progress 10.00%
Iter 6843, Fx = 1.000000e+00, X=[3, 1, 5, 1, 1, 1, 1, 1, 2, ]
Regression model
Adjusted R-square = 1.000000000000

mdl =


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

Estimated Coefficients:
                   Estimate            SE                 tStat
pValue</strong>
```

```
                   ________   ____________________   ________________
______</strong>

    (Intercept)       2        5.2353692809189e-08    38201698.7280974      0
    x1              0.4       9.44597990772783e-10    423460566.195739      0
    x2              0.3       8.12987101378823e-10    369009544.544066      0
    x3              0.1       2.29281674595255e-10     436144755.90571      0


Number of observations: 100, Error degrees of freedom: 96
Root Mean Squared Error: 1.95e-07
R-squared: 1,  Adjusted R-Squared: 1
F-statistic vs. constant model: 1.09e+32, p-value = 0
Y = (2.000000e+00 + (4.000000e-01)*X3 + (3.000000e-01)*X1)/
(1 + (1.000000e-01)*X5)
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 four and five variables. In these cases, we have the first selected variable always in the numerator, the last variable always in the denominator, and all other variables can appear in either the numerator or denominator in a manner that maximizes the adjusted R-Square value.