

# Best Rational Heteronomial Model Selection III

by  
Namir Clement Shammas

## Contents

Introduction .....	1
The Excel Sheet .....	3
The MATLAB Code.....	9

## Introduction

This third part of the study expands on Part 2 and looks at rational heteronomials with two independent variables and with a **cross product**. This cross product multiplies the two independent variables, each raised to a distinct power. Each cross product requires arrays, matrices, and loops to handle two variables. Here is the full model:

$$Y^{yp} = (a + b_1 * X_1^{px11} + b_2 * X_2^{px12} + b_3 * X_1^{px13} * X_2^{px14}) / (1 + c_1 * X_1^{px211} + c_2 * X_2^{px22} + c_3 * X_1^{px23} * X_2^{px24}) \quad (1)$$

To use equation (1) in a multiple linearized form, given all the powers, we use:

$$Y^{yp} = a + b_1 * X_1^{px11} + b_2 * X_2^{px12} + b_3 * X_1^{px13} * X_2^{px14} - c_1 * Y^{yp} * X_1^{px211} - c_2 * Y^{yp} * X_2^{px22} - c_3 * Y^{yp} * X_1^{px23} * X_2^{px24} \quad (1b)$$

Notice that the denominator terms in equation (1) are multiplied by  $Y^{yp}$  and also **subtracted** from the terms that are in the numerator in equation (1).

The subset model sets that I will also cover are:

$$Y^{yp} = (a + b_1 * X_1^{px11} + b_2 * X_2^{px12}) / (1 + c_1 * X_1^{px211} + c_2 * X_2^{px22} + c_3 * X_1^{px23} * X_2^{px24}) \quad (2)$$

$$Y^{yp} = (a + b_1 * X_1^{px11} + b_2 * X_1^{px12} * X_2^{px13}) /$$

$$(1 + c_1 * X_1^{px211} + c_2 * X_2^{px22} + c_3 * X_1^{px23} * X_2^{px24}) \quad (3)$$

$$Y^{yp} = (a + b_1 * X_2^{px11} + b_2 * X_1^{px12} * X_2^{px13}) / (1 + c_1 * X_1^{px211} + c_2 * X_2^{px22} + c_3 * X_1^{px23} * X_2^{px24}) \quad (4)$$

$$Y^{yp} = (a + b_1 * X_1^{px11}) / (1 + c_1 * X_1^{px211} + c_2 * X_2^{px22} + c_3 * X_1^{px23} * X_2^{px24}) \quad (5)$$

$$Y^{yp} = (a + b_1 * X_2^{px11}) / (1 + c_1 * X_1^{px211} + c_2 * X_2^{px22} + c_3 * X_1^{px23} * X_2^{px24}) \quad (6)$$

$$Y^{yp} = (a + b_1 * X_1^{px11} * X_2^{px12}) / (1 + c_1 * X_1^{px211} + c_2 * X_2^{px22} + c_3 * X_1^{px23} * X_2^{px24}) \quad (7)$$

Equations (1) to (7) show that the denominator terms remain fixed. If we swap the numerator and denominator in equations (2) to (7) we add six more model sets. In this case, and the case of rational heteronomials with three or more independent variables, having cross product terms, you might be better off using numerical optimization.

I will limit working with seven model sets. The following table lists the model selection functions and the configurations of the independent variables appearing in the numerator and denominator of the rational heteronomials.

Model Selection Function	Variables in the Numerator	Variables in the Denominator	Number of Cases
gx633	X1, X2, and cross-product term.	X1, X2, and cross-product term.	1
gx623	X1 or X2, and cross-product term	X1, X2, and cross-product term.	2
gx623x	The cross-product term	X1, X2, and cross-product term.	1
gx613	X1 or X2	X1, X2, and cross-product term.	2
gx613x	The cross-product term	X1, X2, and cross-product term.	2

*Table 1. Cases for two independent variables.*

## The Excel Sheet

Let's look at the Excel sheet go66x.xlsx. It is the one I use to handle modeling with scheme outlined by table 1. The workbook has the following sheets:

- The **Data** sheet has the following columns (see Figure 1):
  - Column A contains the values for the dependent variable Y. The column has the header Y in the first row.
  - Column B and C contain the values for the two independent variables. The columns have the headers X1 and X2 in the first row.
- The **T633** sheet contains information for the transformation ranges. It has the following columns (see Figure 2.1):
  - Column A has the header Y in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable Y.
  - Column B has the header X1 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X1 in the numerator part of the rational heteronomial.
  - Column C has the header X2 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X2 in the numerator part of the rational heteronomial.
  - Column D has the header ZX1 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X1 in the cross-product of the numerator part of the rational heteronomial.
  - Column E has the header ZX2 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X2 in the cross-product of the numerator part of the rational heteronomial.
  - Column F has the header YX1 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X1 in the denominator part of the rational heteronomial.
  - Column G has the header YX2 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X2 in the denominator part of the rational heteronomial.
  - Column H has the header YZX1 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power

used with variable  $X_1$  in the cross-product of the denominator part of the rational heteronomial.

- Column I has the header  $YZX_2$  in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable  $X_2$  in the cross-product of the denominator part of the rational heteronomial.
- Sheet **T623** contains information for the transformation ranges. It has the following columns (see Figure 2.2):
  - Column A has the header  $Y$  in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable  $Y$ .
  - Column B has the header  $X_1$  in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable  $X_1$  in the numerator part of the rational heteronomial.
  - Column C has the header  $ZX_1$  in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable  $X_1$  in the cross-product of the numerator part of the rational heteronomial.
  - Column D has the header  $ZX_2$  in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable  $X_2$  in the cross-product of the numerator part of the rational heteronomial.
  - Column E has the header  $YX_1$  in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable  $X_1$  in the denominator part of the rational heteronomial.
  - Column F has the header  $YX_2$  in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable  $X_2$  in the denominator part of the rational heteronomial.
  - Column G has the header  $YZX_1$  in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable  $X_1$  in the cross-product of the denominator part of the rational heteronomial.
  - Column H has the header  $YZX_2$  in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable  $X_2$  in the cross-product of the denominator part of the rational heteronomial.

- Sheet **T623x** contains information for the transformation ranges. It has the following columns (see Figure 2.3):
  - Column A has the header Y in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable Y.
  - Column B has the header X1 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X1 in the numerator part of the rational heteronomial.
  - Column C has the header ZX1 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X1 in the cross-product of the numerator part of the rational heteronomial.
  - Column D has the header ZX2 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X2 in the cross-product of the numerator part of the rational heteronomial.
  - Column E has the header YX1 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X1 in the denominator part of the rational heteronomial.
  - Column F has the header YX2 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X2 in the denominator part of the rational heteronomial.
  - Column G has the header YZX1 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X1 in the cross-product of the denominator part of the rational heteronomial.
  - Column H has the header YZX2 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X2 in the cross-product of the denominator part of the rational heteronomial.
- The **T613** sheet contains information for the transformation ranges. It has the following columns (see Figure 2.4):
  - Column A has the header Y in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable Y.

- Column B has the header X1 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X1 in the numerator part of the rational heteronomial.
- Column C has the header YX1 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X1 in the denominator part of the rational heteronomial.
- Column D has the header YX2 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X2 in the denominator part of the rational heteronomial.
- Column E has the header YZX1 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X1 in the cross-product of the denominator part of the rational heteronomial.
- Column F has the header YZX2 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X2 in the cross-product of the denominator part of the rational heteronomial.
- The **T613x** sheet contains information for the transformation ranges. It has the following columns (see Figure 2.5):
  - Column A has the header Y in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable Y.
  - Column B has the header ZX1 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X1 in the cross-product of the numerator part of the rational heteronomial.
  - Column C has the header ZX2 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X2 in the cross-product of the numerator part of the rational heteronomial.
  - Column D has the header YX1 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X1 in the denominator part of the rational heteronomial.
  - Column E has the header YX2 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X2 in the denominator part of the rational heteronomial.

- Column F has the header YZX1 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X1 in the cross-product of the denominator part of the rational heteronomial.
- Column G has the header YZX2 in row 1. Rows 2, 3, and 4 contain the values for the initial power, power increment, and final power used with variable X2 in the cross-product of the denominator part of the rational heteronomial.

Y	X1	X2
12.18213323	1	55
12.54696677	2	67
8.430565344	3	41
6.604571883	4	29
6.00516746	5	25
5.482942824	6	21
5.63198061	7	22
6.274235797	8	27
4.363787374	9	10
8.4397117	10	45
4.759636712	11	12
5.167160174	12	15
5.364859283	13	16
8.9211312	14	48
5.077018806	15	11

Figure 1. The **Data** sheet in file *go44.xlsx*.

Y	X1	X2	ZX1	ZX2	YX1	YX2	YZX1	YZX2
0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2

Figure 2.1. The **T633** sheet in file *go66x.xlsx*.

Y	X1	X2	ZX1	ZX2	YX1	YX2	YZX1	YZX2
0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2

Figure 2.2. The **T623** sheet in file *go44.xlsxt*.

Y	X1	ZX1	ZX2	YX1	YX2	YZX1	YZX2
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2

Figure 2. The **T623x** sheet in file *go44.xlsxt*.

Y	X1	YX1	YX2	YZX1	YZX2
0	0	0	0	0	0
1	1	1	1	1	1
2	2	2	2	2	2

Figure 2.3. The **T613** sheet in file *go44.xlsxt*.



Y	ZX1	ZX2	YX1	YX2	YZX1	YZX2
0	0	0	0	0	0	0
1	1	1	1	1	1	1
2	2	2	2	2	2	2

Figure 2.4. The **T613x** sheet in file *go44.xlsxt*.

## The MATLAB Code

Here is the MATLAB code for function `gx633()`:

```
end function [bestAdjR2,S] = gx633(ydata,xdata,xyTransf)
%GX633 Summary of this function goes here
% Detailed explanation goes here
S.model = "(X1+X2+X1*X2)/(X1+X2+X1*X2)";
maxrows = length(ydata);
x1data = xdata(:,1);
x2data = xdata(:,2);

yPwrs = xyTransf(1:3,1);
x1Pwrs = xyTransf(1:3,2);
x2Pwrs = xyTransf(1:3,3);
zx1Pwrs = xyTransf(1:3,4);
zx2Pwrs = xyTransf(1:3,5);
yx1Pwrs = xyTransf(1:3,6);
yx2Pwrs = xyTransf(1:3,7);
yzx1Pwrs = xyTransf(1:3,8);
yzx2Pwrs = xyTransf(1:3,9);

yBuffer = zeros(maxrows,1);
x1Buffer = zeros(maxrows,1);
x2Buffer = zeros(maxrows,1);
zx1Buffer = zeros(maxrows,1);
zx2Buffer = zeros(maxrows,1);
yx1Buffer = zeros(maxrows,1);
yx2Buffer = zeros(maxrows,1);
yzx1Buffer = zeros(maxrows,1);
yzx2Buffer = zeros(maxrows,1);

yCols = 0;
for yPwr =yPwrs(1):yPwrs(2):yPwrs(3)
    yCols = yCols + 1;
    yBuffer(:,yCols)= fx(ydata,yPwr);
end

x1Cols = 0;
for x1Pwr =x1Pwrs(1):x1Pwrs(2):x1Pwrs(3)
    x1Cols = x1Cols + 1;
    x1Buffer(:,x1Cols)= fx(x1data,x1Pwr);
end
```

```

x2Cols = 0;
for x2Pwr =x2Pwrs(1):x2Pwrs(2):x2Pwrs(3)
    x2Cols = x2Cols + 1;
    x2Buffer(:,x2Cols)= fx(x2data,x2Pwr);
end

zx1Cols = 0;
for zx1Pwr =zx1Pwrs(1):zx1Pwrs(2):zx1Pwrs(3)
    zx1Cols = zx1Cols + 1;
    zx1Buffer(:,zx1Cols)= fx(x1data,zx1Pwr);
end

zx2Cols = 0;
for zx2Pwr =zx2Pwrs(1):zx2Pwrs(2):zx2Pwrs(3)
    zx2Cols = zx2Cols + 1;
    zx2Buffer(:,zx2Cols) = fx(x2data,zx2Pwr);
end

yx1Cols = 0;
for yx1Pwr =yx1Pwrs(1):yx1Pwrs(2):yx1Pwrs(3)
    yx1Cols = yx1Cols + 1;
    yx1Buffer(:,yx1Cols)= fx(x1data,yx1Pwr);
end

yx2Cols = 0;
for yx2Pwr =yx2Pwrs(1):yx2Pwrs(2):yx2Pwrs(3)
    yx2Cols = yx2Cols + 1;
    yx2Buffer(:,yx2Cols)= fx(x2data,yx2Pwr);
end

yzx1Cols = 0;
for yzx1Pwr =yzx1Pwrs(1):yzx1Pwrs(2):yzx1Pwrs(3)
    yzx1Cols = yzx1Cols + 1;
    yzx1Buffer(:,yzx1Cols)= fx(x1data,yzx1Pwr);
end

yzx2Cols = 0;
for yzx2Pwr =yzx2Pwrs(1):yzx2Pwrs(2):yzx2Pwrs(3)
    yzx2Cols = yzx2Cols + 1;
    yzx2Buffer(:,yzx2Cols)= fx(x2data,yzx2Pwr);
end

bestAdjR2 = 0;

yCols = 0;
for yPwr =yPwrs(1):yPwrs(2):yPwrs(3)
    yCols = yCols + 1;
    y = yBuffer(:,yCols);

    x1Cols = 0;
    for x1Pwr =x1Pwrs(1):x1Pwrs(2):x1Pwrs(3)
        x1Cols = x1Cols + 1;
        x1 = x1Buffer(:,x1Cols);

        x2Cols = 0;

```

```

for x2Pwr =x2Pwrs(1):x2Pwrs(2):x2Pwrs(3)
    x2Cols = x2Cols + 1;
    x2 = x2Buffer(:,x2Cols);

    zx1Cols = 0 ;
    for zx1Pwr =zx1Pwrs(1):zx1Pwrs(2):zx1Pwrs(3)
        zx1Cols = zx1Cols + 1;
        zx1 = zx1Buffer(:, zx1Cols);

        zx2Cols = 0;
        for zx2Pwr =zx2Pwrs(1):zx2Pwrs(2):zx2Pwrs(3)
            zx2Cols = zx2Cols + 1;
            zx2 = zx2Buffer(:, zx2Cols);

            zx = zx1 .* zx2;

            yx1Cols = 0;
            for yx1Pwr =yx1Pwrs(1):yx1Pwrs(2):yx1Pwrs(3)
                yx1Cols = yx1Cols + 1;
                yx1 = yx1Buffer(:,yx1Cols);

                yx1 = -1 .* y .* yx1;

                yx2Cols = 0;
                for yx2Pwr =yx2Pwrs(1):yx2Pwrs(2):yx2Pwrs(3)
                    yx2Cols = yx2Cols + 1;
                    yx2 = yx2Buffer(:,yx2Cols);

                    yx2 = -1 .* y .* yx2;

                    yzx1Cols = 0;
                    for yzx1Pwr =yzx1Pwrs(1):yzx1Pwrs(2):yzx1Pwrs(3)
                        yzx1Cols = yzx1Cols + 1;
                        yzx1 = yzx1Buffer(:, yzx1Cols);

                        yzx2Cols = 0;
                        for yzx2Pwr =yzx2Pwrs(1):yzx2Pwrs(2):yzx2Pwrs(3)
                            yzx2Cols = yzx2Cols + 1;
                            yzx2 = yzx2Buffer(:, yzx2Cols);

                            yzx = -1 .* y .* yzx1 .* yzx2;

                        X = [x1 x2 zx yx1 yx2 yzx];

                        mdl = fitlm(X,y);

                        if mdl.Rsquared.Adjusted >= bestAdjR2
                            bestAdjR2 = mdl.Rsquared.Adjusted;
                            S.bestAdjR2 = bestAdjR2;
                            S.bestYpwr = yPwr;
                            S.bestX1pwr = x1Pwr;
                            S.bestX2pwr = x2Pwr;
                            S.bestZX1pwr = zx1Pwr;
                            S.bestZX2pwr = zx2Pwr;
                            S.bestYX1pwr = yx1Pwr;
                            S.bestYX2pwr = yx2Pwr;
                            S.bestZYX1pwr = yzx1Pwr;

```



```

yPwrs = xyTransf(1:3,1);
x1Pwrs = xyTransf(1:3,2);
x2Pwrs = xyTransf(1:3,3);
yx1Pwrs = xyTransf(1:3,4);
yx2Pwrs = xyTransf(1:3,5);
yzx1Pwrs = xyTransf(1:3,6);
yzx2Pwrs = xyTransf(1:3,7);
yBuffer = zeros(maxrows,1);
x1Buffer = zeros(maxrows,1);
x2Buffer = zeros(maxrows,1);
yx1Buffer = zeros(maxrows,1);
yx2Buffer = zeros(maxrows,1);
yzx1Buffer = zeros(maxrows,1);
yzx2Buffer = zeros(maxrows,1);

yCols = 0;
for yPwr =yPwrs(1):yPwrs(2):yPwrs(3)
    yCols = yCols + 1;
    yBuffer(:,yCols)= fx(ydata,yPwr);
end

x1Cols = 0;
for x1Pwr =x1Pwrs(1):x1Pwrs(2):x1Pwrs(3)
    x1Cols = x1Cols + 1;
    x1Buffer(:,x1Cols)= fx(x1data,x1Pwr);
end

x2Cols = 0;
for x2Pwr =x2Pwrs(1):x2Pwrs(2):x2Pwrs(3)
    x2Cols = x2Cols + 1;
    x2Buffer(:,x2Cols)= fx(x2data,x2Pwr);
end

yx1Cols = 0;
for yx1Pwr =yx1Pwrs(1):yx1Pwrs(2):yx1Pwrs(3)
    yx1Cols = yx1Cols + 1;
    yx1Buffer(:,yx1Cols)= fx(x1data,yx1Pwr);
end

yx2Cols = 0;
for yx2Pwr =yx2Pwrs(1):yx2Pwrs(2):yx2Pwrs(3)
    yx2Cols = yx2Cols + 1;
    yx2Buffer(:,yx2Cols)= fx(x2data,yx2Pwr);
end

yzx1Cols = 0;
for yzx1Pwr =yzx1Pwrs(1):yzx1Pwrs(2):yzx1Pwrs(3)
    yzx1Cols = yzx1Cols + 1;
    yzx1Buffer(:,yzx1Cols)= fx(x1data,yzx1Pwr);
end

yzx2Cols = 0;
for yzx2Pwr =yzx2Pwrs(1):yzx2Pwrs(2):yzx2Pwrs(3)
    yzx2Cols = yzx2Cols + 1;
    yzx2Buffer(:,yzx2Cols)= fx(x2data,yzx2Pwr);
end

```

```

bestAdjR2 = 0;

yCols = 0;
for yPwr =yPwrs(1):yPwrs(2):yPwrs(3)
    yCols = yCols + 1;
    y = yBuffer(:,yCols);

    x1Cols = 0;
    for x1Pwr =x1Pwrs(1):x1Pwrs(2):x1Pwrs(3)
        x1Cols = x1Cols + 1;
        x1 = x1Buffer(:,x1Cols);

        x2Cols = 0;
        for x2Pwr =x2Pwrs(1):x2Pwrs(2):x2Pwrs(3)
            x2Cols = x2Cols + 1;
            x2 = x2Buffer(:,x2Cols);

            yx1Cols = 0;
            for yx1Pwr =yx1Pwrs(1):yx1Pwrs(2):yx1Pwrs(3)
                yx1Cols = yx1Cols + 1;
                yx1 = yx1Buffer(:,yx1Cols);

                yx1 = -1 .* y .* yx1;

                yx2Cols = 0;
                for yx2Pwr =yx2Pwrs(1):yx2Pwrs(2):yx2Pwrs(3)
                    yx2Cols = yx2Cols + 1;
                    yx2 = yx2Buffer(:,yx2Cols);

                    yx2 = -1 .* y .* yx2;

                    yzx1Cols = 0;
                    for yzx1Pwr =yzx1Pwrs(1):yzx1Pwrs(2):yzx1Pwrs(3)
                        yzx1Cols = yzx1Cols + 1;
                        yzx1 = yzx1Buffer(:, yzx1Cols);

                        yzx2Cols = 0;
                        for yzx2Pwr =yzx2Pwrs(1):yzx2Pwrs(2):yzx2Pwrs(3)
                            yzx2Cols = yzx2Cols + 1;
                            yzx2 = yzx2Buffer(:, yzx2Cols);

                            yzx = -1 .* y .* yzx1 .* yzx2;

                        X = [x1 x2 yx1 yx2 yzx];

                        mdl = fitlm(X,y);

                        if mdl.Rsquared.Adjusted >= bestAdjR2
                            bestAdjR2 = mdl.Rsquared.Adjusted;
                            S.bestAdjR2 = bestAdjR2;
                            S.bestYpwr = yPwr;
                            S.bestX1pwr = x1Pwr;
                            S.bestX2pwr = x2Pwr;
                            S.bestYX1pwr = yx1Pwr;
                            S.bestYX2pwr = yx2Pwr;
                            S.bestZYX1pwr = yzx1Pwr;

```

```

                S.bestZYX2pwr = yzx2Pwr;
                S.bestmdl = mdl;
            end
        end
    end
end
end
end
end
end
end
end
end
end

function y = fx(x,pwr)
    if pwr > 0
        y = x.^pwr;
    elseif pwr < 0
        y = 1./x.^abs(pwr);
    else
        y = log(x);
    end
end
end

```

The code for function gx623() is a subset of that of function gx633() as it works with five terms. The different variables, arrays, matrices, and loops are organized to handle the two terms in numerator part and three terms in the denominator part.

Here is the MATLAB code for function gx623x():

```

function [bestAdjR2,S] = gx623x(ydata,xdata,xidx,xyTransf)
%GX623x Summary of this function goes here
% Detailed explanation goes here
    S.model = strcat("X1",num2str(xidx),"X1*X2)/(X1+X2+X1*X2)");
    maxrows = length(ydata);
    x1data = xdata(:,1);
    x2data = xdata(:,2);
    zdata = xdata(:,xidx);
    yPwrs = xyTransf(1:3,1);
    x1Pwrs = xyTransf(1:3,2);
    zx1Pwrs = xyTransf(1:3,3);
    zx2Pwrs = xyTransf(1:3,4);
    yx1Pwrs = xyTransf(1:3,5);
    yx2Pwrs = xyTransf(1:3,6);
    yzx1Pwrs = xyTransf(1:3,7);
    yzx2Pwrs = xyTransf(1:3,8);
    yBuffer = zeros(maxrows,1);
    x1Buffer = zeros(maxrows,1);
    zx1Buffer = zeros(maxrows,1);
    zx2Buffer = zeros(maxrows,1);
    yx1Buffer = zeros(maxrows,1);
    yx2Buffer = zeros(maxrows,1);
    yzx1Buffer = zeros(maxrows,1);

```

```

yzx2Buffer = zeros(maxrows,1);

yCols = 0;
for yPwr =yPwrs(1):yPwrs(2):yPwrs(3)
    yCols = yCols + 1;
    yBuffer(:,yCols)= fx(ydata,yPwr);
end

x1Cols = 0;
for x1Pwr =x1Pwrs(1):x1Pwrs(2):x1Pwrs(3)
    x1Cols = x1Cols + 1;
    x1Buffer(:,x1Cols)= fx(zdata,x1Pwr);
end

zx1Cols = 0;
for zx1Pwr =zx1Pwrs(1):zx1Pwrs(2):zx1Pwrs(3)
    zx1Cols = zx1Cols + 1;
    zx1Buffer(:,zx1Cols)= fx(x1data,zx1Pwr);
end

zx2Cols = 0;
for zx2Pwr =zx2Pwrs(1):zx2Pwrs(2):zx2Pwrs(3)
    zx2Cols = zx2Cols + 1;
    zx2Buffer(:,zx2Cols)= fx(x2data,zx2Pwr);
end

yx1Cols = 0;
for yx1Pwr =yx1Pwrs(1):yx1Pwrs(2):yx1Pwrs(3)
    yx1Cols = yx1Cols + 1;
    yx1Buffer(:,yx1Cols)= fx(x1data,yx1Pwr);
end

yx2Cols = 0;
for yx2Pwr =yx2Pwrs(1):yx2Pwrs(2):yx2Pwrs(3)
    yx2Cols = yx2Cols + 1;
    yx2Buffer(:,yx2Cols)= fx(x2data,yx2Pwr);
end

yzx1Cols = 0;
for yzx1Pwr =yzx1Pwrs(1):yzx1Pwrs(2):yzx1Pwrs(3)
    yzx1Cols = yzx1Cols + 1;
    yzx1Buffer(:,yzx1Cols)= fx(x1data,yzx1Pwr);
end

yzx2Cols = 0;
for yzx2Pwr =yzx2Pwrs(1):yzx2Pwrs(2):yzx2Pwrs(3)
    yzx2Cols = yzx2Cols + 1;
    yzx2Buffer(:,yzx2Cols)= fx(x2data,yzx2Pwr);
end

bestAdjR2 = 0;

yCols = 0;
for yPwr =yPwrs(1):yPwrs(2):yPwrs(3)
    yCols = yCols + 1;
    y = yBuffer(:,yCols);

```



```

x1Cols = 0;
for x1Pwr =x1Pwrs(1):x1Pwrs(2):x1Pwrs(3)
    x1Cols = x1Cols + 1;
    x1 = x1Buffer(:,x1Cols);

    zx1Cols = 0;
    for zx1Pwr =zx1Pwrs(1):zx1Pwrs(2):zx1Pwrs(3)
        zx1Cols = zx1Cols + 1;
        zx1 = zx1Buffer(:, zx1Cols);

        zx2Cols = 0;
        for zx2Pwr =zx2Pwrs(1):zx2Pwrs(2):zx2Pwrs(3)
            zx2Cols = zx2Cols + 1;
            zx2 = zx2Buffer(:, zx2Cols);

            zx = zx1 .* zx2;

            yx1Cols = 0;
            for yx1Pwr =yx1Pwrs(1):yx1Pwrs(2):yx1Pwrs(3)
                yx1Cols = yx1Cols + 1;
                yx1 = yx1Buffer(:,yx1Cols);

                yx1 = -1 .* y .* yx1;

                yx2Cols = 0;
                for yx2Pwr =yx2Pwrs(1):yx2Pwrs(2):yx2Pwrs(3)
                    yx2Cols = yx2Cols + 1;
                    yx2 = yx2Buffer(:,yx2Cols);

                    yx2 = -1 .* y .* yx2;

                    yzx1Cols = 0;
                    for yzx1Pwr =yzx1Pwrs(1):yzx1Pwrs(2):yzx1Pwrs(3)
                        yzx1Cols = yzx1Cols + 1;
                        yzx1 = yzx1Buffer(:, yzx1Cols);

                        yzx2Cols = 0;
                        for yzx2Pwr =yzx2Pwrs(1):yzx2Pwrs(2):yzx2Pwrs(3)
                            yzx2Cols = yzx2Cols + 1;
                            yzx2 = yzx2Buffer(:, yzx2Cols);

                            yzx = -1 .* y .* yzx1 .* yzx2;

                        X = [x1 zx yx1 yx2 yzx];

                        mdl = fitlm(X,y);

                        if mdl.Rsquared.Adjusted >= bestAdjR2
                            bestAdjR2 = mdl.Rsquared.Adjusted;
                            S.bestAdjR2 = bestAdjR2;
                            S.bestYpwr = yPwr;
                            S.bestX1pwr = x1Pwr;
                            S.bestZX1pwr = zx1Pwr;
                            S.bestZX2pwr = zx2Pwr;
                            S.bestYX1pwr = yx1Pwr;

```



```

yzx2Buffer = zeros(maxrows,1);

yCols = 0;
for yPwr =yPwrs(1):yPwrs(2):yPwrs(3)
    yCols = yCols + 1;
    yBuffer(:,yCols)= fx(ydata,yPwr);
end

zCols = 0;
for zPwr =zPwrs(1):zPwrs(2):zPwrs(3)
    zCols = zCols + 1;
    zBuffer(:,zCols)= fx(zdata,zPwr);
end

yx1Cols = 0;
for yx1Pwr =yx1Pwrs(1):yx1Pwrs(2):yx1Pwrs(3)
    yx1Cols = yx1Cols + 1;
    yx1Buffer(:,yx1Cols)= fx(x1data,yx1Pwr);
end

yx2Cols = 0;
for yx2Pwr =yx2Pwrs(1):yx2Pwrs(2):yx2Pwrs(3)
    yx2Cols = yx2Cols + 1;
    yx2Buffer(:,yx2Cols)= fx(x2data,yx2Pwr);
end

yzx1Cols = 0;
for yzx1Pwr =yzx1Pwrs(1):yzx1Pwrs(2):yzx1Pwrs(3)
    yzx1Cols = yzx1Cols + 1;
    yzx1Buffer(:,yzx1Cols)= fx(x1data,yzx1Pwr);
end

yzx2Cols = 0;
for yzx2Pwr =yzx2Pwrs(1):yzx2Pwrs(2):yzx2Pwrs(3)
    yzx2Cols = yzx2Cols + 1;
    yzx2Buffer(:,yzx2Cols)= fx(x2data,yzx2Pwr);
end

bestAdjR2 = 0;

yCols = 0;
for yPwr =yPwrs(1):yPwrs(2):yPwrs(3)
    yCols = yCols + 1;
    y = yBuffer(:,yCols);

    zCols = 0;
    for zPwr =zPwrs(1):zPwrs(2):zPwrs(3)
        zCols = zCols + 1;
        z = zBuffer(:,zCols);

        yx1Cols = 0;
        for yx1Pwr =yx1Pwrs(1):yx1Pwrs(2):yx1Pwrs(3)
            yx1Cols = yx1Cols + 1;
            yx1 = yx1Buffer(:,yx1Cols);

            yx1 = -1 .* y .* yx1;

```

```

    yx2Cols = 0;
    for yx2Pwr = yx2Pwrs(1):yx2Pwrs(2):yx2Pwrs(3)
        yx2Cols = yx2Cols + 1;
        yx2 = yx2Buffer(:, yx2Cols);

        yx2 = -1 .* y .* yx2;

        yzx1Cols = 0;
        for yzx1Pwr = yzx1Pwrs(1):yzx1Pwrs(2):yzx1Pwrs(3)
            yzx1Cols = yzx1Cols + 1;
            yzx1 = yzx1Buffer(:, yzx1Cols);

            yzx2Cols = 0;
            for yzx2Pwr = yzx2Pwrs(1):yzx2Pwrs(2):yzx2Pwrs(3)
                yzx2Cols = yzx2Cols + 1;
                yzx2 = yzx2Buffer(:, yzx2Cols);

                yzx = -1 .* y .* yzx1 .* yzx2;

                X = [z yx1 yx2 yzx];

                mdl = fitlm(X, y);

                if mdl.Rsquared.Adjusted >= bestAdjR2
                    bestAdjR2 = mdl.Rsquared.Adjusted;
                    S.bestAdjR2 = bestAdjR2;
                    S.bestYpwr = yPwr;
                    S.bestZpwr = zPwr;
                    S.bestYX1pwr = yx1Pwr;
                    S.bestYX2pwr = yx2Pwr;
                    S.bestZYX1pwr = yzx1Pwr;
                    S.bestZYX2pwr = yzx2Pwr;
                    S.bestmdl = mdl;
                end
            end
        end
    end
end
end
end
end
end
end
end
end

function y = fx(x, pwr)
    if pwr > 0
        y = x.^pwr;
    elseif pwr < 0
        y = 1./x.^abs(pwr);
    else
        y = log(x);
    end
end
end

```

The code for function gx613() is a subset of that of function gx633() as it works with four terms. The different variables, arrays, matrices, and loops are organized

are organized to handle the one term in numerator part and three terms in the denominator part.

Here is the code for the function gx613x():

```
function [bestAdjR2,S] = gx613x(ydata,xdata,xyTransf)
%GX613x Summary of this function goes here
% Detailed explanation goes here
S.model = "(X1*X2)/(X1+X2+X1*X2)";
maxrows = length(ydata);
x1data = xdata(:,1);
x2data = xdata(:,2);
yPwrs = xyTransf(1:3,1);
zx1Pwrs = xyTransf(1:3,2);
zx2Pwrs = xyTransf(1:3,3);
yx1Pwrs = xyTransf(1:3,4);
yx2Pwrs = xyTransf(1:3,5);
yzx1Pwrs = xyTransf(1:3,6);
yzx2Pwrs = xyTransf(1:3,7);
yBuffer = zeros(maxrows,1);
zx1Buffer = zeros(maxrows,1);
zx2Buffer = zeros(maxrows,1);
yx1Buffer = zeros(maxrows,1);
yx2Buffer = zeros(maxrows,1);
yzx1Buffer = zeros(maxrows,1);
yzx2Buffer = zeros(maxrows,1);

yCols = 0;
for yPwr =yPwrs(1):yPwrs(2):yPwrs(3)
    yCols = yCols + 1;
    yBuffer(:,yCols)= fx(ydata,yPwr);
end

zx1Cols = 0;
for zx1Pwr =zx1Pwrs(1):zx1Pwrs(2):zx1Pwrs(3)
    zx1Cols = zx1Cols + 1;
    zx1Buffer(:,zx1Cols)= fx(x1data,zx1Pwr);
end

zx2Cols = 0;
for zx2Pwr =zx2Pwrs(1):zx2Pwrs(2):zx2Pwrs(3)
    zx2Cols = zx2Cols + 1;
    zx2Buffer(:,zx2Cols)= fx(x2data,zx2Pwr);
end

yx1Cols = 0;
for yx1Pwr =yx1Pwrs(1):yx1Pwrs(2):yx1Pwrs(3)
    yx1Cols = yx1Cols + 1;
    yx1Buffer(:,yx1Cols)= fx(x1data,yx1Pwr);
end

yx2Cols = 0;
for yx2Pwr =yx2Pwrs(1):yx2Pwrs(2):yx2Pwrs(3)
    yx2Cols = yx2Cols + 1;
    yx2Buffer(:,yx2Cols)= fx(x2data,yx2Pwr);
end
```

```

end

yzx1Cols = 0;
for yzx1Pwr = yzx1Pwrs(1):yzx1Pwrs(2):yzx1Pwrs(3)
    yzx1Cols = yzx1Cols + 1;
    yzx1Buffer(:, yzx1Cols) = fx(x1data, yzx1Pwr);
end

yzx2Cols = 0;
for yzx2Pwr = yzx2Pwrs(1):yzx2Pwrs(2):yzx2Pwrs(3)
    yzx2Cols = yzx2Cols + 1;
    yzx2Buffer(:, yzx2Cols) = fx(x2data, yzx2Pwr);
end

bestAdjR2 = 0;

yCols = 0;
for yPwr = yPwrs(1):yPwrs(2):yPwrs(3)
    yCols = yCols + 1;
    y = yBuffer(:, yCols);

    zx1Cols = 0 ;
    for zx1Pwr = zx1Pwrs(1):zx1Pwrs(2):zx1Pwrs(3)
        zx1Cols = zx1Cols + 1;
        zx1 = zx1Buffer(:, zx1Cols);

        zx2Cols = 0 ;
        for zx2Pwr = zx2Pwrs(1):zx2Pwrs(2):zx2Pwrs(3)
            zx2Cols = zx2Cols + 1;
            zx2 = zx2Buffer(:, zx2Cols);

            zx = zx1 .* zx2;

            yx1Cols = 0 ;
            for yx1Pwr = yx1Pwrs(1):yx1Pwrs(2):yx1Pwrs(3)
                yx1Cols = yx1Cols + 1;
                yx1 = yx1Buffer(:, yx1Cols);

                yx1 = -1 .* y .* yx1;

                yx2Cols = 0 ;
                for yx2Pwr = yx2Pwrs(1):yx2Pwrs(2):yx2Pwrs(3)
                    yx2Cols = yx2Cols + 1;
                    yx2 = yx2Buffer(:, yx2Cols);

                    yx2 = -1 .* y .* yx2;

                    yzx1Cols = 0 ;
                    for yzx1Pwr = yzx1Pwrs(1):yzx1Pwrs(2):yzx1Pwrs(3)
                        yzx1Cols = yzx1Cols + 1;
                        yzx1 = yzx1Buffer(:, yzx1Cols);

                        yzx2Cols = 0 ;
                        for yzx2Pwr = yzx2Pwrs(1):yzx2Pwrs(2):yzx2Pwrs(3)
                            yzx2Cols = yzx2Cols + 1;

```

```

        yzx2 = yzx2Buffer(:, yzx2Cols);

        yzx = -1 .* y .* yzx1 .* yzx2;

        X = [zx yx1 yx2 yzx];

        mdl = fitlm(X,y);

        if mdl.Rsquared.Adjusted >= bestAdjR2
            bestAdjR2 = mdl.Rsquared.Adjusted;
            S.bestAdjR2 = bestAdjR2;
            S.bestYpwr = yPwr;
            S.bestZX1pwr = zx1Pwr;
            S.bestZX2pwr = zx2Pwr;
            S.bestYX1pwr = yx1Pwr;
            S.bestYX2pwr = yx2Pwr;
            S.bestZYX1pwr = yzx1Pwr;
            S.bestZYX2pwr = yzx2Pwr;
            S.bestmdl = mdl;
        end
    end
end
end
end
end
end
end
end

function y = fx(x,pwr)
    if pwr > 0
        y = x.^pwr;
    elseif pwr < 0
        y = 1./x.^abs(pwr);
    else
        y = log(x);
    end
end
end

```

The code for function gx613x() is a subset of that of function gx633() as it works with four terms. The different variables, arrays, matrices, and loops are organized to handle the one term in numerator part and three terms in the denominator part.

Here is the code for the test program go66x.m:

```

clc
clear all

zBest = 0;

filename = "Go66x.xlsx";

```

```

xyData = readmatrix(filename,'Sheet','Data');
ydata =xyData(:,1);
xdata = xyData(:,2:end);

xyTransf = readmatrix(filename,'Sheet','T633');
[R633,S633] = gx633(ydata,xdata,xyTransf)

S633.bestmdl
if R633 > zBest
    zBest = R633;
    bestS = S633;
end

xyTransf = readmatrix(filename,'Sheet','T623');
[R623, S623] = gx623(ydata,xdata,xyTransf)
S623.bestmdl
if R623 > zBest
    zBest = R623;
    bestS = S623;
end

xyTransf = readmatrix(filename,'Sheet','T623x');
[R623x1,S623x1] = gx623x(ydata,xdata,1,xyTransf)
S623x1.bestmdl
if R623x1 > zBest
    zBest = R623x1;
    bestS = S623x1;
end

xyTransf = readmatrix(filename,'Sheet','T623x');
[R6232x2,S623x2] = gx623x(ydata,xdata,2,xyTransf)
S623x2.bestmdl
if R6232x2 > zBest
    zBest = R6232x2;
    bestS = S623x2;
end

xyTransf = readmatrix(filename,'Sheet','T613');
[R6131,S6131] = gx613(ydata,xdata,1,xyTransf)
S6131.bestmdl
if R6131 > zBest
    zBest = R6131;
    bestS = S6131;
end

xyTransf = readmatrix(filename,'Sheet','T613');
[R6132,S6132] = gx613(ydata,xdata,2,xyTransf)
S6132.bestmdl
if R6132 > zBest
    zBest = R6132;
    bestS = S6132;
end

xyTransf = readmatrix(filename,'Sheet','T613x');
[R613x,S613x] = gx613x(ydata,xdata,xyTransf)
S613x.bestmdl

```



```

if R613x > zBest
    zBest = R613x;
    bestS = S613x;
end

fprintf("\n===== \n\n")
fprintf("----- Best Model ----- \n\n");
zBest
bestS
bestS.bestmdl

```

The code for the test program performs the following tasks:

1. Initialize the best adjusted R-square value stored in variable zBest.
2. Read the data matrix in sheet **Data** of files go66x.xlsx. This task stores the input matrix in variable xyData,
3. Copy the first column of matrix xyData into the column vector ydata.
4. Copy the rest of the columns in matrix xyData into the matrix xdata.
5. Read the transformation in sheet **T633** into variable xyTransf.
6. Call function gx633() with arguments ydata, xdata, and xyTransf. This task stores the results into variable R633 and structure S633 and displays them.
7. Display the best model stored in object S633.bestmdl.
8. If R633 is greater than variable zBest, update zBest with the value in R633 and copy structure S633 into bestS.
9. Repeat steps 5 to 8 by using the transformations in sheet **T623** and calling function gx623(). This set of tasks works with the results into variable R623 and structure S623.
10. Repeat steps 5 to 8 by using the transformations in sheet **T623x** and calling function gx623x(). This set of tasks works with the results into variable R623x1 and structure S623x1.
11. Repeat steps 5 to 8 by using the transformations in sheet **T623x** and calling function gx623x(). This set of tasks works with the results into variable R623x2 and structure S623x2.
12. Repeat steps 5 to 8 by using the transformations in sheet **T613** and calling function gx613(). This set of tasks works with the results into variable R613 and structure S613.

- 13.Repeat steps 5 to 8 by using the transformations in sheet **T613x** and calling function `gx613x()` This set of tasks works with the results into variable `R613x1` and structure `S613x1`.
- 14.Repeat steps 5 to 8 by using the transformations in sheet **T613x** and calling function `gx613x()` This set of tasks works with the results into variable `R613x2` and structure `S613x2`.
- 15.Display the very best model object stored in structure `bestS`.
- 16.Display the best regression results object stored in object `bestS.bestmdl`.

Open the Excel file `go66x.xlsx` before you run the MATLAB program. If needed, enter or edit the data in sheets **Data** and various transformation sheets. Once you are done you **must close the Excel file** before you run the MATLAB program. Examine the results in the MATLAB console. Here is the console output:

```
R633 =
    1

S633 =
    struct with fields:
        model: "(X1+X2+X1*X2) / (X1+X2+X1*X2) "
        bestAdjR2: 1
        bestYpwr: 1
        bestX1pwr: 1
        bestX2pwr: 1
        bestZX1pwr: 1
        bestZX2pwr: 1
        bestYX1pwr: 0
        bestYX2pwr: 0
        bestZYG1pwr: 0
        bestZYG2pwr: 0
        bestmdl: [1x1 LinearModel]
```

ans =

Linear regression model:

$$y \sim 1 + x_1 + x_2 + x_3 + x_4 + x_5 + x_6$$

Estimated Coefficients:

	Estimate	SE	tStat	pValue
(Intercept)	5	3.9933e-06	1.2521e+06	1.8542e-46
x1	1	2.9792e-06	3.3566e+05	6.9506e-42
x2	1	2.326e-06	4.2993e+05	9.5949e-43

x3	1	2.1689e-06	4.6107e+05	5.4842e-43
x4	1	3.1997e-06	3.1253e+05	1.2307e-41
x5	1	2.7356e-06	3.6556e+05	3.5122e-42
x6	1	1.9354e-06	5.167e+05	2.2046e-43

Number of observations: 15, Error degrees of freedom: 8

Root Mean Squared Error: 5.84e-07

R-squared: 1, Adjusted R-Squared: 1

F-statistic vs. constant model: 7.27e+28, p-value = 1.69e-114

R623 =

1.0000

S623 =

struct with fields:

```

    model: "(X1+X2)/(X1+X2+X1*X2)"
    bestAdjR2: 1.0000
    bestYpwr: 1
    bestX1pwr: 2
    bestX2pwr: 2
    bestYX1pwr: 0
    bestYX2pwr: 0
    bestZYX1pwr: 0
    bestZYX2pwr: 0
    bestmdl: [1x1 LinearModel]

```

ans =

Linear regression model:

$y \sim 1 + x1 + x2 + x3 + x4 + x5$

Estimated Coefficients:

	Estimate	SE	tStat	pValue
(Intercept)	2.0302	0.056175	36.14	4.7071e-11
x1	-0.0013667	0.00043624	-3.1328	0.012066
x2	-0.00066629	1.4863e-05	-44.828	6.839e-12
x3	-0.27109	0.0069621	-38.938	2.4153e-11
x4	-0.24938	0.00092837	-268.62	6.9893e-19
x5	0.067221	0.0017154	39.187	2.2814e-11

Number of observations: 15, Error degrees of freedom: 9

Root Mean Squared Error: 0.0211

R-squared: 1, Adjusted R-Squared: 1

F-statistic vs. constant model: 4.11e+05, p-value = 7.99e-24

R623x1 =

```

1.0000

S623x1 =

struct with fields:

    model: "X11+X1*X2) / (X1+X2+X1*X2) "
    bestAdjR2: 1.0000
    bestYpwr: 2
    bestX1pwr: 2
    bestZX1pwr: 1
    bestZX2pwr: 2
    bestYX1pwr: 0
    bestYX2pwr: 0
    bestZYX1pwr: 0
    bestZYX2pwr: 2
    bestmdl: [1×1 LinearModel]

ans =

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

Estimated Coefficients:

```

	Estimate	SE	tStat	pValue
(Intercept)	23.445	0.77536	30.238	2.3149e-10
x1	0.083897	0.0070123	11.964	7.8978e-07
x2	-0.0019221	0.00025615	-7.5037	3.6785e-05
x3	-0.099867	0.0017824	-56.03	9.2467e-13
x4	-0.24125	0.00069736	-345.95	7.172e-20
x5	3.0385e-05	7.3882e-07	41.126	1.4804e-11

```

Number of observations: 15, Error degrees of freedom: 9
Root Mean Squared Error: 0.872
R-squared: 1, Adjusted R-Squared: 1
F-statistic vs. constant model: 6.66e+05, p-value = 9.05e-25

R6232x2 =

1.0000

S623x2 =

struct with fields:

    model: "X12+X1*X2) / (X1+X2+X1*X2) "
    bestAdjR2: 1.0000
    bestYpwr: 1
    bestX1pwr: 2
    bestZX1pwr: 1

```

```

bestZX2pwr: 0
bestYX1pwr: 0
bestYX2pwr: 0
bestZYX1pwr: 0
bestZYX2pwr: 0
bestmdl: [1×1 LinearModel]

ans =

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

Estimated Coefficients:

```

	Estimate	SE	tStat	pValue
(Intercept)	2.2747	0.018139	125.41	6.6207e-16
x1	-0.00058089	1.3432e-05	-43.245	9.4382e-12
x2	0.10491	0.012809	8.1905	1.8333e-05
x3	-0.093523	0.019134	-4.8877	0.00086216
x4	-0.23941	0.0011238	-213.03	5.6306e-18
x5	0.03254	0.0036275	8.9703	8.7721e-06

```

Number of observations: 15, Error degrees of freedom: 9
Root Mean Squared Error: 0.0105
R-squared: 1, Adjusted R-Squared: 1
F-statistic vs. constant model: 1.66e+06, p-value = 1.49e-26

R6131 =

1.0000

S6131 =

struct with fields:

    model: "(X1)/(X1+X2+X1*X2)"
    bestAdjR2: 1.0000
    bestYpwr: 2
    bestZpwr: 2
    bestYX1pwr: 0
    bestYX2pwr: 0
    bestZYX1pwr: 0
    bestZYX2pwr: 2
    bestmdl: [1×1 LinearModel]

ans =

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

```

Estimated Coefficients:

	Estimate	SE	tStat	pValue
(Intercept)	21.013	1.8	11.674	3.784e-07
x1	0.11348	0.01482	7.6568	1.7251e-05
x2	-0.092653	0.0038355	-24.157	3.3641e-10
x3	-0.2382	0.0014471	-164.6	1.6828e-18
x4	3.171e-05	1.8333e-06	17.296	8.8331e-09

Number of observations: 15, Error degrees of freedom: 10

Root Mean Squared Error: 2.23

R-squared: 1, Adjusted R-Squared: 1

F-statistic vs. constant model: 1.28e+05, p-value = 1.73e-23

R6132 =

1.0000

S6132 =

struct with fields:

```

    model: "(X2)/(X1+X2+X1*X2)"
    bestAdjR2: 1.0000
    bestYpwr: 2
    bestZpwr: 2
    bestYX1pwr: 1
    bestYX2pwr: 0
    bestZYX1pwr: 1
    bestZYX2pwr: 2
    bestmdl: [1x1 LinearModel]
```

ans =

Linear regression model:

$y \sim 1 + x1 + x2 + x3 + x4$

Estimated Coefficients:

	Estimate	SE	tStat	pValue
(Intercept)	34.027	1.0092	33.715	1.2457e-11
x1	-0.0091391	0.0009061	-10.086	1.4692e-06
x2	-0.018957	0.00046469	-40.795	1.8755e-12
x3	-0.24701	0.0016161	-152.84	3.5309e-18
x4	7.108e-06	1.8023e-07	39.439	2.6249e-12

Number of observations: 15, Error degrees of freedom: 10

Root Mean Squared Error: 1.3

R-squared: 1, Adjusted R-Squared: 1

F-statistic vs. constant model: 3.72e+05, p-value = 8.19e-26

```

R613x =
    1.0000

S613x =
    struct with fields:

        model: "(X1*X2)/(X1+X2+X1*X2)"
    bestAdjR2: 1.0000
    bestYpwr: 2
    bestZX1pwr: 2
    bestZX2pwr: 1
    bestYX1pwr: 0
    bestYX2pwr: 0
    bestZYX1pwr: 0
    bestZYX2pwr: 2
    bestmdl: [1×1 LinearModel]

ans =

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

Estimated Coefficients:

```

	Estimate	SE	tStat	pValue
(Intercept)	26.428	0.97582	27.083	1.0894e-10
x1	0.016647	0.0010834	15.365	2.7738e-08
x2	-0.057585	0.0039587	-14.547	4.6943e-08
x3	-0.23664	0.00075931	-311.65	2.8462e-21
x4	2.8476e-05	1.0719e-06	26.566	1.3177e-10

```

Number of observations: 15, Error degrees of freedom: 10
Root Mean Squared Error: 1.18
R-squared: 1, Adjusted R-Squared: 1
F-statistic vs. constant model: 4.57e+05, p-value = 2.93e-26

=====

----- Best Model -----

zBest =
    1

bestS =
    struct with fields:

```

```

        model: " (X1+X2+X1*X2) / (X1+X2+X1*X2) "
    bestAdjR2: 1
    bestYpwr: 1
    bestX1pwr: 1
    bestX2pwr: 1
    bestZX1pwr: 1
    bestZX2pwr: 1
    bestYX1pwr: 0
    bestYX2pwr: 0
    bestZYX1pwr: 0
    bestZYX2pwr: 0
    bestmdl: [1×1 LinearModel]

ans =

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

Estimated Coefficients:

```

	Estimate	SE	tStat	pValue
(Intercept)	5	3.9933e-06	1.2521e+06	1.8542e-46
x1	1	2.9792e-06	3.3566e+05	6.9506e-42
x2	1	2.326e-06	4.2993e+05	9.5949e-43
x3	1	2.1689e-06	4.6107e+05	5.4842e-43
x4	1	3.1997e-06	3.1253e+05	1.2307e-41
x5	1	2.7356e-06	3.6556e+05	3.5122e-42
x6	1	1.9354e-06	5.167e+05	2.2046e-43

```

Number of observations: 15, Error degrees of freedom: 8
Root Mean Squared Error: 5.84e-07
R-squared: 1, Adjusted R-Squared: 1
F-statistic vs. constant model: 7.27e+28, p-value = 1.69e-114

```

The above output appears as follows using the format long MATLAB command:

```

R633 =

    1

S633 =

struct with fields:

    model: " (X1+X2+X1*X2) / (X1+X2+X1*X2) "
    bestAdjR2: 1
    bestYpwr: 1
    bestX1pwr: 1

```



```

    bestX2pwr: 1
    bestZX1pwr: 1
    bestZX2pwr: 1
    bestYX1pwr: 0
    bestYX2pwr: 0
    bestZYX1pwr: 0
    bestZYX2pwr: 0
    bestmdl: [1×1 LinearModel]

ans =

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

Estimated Coefficients:

```

	Estimate	SE	tStat	pValue
(Intercept)	4.99999999999986	3.99333896273797e-06	1252085.0462871	1.85416078847951e-46
x1	0.999999999999879	2.97920863059117e-06	335659.607632597	6.95064906897506e-42
x2	0.999999999999904	2.32596209576329e-06	429929.620014613	9.5948698380445e-43
x3	0.999999999999991	2.16888490302031e-06	461066.421093782	5.48415939305512e-43
x4	0.999999999999871	3.19974052552418e-06	312525.341359062	1.23065442284874e-41
x5	0.999999999999888	2.73555247792209e-06	365556.869433367	3.51219343681402e-42
x6	0.999999999999919	1.93537050228258e-06	516696.931580034	2.2046058868869e-43

```

Number of observations: 15, Error degrees of freedom: 8
Root Mean Squared Error: 5.84e-07
R-squared: 1, Adjusted R-Squared: 1
F-statistic vs. constant model: 7.27e+28, p-value = 1.69e-114

R623 =

    0.999993183578219

S623 =

struct with fields:

    model: "(X1+X2)/(X1+X2+X1*X2)"
    bestAdjR2: 0.999993183578219
    bestYpwr: 1
    bestX1pwr: 2
    bestX2pwr: 2
    bestYX1pwr: 0
    bestYX2pwr: 0
    bestZYX1pwr: 0
    bestZYX2pwr: 0
    bestmdl: [1×1 LinearModel]

ans =

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

```

## Estimated Coefficients:

	Estimate	SE	tStat	pValue
(Intercept)	2.03018367638321	0.0561751705059501	36.1402316734966	4.70707904217856e-11
x1	-0.00136668238244354	0.00043624316443798	-3.13284538040673	0.0120662644057679
x2	-0.000666287528364062	1.48632205608606e-05	-44.8279379045615	6.83899832547686e-12
x3	-0.271089645595376	0.00696205731644474	-38.938151938946	2.41533489168166e-11
x4	-0.249375729165003	0.000928365121025348	-268.61815843488	6.98925579676204e-19
x5	0.0672210808287063	0.00171539152790802	39.1870192519168	2.28143477482563e-11

Number of observations: 15, Error degrees of freedom: 9

Root Mean Squared Error: 0.0211

R-squared: 1, Adjusted R-Squared: 1

F-statistic vs. constant model: 4.11e+05, p-value = 7.99e-24

R623x1 =

0.999995798691946

S623x1 =

struct with fields:

```

    model: "X11+X1*X2) / (X1+X2+X1*X2) "
    bestAdjR2: 0.999995798691946
    bestYpwr: 2
    bestX1pwr: 2
    bestZX1pwr: 1
    bestZX2pwr: 2
    bestYX1pwr: 0
    bestYX2pwr: 0
    bestZYX1pwr: 0
    bestZYX2pwr: 2
    bestmdl: [1×1 LinearModel]
```

ans =

Linear regression model:

y ~ 1 + x1 + x2 + x3 + x4 + x5

## Estimated Coefficients:

	Estimate	SE	tStat	pValue
(Intercept)	23.4451198642229	0.775360876125459	30.2376874899596	2.31488598049099e-10
x1	0.0838971538932084	0.00701228843153382	11.9643044795374	7.89775114625668e-07
x2	-0.00192205398225558	0.000256148500401524	-7.50367064122051	3.67854947884378e-05
x3	-0.0998671694533684	0.00178238193158379	-56.0301738273503	9.24670382966644e-13
x4	-0.2412514641735	0.000697363952852633	-345.947712362587	7.17204593435981e-20
x5	3.03849708126189e-05	7.38824108143577e-07	41.1261225475795	1.48040913072391e-11

Number of observations: 15, Error degrees of freedom: 9

Root Mean Squared Error: 0.872

R-squared: 1, Adjusted R-Squared: 1

F-statistic vs. constant model: 6.66e+05, p-value = 9.05e-25

R6232x2 =

0.999998314371382

S623x2 =

struct with fields:

```

    model: "X12+X1*X2) / (X1+X2+X1*X2) "
    bestAdjR2: 0.999998314371382
    bestYpwr: 1
    bestX1pwr: 2
    bestZX1pwr: 1
    bestZX2pwr: 0
    bestYX1pwr: 0
    bestYX2pwr: 0
    bestZYX1pwr: 0
    bestZYX2pwr: 0
    bestmdl: [1×1 LinearModel]

```

ans =

Linear regression model:

$y \sim 1 + x1 + x2 + x3 + x4 + x5$

Estimated Coefficients:

	Estimate	SE	tStat	pValue
(Intercept)	2.27474904109574	0.0181386662259681	125.408837273774	6.62065796417472e-16
x1	-0.000580887925523303	1.34323909269447e-05	-43.2453111797149	9.43821279750848e-12
x2	0.104913541024265	0.0128092188339976	8.19047143966412	1.83325207958476e-05
x3	-0.0935229933721846	0.0191343647929303	-4.88769783498324	0.000862163083484542
x4	-0.239405584462527	0.00112381253686476	-213.029821797883	5.63060272162464e-18
x5	0.0325400859282759	0.00362754303112934	8.97028254359409	8.77214356312893e-06

Number of observations: 15, Error degrees of freedom: 9

Root Mean Squared Error: 0.0105

R-squared: 1, Adjusted R-Squared: 1

F-statistic vs. constant model: 1.66e+06, p-value = 1.49e-26

R6131 =

0.999972563327046

S6131 =

struct with fields:

```

    model: "(X1) / (X1+X2+X1*X2) "
    bestAdjR2: 0.999972563327046
    bestYpwr: 2
    bestZpwr: 2

```

```

bestYX1pwr: 0
bestYX2pwr: 0
bestZYX1pwr: 0
bestZYX2pwr: 2
bestmdl: [1×1 LinearModel]

ans =

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

Estimated Coefficients:

```

	Estimate	SE	tStat	pValue
(Intercept)	21.0131986055365	1.80002230400763	11.6738545732194	3.78395731877652e-07
x1	0.113476467247468	0.014820272681131	7.6568407133254	1.72511247710507e-05
x2	-0.0926533748903763	0.00383548991531333	-24.1568553004023	3.36412317486609e-10
x3	-0.238197500970936	0.00144711544904812	-164.601588026455	1.68277414449222e-18
x4	3.1709943583854e-05	1.83333930426234e-06	17.2962765321899	8.83309303612569e-09

```

Number of observations: 15, Error degrees of freedom: 10
Root Mean Squared Error: 2.23
R-squared: 1, Adjusted R-Squared: 1
F-statistic vs. constant model: 1.28e+05, p-value = 1.73e-23

R6132 =

0.999990598911919

S6132 =

struct with fields:

    model: "(X2)/(X1+X2+X1*X2)"
    bestAdjR2: 0.999990598911919
    bestYpwr: 2
    bestZpwr: 2
    bestYX1pwr: 1
    bestYX2pwr: 0
    bestZYX1pwr: 1
    bestZYX2pwr: 2
    bestmdl: [1×1 LinearModel]

ans =

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

Estimated Coefficients:

```

	Estimate	SE	tStat	pValue
(Intercept)	34.0266621090917	1.00924077109678	33.7151085088583	1.24571703857129e-11
x1	-0.0091391384452242	0.000906102440296805	-10.0862088421598	1.46920208430713e-06

```

x2      -0.0189571076956945    0.000464692867481076    -40.7949185845133    1.87548944606791e-12
x3      -0.247009022955963    0.00161613171421252    -152.839660767576    3.53092105335111e-18
x4      7.10796434991382e-06    1.80228110509648e-07    39.4387109192566    2.62492312216979e-12

```

Number of observations: 15, Error degrees of freedom: 10  
 Root Mean Squared Error: 1.3  
 R-squared: 1, Adjusted R-Squared: 1  
 F-statistic vs. constant model: 3.72e+05, p-value = 8.19e-26

R613x =

0.999992348848102

S613x =

struct with fields:

```

        model: "(X1*X2)/(X1+X2+X1*X2)"
    bestAdjR2: 0.999992348848102
    bestYpwr: 2
    bestZX1pwr: 2
    bestZX2pwr: 1
    bestYX1pwr: 0
    bestYX2pwr: 0
    bestZYX1pwr: 0
    bestZYX2pwr: 2
    bestmdl: [1×1 LinearModel]

```

ans =

Linear regression model:  
 $y \sim 1 + x1 + x2 + x3 + x4$

Estimated Coefficients:

	Estimate	SE	tStat	pValue
(Intercept)	26.4279918067684	0.975820092321988	27.0828526843327	1.08943097502639e-10
x1	0.0166469728665142	0.00108340997192297	15.3653494964303	2.77384351681382e-08
x2	-0.0575851875919186	0.00395868965508896	-14.5465274141639	4.69426810391666e-08
x3	-0.236635664409071	0.000759307113426336	-311.646842528927	2.84621915096054e-21
x4	2.84763890253295e-05	1.07191137711217e-06	26.5659919591932	1.31774574092032e-10

Number of observations: 15, Error degrees of freedom: 10  
 Root Mean Squared Error: 1.18  
 R-squared: 1, Adjusted R-Squared: 1  
 F-statistic vs. constant model: 4.57e+05, p-value = 2.93e-26

=====

----- Best Model -----

zBest =

```

1

bestS =

  struct with fields:

    model: "(X1+X2+X1*X2) / (X1+X2+X1*X2) "
    bestAdjR2: 1
    bestYpwr: 1
    bestX1pwr: 1
    bestX2pwr: 1
    bestZX1pwr: 1
    bestZX2pwr: 1
    bestYX1pwr: 0
    bestYX2pwr: 0
    bestZYX1pwr: 0
    bestZYX2pwr: 0
    bestmdl: [1×1 LinearModel]

ans =

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

Estimated Coefficients:

```

	Estimate	SE	tStat	pValue
(Intercept)	4.9999999999986	3.99333896273797e-06	1252085.0462871	1.85416078847951e-46
x1	0.99999999999879	2.97920863059117e-06	335659.607632597	6.95064906897506e-42
x2	0.99999999999904	2.32596209576329e-06	429929.620014613	9.5948698380445e-43
x3	0.99999999999991	2.16888490302031e-06	461066.421093782	5.48415939305512e-43
x4	0.99999999999871	3.19974052552418e-06	312525.341359062	1.23065442284874e-41
x5	0.99999999999888	2.73555247792209e-06	365556.869433367	3.51219343681402e-42
x6	0.99999999999919	1.93537050228258e-06	516696.931580034	2.2046058868869e-43

```

Number of observations: 15, Error degrees of freedom: 8
Root Mean Squared Error: 5.84e-07
R-squared: 1, Adjusted R-Squared: 1
F-statistic vs. constant model: 7.27e+28, p-value = 1.69e-114

```

The console output shows the regression table including the regression coefficients, their standard errors, the student-t values, and the p-Values. The latter values can shed valuable insight as to the eligibility of a variable/term to be part of the model. The best model is:

$$Y = (5 + X_1 + X_2 + X_3 * X_4) / (1 + \ln(X_1) + \ln(X_2) + \ln(X_3) * \ln(X_4))$$

The constant 1 in the denominator part is NOT calculated. It is an implicit part of the definition of the rational heteronomial. This fact is true for ALL rational heteronomials.

Using the format long allows us to see more digits in the adjusted R-square values. We can see clearer (than the first output) which models are superior. If the data includes errors, the result may favor a different model.

The MATLAB code contains the function `fx()` that transforms the value of a variable using a power value. The current implementation of function `fx()` handle negative, zero, and positive powers. Notice that the power zero causes function `fx()` to return the natural logarithm value of `x`.

Should you desire to use other functions for transformations, then you need to code function `fx()` to detect the supplied power values and use them to evaluate a special function. For example, if you are using the range of -3 to 3 in increments of 1 for regular powers, you can use the powers of 4 and 5 to evaluate, for example, the sine and cosine functions. The function `fx()` should use separate if statements to detect the value of 4 and 5 and return `sin(x)` and `cos(x)`, respectively. The code for function `fx()` would look like:

```
function y = fx(x,pwr):  
    if pwr == 4  
        y = sin(x)  
        return;  
    if pwr == 5  
        y = cos(x);  
        return;  
    if pwr > 0:  
        y = x.^pwr;  
    elif pwr < 0:  
        y = 1./x.^abs(pwr);  
    else:  
        y = log(x);  
    end
```

As shown above, function `fx()` should detect the special coded powers first.