HP 39gII Regression: Part III
The Best Curve Fits in Town!

Namir Shammas

Introduction

In the last two articles I presented various HP 39gII functions that performed regression calculations on different models. Using these functions assumed that you either knew the regression model suitable for your data or were willing to try a few different regression models to see which one best fit your data. This article looks at HP 39gII functions that systematically try a large number of regression models to see which ones are the best. In this article I discuss the following topics:

1. The best regression models for two variables, fitting up to 81 regression models.

2. The best regression models for three variables, fitting up to 729 regression models.

3. The best regression models for four variables fitting up to 6561 regression models.

4. The best polynomial for two variables.

☞ In this series of articles, I use the term *regression model* to mean the equation that is used in the regression calculations to describe the relationship between a dependent variable and one or more independent variables.

The Best Regression Model for Two Variables

Given two variables, x and y, what is the best linearized regression model when each variable has a list of powers used to transform it? Here is a partial list of linearized regression models that can be tested:

$y = a + b\ x$
$y = a + b/\ln(x)$
$\ln(y) = a + b\ x$
$\ln(y) = a + b\ \ln(x)$
$y = a + b/x$
$1/y = a + b\ x$
$1/y = a + b/x$
$\ln(y) = a + b/x$
$1/y = a + b\ \ln(x)$

And so on! I can further extend the list by adding more regression models that use combinations of square root, squared, cubed, reciprocal square root, reciprocal square, and reciprocal cube transformations for each of x and y. The total number of regression models equals the number of product of the transformation applied to x and y.

What is my end game here? Rather than selecting just the very best regression model (with the highest coefficient of determination), I want to apply a more informative approach. I want to find out the best N regression models. In this article I choose N to be 20. You can easily change that limit to 10, 15, 25, or any other number you want. In general it's a good idea to look at the top N regression model and not be fixated with just the very best regression model.

Table 1 shows you the source code for function **BestLR**. This function has the following parameters:

1. The parameter **Data** is the source data matrix that contains the variables x and y. The matrix must have at least two columns of data.

2. The parameter **lstSel** is a list containing two elements. The first element is the index of parameter **Data** that selects the variable x. The second list element is the index of parameter **Data** that selects the variable y.

3. The parameter **lstX** is a list that enumerates the set of powers used to transform the variable x. These powers can be integers and non-integers, and also positive, zero, and negative. The function treats the zero power as a special case and applies the natural logarithm. If you supply an empty list to this parameter, the function automatically uses the list {–3, –2, –1, –0.5,0,0.5,1,2,3}. This list allows the values of x to be transformed into reciprocal cube, reciprocal square, reciprocal, reciprocal square root, natural logarithm, square root, linear, square, and cube values. You can pass arguments for this parameter that are subsets of these values to choose fewer transformations. You can also pass arguments for this parameter that are supersets of these values to choose more transformations. You can also pick and choose any valid combination of powers. The key point in all these cases is to use valid powers that lead to error-free transformations. All of the transformation you choose must NOT GENERATE RUNTIME errors. Otherwise, the function will stop executing.

4. The parameter **lstY** is a list that enumerates the set of powers used to transform the variable y. It works just like parameter **lstX** but on the data for variable y.

The function uses the variable **MaxRes** to manage the number of best regression models to report back to you. The function assigns the value of 20 to this variable. You can change this value to alter the number of best regression models stored in the results matrix.

The function returns a results matrix containing the best 20 results. These results are sorted by the coefficient of determination of each regression model. The results matrix contains the following columns:

1. The values of the coefficient of determination.

2. The powers used to transform variable y.

3. The powers used to transform variable x.

4. The intercept values.

5. The slope values.

Statement
EXPORT BestLR(Data,lstSel,lstX,lstY)
BEGIN
** LOCAL Tx,Ty,i,j,k;**
** LOCAL SelXCol,SelYCol;**
** LOCAL PowerX,PowerY;**
** LOCAL MatX,VectY,RegCoeff,MatRes;**
** LOCAL lstDim,NumRows,ResUpdated;**
** LOCAL Sum1,Sum2,Rsqr,YMean,Yhat;**
** LOCAL MaxRes,NumColRes;**


```
  // use the default transformation list
  // for variable x if lstX is empty
  IF SIZE(lstX)==0 THEN
    lstX:={-3,-2,-1,-0.5,0,0.5,1,2,3};
  END;

  // use the default transformation list
  // for variable y if lstY is empty
  IF SIZE(lstY)==0 THEN
    lstY:={-3,-2,-1,-0.5,0,0.5,1,2,3};
  END;
  // set the maximum number of results
  MaxRes:=20;
  // set the number of columns in
  // the results matrix
  NumColRes:=5;
  // get the indices for variable x and y
  SelXCol:=lstSel(1);
  SelYCol:=lstSel(2);
  // get the number of rows of data
  lstDim:=SIZE(Data);
  NumRows:=lstDim(1);
  // create the result and regression matrices
  MatX:=MAKEMAT(1,NumRows,2);
  VectY:=MAKEMAT(1,NumRows,1);
  MatRes:=MAKEMAT(0,MaxRes,NumColRes);

  // iterate for each transformation in x
  FOR Tx FROM 1 TO SIZE(lstX) DO
    // get the current power of x
    PowerX:=lstX(Tx);

    // transform x
    IF PowerX==0 THEN
      FOR i FROM 1 TO NumRows DO
        MatX(i,2):=LN(Data(i,SelXCol));
      END;
    ELSE
      FOR i FROM 1 TO NumRows DO
        MatX(i,2):=Data(i,SelXCol)^PowerX;
      END;
    END;

    // iterate for each transformation in y
    FOR Ty FROM 1 TO SIZE(lstY) DO
      // get the current power of y
      PowerY:=lstY(Ty);
```



```
      // transform y
      IF PowerY==0 THEN
        FOR i FROM 1 TO NumRows DO
          VectY(i,1):=LN(Data(i,SelYCol));
        END;
      ELSE
        FOR i FROM 1 TO NumRows DO
          VectY(i,1):=Data(i,SelYCol)^PowerY;
        END;
      END;

      // calculate regression coefficients
      RegCoeff:=LSQ(MatX,VectY);

      // calculate ymean
      Sum1:= 0;
      FOR i FROM 1 TO NumRows DO
        Sum1:=Sum1+VectY(i,1);
      END;
      YMean:=Sum1/NumRows;

      // calculate coefficient of determination
      Sum1:=0;
      Sum2:=0;
      Yhat:=MatX*RegCoeff;
      FOR i FROM 1 TO NumRows DO
        Sum1:=Sum1+(Yhat(i,1)-YMean)^2;
        Sum2:=Sum2+(VectY(i,1)-YMean)^2;
      END;
      Rsqr:=Sum1/Sum2;

      // Rsqr is better than last entry
      // in the results matrix?
      IF Rsqr>MatRes(MaxRes,1) THEN
        ResUpdated:=0;
        // loop for each row in the
        // results matrix
        FOR i FROM 1 TO MaxRes DO
          // compare with other Rsqr values
          IF ResUpdated==0 AND Rsqr>MatRes(i,1) THEN
            // found better Rsqr for row i
            // push rows below?
            IF i<MaxRes THEN
              j:=MaxRes-1;
              REPEAT
              // FOR j FROM MaxRes-1 TO n STEP -1 DO
                FOR k FROM 1 TO NumColRes DO
```

```
                    MatRes(j+1,k):=MatRes(j,k);
                  END;
                  j:=j-1;
                //END;
                UNTIL j<i;
              END;
              // store new best results
              MatRes(i,1):=Rsqr;
              MatRes(i,2):=PowerY;
              MatRes(i,3):=PowerX;
              MatRes(i,4):=RegCoeff(1,1);
              MatRes(i,5):=RegCoeff(2,1);
              ResUpdated:=1;
            END;
          END;
        END;
      END; // FOR Ty
  END; // FOR Tx

  RETURN MatRes;

END;
```

Table 1 – The source code for function BestLR.

The source code for function **BestLR** uses two nested **FOR** loops to go through each transformation for the variables x and y. Inside the nested loops, the function calculates the regression coefficients and the coefficient of determination. The next phase compares the newly calculated coefficient of determination with similar values stored in the first column of the results matrix. If the newly calculated coefficient of determination is better than any value in the first column of the results matrix, the function inserts the data from the newly calculated regression into the results matrix. The function determines which row will store the new data. Then, the insertion process copies old data from the insertion row downward in the results matrix. The last row in the results matrix has its data overwritten by the ones in the row above it, or possibly by the newly calculated regression data.

Let's test the function **BestLR** using the data in Table 2. The values in the table come from the equation:

$y = 3 + 2\ x^2$

x	*y*
1	5
2	11
3	21
4	35
5	53
6	75
7	101
8	131
9	165

x	*y*
10	203

Table 2 – Sample data used to test the function BestLR.

Enter the values of Table 2 in the global matrix M1 and then execute the following command:

`BestLR(M1,{1,2},{},{})→M2`

The above call for function **BestLR** uses the default transformations for both variables x and y. Since the function uses 9 transformations (including the linear one) on each variable, the total number of regression models tested is 81. Figure 1 shows the contents of matrix M2 which stores the results of the best model selection. Table 3 shows the best five regression models. The best model in that table is indeed the one used to create the data in Table 2. Keep in mind that if the values of y included an error component, the function **BestLR** may not specify the equation $y = 3 + 2\ x^2$ as the best model. That model may be superseded by other models depending on the error components associated with the values of variable y.

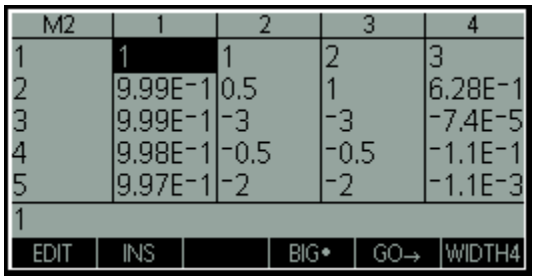

Fig. 1 – The values in matrix M2.

Rsqr	*Power y*	*Power x*	*Regression Model*	*Intercept*	*Slope*
1.000000	1	2	$y = a+b\ x^2$	3	2
0.999189	0.5	1	$\sqrt{y} = a+b\ x$	0.6279425	1.3509222
0.998873	-3	-3	$1/y^3 = a + b/x^3$	-7.441533E-5	8.0478988E-3
0.997834	-0.5	0.5	$1/\sqrt{y} = a+b/\sqrt{x}$	-0.1115322	0.56543287
0.997208	-2	-2	$1/y^2 = a + b/x^2$	-1.1075566E-3	0.0407642

Table 3 – The Best five regression models found by function BestLR.

The Best Regression Model for Three Variables

The last section gave you have a taste of finding the best regression model between one dependent variable and one independent variable. This section presents a function that finds the best regression model between the dependent variable, y, and the independent variables x and z.

Table 4 shows you the source code for function **BestMLR2**. This function has the following parameters:

1. The parameter **Data** is the source data matrix that contains the variables x, z, and y. The matrix must have at least two columns of data.

2. The parameter **lstSel** is a list containing three elements. The first element is the index of parameter **Data** that selects the independent variable x. The second list element is the index of parameter Data

that selects the independent variable z. The third list element is the index of parameter Data that selects the dependent variable y.

3. The parameter **lstX** is a list that enumerates the set of powers used to transform the variable x. These powers can be integers and non-integers, and also positive, zero, and negative. The function treats the zero power as a special case and applies the natural logarithm. If you supply an empty list to this parameter, the function automatically uses the list {–3, –2, –1, –0.5,0,0.5,1,2,3}. This list allows the values of x to be transformed into reciprocal cube, reciprocal square, reciprocal, reciprocal square root, natural logarithm, square root, linear, square, and cube values. You can pass arguments for this parameter that are subsets of these values to choose fewer transformations. You can also pass arguments for this parameter that are supersets of these values to choose more transformations. All of the transformation you choose must NOT GENERATE RUNTIME errors.

4. The parameter **lstZ** is a list that enumerates the set of powers used to transform the variable z. It works just like parameter **lstX** but on the data for variable z.

5. The parameter **lstY** is a list that enumerates the set of powers used to transform the variable y. It works just like parameter **lstX** but on the data for variable y.

The function uses the variable **MaxRes** to manage the number of best regression models to store and report back to you. The function assigns the value of 20 to this variable. You can change this value to alter the number of best regression models stored in the results matrix.

The function returns a results matrix containing the best 20 results. These results are sorted by the coefficient of determination of each regression model. The results matrix contains the following columns:

1. The values for the coefficient of determination.

2. The powers used to transform variable y.

3. The powers used to transform variable x.

4. The powers used to transform variable z.

5. The intercept values.

6. The values for the regression coefficient of variable x.

7. The values for the regression coefficient of variable z.

Statement
`EXPORT BestMLR2(Data,lstSel,lstX,lstZ,lstY)`
`BEGIN`
`  LOCAL Tx,Tz,Ty,i,j,k;`
`  LOCAL SelXCol,SelZCol,SelYCol;`
`  LOCAL PowerX,PowerZ,PowerY;`
`  LOCAL MatX,VectY,RegCoeff,MatRes;`
`  LOCAL lstDim,NumRows,ResUpdated;`
`  LOCAL Sum1,Sum2,Rsqr,YMean,Yhat;`
`  LOCAL MaxRes,NumColRes;`
`  // use default transformation list`
`  // if lstX is an empty list`


```
IF SIZE(lstX)==0 THEN
  lstX:={-3,-2,-1,-0.5,0,0.5,1,2,3};
END;

// use default transformation list
// if lstZ is an empty list
IF SIZE(lstZ)==0 THEN
  lstZ:={-3,-2,-1,-0.5,0,0.5,1,2,3};
END;

// use default transformation list
// if lstY is an empty list
IF SIZE(lstY)==0 THEN
  lstY:={-3,-2,-1,-0.5,0,0.5,1,2,3};
END;
// Set the number of rows and columns in the results matrix
MaxRes:=20;
NumColRes:=7;
// get the variable selectors
SelXCol:=lstSel(1);
SelZCol:=lstSel(2);
SelYCol:=lstSel(3);
// get the number of rows in matrix Data
lstDim:=SIZE(Data);
NumRows:=lstDim(1);
// create the regression and results matrices
MatX:=MAKEMAT(1,NumRows,3);
VectY:=MAKEMAT(1,NumRows,1);
MatRes:=MAKEMAT(0,MaxRes,NumColRes);

// start the calculations

// process transformations for variable x
FOR Tx FROM 1 TO SIZE(lstX) DO
 PowerX:=lstX(Tx);

 // transform x
 IF PowerX==0 THEN
   FOR i FROM 1 TO NumRows DO
     MatX(i,2):=LN(Data(i,SelXCol));
   END;
 ELSE
   FOR i FROM 1 TO NumRows DO
     MatX(i,2):=Data(i,SelXCol)^PowerX;
   END;
 END;

```

Statement
`// process transformations for variable z`
`FOR Tz FROM 1 TO SIZE(lstZ) DO`
`  PowerZ:=lstZ(Tz);`
`  // transform z`
`  IF PowerZ==0 THEN`
`    FOR i FROM 1 TO NumRows DO`
`      MatX(i,3):=LN(Data(i,SelZCol));`
`    END;`
`  ELSE`
`    FOR i FROM 1 TO NumRows DO`
`      MatX(i,3):=Data(i,SelZCol)^PowerZ;`
`    END;`
`  END;`
`  // process transformations for variable y`
`  FOR Ty FROM 1 TO SIZE(lstY) DO`
`    PowerY:=lstY(Ty);`
`    // transform y`
`    IF PowerY==0 THEN`
`      FOR i FROM 1 TO NumRows DO`
`        VectY(i,1):=LN(Data(i,SelYCol));`
`      END;`
`    ELSE`
`      FOR i FROM 1 TO NumRows DO`
`        VectY(i,1):=Data(i,SelYCol)^PowerY;`
`      END;`
`    END;`
`    // calculate regression coefficients`
`    RegCoeff:=LSQ(MatX,VectY);`
`    // calculate ymean`
`    Sum1:=0;`
`    FOR i FROM 1 TO NumRows DO`
`      Sum1:=Sum1+VectY(i,1);`
`    END;`
`    Ymean:=Sum1/NumRows;`
`    // calculate the coefficient of determination`
`    Sum1:=0;`
`    Sum2:=0;`
`    Yhat:=MatX*RegCoeff;`
`    FOR i FROM 1 TO NumRows DO`
`      Sum1:=Sum1+(Yhat(i,1)-Ymean)^2;`
`      Sum2:=Sum2+(VectY(i,1)-Ymean)^2;`

```
          END;
          Rsqr:=Sum1/Sum2;

          // Rsqr is better than last entry
          // in the results matrix?
          IF Rsqr>MatRes(MaxRes,1) THEN
            ResUpdated:=0;
            // check which row to insert better
            // regression results
            FOR i FROM 1 TO MaxRes DO
              // insert new results in row i?
              IF ResUpdated==0 AND Rsqr>MatRes(i,1) THEN
                // inserting inside the results matrix?
                IF i<MaxRes THEN
                  // downward copy rows from MaxRes-1 to i
                  j:=MaxRes-1;
                  REPEAT
                    FOR k FROM 1 TO NumColRes DO
                      MatRes(j+1,k):=MatRes(j,k);
                    END;
                    j:=j-1;
                  UNTIL j<i;
                END;
                // insert better results in row i
                MatRes(i,1):=Rsqr;
                MatRes(i,2):=PowerY;
                MatRes(i,3):=PowerX;
                MatRes(i,4):=PowerZ;
                FOR k FROM 1 TO 3 DO
                  MatRes(i,4+k):=RegCoeff(k,1);
                END;
                ResUpdated:=1;
              END;
            END;
          END;
        END; // FOR Ty
      END; // FOR Tz
    END; // FOR Tx

  RETURN MatRes;

END;
```

Statement

Table 4 – The source code for function BestMLR2.

The source code for function **BestMLR2** uses three nested **FOR** loops to go through each transformation for the variables x, z, and y. Inside the nested loops, the function calculates the regression coefficients and the coefficient of determination. The next phase compares the newly calculated coefficient of

determination with similar values stored in the first column of the results matrix. If the newly calculated coefficient of determination is better than any value in the first column of the results matrix, the function inserts the data from the newly calculated regression into the results matrix. The function determines which row will store the new data. Then, the insertion process copies old data from the insertion row downward in the results matrix. The last row in the results matrix has its data overwritten by the ones in the row above it, or possibly by the newly calculated regression data.

Let's test the function **BestMLR2** using the data in Table 5. The values in the table come from to the equation:

$y = 3 + 2\,x^2 + 20/z$

x	*z*	*y*
1	1	25
2	1	31
3	2	31
4	2	45
5	4	58
6	5	79
7	2	111
8	4	136
9	5	169
10	5	207

Table 5 – Sample data used to test the function BestMLR2.

Enter the values of Table 5 in the global matrix M1 and then execute the following command:

`BestMLR2(M1,{1,2,3},{},{},{})→M2`

The above call for function **BestMLR2** uses the default transformations for variables x, z, and y. Since the function uses 9 transformations (including the linear one) on each variable, the total number of regression models tested is 729. Figure 2 shows the contents of matrix M2 which stores the results of the best model selection. Table 6 shows the best five regression models. The best model is the same one used to create the data in Table 5.

Fig. 2 – The values in matrix M2.

Rsqr	*Power y*	*Power z*	*Power z*	*Intercept*	*Slope X*	*Slope Z*
1.000000	1	2	-1	3	2	20
0.999893	1	2	-0.5	-7.371050	2.00917	29.73563

Rsqr	*Power y*	*Power z*	*Power z*	*Intercept*	*Slope X*	*Slope Z*
0.999772	1	2	-2	8.661370	1.976927	14.78107
0.999542	1	2	0	21.432295	2.011851	-10.008505
0.999459	1	2	-3	10.607556	1.960595	12.795117

Table 6 – The Best five regression models found by function BestMLR2.

The appendix contains the listing for function **BestML3** which obtains the best regression models for the independent variables x, z, and t, and the dependent variable y. This function performs tasks that are similar to functions **BestLR** and **BetMLR2**.

The Best Polynomial Fit

If you want to fit pairs of (x, y) data points with a polynomial, one of the first and common questions you may ask regards the best order of the polynomial. This section deals with how to obtain the best polynomial order in fitting (x, y) data points. The first issue in dealing with fitting polynomials that have different orders is how to compare the goodness of fit for regression models that have a different number of terms. Up till now, the functions I presented used the coefficient of determination to compare models that have the same number of terms. To compare models that have different number of terms we need to calculate the *adjusted coefficient of determination*[2]. The following equation calculates this statistic based on the coefficient of determination:

$R^2_{adj} = 1 - (1 - R^2)\ (n - 1) / (n - k - 1)$

Where n is the number of data points and k is the number of independent variables that are in the regression model.

Using the adjusted coefficient of determination seems like a good idea at first. The reality is that high-order polynomials also generate high value for their adjusted coefficient of determination. Often, you get to a certain polynomial order where the gain in the adjusted coefficient of determination is small. You ask yourself if that small gain is justified!

What would be nice is find a different statistic that balances the following aspects of goodness of fit:

- Reward the regression models that generate smaller values for the sum of squared errors.
- Penalize the regression models for using more independent variables or terms.

The *Akaike information criterion*[1] (AIC) is one of the new goodness-of-fit statistics that follow the above two rules. The corrected AIC statistic, AIC_C, is a refined version that I use in this article. To calculate the AIC_C I use the following equation:

$$AIC_C = n\ ln\left(\sum_1^n(\hat{y}_i - y_i)^2/n\right) + 2k + (2\ k\ (k+1))/(n-k-1)$$

Where n is the number of observations, $\hat{y}_i$ is the projected value of y, y_i is the observed value of y (or its transformed value), and k is the polynomial order plus 1 (that is, the total number of regression coefficients, including the constant term). What kind of values for the AIC_C statistics are we looking for? The smallest value of AIC_C picks the best the regression model. The AIC_C statistic works better than the adjusted coefficient of determination with polynomials. The AIC_C is able to penalize higher order polynomials if they fail to significantly reduce the sum of the errors squared. The AIC_C does not justify a very small increase in the coefficient of determination in a higher order polynomial fit.

Table 7 presents the source code for the **BestPolyReg** function. This function has the following parameters:

- The parameter **Data** is the source data matrix that contains the values for variables x and y.
- The parameter **SelXCol** selects the column in matrix **Data** that contains the values for variable x.
- The parameter **SelYCol** selects the column in matrix **Data** that contains the values for variable y.
- The parameters **MinOrder** and **MaxOrder** define the range of polynomial orders to test.

The function returns the following list of results for the best polynomial fit:

- The polynomial order.
- The coefficient of determination.
- The adjusted coefficient of determination.
- The value for the AIC_C statistic.
- The vector column containing the regression coefficients.

During the calculation phase, the function also displays the values for the polynomial order, sum of squared errors, and AIC_C statistic for all the polynomial order. This output should give you an idea of how the different polynomial order compare with each other. When you press the [Home] button, the calculator switches to the Home display and shows you the list of final results.

Statment
`EXPORT BestPolyReg(Data,SelXCol,SelYCol,MinOrder,MaxOrder)`
`BEGIN`
`  LOCAL i,k,x,y,Order;`
`  LOCAL MatX,VectY;`
`  LOCAL lstDim,NumRows;`
`  LOCAL Sum1,Sum2,Sum3,AICc,BestAICc;`
`  LOCAL Rsqr,RsqrAdj,YMean,Yhat,RegCoeff;`
`  LOCAL BestOrder,BestRsqr,BestRsqrAdj,BestRegCoeff;`
`  lstDim:=SIZE(Data);`
`  NumRows:=lstDim(1);`
`  IF MinOrder<1 THEN`
`    MinOrder:=1;`
`  END;`
`  // initialize best regression data`
`  BestOrder=0;`
`  BestRsqr:=0;`
`  BestRsqrAdj=0;`
`  BestAICc:=1E499;`
`  // initialize vector y`
`  VectY:=MAKEMAT(1,NumRows,1);`
`  // calculate ymean … needs to be done once!`
`  Sum1:=0;`

Statment
`FOR i FROM 1 TO NumRows DO`
`  y:=Data(i,SelYCol);`
`  VectY(i,1):=y;`
`  Sum1:=Sum1+y;`
`END;`
`YMean:=Sum1/NumRows;`
`// calculate sum of y - ymean squared`
`Sum2:=0;`
`FOR i FROM 1 TO NumRows DO`
`  y:=Data(i,SelYCol);`
`  Sum2:=Sum2+(y-YMean)^2;`
`END;`
`// iterate for the specified range of polynomial orders`
`FOR Order FROM MinOrder TO MaxOrder DO`
`  // (re)create the matrix MatX`
`  MatX:=MAKEMAT(1,NumRows,1+Order);`
`  // fill the columns to to Order+1 with x^power values`
`  FOR i FROM 1 TO NumRows DO`
`    x:=Data(i,SelXCol);`
`    FOR k FROM 1 TO Order DO`
`      MatX(i,k+1):=x^k;`
`    END;`
`  END;`
`  // calculate regression coefficients`
`  RegCoeff:=LSQ(MatX,VectY);`
`  // calculate the coefficient of determination`
`  Sum1:=0;`
`  Sum3:=0;`
`  Yhat:=MatX*RegCoeff;`
`  FOR i FROM 1 TO NumRows DO`
`    Sum1:=Sum1+(Yhat(i,1)-YMean)^2;`
`    Sum3:=Sum3+(VectY(i,1)-Yhat(i,1))^2;`
`  END;`
`  Rsqr:=Sum1/Sum2;`
`  // calculate the adjusted coefficient of determination`
`  RsqrAdj:=1-(1-Rsqr)*(NumRows-1)/(NumRows-Order-1);`
`  k:=Order+1;`
`  // if Sum3 is 0 then adjust it to a small value`
`  // to avoid getting a LN(0) error`
`  IF Sum3==0 THEN`
`    Sum3:=1E-499;`

Statment
`END;`
`// calculate AICc statistic`
`AICc:=NumRows*LN(Sum3/NumRows)+2*k+(2*k*(k+1))/(NumRows-k-1);`
`// display intermediate results`
`PRINT("Order="+Order+", AICc="+AICc+", Sum3="+Sum3);`
`// found a better fit?`
`IF AICc<BestAICc THEN`
`// update best regression data`
`BestOrder:=Order;`
`BestRsqr:=Rsqr;`
`BestRsqrAdj:=RsqrAdj;`
`BestAICc:=AICc;`
`BestRegCoeff:=RegCoeff;`
`END;`
`END;`
`RETURN {BestOrder,BestRsqr,BestRsqrAdj,BestAICc,BestRegCoeff};`
`END;`

Table 7 – The source code for function BestPolyReg

Let's test the function **BestPolyReg**. Table 8 shows sample (x,y) data. Enter the data in the global matrix M2.

x	*y*
1	1
2	5
3	10
4	15
5	25
6	35
7	50
8	65
9	80
10	100

Table 8 – The sample data used to test function BestPolyReg.

Invoke the function **BestPolyReg** by using the following command:

```
BestPolyReg(M2,1,2,1,5)
```

The above command specifies matrix M2 as the data source matrix. The second and third arguments specify that variable x and y are in columns 1 and 2, respectively, of matrix M2. The last two arguments in the call to function **BestPolyReg** specify that the polynomial orders examined are in the range of 1 to 5.

Figures 3 and 4 show the intermediate output of the function using the **PRINT** statement. Figure 3 shows the upper portion of the **PRINT** statement output. Figure 4 shows the lower portion of the **PRINT** statement output. Notice that the polynomial order of 2 has the smallest AIC_C statistic. At the same time, the sum of errors squared keeps decreasing with the increasing polynomial order. Figure 4 also confirms

this trend. The same figure shows that the higher order polynomials failed to obtain the lowest AIC_C statistics.

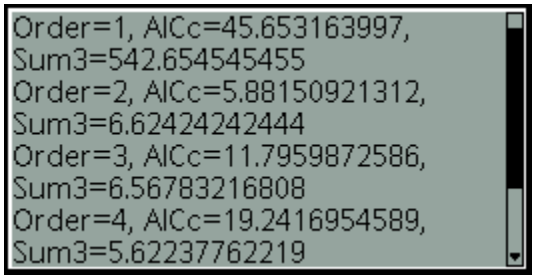

Fig. 3 – The upper portion of the PRINT statements output.

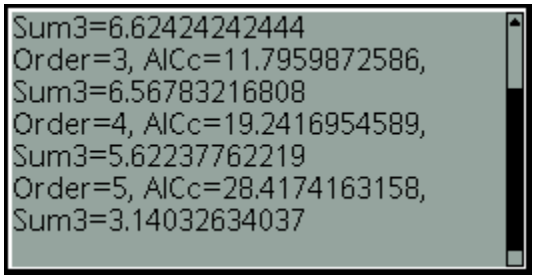

Fig. 4 – The lower portion of the PRINT statements output.

Figures 5 and 6 show the left and right sides of the list of output values. The output indicates that the quadratic polynomial is the best fit for the data in Table 8. The values for the coefficient of determination and adjusted coefficient of determination are 0.999364666385 and 0.999183142495, respectively. The AIC_C statistic for the best polynomial fit is 5.88. The best regression polynomial is:

$y = 0.56666666 - 0.137878789\ x + 1.007575758\ x^2$

Note that values of y in Table 5 are based on the quadratic polynomial $y = x^2$ with added errors. The function **BestPolyReg** has succeeded in identifying the quadratic polynomial as the one providing the best regression model.

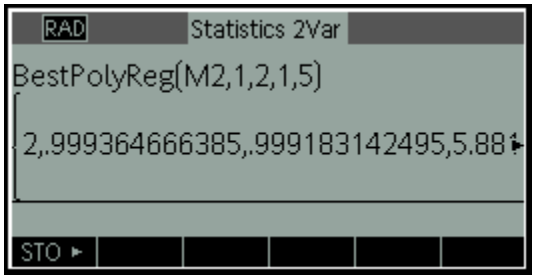

Fig. 5 – The left side of the output.

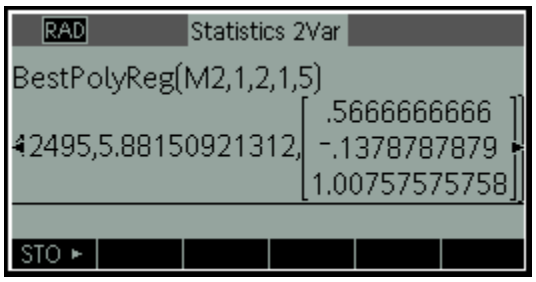

Fig. 6 – The right side of the output.

Observations and Conclusions

This article presented HP 39gII functions that perform bets linearized regression between two, three, and four variables. These functions go through a long list of linearized regression models and find the nest models, sorting the results using the values of the coefficient of determination. The article also offered a function that finds the best polynomial that fits (x, y) data points. The article demonstrated that the modified Akaike information criterion is very suitable to select the best polynomial.

The next article discusses least-squares relative error regression. The article will introduce you to the math behind relative error regression. Moreover, it will present several HP 39gII functions that apply calculations for this type of regression to general linear, polynomial, and linearized models.

Appendix

Here is the listing for function **BestML3**:

Statement
`EXPORT BestMLR3(Data,lstSel,lstX,lstZ,lstT,lstY)`
`BEGIN`
`  LOCAL Tx,Tz,Ty,Tt,i,j,k;`
`  LOCAL SelXCol,SelZCol,SelTCol,SelYCol;`
`  LOCAL PowerX,PowerZ,PowerT,PowerY;`
`  LOCAL MatX,VectY,RegCoeff,MatRes;`
`  LOCAL lstDim,NumRows,ResUpdated;`
`  LOCAL Sum1,Sum2,Rsqr,YMean,Yhat;`
`  LOCAL MaxRes,NumColRes;`
`  // use default transformation list`
`  // if lstX is an empty list`
`  IF SIZE(lstX)==0 THEN`
`    lstX:={-3,-2,-1,-0.5,0,0.5,1,2,3};`
`  END;`
`  // use default transformation list`
`  // if lstZ is an empty list`
`  IF SIZE(lstZ)==0 THEN`
`    lstZ:={-3,-2,-1,-0.5,0,0.5,1,2,3};`
`  END;`

Statement
`// use default transformation list`
`// if lstT is an empty list`
`IF SIZE(lstT)==0 THEN`
`  lstT:={-3,-2,-1,-0.5,0,0.5,1,2,3};`
`END;`
`// use default transformation list`
`// if lstY is an empty list`
`IF SIZE(lstY)==0 THEN`
`  lstY:={-3,-2,-1,-0.5,0,0.5,1,2,3};`
`END;`
`MaxRes:=20;`
`NumColRes:=9;`
`// get the variable selectors`
`SelXCol:=lstSel(1);`
`SelZCol:=lstSel(2);`
`SelTCol:=lstSel(3);`
`SelYCol:=lstSel(4);`
`// get the number of rows in matrix Data`
`lstDim:=SIZE(Data);`
`NumRows:=lstDim(1);`
`// create the regression and results matrices`
`MatX:=MAKEMAT(1,NumRows,3);`
`VectY:=MAKEMAT(1,NumRows,1);`
`MatRes:=MAKEMAT(0,MaxRes,NumColRes);`
`// start the calculations`
`// process transformations for variable x`
`FOR Tx FROM 1 TO SIZE(lstX) DO`
`  PowerX:=lstX(Tx);`
`  // transform x`
`  IF PowerX==0 THEN`
`    FOR i FROM 1 TO NumRows DO`
`      MatX(i,2):=LN(Data(i,SelXCol));`
`    END;`
`  ELSE`
`    FOR i FROM 1 TO NumRows DO`
`      MatX(i,2):=Data(i,SelXCol)^PowerX;`
`    END;`
`  END;`
`  // process transformations for variable z`
`  FOR Tz FROM 1 TO SIZE(lstZ) DO`
`   PowerZ:=lstZ(Tz);`

Statement
`// transform z`
`IF PowerZ==0 THEN`
`  FOR i FROM 1 TO NumRows DO`
`    MatX(i,3):=LN(Data(i,SelZCol));`
`  END;`
`ELSE`
`  FOR i FROM 1 TO NumRows DO`
`    MatX(i,3):=Data(i,SelZCol)^PowerZ;`
`  END;`
`END;`
`// process transformations for variable t`
`FOR Tt FROM 1 TO SIZE(lstT) DO`
`  PowerT:=lstT(Tt);`
`  // transform t`
`  IF PowerT==0 THEN`
`    FOR i FROM 1 TO NumRows DO`
`      MatX(i,4):=LN(Data(i,SelTCol));`
`    END;`
`  ELSE`
`    FOR i FROM 1 TO NumRows DO`
`      MatX(i,4):=Data(i,SelTCol)^PowerT;`
`    END;`
`  END;`
`  // process transformations for variable y`
`  FOR Ty FROM 1 TO SIZE(lstY) DO`
`    PowerY:=lstY(Ty);`
`    // transform y`
`    IF PowerY==0 THEN`
`      FOR i FROM 1 TO NumRows DO`
`        VectY(i,1):=LN(Data(i,SelYCol));`
`      END;`
`    ELSE`
`      FOR i FROM 1 TO NumRows DO`
`        VectY(i,1):=Data(i,SelYCol)^PowerY;`
`      END;`
`    END;`
`    // calculate regression coefficients`
`    RegCoeff:=LSQ(MatX,VectY);`
`    // calculate ymean`
`    Sum1:=0;`

Statement
FOR i FROM 1 TO NumRows DO
Sum1:=Sum1+VectY(i,1);
END;
YMean:=Sum1/NumRows;
// calculate the coefficient of determination
Sum1:=0;
Sum2:=0;
Yhat:=MatX*RegCoeff;
FOR i FROM 1 TO NumRows DO
Sum1:=Sum1+(Yhat(i,1)-YMean)^2;
Sum2:=Sum2+(VectY(i,1)-YMean)^2;
END;
Rsqr:=Sum1/Sum2;
// Rsqr is better than last entry
// in the results matrix?
IF Rsqr>MatRes(MaxRes,1) THEN
ResUpdated:=0;
// check which row to insert better
// regression results
FOR i FROM 1 TO MaxRes DO
// insert new results in row i?
IF ResUpdated==0 AND Rsqr>MatRes(i,1) THEN
// inserting inside the results matrix?
IF i<MaxRes THEN
// downward copy rows from MaxRes-1 to i
j:=MaxRes-1;
REPEAT
FOR k FROM 1 TO NumColRes DO
MatRes(j+1,k):=MatRes(j,k);
END;
j:=j-1;
UNTIL j<i;
END;
// insert better results in row i
MatRes(i,1):=Rsqr;
MatRes(i,2):=PowerY;
MatRes(i,3):=PowerX;
MatRes(i,4):=PowerZ;
MatRes(i,5):=PowerT;
FOR k FROM 1 TO 4 DO
MatRes(i,5+k):=RegCoeff(k,1);
END;
ResUpdated:=1;
END;
END;

Statement
`END;`
`END; // FOR Ty`
`END; // FOR Tt`
`END; // FOR Tz`
`END; // FOR Tx`
`RETURN MatRes;`
`END;`

References

1. Wikipedia article *Akaike information criterion*.
2. Wikipedia article *Coefficient of Determination*.
3. Wikipedia article *Linear Regression*.
4. Draper and Smith, *Applied Regression Analysis*, Wiley-Interscience; 3rd edition (April 23, 1998)
5. Neter, Kuther, Wasserman, and Nachtsheim, *Applied Linear Statistical Models*, McGraw-Hill/Irwin; 4th edition (February 1, 1996).
6. Fox, *Applied Regression Analysis and Generalized Linear Models*, Sage Publications, Inc; 2nd edition (April 16, 2008).
7. Montgomery, Peck, and Vining, *Introduction to Linear Regression Analysis*, Wiley-Interscience; 4th edition (2006).
8. Seber and Lee, *Linear Regression Analysis*, Wiley; 2nd edition (February 5, 2003).

