UNIVERSIDAD AUTÓNOMA METROPOLITANA
Casa abierta al tiempo

CBI DIVISIÓN DE CIENCIAS BÁSICAS E INGENIERÍA
UAM - Azcapotzalco

MÉTODOS NUMÉRICOS EN INGENIERÍA

Lineal	10
Parabolico	10
Cubico	10
Exponencial	10
Potencios	10
Crecimiento	10
Gráficas	10
Problema	10

Edif. 10

PROFESOR:

JOSE Luis Pantoja Gallegos

TRIMESTRE:

19-I

EQUIPO:

Panteras Negras

INTEGRANTES:

Martínez Magaña Áyax
Sánchez Castrejón Dulce Esperanza
Villegas Carrasco Maetzin

TAREA 3.
CURVAS DE APROXIMACIÓN

```
30
0.54  0.0498
0.61  0.0635
0.65  0.0652
0.73  0.0788
0.78  0.0788
0.81  0.064
0.86  0.0788
0.875 0.0681
0.89  0.0703
0.95  0.0703
1.02  0.0681
1.03  0.0681
1.055 0.079
1.135 0.0575
1.14  0.0681
1.20  0.0633
1.245 0.0575
1.28  0.0550
1.32  0.0511
1.385 0.0575
1.43  0.049
1.445 0.0532
1.535 0.0511
1.57  0.049
1.63  0.0532
1.70  0.0443
1.755 0.0426
1.81  0.0347
1.88  0.0336
2.01  0.025

PRONÓSTICOS:
X internas: 0.7 1.1 1.6
X externa inferior: 0.25
X externa superior: 2.5
```

```
/* EQUIPO: PANTERAS NEGRAS
   INTEGRANTES: Ayax Martinez Magaña
                Maetzin Villegas Carrasco
                Dulce Esperanza Sanches Castrejón

Fecha de creación: JULIO 2019

El proposito de este programa es dado un conjunto de puntos encontrar una función
que mejor se ajuste a dichos puntos con los modelos:
                        -Lineal
                        -Parabólico
                        -Cúbico
                        -Exponencial
                        -Potencia
                        -Crecimiento

----------------------DICCIONARIO DE DATOS-------------------------

*****VARIABLES DE ENTRADA*****

npunt : tipo entero. Almacena la cantidad de parejas ordenadas que se le
        dan al programa para llevar acabo su ajuste.

xdat : tipo apuntador flotante de doble precisión. Se almacenan los valores
       de x de las parejas ordenadas que se le dan al programa para llevar
       a cabo su ajuste.

ydat : tipo apuntador flotante de doble precisión. Se almacenan los valores
       de y de las parejas ordenadas que se le dan al programa para llevar
       a cabo su ajuste.

Xints : tipo apuntador flotante de doble precisión. Almacena los valores para
        los pronósticos que se encuentran dentro del rango de valores de los
        puntos xdat.

Xextinf : tipo flotante de doble precisión. Almacena el valor para el pronóstico
          que se encuentra por debajo del rango de valores de los puntos xdat.

Xextsup : tipo flotante de doble precisión. Almacena el valor para el pronóstico
          que se encuentra por encima del rango de valores de los puntos xdat.

Yints : tipo apuntador flotante de doble precisión. Almacena los valores de los
        pronósticos encontrados al evaluar los Xints en el modelo.

Yextinf : tipo flotante de doble precisión. Almacena el valor del pronóstico
          encontrado al evaluar Xextinf en el modelo.

Yextsup : tipo flotante de doble precisión. Almacena el valor del pronóstico
          encontrado al evaluar Xextsup en el modelo.


*****VARIABLES DE SALIDA*****

Sumx : tipo apuntador flotante de doble precisión. Sumx[i]  representa la suma
       de las x  a la i-ésima potencia.

x : tipo apuntador a apuntador de doble precision. Es una matriz donde cada fila
    tendrá npunt valores que indica la cantidad de puntos. El primer indice indica
    la potencia de x; el segundo, el punto de los npunts que se elevará a dicha
    potencia, e.g., x[2][i] significa que el punto i se elevará al cuadrado.

xNy : tipo apuntador flotante de doble precisión. Aqui se guardaran los valores de
      xi*N*yi para el i-ésimo punto, con N = 1,2,3, i.e., xy,x2y,x3y

SumxNy: tipo flotante de doble precision. Aqui se guardaran las sumas de las xNy,
        con N = 1,2,3, i.e., Sumxy, Sumx2y, Sumx3y.
```

```
67
68  Sumy : tipo flotante de doble precision. Almacena las sumas de los puntos y.
69
70  ymed : tipo flotante de doble precisión. Almacena el valor de la media aritmetica
71         de los puntos y.
72
73  a : tipo apuntador flotante de doble precisión. Se usa para almacenar los valores
74      de a0, a1, a2 y a3, dependiendo del modelo que se use.
75
76  Sr : tipo flotante de doble precision. Almacena las sumas de las diferencias al
77       cuadrado de los puntos originales y los puntos encontrados con el modelo.
78
79  St : tipo flotante de doble precision. Almacena las sumas de las diferencias al
80       cuadrado de los puntos originales y la media aritmetica, i.e., ymed.
81
82  YY : tipo apuntador flotante de doble precisión. Guarda los valores calculados de
83       y del modelo encontrado para cada punto i de los datos originales.
84
85  deltaN : tipo apuntador flotante de doble precisión. Vectores donde
86           se guardaran las diferencias(N=1,i.e., delta) y diferencias
87           al cuadrado (N=2,i.e., delta2) de los puntos originales y
88           los puntos encontrados con el modelo.
89
90  betaN : tipo apuntador flotante de doble precisión. Vectores donde se guardaran las
91          diferencias(N=1,i.e., beta) y diferencias al cuadrado(N=2,i.e., beta2) de
92          los puntos originales y la media aritmetica, i.e. ymed.
93
94  stdDev : tipo flotante de doble precision. Almacena la desviación estandar.
95
96  desEst : tipo flotante de doble precision. Almacena la desviación del estimado.
97
98  coefdet: tipo flotante de doble precision. Almacena el coeficiente de determinación
99
100 coefcor : tipo flotante de doble precision. Almacena el coeficiente de correlación.
101
102
103 *****VARIABLES INTERMEDIAS*****
104
105 A: tipo apuntador a apuntador de doble precision. Almacena los coeficientes y
106    valores independientes del sistema de ecns que se le pasará al método gauss
107    para calcular los valores de a0,a1 y en su caso a2, en el modelo dador por
108    y = a0 + a1x + a2x^2 + a3x^3 donde los valores de a2 y a3 pueden ser 0
109    dependiendo del tipo de modelo.
110
111 X: tipo apuntador de doble precisión. Almacena copias de los valores de 'x'
112    de las parejas ordenadas que se le dan a la función Lineal para llevar a
113    cabo su ajuste.
114
115 Y: tipo apuntador de doble precisión. Almacena copias de los valores de 'y'
116    de las parejas ordenadas que se le dan a la función Lineal para llevar a
117    cabo su ajuste.
118
119 Z: tipo apuntador de doble precisión. Almacena los valores linealizados de las
120    variables 'y' de entrada que se usan antes de pasarse a la funcion Lineal.
121
122 W: tipo apuntador de doble precisión. Almacena los valores linealizados de las
123    variables 'x' de entrada que se usan antes de pasarse a la funcion Lineal.
124
125 i: tipo entero. Indicador del numero de fila o columna de un arreglo o matriz
126    y variable para llevar la cuenta del i-ésimo punto.
127
128 j: tipo entero. Indicador del numero de fila o columna de un arreglo o matriz
129    y variable para llevar la cuenta del i-ésimo punto.
130
131 modType: tipo char. Variable que se pasa a la función lineal que indica el
132          tipo de modelo que se va a encontrar; dónde s - lineal estándar,
```

```
133           e - exponencial, p - potencias y c - crecimiento.
134
135   fptr1: tipo apuntador a estructura de datos de tipo archivo. Hace referencia
136          a los datos de entrada.
137
138   fptr2: tipo apuntador a estructura de datos de tipo archivo. Hace referencia
139          a los datos de salida.
140
141   */
142   #include <stdio.h>
143   #include <stdlib.h>
144   #include <math.h>
145   #include "Gauss.h"
146
147   //PROTOTIPOS DE FUNCIONES
148
149   //Función que realiza el ajuste lineal
150   double * Lineal(int npunt, double * X, double *Y, char modType);
151
152   //Función que realiza el ajuste parabólico
153   void Parabolico(int npunt, double * xdat, double *ydat);
154
155   //Función que realiza el ajuste cúbico
156   void Cubico(int npunt, double * xdat, double *ydat);
157
158   //Función que realiza el ajuste con modelo de potencias
159   void Potencias(int npunt, double * xdat, double *ydat);
160
161   //Función que realiza el ajuste exponencial
162   void Exponencial(int npunt, double * xdat, double *ydat);
163
164   //Función que realiza el ajuste con modelo de crecimiento
165   void Crecimiento(int npunt, double * xdat, double *ydat);
166
167   //Declaración de variables globales
168   FILE * fptr1, *fptr2;
169   double Xextinf, Xextsup, Yextinf, Yextsup;
170   double stdDev, desEst, coefdet, coefcor;
171   double *Xints,*Yints, * xdat, * ydat;
172   int npunt;
173
174   int main(){
175       Xints = (double *) malloc(sizeof(double)*3);
176       Yints = (double *) malloc(sizeof(double)*3);
177       fptr1 = fopen("Entrada.dat","r+");
178       fptr2 = fopen("Resultados.dat","w+");
179       fscanf(fptr1, "%d",&npunt);
180       xdat = (double *) malloc(sizeof(double)*(npunt+1));
181       ydat = (double *) malloc(sizeof(double)*npunt+1);
182       fscanf(fptr1, "%*c");
183
184       //Lectura de las coordenadas de entrada
185       for (int i=1; i<=npunt; i++)
186           fscanf(fptr1, "%lf%lf%*c", &xdat[i], &ydat[i]);
187
188       //Lectura de los pronósticos que están dentro del rango de valores
189       //de los x de muestreo
190       fscanf(fptr1,"%*c%*c%*[^\n]%*c%*[^0-9]");
191       for(int i=0;i<=2;i++)
192           fscanf(fptr1,"%lf", &Xints[i]);
193
194       //Lectura de los pronósticos que están fuera del rango de valores
195       //de los x de muestreo
196       fscanf(fptr1,"%*c%*[^0-9]%lf", &Xextinf);
197       fscanf(fptr1,"%*c%*[^0-9]%lf", &Xextsup);
198
```

```
199       //Llamada a funciones que realizan ajustes
200       Lineal(npunt,xdat,ydat,'c');
201       Exponencial(npunt,xdat,ydat);
202       Parabolico(npunt,xdat,ydat);
203       Potencias(npunt,xdat,ydat);
204       Crecimiento(npunt,xdat,ydat);
205       Cubico(npunt,xdat,ydat);
206   }
207
208   double * Lineal(int npunt, double * X, double *Y, char modType){
209       //Matriz que se le pasará al método gauss para
210       //calcular los valores de a0 y a1 de la ecn lineal
211       //que buscamos.
212       double ** A;
213       //filas (Solo se usaran 1 y 2)
214       A = (double **) malloc ((3) *sizeof(double));
215       for (int i=1; i<=2; i++)
216       //columnas (Solo se usaran 1,2 y 3)
217       A[i] = (double *) malloc ((4)*sizeof(double));
218       //Se declara un elemento más de los necesarios por conveniencia
219       //de manejo de indices. NO se usará Sumx[0]. Tambien se inicializan a 0.
220       double * Sumx = (double *) malloc(sizeof(double) * 3);
221       Sumx[1] = Sumx[2] = 0.;
222       //Se declaran 3 filas: x[0], x[1] y x[2]. Por practicidad,
223       //se usaran SOLO la 1 y 2.
224       double ** x = (double **) malloc(sizeof(double)*3);
225       x[1] = (double *) malloc(sizeof(double) * (npunt+1));
226       x[2] = (double *) malloc(sizeof(double) * (npunt+1));
227       double *xy = (double *) malloc(sizeof(double) * (npunt+1));
228       double Sumxy = 0., Sumy = 0., Sumydat = 0.0, Sumxdat = 0.0;
229       //Se realizan los cálculos de las potencias de x y sus sumas totales,
230       //de las xy[i] y sus sumas totales y de la suma total de las y
231       for (int i=1; i<=npunt; i++){
232           x[1][i] = X[i];
233           x[2][i] = pow(X[i],2);
234           xy[i] = X[i] * Y[i];
235           Sumx[1] += x[1][i];
236           Sumx[2] += x[2][i];
237           Sumydat += ydat[i];
238           Sumxdat += xdat[i];
239           Sumy += Y[i];
240           Sumxy += xy[i];
241       }
242       double ymed;
243       ymed = Sumydat / (double)npunt;
244       //Se copian los elementos a la matriz A para calcular a0 y a1
245       A[1][1] = npunt;
246       A[1][2] = A[2][1] = Sumx[1];
247       A[2][2] = Sumx[2];
248       A[1][3] = Sumy;
249       A[2][3] = Sumxy;
250       //aquí se guardaran los valores de a0 y a1
251       double *a = (double *) malloc(3*sizeof(double));
252       //Se pasa la matriz de coeficientes y su tamño al método
253       //Gauss para que calcule los valores de a0 y a1 y almacene
254       //en a, tal que a[1] = a0, a[2] = a1
255       a = Gauss(2,A);
256       double * YY = (double *) malloc(sizeof(double)*(npunt+1));
257       double Sr=0., St=0.;
258       double * ZZ = (double *) malloc(sizeof(double)*(npunt+1));
259       double * delta = (double *) malloc(sizeof(double) * (npunt+1));
260       double * delta2 = (double *) malloc(sizeof(double) * (npunt+1));
261       double * beta = (double *) malloc(sizeof(double) * (npunt+1));
262       double * beta2 = (double *) malloc(sizeof(double) * (npunt+1));
263       //Se hacen los cálculos para encontrar Sr y St
264       for (int i=1;i<=npunt; i++) {
```

```
265         ZZ[i] = a[1] + a[2]* x[1][i];
266         switch (modType){
267         case 's': YY[i] = ZZ[i]; break;
268         case 'e': YY[i] = exp(ZZ[i]); break;
269         case 'p': YY[i] = pow(10,ZZ[i]); break;
270         case 'c': YY[i] = 1. / ZZ[i]; break;
271         }
272         delta[i] = ydat[i] - YY[i];
273         delta2[i] = delta[i]*delta[i];
274         beta[i] = ydat[i] - ymed;
275         beta2[i] = beta[i] * beta[i];
276         Sr += delta2[i];
277         St += beta2[i];
278         //Dependiendo del modelo, se imprimen los resultados apropiados
279         switch (modType){
280                 case 's':
281                 if (i==1){
282                     for (int i=1;i<=84;i++){
283                         if(i!=42 )
284                             fprintf(fptr2,"-");
285                         else
286                             fprintf(fptr2,"MODELO LINEAL");
287                         }
288                 fprintf(fptr2,"\n\n");
289                 //Se escriben los encabezados de las columnas a imprimir
290                 fprintf(fptr2,"%6s%7s%8s%12s%9s%10s%11s%13s%10s%13s",
291                 "Punto ","X","Y","X²","XY","ZZ","Delta",
292                 "Delta²","Beta","Beta²\n");
293                 //Imprimir linea divisoria
294                 for(int i=1; i<=96;i++)
295                     fprintf(fptr2,"_");
296                 }
297                 fprintf(fptr2,"\n%-6d| %4.3lf| %5.4lf| %7.6lf|"
298                 " %7.6lf| %7.6lf| %9.6lf| %9.6lf| %9.6lf| %9.6lf|"
299                 , i, X[i],Y[i],x[2][i],xy[i],ZZ[i],
300                 delta[i],delta2[i],beta[i],beta2[i]);
301                 if (i==npunt){
302                 //Imprimir linea divisoria
303                 fprintf(fptr2,"\n");
304                 for (int i=1;i<=96;i++)
305                     fprintf(fptr2,"?");
306                 //Impresión de sumas totales, ymedia, coeficientes del modelo,
307                 //y el modelo encontrado
308                 fprintf(fptr2,"\n%s|%6.2lf| %6.4lf| %8.5lf| %8.6lf|"
309                   " %8s|  %10s %6.6lf| %11s %7.6lf|","Sumas:",Sumx[1],
310                   Sumy,Sumx[2],Sumxy,"","Sr = ",Sr ,"St = ",St);
311                 fprintf(fptr2, "\n\nYmedia = %lf\n\na0 = %.4lf, a1 = %.4lf"
312                     "\n\n******MODELO LINEAL******\nY = %.4lf %+.4lfx \n\n",
313                     ymed,a[1],a[2],a[1],a[2]);
314                 }
315                 break; //FIN CASE 's' (MODELO LINEAL)
316
317                 case 'e':
318                 if (i==1){
319                     for (int i=1;i<=103;i++){
320                         if(i!=51 )
321                             fprintf(fptr2,"-");
322                         else
323                             fprintf(fptr2,"MODELO EXPONENCIAL");
324                         }
325                     fprintf(fptr2,"\n\n");
326                     //Se escriben los encabezados de las columnas a imprimir
327                     fprintf(fptr2,"%6s%7s%8s%12s%11s%11s%11s%10s%11s%13s%10s%14s",
328                     "Punto ","X","Y","Z = ln(Y)","X²","XZ","ZZ","YY","Delta",
329                     "Delta²","Beta","Beta²\n");
330                     //Imprimir linea divisoria
```

```
331                 for(int i=1; i<=120;i++)
332                 fprintf(fptr2,"_");
333             }
334             fprintf(fptr2,"\n%-6d| %4.3lf| %5.4lf| %7.6lf| %7.6lf|  %7.6lf|"
335                 " %7.6lf| %7.6lf| %9.6lf| %9.6lf| %9.6lf| %9.6lf|",
336                 i, X[i],ydat[i],Y[i],x[2][i],xy[i],ZZ[i],YY[i], delta[i],
337                 delta2[i],beta[i],beta2[i]);
338             if (i==npunt){
339                 //Imprimir linea divisoria
340                 fprintf(fptr2,"\n");
341                 for (int i=1;i<=120;i++)
342                     fprintf(fptr2,"?");
343                 //Impresión de sumas totales, ymedia, coeficientes del modelo,
344                 //y el modelo encontrado
345                 fprintf(fptr2,"\n%s|%6.2lf| %6.4lf| %6.5lf| %8.5lf| %8.6lf|"
346                 " %8s|  %10s %6.6lf| %11s %7.6lf|","Sumas:",Sumx[1],
347                 Sumydat,Sumy,Sumx[2],Sumxy,"","Sr = ",Sr ,"St = ",St);
348                 fprintf(fptr2, "\n\nYmedia = %lf\n\na0 = %.4lf, a1 = %.4lf"
349                 "\n\n******MODELO LINEALIZADO******\nZ = %.4lf %+.4lfx \n\n",
350                 ymed,a[1],a[2],a[1],a[2]);
351             }
352             break; //FIN CASE 'e' (MODELO EXPONENCIAL)
353
354             case 'p':
355             if (i==1){
356                 for (int i=1;i<=113;i++){
357                     if(i!=56 )
358                         fprintf(fptr2,"-");
359                     else
360                         fprintf(fptr2,"MODELO DE POTENCIAS");
361                     }
362                 fprintf(fptr2,"\n\n");
363                 //Se escriben los encabezados de las columnas a imprimir
364                 fprintf(fptr2,"%6s%7s%8s%12s%11s%11s%10s%11s"
365                 "%11s%11s%13s%10s%14s","Punto ","X","Y","W = log(X)",
366                 "Z = log(Y)","W²","WZ","ZZ","YY","Delta",
367                 "Delta²","Beta","Beta²\n");
368                 //Imprimir linea divisoria
369                 for(int i=1; i<=131;i++)
370                 fprintf(fptr2,"_");
371             }
372             fprintf(fptr2,"\n%-6d| %4.3lf| %5.4lf| %9.6lf| %6.6lf| %6.6lf|"
373                 " %9.6lf| %7.6lf| %9.6lf| %9.6lf| %9.6lf| %9.6lf| %9.6lf|",
374                 i, xdat[i],ydat[i],X[i],Y[i],x[2][i],xy[i],ZZ[i],YY[i],
375                  delta[i],delta2[i],beta[i],beta2[i]);
376             if (i==npunt){
377                 //Imprimir linea divisoria
378                 fprintf(fptr2,"\n");
379                 for (int i=1;i<=131;i++)
380                     fprintf(fptr2,"?");
381                 //Impresión de sumas totales, ymedia, coeficientes del modelo,
382                 //y el modelo encontrado
383                 fprintf(fptr2,"\n%s|%6.2lf| %6.4lf| %6.5lf| %8.5lf|"
384                 " %8.5lf| %9.6lf|%21s|  %10s %6.6lf| %11s %7.6lf|",
385                 "Sumas:",Sumxdat, Sumydat,Sumx[1],Sumy, Sumx[2],
386                 Sumxy, "","Sr = ",Sr ,"St = ",St);
387                 fprintf(fptr2, "\n\nYmedia = %lf\n\na0 = %.4lf, a1 = %.4lf"
388                 "\n\n******MODELO LINEALIZADO******\nZ = %.4lf %+.4lfx \n\n",
389                 ymed,a[1],a[2],a[1],a[2]);
390             }
391             break;//FIN CASE 'p' (MODELO DE POTENCIAS)
392
393             case 'c':
394              if (i==1){
395                 for (int i=1;i<=111;i++){
396                     if(i!=56 )
```

```
                        fprintf(fptr2,"-");
                    else
                        fprintf(fptr2,"MODELO DE CRECIMIENTO");
                    }
                fprintf(fptr2,"\n\n");
                //Se escriben los encabezados de las columnas a imprimir
                fprintf(fptr2,"%6s%7s%8s%12s%11s%11s%10s%11s"
                "%11s%11s%13s%10s%14s",
                "Punto ","X","Y","V = 1/X","Z = 1/Y","V²",
                "VZ","ZZ","YY","Delta",
                "Delta²","Beta","Beta²\n");
                //Imprimir linea divisoria
                for(int i=1; i<=131;i++)
                fprintf(fptr2,"_");
            }
            fprintf(fptr2,"\n%-6d| %4.3lf| %5.4lf| %9.6lf| %6.6lf| %6.6lf|"
                " %9.6lf| %10.6lf| %9.6lf| %9.6lf| %9.6lf| %9.6lf|",
                i, xdat[i],ydat[i],X[i],Y[i],x[2][i],xy[i],ZZ[i],YY[i],
                delta[i],delta2[i],beta[i],beta2[i]);
            if (i==npunt){
                //Imprimir linea divisoria
                fprintf(fptr2,"\n");
                for (int i=1;i<=131;i++)
                    fprintf(fptr2,"?");
                //Impresión de sumas totales, ymedia, coeficientes del modelo,
                //y el modelo encontrado
                fprintf(fptr2,"\n%s|%6.2lf| %6.4lf| %6.5lf| %8.5lf| %8.5lf|"
                " %9.6lf|%21s|  %10s %6.6lf| %11s %7.6lf|","Sumas:",Sumxdat,
                Sumydat,Sumx[1],Sumy, Sumx[2],Sumxy, "","Sr = ",Sr ,"St = ",St);
                fprintf(fptr2, "\n\nYmedia = %lf\n\na0 = %.4lf, a1 = %.4lf"
                "\n\n******MODELO LINEALIZADO******\nZ = %.4lf %+.4lfx \n\n",
                ymed,a[1],a[2],a[1],a[2]);
            }
            break;//FIN CASE 'c' (MODELO DE CRECIMIENTO)
        } //FIN SWITCH
    }
    //Cálculo e impresión de desviaciones del estimado y estándar
    stdDev = sqrt(St/(npunt-1));
    desEst = sqrt(Sr/(npunt-2));
    fprintf(fptr2, "Desviación del estimado = %.6lf, "
                   "Desviación estándar = %.6lf\n\n", desEst, stdDev);
    //Cálculo e impresión de coeficientes de determinación y correlación
    coefdet= (St-Sr)/St;
    coefcor= sqrt(coefdet);
    fprintf(fptr2, "Coeficiente de Determinación = %.6lf, "
                   "Coeficiente de Correlación = %.6lf\n\n", coefdet, coefcor);

    //Transformación de modelo linealizado a modelo exponencial,
    //de potencias o de crecimiento; segun sea el caso.
    //Cálculo e impresion de los pronósticos
    switch(modType){

    case 's': //MODELO LINEAL
    fprintf(fptr2,"PRONÓSTICOS:\n");
    for(int i=0; i<=2; i++)
    {
        Yints[i]  = a[1] + a[2]*Xints[i];
        fprintf(fptr2,"Xint[%d] = %.3lf, Yint[%d] = %.4lf\n",
                i+1, Xints[i], Yints[i], i+1 );
    }
    Yextinf  = a[1] + a[2]*Xextinf;
    Yextsup  = a[1] + a[2]*Xextsup;
    fprintf(fptr2,"Xextinf = %.3lf, Yextinf = %.4lf\n",Xextinf, Yextinf);
    fprintf(fptr2,"Xextsup = %.3lf, Yextsup = %.4lf\n\n",Xextsup, Yextsup);
    break;//FIN CASE 's' (MODELO LINEAL)
```

```
463     case 'e': //MODELO EXPONENCIAL
464     //DADA LA ECN: Y = Ae^(Bx), se calculan los valores de A y B
465     //para obtener el modelo exponencial y se guardan en el vector a
466     //tal que a[1] = A y a[2] = B
467     fprintf(fptr2,"\nY = Ae^(Bx)\nTenemos que: \na0 = ln(A) = %.6lf ? "
468                   "A = e^a0 = e^(%.6lf) = %.6lf\na1 = B = %.6lf\n\n"
469                   ,a[1],a[1],exp(a[1]),a[2]);
470     a[1] = exp(a[1]); //SE ALMACENA EL COEFICIENTE A EN a[1]
471     a[2];// EL COEFICIENTE B ESTÁ EN a[2]
472     fprintf(fptr2,"*****MODELO EXPONENCIAL*****\nY = "
473                   "%+.6lfe^%.6lfx",a[1],a[2]);
474     fprintf(fptr2,"\n\nPRONÓSTICOS:\n");
475     for(int i=0; i<=2; i++)
476     {
477         Yints[i]  = a[1]*exp(a[2]*Xints[i]);
478         fprintf(fptr2,"Xint[%d] = %.3lf, Yint[%d] = %.4lf\n",
479                 i+1, Xints[i], Yints[i], i+1 );
480     }
481     Yextinf  = a[1]*exp(a[2]*Xextinf);
482     Yextsup  = a[1]*exp(a[2]*Xextsup);
483     fprintf(fptr2,"Xextinf = %.3lf, Yextinf = %.4lf\n",Xextinf, Yextinf);
484     fprintf(fptr2,"Xextsup = %.3lf, Yextsup = %.4lf\n\n",Xextsup, Yextsup);
485     break;//FIN CASE 'e' (MODELO EXPONENCIAL)
486
487     case 'p':
488     //MODELO DE POTENCIAS
489     //DADA LA ECN: Y = Ax^B, se calculan los valores de A y B
490     //para obtener el modelo exponencial y se guardan en el vector a
491     //tal que a[1] = A y a[2] = B
492     fprintf(fptr2,"\nY = Ax^B\nTenemos que: \na0 = log(A) = %.6lf ? "
493                   "A = 10^a0 = 10^(%.6lf) = %.6lf\na1 = B = %.6lf\n\n"
494                   ,a[1],a[1],pow(10,a[1]),a[2]);
495     a[1] = pow(10,a[1]); //SE ALMACENA EL COEFICIENTE A EN a[1]
496     a[2];// EL COEFICIENTE B ESTÁ EN a[2]
497     fprintf(fptr2,"*****MODELO DE POTENCIAS*****\nY = "
498                   "%+.6lfx^%.6lf",a[1],a[2]);
499     fprintf(fptr2,"\n\nPRONÓSTICOS:\n");
500     for(int i=0; i<=2; i++)
501     {
502         Yints[i]  = a[1]*pow(Xints[i],a[2]);
503         fprintf(fptr2,"Xint[%d] = %.4lf, Yint[%d] = %.4lf\n",
504                 i+1, Xints[i], Yints[i], i+1 );
505     }
506     Yextinf  = a[1]*pow(Xextinf,a[2]);
507     Yextsup  = a[1]*pow(Xextsup,a[2]);
508     fprintf(fptr2,"Xextinf = %.4lf, Yextinf = %.4lf\n",Xextinf, Yextinf);
509     fprintf(fptr2,"Xextsup = %.4lf, Yextsup = %.4lf\n\n",Xextsup, Yextsup);
510     break;//FIN CASE 'p' (MODELO DE POTENCIAS)
511
512     case 'c': //MODELO DE CRECIMIENTO
513     //DADA LA ECN: Y = Ax/(B+x), se calculan los valores de A y B
514     //para obtener el modelo exponencial y se guardan en el vector a
515     //tal que a[1] = A y a[2] = B
516     fprintf(fptr2,"\nY = Ax/(B+x)\nTenemos que: \na0 = 1/A = %.6lf ? "
517                   "A = 1/a0 = 1/%.6lf = %.6lf\na1 = B/A ?  B = a1*A = "
518                   "a1*%.6lf = %.6lf \n\n",a[1],a[1],1./a[1],
519                   1./a[1],a[2]*1./a[1]);
520     a[1] = 1/a[1]; //SE ALMACENA EL COEFICIENTE A EN a[1]
521     a[2]= a[2]*a[1];//SE ALMACENA EL COEFICIENTE B EN a[2]
522     fprintf(fptr2,"*****MODELO DE CRECIMIENTO*****\nY = "
523                   "%+.6lfx / (%.6lf+x)",a[1],a[2]);
524     fprintf(fptr2,"\n\nPRONÓSTICOS:\n");
525     for(int i=0; i<=2; i++)
526     {
527         Yints[i]  = a[1]*Xints[i]/(a[2]+Xints[i]);
528         fprintf(fptr2,"Xint[%d] = %.4lf, Yint[%d] = %.4lf\n",
```

```
529                 i+1, Xints[i], Yints[i], i+1 );
530     }
531     Yextinf  = a[1]*Xextinf/(a[2]+Xextinf);
532     Yextsup  = a[1]*Xextsup/(a[2]+Xextsup);
533     fprintf(fptr2,"Xextinf = %.4lf, Yextinf = %.4lf\n",Xextinf, Yextinf);
534     fprintf(fptr2,"Xextsup = %.4lf, Yextsup = %.4lf\n\n",Xextsup, Yextsup);
535     break;//FIN CASE 'c' (MODELO DE CRECIMIENTO)
536
537     }//FIN SWITCH
538
539     return a;
540 }
541
542 void Exponencial(int npunt, double * xdat, double *ydat){
543         double * Z = (double *) malloc(sizeof(double)*(npunt+1));
544         for(int i=1; i<=npunt; i++){
545             Z[i] = log(ydat[i]);
546         }
547         Lineal(npunt,xdat,Z,'e');
548 }
549
550 void Potencias(int npunt, double * xdat, double *ydat){
551         double * Z = (double *) malloc(sizeof(double)*(npunt+1));
552         double * W = (double *) malloc(sizeof(double)*(npunt+1));
553         for(int i=1; i<=npunt; i++){
554             W[i] = log10(xdat[i]);
555             Z[i] = log10(ydat[i]);
556         }
557         Lineal(npunt,W,Z,'p');
558 }
559
560 void Crecimiento(int npunt, double * xdat, double *ydat){
561         double * V = (double *) malloc(sizeof(double)*(npunt+1));
562         double * Z = (double *) malloc(sizeof(double)*(npunt+1));
563         for(int i=1; i<=npunt; i++){
564             V[i] = 1./(xdat[i]);
565             Z[i] = 1./(ydat[i]);
566         }
567         Lineal(npunt,V,Z,'c');
568 }
569
570 void Parabolico(int npunt, double * xdat, double *ydat){
571     fprintf(fptr2,"\n");
572     for (int i=1;i<=111;i++){
573     if(i!=55 )
574     fprintf(fptr2,"-");
575     else
576     fprintf(fptr2,"MODELO PARABÓLICO");
577     }
578     fprintf(fptr2,"\n\n");
579     //Matriz que se le pasará al método gauss para
580     //calcular los valores de a0, a1 y a2 de la ecn
581     //que buscamos.
582     double ** A;
583     //filas (Solo se usaran 1,2 y 3 )
584     A = (double **) malloc ((4) *sizeof(double));
585     for (int i=1; i<=3; i++)
586     //columnas (Solo se usaran 1,2,3 y 4)
587     A[i] = (double *) malloc ((5)*sizeof(double));
588
589     //Se declara un elemento más de los necesarios por conveniencia
590     //de manejo de índices. NO se usará Sumx[0]. Tambien se inicializan a 0.
591     double * Sumx = (double *) malloc(sizeof(double) * 5);
592     for (int i=1; i<=4;i++)
593     Sumx[i] = 0.;
594
```

```
595     //Se declaran 5 filas: de la x[0] a la x[4]. Por practicidad,
596     //se usaran SOLO la 1,2,3 y 4.
597     double ** x = (double **) malloc(sizeof(double)*5);
598     for(int i=1; i<=4; i++)
599     x[i] = (double *) malloc(sizeof(double) * (npunt+1));
600
601     double *xy = (double *) malloc(sizeof(double) * (npunt+1));
602     double *x2y = (double *) malloc(sizeof(double) * (npunt+1));
603     double Sumxy = 0., Sumx2y = 0., Sumy = 0.;
604
605     //Se realizan los cálculos de las potencias de x y sus sumas totales,
606     //de las xy[i] y x2y[i] y sus sumas totales y de la suma total de las y
607     for (int i=1; i<=npunt; i++){
608         xy[i] = xdat[i] * ydat[i];
609         x2y[i] = xdat[i] * xdat[i] * ydat[i];
610         Sumy += ydat[i];
611         Sumxy += xy[i];
612         Sumx2y += x2y[i];
613     }
614     for(int j=1; j<=4;j++){
615         for (int i=1; i<=npunt; i++){
616         x[j][i] = pow(xdat[i],j);
617         Sumx[j] += x[j][i];
618         }
619     }
620     double ymed;
621     ymed = (double) Sumy / npunt;
622     //Se copian los elementos a la matriz A para calcular a0, a1 y a2
623     A[1][1] = npunt;
624     A[1][2] = A[2][1] = Sumx[1];
625     A[1][3] = A[3][1] = A[2][2] = Sumx[2];
626     A[3][2] = A[2][3] = Sumx[3];
627     A[3][3] = Sumx[4];
628     A[1][4] = Sumy;
629     A[2][4] = Sumxy;
630     A[3][4] = Sumx2y;
631     //aquí se guardarán los valores de a0, a1 y a2
632     double *a = (double *) malloc(4*sizeof(double));
633     //Se pasa la matriz de coeficientes y su tamño al método
634     //Gauss para que calcule los valores de a0, a1 y a2 y almacene
635     //en a, tal que a[1] = a0, a[2] = a1 y a[3] = a2
636     a = Gauss(3,A);
637     double Sr=0., St=0.;
638     double * YY = (double *) malloc(sizeof(double)*(npunt+1));
639     double * delta = (double *) malloc(sizeof(double) * (npunt+1));
640     double * delta2 = (double *) malloc(sizeof(double) * (npunt+1));
641     double * beta = (double *) malloc(sizeof(double) * (npunt+1));
642     double * beta2 = (double *) malloc(sizeof(double) * (npunt+1));
643     //Se escriben los encabezados de las columnas a imprimir
644     fprintf(fptr2,"%6s%7s%8s%12s%11s%13s%9s%11s%10s%11s%13s%10s%13s",
645             "Punto ","X","Y","X²","X³","X?","XY","X²Y","YY","Delta",
646             "Delta²","Beta","Beta²\n");
647     for(int i=1; i<=127;i++)
648         fprintf(fptr2,"_");
649     //Se hacen los cálculos para encontrar Sr y St
650     for (int i=1;i<=npunt; i++){
651         YY[i] = a[1] + a[2]* x[1][i] + a[3]*x[2][i];
652         delta[i] = ydat[i] - YY[i];
653         delta2[i] = delta[i]*delta[i];
654         beta[i] = ydat[i] - ymed;
655         beta2[i] = beta[i] * beta[i];
656         Sr += delta2[i];
657         St += beta2[i];
658         fprintf(fptr2,"\n%-6d| %4.3lf| %5.4lf| %7.6lf| %7.6lf| %9.6lf| "
659                 "%7.6lf| %7.6lf| %7.6lf| %9.6lf| %9.6lf| %9.6lf| %9.6lf|",
660                 i,xdat[i],ydat[i],x[2][i],x[3][i],x[4][i],xy[i],
```

```
661                 x2y[i], YY[i],delta[i],delta2[i],beta[i],beta2[i]);
662     }
663     fprintf(fptr2,"\n");
664     //Imprimir linea divisoria
665     for (int i=1;i<=127;i++)
666     fprintf(fptr2,"?");
667
668     //Impresión de sumas totales, ymedia, coeficientes del modelo,
669     //y el modelo encontrado
670     fprintf(fptr2,"\n%s|%6.2lf| %6.4lf| %8.5lf| %8.5lf| %9.4lf| %8.6lf|"
671                   " %8.4lf| %8s|  %10s %6.6lf| %11s %7.6lf|","Sumas:",Sumx[1],
672                   Sumy,Sumx[2],Sumx[3],Sumx[4],Sumxy,Sumx2y,
673                   "","Sr = ",Sr ,"St = ",St);
674     fprintf(fptr2, "\n\nYmedia = %lf\n\na0 = %.4lf, a1 = %.4lf, a2 = %.4lf"
675                    "\n\n******MODELO PARABÓLICO******\nY ="
676                    " %.4lf %+.4lfx %+.4lfx^2 \n\n",
677                    ymed,a[1],a[2],a[3],a[1],a[2],a[3]);
678
679     //Cálculo e impresión de desviaciones del estimado y estándar
680     stdDev = sqrt(St/(npunt-1));
681     desEst = sqrt(Sr/(npunt-2));
682     fprintf(fptr2, "Desviación del estimado = %.6lf, "
683                    "Desviación estándar = %.6lf\n\n", desEst, stdDev);
684     //Cálculo e impresión de coeficientes de determinación y correlación
685     coefdet= (St-Sr)/St;
686     coefcor= sqrt(coefdet);
687     fprintf(fptr2, "Coeficiente de Determinación = %.6lf, "
688                    "Coeficiente de Correlación = %.6lf\n\n", coefdet, coefcor);
689     //Cálculo e impresión de los pronósticos
690     fprintf(fptr2,"PRONÓSTICOS:\n");
691     for(int i=0; i<=2; i++)
692     {
693         Yints[i]  = a[1] + a[2]*Xints[i] + a[3]*pow(Xints[i],2);
694         fprintf(fptr2,"Xint[%d] = %.3lf, Yint[%d] = %.4lf\n",
695                 i+1, Xints[i], Yints[i], i+1 );
696     }
697     Yextinf  = a[1] + a[2]*Xextinf + a[3]*pow(Xextinf,2);
698     Yextsup  = a[1] + a[2]*Xextsup + a[3]*pow(Xextsup,2);
699     fprintf(fptr2,"Xextinf = %.3lf, Yextinf = %.4lf\n",Xextinf, Yextinf);
700     fprintf(fptr2,"Xextsup = %.3lf, Yextsup = %.4lf\n\n",Xextsup, Yextsup);
701 }
702
703 void Cubico(int npunt, double * xdat, double *ydat){
704     for (int i=1;i<=142;i++){
705     if(i!=71 )
706     fprintf(fptr2,"-");
707     else
708     fprintf(fptr2,"MODELO CÚBICO");
709     }
710     fprintf(fptr2,"\n\n");
711     //Matriz que se le pasará al método gauss para
712     //calcular los valores de a0, a1, a2 y  a3 de la ecn
713     //que buscamos.
714     double ** A;
715     //filas (Solo se usaran de la 1 a la 4)
716     A = (double **) malloc ((5) *sizeof(double));
717     for (int i=1; i<=4; i++)
718     //columnas (Solo se usaran de la 1 a la 5)
719     A[i] = (double *) malloc ((6)*sizeof(double));
720
721     //Se declara un elemento más de los necesarios por conveniencia
722     //de manejo de índices. NO se usará Sumx[0]. Tambien se inicializan a 0.
723     double * Sumx = (double *) malloc(sizeof(double) * 7);
724     for (int i=1; i<=6;i++)
725     Sumx[i] = 0.;
726
```

```
727     //Se declaran 7 filas: de la x[0] a la x[6]. Por practicidad,
728     //se usaran SOLO de la 1 a las 6.
729     double ** x = (double **) malloc(sizeof(double)*7);
730     for(int i=1; i<=6; i++)
731     x[i] = (double *) malloc(sizeof(double) * (npunt+1));
732
733     double *xy = (double *) malloc(sizeof(double) * (npunt+1));
734     double *x2y = (double *) malloc(sizeof(double) * (npunt+1));
735     double *x3y = (double *) malloc(sizeof(double) * (npunt+1));
736     double Sumxy = 0., Sumx2y = 0.,Sumx3y = 0., Sumy = 0.;
737
738     //Se realizan los cálculos de las potencias de x y sus sumas totales,
739     //de las xy[i] y x2y[i] y sus sumas totales y de la suma total de las y
740     for (int i=1; i<=npunt; i++){
741         xy[i] = xdat[i] * ydat[i];
742         x2y[i] = xdat[i] * xdat[i] * ydat[i];
743         x3y[i] = xdat[i] * xdat[i] * xdat[i] * ydat[i];
744         Sumy += ydat[i];
745         Sumxy += xy[i];
746         Sumx2y += x2y[i];
747         Sumx3y += x3y[i];
748     }
749     for(int j=1; j<=6;j++){
750         for (int i=1; i<=npunt; i++){
751         x[j][i] = pow(xdat[i],j);
752         Sumx[j] += x[j][i];
753         }
754     }
755     double ymed;
756     ymed = (double) Sumy / npunt;
757
758     //Se copian los elementos a la matriz A para calcular a0, a1, a2 y a3
759     A[1][1] = npunt;
760     A[1][2] = A[2][1] = Sumx[1];
761     A[1][3] = A[3][1] = A[2][2] = Sumx[2];
762     A[1][4] = A[4][1] = A[3][2] = A[2][3] = Sumx[3];
763     A[2][4] = A[4][2] = A[3][3] = Sumx[4];
764     A[3][4] = A[4][3] = Sumx[5];
765     A[4][4] = Sumx[6];
766     A[1][5] = Sumy;
767     A[2][5] = Sumxy;
768     A[3][5] = Sumx2y;
769     A[4][5] = Sumx3y;
770     //aqui se guardaran los valores de a0, a1, a2 y a3
771     double *a = (double *) malloc(5*sizeof(double));
772     //Se pasa la matriz de coeficientes y su tamño al método
773     //Gauss para que calcule los valores de a0, a1, a2 y a3, y almacene
774     //en a, tal que a[1] = a0, a[2] = a1, a[3] = a2, a[4] = a3.
775     a = Gauss(4,A);
776     //printf("\na0 = %lf\na1 = %lf\na2=%lf\na3=%lf",a[1],a[2],a[3],a[4]);
777
778     double Sr=0., St=0.;
779     double * YY = (double *) malloc(sizeof(double)*(npunt+1));
780     double * delta = (double *) malloc(sizeof(double) * (npunt+1));
781     double * delta2 = (double *) malloc(sizeof(double) * (npunt+1));
782     double * beta = (double *) malloc(sizeof(double) * (npunt+1));
783     double * beta2 = (double *) malloc(sizeof(double) * (npunt+1));
784
785     //Se escriben los encabezados de las columnas a imprimir
786     fprintf(fptr2,"%6s %5s %7s %10s %10s %12s %12s %12s %9s"
787             "%10s %10s %9s %9s %10s %9s %11s \n",
788             "Punto ","X","Y","X²","X³","X?","X?","X?","XY",
789             "X²Y","X³Y","YY","Delta","Delta²","Beta","Beta²");
790     for(int i=1; i<=154;i++)
791         fprintf(fptr2,"_");
792     //Se hacen los cálculos para encontrar Sr y St
```

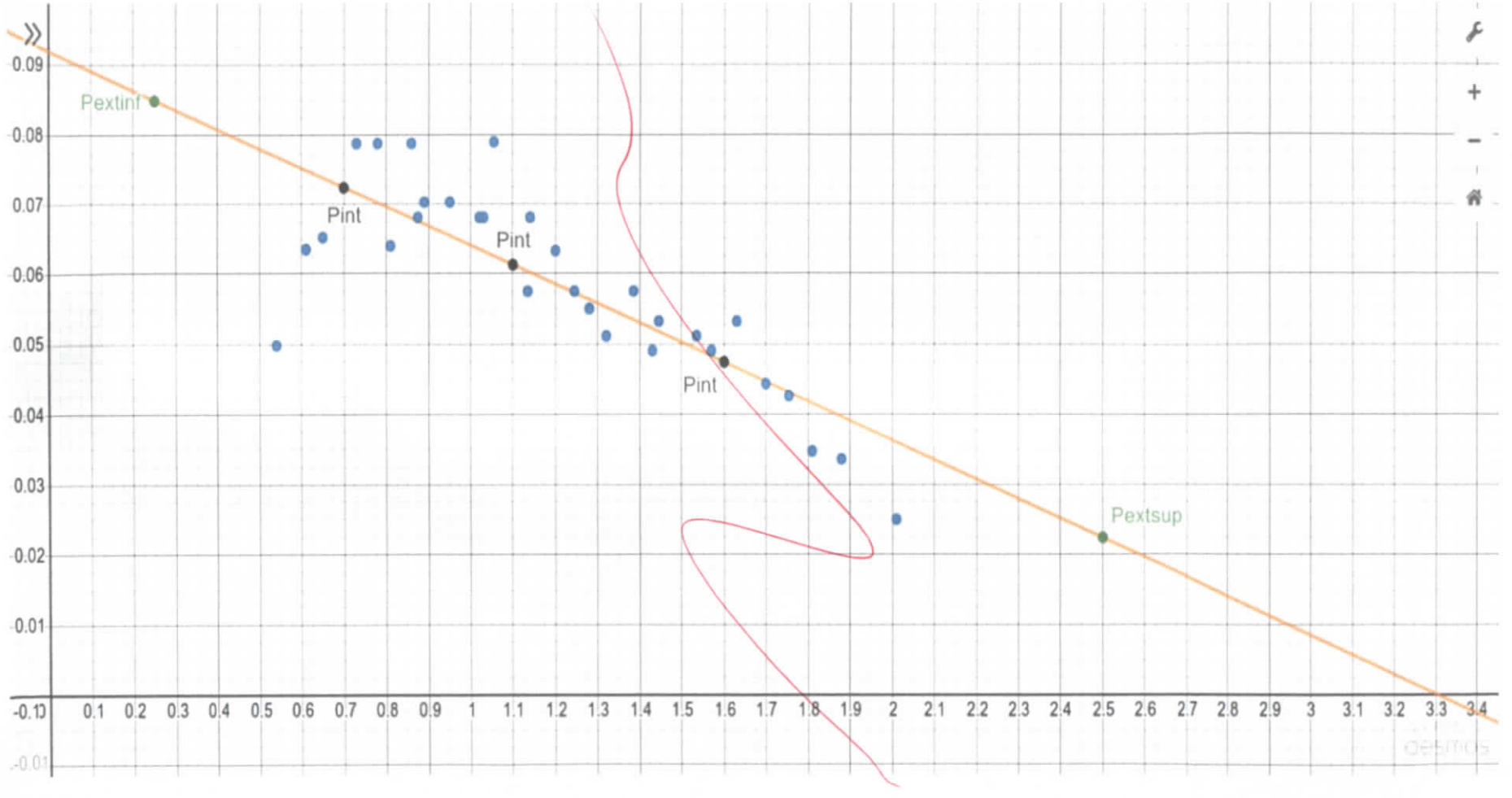
```
793     for (int i=1;i<=npunt; i++){
794         YY[i] = a[1] + a[2]* x[1][i] + a[3]*x[2][i] + a[4]*x[3][i];
795         delta[i] = ydat[i] - YY[i];
796         delta2[i] = delta[i]*delta[i];
797         beta[i] = ydat[i] - ymed;
798         beta2[i] = beta[i] * beta[i];
799         Sr += delta2[i];
800         St += beta2[i];
801
802         fprintf(fptr2,"\n%-6d| %4.2lf| %5.4lf| %7.6lf| %7.6lf| %9.6lf|"
803         " %9.6lf| %9.6lf|%9.6lf|%9.6lf|%9.6lf|%9.6lf|%9.6lf|%9.6lf|"
804         "%9.6lf|", i,xdat[i],ydat[i],x[2][i],x[3][i],x[4][i],x[5][i],x[6][i],
805         xy[i],x2y[i],x3y[i], YY[i],delta[i],delta2[i],beta[i],beta2[i]);
806     }
807     fprintf(fptr2,"\n");
808     //Imprimir linea divisoria
809     for (int i=1;i<=154;i++)
810     fprintf(fptr2,"?");
811     //Impresión de sumas totales, ymedia, coeficientes del modelo,
812     //y el modelo encontrado
813     fprintf(fptr2,"\n%s|%5.2lf| %6.4lf| %8.5lf| %8.5lf| %8.5lf| %8.5lf|"
814             "% 8.5lf| %8.5lf| %8.5lf|%9.5lf|%9s|%10s%9.6lf|%10s%9.6lf|",
815             "Sumas:",Sumx[1],Sumy,Sumx[2],Sumx[3],Sumx[4],Sumx[5],
816             Sumx[6],Sumxy,Sumx2y,Sumx3y,"","Sr = ",Sr,"St = ",St);
817
818     fprintf(fptr2, "\n\nYmedia = %lf\n\na0 = %.4lf, a1 = %.4lf,"
819             "a2 = %.4lf, a3 = %.4lf\n\n******MODELO CÚBICO******\n"
820             "Y = %.4lf %+.4lfx %+.4lfx^2 %+.4lfx^3\n\n",
821             ymed,a[1],a[2],a[3],a[4],a[1],a[2],a[3],a[4]);
822
823     //Cálculo e impresión de desviaciones del estimado y estándar
824     stdDev = sqrt(St/(npunt-1));
825     desEst = sqrt(Sr/(npunt-2));
826     fprintf(fptr2, "Desviación del estimado = %.6lf, "
827                    "Desviación estándar = %.6lf\n\n", desEst, stdDev);
828
829     //Cálculo e impresión de coeficientes de determinación y correlación
830     coefdet= (St-Sr)/St;
831     coefcor= sqrt(coefdet);
832     fprintf(fptr2, "Coeficiente de Determinación = %.6lf, "
833                    "Coeficiente de Correlación = %.6lf\n\n", coefdet, coefcor);
834
835     //Cálculo e impresion de los pronósticos
836     fprintf(fptr2,"PRONÓSTICOS:\n");
837     for(int i=0; i<=2; i++)
838     {
839         Yints[i]  = a[1] + a[2]*Xints[i] +
840         a[3]*pow(Xints[i],2) + a[4]*pow(Xints[i],3);
841         fprintf(fptr2,"Xint[%d] = %lf, Yint[%d] = %lf\n",
842                 i+1, Xints[i], Yints[i], i+1 );
843     }
844     Yextinf  = a[1] + a[2]*Xextinf + a[3]*pow(Xextinf,2) + a[4]*pow(Xextinf,3);
845     Yextsup  = a[1] + a[2]*Xextsup + a[3]*pow(Xextsup,2) + a[4]*pow(Xextsup,3);
846     fprintf(fptr2,"Xextinf = %lf, Yextinf = %lf\n",Xextinf, Yextinf);
847     fprintf(fptr2,"Xextsup = %lf, Yextsup = %lf\n",Xextsup, Yextsup);
848 }
```

```c
//ESTA CABECERA CONTIENE LAS FUNCIONES QUE RESUELVEN UN SISTEMA DE ECNS
//POR EL MÉTODO DE GAUSS QUE OCUPA LA FUNCIÓN MAIN.C PARA REALIZAR LOS
//AJUSTES APROPIADOS
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
void Elimina(int matsize, double **C);
double * Sust_Atras(int matsize, double ** C);
double * Gauss (int matsize, double ** A);
int ValMaxCol(int rowpvt, int matsize, double * vec);
void Pivoteo(int matsize, double **C, int r, int rmax);

double * Gauss (int matsize, double ** A){
    //Declaracion de la matriz C donde se copiara la matriz A
    double ** C, *X;
    //Numero de filas ira de 0 a n. Pero por practicidad se usaran solo las filas
    //de la 1 a la n. Por eso se declaran n+1 filas.
    C = (double **) malloc ((matsize+1) * sizeof(double *));
    for (int j=1; j <= matsize; j++)
    //Numero de columnas ira de 0 a n+1; por eso se declaran n+2 columnas
    C[j] = (double *) malloc ((matsize+2)*sizeof(double));
    //Copia la matriz A a la matriz C
    for(int i=1; i<=matsize; i++)
    for (int j=1; j<=matsize+1; j++)
        C[i][j] = A[i][j];
    Elimina(matsize,C);
    X = Sust_Atras(matsize,C);
    return X;
}
void Elimina(int matsize, double **C){
    //Aqui se hace dominante a la matriz
    double * copia;
    double pivot,tempA, tempB, cons;

    //se crea el vector donde se copiaran los elementos de la columna r
    copia = (double *) malloc ((matsize+1) * sizeof(double));

    //Se fija la columna r
    for (int r=1; r<=matsize-1; r++){
        // se copian los elementos de esta columna (r)
        // desde la fila r hasta la fila n.
        for (int i=r; i<=matsize; i++){
            copia[i] = C[i][r];
        }
    int rmax = ValMaxCol(r,matsize+1,copia);
    if (rmax != r) Pivoteo(matsize, C, r, rmax);
    pivot = C[r][r];
    for (int i=r+1; i<=matsize;i++){
            cons = -C[i][r];
        for (int j=1; j<=matsize+1; j++){
            tempA = C[r][j] * cons;
            tempB = C[i][j] * pivot;
            C[i][j] = tempA + tempB;
            }
        }
    }
}
double * Sust_Atras(int matsize, double ** C){
     //En este vector se regresaran los valores de Xi
     double * X = (double *) malloc((matsize+1) * sizeof(double));

     X[matsize] = C[matsize][matsize+1]/C[matsize][matsize];
     for (int i=matsize-1 ;i>=1; i--){
     double Suma=0.;
     for (int j=i+1; j<=matsize; j++)
        Suma += (C[i][j]*X[j]);
```

```
        X[i]= (C[i][matsize+1] - Suma)/C[i][i];
    }
    return X;
}
int ValMaxCol(int rowpvt, int matsize, double * vec){
   int rmax = rowpvt;
   for (int i=rowpvt+1; i<=matsize; i++)
   if (fabs(vec[i]) > fabs(vec[rmax])) rmax = i;
   return rmax;
}
void Pivoteo(int matsize, double **C, int r, int rmax){
    double temp;
    for (int j=1; j<=matsize+1; j++){
            temp = C[rmax][j];
            C[rmax][j] = C[r][j];
            C[r][j] = temp;
    }
}
```

GRÁFICAS

AJUSTE LINEAL

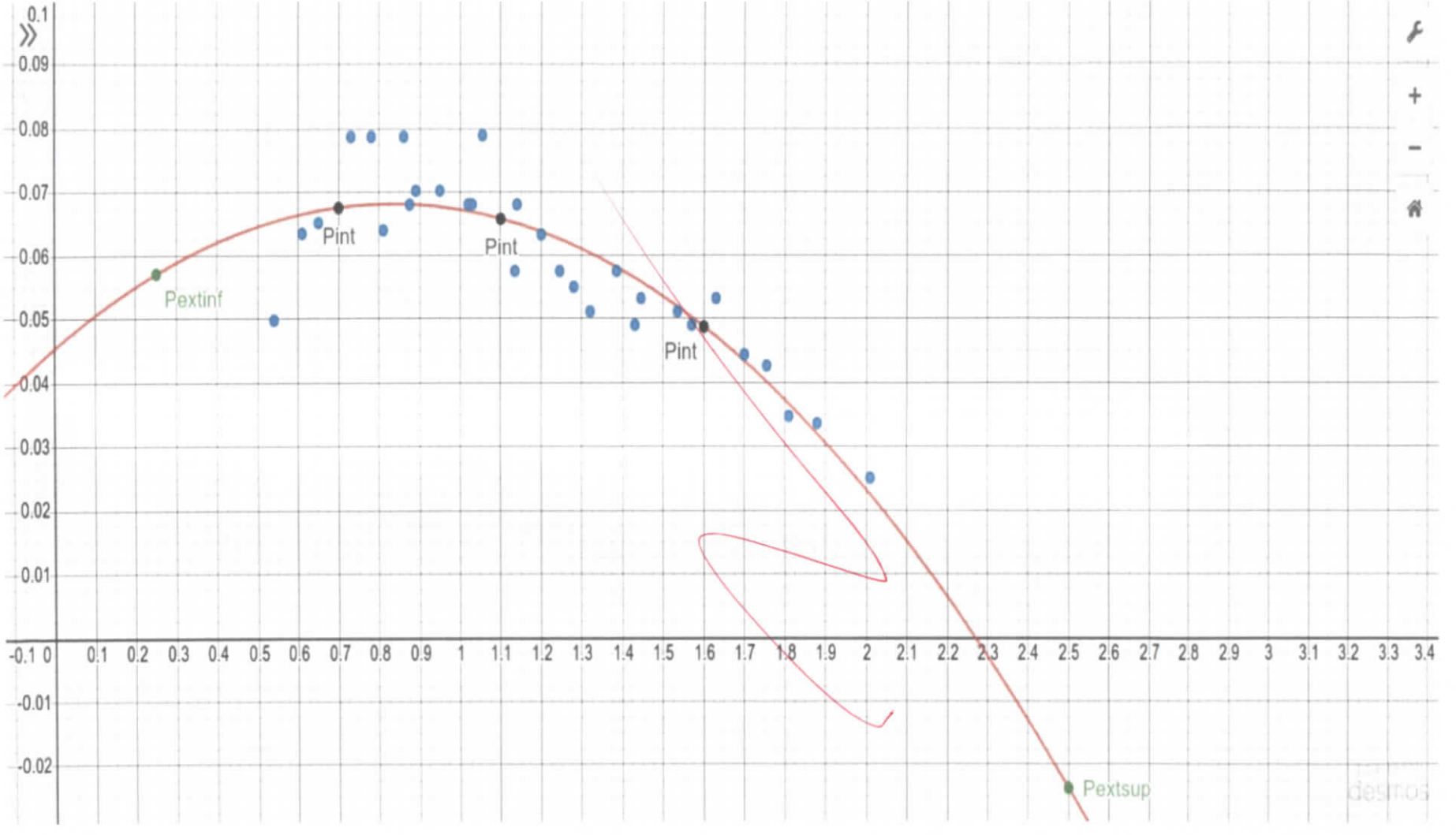


$Y = a_0 + a_1x$

$Y = 0.0919 - 0.0278x$

Pint - Pronósticos internos
Pextsup - Pronóstico externo superior
Pextinf - Pronóstico externo inferior

AJUSTE PARABÓLICO



$Y = a_0 + a_1x + a_2x^2$

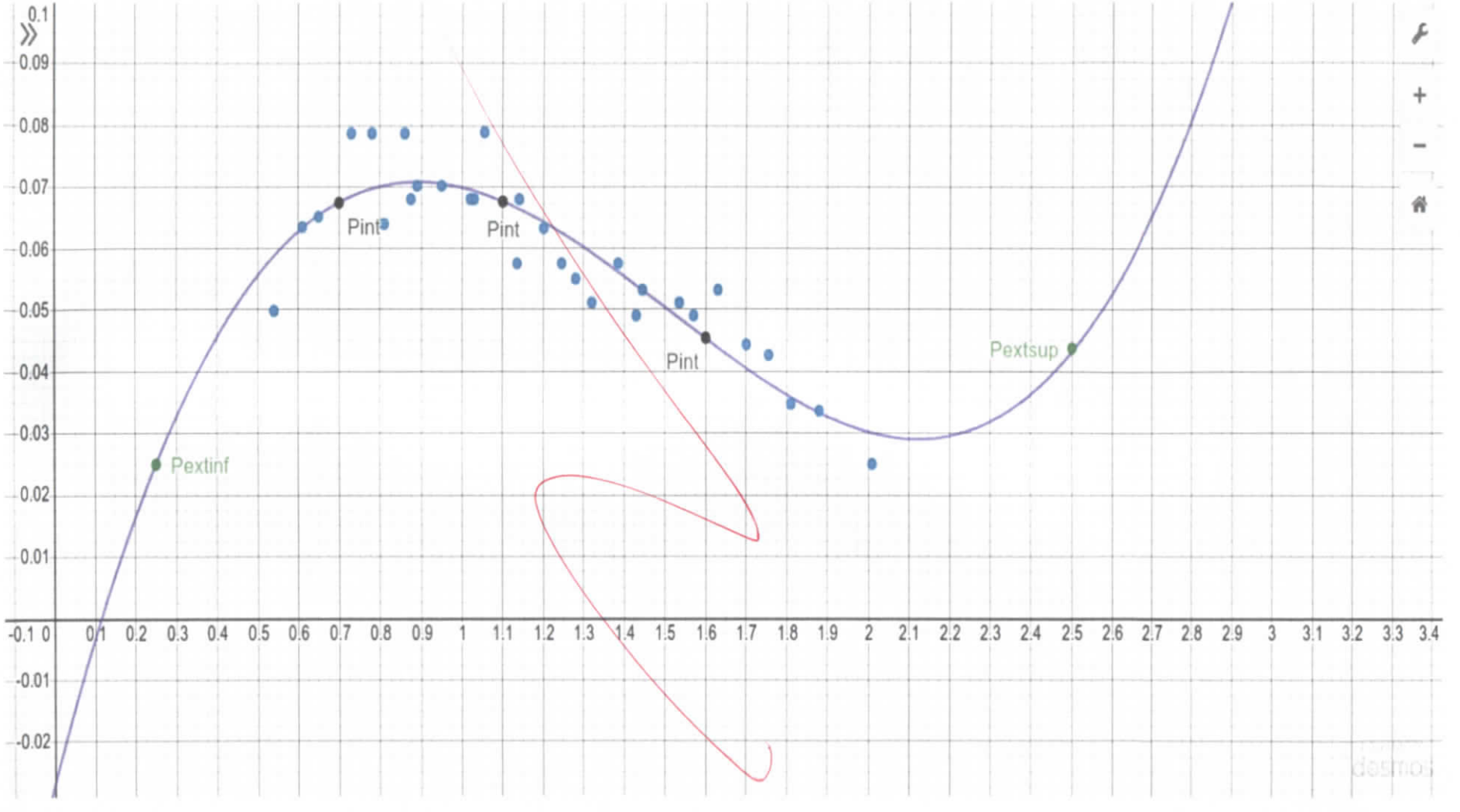
Y=0.0454+0.0549x-0.0330x^2

Pint - Pronósticos internos

Pextsup - Pronóstico externo superior

Pextinf - Pronóstico externo inferior

AJUSTE CÚBICO

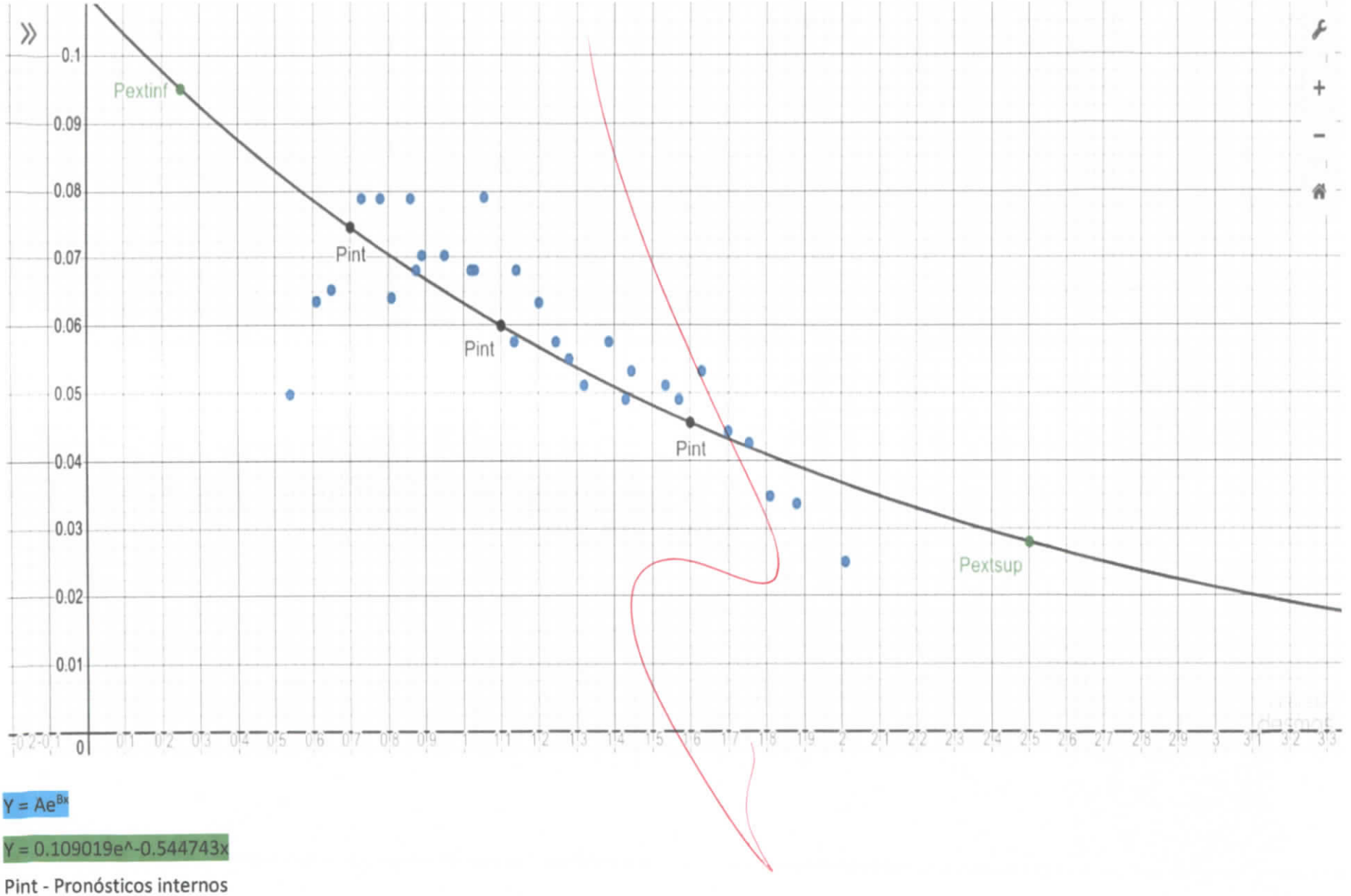


$Y = a_0 + a_1x + a_2x^2 + a_3x^3$

Y = -0.0266 +0.2544x -0.2026x^2 +0.0448x^3

Pint - Pronósticos internos
Pextsup - Pronóstico externo superior
Pextinf - Pronóstico externo inferior

AJUSTE O MODELO EXPONENCIAL



$Y = Ae^{Bx}$

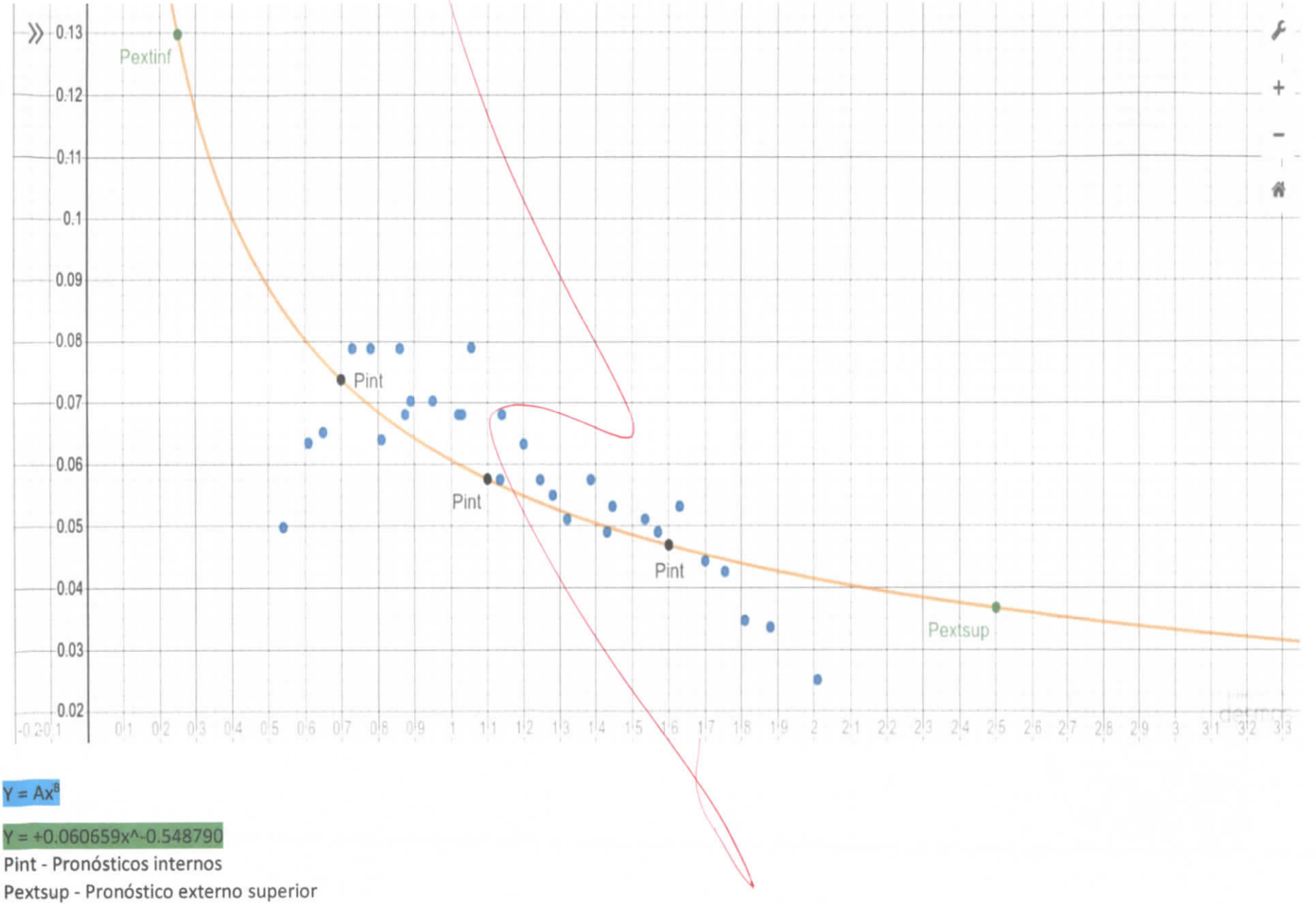
Y = 0.109019e^-0.544743x

Pint - Pronósticos internos

Pextsup - Pronóstico externo superior

Pextinf - Pronóstico externo inferior

AJUSTE O MODELO DE ECUACIÓN DE POTENCIAS



$Y = Ax^B$

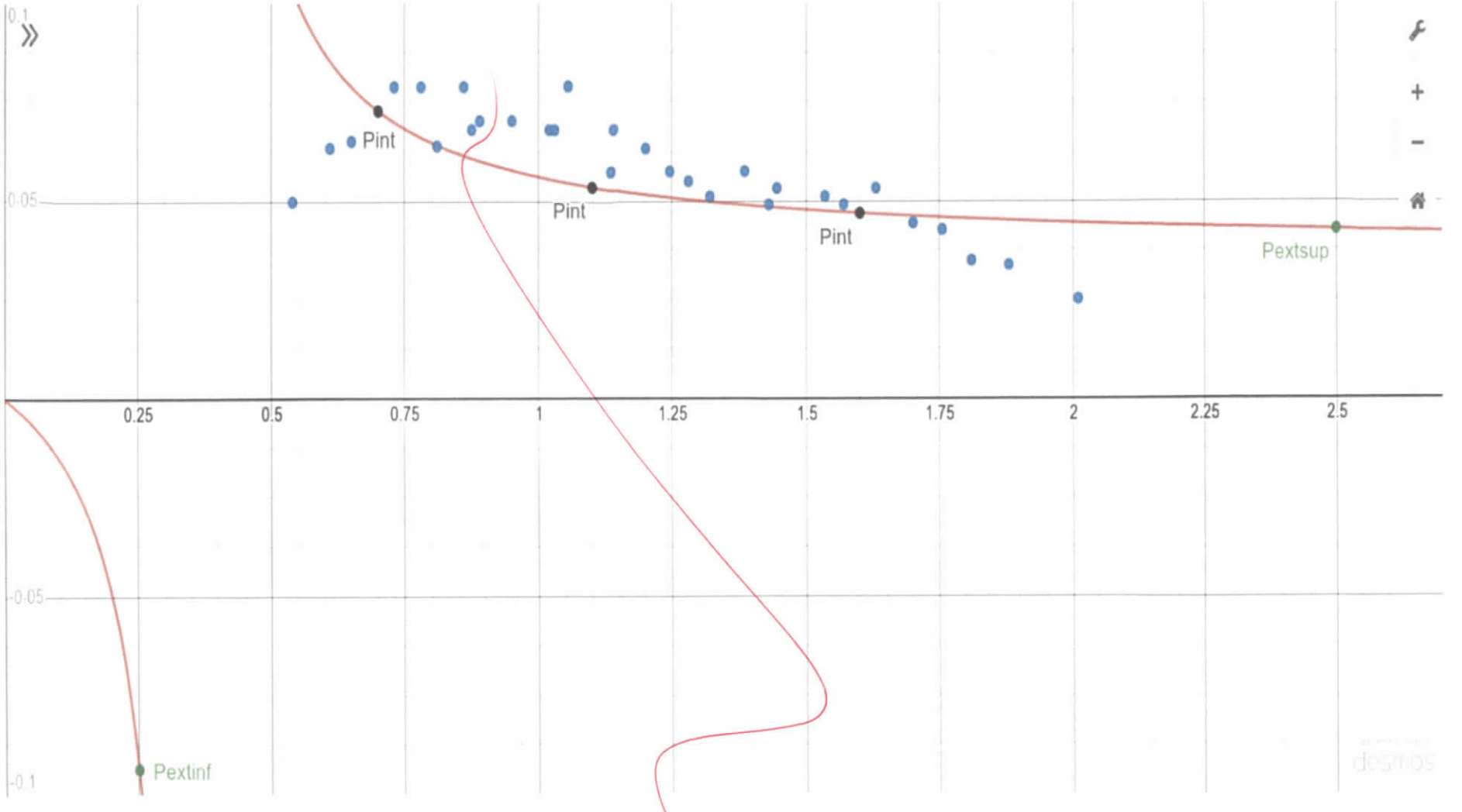
Y = +0.060659x^-0.548790

Pint - Pronósticos internos

Pextsup - Pronóstico externo superior

Pextinf - Pronóstico externo inferior

AJUSTE O MODELO DE LA RAZÓN DE CRECIMIENTO



$Y = A\frac{x}{B+x}$

Y = +0.036649x / (-0.347734+x)

Pint - Pronósticos internos

Pextsup - Pronóstico externo superior

Pextinf - Pronóstico externo inferior

File: /home/ayax/Documents/C/Sistemas3/Resultados.dat

----------MODELO LINEAL----------

Punto	X	Y	X²	XY	ZZ	Delta	Delta²	Beta	Beta²
1	0.540	0.0498	0.291600	0.026892	0.076878	-0.027078	0.000733	-0.008483	0.000072
2	0.610	0.0635	0.372100	0.038735	0.074932	-0.011432	0.000131	0.005217	0.000027
3	0.650	0.0652	0.422500	0.042380	0.073820	-0.008620	0.000074	0.006917	0.000048
4	0.730	0.0788	0.532900	0.057524	0.071597	0.007203	0.000052	0.020517	0.000421
5	0.780	0.0788	0.608400	0.061464	0.070207	0.008593	0.000074	0.020517	0.000421
6	0.810	0.0640	0.656100	0.051840	0.069373	-0.005373	0.000029	0.005717	0.000033
7	0.860	0.0788	0.739600	0.067768	0.067984	0.010816	0.000117	0.020517	0.000421
8	0.875	0.0681	0.765625	0.059587	0.067567	0.000533	0.000000	0.009817	0.000096
9	0.890	0.0703	0.792100	0.062567	0.067150	0.003150	0.000010	0.012017	0.000144
10	0.950	0.0703	0.902500	0.066785	0.065482	0.004818	0.000023	0.012017	0.000144
11	1.020	0.0681	1.040400	0.069462	0.063536	0.004564	0.000021	0.009817	0.000096
12	1.030	0.0681	1.060900	0.070143	0.063259	0.004841	0.000023	0.009817	0.000096
13	1.055	0.0790	1.113025	0.083345	0.062564	0.016436	0.000270	0.020717	0.000429
14	1.135	0.0575	1.288225	0.065263	0.060340	-0.002840	0.000008	-0.000783	0.000001
15	1.140	0.0681	1.299600	0.077634	0.060201	0.007899	0.000062	0.009817	0.000096
16	1.200	0.0633	1.440000	0.075960	0.058533	0.004767	0.000023	0.005017	0.000025
17	1.245	0.0575	1.550025	0.071588	0.057283	0.000217	0.000000	-0.000783	0.000001
18	1.280	0.0550	1.638400	0.070400	0.056310	-0.001310	0.000002	-0.003283	0.000011
19	1.320	0.0511	1.742400	0.067452	0.055198	-0.004098	0.000017	-0.007183	0.000052
20	1.385	0.0575	1.918225	0.079637	0.053392	0.004108	0.000017	-0.000783	0.000001
21	1.430	0.0490	2.044900	0.070070	0.052141	-0.003141	0.000010	-0.009283	0.000086
22	1.445	0.0532	2.088025	0.076874	0.051724	0.001476	0.000002	-0.005083	0.000026
23	1.535	0.0511	2.356225	0.078438	0.049222	0.001878	0.000004	-0.007183	0.000052
24	1.570	0.0490	2.464900	0.076930	0.048250	0.000750	0.000001	-0.009283	0.000086
25	1.630	0.0532	2.656900	0.086716	0.046582	0.006618	0.000044	-0.005083	0.000026
26	1.700	0.0443	2.890000	0.075310	0.044636	-0.000336	0.000000	-0.013983	0.000196
27	1.755	0.0426	3.080025	0.074763	0.043108	-0.000508	0.000000	-0.015683	0.000246
28	1.810	0.0347	3.276100	0.062807	0.041579	-0.006879	0.000047	-0.023583	0.000556
29	1.880	0.0336	3.534400	0.063168	0.039633	-0.006033	0.000036	-0.024683	0.000609
30	2.010	0.0250	4.040100	0.050250	0.036020	-0.011020	0.000121	-0.033283	0.001108
Sumas:	36.27	1.7485	48.60620	1.981752		Sr =	0.001952	St =	0.005626

Ymedia = 0.058283

a0 = 0.0919, a1 = -0.0278

******MODELO LINEAL******
Y = 0.0919 -0.0278x

File: /home/ayax/Documents/C/Sistemas3/Resultados.dat

Desviación del estimado = 0.008349, Desviación estándar = 0.013928

Coeficiente de Determinación = 0.653071, Coeficiente de Correlación = 0.808128

PRONÓSTICOS:
Xint[1] = 0.700, Yint[1] = 0.0724
Xint[2] = 1.100, Yint[2] = 0.0613
Xint[3] = 1.600, Yint[3] = 0.0474
Xextinf = 0.250, Yextinf = 0.0849
Xextsup = 2.500, Yextsup = 0.0224

----------MODELO EXPONENCIAL----------

Punto	X	Y	Z = ln(Y)	X²	XZ	ZZ	YY	Delta	Delta²	Beta	Beta²
1	0.540	0.0498	-2.999740	0.291600	-1.619860	-2.510392	0.081236	-0.031436	0.000988	-0.008483	0.000072
2	0.610	0.0635	-2.756715	0.372100	-1.681596	-2.548524	0.078197	-0.014697	0.000216	0.005217	0.000027
3	0.650	0.0652	-2.730296	0.422500	-1.774692	-2.570314	0.076512	-0.011312	0.000128	0.006917	0.000048
4	0.730	0.0788	-2.540842	0.532900	-1.854815	-2.613893	0.073249	0.005551	0.000031	0.020517	0.000421
5	0.780	0.0788	-2.540842	0.608400	-1.981857	-2.641130	0.071281	0.007519	0.000057	0.020517	0.000421
6	0.810	0.0640	-2.748872	0.656100	-2.226586	-2.657472	0.070125	-0.006125	0.000038	0.005717	0.000033
7	0.860	0.0788	-2.540842	0.739600	-2.185124	-2.684710	0.068241	0.010559	0.000111	0.020517	0.000421
8	0.875	0.0681	-2.686778	0.765625	-2.350931	-2.692881	0.067686	0.000414	0.000000	0.009817	0.000096
9	0.890	0.0703	-2.654983	0.792100	-2.362935	-2.701052	0.067135	0.003165	0.000010	0.012017	0.000144
10	0.950	0.0703	-2.654983	0.902500	-2.522234	-2.733736	0.064976	0.005324	0.000028	0.012017	0.000144
11	1.020	0.0681	-2.686778	1.040400	-2.740514	-2.771868	0.062545	0.005555	0.000031	0.009817	0.000096
12	1.030	0.0681	-2.686778	1.060900	-2.767381	-2.777316	0.062205	0.005895	0.000035	0.009817	0.000096
13	1.055	0.0790	-2.538307	1.113025	-2.677914	-2.790934	0.061364	0.017636	0.000311	0.020717	0.000429
14	1.135	0.0575	-2.855970	1.288225	-3.241526	-2.834514	0.058747	-0.001247	0.000002	-0.000783	0.000001
15	1.140	0.0681	-2.686778	1.299600	-3.062927	-2.837237	0.058587	0.009513	0.000090	0.009817	0.000096
16	1.200	0.0633	-2.759870	1.440000	-3.311844	-2.869922	0.056703	0.006597	0.000044	0.005017	0.000025
17	1.245	0.0575	-2.855970	1.550025	-3.555683	-2.894435	0.055330	0.002170	0.000005	-0.000783	0.000001
18	1.280	0.0550	-2.900422	1.638400	-3.712540	-2.913501	0.054285	0.000715	0.000001	-0.003283	0.000011
19	1.320	0.0511	-2.973971	1.742400	-3.925641	-2.935291	0.053115	-0.002015	0.000004	-0.007183	0.000052
20	1.385	0.0575	-2.855970	1.918225	-3.955519	-2.970699	0.051267	0.006233	0.000039	-0.000783	0.000001
21	1.430	0.0490	-3.015935	2.044900	-4.312787	-2.995213	0.050026	-0.001026	0.000001	-0.009283	0.000086
22	1.445	0.0532	-2.933697	2.088025	-4.239192	-3.003384	0.049619	0.003581	0.000013	-0.005083	0.000026
23	1.535	0.0511	-2.973971	2.356225	-4.565045	-3.052411	0.047245	0.003855	0.000015	-0.007183	0.000052
24	1.570	0.0490	-3.015935	2.464900	-4.735018	-3.071477	0.046353	0.002647	0.000007	-0.009283	0.000086
25	1.630	0.0532	-2.933697	2.656900	-4.781926	-3.104161	0.044862	0.008338	0.000070	-0.005083	0.000026
26	1.700	0.0443	-3.116771	2.890000	-5.298510	-3.142293	0.043184	0.001116	0.000001	-0.013983	0.000196
27	1.755	0.0426	-3.155901	3.080025	-5.538606	-3.172254	0.041909	0.000691	0.000000	-0.015683	0.000246
28	1.810	0.0347	-3.361016	3.276100	-6.083438	-3.202215	0.040672	-0.005972	0.000036	-0.023583	0.000556
29	1.880	0.0336	-3.393229	3.534400	-6.379271	-3.240347	0.039150	-0.005550	0.000031	-0.024683	0.000609

File: /home/ayax/Documents/C/Sistemas3/Resultados.dat

```
30    | 2.010| 0.0250| -3.688879| 4.040100|  -7.414648| -3.311164| 0.036474| -0.011474|  0.000132| -0.033283|  0.001108|
Sumas:| 36.27| 1.7485| -86.24474| 48.60620| -106.860563|          |       Sr =  0.002473|       St =  0.005626|

Ymedia = 0.058283

a0 = -2.2162, a1 = -0.5447

******MODELO LINEALIZADO******
Z = -2.2162 -0.5447x

Desviación del estimado = 0.009397, Desviación estándar = 0.013928

Coeficiente de Determinación = 0.560493, Coeficiente de Correlación = 0.748661


Y = Ae^(Bx)
Tenemos que:
a0 = ln(A) = -2.216231 ∴ A = e^a0 = e^(-2.216231) = 0.109019
a1 = B = -0.544743

*****MODELO EXPONENCIAL*****
Y = +0.109019e^-0.544743x

PRONÓSTICOS:
Xint[1] = 0.700, Yint[1] = 0.0745
Xint[2] = 1.100, Yint[2] = 0.0599
Xint[3] = 1.600, Yint[3] = 0.0456
Xextinf = 0.250, Yextinf = 0.0951
Xextsup = 2.500, Yextsup = 0.0279

----------------------------------------------------MODELO PARABÓLICO----------------------------------------------------

Punto       X       Y        X²        X³        X⁴        XY        X²Y        YY       Delta      Delta²      Beta       Beta²
1     | 0.540| 0.0498| 0.291600| 0.157464| 0.085031| 0.026892| 0.014522| 0.065414| -0.015614| 0.000244| -0.008483| 0.000072|
2     | 0.610| 0.0635| 0.372100| 0.226981| 0.138458| 0.038735| 0.023628| 0.066600| -0.003100| 0.000010|  0.005217| 0.000027|
3     | 0.650| 0.0652| 0.422500| 0.274625| 0.178506| 0.042380| 0.027547| 0.067132| -0.001932| 0.000004|  0.006917| 0.000048|
4     | 0.730| 0.0788| 0.532900| 0.389017| 0.283982| 0.057524| 0.041993| 0.067880|  0.010920| 0.000119|  0.020517| 0.000421|
5     | 0.780| 0.0788| 0.608400| 0.474552| 0.370151| 0.061464| 0.047942| 0.068132|  0.010668| 0.000114|  0.020517| 0.000421|
6     | 0.810| 0.0640| 0.656100| 0.531441| 0.430467| 0.051840| 0.041990| 0.068205| -0.004205| 0.000018|  0.005717| 0.000033|
7     | 0.860| 0.0788| 0.739600| 0.636056| 0.547008| 0.067768| 0.058280| 0.068193|  0.010607| 0.000113|  0.020517| 0.000421|
8     | 0.875| 0.0681| 0.765625| 0.669922| 0.586182| 0.059587| 0.052139| 0.068158| -0.000058| 0.000000|  0.009817| 0.000096|
```

File: /home/ayax/Documents/C/Sistemas3/Resultados.dat

9	0.890	0.0703	0.792100	0.704969	0.627422	0.062567	0.055685	0.068107	0.002193	0.000005	0.012017	0.000144
10	0.950	0.0703	0.902500	0.857375	0.814506	0.066785	0.063446	0.067757	0.002543	0.000006	0.012017	0.000144
11	1.020	0.0681	1.040400	1.061208	1.082432	0.069462	0.070851	0.067048	0.001052	0.000001	0.009817	0.000096
12	1.030	0.0681	1.060900	1.092727	1.125509	0.070143	0.072247	0.066920	0.001180	0.000001	0.009817	0.000096
13	1.055	0.0790	1.113025	1.174241	1.238825	0.083345	0.087929	0.066572	0.012428	0.000154	0.020717	0.000429
14	1.135	0.0575	1.288225	1.462135	1.659524	0.065263	0.074073	0.065180	-0.007680	0.000059	-0.000783	0.000001
15	1.140	0.0681	1.299600	1.481544	1.688960	0.077634	0.088503	0.065079	0.003021	0.000009	0.009817	0.000096
16	1.200	0.0633	1.440000	1.728000	2.073600	0.075960	0.091152	0.063738	-0.000438	0.000000	0.005017	0.000025
17	1.245	0.0575	1.550025	1.929781	2.402578	0.071588	0.089126	0.062577	-0.005077	0.000026	-0.000783	0.000001
18	1.280	0.0550	1.638400	2.097152	2.684355	0.070400	0.090112	0.061581	-0.006581	0.000043	-0.003283	0.000011
19	1.320	0.0511	1.742400	2.299968	3.035958	0.067452	0.089037	0.060343	-0.009243	0.000085	-0.007183	0.000052
20	1.385	0.0575	1.918225	2.656742	3.679587	0.079637	0.110298	0.058108	-0.000608	0.000000	-0.000783	0.000001
21	1.430	0.0490	2.044900	2.924207	4.181616	0.070070	0.100200	0.056397	-0.007397	0.000055	-0.009283	0.000086
22	1.445	0.0532	2.088025	3.017196	4.359848	0.076874	0.111083	0.055796	-0.002596	0.000007	-0.005083	0.000026
23	1.535	0.0511	2.356225	3.616805	5.551796	0.078438	0.120403	0.051884	-0.000784	0.000001	-0.007183	0.000052
24	1.570	0.0490	2.464900	3.869893	6.075732	0.076930	0.120780	0.050218	-0.001218	0.000001	-0.009283	0.000086
25	1.630	0.0532	2.656900	4.330747	7.059118	0.086716	0.141347	0.047173	0.006027	0.000036	-0.005083	0.000026
26	1.700	0.0443	2.890000	4.913000	8.352100	0.075310	0.128027	0.043321	0.000979	0.000001	-0.013983	0.000196
27	1.755	0.0426	3.080025	5.405444	9.486554	0.074763	0.131209	0.040068	0.002532	0.000006	-0.015683	0.000246
28	1.810	0.0347	3.276100	5.929741	10.732831	0.062807	0.113681	0.036615	-0.001915	0.000004	-0.023583	0.000556
29	1.880	0.0336	3.534400	6.644672	12.491983	0.063168	0.118756	0.031931	0.001669	0.000003	-0.024683	0.000609
30	2.010	0.0250	4.040100	8.120601	16.322408	0.050250	0.101002	0.022374	0.002626	0.000007	-0.033283	0.001108
Sumas:	36.27	1.7485	48.60620	70.67821	109.3470	1.981752	2.4770		Sr =	0.001132	St =	0.005626

Ymedia = 0.058283

a0 = 0.0454, a1 = 0.0549, a2 = -0.0330

******MODELO PARABÓLICO******
Y = 0.0454 +0.0549x -0.0330x^2

Desviación del estimado = 0.006359, Desviación estándar = 0.013928

Coeficiente de Determinación = 0.798710, Coeficiente de Correlación = 0.893706

PRONÓSTICOS:
Xint[1] = 0.700, Yint[1] = 0.0676
Xint[2] = 1.100, Yint[2] = 0.0658
Xint[3] = 1.600, Yint[3] = 0.0487
Xextinf = 0.250, Yextinf = 0.0571
Xextsup = 2.500, Yextsup = -0.0237

File: /home/ayax/Documents/C/Sistemas3/Resultados.dat

----------MODELO DE POTENCIAS----------

Punto	X	Y	W = log(X)	Z = log(Y)	W²	WZ	ZZ	YY	Delta	Delta²	Beta	Beta²
1	0.540	0.0498	-0.267606	-1.302771	0.071613	0.348630	-1.070243	0.085066	-0.035266	0.001244	-0.008483	0.000072
2	0.610	0.0635	-0.214670	-1.197226	0.046083	0.257009	-1.099293	0.079562	-0.016062	0.000258	0.005217	0.000027
3	0.650	0.0652	-0.187087	-1.185752	0.035001	0.221838	-1.114431	0.076837	-0.011637	0.000135	0.006917	0.000048
4	0.730	0.0788	-0.136677	-1.103474	0.018681	0.150820	-1.142095	0.072095	0.006705	0.000045	0.020517	0.000421
5	0.780	0.0788	-0.107905	-1.103474	0.011644	0.119071	-1.157885	0.069521	0.009279	0.000086	0.020517	0.000421
6	0.810	0.0640	-0.091515	-1.193820	0.008375	0.109252	-1.166880	0.068096	-0.004096	0.000017	0.005717	0.000033
7	0.860	0.0788	-0.065502	-1.103474	0.004290	0.072279	-1.181156	0.065894	0.012906	0.000167	0.020517	0.000421
8	0.875	0.0681	-0.057992	-1.166853	0.003363	0.067668	-1.185277	0.065271	0.002829	0.000008	0.009817	0.000096
9	0.890	0.0703	-0.050610	-1.153045	0.002561	0.058356	-1.189328	0.064665	0.005635	0.000032	0.012017	0.000144
10	0.950	0.0703	-0.022276	-1.153045	0.000496	0.025686	-1.204877	0.062391	0.007909	0.000063	0.012017	0.000144
11	1.020	0.0681	0.008600	-1.166853	0.000074	-0.010035	-1.221822	0.060004	0.008096	0.000066	0.009817	0.000096
12	1.030	0.0681	0.012837	-1.166853	0.000165	-0.014979	-1.224147	0.059683	0.008417	0.000071	0.009817	0.000096
13	1.055	0.0790	0.023252	-1.102373	0.000541	-0.025633	-1.229863	0.058903	0.020097	0.000404	0.020717	0.000429
14	1.135	0.0575	0.054996	-1.240332	0.003025	-0.068213	-1.247284	0.056587	0.000913	0.000001	-0.000783	0.000001
15	1.140	0.0681	0.056905	-1.166853	0.003238	-0.066400	-1.248331	0.056451	0.011649	0.000136	0.009817	0.000096
16	1.200	0.0633	0.079181	-1.198596	0.006270	-0.094906	-1.260556	0.054884	0.008416	0.000071	0.005017	0.000025
17	1.245	0.0575	0.095169	-1.240332	0.009057	-0.118042	-1.269330	0.053786	0.003714	0.000014	-0.000783	0.000001
18	1.280	0.0550	0.107210	-1.259637	0.011494	-0.135046	-1.275938	0.052974	0.002026	0.000004	-0.003283	0.000011
19	1.320	0.0511	0.120574	-1.291579	0.014538	-0.155731	-1.283272	0.052087	-0.000987	0.000001	-0.007183	0.000052
20	1.385	0.0575	0.141450	-1.240332	0.020008	-0.175445	-1.294729	0.050731	0.006769	0.000046	-0.000783	0.000001
21	1.430	0.0490	0.155336	-1.309804	0.024129	-0.203460	-1.302349	0.049848	-0.000848	0.000001	-0.009283	0.000086
22	1.445	0.0532	0.159868	-1.274088	0.025558	-0.203686	-1.304836	0.049564	0.003636	0.000013	-0.005083	0.000026
23	1.535	0.0511	0.186108	-1.291579	0.034636	-0.240374	-1.319237	0.047947	0.003153	0.000010	-0.007183	0.000052
24	1.570	0.0490	0.195900	-1.309804	0.038377	-0.256590	-1.324610	0.047358	0.001642	0.000003	-0.009283	0.000086
25	1.630	0.0532	0.212188	-1.274088	0.045024	-0.270346	-1.333549	0.046393	0.006807	0.000046	-0.005083	0.000026
26	1.700	0.0443	0.230449	-1.353596	0.053107	-0.311935	-1.343571	0.045335	-0.001035	0.000001	-0.013983	0.000196
27	1.755	0.0426	0.244277	-1.370590	0.059671	-0.334804	-1.351159	0.044549	-0.001949	0.000004	-0.015683	0.000246
28	1.810	0.0347	0.257679	-1.459671	0.066398	-0.376126	-1.358514	0.043801	-0.009101	0.000083	-0.023583	0.000556
29	1.880	0.0336	0.274158	-1.473661	0.075163	-0.404016	-1.367558	0.042899	-0.009299	0.000086	-0.024683	0.000609
30	2.010	0.0250	0.303196	-1.602060	0.091928	-0.485738	-1.383493	0.041353	-0.016353	0.000267	-0.033283	0.001108
Sumas:	36.27	1.7485	1.71749	-37.45562	0.78451	-2.520894			Sr = 0.003381		St = 0.005626	

Ymedia = 0.058283

a0 = -1.2171, a1 = -0.5488

******MODELO LINEALIZADO******
Z = -1.2171 -0.5488x

File: /home/ayax/Documents/C/Sistemas3/Resultados.dat

Desviación del estimado = 0.010988, Desviación estándar = 0.013928

Coeficiente de Determinación = 0.399065, Coeficiente de Correlación = 0.631716

Y = Ax^B
Tenemos que:
a0 = log(A) = -1.217102 ∴ A = 10^a0 = 10^(-1.217102) = 0.060659
a1 = B = -0.548790

*****MODELO DE POTENCIAS*****
Y = +0.060659x^-0.548790

PRONÓSTICOS:
Xint[1] = 0.7000, Yint[1] = 0.0738
Xint[2] = 1.1000, Yint[2] = 0.0576
Xint[3] = 1.6000, Yint[3] = 0.0469
Xextinf = 0.2500, Yextinf = 0.1298
Xextsup = 2.5000, Yextsup = 0.0367

----------MODELO DE CRECIMIENTO----------

Punto	X	Y	V = 1/X	Z = 1/Y	V²	VZ	ZZ	YY	Delta	Delta²	Beta	Beta²
1	0.540	0.0498	1.851852	20.080321	3.429355	37.185780	9.715075	0.102933	-0.053133	0.002823	-0.008483	0.000072
2	0.610	0.0635	1.639344	15.748031	2.687450	25.816445	11.731386	0.085241	-0.021741	0.000473	0.005217	0.000027
3	0.650	0.0652	1.538462	15.337423	2.366864	23.596036	12.688580	0.078811	-0.013611	0.000185	0.006917	0.000048
4	0.730	0.0788	1.369863	12.690355	1.876525	17.384048	14.288274	0.069987	0.008813	0.000078	0.020517	0.000421
5	0.780	0.0788	1.282051	12 690355	1.643655	16.269686	15.121448	0.066131	0.012669	0.000160	0.020517	0.000421
6	0.810	0.0640	1.234568	15.625000	1.524158	19.290123	15.571979	0.064218	-0.000218	0.000000	0.005717	0.000033
7	0.860	0.0788	1.162791	12.690355	1.352082	14.756227	16.253014	0.061527	0.017273	0.000298	0.020517	0.000421
8	0.875	0.0681	1.142857	14.684288	1.306122	16.782043	16.442148	0.060819	0.007281	0.000053	0.009817	0.000096
9	0.890	0.0703	1.123596	14.224751	1.262467	15.982866	16.624906	0.060151	0.010149	0.000103	0.012017	0.000144
10	0.950	0.0703	1.052632	14.224751	1.108033	14.973422	17.298224	0.057809	0.012491	0.000156	0.012017	0.000144
11	1.020	0.0681	0.980392	14.684288	0.961169	14.396361	17.983645	0.055606	0.012494	0.000156	0.009817	0.000096
12	1.030	0.0681	0.970874	14.684288	0.942596	14.256590	18.073958	0.055328	0.012772	0.000163	0.009817	0.000096
13	1.055	0.0790	0.947867	12.658228	0.898452	11.998320	18.292247	0.054668	0.024332	0.000592	0.020717	0.000429
14	1.135	0.0575	0.881057	17.391304	0.776262	15.322735	18.926153	0.052837	0.004663	0.000022	-0.000783	0.000001
15	1.140	0.0681	0.877193	14.684288	0.769468	12.880954	18.962818	0.052735	0.015365	0.000236	0.009817	0.000096
16	1.200	0.0633	0.833333	15.797788	0.694444	13.164824	19.378967	0.051602	0.011698	0.000137	0.005017	0.000025
17	1.245	0.0575	0.803213	17.391304	0.645151	13.968919	19.664755	0.050852	0.006648	0.000044	-0.000783	0.000001
18	1.280	0.0550	0.781250	18.181818	0.610352	14.204545	19.873143	0.050319	0.004681	0.000022	-0.003283	0.000011
19	1.320	0.0511	0.757576	19.569472	0.573921	14.825357	20.097769	0.049757	0.001343	0.000002	-0.007183	0.000052
20	1.385	0.0575	0.722022	17.391304	0.521315	12.556898	20.435112	0.048935	0.008565	0.000073	-0.000783	0.000001

File: /home/ayax/Documents/C/Sistemas3/Resultados.dat

21	1.430	0.0490	0.699301	20.408163	0.489021	14.271443	20.650693	0.048425	0.000575	0.000000	-0.009283	0.000086
22	1.445	0.0532	0.692042	18.796992	0.478921	13.008299	20.719570	0.048264	0.004936	0.000024	-0.005083	0.000026
23	1.535	0.0511	0.651466	19.569472	0.424408	12.748841	21.104559	0.047383	0.003717	0.000014	-0.007183	0.000052
24	1.570	0.0490	0.636943	20.408163	0.405696	12.998830	21.242358	0.047076	0.001924	0.000004	-0.009283	0.000086
25	1.630	0.0532	0.613497	18.796992	0.376378	11.531897	21.464815	0.046588	0.006612	0.000044	-0.005083	0.000026
26	1.700	0.0443	0.588235	22.573363	0.346021	13.278449	21.704502	0.046073	-0.001773	0.000003	-0.013983	0.000196
27	1.755	0.0426	0.569801	23.474178	0.324673	13.375600	21.879414	0.045705	-0.003105	0.000010	-0.015683	0.000246
28	1.810	0.0347	0.552486	28.818444	0.305241	15.921792	22.043696	0.045364	-0.010664	0.000114	-0.023583	0.000556
29	1.880	0.0336	0.531915	29.761905	0.282933	15.830800	22.238881	0.044966	-0.011366	0.000129	-0.024683	0.000609
30	2.010	0.0250	0.497512	40.000000	0.247519	19.900498	22.565297	0.044316	-0.019316	0.000373	-0.033283	0.001108
Sumas:	36.27	1.7485	27.98599	553.03739	29.63065	482.478633			Sr =	0.006492	St =	0.005626

Ymedia = 0.058283

a0 = 27.2858, a1 = -9.4882

******MODELO LINEALIZADO******
Z = 27.2858 -9.4882x

Desviación del estimado = 0.015226, Desviación estándar = 0.013928

Coeficiente de Determinación = -0.153914, Coeficiente de Correlación = -nan

Y = Ax/(B+x)
Tenemos que:
a0 = 1/A = 27.285787 ∴ A = 1/a0 = 1/27.285787 = 0.036649
a1 = B/A ∴ B = a1*A = a1*0.036649 = -0.347734

*****MODELO DE CRECIMIENTO*****
Y = +0.036649x / (-0.347734+x)

PRONÓSTICOS:
Xint[1] = 0.7000, Yint[1] = 0.0728
Xint[2] = 1.1000, Yint[2] = 0.0536
Xint[3] = 1.6000, Yint[3] = 0.0468
Xextinf = 0.2500, Yextinf = -0.0937
Xextsup = 2.5000, Yextsup = 0.0426

MODELO CÚBICO

Punto	X	Y	X²	X³	X⁴	X⁵	X⁶	XY	X²Y	X³Y	YY	Delta	Delta²	Beta	Beta²
1	0.54	0.0498	0.291600	0.157464	0.085031	0.045917	0.024795	0.026892	0.014522	0.007842	0.058690	-0.008890	0.000079	-0.008483	0.000072
2	0.61	0.0635	0.372100	0.226981	0.138458	0.084460	0.051520	0.038735	0.023628	0.014413	0.063304	0.000196	0.000000	0.005217	0.000027
3	0.65	0.0652	0.422500	0.274625	0.178506	0.116029	0.075419	0.042380	0.027547	0.017906	0.065404	-0.000204	0.000000	0.006917	0.000048
4	0.73	0.0788	0.532900	0.389017	0.283982	0.207307	0.151334	0.057524	0.041993	0.030655	0.068517	0.010283	0.000106	0.020517	0.000421
5	0.78	0.0788	0.608400	0.474552	0.370151	0.288717	0.225200	0.061464	0.047942	0.037395	0.069775	0.009025	0.000081	0.020517	0.000421
6	0.81	0.0640	0.656100	0.531441	0.430467	0.348678	0.282430	0.051840	0.041990	0.034012	0.070293	-0.006293	0.000040	0.005717	0.000033
7	0.86	0.0788	0.739600	0.636056	0.547008	0.470427	0.404567	0.067768	0.058280	0.050121	0.070786	0.008014	0.000064	0.020517	0.000421
8	0.88	0.0681	0.765625	0.669922	0.586182	0.512909	0.448795	0.059587	0.052139	0.045622	0.070848	-0.002748	0.000008	0.009817	0.000096
9	0.89	0.0703	0.792100	0.704969	0.627422	0.558406	0.496981	0.062567	0.055685	0.049559	0.070871	-0.000571	0.000000	0.012017	0.000144
10	0.95	0.0703	0.902500	0.857375	0.814506	0.773781	0.735092	0.066785	0.063446	0.060273	0.070601	-0.000301	0.000000	0.012017	0.000144
11	1.02	0.0681	1.040400	1.061208	1.082432	1.104081	1.126162	0.069462	0.070851	0.072268	0.069610	-0.001510	0.000002	0.009817	0.000096
12	1.03	0.0681	1.060900	1.092727	1.125509	1.159274	1.194052	0.070143	0.072247	0.074415	0.069414	-0.001314	0.000002	0.009817	0.000096
13	1.05	0.0790	1.113025	1.174241	1.238825	1.306960	1.378843	0.083345	0.087929	0.092765	0.068868	0.010132	0.000103	0.020717	0.000429
14	1.14	0.0575	1.288225	1.462135	1.659524	1.883559	2.137840	0.065263	0.074073	0.084073	0.066634	-0.009134	0.000083	-0.000783	0.000001
15	1.14	0.0681	1.299600	1.481544	1.688960	1.925415	2.194973	0.077634	0.088503	0.100893	0.066472	0.001628	0.000003	0.009817	0.000096
16	1.20	0.0633	1.440000	1.728000	2.073600	2.488320	2.985984	0.075960	0.091152	0.109382	0.064342	-0.001042	0.000001	0.005017	0.000025
17	1.25	0.0575	1.550025	1.929781	2.402578	2.991209	3.724055	0.071588	0.089126	0.110962	0.062547	-0.005047	0.000025	-0.000783	0.000001
18	1.28	0.0550	1.638400	2.097152	2.684355	3.435974	4.398047	0.070400	0.090112	0.115343	0.061051	-0.006051	0.000037	-0.003283	0.000011
19	1.32	0.0511	1.742400	2.299968	3.035958	4.007464	5.289853	0.067452	0.089037	0.117528	0.059251	-0.008151	0.000066	-0.007183	0.000052
20	1.39	0.0575	1.918225	2.656742	3.679587	5.096228	7.058276	0.079637	0.110298	0.152763	0.056164	0.001336	0.000002	-0.000783	0.000001
21	1.43	0.0490	2.044900	2.924207	4.181616	5.979711	8.550987	0.070070	0.100200	0.143286	0.053941	-0.004941	0.000024	-0.009283	0.000086
22	1.45	0.0532	2.088025	3.017196	4.359848	6.299981	9.103472	0.076874	0.111083	0.160515	0.053190	0.000010	0.000000	-0.005083	0.000026
23	1.53	0.0511	2.356225	3.616805	5.551796	8.522007	13.081281	0.078438	0.120403	0.184819	0.048637	0.002463	0.000006	-0.007183	0.000052
24	1.57	0.0490	2.464900	3.869893	6.075732	9.538899	14.976072	0.076930	0.120780	0.189625	0.046873	0.002127	0.000005	-0.009283	0.000086
25	1.63	0.0532	2.656900	4.330747	7.059118	11.506362	18.755370	0.086716	0.141347	0.230396	0.043905	0.009295	0.000086	-0.005083	0.000026
26	1.70	0.0443	2.890000	4.913000	8.352100	14.198570	24.137569	0.075310	0.128027	0.217646	0.040597	0.003703	0.000014	-0.013983	0.000196
27	1.75	0.0426	3.080025	5.405444	9.486554	16.648902	29.218823	0.074763	0.131209	0.230272	0.038174	0.004426	0.000020	-0.015683	0.000246
28	1.81	0.0347	3.276100	5.929741	10.732831	19.426424	35.161828	0.062807	0.113681	0.205762	0.035953	-0.001253	0.000002	-0.023583	0.000556
29	1.88	0.0336	3.534400	6.644672	12.491983	23.484929	44.151666	0.063168	0.118756	0.223261	0.033490	0.000110	0.000000	-0.024683	0.000609
30	2.01	0.0250	4.040100	8.120601	16.322408	32.808040	65.944161	0.050250	0.101002	0.203015	0.030296	-0.005296	0.000028	-0.033283	0.001108
Sumas:	36.27	1.7485	48.60620	70.67821	109.34703	177.21894	297.46545	1.98175	2.47699	3.36679		Sr =	0.000887	St =	0.005626

Ymedia = 0.058283

a0 = -0.0266, a1 = 0.2544, a2 = -0.2026, a3 = 0.0448

******MODELO CÚBICO******
Y = -0.0266 +0.2544x -0.2026x^2 +0.0448x^3

```
Desviación del estimado = 0.005627, Desviación estándar = 0.013928

Coeficiente de Determinación = 0.842405, Coeficiente de Correlación = 0.917826

PRONÓSTICOS:
Xint[1] = 0.700000, Yint[1] = 0.067514
Xint[2] = 1.100000, Yint[2] = 0.067698
Xint[3] = 1.600000, Yint[3] = 0.045377
Xextinf = 0.250000, Yextinf = 0.024980
Xextsup = 2.500000, Yextsup = 0.043766
```