

UNIVERSIDAD AUTONOMA METROPOLITANA
UNIDAD AZCAPOTZALCO

ASIGNATURA: METODOS NUMERICOS

PROFESOR: JOSE LUIS PANTOJA GALLEGOS

Lineal	10
Parabólico	10
exponencial	10
Potencias	10
crecimiento	10
gompelz	10
Problema Gráficas Conclusiones	3.33

Calif 9.04

EQUIPO: WAKANDA

INTEGRANTES:

HERNÁNDEZ MÁFARA SANTIAGO

INZUNZA SAMORA SAMANTHA DESIRÉE

NOLASCO VAZQUEZ DIEGO ARMANDO

* Faltan graficos y conclusiones con tus resultados obtenidos

TAREA: METODOS DE MINIMOS CUADRADOS

El plano inclinado.

Se dice que el movimiento de un cuerpo es rectilíneo y uniformemente acelerado cuando se mueve con velocidad constante y en línea recta.

Los cuerpos en caída por un plano inclinado están sometidos a la atracción de la Tierra y experimentan un movimiento uniformemente acelerado.

Esta aceleración aumenta con la inclinación del plano. Su valor máximo es igual a la aceleración de la gravedad $g = 9.8 \frac{m}{s^2}$ en caída libre (inclinación de 90°).

Para cualquier otro ángulo, el valor de la aceleración es:

$$a = g \sin \theta$$

La aceleración se determina midiendo los tiempos de paso por cada una de las marcas de espacio conocido, a partir de la ecuación:

$$e = V_0 t + \frac{1}{2} a t^2$$

El siguiente trabajo tiene como propósito englobar los ajustes de tipo parabólico, potencial, lineal, exponencial, de Gompertz y de crecimiento.

Por fines de practicidad, se utilizó un problema de Laboratorio de Física 1: Movimiento de una partícula, correspondiente a la práctica que estudia el movimiento de un cuerpo en un plano inclinado.

El presente programa realizará los ajustes obtenidos con la siguiente fórmula:

$$x = \frac{1}{2} g \sin \theta t^2$$

Archivo de Entrada: Entrada.input

35

0.64080025	3.19801767518056
0.81090025	4.058929465414
1.00120036	5.01146921830041
1.21154049	6.06431850699653
1.44192064	7.21747733150238
1.69234081	8.47094569181795
1.962801	9.82472358794323
2.25330121	11.2788110198782
2.56416169	12.8348109864644
2.89476196	14.4896176993454
3.24540225	16.244733948036
3.61646289	18.1020634594665
4.00720324	20.0578989891031
4.418404	22.1161483002417
4.84924441	24.2727031108242
5.30058529	26.5318722216709
5.77152576	28.8891463131994
6.26300676	31.3492352237542
6.77456784	33.9098341888809
7.306209	36.5709432085795
7.85793024	39.3325622828501
8.42973156	42.1946914116925
9.02161296	45.1573305951069
9.63357444	48.2204798330931
10.265616	51.3841391256513
10.91773764	54.6483084727813
11.58993936	58.0129878744833
12.28222116	61.4781773307572
12.99386209	65.0402681545935
13.72554304	68.7026685142396
14.478025	72.4691875153578
15.24980601	76.3323071559498
16.04162704	80.2957363323516
16.85430916	84.3635848783141
17.68623025	88.5277333356617

5

2.1

6.2

7.3

9.2

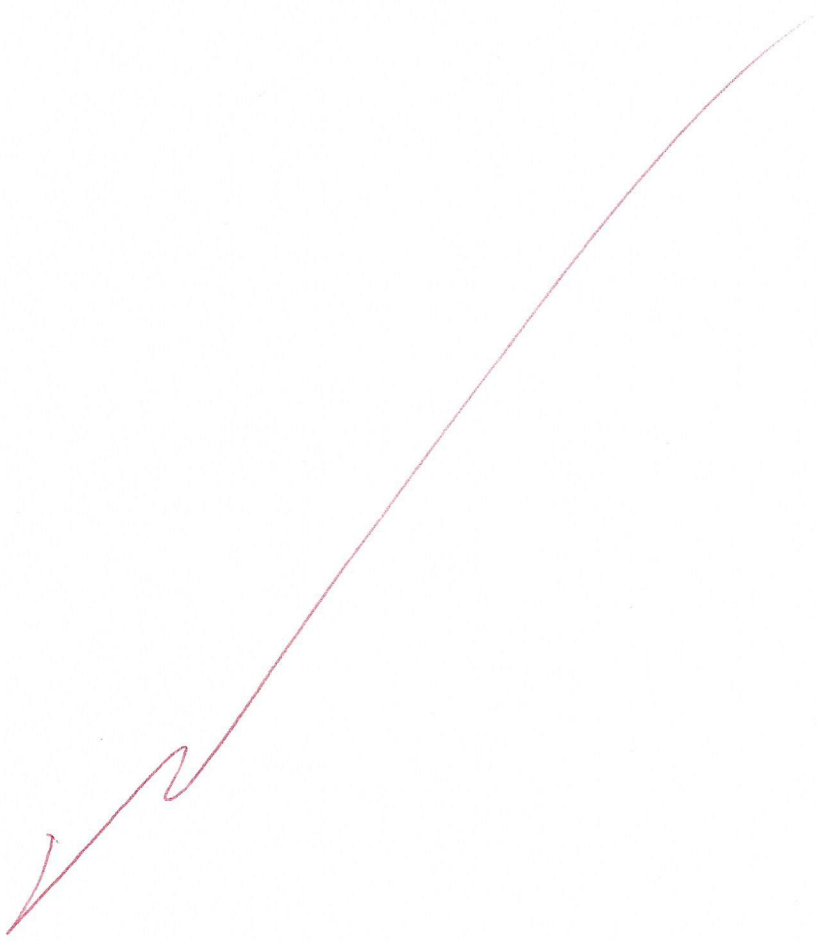
9.8

3

18

19

20



AE2.CD1.I1

Archivo de Salida: Salida.output

-----MODELO LINEAL [Y = a0 + (a1x)]-----

Pto.	x	y	x2	xy	YY	delta	delta2	beta	beta2
1	0.6408	3.1980	0.4106	2.0493	3.2068	-0.0088	0.0001	0.0000	1107.4124
2	0.8109	4.0589	0.6576	3.2914	4.0582	0.0007	0.0000	0.0000	1050.8551
3	1.0012	5.0115	1.0024	5.0175	5.0108	0.0007	0.0000	0.0000	990.0057
4	1.2115	6.0643	1.4678	7.3472	6.0636	0.0007	0.0000	0.0000	924.8597
5	1.4419	7.2175	2.0791	10.4070	7.2168	0.0006	0.0000	0.0000	856.0509
6	1.6923	8.4709	2.8640	14.3357	8.4703	0.0006	0.0000	0.0000	784.2732
7	1.9628	9.8247	3.8526	19.2840	9.8241	0.0006	0.0000	0.0000	710.2812
8	2.2533	11.2788	5.0774	25.4146	11.2782	0.0006	0.0000	0.0000	634.8895
9	2.5642	12.8348	6.5749	32.9105	12.8342	0.0006	0.0000	0.0000	558.8975
10	2.8948	14.4896	8.3796	41.9440	14.4891	0.0006	0.0000	0.0000	483.3933
11	3.2454	16.2447	10.5326	52.7207	16.2442	0.0005	0.0000	0.0000	409.2970
12	3.6165	18.1021	13.0788	65.4654	18.1015	0.0005	0.0000	0.0000	337.5951
13	4.0072	20.0579	16.0577	80.3761	20.0574	0.0005	0.0000	0.0000	269.5483
14	4.4184	22.1161	19.5223	97.7181	22.1157	0.0004	0.0000	0.0000	206.2003
15	4.8492	24.2727	23.5152	117.7043	24.2723	0.0004	0.0000	0.0000	148.9162
16	5.3006	26.5319	28.0962	140.6345	26.5315	0.0004	0.0000	0.0000	98.8822
17	5.7715	28.8891	33.3105	166.7345	28.8888	0.0004	0.0000	0.0000	57.5577
18	6.2630	31.3492	39.2253	196.3405	31.3489	0.0003	0.0000	0.0000	26.2819
19	6.7746	33.9098	45.8948	229.7245	33.9095	0.0003	0.0000	0.0000	6.5843
20	7.3062	36.5709	53.3807	267.1950	36.5707	0.0003	0.0000	0.0000	0.0090
21	7.8579	39.3326	61.7471	309.0725	39.3323	0.0002	0.0000	0.0000	8.1609
22	8.4297	42.1947	71.0604	355.6899	42.1945	0.0002	0.0000	0.0000	32.7054
23	9.0216	45.1573	81.3895	407.3920	45.1572	0.0002	0.0000	0.0000	75.3685
24	9.6336	48.2205	92.8058	464.5356	48.2204	0.0001	0.0000	0.0000	137.9369
25	10.2656	51.3841	105.3829	527.4898	51.3841	0.0001	0.0000	0.0000	222.2578
26	10.9177	54.6483	119.1970	596.6359	54.6483	0.0000	0.0000	0.0000	330.2392
27	11.5899	58.0130	134.3267	672.3670	58.0130	-0.0000	0.0000	0.0000	463.8494
28	12.2822	61.4782	150.8530	755.0886	61.4782	-0.0001	0.0000	0.0000	625.1176
29	12.9939	65.0403	168.8405	845.1243	65.0404	-0.0001	0.0000	0.0000	815.9274
30	13.7255	68.7027	188.3905	942.9814	68.7028	-0.0001	0.0000	0.0000	1038.5695
31	14.4780	72.4692	209.6132	1049.2107	72.4694	-0.0002	0.0000	0.0000	1295.5222
32	15.2498	76.3323	232.5566	1164.0529	76.3326	-0.0003	0.0000	0.0000	1588.5392
33	16.0416	80.2957	257.3338	1288.0743	80.2961	-0.0003	0.0000	0.0000	1920.1847
34	16.8543	84.3636	284.0677	1421.8899	84.3640	-0.0004	0.0000	0.0000	2293.2376
35	17.6862	88.5277	312.8027	1565.7219	88.5282	-0.0004	0.0000	0.0000	2709.4012

Ymedia: 36.475825

a0: -0.00076, a1: 5.00553

Modelo lineal: Y = -0.00076 + (-5.00553x)

-----Calculo de Y internas-----

- X = 2.100000; Y = 10.510846
- X = 6.200000; Y = 31.033515
- X = 7.300000; Y = 36.539596
- X = 9.200000; Y = 46.050101
- X = 9.800000; Y = 49.053418

-----Calculo de Y externas-----

- X = 18.000000; Y = 90.098755
- X = 19.000000; Y = 95.104283
- X = 20.000000; Y = 100.109812

-----DESVIACIONES Y COEFICIENTES-----

St: 23218.808195, Sr: 0.000083

Desviacion del Estimado: 0.001587; Desviacion Estandar: 26.132473

Coefficiente Determinado: 1.000000; Coeficiente Correl.: 1.000000

AE2.CD1.11

-----MODELO PARABOLICO [Y=a0+(a1x)+(a2x**2)]-----

Pto.	x	y	xy	x2	x3	x4	x2y	YY	delta	delta2	beta	beta2
1	0.6408	3.1980	2.0493	0.4106	0.2631	0.1686	0.1686	0.1686	1.3132	1.3132	0.0001	-33.2778
2	0.8109	4.0589	3.2914	0.6576	0.5332	0.4324	0.4324	0.4324	2.6690	2.6690	0.0000	-32.4169
3	1.0012	5.0115	5.0175	1.0024	1.0036	1.0048	1.0048	1.0048	5.0235	5.0235	0.0000	-31.4644
4	1.2115	6.0643	7.3472	1.4678	1.7783	2.1545	2.1545	2.1545	8.9014	8.9014	0.0000	-30.4115
5	1.4419	7.2175	10.4070	2.0791	2.9979	4.3228	4.3228	4.3228	15.0061	15.0061	0.0000	-29.2583
6	1.6923	8.4709	14.3357	2.8640	4.8469	8.2026	8.2026	8.2026	24.2609	24.2609	0.0000	-28.0049
7	1.9628	9.8247	19.2840	3.8526	7.5619	14.8424	14.8424	14.8424	37.8506	37.8506	0.0000	-26.6511
8	2.2533	11.2788	25.4146	5.0774	11.4408	25.7796	25.7796	25.7796	57.2667	57.2667	0.0000	-25.1970
9	2.5642	12.8348	32.9105	6.5749	16.8592	43.2296	43.2296	43.2296	84.3879	84.3879	0.0000	-23.6410
10	2.8948	14.4896	41.9440	8.3796	24.2571	70.2185	70.2185	70.2185	121.4179	121.4179	0.0000	-21.9862
11	3.2454	16.2447	52.7207	10.5326	34.1826	110.9364	110.9364	110.9364	171.0999	171.0999	0.0000	-20.2311
12	3.6165	18.1021	65.4654	13.0788	47.2990	171.0551	171.0551	171.0551	236.7533	236.7533	0.0000	-18.3738
13	4.0072	20.0579	80.3761	16.0577	64.3464	257.8490	257.8490	257.8490	322.0833	322.0833	0.0000	-16.4179
14	4.4184	22.1161	97.7181	19.5223	86.2574	381.1200	381.1200	381.1200	431.7579	431.7579	0.0000	-14.3597
15	4.8492	24.2727	117.7043	23.5152	114.0308	552.9633	552.9633	552.9633	570.7768	570.7768	0.0000	-12.2031
16	5.3006	26.5319	140.6345	28.0962	148.9263	789.3967	789.3967	789.3967	745.4449	745.4449	0.0000	-9.9440
17	5.7715	28.8891	166.7345	33.3105	192.2525	1109.5900	1109.5900	1109.5900	962.3122	962.3122	0.0000	-7.5867
18	6.2630	31.3492	196.3405	39.2253	245.6680	1538.6205	1538.6205	1538.6205	1229.6817	1229.6817	0.0000	-5.1266
19	6.7746	33.9098	229.7245	45.8948	310.9172	2106.3299	2106.3299	2106.3299	1556.2840	1556.2840	0.0000	-2.5660
20	7.3062	36.5709	267.1950	53.3807	390.0105	2849.4981	2849.4981	2849.4981	1952.1822	1952.1822	0.0000	0.0951
21	7.8579	39.3326	309.0725	61.7471	485.2042	3812.7004	3812.7004	3812.7004	2428.6704	2428.6704	0.0000	2.8567
22	8.4297	42.1947	355.6899	71.0604	599.0199	5049.5768	5049.5768	5049.5768	2998.3706	2998.3706	0.0000	5.7189
23	9.0216	45.1573	407.3920	81.3895	734.2646	6624.2508	6624.2508	6624.2508	3675.3326	3675.3326	0.0000	8.6815
24	9.6336	48.2205	464.5356	92.8058	894.0512	8612.9084	8612.9084	8612.9084	4475.1381	4475.1381	0.0000	11.7447
25	10.2656	51.3841	527.4898	105.3829	1081.8201	11105.5497	11105.5497	11105.5497	5415.0081	5415.0081	0.0000	14.9083
26	10.9177	54.6483	596.6359	119.1970	1301.3615	14207.9237	14207.9237	14207.9237	6513.9142	6513.9142	0.0000	18.1725
27	11.5899	58.0130	672.3670	134.3267	1556.8382	18043.6608	18043.6608	18043.6608	7792.6929	7792.6929	0.0000	21.5372
28	12.2822	61.4782	755.0886	150.8530	1852.8094	22756.6145	22756.6145	22756.6145	9274.1648	9274.1648	0.0000	25.0024

29	12.9939	65.0403	845.1243	168.8405	2193.8895	28507.0982	10981.4283	65.0405	-0.0002	0.0000	28.5644	815.9274
30	13.7255	68.7027	942.9814	188.3905	2585.7624	35490.9925	12942.9323	68.7028	-0.0002	0.0000	32.2268	1038.5695
31	14.4780	72.4692	1049.2107	209.6132	3034.7853	43937.6969	15190.4989	72.4693	-0.0001	0.0000	35.9934	1295.5222
32	15.2498	76.3323	1164.0529	232.5566	3546.4428	54082.5645	17751.5806	76.3323	0.0000	0.0000	39.8565	1588.5392
33	16.0416	80.2957	1288.0743	257.3338	4128.0528	66220.6836	20662.8068	80.2956	0.0002	0.0000	43.8199	1920.1847
34	16.8543	84.3636	1421.8899	284.0677	4787.7655	80694.4794	23964.9727	84.3632	0.0003	0.0000	47.8878	2293.2376
35	17.6862	88.5277	1565.7219	312.8027	5532.3013	97845.5544	27691.7176	88.5272	0.0005	0.0000	52.0519	2709.4012

Ymedia: 36.475825

a0: -0.00152, a1: 5.00582, a2: -0.00002

Modelo parabolico: $Y = -0.00152x + (5.00582x) + (-0.00002x^2)$

-----DESVIACIONES Y COEFICIENTES-----

St: 23218.808195, Sr: 0.000078

AE2.CD1.11

Desviacion del Estimado: 0.001534;

Desviacion Estandar: 26.132473

Coefficiente Determinado: 1.000000;

Coefficiente Correl.: 1.000000

-----Calculo de Y internas-----

X = 2.100000; Y = 10.510623
 X = 6.200000; Y = 31.033898
 X = 7.300000; Y = 36.540045
 X = 9.200000; Y = 46.050565
 X = 9.800000; Y = 49.053861

-----Calculo de Y externas-----

X = 18.000000; Y = 90.097682
 X = 19.000000; Y = 95.102869
 X = 20.000000; Y = 100.108021

-----MODELO EXPONENCIAL [Y=Ae**(Bx)]-----

Pto.	x	y	z=lny	x ²	xz	Z	YY	delta	delta ²	beta	beta ²	deltaZ	
1	0.6408	3.1980	1.1625	0.4106	0.4106	2.0493	2.1356	2.1356	8.4620	-5.2640	27.7098	-33.2778	1107.4124
2	0.8109	4.0589	1.4009	0.6576	3.2914	3.2914	2.1641	2.1641	8.7068	-4.6479	21.6031	-32.4169	1050.8551
3	1.0012	5.0115	1.6117	1.0024	5.0175	5.0175	2.1960	2.1960	8.9891	-3.9777	15.8218	-31.4644	990.0057
4	1.2115	6.0643	1.8024	1.4678	7.3472	7.3472	2.2313	2.2313	9.3118	-3.2475	10.5462	-30.4115	924.8597
5	1.4419	7.2175	1.9765	2.0791	10.4070	10.4070	2.2699	2.2699	9.6785	-2.4611	6.0568	-29.2583	856.0509
6	1.6923	8.4709	2.1366	2.8640	14.3357	14.3357	2.3119	2.3119	10.0936	-1.6226	2.6329	-28.0049	784.2732
7	1.9628	9.8247	2.2849	3.8526	19.2840	19.2840	2.3572	2.3572	10.5618	-0.7371	0.5433	-26.6511	710.2812
8	2.2533	11.2788	2.4229	5.0774	25.4146	25.4146	2.4060	2.4060	11.0890	0.1898	0.0360	-25.1970	634.8895
9	2.5642	12.8348	2.5522	6.5749	32.9105	32.9105	2.4581	2.4581	11.6823	1.1525	1.3283	-23.6410	558.8975
10	2.8948	14.4896	2.6734	8.3796	41.9440	41.9440	2.5135	2.5135	12.3481	2.1415	4.5860	-21.9862	483.3933
11	3.2454	16.2447	2.7878	10.5326	52.7207	52.7207	2.5723	2.5723	13.0958	3.1489	9.9155	-20.2311	409.2970

12	3.6165	18.1021	2.8960	13.0788	65.4654	2.6345	13.9365	4.1656	17.3522	-18.2738	337.5951
13	4.0072	20.0579	2.9986	16.0577	80.3761	2.7000	14.8801	5.1778	26.8098	-16.4179	269.5483
14	4.4184	22.1161	3.0963	19.5223	97.7181	2.7690	15.9422	6.1740	38.1179	-14.3597	206.2003
15	4.8492	24.2727	3.1894	23.5152	117.7043	2.8412	17.1364	7.1363	50.9266	-12.2031	148.9162
16	5.3006	26.5319	3.2783	28.0962	140.6345	2.9169	18.4835	8.0483	64.7756	-9.9440	98.8822
17	5.7715	28.8891	3.3635	33.3105	166.7345	2.9958	20.0022	8.8870	78.9779	-7.5867	57.5577
18	6.2630	31.3492	3.4452	39.2253	196.3405	3.0782	21.7203	9.6289	92.7165	-5.1266	26.2819
19	6.7746	33.9098	3.5237	45.8948	229.7245	3.1640	23.6655	10.2443	104.9461	-2.5660	6.5843
20	7.3062	36.5709	3.5993	53.3807	267.1950	3.2532	25.8719	10.6990	114.4695	0.0951	0.0090
21	7.8579	39.3326	3.6721	61.7471	309.0725	3.3457	28.3794	10.9532	119.9722	2.8567	8.1609
22	8.4297	42.1947	3.7423	71.0604	355.6899	3.4415	31.2349	10.9598	120.1177	5.7189	32.7054
23	9.0216	45.1573	3.8102	81.3895	407.3920	3.5408	34.4936	10.6637	113.7150	8.6815	75.3685
24	9.6336	48.2205	3.8758	92.8058	464.5356	3.6434	38.2208	9.9937	99.9937	11.7447	137.9369
25	10.2656	51.3841	3.9393	105.3829	527.4898	3.7494	42.4935	8.8906	79.0427	14.9083	222.2578
26	10.9177	54.6483	4.0009	119.1970	596.6359	3.8587	47.4033	7.2450	52.4905	18.1725	330.2392
27	11.5899	58.0130	4.0607	134.3267	672.3670	3.9714	53.0586	4.9544	24.5458	21.5372	463.8494
28	12.2822	61.4782	4.1187	150.8530	755.0886	4.0875	59.5889	1.8892	3.5692	25.0024	625.1176
29	12.9939	65.0403	4.1750	168.8405	845.1243	4.2068	67.1406	-2.1003	4.4113	28.5644	815.9274
30	13.7255	68.7027	4.2298	188.3905	942.9814	4.3295	75.9039	-7.2012	51.8570	32.2268	1038.5695
31	14.4780	72.4692	4.2832	209.6132	1049.2107	4.4556	86.1107	-13.6415	186.0910	35.9934	1295.5222
32	15.2498	76.3323	4.3351	232.5566	1164.0529	4.5850	98.0067	-21.6744	469.7795	39.8565	1588.5392
33	16.0416	80.2957	4.3857	257.3338	1288.0743	4.7178	111.9215	-31.6258	1000.1914	43.8199	1920.1847
34	16.8543	84.3636	4.4351	284.0677	1421.8899	4.8541	128.2598	-43.8962	1926.8787	47.8878	2293.2376
35	17.6862	88.5277	4.4833	312.8027	1565.7219	4.9935	147.4580	-58.9303	3472.7805	52.0519	2709.4012

Ymedia: 36.475825

a0: 2.02815, a1: 0.16767

Modelo lineal: $Z = 2.02815 + 0.16767x$

$A = e^{**}a0$ $A = 7.60000$

$B = a1$ $B = 0.16767$

Modelo exponencial: $Y = 7.60000e^{**}(0.16767x)$

-----DESVIACIONES Y COEFICIENTES-----

St: 23218.808195, Sr: 8415.308227

Desviacion del Estimado: 15.969012;

Desviacion Estandar: 26.132473

Coficiente Deierminado: 0.637565; Coficiente Correl.: 0.798477

-----Calculo de Y internas-----

X = 2.100000; Y = 10.807588

X = 6.200000; Y = 21.492045

X = 7.300000; Y = 25.844982

X = 9.200000; Y = 35.540880

X = 9.800000; Y = 39.302327

-----Calculo de Y externas-----

X = 18.000000; Y = 155.423311

X = 19.000000; Y = 183.794695

X = 20.000000; Y = 217.345067

-----MODELO DE POTENCIAS [Y=Ax**B]-----

Pro. x y w=logx z=logy wZ Z YY delta delta2 beta beta2

AE2.CD1.11

1	0.6408	3.1980	-0.1933	0.5049	0.0374	-0.0976	0.5059	3.2059	-0.0079	0.0001	-33.2778	1107.4124
2	0.8109	4.0589	-0.0910	0.6084	0.0083	-0.0554	0.6082	4.0571	0.0019	0.0000	-32.4169	1050.8551
3	1.0012	5.0115	0.0005	0.7000	0.0000	0.0004	0.6998	5.0094	0.0021	0.0000	-31.4644	990.0057
4	1.2115	6.0643	0.0833	0.7828	0.0069	0.0652	0.7826	6.0620	0.0023	0.0000	-30.4115	924.8597
5	1.4419	7.2175	0.1589	0.8584	0.0253	0.1364	0.8582	7.2150	0.0025	0.0000	-29.2583	856.0509
6	1.6923	8.4709	0.2285	0.9279	0.0522	0.2120	0.9278	8.4683	0.0026	0.0000	-28.0049	784.2732
7	1.9628	9.8247	0.2929	0.9923	0.0858	0.2906	0.9922	9.8220	0.0028	0.0000	-26.6511	710.2812
8	2.2533	11.2788	0.3528	1.0523	0.1245	0.3713	1.0522	11.2760	0.0028	0.0000	-25.1970	634.8895
9	2.5642	12.8348	0.4089	1.1084	0.1672	0.4533	1.1083	12.8319	0.0029	0.0000	-23.6410	558.8975
10	2.8948	14.4896	0.4616	1.1611	0.2131	0.5360	1.1610	14.4867	0.0029	0.0000	-21.9862	483.3933
11	3.2454	16.2447	0.5113	1.2107	0.2614	0.6190	1.2106	16.2418	0.0029	0.0000	-20.2311	409.2970
12	3.6165	18.1021	0.5583	1.2577	0.3117	0.7022	1.2577	18.0992	0.0028	0.0000	-18.3738	337.5951
13	4.0072	20.0579	0.6028	1.3023	0.3634	0.7851	1.3022	20.0552	0.0026	0.0000	-16.4179	269.5483
14	4.4184	22.1161	0.6453	1.3447	0.4164	0.8677	1.3447	22.1136	0.0024	0.0000	-14.3597	206.2003
15	4.8492	24.2727	0.6857	1.3851	0.4701	0.9497	1.3851	24.2704	0.0021	0.0000	-12.2031	148.9162
16	5.3006	26.5319	0.7243	1.4238	0.5246	1.0313	1.4237	26.5298	0.0021	0.0000	-9.9440	98.8822
17	5.7715	28.8891	0.7613	1.4607	0.5796	1.1120	1.4607	28.8874	0.0018	0.0000	-7.5867	57.5577
18	6.2630	31.3492	0.7968	1.4962	0.6349	1.1922	1.4962	31.3478	0.0014	0.0000	-5.1266	26.2819
19	6.7746	33.9098	0.8309	1.5303	0.6904	1.2715	1.5303	33.9089	0.0010	0.0000	-2.5660	6.5843
20	7.3062	36.5709	0.8637	1.5631	0.7460	1.3501	1.5631	36.5704	0.0005	0.0000	0.0951	0.0090
21	7.8579	39.3326	0.8953	1.5948	0.8016	1.4278	1.5948	39.3326	-0.0000	0.0000	2.8567	8.1609
22	8.4297	42.1947	0.9258	1.6253	0.8571	1.5047	1.6253	42.1953	-0.0007	0.0000	5.7189	32.7054
23	9.0216	45.1573	0.9553	1.6547	0.9126	1.5807	1.6547	45.1587	-0.0013	0.0000	8.6815	75.3685
24	9.6336	48.2205	0.9838	1.6832	0.9678	1.6559	1.6833	48.2225	-0.0021	0.0000	11.7447	137.9369
25	10.2656	51.3841	1.0114	1.7108	1.0229	1.7303	1.7109	51.3870	-0.0029	0.0000	14.9083	222.2578
26	10.9177	54.6483	1.0381	1.7376	1.0777	1.8038	1.7376	54.6520	-0.0037	0.0000	18.1725	330.2392
27	11.5899	58.0130	1.0641	1.7635	1.1323	1.8765	1.7636	58.0176	-0.0046	0.0000	21.5372	463.8494
28	12.2822	61.4782	1.0893	1.7887	1.1865	1.9484	1.7888	61.4838	-0.0057	0.0000	25.0024	625.1176
29	12.9939	65.0403	1.1137	1.8132	1.2404	2.0194	1.8132	65.0470	-0.0067	0.0000	28.5644	815.9274
30	13.7255	68.7027	1.1375	1.8370	1.2940	2.0896	1.8370	68.7105	-0.0079	0.0001	32.2268	1038.5695
31	14.4780	72.4692	1.1607	1.8602	1.3472	2.1591	1.8602	72.4783	-0.0091	0.0001	35.9934	1295.5222
32	15.2498	76.3323	1.1833	1.8827	1.4001	2.2277	1.8828	76.3427	-0.0104	0.0001	39.8565	1588.5392
33	16.0416	80.2957	1.2052	1.9047	1.4526	2.2956	1.9048	80.3075	-0.0117	0.0001	43.8199	1920.1847
34	16.8543	84.3636	1.2267	1.9262	1.5066	2.3628	1.9262	84.3768	-0.0132	0.0002	47.8878	2293.2376
35	17.6862	88.5277	1.2476	1.9471	1.5566	2.4292	1.9472	88.5424	-0.0147	0.0002	52.0519	2709.4012

Y mediar: 36.475825

a0: 0.69926, a1: 1.00020

Modelo lineal: $Z = 0.69926 + 1.00020x$

A=10**a0 A= 5.00337

B=a1 B= 1.00020

Modelo potencial: $Y = 5.00337x^{**}(1.00020)$

-----DESVIACIONES Y COEFICIENTES-----

St: 23248.808195, Sr: 0.001074

Desviacion del Estimado: 0.005705, Desviacion Estandar: 26.132473

Coefficiente Determinado: 1.000000, Coeficiente Correl.: 1.000000

-----Calculo de Y internas-----

X = 2.100000; Y = 10.512101

X = 6.200000; Y = 31.042555

X = 7.300000; Y = 36.551318

AE2.CD1.I1

X = 9.200000; Y = 46.066841
 X = 9.800000; Y = 49.071830

-----Calculo de Y extremas-----

X = 18.000000; Y = 90.143070
 X = 19.000000; Y = 95.152063
 X = 20.000000; Y = 100.161111

AE2.CD1.I1

-----MODELO DE CRECIMIENTO [Y=Ax/(B+x)]-----

Pro.	x	y	v=1/x	z=1/y	v2	vz	Z	Yy	delta	delta2	beta	beta2	
1	0.6408	3.1980	1.5609	0.3127	0.6408	0.3127	2.4353	0.4880	0.3121	3.2038	0.0000	-33.2778	1107.4124
2	0.8109	4.0589	1.2332	0.2464	0.8109	0.2464	1.5208	0.3038	0.2466	4.0545	0.0000	-32.4169	1050.8551
3	1.0012	5.0115	0.9988	0.1995	1.0012	0.1995	0.9976	0.1993	0.1998	5.0062	0.0000	-31.4644	990.0057
4	1.2115	6.0643	0.8254	0.1649	1.2115	0.1649	0.6813	0.1361	0.1651	6.0583	0.0000	-30.4115	924.8597
5	1.4419	7.2175	0.6935	0.1386	1.4419	0.1386	0.4810	0.0961	0.1387	7.2108	0.0000	-29.2583	856.0509
6	1.6923	8.4709	0.5909	0.1181	1.6923	0.1181	0.3492	0.0698	0.1182	8.4638	0.0001	-28.0049	784.2732
7	1.9628	9.8247	0.5095	0.1018	1.9628	0.1018	0.2596	0.0519	0.1019	9.8171	0.0001	-26.6511	710.2812
8	2.2533	11.2788	0.4438	0.0887	2.2533	0.0887	0.1970	0.0393	0.0887	11.2710	0.0001	-25.1970	634.8895
9	2.5642	12.8348	0.3900	0.0779	2.5642	0.0779	0.1521	0.0304	0.0780	12.8271	0.0001	-23.6410	558.8975
10	2.8948	14.4896	0.3455	0.0690	2.8948	0.0690	0.1193	0.0238	0.0691	14.8222	0.0000	-21.9862	483.3933
11	3.2454	16.2447	0.3081	0.0616	3.2454	0.0616	0.0949	0.0190	0.0616	16.2381	0.0000	-20.2311	409.2970
12	3.6165	18.1021	0.2765	0.0552	3.6165	0.0552	0.0765	0.0153	0.0553	18.0966	0.0000	-18.3738	337.5951
13	4.0072	20.0579	0.2496	0.0499	4.0072	0.0499	0.0623	0.0124	0.0499	20.0540	0.0000	-16.4179	269.5483
14	4.4184	22.1161	0.2263	0.0452	4.4184	0.0452	0.0512	0.0102	0.0452	22.1144	0.0000	-14.3597	206.2003
15	4.8492	24.2727	0.2062	0.0412	4.8492	0.0412	0.0425	0.0085	0.0412	24.2738	0.0000	-12.2031	148.9162
16	5.3006	26.5319	0.1887	0.0377	5.3006	0.0377	0.0356	0.0071	0.0377	26.5365	0.0001	-7.5867	57.5577
17	5.7715	28.8891	0.1733	0.0346	5.7715	0.0346	0.0300	0.0060	0.0346	28.8980	0.0002	-5.1266	26.2819
18	6.2630	31.3492	0.1597	0.0319	6.2630	0.0319	0.0255	0.0051	0.0319	31.3632	0.0004	-2.5660	6.5843
19	6.7746	33.9098	0.1476	0.0295	6.7746	0.0295	0.0218	0.0044	0.0295	33.9299	0.0007	0.0951	0.0090
20	7.3062	36.5709	0.1369	0.0273	7.3062	0.0273	0.0187	0.0037	0.0273	36.5980	0.0012	2.8567	8.1609
21	7.8579	39.3326	0.1273	0.0254	7.8579	0.0254	0.0162	0.0032	0.0254	39.3679	0.0020	5.7189	32.7054
22	8.4297	42.1947	0.1186	0.0237	8.4297	0.0237	0.0141	0.0028	0.0237	42.2394	0.0047	8.6815	75.3685
23	9.0216	45.1573	0.1108	0.0221	9.0216	0.0221	0.0123	0.0025	0.0221	45.2128	0.0066	11.7447	137.9369
24	9.6336	48.2205	0.1038	0.0207	9.6336	0.0207	0.0108	0.0022	0.0207	48.2881	0.0093	14.9083	222.2578
25	10.2656	51.3841	0.0974	0.0195	10.2656	0.0195	0.0095	0.0019	0.0194	51.4654	0.0129	18.1725	330.2392
26	10.9177	54.6483	0.0916	0.0183	10.9177	0.0183	0.0084	0.0017	0.0183	54.7448	0.0175	21.5372	463.8494
27	11.5899	58.0130	0.0863	0.0172	11.5899	0.0172	0.0074	0.0015	0.0172	58.1266	0.0235	25.0024	625.1176
28	12.2822	61.4782	0.0814	0.0163	12.2822	0.0163	0.0066	0.0013	0.0162	61.6106	0.0306	28.5644	815.9274
29	12.9939	65.0403	0.0770	0.0154	12.9939	0.0154	0.0059	0.0012	0.0153	65.1936	0.0406	32.2268	1038.5695
30	13.7255	68.7027	0.0729	0.0146	13.7255	0.0146	0.0053	0.0011	0.0145	68.8789	0.0524	35.9934	1295.5222
31	14.4780	72.4692	0.0691	0.0138	14.4780	0.0138	0.0048	0.0010	0.0138	72.6706	0.0670	39.8565	1588.5392
32	15.2498	76.3323	0.0656	0.0131	15.2498	0.0131	0.0043	0.0009	0.0131	76.5612	0.0850	43.8199	1920.1847
33	16.0416	80.2957	0.0623	0.0125	16.0416	0.0125	0.0039	0.0008	0.0124	80.5547	0.1069	47.8878	2293.2376
34	16.8543	84.3636	0.0593	0.0119	16.8543	0.0119	0.0035	0.0007	0.0118	84.6552	0.1370	52.0519	2709.4012
35	17.6862	88.5277	0.0565	0.0113	17.6862	0.0113	0.0032	0.0006	0.0113	88.8547	0.1699	57.5577	3300.0000

Ymedia: 36.475825

ao: -0.000006; ai: 0.200005

Modelo lineal: Z = -0.000006 + 0.200005x

A=1/a0 A=-17674.67826

B=a1*A B=-3535.77032

Modelo de crecimiento: Y = -17674.67826x / (-3535.77032 + x)

-----DESVIACIONES Y COEFICIENTES-----

St: 23218.808195, Sr: 0.465758

Desviación del Estimado: 0.118802; Desviación Estandar: 26.132473

Coefficiente Determinado: 0.999980; Coeficiente Correl.: 0.999990

-----Calculo de Y internas-----
 X = 2.100000; Y = 10.503760
 X = 6.200000; Y = 31.047123
 X = 7.300000; Y = 36.566880
 X = 9.200000; Y = 46.109116
 X = 9.800000; Y = 49.124590

-----Calculo de Y extremas-----
 X = 18.000000; Y = 90.439164
 X = 19.000000; Y = 95.490708
 X = 20.000000; Y = 100.545125

-----MODELO DE GOWPETZ [Y=AB**(e**x)]-----

Pto.	x	y	u=e**x	u2	z=lny	uz	Z	YY	delta	dellaa2	beta	beta2
1	0.6408	3.1980	1.8980	1.1625	1.0980	3.6024	2.2065	3.1542	23.4337	409.4848	-33.2778	1107.4124
2	0.8109	4.0589	2.2499	1.4009	1.6117	5.0622	3.1520	23.4337	-19.3748	375.3836	-32.4169	1050.8551
3	1.0012	5.0115	2.7215	1.6117	2.3086	7.4068	3.1542	23.4337	-18.4223	339.3804	-31.4644	990.0057
4	1.2115	6.0643	3.3587	1.8024	3.1542	11.2806	6.0537	23.4337	-17.3694	301.6971	-30.4115	924.8597
5	1.4419	7.2175	4.2288	1.9765	4.3337	17.8828	8.3583	23.4337	-16.2163	262.9675	-29.2583	856.0509
6	1.6923	8.4709	5.4322	2.1366	5.9986	29.5086	11.6066	23.4338	-14.9628	223.8855	-28.0049	784.2732
7	1.9628	9.8247	7.1192	2.2849	8.0134	50.6836	16.2668	23.4338	-13.6090	185.2057	-26.6511	710.2812
8	2.2533	11.2788	9.5191	2.4229	10.6134	90.6134	23.0641	23.4338	-12.1549	147.7427	-25.1970	634.8895
9	2.5642	12.8348	12.9898	2.5522	13.7340	168.7340	33.1520	23.4338	-10.5989	112.3377	-23.6410	558.8975
10	2.8948	14.4896	18.0792	2.6734	18.8335	326.8574	48.3335	23.4338	-8.9441	79.9977	-21.9862	483.3933
11	3.2454	16.2447	25.6720	2.7878	26.90534	659.0534	71.5677	23.4338	-7.1890	51.6822	-20.2311	409.2970
12	3.6165	18.1021	37.2057	2.8960	38.42666	1384.2666	107.7488	23.4338	-5.3317	28.4272	-18.3738	337.5951
13	4.0072	20.0579	54.9929	2.9986	52.9929	3024.2139	164.9028	23.4338	-3.3759	11.3967	-16.4179	269.5483
14	4.4184	22.1161	82.9638	3.0963	68.82.9870	6882.9870	256.8814	23.4338	-1.3177	1.7363	-14.3597	206.2003
15	4.8492	24.2727	127.6439	3.1894	92.92.9670	16292.9670	407.1014	3.1542	23.4339	0.8388	0.7037	-12.2031
16	5.3006	26.5319	200.4541	3.2783	127.6439	40181.8460	657.1580	3.1542	23.4339	3.0979	9.5973	-9.9440
17	5.7715	28.8891	321.0272	3.3635	174.4616	103058.4436	1079.7640	3.1542	23.4340	5.4551	29.7582	-7.5867
18	6.2630	31.3492	524.7945	3.4452	242.7945	275409.2687	1808.0167	3.1542	23.4342	7.9150	62.6474	-5.1266
19	6.7746	33.9098	875.3010	3.5237	333.9098	766151.8602	3084.3026	3.1542	23.4345	10.4753	109.7317	-2.5660
20	7.3062	36.5709	1489.5197	3.5993	457.5197	2218668.9456	5361.1598	3.1542	23.4351	13.1358	172.5502	0.0951
21	7.8579	39.3326	2586.1621	3.6721	633.3326	6688234.3934	9496.5236	3.1543	23.4361	15.8965	252.6973	2.8567
22	8.4297	42.1947	4581.2701	3.7423	871.2701	20988036.0199	17144.4616	3.1544	23.4379	18.7568	351.8161	5.7189
23	9.0216	45.1573	8280.1218	3.8102	1180.1218	68560041.3772	31548.5279	3.1545	23.4413	21.7160	471.5856	8.6815
24	9.6336	48.2205	15268.9165	3.8758	1600.9165	233139811.2835	59179.0196	3.1548	23.4477	24.7728	613.6910	11.7447
25	10.2656	51.3841	28777.6686	3.9393	2187.6686	825278943.0999	113167.7537	3.1553	23.4600	27.9241	779.7580	14.9083
26	10.9177	54.6483	55145.8978	4.0009	3041.070048.2905	3041070048.2905	220634.2297	3.1563	23.4842	31.1642	971.2045	18.1725
27	11.5899	58.0130	108005.7084	4.0607	4116.7084	11665233040.8008	438575.2065	3.1584	23.5326	34.4804	1188.8989	21.5372
28	12.2822	61.4782	215824.5655	4.1187	5582.5655	46380243082.9009	888912.8118	3.1626	23.6316	37.8465	1432.3597	25.0024
29	12.9939	65.0403	439706.2151	4.1750	7662.2151	193341555623.2102	1835776.3450	3.1713	23.8387	41.2016	1697.5688	28.5644
30	13.7255	68.7021	913960.8721	4.2298	10590.8721	83532447527.5637	3865860.7672	3.1898	24.2833	44.4193	1973.0775	32.2268
31	14.4780	72.4692	1939663.3790	4.2832	14590.3790	3762294023799.9150	8307891.4530	3.2298	25.2735	47.1957	2227.4308	35.9934
32	15.2498	76.3323	4196687.1995	4.3351	20000.1995	176121834506681.3945	181930403.0310	3.3177	27.5969	48.7354	2375.1394	39.8565
33	16.0416	80.2957	9263819.9208	4.3857	27500.9208	85818359524162.4219	40628488.0908	3.5151	33.6209	46.6748	2759.5199	43.8199
34	16.8543	84.3636	20880142.7993	4.4351	37500.7993	435980363317188.1250	92606269.8760	5.0236	3.9678	52.8673	31.4963	992.0166
35	17.6862	88.5277	47976902.2533	4.4833	50000.2533	2301783149820088.5000	215095607.4549	5.0236	5.0236	151.9614	-63.4337	4023.8349

AE2.CD1.11

Ymedia: 36.475825

a0: 3.15418; a1: 0.00000

Modelo lineal: $Z = 3.15418 + 0.00000x$

$A = e^{**}a0$ $A = 23.43375$
 $B = e^{**}a1$ $B = 1.00000$

Modelo de Gompertz: $Y = 23.43375^{**}(e^{**}x)$

-----DESVIACIONES Y COEFICIENTES-----

St: 23218.808195, Sr: 24445.933883

Desviacion del Estimado: 27.217382;

Desviacion Estandar: 26.132473

Coefficiente Determinado: -0.052851;

Coefficiente Correl.: -nan

-----Calculo de Y internas-----

$X = 2.100000$; $Y = 153589083260.458405$

$X = 6.200000$; $Y = \text{inf}$

$X = 7.300000$; $Y = \text{inf}$

$X = 9.200000$; $Y = \text{inf}$

$X = 9.800000$; $Y = \text{inf}$

-----Calculo de Y externas-----

$X = 18.000000$; $Y = \text{inf}$

$X = 19.000000$; $Y = \text{inf}$

$X = 20.000000$; $Y = \text{inf}$

chequear

Main.c

```
#include "mmc.h"

int main()
{
//LEE LOS ELEMENTOS CONTENIDOS DENTRO DEL ARCHIVO Y LOS ALMACENA EN MEMORIA
FILE *input = fopen("Entrada.input", "r");
int n_elementos;
fscanf(input, "%d", &n_elementos);
double *x = (double *)malloc(n_elementos * sizeof(double));
double *y = (double *)malloc(n_elementos * sizeof(double));
for (int i = 0; i < n_elementos; i++)
{
fscanf(input, "%lf %lf", &x[i], &y[i]);
}
int n_xInt;
fscanf(input, "%d", &n_xInt);
double *xInt = (double *)malloc(n_xInt * sizeof(double));
for (int i = 0; i < n_xInt; i++)
{
fscanf(input, "%lf", &xInt[i]);
}
int n_xExt;
fscanf(input, "%d", &n_xExt);
double *xExt = (double *)malloc(n_xExt * sizeof(double));
for (int i = 0; i < n_xExt; i++)
{
fscanf(input, "%lf", &xExt[i]);
}
fclose(input);

//LLAMA LA LIBRERIA DE MMC
mmc(n_elementos, x, y, n_xInt, xInt, n_xExt, xExt);

free(x);
free(y);
return 0;
}
```

```
//mmc.h
```

```
#ifndef __MMC_H__  
#define __MMC_H__
```

```
#include <stdio.h>  
#include "ajuste_lineal.h"  
#include "ajuste_parabolico.h"  
#include "ajuste_exponencial.h"  
#include "ajuste_potencias.h"  
#include "ajuste_crecimiento.h"  
#include "ajuste_gowpetz.h"
```

```
void mmc(int n_elementos, double *x, double *y, int n_xInt, double *xInt, int  
n_xExt, double *xExt);
```

```
#endif
```

```
//mmc.c
```

```
#include "mmc.h"
```

```
void mmc(int n_elementos, double *x, double *y, int n_xInt, double *xInt, int  
n_xExt, double *xExt)
```

```
{
```

```
//Inicializa un archivo universal a ser usado por todos los metodos
```

```
FILE *foutput = fopen("Salida.output", "w+");
```

```
lineal(n_elementos, x, y, n_xInt, xInt, n_xExt, xExt, &foutput);
```

```
parabolico(n_elementos, x, y, n_xInt, xInt, n_xExt, xExt, &foutput);
```

```
exponencial(n_elementos, x, y, n_xInt, xInt, n_xExt, xExt, &foutput);
```

```
potencias(n_elementos, x, y, n_xInt, xInt, n_xExt, xExt, &foutput);
```

```
crecimiento(n_elementos, x, y, n_xInt, xInt, n_xExt, xExt, &foutput);
```

```
gowpetz(n_elementos, x, y, n_xInt, xInt, n_xExt, xExt, &foutput);
```

```
fclose(foutput);
```

```
free(x);
```

```
free(y);
```

```
free(xInt);
```

```
free(xExt);
```

```
}
```

AE2.CD1.11

```

//ajuste_lineal.h
#ifndef __LINEAL_H__
#define __LINEAL_H__

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "gauss_jordan.h"
#include "error_info.h"

void lineal(int n_elementos, double *x, double *y, int n_xInt, double *x_Int,
int n_xExt, double *x_Ext, FILE **foutput);

#endif

```

```

//ajuste_lineal.c
#include "ajuste_lineal.h"

void lineal(int n_elementos, double *x, double *y, int n_xInt, double *x_Int,
int n_xExt, double *x_Ext, FILE **foutput)
{
    fprintf(*foutput, "-----MODELO LINEAL [Y = a0 +
(a1x)]-----\n\n");
    double Sumx = 0;
    double Sumx2 = 0;
    double Sumy = 0;
    double Sumxy = 0;
    double *x2 = (double *)malloc(n_elementos * sizeof(double));
    double *xy = (double *)malloc(n_elementos * sizeof(double));
    double *YY = (double *)malloc(n_elementos * sizeof(double));
    double *delta = (double *)malloc(n_elementos * sizeof(double));
    double *delta2 = (double *)malloc(n_elementos * sizeof(double));
    double *beta = (double *)malloc(n_elementos * sizeof(double));
    double *beta2 = (double *)malloc(n_elementos * sizeof(double));

    for (int i = 0; i < n_elementos; i++)
    {
        x2[i] = pow(x[i], 2);
        xy[i] = x[i] * y[i];

        Sumx = Sumx + x[i];
        Sumy = Sumy + y[i];
        Sumx2 = Sumx2 + x2[i];
        Sumxy = Sumxy + xy[i];
    }
    double Ymedia = Sumy / n_elementos;

```

AE2.CD1.I1

```
Matrix a = generate_matrix(2, 3);
a.rows[0].columns[0] = n_elementos;
a.rows[0].columns[1] = Sumx;
a.rows[0].columns[2] = Sumy;
a.rows[1].columns[0] = Sumx;
a.rows[1].columns[1] = Sumx2;
a.rows[1].columns[2] = Sumxy;
double a0, a1;
GaussJordan(2, &a, &a0, &a1);
double St = 0, Sr = 0;
```

AE2.CD1.I1

```
for (int i = 0; i < n_elementos; i++)
{
YY[i] = a0 + (a1 * x[i]);
delta[i] = y[i] - YY[i];
delta2[i] = pow(delta[i], 2);
beta[i] = y[i] - Ymedia;
beta2[i] = pow(beta[i], 2);
Sr = Sr + delta2[i];
St = St + beta2[i];
}
printf("a0: %lf, a1:%lf \n", a0, a1);
printf("Sr: %lf, St:%lf \n", Sr, St);
fprintf(*foutput, "Pto. x y x2 xy YY delta delta2 beta beta2\n");
for (int i = 0; i < n_elementos; i++)
{
fprintf(*foutput, "%d \t %9.4lf \t %9.4lf \t %9.4lf \t %9.4lf \t %9.4lf \t
%9.4lf \t %9.4lf \t %9.4lf \t %9.4lf \n", i + 1, x[i], y[i], x2[i], x[i] *
y[i], YY[i], delta[i], delta2[i], beta[i], beta2[i]);
}
fprintf(*foutput, "\nYmedia: %lf \n", Ymedia);
fprintf(*foutput, "\na0: %9.5lf, a1:%9.5lf\n", a0, a1);
fprintf(*foutput, "\nModelo lineal: Y = %9.5lf + (%9.5lfx)\n", a0, a1);

fprintf(*foutput, "\n-----Calculo de Y internas-----\n");
double resultado_Y;
for (int i = 0; i < n_xInt; i++)
{
resultado_Y = a0 + (a1 * x_Int[i]);
fprintf(*foutput, "X = %lf; Y = %lf\n", x_Int[i], resultado_Y);
}

fprintf(*foutput, "\n-----Calculo de Y externas-----\n");
for (int i = 0; i < n_xExt; i++)
{
resultado_Y = a0 + (a1 * x_Ext[i]);
fprintf(*foutput, "X = %lf; Y = %lf\n", x_Ext[i], resultado_Y);
}
```

AE2.CD1.I1

```
}  
fprintf(*foutput, "\n");  
  
print_error_info(n_elementos, St, Sr, &foutput);  
  
free(x2);  
free(xy);  
free(YY);  
free(delta);  
free(delta2);  
free(beta);  
free(beta2);  
}
```



```

//ajuste_parabolico.h
#ifndef __PARABOLICO_H__
#define __PARABOLICO_H__

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "gauss_jordan.h"
#include "error_info.h"

void parabolico(int n_elementos, double *x, double *y, int n_xInt, double
*x_Int, int n_xExt, double *x_Ext, FILE **foutput);

#endif

```

```

//ajuste_parabolico.c
#include "ajuste_parabolico.h"

void parabolico(int n_elementos, double *x, double *y, int n_xInt, double
*x_Int, int n_xExt, double *x_Ext, FILE **foutput)
{
    fprintf(*foutput, "-----MODELO PARABOLICO [Y=a0+(a1x)+
(a2x**2)]-----\n\n");
    double Sumx = 0;
    double Sumx2 = 0;
    double Sumx3 = 0;
    double Sumx4 = 0;
    double Sumy = 0;
    double Sumxy = 0;
    double Sumx2y = 0;
    double *x2 = (double *)malloc(n_elementos * sizeof(double));
    double *x3 = (double *)malloc(n_elementos * sizeof(double));
    double *x4 = (double *)malloc(n_elementos * sizeof(double));
    double *xy = (double *)malloc(n_elementos * sizeof(double));
    double *x2y = (double *)malloc(n_elementos * sizeof(double));

    double *YY = (double *)malloc(n_elementos * sizeof(double));
    double *delta = (double *)malloc(n_elementos * sizeof(double));
    double *delta2 = (double *)malloc(n_elementos * sizeof(double));
    double *beta = (double *)malloc(n_elementos * sizeof(double));
    double *beta2 = (double *)malloc(n_elementos * sizeof(double));

    for (int i = 0; i < n_elementos; i++)
    {
        x2[i] = pow(x[i], 2);
        x3[i] = pow(x[i], 3);
        x4[i] = pow(x[i], 4);
    }
}

```

AE2.CD1.11

```

xy[i] = x[i] * y[i];
x2y[i] = x2[i] * y[i];

Sumx = Sumx + x[i];
Sumx2 = Sumx2 + x2[i];
Sumx3 = Sumx3 + x3[i];
Sumx4 = Sumx4 + x4[i];
Sumy = Sumy + y[i];
Sumxy = Sumxy + xy[i];
Sumx2y = Sumx2y + x2y[i];
}
double Ymedia = Sumy / n_elementos;

Matrix a = generate_matrix(3, 4);
a.rows[0].columns[0] = n_elementos;
a.rows[0].columns[1] = Sumx;
a.rows[0].columns[2] = Sumx2;
a.rows[0].columns[3] = Sumy;
a.rows[1].columns[0] = Sumx;
a.rows[1].columns[1] = Sumx2;
a.rows[1].columns[2] = Sumx3;
a.rows[1].columns[3] = Sumxy;
a.rows[2].columns[0] = Sumx2;
a.rows[2].columns[1] = Sumx3;
a.rows[2].columns[2] = Sumx4;
a.rows[2].columns[3] = Sumx2y;
double a0, a1, a2;
GaussJordan_vParabolica(3, &a, &a0, &a1, &a2);
double St = 0, Sr = 0;

for (int i = 0; i < n_elementos; i++)
{
YY[i] = a0 + (a1 * x[i]) + (a2 * pow(x[i], 2));
delta[i] = y[i] - YY[i];
delta2[i] = pow(delta[i], 2);
beta[i] = y[i] - Ymedia;
beta2[i] = pow(beta[i], 2);
Sr = Sr + delta2[i];
St = St + beta2[i];
}
printf("a0: %lf, a1:%lf \n", a0, a1);
printf("Sr: %lf, St:%lf \n", Sr, St);
fprintf(*foutput, "Pto. x y xy x2 x3 x4 x2y YY delta delta2 beta beta2\n");
for (int i = 0; i < n_elementos; i++)
{
fprintf(*foutput, "%2d %8.4lf %8.4lf %13.4lf %13.4lf %13.4lf \t %13.4lf \t
%13.4lf \t %13.4lf %9.4lf %9.4lf %9.4lf %9.4lf\n", i + 1, x[i], y[i], x[i] *

```

AE2.CD1.I1

```

y[i], x2[i], x3[i], x4[i], x2y[i], YY[i], delta[i], delta2[i], beta[i],
beta2[i]);
}
fprintf(*foutput, "\nYmedia: %lf \n", Ymedia);
fprintf(*foutput, "\na0: %9.5lf, a1:%9.5lf, a2:%9.5lf\n", a0, a1, a2);
fprintf(*foutput, "\nModelo parabolico: Y = %9.5lf + (%9.5lfx) + (%9.5lfx**2)\n", a0, a1, a2);

print_error_info(n_elementos, St, Sr, &foutput);

double resultado_Y;
fprintf(*foutput, "\n-----Calculo de Y internas-----\n");
for (int i = 0; i < n_xInt; i++)
{
resultado_Y = a0 + (a1 * x_Int[i]) + (a2 * pow(x_Int[i], 2));
fprintf(*foutput, "X = %lf; Y = %lf\n", x_Int[i], resultado_Y);
}

fprintf(*foutput, "\n-----Calculo de Y externas-----\n");
for (int i = 0; i < n_xExt; i++)
{
resultado_Y = a0 + (a1 * x_Ext[i]) + (a2 * pow(x_Ext[i], 2));
fprintf(*foutput, "X = %lf; Y = %lf\n", x_Ext[i], resultado_Y);
}
fprintf(*foutput, "\n");

free(x2);
free(x3);
free(x4);
free(xy);
free(x2y);
free(YY);
free(delta);
free(delta2);
free(beta);
free(beta2);
}

```

AE2.CD1.11

AE2.CD1.11

```
//gauss_jordan.h
#ifndef __GAUSS_JORDAN_H__
#define __GAUSS_JORDAN_H__
```

```
#include <stdlib.h>
#include <math.h>
#include "matrix.h"
```

```
typedef struct Row Row;
typedef struct Matrix Matrix;
```

```
void GaussJordan(int, Matrix *, double *, double *);
void GaussJordan_vParabolica(int, Matrix *, double *, double *, double *);
void Elimina_GJ(int, Matrix *, Matrix *);
Matrix generate_matrix(int, int);
void pivoteo(int, Matrix *, int, int);
int val_max_col(int, int, double *, int);
```

```
#endif
```

```
//gauss_jordan.c
#include "gauss_jordan.h"
```

```
Matrix generate_matrix(int rows, int columns)
{
    Matrix m;
    m.nRows = rows;
    m.nColumns = columns;
    m.rows = (Row *)malloc(rows * sizeof(Row));
    for (int i = 0; i < rows; i++)
    {
        m.rows[i].columns = (double *)malloc(columns * sizeof(double));
    }
    return m;
}
```

```
void GaussJordan(int n, Matrix *matrix, double *a0, double *a1)
{
```

```
    //fprintf(foutput, "-----METODO GAUSS JORDAN-----\n");
```

```
    Matrix c_matrix = generate_matrix(matrix->nRows, matrix->nColumns);
```

```
    for (int i = 0; i < matrix->nRows; i++)
```

```
    {
        for (int j = 0; j < matrix->nColumns; j++)
```

```
        {
            c_matrix.rows[i].columns[j] = matrix->rows[i].columns[j];
        }
    }
```

```
//print_matrix(&c_matrix, &foutput);
Elimina_GJ(n, matrix, &c_matrix);
*a0 = c_matrix.rows[n - 2].columns[n];
*a1 = c_matrix.rows[n - 1].columns[n];
}
```

```
void GaussJordan_vParabolica(int n, Matrix *matrix, double *a0, double *a1,
double *a2)
```

AE2.CD1.I1

```
{
//fprintf(foutput, "-----METODO GAUSS JORDAN-----\n");
Matrix c_matrix = generate_matrix(matrix->nRows, matrix->nColumns);
for (int i = 0; i < matrix->nRows; i++)
{
for (int j = 0; j < matrix->nColumns; j++)
{
c_matrix.rows[i].columns[j] = matrix->rows[i].columns[j];
}
}
//print_matrix(&c_matrix, &foutput);
Elimina_GJ(n, matrix, &c_matrix);
*a0 = c_matrix.rows[n - 3].columns[n];
*a1 = c_matrix.rows[n - 2].columns[n];
*a2 = c_matrix.rows[n - 1].columns[n];
}
```

```
void Elimina_GJ(int n, Matrix *matrix, Matrix *c_matrix)
```

```
{
int r, rm, i, j, c = 0;
double *copia = (double *)malloc(n * sizeof(double));
double *tempr = (double *)malloc(n * sizeof(double));
float constt, pivot;
for (r = 0; r < n; r++)
{
rm = 0;
for (i = 0; i < n; i++)
{
copia[i] = c_matrix->rows[i].columns[r];
}
rm = val_max_col(r, n, copia, rm);
if (rm != r)
{
pivoteo(n, &(*c_matrix), r, rm);
}
pivot = c_matrix->rows[r].columns[r];
for (j = 0; j ≤ n; j++)
{
c_matrix->rows[r].columns[j] = c_matrix->rows[r].columns[j] / pivot;
}
```

AE2.CD1.I1

```
}
for (i = 0; i < n; i++)
{
if (r ≠ i)
{
constt = -(c_matrix→rows[i].columns[r]);
for (j = 0; j ≤ n; j++)
{
AE2.CD1.I1 tempr[j] = constt * c_matrix→rows[r].columns[j];
c_matrix→rows[i].columns[j] = tempr[j] + c_matrix→rows[i].columns[j];
}
}
}
c++;
//fprintf(*foutput, "ITERACION No %d\n", r + 1);
//print_matrix(c_matrix, foutput);
}
//Comprueba(n, &>(*matrix), &>(*c_matrix), foutput);
}
```

```
int val_max_col(int rowpvt, int n, double *vec, int rmax)
{
rmax = rowpvt;
double pvtmax = fabs(vec[rowpvt]);
for (int i = rowpvt; i < n; i++)
{
AE2.CD1.I1 if (fabs(vec[i]) > pvtmax)
{
pvtmax = fabs(vec[i]);
rmax = i;
}
}
return rmax;
}
```

```
void pivoteo(int n, Matrix *matrix, int rowpvt, int rmax)
{
for (int i = 0; i ≤ n; i++)
{
AE2.CD1.I1 double temp = matrix→rows[rowpvt].columns[i];
matrix→rows[rowpvt].columns[i] = matrix→rows[rmax].columns[i];
matrix→rows[rmax].columns[i] = temp;
}
}
```

```

//matrix.h
#ifndef __MATRIX_H__
#define __MATRIX_H__

struct Row
{
double *columns;
};

struct Matrix
{
struct Row *rows;
int nColumns, nRows;
};

#endif

//error_info.h
#ifndef __ERROR_INFO_H__
#define __ERROR_INFO_H__

#include <stdio.h>
#include <math.h>

void print_error_info(int n_elementos, double St, double Sr, FILE ** foutput);

#endif

//error_info.c
#include "error_info.h"

void print_error_info(int n_elementos, double St, double Sr, FILE **foutput)
{
double desest = sqrt(Sr / (n_elementos - 2));
double destip = sqrt(St / (n_elementos - 1));
double coefdet = (St - Sr) / St;
double coefcor = sqrt(coefdet);

fprintf(**foutput, "\n-----DESVIACIONES Y COEFICIENTES-----\n");
fprintf(**foutput, "St: %lf, Sr: %lf\n\n", St, Sr);
fprintf(**foutput, "Desviacion del Estimado: %lf; \t Desviacion Estandar: %lf \n\n", desest, destip);
fprintf(**foutput, "Coeficiente Determinado: %lf; \t Coeficiente Correl.: %lf \n\n\n", coefdet, coefcor);
}

```

AE2.CD1.I1

```
//ajuste_exponencial.h
#ifndef __EXPONENCIAL_H__
#define __EXPONENCIAL_H__

#include <stdio.h>
#include "gauss_jordan.h"
#include "error_info.h"

void exponencial(int n_elementos, double *x, double *y, int n_xInt, double
*x_Int, int n_xExt, double *x_Ext, FILE **foutput);
void lineal_e(int n_elementos, double **x, double **y, double *a0, double *a1,
double **YY);

#endif
```

```
//ajuste_exponencial.c
#include "ajuste_exponencial.h"
```

```
void exponencial(int n_elementos, double *x, double *y, int n_xInt, double
*x_Int, int n_xExt, double *x_Ext, FILE **foutput)
{
fprintf(*foutput, "-----MODELO EXPONENCIAL
[Y=Ae**(Bx)]-----\n\n");
double a0, a1, Sumy = 0;
double *z = (double *)malloc(n_elementos * sizeof(double));
double *YY = (double *)malloc(n_elementos * sizeof(double));
double *ZZ = (double *)malloc(n_elementos * sizeof(double));
double *delta = (double *)malloc(n_elementos * sizeof(double));
double *delta2 = (double *)malloc(n_elementos * sizeof(double));
double *beta = (double *)malloc(n_elementos * sizeof(double));
double *beta2 = (double *)malloc(n_elementos * sizeof(double));
for (int i = 0; i < n_elementos; i++)
{
z[i] = log(y[i]);
Sumy = Sumy + y[i];
}
lineal_e(n_elementos, &x, &z, &a0, &a1, &ZZ);
double St = 0, Sr = 0;
double Ymedia = Sumy / n_elementos;
for (int i = 0; i < n_elementos; i++)
{
YY[i] = exp(ZZ[i]);
delta[i] = y[i] - YY[i];
delta2[i] = pow(delta[i], 2);
beta[i] = y[i] - Ymedia;
beta2[i] = pow(beta[i], 2);
Sr = Sr + delta2[i];
}
```

AE2.CD1.11


```

St = St + beta2[i]; ← AE2.CD1.I1
}
printf("a0: %lf, a1:%lf \n", a0, a1);
printf("Sr: %lf, St:%lf \n", Sr, St);
fprintf(*foutput, "Pto. x y z=lny x2 xz Z YY delta delta2 beta beta2\n");
for (int i = 0; i < n_elementos; i++)
{
fprintf(*foutput, "%d %10.4lf %10.4lf \t %10.4lf \t %10.4lf \t %10.4lf \t
%10.4lf \t %10.4lf \t %10.4lf \t %10.4lf \t %10.4lf \t %10.4lf\n", i + 1, x[i],
y[i], z[i], pow(x[i], 2), x[i] * y[i], ZZ[i], YY[i], delta[i], delta2[i],
beta[i], beta2[i]);
}
fprintf(*foutput, "\nYmedia: %lf \n", Ymedia);
fprintf(*foutput, "\na0: %9.5lf, a1:%9.5lf\n", a0, a1);
fprintf(*foutput, "\nModelo lineal: Z = %9.5lf + %9.5lfx\n", a0, a1);
double A = exp(a0); ← AE2.CD1.I1
double B = a1; ← AE2.CD1.I1
fprintf(*foutput, "\nA=e**a0 \t A=%9.5lf\nB=a1 \t B=%9.5lf\n", A, B);
fprintf(*foutput, "\nModelo exponencial: Y = %9.5lfe**(%9.5lfx)\n\n", A, B);

print_error_info(n_elementos, St, Sr, &foutput);

double resultado_Y;
fprintf(*foutput, "\n-----Calculo de Y internas-----\n");
for (int i = 0; i < n_xInt; i++)
{
resultado_Y = A * exp(B * x_Int[i]); ← AE2.CD1.I1
fprintf(*foutput, "X = %lf; Y = %lf\n", x_Int[i], resultado_Y);
}
fprintf(*foutput, "\n-----Calculo de Y externas-----\n");
for (int i = 0; i < n_xExt; i++)
{
resultado_Y = A * exp(B * x_Ext[i]); ← AE2.CD1.I1
fprintf(*foutput, "X = %lf; Y = %lf\n", x_Ext[i], resultado_Y);
}
fprintf(*foutput, "\n");

free(z);
free(YY);
free(ZZ);
free(delta);
free(delta2);
free(beta);
free(beta2);
}

```

```

void lineal_e(int n_elementos, double **x, double **y, double *a0, double *a1,
double **YY)
{
double Ymedia = 0;
double Sumx = 0;
double Sumy = 0;
double Sumx2 = 0;
double Sumxy = 0;
double *x2 = (double *)malloc(n_elementos * sizeof(double));
double *xy = (double *)malloc(n_elementos * sizeof(double));

for (int i = 0; i < n_elementos; i++)
{
x2[i] = pow((*x)[i], 2);
xy[i] = (*x)[i] * (*y)[i];
Sumx = Sumx + (*x)[i];
Sumy = Sumy + (*y)[i];
Sumx2 = Sumx2 + x2[i];
Sumxy = Sumxy + xy[i];
}
Ymedia = Sumy / n_elementos;

Matrix a = generate_matrix(2,3);
a.rows[0].columns[0] = n_elementos;
a.rows[0].columns[1] = Sumx;
a.rows[0].columns[2] = Sumy;
a.rows[1].columns[0] = Sumx;
a.rows[1].columns[1] = Sumx2;
a.rows[1].columns[2] = Sumxy;

GaussJordan(2, &a, &(*a0), &(*a1));

double Sr = 0;
double St = 0;

for (int i = 0; i < n_elementos; i++)
{
(*YY)[i] = *a0 + ((*a1) * (*x)[i]);
}

free(x2);
free(xy);
}

```

AE2.CD1.I1

AE2.CD1.I1

```

//ajuste_potencias.h
#ifndef __POTENCIAS_H__
#define __POTENCIAS_H__

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "error_info.h"

void potencias(int n_elementos, double *x, double *y, int n_xInt, double
*x_Int, int n_xExt, double *x_Ext, FILE **foutput);
void lineal_p(int n_elementos, double *x, double *y, double *a0, double *a1,
double **YY);

#endif

```

```

//ajuste_potencias.c

```

```

#include "ajuste_potencias.h"

void potencias(int n_elementos, double *x, double *y, int n_xInt, double
*x_Int, int n_xExt, double *x_Ext, FILE **foutput)
{
fprintf(*foutput, "-----MODELO DE POTENCIAS
[Y=Ax**B]-----\n\n");
double a0, a1;
double *w = (double *)malloc(n_elementos * sizeof(double));
double *z = (double *)malloc(n_elementos * sizeof(double));
double *YY = (double *)malloc(n_elementos * sizeof(double));
double *ZZ = (double *)malloc(n_elementos * sizeof(double));

double *delta = (double *)malloc(n_elementos * sizeof(double));
double *delta2 = (double *)malloc(n_elementos * sizeof(double));
double *beta = (double *)malloc(n_elementos * sizeof(double));
double *beta2 = (double *)malloc(n_elementos * sizeof(double));

double Sumy = 0, Ymedia, St = 0, Sr = 0;
for (int i = 0; i < n_elementos; i++)
{
w[i] = log10(x[i]);
z[i] = log10(y[i]);
Sumy = Sumy + y[i];
}
Ymedia = Sumy / n_elementos;
lineal_p(n_elementos, w, z, &a0, &a1, &ZZ);
for (int i = 0; i < n_elementos; i++)
{
YY[i] = pow(10, ZZ[i]);
}
}

```

AE2.CD1.I1

```
delta[i] = y[i] - YY[i];
delta2[i] = pow(delta[i], 2);
beta[i] = y[i] - Ymedia;
beta2[i] = pow(beta[i], 2);
Sr = Sr + delta2[i];
St = St + beta2[i];
}
```



//METODO PARA IMPRIMIR LA TABLA CORRESPONDIENTE

```
printf("Sr: %lf, St:%lf \n", Sr, St);
fprintf(*foutput, "Pto. x y w=logx z=logy w2 wz Z YY delta delta2 beta beta2\n");
for (int i = 0; i < n_elementos; i++)
{
  fprintf(*foutput, "%d \t %9.4lf \t %9.4lf \t %9.4lf \t %9.4lf \t %9.4lf \t %9.4lf \t %9.4lf \t %9.4lf \t %9.4lf \t %9.4lf \t %9.4lf \t %9.4lf \t %9.4lf \t %9.4lf \t %9.4lf \t %9.4lf \n", i + 1, x[i], y[i], w[i], z[i], pow(w[i], 2), w[i] * z[i], ZZ[i], YY[i], delta[i], delta2[i], beta[i], beta2[i]);
}
fprintf(*foutput, "\nYmedia: %lf \n", Ymedia);
fprintf(*foutput, "\na0: %9.5lf, a1:%9.5lf\n", a0, a1);
fprintf(*foutput, "\nModelo lineal: Z = %9.5lf + %9.5lfx\n", a0, a1);
double A = pow(10, a0);
double B = a1;
fprintf(*foutput, "\nA=10**a0 \t A=%9.5lf\nB=a1 \t B=%9.5lf\n", A, B);
fprintf(*foutput, "\nModelo potencial: Y = %9.5lfx**(%9.5lf)\n\n", A, B);

print_error_info(n_elementos, St, Sr, &foutput);

fprintf(*foutput, "\n-----Calculo de Y internas-----\n");
for (int i = 0; i < n_xInt; i++)
{
  double resultado_Y = pow(A * x_Int[i], B);
  fprintf(*foutput, "X = %lf; Y = %lf\n", x_Int[i], resultado_Y);
}

fprintf(*foutput, "\n-----Calculo de Y externas-----\n");
for (int i = 0; i < n_xExt; i++)
{
  double resultado_Y = pow(A * x_Ext[i], B);
  fprintf(*foutput, "X = %lf; Y = %lf\n", x_Ext[i], resultado_Y);
}
fprintf(*foutput, "\n");

free(z);
free(YY);
```

AE2.CD1.I1

AE2.CD1.I1

AE2.CD1.I1

```
free(ZZ);
free(delta);
free(delta2);
free(beta);
free(beta2);
}
```

```
void lineal_p(int n_elementos, double *x, double *y, double *a0, double *a1,
double **YY)
```

```
{
double Ymedia = 0;
double Sumx = 0;
double Sumy = 0;
double Sumx2 = 0;
double Sumxy = 0;
double *x2 = (double *)malloc(n_elementos * sizeof(double));
double *xy = (double *)malloc(n_elementos * sizeof(double));
```

```
for (int i = 0; i < n_elementos; i++)
```

```
{
x2[i] = pow(x[i], 2);
xy[i] = (x[i]) * (y[i]);
Sumx = Sumx + x[i];
Sumy = Sumy + y[i];
Sumx2 = Sumx2 + x2[i];
Sumxy = Sumxy + xy[i];
}
```

```
Ymedia = Sumy / n_elementos;
```

```
*a0 = ((Sumx2 * Sumy) - (Sumx * Sumxy)) / ((n_elementos * Sumx2) - pow(Sumx,
2));
```

```
*a1 = ((n_elementos * Sumxy) - (Sumx * Sumy)) / ((n_elementos * Sumx2) -
pow(Sumx, 2));
```

```
printf("a0: %lf, a1: %lf\n", *a0, *a1);
```

```
for (int i = 0; i < n_elementos; i++)
```

```
{
(*YY)[i] = *a0 + ((*a1) * (x[i]));
}
```

```
free(x2);
```

```
free(xy);
```

```
}
```

AE2.CD1.I1

```

//ajuste_crecimiento.h
#ifndef __CRECIMIENTO_H__
#define __CRECIMIENTO_H__

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "error_info.h"

void crecimiento(int n_elementos, double *x, double *y, int n_xInt, double
*x_Int, int n_xExt, double *x_Ext, FILE **foutput);
void lineal_c(int n_elementos, double *x, double *y, double *a0, double *a1,
double **YY);

#endif

```

```

//ajuste_crecimiento.c
#include "ajuste_crecimiento.h"

void crecimiento(int n_elementos, double *x, double *y, int n_xInt, double
*x_Int, int n_xExt, double *x_Ext, FILE **foutput)
{
fprintf(*foutput, "-----MODELO DE CRECIMIENTO
[Y=Ax/(B+x)]-----\n\n");
double a0, a1;
double *v = (double *)malloc(n_elementos * sizeof(double));
double *z = (double *)malloc(n_elementos * sizeof(double));
double *YY = (double *)malloc(n_elementos * sizeof(double));
double *ZZ = (double *)malloc(n_elementos * sizeof(double));

double *delta = (double *)malloc(n_elementos * sizeof(double));
double *delta2 = (double *)malloc(n_elementos * sizeof(double));
double *beta = (double *)malloc(n_elementos * sizeof(double));
double *beta2 = (double *)malloc(n_elementos * sizeof(double));

double Sumy = 0;
for (int i = 0; i < n_elementos; i++)
{
v[i] = 1 / x[i];
z[i] = 1 / y[i];
Sumy = Sumy + y[i];
//printf("Point:%d, v: %lf, z: %lf \n", i + 1, v[i], z[i]);
}
double Sr = 0;
double St = 0;
double Ymedia = Sumy / n_elementos;
lineal_c(n_elementos, v, z, &a0, &a1, &ZZ);
}

```

AE2.CD1.11

```

for (int i = 0; i < n_elementos; i++)
{
YY[i] = 1 / ZZ[i];
delta[i] = y[i] - YY[i];
delta2[i] = pow(delta[i], 2);
beta[i] = y[i] - Ymedia;
beta2[i] = pow(beta[i], 2);
Sr = Sr + delta2[i];
St = St + beta2[i];
}
printf("Sr: %lf, St:%lf \n", Sr, St);
fprintf(*foutput, "Pto. x y v=1/x z=1/y v2 vz Z YY delta delta2 beta beta2\n");
for (int i = 0; i < n_elementos; i++)
{
fprintf(*foutput, "%d \t %9.4lf \t %9.4lf \t %9.4lf \t %9.4lf \t %9.4lf \t
%9.4lf \t %9.4lf \t %9.4lf \t %9.4lf \t %9.4lf \t %9.4lf \t %9.4lf \t %9.4lf \n", i + 1,
x[i], y[i], v[i], z[i], pow(v[i], 2), v[i] * z[i], ZZ[i], YY[i], delta[i],
delta2[i], beta[i], beta2[i]);
}
fprintf(*foutput, "\nYmedia: %lf \n", Ymedia);
fprintf(*foutput, "\na0: %9.5lf, a1:%9.5lf\n", a0, a1);
fprintf(*foutput, "\nModelo lineal: Z = %9.5lf + %9.5lfx\n", a0, a1);
double A = 1 / a0;
double B = a1 * A;
fprintf(*foutput, "\nA=1/a0 \t A=%9.5lf\nB=a1*A \t B=%9.5lf\n", A, B);
fprintf(*foutput, "\nModelo de crecimiento: Y = %9.5lfx/(%9.5lf + x)\n\n", A,
B);

print_error_info(n_elementos, St, Sr, &foutput);

fprintf(*foutput, "\n-----Calculo de Y internas-----\n");
for (int i = 0; i < n_xInt; i++)
{
double resultado_Y = (A * x_Int[i]) / (B + x_Int[i]);
fprintf(*foutput, "X = %lf; Y = %lf\n", x_Int[i], resultado_Y);
}

fprintf(*foutput, "\n-----Calculo de Y externas-----\n");
for (int i = 0; i < n_xExt; i++)
{
double resultado_Y = (A * x_Ext[i]) / (B + x_Ext[i]);
fprintf(*foutput, "X = %lf; Y = %lf\n", x_Ext[i], resultado_Y);
}
fprintf(*foutput, "\n");

free(v);

```

AE2.CD1.I1

AE2.CD1.I1

AE2.CD1.I1

```
free(z);
free(YY);
free(ZZ);
free(delta);
free(delta2);
free(beta);
free(beta2);
}
```

```
void lineal_c(int n_elementos, double *x, double *y, double *a0, double *a1,
double **YY)
```

```
{
double Ymedia = 0;
double Sumx = 0;
double Sumy = 0;
double Sumx2 = 0;
double Sumxy = 0;
double *x2 = (double *)malloc(n_elementos * sizeof(double));
double *xy = (double *)malloc(n_elementos * sizeof(double));
```

```
for (int i = 0; i < n_elementos; i++)
```

```
{
x2[i] = pow(x[i], 2);
xy[i] = (x[i]) * (y[i]);
Sumx = Sumx + x[i];
Sumy = Sumy + y[i];
Sumx2 = Sumx2 + x2[i];
Sumxy = Sumxy + xy[i];
}
```

```
Ymedia = Sumy / n_elementos;
```

```
*a0 = ((Sumx2 * Sumy) - (Sumx * Sumxy)) / ((n_elementos * Sumx2) - pow(Sumx,
2));
```

```
*a1 = ((n_elementos * Sumxy) - (Sumx * Sumy)) / ((n_elementos * Sumx2) -
pow(Sumx, 2));
```

```
double Sr = 0;
double St = 0;
printf("a0: %lf, a1: %lf\n", *a0, *a1);
```

```
for (int i = 0; i < n_elementos; i++)
```

```
{
(*YY)[i] = *a0 + ((*a1) * (x[i]));
}
```

```
free(x2);
```

```
free(xy);
```

```
}
```

AE2.CD1.I1


```

//ajuste_gowpetz.h
#ifndef __GOWPETZ_H__
#define __GOWPETZ_H__

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "error_info.h"

void gowpetz(int n_elementos, double *x, double *y, int n_xInt, double *x_Int,
int n_xExt, double *x_Ext, FILE **foutput);
void lineal_g(int n_elementos, double **x, double **y, double *a0, double *a1,
double **YY);

#endif

//ajuste_gowpetz.c
#include "ajuste_gowpetz.h"

void gowpetz(int n_elementos, double *x, double *y, int n_xInt, double *x_Int,
int n_xExt, double *x_Ext, FILE **foutput)
{
    fprintf(*foutput, "-----MODELO DE GOWPETZ
[Y=AB**(e**x)]-----\n\n");
    double a0, a1;
    double *u = (double *)malloc(n_elementos * sizeof(double));
    double *z = (double *)malloc(n_elementos * sizeof(double));
    double *YY = (double *)malloc(n_elementos * sizeof(double));
    double *ZZ = (double *)malloc(n_elementos * sizeof(double));
    double *delta = (double *)malloc(n_elementos * sizeof(double));
    double *delta2 = (double *)malloc(n_elementos * sizeof(double));
    double *beta = (double *)malloc(n_elementos * sizeof(double));
    double *beta2 = (double *)malloc(n_elementos * sizeof(double));
    double Sumy = 0, Ymedia, Sr = 0, St = 0;
    for (int i = 0; i < n_elementos; i++)
    {
        u[i] = exp(x[i]);
        z[i] = log(y[i]);
        Sumy = Sumy + y[i];
    }

    lineal_g(n_elementos, &u, &z, &a0, &a1, &ZZ);
    Ymedia = Sumy / n_elementos;
    for (int i = 0; i < n_elementos; i++)
    {
        YY[i] = exp(ZZ[i]);
        delta[i] = y[i] - YY[i];
    }
}

```

AE2.CD1.11

```
delta2[i] = pow(delta[i], 2);
beta[i] = y[i] - Ymedia;
beta2[i] = pow(beta[i], 2);
Sr = Sr + delta2[i];
St = St + beta2[i];
}
```

AE2.CD1.I1

```
printf("a0: %lf, a1:%lf \n", a0, a1);
printf("Sr: %lf, St:%lf \n", Sr, St);
fprintf(*foutput, "Pto. x y u=e**x z=lny u2 uz Z YY delta delta2 beta beta2\n");
for (int i = 0; i < n_elementos; i++)
{
fprintf(*foutput, "%2d %9.4lf %9.4lf \t %12.4lf \t %6.4lf \t %20.4lf \t %10.4lf\n", i + 1, x[i], y[i], u[i], z[i], pow(u[i], 2), u[i] * z[i], ZZ[i], YY[i], delta[i], delta2[i], beta[i], beta2[i]);
}
fprintf(*foutput, "\nYmedia: %lf \n", Ymedia);
fprintf(*foutput, "\na0: %9.5lf, a1:%9.5lf\n", a0, a1);
fprintf(*foutput, "\nModelo lineal: Z = %9.5lf + %9.5lfx\n", a0, a1);
double A = exp(a0);
double B = exp(a1);
fprintf(*foutput, "\nA=e**a0 \t A=%9.5lf\nB=e**a1 \t B=%9.5lf\n", A, B);
fprintf(*foutput, "\nModelo de Gowpetz: Y = %9.5lf**(e**x)\n\n", A * B);

print_error_info(n_elementos, St, Sr, &foutput);
```

AE2.CD1.I1

```
double resultado_Y;
fprintf(*foutput, "\n-----Calculo de Y internas-----\n");
for (int i = 0; i < n_xInt; i++)
{
resultado_Y = pow(A * B, exp(x_Int[i]));
fprintf(*foutput, "X = %lf; Y = %lf\n", x_Int[i], resultado_Y);
}

fprintf(*foutput, "\n-----Calculo de Y externas-----\n");
for (int i = 0; i < n_xExt; i++)
{
resultado_Y = pow(A * B, exp(x_Ext[i]));
fprintf(*foutput, "X = %lf; Y = %lf\n", x_Ext[i], resultado_Y);
}
fprintf(*foutput, "\n");
/*
free(u);
free(z);
free(YY);
```

AE2.CD1.I1

```

free(ZZ);
free(delta);
free(delta2);
free(beta);
free(beta2);
*/
}

void lineal_g(int n_elementos, double **x, double **y, double *a0, double *a1,
double **YY)
{
double Ymedia = 0;
double Sumx = 0;
double Sumy = 0;
double Sumx2 = 0;
double Sumxy = 0;
double *x2 = (double *)malloc(n_elementos * sizeof(double));
double *xy = (double *)malloc(n_elementos * sizeof(double));

for (int i = 0; i < n_elementos; i++)
{
x2[i] = pow((*x)[i], 2);
xy[i] = (*x)[i] * (*y)[i];
Sumx = Sumx + (*x)[i];
Sumy = Sumy + (*y)[i];
Sumx2 = Sumx2 + x2[i];
Sumxy = Sumxy + xy[i];
}
Ymedia = Sumy / n_elementos;

*a0 = (Sumx2 * Sumy - Sumx * Sumxy) / (n_elementos * Sumx2 - pow(Sumx, 2));
*a1 = (n_elementos * Sumxy - Sumx * Sumy) / (n_elementos * Sumx2 - pow(Sumx,
2));

for (int i = 0; i < n_elementos; i++)
{
(*YY)[i] = *a0 + ((*a1) * (*x)[i]);
}

free(x2);
free(xy);
}

```

AE2.CD1.I1