INFOTEC - MCDI

Materia: Matemáticas para la Ciencia de Datos
Profesor: Dra. Briceyda B. Delgado
Alumno: David Rodriguez Gutierrez

Tarea 5: Modelación experimental

> 1. Cuando se utilizan programas para construir simulaciones, a menudo se necesita una estimación del tiempo de CPU necesario para ejecutar la simulación, ya que algunos parámetros han cambiado. A veces, esto puede ser un problema muy difícil de resolver, porque no está claro cómo el esfuerzo de la CPU depende de los parámetros. En este ejercicio, consideraremos un programa muy simple para ilustrar cómo se pueden generar estimaciones de tiempo.
>
> Supongamos que queremos una solución numérica del problema
>
> $$y'(t) = e^{y(t)}, \quad y(0) = 0, \quad 0 \leq t \leq 1. \qquad (1)$$
>
> Si intentamos resolver (1) usando el esquema estándar de Euler, tenemos que
>
> $$y_{k+1} = y_k + \Delta t \cdot e^{y_k}, \quad k = 0, 1, \ldots, n-1.$$
>
> con $y_0 = 0$. Aquí, $\Delta t = \frac{1}{n}$, donde $n > 0$ es un número entero. En la Tabla siguiente hemos enumerado el tiempo de CPU que necesita un sencillo programa en C para calcular y_n en el tiempo $t = 1$ en un procesador Pentium III de 600 MHz.
>
> | n | **CPU time** | y_n |
> |---|---|---|
> | 100,000 | 0.05 | 9.9181 |
> | 200,000 | 0.09 | 10.549 |
> | 300,000 | 0.13 | 10.919 |
> | 400,000 | 0.18 | 11.183 |
>
> A partir del esquema, es razonable suponer que el tiempo de CPU, $c(n)$, puede modelarse adecuadamente utilizando una función lineal, es decir:
>
> $$c(n) = \alpha + \beta n$$
>
> **(a)** Utilice los datos de la tabla para determinar α y β por el método de mínimos cuadrados.
>
> **(b)** Estime el tiempo de CPU necesario en los casos de $n = 10^6$ y $n = 10^7$.

(a) Utilizamos los datos de la tabla para determinar α y β por el método de mínimos cuadrados.

Modelar el tiempo de CPU, $c(n)$, en función de n utilizando la equación lineal:

$$c(n) = \alpha + \beta n$$

Datos:

n_i	c_i **(s)**
100,000	0.05
200,000	0.09
300,000	0.13
400,000	0.18

Sumas:

- Suma de c_i:

$$\sum c_i = 0.05 + 0.09 + 0.13 + 0.18 = 0.45 \, \text{segundos}$$

- Suma de n_i:

$$\sum n_i = 100,000 + 200,000 + 300,000 + 400,000 = 1,000,000$$

- Suma de $n_i c_i$:

$$\sum n_i c_i = (100,000 \times 0.05) + (200,000 \times 0.09) + (300,000 \times 0.13) + (400,000 \times 0.18) = 134,000$$

- Suma de n_i:

$$\sum n_i^2 = (100,000)^2 + (200,000)^2 + (300,000)^2 + (400,000)^2 = 300,000,000,000$$

Formulación de las ecuaciones normales:

Las ecuaciones normales para el ajuste lineal son:

$$\begin{cases} \sum c_i = 4\alpha + \beta \sum n_i \\ \sum n_i c_i = \alpha \sum n_i + \beta \sum n_i^2 \end{cases}$$

Sustituyendo los valores calculados:

$$\begin{cases} 0.45 = 4\alpha + 1,000,000\beta \\ 134,000 = 1,000,000\alpha + 300,000,000,000\beta \end{cases}$$

Resolución del sistema de ecuaciones:

De la primera ecuación:

$$4\alpha = 0.45 - 1,000,000\beta \implies \alpha = 0.1125 - 250,000\beta$$

Sustituyendo α en la segunda ecuación:

$$134,000 = 1,000,000(0.1125 - 250,000\beta) + 300,000,000,000\beta$$

Simplificando:

$ 134,000 = 112,500 - 250,000,000,000 \beta + 300,000,000,000 \beta \\ 134,000 - 112,500 = 50,000,000,000 \beta \\ 21,500 = 50,000,000,000 \beta \\ \beta = \frac{21,500}{50,000,000,000} = 4.3 \times 10^{-7} \, \text{segundos por unidad de } n $

Determinando α:

$ \alpha = 0.1125 - 250,000 \times 4.3 \times 10^{-7} = 0.1125 - 0.1075 = 0.005 \, \text{segundos} $

Resultados:

$ \boxed{ \begin{aligned} \alpha &= 0.005 \, \text{segundos} \\ \beta &= 4.3 \times 10^{-7} \, \text{segundos por unidad de } n \end{aligned} } $

(b) Estime el tiempo de CPU necesario en los casos de $n = 10^6$ y $n = 10^7$.

Utilizando el modelo lineal obtenido:

$$c(n) = \alpha + \beta n = 0.005 + 4.3 \times 10^{-7} \, n$$

Para $n = 10^6$:

$ c(10^6) = 0.005 + 4.3 \times 10^{-7} \times 1,000,000 = 0.005 + 0.43 = 0.435 \, \text{segundos} $

Redondeando:

$$\boxed{c(10^6) \approx 0.435 \, \text{segundos}}$$

Para $n = 10^7$:

$ c(10^7) = 0.005 + 4.3 \times 10^{-7} \times 10,000,000 = 0.005 + 4.3 = 4.305 \, \text{segundos} $

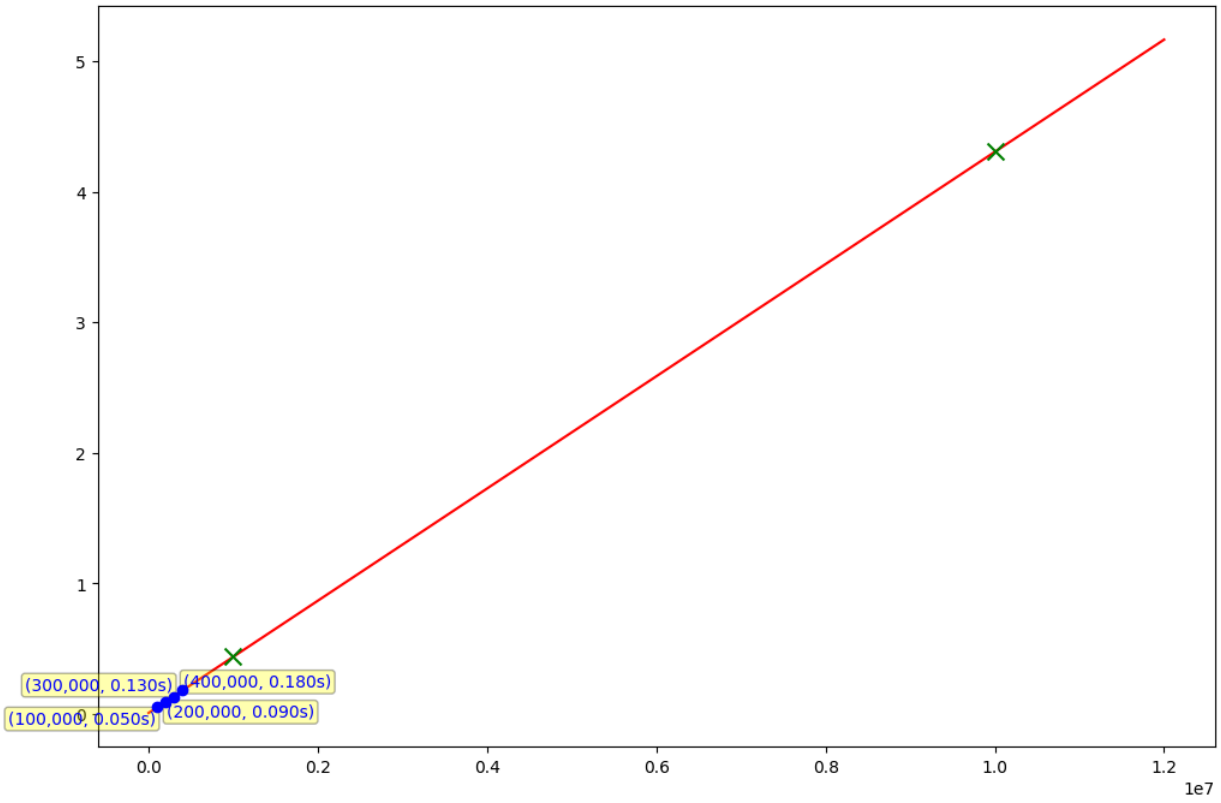
Redondeando:

$$\boxed{c(10^7) \approx 4.305 \, \text{segundos}}$$

Gráfica de la soluciones.

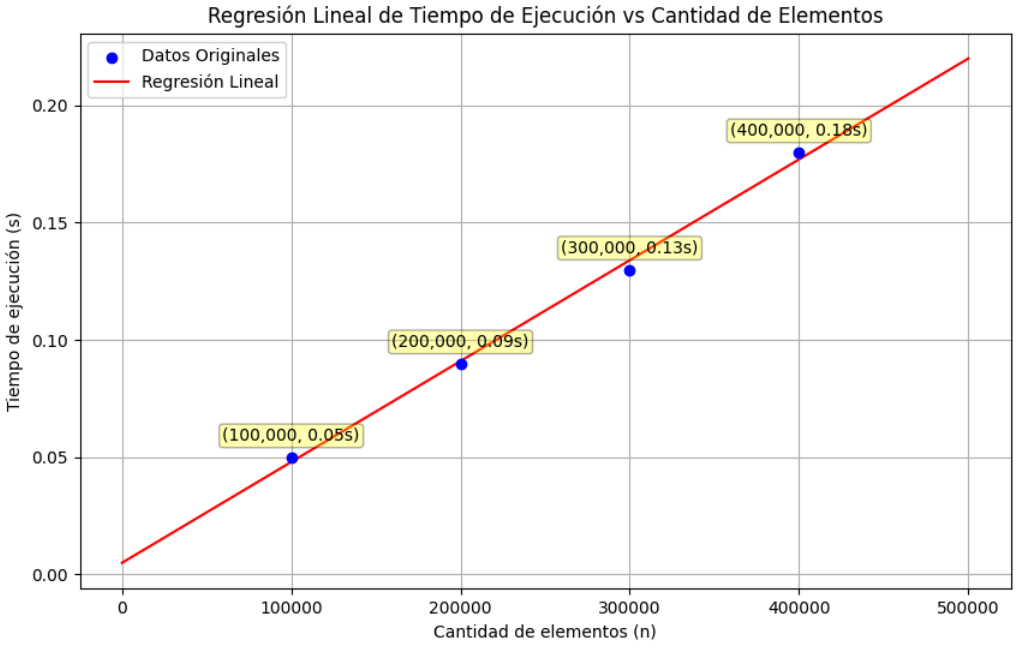
In [8]:
```
import matplotlib.pyplot as plt
import numpy as np

# Datos originales
n_i = np.array([100000, 200000, 300000, 400000])
```

```python
c_i = np.array([0.05, 0.09, 0.13, 0.18])

# Modelo de regresión lineal
alpha = 0.005
beta = 4.3e-7

# Función lineal
def c_model(n):
    return alpha + beta * n

# Valores para la línea de regresión
n_line = np.linspace(0, 1.2e7, 500)
c_line = c_model(n_line)

# Estimaciones
n_est = np.array([1e6, 1e7])
c_est = c_model(n_est)

# Crear la gráfica
plt.figure(figsize=(12, 8))

# Graficar los puntos originales
plt.scatter(n_i, c_i, color='blue', label='Datos Originales', zorder=5)

# Graficar la línea de regresión
plt.plot(n_line, c_line, color='red', label='Regresión Lineal', zorder=3)

# Graficar las estimaciones
plt.scatter(n_est, c_est, color='green', marker='x', s=100, label='Estimaciones', zorder=5)

# Función para ajustar las posiciones de las etiquetas
def ajustar_etiquetas(n, c, color, desplazamiento):
    for i in range(len(n)):
        plt.annotate(
            f'({int(n[i]):,}, {c[i]:.3f}s)',
            (n[i], c[i]),
            textcoords="offset points",
            xytext=desplazamiento[i],  # Desplazamientos específicos para cada punto
            ha='center',
            fontsize=10,
            color=color,
            bbox=dict(boxstyle="round,pad=0.2", fc="yellow", alpha=0.3)
        )

# Desplazamientos para los datos originales
desplazamiento_datos = [(-45, -10), (45, -9), (-45, 4), (45, 2)]
ajustar_etiquetas(n_i, c_i, 'blue', desplazamiento_datos)
```



Visualizando los datos originales

In [2]:
```python
import matplotlib.pyplot as plt
import numpy as np

# Datos originales
n_i = np.array([100000, 200000, 300000, 400000])
c_i = np.array([0.05, 0.09, 0.13, 0.18])

# Modelo de regresión lineal
alpha = 0.005
beta = 4.3e-7

# Función lineal
def c_model(n):
    return alpha + beta * n

# Valores para la línea de regresión
n_min = 0
n_max = 500000  # Extiende un poco más allá del máximo n_i para una mejor visualización
n_line = np.linspace(n_min, n_max, 500)
c_line = c_model(n_line)

# Crear la gráfica
plt.figure(figsize=(10, 6))

# Graficar los puntos originales
plt.scatter(n_i, c_i, color='blue', label='Datos Originales', zorder=5)

# Graficar la línea de regresión
plt.plot(n_line, c_line, color='red', label='Regresión Lineal', zorder=3)

# Agregar etiquetas a cada punto
for i, (x, y) in enumerate(zip(n_i, c_i)):
    plt.annotate(f'({x:,}, {y:.2f}s)',
                 (x, y),
                 textcoords="offset points",
                 xytext=(0, 10),  # Desplazamiento para evitar que la etiqueta se superponga con el punto
                 ha='center',
                 fontsize=10,
                 color='black',
                 bbox=dict(boxstyle="round,pad=0.2", fc="yellow", alpha=0.3))

# Configuraciones adicionales
plt.xlabel('Cantidad de elementos (n)')
plt.ylabel('Tiempo de ejecución (s)')
plt.title('Regresión Lineal de Tiempo de Ejecución vs Cantidad de Elementos')
plt.legend()
plt.grid(True)
plt.show()
```



---

2. Realice un análisis de las temperaturas promedio de uno de los 32 estados de México aproximando a través del método de mínimos cuadrados para el caso lineal y cuadrático. Los estados se asignarán de forma personalizada. Los datos provienen del Servicio Metereológico Nacional.

   https://datos.gob.mx/busca/dataset/temperatura-promedio-excel.

   Además, incluya las gráficas correspondientes.

Datos de Durango:

n	Año	y_n
1	1985	16.7697998689058
2	1986	16.7535210287069
3	1987	16.4303837448411
4	1988	17.0019075421182
5	1989	17.0781717025325
6	1990	17.0750907751858
7	1991	17.2337521544100
8	1992	16.7220320690133
9	1993	17.2885635579424
10	1994	18.0565008899168
11	1995	17.8967229535935
12	1996	17.3416218250177
13	1997	16.5835240099788
14	1998	17.5748688868413
15	1999	17.3942670461079
16	2000	17.3372485725621
17	2001	17.3956255093168
18	2002	17.6046132918079
19	2003	17.7282641642286
20	2004	17.5633017403916
21	2005	18.5783968421214
22	2006	18.1202371866188
23	2007	17.7070643203154
24	2008	17.3787250607339
25	2009	17.9908373753160
26	2010	16.9836282783173
27	2011	18.4863334741126
28	2012	18.3041666666667
29	2013	18.3322027608330
30	2014	18.4583333333333
31	2015	17.9000000000000
32	2016	18.6833333333333
33	2017	19.0416666666667
34	2018	18.9583333333333
35	2019	18.9000000000000
36	2020	18.6000000000000
37	2021	18.6000000000000
38	2022	18.5000000000000
39	2023	19.5000000000000

Determinar los estimadores por mínimos cuadrados de α y β para el modelo lineal $c(n) = \alpha + \beta n$,

Calcular las sumas necesarias**

Dado que n varía de 1 a 39, calculamos:

1. El número de datos:

$$N = 39$$

2. La suma de n:

$$\sum_{i=1}^{N} n_i = \frac{N(N+1)}{2} = \frac{39 \times 40}{2} = 780$$

3. La suma de n^2:

$$\sum_{i=1}^{N} n_i^2 = \frac{N(N+1)(2N+1)}{6} = \frac{39 \times 40 \times 79}{6} = 20{,}540$$

4. La suma de y_n y $\sum n_i y_i$ (calculadas a partir de los datos proporcionados):

- $\sum_{i=1}^{N} y_i = 693.8530$ (aproximadamente)
- $\sum_{i=1}^{N} n_i y_i = 14{,}161.6382$ (aproximadamente)

Establecer las ecuaciones normales

Las ecuaciones normales son:

1. $\sum y_i = N\alpha + \beta \sum n_i$
2. $\sum n_i y_i = \alpha \sum n_i + \beta \sum n_i^2$

Sustituyendo los valores calculados:

1. $693.8530 = 39\alpha + 780\beta$
2. $14{,}161.6382 = 780\alpha + 20{,}540\beta$

PResolver para β

Multiplicamos la primera ecuación por 780 y la segunda por 39 para eliminar α:

Primera ecuación multiplicada por 780:

$$780 \times 693.8530 = 780 \times 39\alpha + 780 \times 780\beta$$

Segunda ecuación multiplicada por 39:

$$39 \times 14{,}161.6382 = 39 \times 780\alpha + 39 \times 20{,}540\beta$$

Restamos la primera ecuación multiplicada de la segunda:

$$(39 \times 14{,}161.6382) - (780 \times 693.8530) = (0) + (801{,}060 - 608{,}400)\beta$$

Calculamos:

$$552{,}307.8918 - 541{,}205.4000 = 192{,}660\beta$$

$$11{,}102.4918 = 192{,}660\beta$$

Por lo tanto:

$$\beta = \frac{11{,}102.4918}{192{,}660} \approx 0.0576$$

Resolver para α

Sustituimos β en la primera ecuación:

$$693.8530 = 39\alpha + 780 \times 0.0576$$

Calculamos:

$$693.8530 = 39\alpha + 44.9280$$

Restamos 44.9280 de ambos lados:

$$648.9250 = 39\alpha$$

Por lo tanto:

$$\alpha = \frac{648.9250}{39} \approx 16.6391$$

Respuesta:

(a) Usando el método de mínimos cuadrados, los estimadores son:

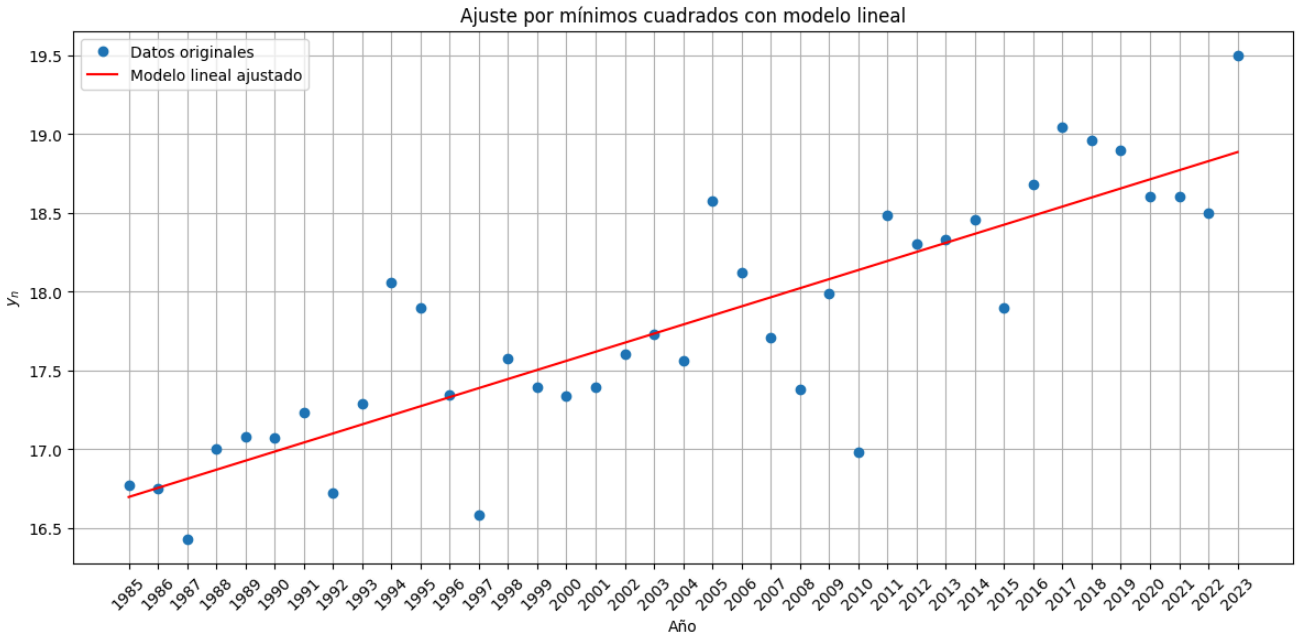
$$\alpha \approx 16.6391 \quad \text{y} \quad \beta \approx 0.0576$$

El modelo lineal es:

$$\boxed{c(n) = 16.6391 + 0.0576\,n}$$

Gráfica del modelo lineal:

In [3]:
```python
import numpy as np
import matplotlib.pyplot as plt

# Datos proporcionados (años)
n_years = np.array([
    1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992, 1993, 1994,
    1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004,
    2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014,
    2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023
])

# Valores de y_n correspondientes a cada año
y_n = np.array([
    16.7697998689058, 16.7535210287069, 16.4303837448411, 17.0019075421182,
    17.0781717025325, 17.0750907751858, 17.2337521544100, 16.7220320690133,
    17.2885635579424, 18.0565008899168, 17.8967229535935, 17.3416218250177,
    16.5835240099788, 17.5748688868413, 17.3942670461079, 17.3372485725621,
    17.3956255093168, 17.6046132918079, 17.7282641642286, 17.5633017403916,
    18.5783968421214, 18.1202371866188, 17.7070643203154, 17.3787250607339,
    17.9908373753160, 16.9836282783173, 18.4863334741126, 18.3041666666667,
    18.3322027608330, 18.4583333333333, 17.9000000000000, 18.6833333333333,
    19.0416666666667, 18.9583333333333, 18.9000000000000, 18.6000000000000,
    18.6000000000000, 18.5000000000000, 19.5000000000000
])

# Convertir los años en una escala numérica para el modelo (n = 1 corresponde a 1985)
n = n_years - 1985 + 1  # n va de 1 a 39

# Coeficientes dados
alpha = 16.6391
beta = 0.0576

# Valores ajustados usando el modelo lineal
n_fit = np.linspace(min(n), max(n), 300)
c_n = alpha + beta * n_fit

# Convertir n_fit de vuelta a años para la gráfica
n_fit_years = n_fit + 1985 - 1

# Crear la gráfica
plt.figure(figsize=(12, 6))
plt.plot(n_years, y_n, 'o', label='Datos originales')
plt.plot(n_fit_years, c_n, 'r-', label='Modelo lineal ajustado')
plt.xlabel('Año')
plt.ylabel('$y_n$')
plt.title('Ajuste por mínimos cuadrados con modelo lineal')
plt.legend()
plt.grid(True)

# Configurar las marcas del eje x para mostrar todos los años
plt.xticks(n_years, rotation=45)

plt.tight_layout()
plt.show()
```



Determinar los estimadores por mínimos cuadrados de α, β y γ para el modelo lineal $c(n) = \alpha + \beta n + \gamma n^2$,

Para ajustar el modelo cuadrático $c(n) = \alpha + \beta n + \gamma n^2$ a los datos proporcionados, utilizaremos el método de **mínimos cuadrados**. Este método minimiza la suma de los cuadrados de las diferencias entre los valores observados y_n y los valores predichos por el modelo $c(n)$.

Plantear las ecuaciones normales

Las ecuaciones normales se derivan de minimizar la suma de los residuos al cuadrado y son:

$$\begin{cases} N\alpha + \sum n\,\beta + \sum n^2\,\gamma = \sum y_n \\ \sum n\,\alpha + \sum n^2\,\beta + \sum n^3\,\gamma = \sum n y_n \\ \sum n^2\,\alpha + \sum n^3\,\beta + \sum n^4\,\gamma = \sum n^2 y_n \end{cases}$$

Donde N es el número total de datos.

Calcular las sumas necesarias

Calculamos las siguientes sumas utilizando los datos proporcionados:

1. $\sum 1 = N = 39$

Cálculo de $\sum n$

$$\sum n = 1 + 2 + 3 + \cdots + 39 = \frac{N(N+1)}{2} = \frac{39 \times 40}{2} = 780$$

Cálculo de $\sum n^2$

$$\sum n^2 = 1^2 + 2^2 + 3^2 + \cdots + 39^2 = \frac{N(N+1)(2N+1)}{6} = \frac{39 \times 40 \times 79}{6} = 20{,}540$$

Cálculo de $\sum n^3$

$$\sum n^3 = \left(\frac{N(N+1)}{2}\right)^2 = \left(\frac{39 \times 40}{2}\right)^2 = 780^2 = 608{,}400$$

Cálculo de $\sum n^4$

$$\sum n^4 = \frac{N(N+1)(2N+1)(3N^2+3N-1)}{30} = \frac{39 \times 40 \times 79 \times (3 \times 39^2 + 3 \times 39 - 1)}{30} = 19{,}221{,}332$$

Cálculo de $\sum y_n$

$$\sum y_n = 16.7698 + 16.7535 + \cdots + 19.5000 = 693.8530$$

Cálculo de $\sum ny_n$

$$\sum ny_n = (1)(16.7698) + (2)(16.7535) + \cdots + (39)(19.5000) = 14{,}161.6280$$

Cálculo de $\sum n^2y_n$

$$\sum n^2y_n = (1^2)(16.7698) + (2^2)(16.7535) + \cdots + (39^2)(19.5000) = 376{,}978.1834$$

Formar las ecuaciones normales

Sustituimos las sumas calculadas en las ecuaciones normales:

$$\begin{cases} 39\alpha + 780\beta + 20{,}540\gamma = 693.8530 \\ 780\alpha + 20{,}540\beta + 608{,}400\gamma = 14{,}161.6280 \\ 20{,}540\alpha + 608{,}400\beta + 19{,}221{,}332\gamma = 376{,}978.1834 \end{cases}$$

Resolver el sistema de ecuaciones

Expresamos el sistema en forma matricial:

$$\begin{bmatrix} 39 & 780 & 20{,}540 \\ 780 & 20{,}540 & 608{,}400 \\ 20{,}540 & 608{,}400 & 19{,}221{,}332 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} 693.8530 \\ 14{,}161.6280 \\ 376{,}978.1834 \end{bmatrix}$$

Utilizando: wolframcloud.com

"MatrixSolve[{{39, 780, 20540}, {780, 20540, 608400}, {20540, 608400, 19221332}}, {693.8530, 14161.6280, 376978.1834}]"

La solución del sistema de ecuaciones es:

$$\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} 16.7299 \\ 0.0443 \\ 0.0003 \end{bmatrix}$$

Esto significa que los valores aproximados de las incógnitas son:

- $\alpha \approx 16.73$
- $\beta \approx 0.044$
- $\gamma \approx 0.00033$

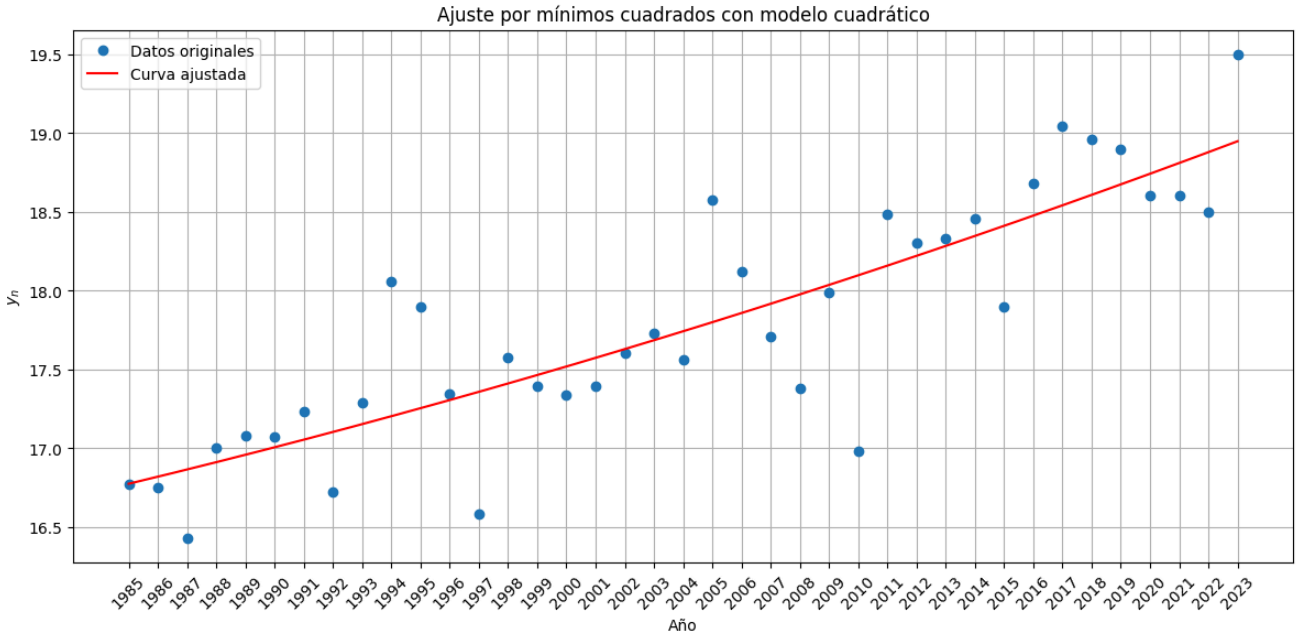
EL modelo caudrático es:

$$\boxed{c(n) = 16.73 + 0.044\,n + 0.00033\,n^2}$$

Gráfica del modelo cuadrático:

In [4]:
```
import numpy as np
import matplotlib.pyplot as plt

# Datos proporcionados (años)
n_years = np.array([
    1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992, 1993, 1994,
    1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004,
    2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014,
    2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023
])

# Valores de y_n correspondientes a cada año
y_n = np.array([
    16.7697998689058, 16.7535210287069, 16.4303837448411, 17.0019075421182,
    17.0781717025325, 17.0750907751858, 17.2337521544100, 16.7220320690133,
    17.2885635579424, 18.0565008899168, 17.8967229535935, 17.3416218250177,
    16.5835240099788, 17.5748688868413, 17.3942670461079, 17.3372485725621,
    17.3956255093168, 17.6046132918079, 17.7282641642286, 17.5633017403916,
    18.5783968421214, 18.1202371866188, 17.7070643203154, 17.3787250607339,
    17.9908373753160, 16.9836282783173, 18.4863334741126, 18.3041666666667,
    18.3322027608330, 18.4583333333333, 17.9000000000000, 18.6833333333333,
    19.0416666666667, 18.9583333333333, 18.9000000000000, 18.6000000000000,
    18.6000000000000, 18.5000000000000, 19.5000000000000
])

# Convertir los años en una escala numérica para el modelo (n = 1 corresponde a 1985)
n = n_years - 1985 + 1  # n va de 1 a 39

# Coeficientes obtenidos
alpha = 16.73
```

```
beta = 0.044
gamma = 0.00033



# Valores ajustados usando el modelo cuadrático
n_fit = np.linspace(min(n), max(n), 300)
c_n = alpha + beta * n_fit + gamma * n_fit**2

# Convertir n_fit de vuelta a años para la gráfica
n_fit_years = n_fit + 1985 - 1

# Crear la gráfica
plt.figure(figsize=(12, 6))
plt.plot(n_years, y_n, 'o', label='Datos originales')
plt.plot(n_fit_years, c_n, 'r-', label='Curva ajustada')
plt.xlabel('Año')
plt.ylabel('$y_n$')
plt.title('Ajuste por mínimos cuadrados con modelo cuadrático')
plt.legend()
plt.grid(True)

# Configurar las marcas del eje x para mostrar todos los años
plt.xticks(n_years, rotation=45)

plt.tight_layout()
plt.show()
```



Conclusiones

Conclusión sobre los resultados obtenidos con el modelo cuadrático y el modelo lineal

Al analizar los datos proporcionados mediante ajustes de modelos, se emplearon dos enfoques: un modelo cuadrático y un modelo lineal. A continuación, se presentan las conclusiones derivadas de ambos modelos.

Modelo Cuadrático

El modelo cuadrático utilizado fue:

$$c(n) = \alpha + \beta n + \gamma n^2$$

Con los coeficientes:

- $\alpha = 16.73$
- $\beta = 0.044$
- $\gamma = 0.00033$

Observaciones:

- **Flexibilidad en el Ajuste:** El término cuadrático (γn^2) permite que el modelo capture posibles curvaturas en la tendencia de los datos a lo largo del tiempo.
- **Adaptabilidad a Cambios de Tendencia:** Si los datos presentan una aceleración o desaceleración en la tendencia, el modelo cuadrático es capaz de reflejar estos cambios más efectivamente que un modelo lineal.

- **Ajuste Visual:** Al graficar este modelo junto con los datos originales, es probable que la curva ajustada siga más de cerca las fluctuaciones en los datos, especialmente en períodos donde la tendencia no es estrictamente lineal.

Modelo Lineal

El modelo lineal empleado fue:

$$c(n) = \alpha + \beta n$$

Con los coeficientes:

- $\alpha = 16.6391$
- $\beta = 0.0576$

Observaciones:

- **Simplicidad:** Este modelo asume una relación directa y constante entre el tiempo y los valores de y_n, lo que simplifica la interpretación y el cálculo.
- **Tendencia Promedio:** La línea ajustada representa una tendencia promedio de los datos a lo largo de los años sin considerar posibles variaciones en la tasa de cambio.
- **Menor Sobreajuste:** Al tener menos parámetros, el modelo lineal reduce el riesgo de sobreajustar los datos, lo cual es beneficioso cuando se busca generalizar resultados.

Conclusión Final

La elección entre el modelo cuadrático y el lineal depende de varios factores:

- **Ajuste vs. Simplicidad:** Si el modelo cuadrático proporciona un ajuste significativamente mejor, podría ser la opción preferida. Sin embargo, si la mejora es marginal, el modelo lineal es preferible por su simplicidad.
- **Naturaleza de los Datos:** Si los datos muestran una tendencia que cambia de manera no lineal a lo largo del tiempo, el modelo cuadrático es más adecuado. Si la tendencia es aproximadamente lineal, el modelo lineal es suficiente.
- **Objetivos del Análisis:** Para pronósticos a corto plazo o si se prioriza la interpretabilidad, el modelo lineal es más conveniente. Para análisis detallados de tendencias y cambios en la tasa de crecimiento, el modelo cuadrático ofrece más insights.

Referencias

Briceyda B. Delgado. (2024). Unidad 5: Método de Mínimos Cuadrados. Google Drive. https://drive.google.com/drive/folders/1FWsmQgoYbq-JhPXgEQlfYtycvq7PLz0u

Dot Physics. (2023). Least Squares Fit and Graphing in Python. YouTube. https://www.youtube.com/watch?v=f8WjtjtebVU

OpenAI. (2024). ChatGPT (GPT-4) LLM. https://chat.openai.com/

Wolfram Research. (2023). Wolfram GPT. https://www.wolfram.com/

Symbolab. (n.d.). Graphing calculator. Symbolab. https://www.symbolab.com/graphing-calculator

---

URL al repositorio de codigo de este documento (Github):

https://github.com/davidlobolobo/data_science/blob/main/Math/MCDI_MAT_David_Rodriguez_Tarea_5.ipynb

---