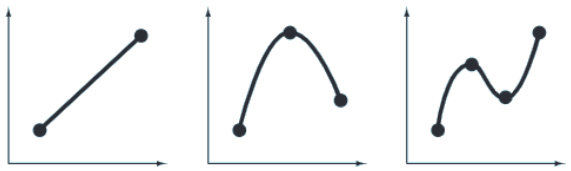
# Marco Teórico

**Interpolación.**

Con frecuencia se tiene que estimar valores intermedios entre datos definidos por puntos. El método más común es la interpolación.

En análisis numérico, se denomina interpolación a la obtención de nuevos puntos partiendo del conocimiento de un conjunto de puntos **[1]**.

Aunque hay uno y sólo un polinomio de a lo más de n-ésimo grado que se ajusta a $n+1$ puntos, existe una gran variedad de formas de interpolar estos puntos.



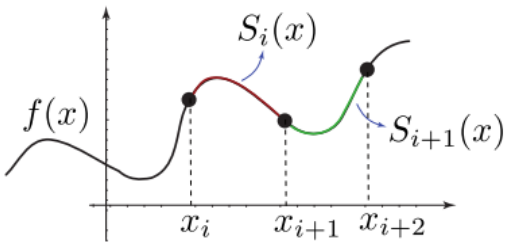
Por ejemplo, para ocho puntos se puede obtener un polinomio de séptimo grado. Podría agrupar todas las curvas sugeridas por los puntos, no obstante, hay casos donde estas funciones llevarían a resultados erróneos (redondeo, puntos lejanos, etc). Un procedimiento alternativo consiste en colocar polinomios de grado inferior en subconjuntos de los datos. Tales polinomios conectores se denominan trazadores o splines.

**Trazadores cúbicos.**

Las curvas de tercer grado empleadas para unir cada par de puntos se llaman trazadores cúbicos. Se construyen de tal forma que las conexiones adyacentes resulten suaves **[2]**.

Se interpola $f(x)$, de la cual se dan un número $n$ de puntos $(x, f(x))$, por los que tendrá que pasar el trazador ($S(x)$). Definiremos cada trazador por cada par de puntos adyacentes ($n-1$ de trazadores). Tendrán la forma de **[3]**:

$$S_{n-1}(x) = a_{n-1} + b_{n-1}(x - x_{n-1}) + c_{n-1}(x - x_{n-1})^2 + d_{n-1}(x - x_{n-1})^3$$



Cuatro incógnitas por cada intervalo ($a_{n-1}, b_{n-1}, c_{n-1}\ y\ d_{n-1}$). Las condiciones por las cuales se forman las ecuaciones para encontrarlas son:

- $S_{n-1}(x_{n-2}) = f(x_{n-2})$ y $S_{n-1}(x_{n-1}) = f(x_{n-1})$.
- $S_{n-1}'(x_{n-1}) = S_n'(x_{n-1})$ y $S_{n-1}''(x_{n-1}) = S_n''(x_{n-1})$

Además, pudiendo tomar dos caminos para poder terminar de determinar las ecuaciones:

- **Natural:** $S_1''(x_0) = S_{n-1}''(x_n) = 0$.
- **Sujeto:** $S_1'(x_0) = f'(x_0)$ y $S_{n-1}'(x_n) = f'(x_n)$.

La mayor diferencia entre esos dos tipos de splines cúbicos radica en que los naturales se reduce la oscilación en los extremos, se utiliza cuando no hay información adicional en los extremos.

El uso de $x - x_{n-1}$ en la forma de los trazadores se debe a facilitar el cálculo de los coeficientes (variable trasladada respecto a un punto base). Representa la separación entre puntos consecutivos.

**Escoliosis idiopática.**

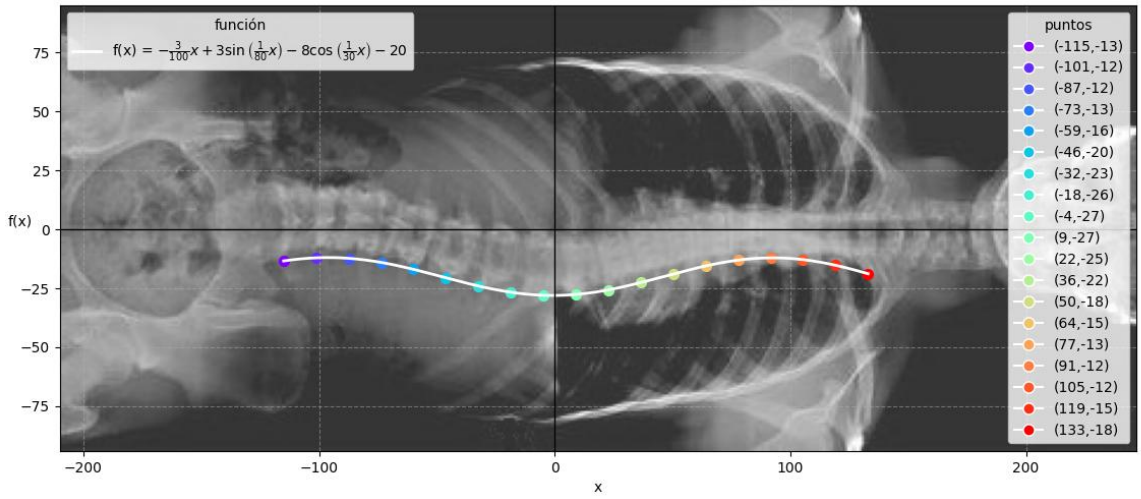
La escoliosis idiopática es el encorvamiento lateral de la columna vertebral. Forma más frecuente de escoliosis y se observa en el $2 - 4\%$ de los niños entre $10 - 16\ años$ **[4]**.

La mayoría de las curvas son convexas a la derecha en la región torácica y a la izquierda en la región lumbar.

El examen radiográfico debe incluir proyecciones anteroposterior y lateral de la columna en bipedestación. La curvatura se cuantifica en grados (el método Cobb).

# *Examen*

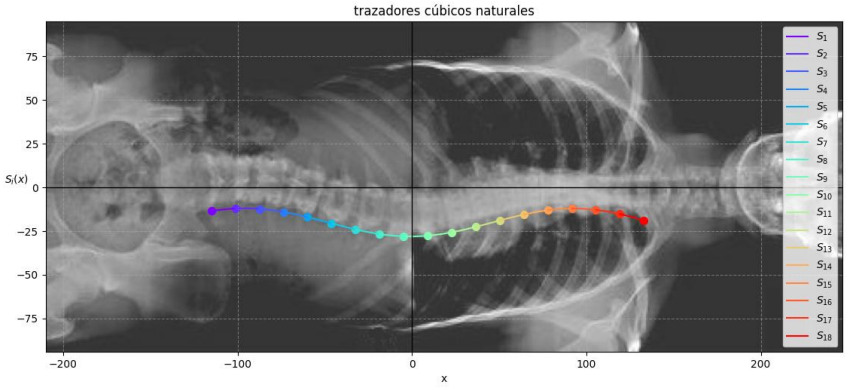
**Función.**




La función del contorno de columna se encontró a prueba y error. Elegimos 19 puntos equidistantes desde T1 hasta S1 para interpolarlos.

**Trazadores cúbicos naturales.**

```
S_1(x) = -13.362292854835303+(0.11918496089602676)*(x--133.0)+(0.0)*(x--133.0)^2+(-0.0001221368142016084 4)*(x--133.0)^3
S_2(x) = -12.039625137415 05+(0.04963030700106 6395)*(x--115.0)+(-0.0050483216536664 79)*(x--115.0)^2+(2.620301901160081e-05)*(x--115.0)^3
S_3(x) = -12.245607078817377+(-0.07455686670284056)*(x--101.22222222222223)+(-0.003965263534520312)*(x--101.22222222222223)^2+(1.427357236639149e-05)*(x--101.22222222222223)^3
S_4(x) = -13.988218687164279+(-0.17569337118348788)*(x--87.44444444444444)+(-0.003375289210042796 6)*(x--87.44444444444444)^2+(4.076449494500951e-05)*(x--87.44444444444444)^3
S_5(x) = -16.942989164255472+(-0.2454867155537609 8)*(x--73.66666666666666)+(-0.0016903567523157378)*(x--73.66666666666666)^2+(4.757866916324787e-05)*(x--73.66666666666666)^3
S_6(x) = -20.521689119426565+(-0.2649702639489021)*(x--59.888888888888886)+(0.0002762282397651732)*(x--59.888888888888886)^2+(4.737329863141924e-05)*(x--59.888888888888886)^3
S_7(x) = -23.996054955148484+(-0.2303804250547542 6)*(x--46.111111111111114)+(0.0022343245831971696)*(x--46.111111111111114)^2+(3.639929576246011e-05)*(x--46.111111111111114)^3
S_8(x) = -26.650851176335028+(-0.1480836449976326 6)*(x--32.333333333333333)+(0.0037388288080455216)*(x--32.333333333333333)^2+(1.793090308750326e-05)*(x--32.333333333333333)^3
S_9(x) = -27.934487053447 98+(-0.0348468228102505 6)*(x--18.555555555555543)+(0.0044799728023289 89)*(x--18.555555555555543)^2+(-4.48309033054661 6e-06)*(x--18.555555555555543)^3
S_10(x) = -27.575903376694963+(0.0860482797068476)*(x--4.777777777777715)+(0.004294671735333063)*(x--4.777777777777715)^2+(-2.612443809887085 7e-05)*(x--4.777777777777715)^3
S_11(x) = -25.643429722167973+(0.1895129615930768 7)*(x-9.0)+(0.0032148616272464 01)*(x-9.0)^2+(-4.253320153917936e-05)*(x-9.0)^3
S_12(x) = -22.533335380858922+(0.2538783891414066)*(x-22.777777777777777)+(0.0014568226302936512)*(x-22.777777777777777)^2+(-5.0262192298640716e-05)*(x-22.777777777777777)^3
S_13(x) = -18.890366407912914+(0.2653985583231685)*(x-36.55555555555557)+(-0.0006206813180501644)*(x-36.55555555555557)^2+(-4.782128752012982e-05)*(x-36.55555555555557)^3
S_14(x) = -15.476657845668906+(0.2210620021271747)*(x-50.33333333333334)+(-0.0025972945355488626)*(x-50.33333333333334)^2+(-3.523783757541758e-05)*(x-50.33333333333334)^3
S_15(x) = -13.016112666617285+(0.12942481231206437)*(x-64.11111111111111)+(-0.0040537918219994 58)*(x-64.11111111111111)^2+(-1.681700742146642 4e-05)*(x-64.11111111111111)^3
S_16(x) = -12.046429295686071+(0.0081433522498397 8)*(x-77.88888888888891)+(-0.0047488947954200 7)*(x-77.88888888888891)^2+(1.006294588647165 6e-05)*(x-77.88888888888891)^3
S_17(x) = -12.809382619738432+(-0.1169844207813506 8)*(x-91.66666666666669)+(-0.0043329596987792 42)*(x-91.66666666666669)^2+(1.591944872563224e-05)*(x-91.66666666666669)^3
S_18(x) = -15.202045739459546+(-0.2273157012364036 8)*(x-105.44444444444446)+(-0.0036749558181976 96)*(x-105.44444444444446)^2+(8.891022140612 35e-05)*(x-105.44444444444446)^3
```

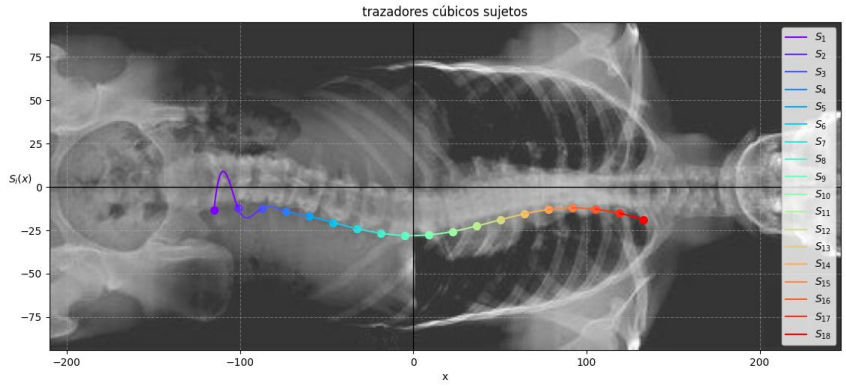



Resultados del algoritmo para trazadores cúbicos naturales.

## Trazadores cúbicos sujetos.

```
trazadores cúbicos sujetos

S_1(x) = -13.362292854835303+(9.413616107607359)*(x-133.0)+(-1.168434215822669)*(x-133.0)^2+(0.03572097335583486)*(x-133.0)^3
S_2(x) = -12.039625137415O5+(-2.440805012864832)*(x--115.0)+(0.3080326288517105)*(x--115.0)^2+(-0.009577929405299335)*(x--115.0)^3
S_3(x) = -12.245607078817377+(0.5927532660494145)*(x--101.22222222222223)+(-0.08785506586720l7)*(x--101.22222222222223)^2+(0.00258769309957
3713)*(x--101.22222222222223)^3
S_4(x) = -13.988218687164279+(-0.35449858232660936)*(x--87.44444444444444)+(0.019102915581845l7)*(x--87.44444444444444)^2+(-0.0006487811895
733392)*(x--87.44444444444444)^3
S_5(x) = -16.942989164255472+(-0.1975760037335299)*(x--73.66666666666666)+(-0.007713373587186171)*(x--73.66666666666666)^2+(0.0002323418800
2931175)*(x--73.66666666666666)^3
S_6(x) = -20.52168911942656S+(-0.277807900086705)*(x--59.888888888888886)+(0.0018900907873589464)*(x--59.888888888888886)^2+(-2.13386031451
06307e-06)*(x--59.888888888888886)^3
S_7(x) = -23.996054955148484+(-0.2269405923237738)*(x--46.111111111111114)+(0.0018018912276925O7)*(x--46.111111111111114)^2+(4.966472068011
0204e-05)*(x--46.111111111111114)^3
S_8(x) = -26.650851176335028+(-0.1490053397837515l)*(x--32.33333333333333)+(0.0038546996824703967)*(x--32.33333333333333)^2+(1.437636236283
2847e-05)*(x--32.33333333333333)^3
S_9(x) = -27.934487053447O8+(-0.03459987639675554)*(x--18.555555555555543)+(0.0044489226601341S4)*(x--18.555555555555543)^2+(-3.53035234951
5136e-06)*(x--18.555555555555543)^3
S_10(x) = -27.575903376694963+(0.085982188840787S)*(x--4.777777777777715)+(0.004303001429687S285)*(x--4.777777777777715)^2+(-2.6380849298
326543e-05)*(x--4.777777777777715)^3
S_11(x) = -25.643429722167973+(0.18953037865102654)*(x-9.0)+(0.003212592992023365)*(x-9.0)^2+(-4.246029472238795e-05)*(x-9.0)^3
S_12(x) = -22.533335380858922+(0.25387481177746907)*(x-22.777777777777777)+(0.0014575674768313271)*(x-22.777777777777777)^2+(-5.0297408366350
646e-05)*(x-22.777777777777777)^3
S_13(x) = -18.890366407912914+(0.2653954507301172)*(x-36.55555555555557)+(-0.000621392068977831S)*(x-36.55555555555557)^2+(-4.7753330066081
564e-05)*(x-36.55555555555557)^3
S_14(x) = -15.476657845668806+(0.2210780098999107)*(x-50.33333333333334)+(-0.0025951963783758l9)*(x-50.33333333333334)^2+(-3.54744513239006
2e-05)*(x-50.33333333333334)^3
S_15(x) = -13.016112666617285+(0.12936388882332006)*(x-64.11111111111111)+(-0.0040614736997637S4)*(x-64.11111111111111)^2+(-1.5938509881582
57e-05)*(x-64.11111111111111)^3
S_16(x) = -12.046429295686071+(0.00837103843208109)*(x-77.88888888888891)+(-0.0047202654415358l3)*(x-77.88888888888891)^2+(6.78556947541923
1e-06)*(x-77.88888888888891)^3
S_17(x) = -12.809382619738432+(-0.11783424202157l5)*(x-91.66666666666669)+(-0.004439795236551848)*(x-91.66666666666669)^2+(2.81504568299582
17e-05)*(x-91.66666666666669)^3
S_18(x) = -15.202045739459546+(-0.22414410245776173)*(x-105.44444444444446)+(-0.003276243020913576)*(x-105.44444444444446)^2+(4.32635653998
728e-05)*(x-105.44444444444446)^3
```




Resultados del algoritmo para trazadores cúbicos sujetos.

## Código.

Se utilizó el lenguaje de programación Python 3.11.1, además de las librerías numpy, matplotlib, sympy y os. Se realizaron dos scripts, *spine_rx.py* y *splines.py* los cuales hacen:

- **spine_rx:** se encarga de definir y graficar una función matemática en un intervalo dado. Utiliza sympy para definir una función $f(x)$ que involucra senos y cosenos, luego la convierte en una función evaluable con numpy. Se generan puntos equidistantes dentro de un rango y se grafican sobre la columna. La función se convierte a formato *LaTeX* con fracciones racionales para mejorar la legibilidad. Finalmente, la función y los puntos se grafican con matplotlib, mostrando los puntos con diferentes colores y agregando una leyenda con las coordenadas de cada uno.

- **splines:** implementa y compara dos tipos de interpolación con trazadores cúbicos: naturales y sujetos. Los trazadores cúbicos naturales imponen la condición de que la segunda derivada en los extremos sea cero, mientras que los sujetos utilizan la derivada real en los extremos para ajustar la curva. Se definen funciones para calcular los coeficientes $a$, $b$, $c$, y $d$ de los polinomios cúbicos que interpolan los puntos dados. Luego, una función splines grafica los trazadores generados sobre la misma imagen de fondo. Un bucle interactivo permite al usuario elegir entre los trazadores naturales o sujetos, mostrando las ecuaciones de cada tramo del spline y su respectiva gráfica.

Códigos: https://github.com/juan-torresf/metnum.mc.

## spine_rx.py

```python
import re,numpy as np
from fractions import Fraction
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from matplotlib.lines import Line2D
from sympy import symbols,sin,cos,lambdify,diff,latex,sympify,simplify

_,ax = plt.subplots()
ax.imshow(mpimg.imread(r'spine.jpg'),
          extent=[-210,247,-94,95],alpha=.8)

x = symbols('x')
f = 3*(sin((1/80)*x)-(1/100)*x)-4*(2*cos((1/30)*x)+5)
fx = lambdify(x,f,'numpy')
dfx = lambdify(x,diff(f,x),'numpy')

points = [(x,fx(x))for x in np.linspace(-115,133,19)]

def frac_latex(match):
    if(Fraction(float(match.group())).limit_denominator(10000).denominator==1):
        return str(Fraction(float(match.group())).limit_denominator(10000).numerator)
    else:
        return f'\\frac{{{Fraction(float(match.group())).limit_denominator(10000).numerator}}}
        {{{Fraction(float(match.group())).limit_denominator(10000).denominator}}}'

f = re.sub(r'\b0\.\d+\b',frac_latex,latex(f,mode='inline',fold_short_frac=False))

plt.plot(np.linspace(-115,133,400),fx(np.linspace(-115,133,400)),linestyle='solid',linewidth=2,
         label=fr'f(x) = {f}',color='white')

for (x_,y_),color in zip(points,plt.cm.rainbow(np.linspace(0,1,len(points)))):
    ax.scatter(x_,y_,color=color,s=50)

ax.axhline(y=0,color='black',linewidth=1)
ax.axvline(x=0,color='black',linewidth=1)
ax.set_xlim(-210,247)
ax.set_ylim(-94,95)
ax.set_xlabel('x')
ax.set_ylabel('f(x)',rotation=0,labelpad=0)
ax.grid(True,linestyle='--',alpha=.5)

flegend = ax.legend(title='función',loc='upper left')
ax.legend(handles=[Line2D([0],[0],marker='o',color='w',markerfacecolor=color,markersize=8,label=f'({int(x)},
                          {int(y)})')for(x,y),color in zip(points,plt.cm.rainbow(np.linspace(0,1,len(points))))],
          title='puntos',loc='upper right')
ax.add_artist(flegend)

plt.show()
```

## splines.py

```python
from os import name,system
from spine_rx import np,points,x,dfx as df,sympify,simplify,lambdify,plt,mpimg

def natural(points=points):

    h,alpha= [0]*(len(points)),[0]*(len(points))
    l,u,z = [0]*(len(points)+1),[0]*(len(points)+1),[0]*(len(points)+1)
    a,b,c,d = [0]*(len(points)),[0]*(len(points)),[0]*(len(points)+1),[0]*(len(points))

    h[1:] = [points[i][0]-points[i-1][0]for i in range(1,len(points))]
    alpha[1:len(points)-1] = [3*((points[i+1][1]-points[i][1])/h[i+1]-(points[i][1]-points[i-1][1])/h[i])
                              for i in range(1,len(points)-1)]

    l[1],u[1],z[1] = 1,0,0
    for i in range(2,len(points)):
        l[i] = 2*(points[i][0]-points[i-2][0])-h[i-1]*u[i-1]
        u[i] = h[i]/l[i]
        z[i] = (alpha[i-1]-h[i-1]*z[i-1])/l[i]

    c[len(points)] = 0
    for i in range(len(points)-1,0,-1):
        c[i] = z[i]-u[i]*c[i+1]
        b[i] = (points[i][1]-points[i-1][1])/h[i]-h[i]*(c[i+1]+2*c[i])/3
        d[i] = (c[i+1]-c[i])/(3*h[i])
        a[i] = points[i-1][1]

    return [a[1:],b[1:],c[1:len(points)],d[1:]]

def sujeto(points=points,df=df):

    h,alpha = [0]*(len(points)),[0]*(len(points))
    l,u,z = [0]*(len(points)+1),[0]*(len(points)+1),[0]*(len(points)+1)
    a,b,c,d = [0]*(len(points)),[0]*(len(points)),[0]*(len(points)+1),[0]*(len(points))

    h = [0]+[points[i][0]-points[i-1][0]for i in range(1,len(points))]

    alpha[0] = 3*((points[1][1]-points[0][1])/h[1]-df(points[0][0]))
    alpha[len(points)-1] = 3*(df(points[len(points)-1][0])-(points[len(points)-1][1]-points[len(points)-2][1])/
                              h[len(points)-1])

    for i in range(1,len(points)-1):
        alpha[i] = 3*((points[i+1][1]-points[i][1])/h[i+1]-(points[i][1]-points[i-1][1])/h[i])

    l[1],u[1],z[1] = 2*h[1],1/2,alpha[0]/2*h[1]
    for i in range(2,len(points)):
        l[i] = 2*(points[i][0]-points[i-2][0])-h[i-1]*u[i-1]
        u[i] = h[i]/l[i]
        z[i] = (alpha[i-1]-h[i-1]*z[i-1])/l[i]

    c[len(points)] = (alpha[len(points)-1]-h[len(points)-1]*z[len(points)-1])/(h[len(points)-1]*
                                                                               (2-u[len(points)-1]))
    for i in range(len(points)-1,0,-1):
        c[i] = z[i]-u[i]*c[i+1]
        b[i] = (points[i][1]-points[i-1][1])/h[i]-h[i]*(c[i+1]+2*c[i])/3
        d[i] = (c[i+1]-c[i])/(3*h[i])
        a[i] = points[i-1][1]

    return a[1:],b[1:],c[1:len(points)],d[1:]

def splines(trazadores):

    _,pt = plt.subplots()
    pt.imshow(mpimg.imread(r'spine.jpg'),extent=[-210,247,-94,95],alpha=.8)

    splinex = []
```

```
    splinex = []

    for i in range(1,len(trazadores[0])+1):
        splinex.append(simplify(sympify(trazadores[0][i-1])+sympify(trazadores[1][i-1])*(x-sympify(points[i-1][0]))
                                +sympify(trazadores[2][i-1])*(x-sympify(points[i-1][0]))**2+
                                sympify(trazadores[3][i-1])*(x-sympify(points[i-1][0]))**3))

    for i,(spline,color) in enumerate(zip(splinex,plt.cm.rainbow(np.linspace(0,1,len(splinex)))),start=1):
        s = lambdify(x,spline,'numpy')
        plt.plot(np.linspace(points[i-1][0],points[i][0],400),s(np.linspace(points[i-1][0],points[i][0],400)),
                 label=fr'$S_{{{i}}}$',color=color)

    for (x_,y_),color in zip(points,plt.cm.rainbow(np.linspace(0,1,len(points)))):
        plt.scatter(x_,y_,color=color,s=50)

    plt.axhline(y=0,color='black',linewidth=1)
    plt.axvline(x=0,color='black',linewidth=1)
    plt.xlim(-210,247)
    plt.ylim(-94,95)
    plt.xlabel('x')
    plt.ylabel(r'$S_i(x)$',rotation=0,labelpad=0)
    plt.grid(True,linestyle='--',alpha=.5)
    plt.title('trazadores cúbicos naturales'if(trazadores==natural())else('trazadores cúbicos sujetos'
                                                    if(trazadores==sujeto())else'trazadores cúbicos'))
    plt.legend()
    plt.show()

while 1:

    system('cls' if name == 'nt' else 'clear')

    if(bool(int(input('1:natural,0:sujeto: ')))):
        print('trazadores cúbicos naturales\n')
            print(f'S_{i}(x) = {natural()[0][i-1]}+({natural()[1][i-1]})*(x-{points[i-2][0]})+
                  ({natural()[2][i-1]})*(x-{points[i-2][0]})^2+({natural()[3][i-1]})*(x-{points[i-2][0]})^3')
        splines(natural())

    else:
        print('trazadores cúbicos sujetos\n')
        for i in range(1,len(sujeto()[0])+1):
            print(f'S_{i}(x) = {sujeto()[0][i-1]}+({sujeto()[1][i-1]})*(x-{points[i-2][0]})+
                  ({sujeto()[2][i-1]})*(x-{points[i-2][0]})^2+({sujeto()[3][i-1]})*(x-{points[i-2][0]})^3')
        splines(sujeto())

    system('cls' if name == 'nt' else 'clear')

    if(bool(int(input('1:salir,0:continuar: ')))):
        system('cls' if name == 'nt' else 'clear')
        break
```

**Aplicaciones en la ingeniería biomédica.**

Los splines se utilizan en ingeniería biomédica para modelar estructuras anatómicas, procesar imágenes médicas y analizar datos biomecánicos. Permiten representar curvas suaves en estudios de movimiento, reconstrucción 3D de órganos y ajuste de señales fisiológicas, mejorando la precisión en diagnósticos y tratamientos.

En el estudio de la escoliosis idiopática, la interpolación mediante trazadores cúbicos se emplea para modelar la curvatura de la columna vertebral a partir de puntos obtenidos en radiografías. Esto permite generar una representación matemática continua de la deformidad, facilitando su análisis y cuantificación mediante métodos como el ángulo de Cobb. Al utilizar trazadores cúbicos naturales y sujetos, se optimiza la precisión en la estimación del contorno vertebral, lo que contribuye a una mejor evaluación clínica y planificación del tratamiento, reduciendo la necesidad de exposiciones adicionales a radiación en pacientes pediátricos **[5]**.

# *Bibliografía*


**[1]** Burden, R. y Douglas, J. (2002). *Análisis Numérico.* Ediciones Paraninfo.
Recuperado el 25 de marzo de 2025 de:
https://evflores.files.wordpress.com/2014/02/analisis-numerico-richard-l-burden-7ma.pdf

**[2]** Chapra, S. y Canale, R. (2006). *Métodos numéricos para ingenieros.* McGraw Hill.
Recuperado el 25 de marzo de 2025 de:
http://artemisa.unicauca.edu.co/~cardila/Chapra.pdf

**[3]** Mata, M. (2024). *Métodos numéricos.* Facultad de Ingeniería Mecánica y Eléctrica de la Universidad Autónoma de Nuevo León.
Recuperado el 5 de febrero de 2025 de:
http://logistica.fime.uanl.mx/miguel/docs/MetNum.pdf

**[4]** Pessler, F.(2022). *Escoliosis idiopática.* Helmholtz Centre for Infection Research.
Recuperado el 26 de marzo de 2025 de:
https://www.msdmanuals.com/es/professional/pediatr%C3%ADa/trastornos-%C3%B3seos-en-ni%C3%B1os/escoliosis-idiop%C3%A1tica

**[5]** Ernst, C., Buls, N., Laumen, A. y Gompel, G. (2018). *Lowered dose full-spine radiography in pediatric patients with idiopathic scoliosis.* European Spine Journal.
Recuperado el 26 de marzo de 2025 de:
https://link.springer.com/article/10.1007/s00586-018-5561-9