Asignatura	Datos del alumno	Fecha
Métodos Numéricos I	**Apellidos:** Avilés Cahill	27/02/2024
	Nombre: Adán	

Laboratorio: El monorraíl

1. Actividad 1

Como primera aproximación, vamos a hacer una interpolación lineal entre cada pareja de puntos contiguos. Utilizaremos el comando de Matlab *polyfit* utilizando dos decimales. Obtendremos la expresión de $l_i(x)$,$i = 0, 1 \ldots, 6$ es decir:

$$l(x) = \begin{cases} l_0(x) & x \in [0, 1{,}5] \\ l_1(x) & x \in [1{,}5, 3] \\ & \vdots \\ l_6(x) & x \in [9, 10] \end{cases}$$

Adjuntamos el código en el 6.1. Lo único que hemos hecho es interpolar cada par de puntos con una recta. Si hacemos un plot de los puntos, automáticamente Matlab haría esta interpolación.

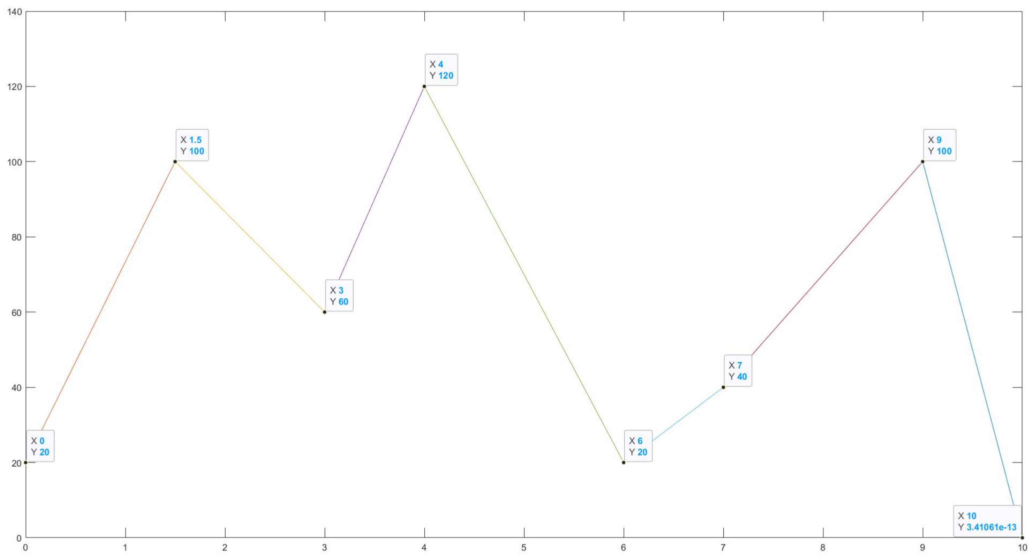

Figura 1: Representación de los múltiples l_i

2. Actividad 2

Si utilizamos interpolación lineal entre las paradas, observamos que hay cambios muy bruscos de la pendiente, o lo que es equivalente, la función $l(x)$ no es derivable. Obtengamos el polinomio de interpolación de Newton $n(x)$ de grado 7 que pasa por las ocho paradas, utilizando dos decimales, que es el siguiente:

$$\begin{aligned} p(x) =&(160x)/3 - (80x(x-3/2))/3 \\ &+ (46x(x-3)(x-3/2))/3 \\ &- (421x(x-3)(x-4)(x-3/2))/81 \\ &+ (9629x(x-3)(x-4)(x-3/2)(x-6))/6237 \\ &- 0{,}30156x(x-3)(x-4)(x-3/2)(x-6)(x-7) \\ &+ 0{,}047x(x-3)(x-4)(x-3/2)(x-6)(x-7)(x-9)) \\ &+ 20 \end{aligned} \tag{1}$$

Donde algunos valores se han redondeado para mayor visibilidad. Representaremos el polinomio y utilizaremos puntos para las ubicaciones de las paradas.

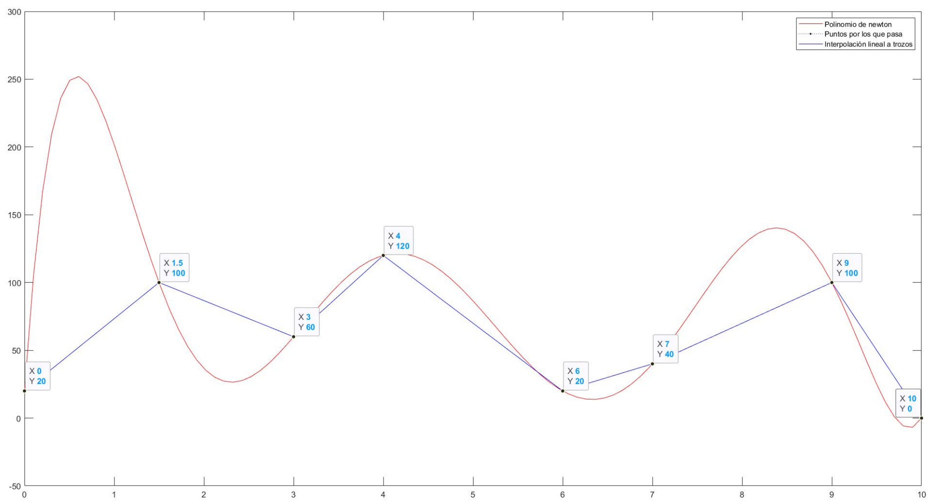

Figura 2: Representación del polinomio de Newton.

3. Actividad 3

El polinomio de grado 7 de Newton hace que para ir de la Central Nuclear al Badulaque recorramos mucha distancia. Obtendremos el polinomio de Hermite $h(x)$ de grado 15 que pasa por las ocho paradas, tomando como derivada en cada punto el valor 0 y redondeando los coeficientes al entero más próximo. El polinomio no lo mostraremos por pantalla dado que nos ocuparía demasiado, y no seria de valor, pues podemos calcularlo en Matlab utilizando 6.3. Representaremos el polinomio y utilizaremos puntos para las ubicaciones de las paradas. Si redondeamos los coeficientes, tendremos

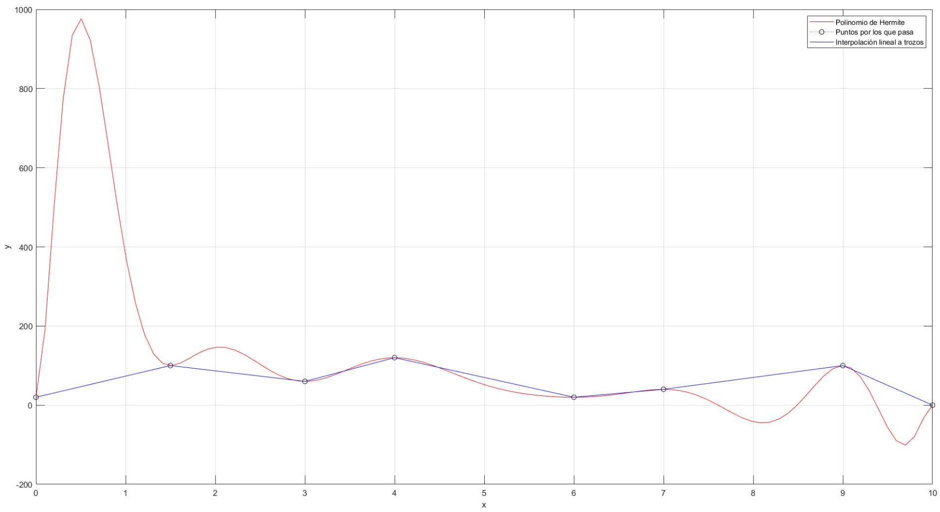

Figura 3: Representación del polinomio de Hermite.

el polinomio:

$$\begin{aligned} p(x) = & x^{12} - 14x^{11} + 137x^{10} - 962x^9 + 4935x^8 \\ & - 18368x^7 + 48839x^6 - 89841x^5 + 107883x^4 - 75580x^3 + 23329x^2 + 20 \end{aligned} \tag{2}$$

Cuya representación es mucho menos exacta.

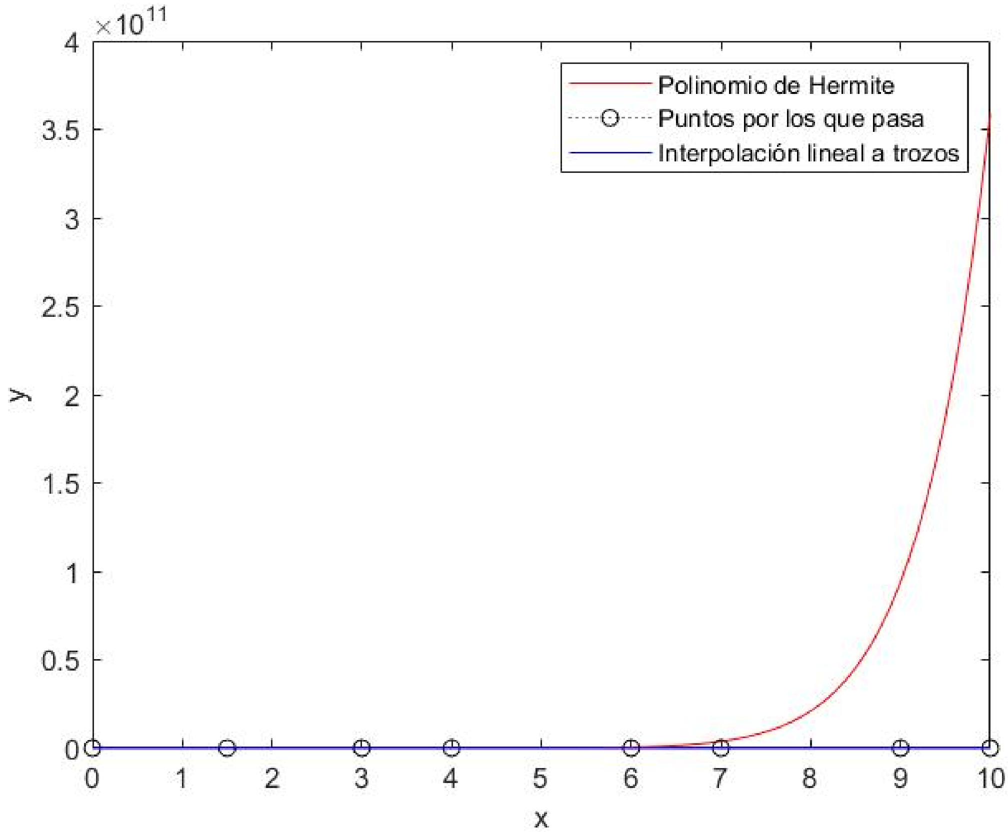

Figura 4: Representación del polinomio de Hermite redondeado.

4. Actividad 4

Parece que utilizar el polinomio de Hermite empeora todavía más el trayecto entre la Central Nuclear y el Badulaque. Así que vamos a utilizar splines naturales cúbicos s(x).

$$s(x) = \begin{cases} s_0(x) = 84{,}43x - 13{,}81x^3 + 20 & x \in [0, 1{,}5] \\ s_1(x) = 33{,}54(x - \frac{3}{2})^3 - 62{,}18(x - \frac{3}{2})^2 - 8{,}86x - 573{,}28 & x \in [1{,}5, 3] \\ & \vdots \\ s_6(x) = 23{,}71(x-9)^3 - 71{,}12(x-9)^2 - 52{,}59x + 573{,}28 & x \in [9, 10] \end{cases}$$

Cuyo código podemos obtener en 6.4, y su representación será:

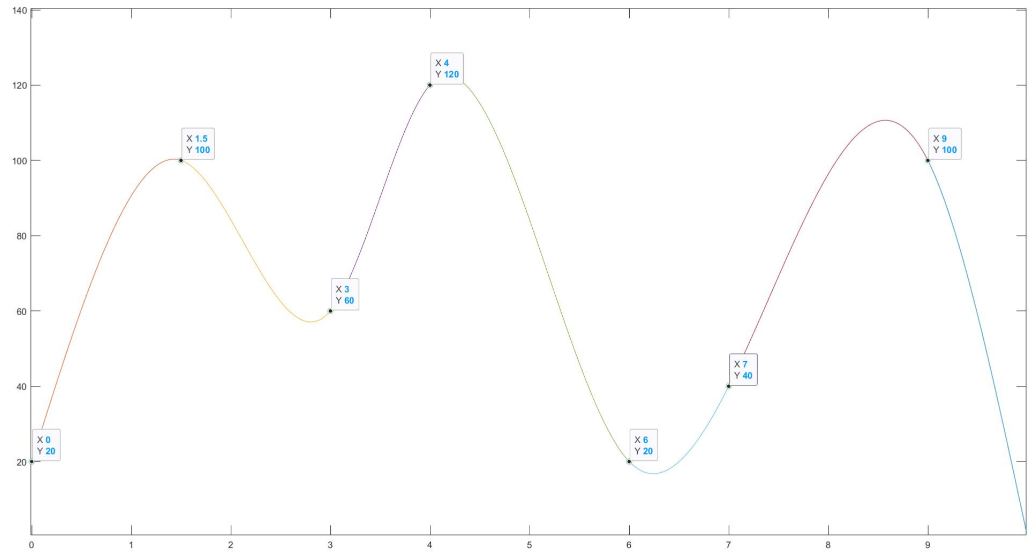

Figura 5: Representación mediante splines.

5. Actividad 5

En último lugar, vamos a representar en una sola gráfica todas las interpolaciones. Podemos observar como la mejor interpolación es la conseguida por splines, pero también la más costosa computacionalmente. La interpolación de Hermite podría haber dado mejores resultados si tuviéramos la evaluación de las derivadas más exactas, no todas iguales a cero, en este caso es superada por la interpolación de Newton. En este caso, quizá podríamos habernos çonformadoçon la interpolación de Newton, pese a que el recorrido entre la Central Nuclear y el Badulaque sea tan distante.

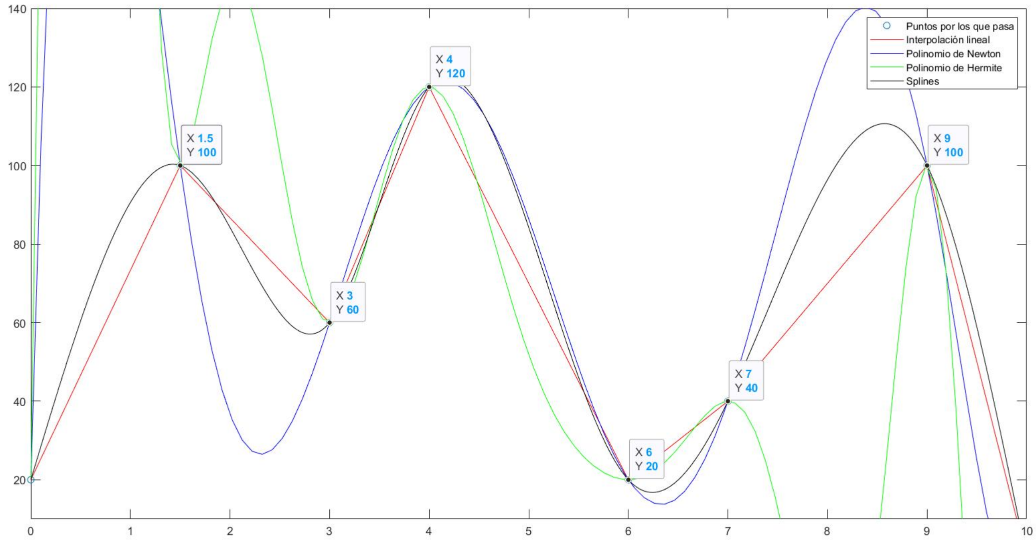

Figura 6: Comparación interpolaciones

6. Anexos: Código

6.1. Código Ejercicio 1

.

```
% Definimos primero x e y
x = [0 1.5 3 4 6 7 9 10];
y = [20 100 60 120 20 40 100 0];
% Calculamos los polinomios lineales para cada "trozo"
% Con x([1,2]) obtenemos los valores con la posicion 1 y 2 de x
p_lineal_1 = polyfit(x([1,2]),y([1,2]),1);
p_lineal_2 = polyfit(x([2,3]),y([2,3]),1);
p_lineal_3 = polyfit(x([3,4]),y([3,4]),1);
p_lineal_4 = polyfit(x([4,5]),y([4,5]),1);
p_lineal_5 = polyfit(x([5,6]),y([5,6]),1);
p_lineal_6 = polyfit(x([6,7]),y([6,7]),1);
```

```
p_lineal_7 = polyfit(x([7,8]),y([7,8]),1);

% Definimos los y1,y2... para graficar. Usamos polyval para
% tener el valor del polinomio p_lineal en la coordenada.
y1 = polyval(p_lineal_1,x([1,2]));y2 = polyval(p_lineal_2,x([2,3]));
y3 = polyval(p_lineal_3,x([3,4]));y4 = polyval(p_lineal_4,x([4,5]));
y5 = polyval(p_lineal_5,x([5,6]));y6 = polyval(p_lineal_6,x([6,7]));
y7 = polyval(p_lineal_7,x([7,8]));

figure;
plot(x,y,'.');
hold on;
plot(x([1,2]),y1); plot(x([2,3]),y2); plot(x([3,4]),y3);
plot(x([4,5]),y4); plot(x([5,6]),y5); plot(x([6,7]),y6);
plot(x([7,8]),y7);
hold off;
```

6.2. Código Ejercicio 2

.

```
% Definimos primero xi e fi y x como variable simbolica.
xi = [0 1.5 3 4 6 7 9 10];
fi = [20 100 60 120 20 40 100 0];
syms x
% Calculamos b y el polinomio p
[b,p] = polinomioNewton(xi,fi);

% Tomamos 100 puntos para representar el polinomio
xaxis = linspace(0,10);
yaxis = double(subs(p,x,xaxis));

% Representamos el polinomio en rojo, los puntos como estrellas
% Y la interpolación lineal automática que hace matlab en azul.
plot(xaxis,yaxis,'red')
hold on
plot(xi,fi, '.:black');
plot(xi,fi, 'blue')
legend('Polinomio de newton','Puntos por los que pasa', ...
    'Interpolación lineal a trozos');

% Definimos la funcion polinomioNewton
function [b,p] = polinomioNewton(xi,fi)
   % Siendo n el grado del polinomio
   % Creamos una matriz de ceros (n+1)*(n+1)
   % Damos a la  primera columna el valor de los f(x)
   syms x
```

```
    n = length(xi);
    F = zeros(n, n);
    F(:, 1) = fi;
    %Inicializo p con b0, vx con 1
    p = F(1,1);
    vx = 1;
    %Sean j las columnas, i las filas.
    % Rellenaré de 1 a n-j+1 filas. La primera columna (cuando j = 1)
    % tiene n filas con valores. La segunda columna (cuando j=2) tiene n-1
    % valores ... la última columna (j=n) tiene solo una fila con valores.
    %
    for j=2:n
        for i=1:n-j+1
            F(i,j) = (F(i+1,j-1)-F(i,j-1))/(xi(i+j-1)-xi(i));
        end
        %En cada iteracion tendré vx = 1, vx=1*(x-x0), vx=(x-x0)*(x-x1)...
        vx = vx*(x-xi(j-1));
        p = p+F(1,j)*vx;
    end
    % Los bs serán la primera fila
    b = F(1,:);
end
```

6.3. Código Ejercicio 3

.

```
    % Definimos primero xi e fi y x como variable simbolica.
xi = [0 1.5 3 4 6 7 9 10];
fi = [20 100 60 120 20 40 100 0];
dfi = zeros(size(xi));
syms x
% Calculamos b y el polinomio p
[b,p] = polinomioHermite(xi,fi,dfi);

% Tomamos 100 puntos para representar el polinomio
xaxis = linspace(0,10);
yaxis = double(subs(p,x,xaxis));

% Representamos el polinomio en rojo, los puntos como estrellas
% Y la interpolación lineal automática que hace matlab en azul.
plot(xaxis,yaxis,'red'), hold on, grid, xlabel('x'), ylabel('y'),
plot(xi,fi, 'o:black'), plot(xi,fi, 'blue'),
legend('Polinomio de Hermite','Puntos por los que pasa', ...
    'Interpolación lineal a trozos');

% Definimos la funcion polinomioHermite
function [b,p] = polinomioHermite(xi,fi,dfi)
    % Siendo n el grado del polinomio
    % Creamos una matriz de ceros (n+1)*(n+1)
    % Damos a la  primera columna el valor de los f(x)
    format shortg;
    syms x;
```

```
%Creamos el vector z
n = length(xi);
z = zeros(1,2*n);
z(1:2:end) = xi;
z(2:2:end) = xi;

%Creamos la matriz F
F = zeros(2*n, 2*n);
% Meto en las columnas pares e impares fi, pues z2i = z2i+1 = xi
F(1:2:end, 1) = fi;
F(2:2:end, 1) = fi;
%Inicializo p con b0, vx con 1
p = F(1,1);
vx = 1;
%Sean j las columnas, i las filas.
% Rellenaré de 1 a n-j+1 filas. La primera columna (cuando j = 1)
% tiene n filas con valores. La segunda columna (cuando j=2) tiene n-1
% valores ... la última columna (j=n) tiene solo una fila con valores.
for j=2:2*n
    for i=1:2*n-j+1
        F(i,j) = (F(i+1,j-1)-F(i,j-1))/(z(i+j-1)-z(i));
    end
    % Para j=2, cambio las impares por dfi
    if j == 2
        F(1:2:end, 2) = dfi;
    end
    vx = vx*(x-z(j-1));
```

```
        p = p+F(1,j)*vx;
    end
    % Los bs serán la primera fila
    b = F(1,:);
end
```

6.4. Código Ejercicio 4

.

```
    % Definimos primero xi e fi y x como variable simbolica.
xi = [0 1.5 3 4 6 7 9 10];
fi = [20 100 60 120 20 40 100 0];
syms x
% Calculamos b y el polinomio p
[ai,bi,ci,di,p] = splineCubicoNatural(xi,fi);

% Tomamos 100 puntos para representar el polinomio
xaxis_1 = linspace(xi(1),xi(2));xaxis_2 = linspace(xi(2),xi(3));
xaxis_3 = linspace(xi(3),xi(4));xaxis_4 = linspace(xi(4),xi(5));
xaxis_5 = linspace(xi(5),xi(6));xaxis_6 = linspace(xi(6),xi(7));
xaxis_7 = linspace(xi(7),xi(8));
% Definimos los y1,y2... para graficar.
yaxis_1 = double(subs(p(1), x, xaxis_1));
yaxis_2 = double(subs(p(2), x, xaxis_2));
yaxis_3 = double(subs(p(3), x, xaxis_3));
yaxis_4 = double(subs(p(4), x, xaxis_4));
yaxis_5 = double(subs(p(5), x, xaxis_5));
yaxis_6 = double(subs(p(6), x, xaxis_6));
yaxis_7 = double(subs(p(7), x, xaxis_7));

plot(xi,fi,'o');
hold on;
plot(xaxis_1,yaxis_1);plot(xaxis_2,yaxis_2);
plot(xaxis_3,yaxis_3);plot(xaxis_4,yaxis_4);
plot(xaxis_5,yaxis_5);plot(xaxis_6,yaxis_6);
```

```
plot(xaxis_7,yaxis_7);
hold off;

function [ai,bi,ci,di,p] = splineCubicoNatural(xi,fi)
    format shortg
    n = length(xi);
    h = zeros(1,n-1);
    for i = 1:n-1
        h(i) = xi(i+1)-xi(i);
    end
    ai  = fi;
    %Relleno la diagonal inferior, superior y principal
    %Relleno también b
    dI = zeros(n-1,1);
    dS = zeros(n-1,1);
    dP = ones(n,1);
    b = zeros(n,1);
    for i = 2:n-1
        dI(i-1) = h(i-1);
        dS(i) = h(i);
        dP(i) = 2*(h(i)+h(i-1));
        b(i) = 3*((ai(i+1)-ai(i))/h(i) - (ai(i)-ai(i-1))/h(i-1));
    end
    % Aplico crout para sacar ci
    ci  = Crout(dP ,dS ,dI ,b);
    bi = zeros(n-1,1);
    di = zeros(n-1,1);
    % Obtengo bi y di
```

```
    for i=1:n-1
        bi(i) = (ai(i+1)-ai(i))/h(i) - h(i)*(2*ci(i)+ci(i+1))/3;
        di(i) = (ci(i+1)-ci(i))/(3*h(i));
    end

    syms x;
    % En p tendré una lista con cada spline
    p = zeros(1,n);
    syms p;
    for i = 1:n-1
        p(i) = ai(i)+bi(i)*(x-xi(i))+ci(i)*(x-xi(i))^2+di(i)*(x-xi(i))^3;
    end
end

function x = Crout(dP ,dS ,dI ,b)
    % La función x=Crout(dP,dS,dI,b) obtiene la ...
    % solución del sistema Ax=b utilizando el ...
    % algoritmo de Crout. dP, dS y dI son las ...
    % diagonales principal , superior e inferior de A.
    n = length(dP);
    % 1. Obtención de las matrices L y U tales que A = LU
    l(1) = dP(1);
    u(1) = dS(1)/l(1);
    for i =2:n-1
        l(i) = dP(i)-dI(i-1)*u(i -1) ;
        u(i) = dS(i)/l(i);
    end
    l(n) = dP(n)-dI(n-1)*u(n-1);
```

```
    % 2. Solución del sistema Lz = b


    z(1) = b(1)/l(1) ;
    for i = 2:n
        z(i) = (1/l(i))*(b(i)-dI(i-1)*z(i-1));
    end
    % 3. Solución del sistema Ux = z
    x(n) = z(n);
    for i = n-1:-1:1
        x(i) = z(i)-u(i)*x(i+1);
    end
    x=x(:);
end
```