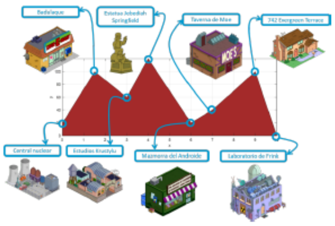
MONORRAÍL DE SPRINGFIELD

Tenemos el siguiente problema:

Descripción del laboratorio

En la ciudad de Springfield se va a instalar un monorraíl. Tras una extensa votación entre los ciudadanos, se ha decidido que debe tener paradas en los lugares marcados en la figura, cuya ubicación se refleja en la tabla.



	Posición	
	x	y
Central nuclear	0	20
Badulaque	1.5	100
Estudios Krustylu	3	60
Estatua Jebediah Springfield	4	120
Mazmorra del Androide	6	20
Taverna de Moe	7	40
742 Evergreen Terrace	9	100
Laboratorio de Frink	10	0

Sujeta a condiciones de interpolación que se especificará en cada problema del presente laboratorio.

PLANTEAMIENTO DEL PROBLEMA

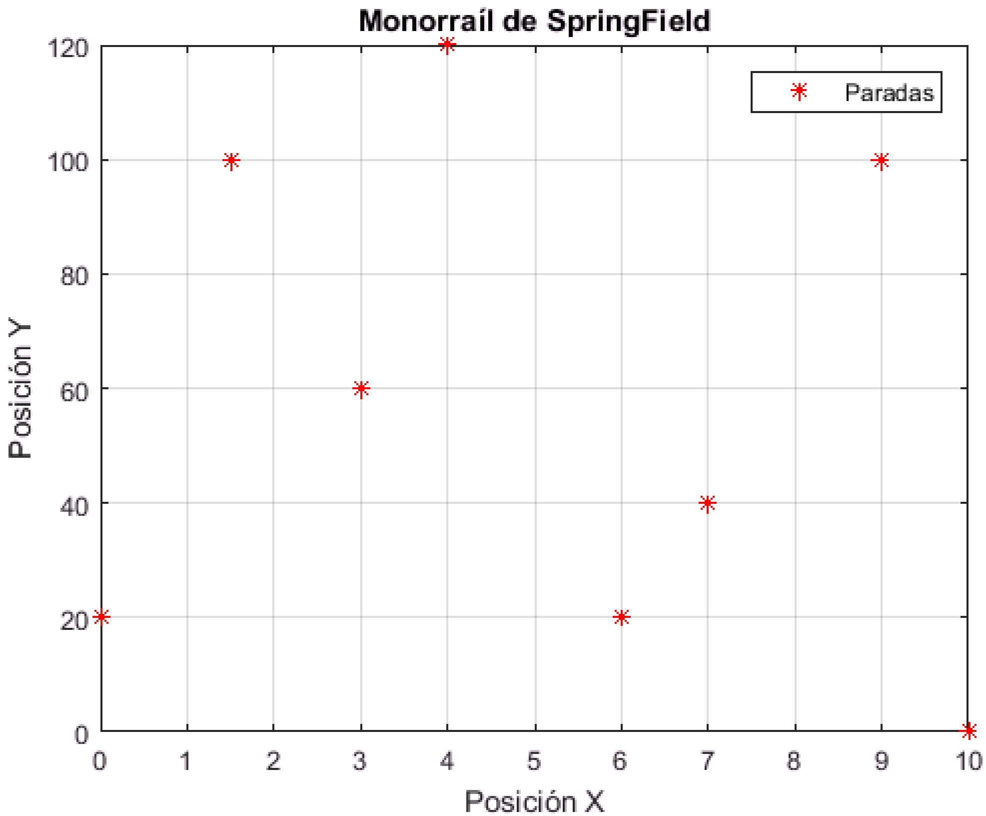
Limpiamos el Workspace y el Command Window:

```
clearvars
clear all
close all
clc
```

Introducimos las posiciones que muestra el problema. Con ello encontramos las posiciones discretas de

cada parada.

```
x = [0 1.5 3 4 6 7 9 10];
y = [20 100 60 120 20 40 100 0];
plot(x,y,'g*')
grid on
xlabel('Posición X')
ylabel('Posición Y')
title('Monorraíl de SpringField')
legend('Paradas')
```



PROBLEMA 1: INTERPOLACIÓN LINEAL

```
n = length(x);
```

Es claro el grado de cada polinomio de la interpolación:

```
ginterp = 1;
nesp = 30;
for i = 1:n-1
    if i > 1
        xinterpaux = xinterp;
        yinterpaux = yinterp;
    end
```

Usamos la función predeterminada de MATLAB para encontrar los parámetros sujetos a cada ecuación de recta:

```
    l(i,:) = polyfit(x(i:i+1),y(i:i+1),ginterp);
```

Introducimos la función correspondiente como anónima:

```
    linterp = @(x) l(i,1)*x+l(i,2);
    xinterp = linspace(x(i),x(i+1),nesp);
    yinterp = linterp(xinterp);
    if i > 1
        xinterp = [xinterpaux xinterp];
        yinterp = [yinterpaux yinterp];
    end
```

Se escriben las ecuaciones, con coeficientes a dos decimales:

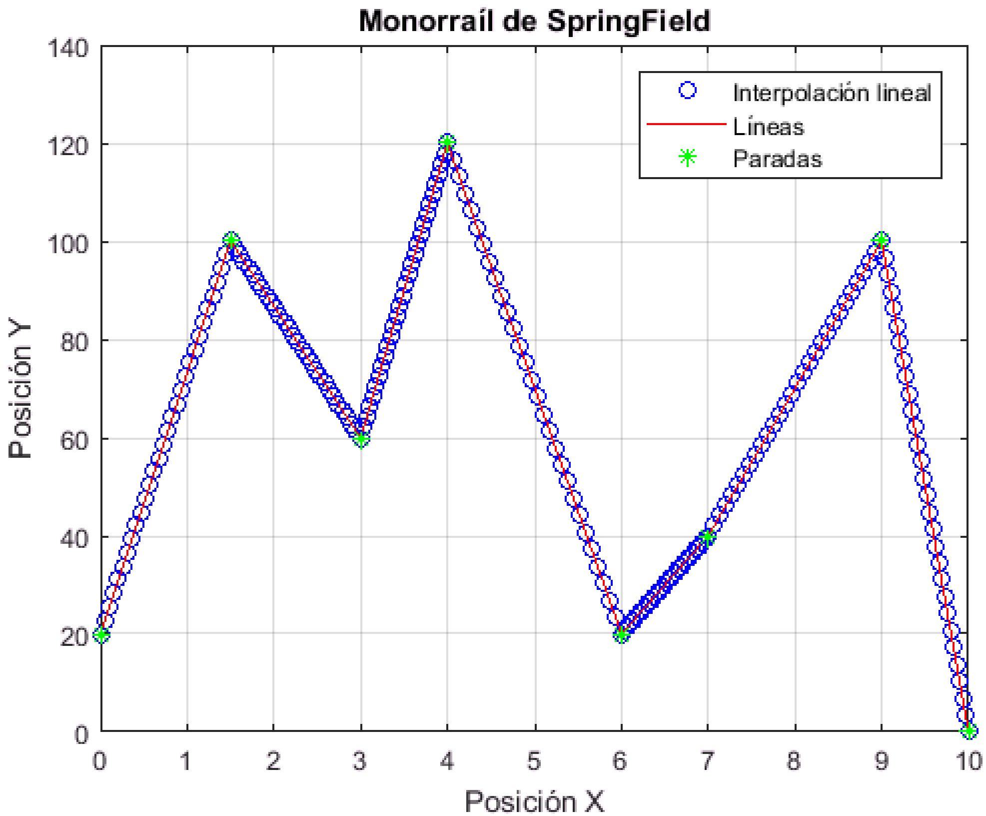
```
    fprintf('Polinomio %d:\n',i)
    fprintf('l(i) = (%.2f)x+(%.2f)',l(i,1),l(i,2))
end
```

```
Polinomio 1:
l(i) = (53.33)x+(20.00)
Polinomio 2:
l(i) = (-26.67)x+(140.00)
Polinomio 3:
l(i) = (60.00)x+(-120.00)
Polinomio 4:
l(i) = (-50.00)x+(320.00)
Polinomio 5:
l(i) = (20.00)x+(-100.00)
Polinomio 6:
l(i) = (30.00)x+(-170.00)
Polinomio 7:
l(i) = (-100.00)x+(1000.00)
```

GRÁFICA (INTERPOLACIÓN LINEAL)

Tenemos:

```
plot(xinterp,yinterp,'bo',x,y,'r',x,y,'g*')
grid on
xlabel('Posición X')
ylabel('Posición Y')
title('Monorraíl de SpringField')
legend('Interpolación lineal','Líneas','Paradas')
```




```
xli = xinterp;
yli = yinterp;
```

OBSERVACIÓN:

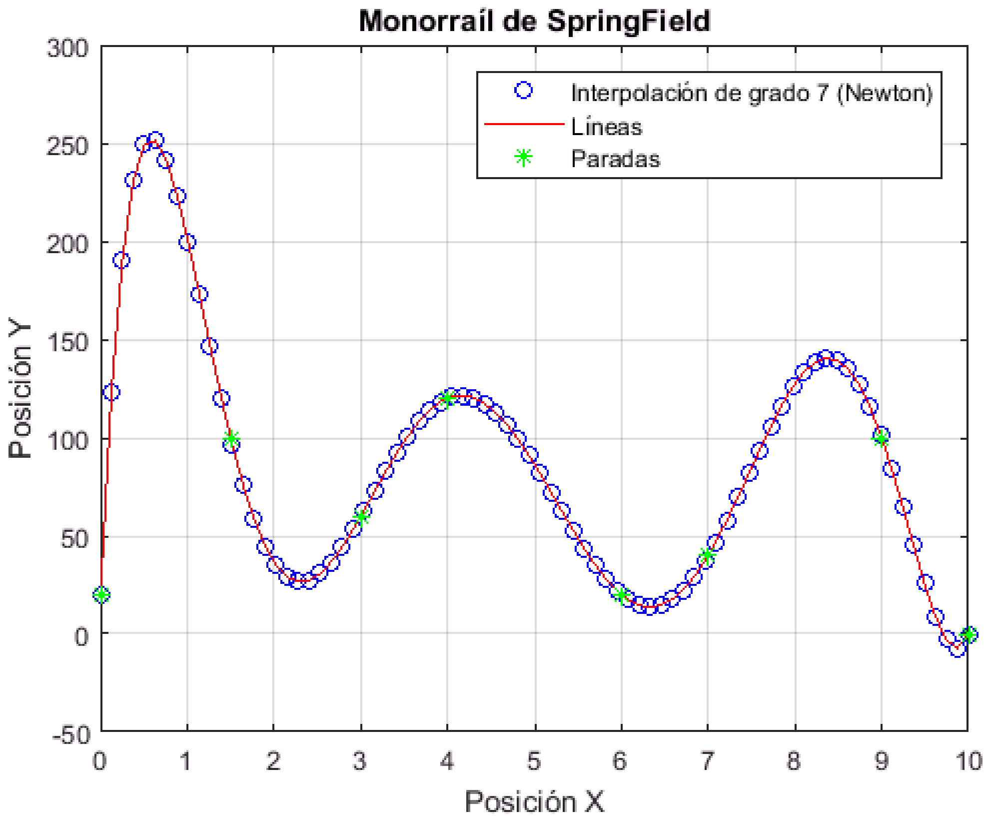
Es claro que se tiene, en conjunto, una función no derivable con puntos picos en cada parada. Asimismo, es claro que el recorrido de cada parada consecutiva es el menor posible.

PROBLEMA 2: INTERPOLACIÓN POR NEWTON

Dado que tenemos un conjunto de 8 puntos, tanto en abscisas como en ordenadas, encontraremos un único polinomio interpolable según Newton. Este será de grado 7.

GRÁFICA (INTERPOLACIÓN POR NEWTON)

```
[xn,yn] = InterpNewton(x,y);
```



OBSERVACIÓN:

Es claro que se tiene, en conjunto, una función un poco más suave. Sin embargo, se nota que el recorrido entre las dos primeras paradas es innecesariamente extenso.

PROBLEMA 3: INTERPOLACIÓN DE HERMITE

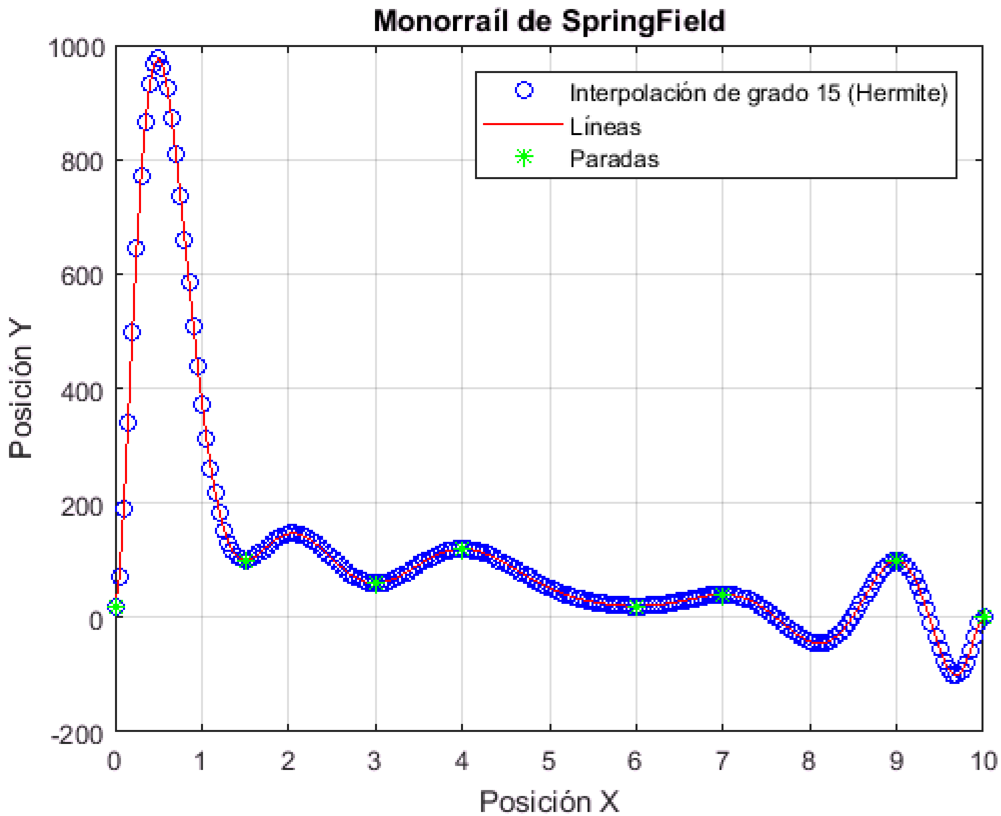
Este polinomio de interpolación será de grado 2*(#datos-1)+1 = 15. Con lo que se tendrá curvas incluso más suaves. La teoría de Hermite trabaja con las derivadas en cada punto; en el caso particular nos pide que sean nulas. Por tanto, hacemos:

```
dydx = zeros(1,n);
```

E interpolando:

GRÁFICA (INTERPOLACIÓN POR HERMITE)

```
[hp,xh,yh] = InterpHermite(x,y,dydx);
```




```
fprintf('Polinomio de interpolación por Hermite:\n\nP(x) = (%.2f)x^15+(%.2f)x^14+(%.2f)x^13+(%
```

```
Polinomio de interpolación por Hermite:

P(x) = (-0.00)x^15+(0.00)x^14+(-0.05)x^13+(1.00)x^12+(-13.97)x^11+
(136.57)x^10+(-962.29)x^9+(4935.14)x^8+(-18368.29)x^7+(48838.65)x^6
+(-89840.63)x^5+(107883.35)x^4+(-75580.13)x^3+(23328.98)x^2+(0.00)x^1+(20.00)
```

<u>OBSERVACIÓN:</u>

Se observa que se trata de un polinomio de grado 12, al reducirse a 2 cifras decimales. Es fácil ver que el recorrido entre las dos primeras paradas es incluso más ridiculamente extenso que el anterior.

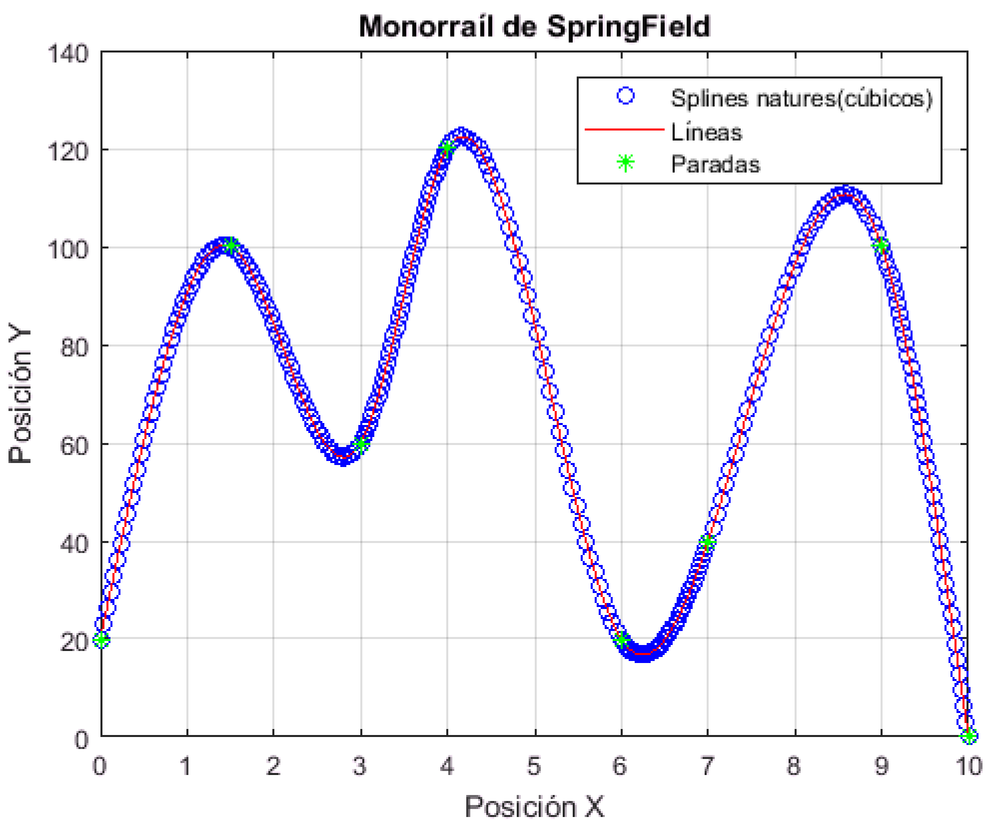
PROBLEMA 4: INTERPOLACIÓN POR SPLINES CÚBICOS

Escogemos, en particular, la interpolación por polinomios naturales de tercer grado:

```
[xsl3,ysl3] = InterpSpline3([x;y]);
```

```
Polinomio 1 de la interpolación por spline cúbico
p1(x):(20.00)+(84.43)*(x-(0.00))+(0.00)*(x-(0.00))^2+(-13.82)*(x-(0.00))^3


Polinomio 2 de la interpolación por spline cúbico
p2(x):(100.00)+(-8.86)*(x-(1.50))+(-62.19)*(x-(1.50))^2+(33.54)*(x-(1.50))^3


Polinomio 3 de la interpolación por spline cúbico
p3(x):(60.00)+(30.99)*(x-(3.00))+(88.76)*(x-(3.00))^2+(-59.75)*(x-(3.00))^3


Polinomio 4 de la interpolación por spline cúbico
```

```
p4(x):(120.00)+(29.26)*(x-(4.00))+(-90.49)*(x-(4.00))^2+(25.43)*(x-(4.00))^3


Polinomio 5 de la interpolación por spline cúbico
p5(x):(20.00)+(-27.52)*(x-(6.00))+(62.11)*(x-(6.00))^2+(-14.58)*(x-(6.00))^3


Polinomio 6 de la interpolación por spline cúbico
p6(x):(40.00)+(52.94)*(x-(7.00))+(18.36)*(x-(7.00))^2+(-14.91)*(x-(7.00))^3


Polinomio 7 de la interpolación por spline cúbico
p7(x):(100.00)+(-52.59)*(x-(9.00))+(-71.12)*(x-(9.00))^2+(23.71)*(x-(9.00))^3
```

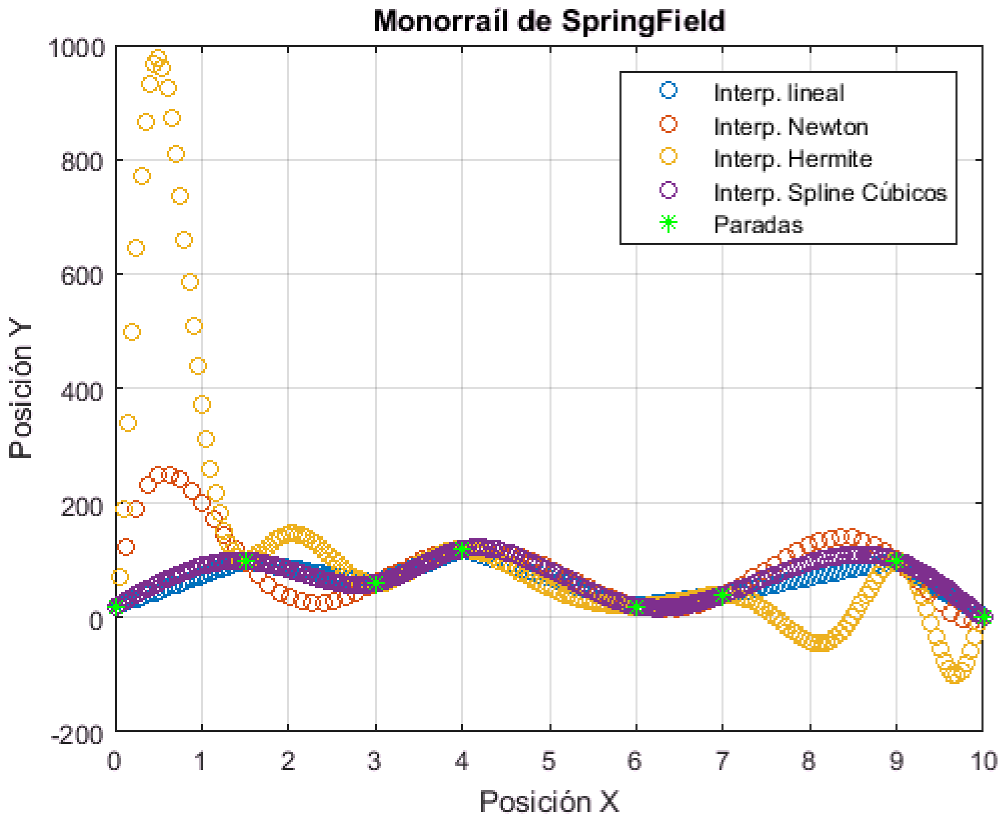


Obteniéndose un conjunto de 7 polinomios. Observamos inmediatamente que este método de interpolación es más cómodo que los anteriores.

PROBLEMA 5: SUPERPOSICIÓN DE GRÁFICAS

Agrupamos lo anterior:

```
plot(xli,yli,'o',xn,yn,'o',xh,yh,'o',xsl3,ysl3,'o',x,y,'g*')
grid on
xlabel('Posición X')
ylabel('Posición Y')
title('Monorraíl de SpringField')
legend('Interp. lineal','Interp. Newton','Interp. Hermite','Interp. Spline Cúbicos','Paradas')
```




```
function [xint,yint] = InterpNewton(x,y)
    n = length(x);
    b = zeros(n);
    b(:,1) = y(:);
    nesp = 80;
    for j = 2:n
        for i = 1:n-j+1
            b(i,j) = (b(i+1,j-1)-b(i,j-1))/(x(i+j-1)-x(i));
        end
    end
    xint = linspace(min(x),max(x),nesp);
    prodint = ones(1,length(xint));
    yint = b(1,1);
    for i = 1:n-1
        prodint = prodint.*(xint-x(i));
        yint = yint+b(1,i+1)*prodint;
    end
    plot(xint,yint,'bo',xint,yint,'r',x,y,'g*')
    grid on
    xlabel('Posición X')
    ylabel('Posición Y')
    title('Monorraíl de SpringField')
    legend('Interpolación de grado 7 (Newton)','Líneas','Paradas')
end
function [hp,xint,yint] = InterpHermite(x,y,yp)
    n = length (x);
    z = zeros (1,2*n);
    f = zeros (1,2*n);
    z(1:2:2*n-1) = x;
    z(2:2:2*n) = x;
```

```
    f(1) = y(1);
    f(3:2:2*n-1) = (y(2:n)-y(1:n-1))./(x(2:n)-x(1:n-1));
    f(2:2:2*n) = yp;
    for i = 3:2*n
        f(i:2*n) = (f(i:2*n)-f(i-1:2*n-1))./(z(i:2*n)-z(1:2*n-i+1));
    end
    hp = zeros (1,2*n);
    p = [1];
    for i = 1:2*n
        hp = hp + f(i)*[zeros(1,2*n-i) p];
        p = conv(p, [1 -z(i)]);
    end
    nesp = 200;
    xint = linspace(min(x),max(x),nesp);
    yint = polyval(hp,xint);
    plot(xint,yint','bo',xint,yint','r',x,y,'g*')
    grid on
    xlabel('Posición X')
    ylabel('Posición Y')
    title('Monorraíl de SpringField')
    legend('Interpolación de grado 15 (Hermite)','Líneas','Paradas')
end
function [xint,yint] = InterpSpline3(X)
    n=length(X(1,:));
    for i=1:n
        a(i)=X(2,i);
    end
    for i=1:n-1
        h(i)=X(1,i+1)-X(1,i);
    end
    for i=2:n-1
        alfa(i)=3/h(i)*(a(i+1)-a(i))-3/h(i-1)*(a(i)-a(i-1));
    end
    l(1)=1;
    mu(1)=0;
    z(1)=0;
    for i=2:n-1
        l(i)=2*(X(1,i+1)-X(1,i-1))-h(i-1)*mu(i-1);
        mu(i)=h(i)/l(i);
        z(i)=(alfa(i)-h(i-1)*z(i-1))/l(i);
    end
    l(n)=1;
    z(n)=0;
    c(n)=0;
    for i=n-1:-1:1
        c(i)=z(i)-mu(i)*c(i+1);
        b(i)=(a(i+1)-a(i))/h(i)-h(i)*(c(i+1)+2*c(i))/3;
        d(i)=(c(i+1)-c(i))/(3*h(i));
    end
    nesp = 40;
    for i=1:n-1
        if i > 1
            xpre = xint;
            ypre = yint;
        end
        xint=linspace(X(1,i),X(1,i+1),nesp);
        yint=a(i)+b(i)*(xint-X(1,i))+c(i)*(xint-X(1,i)).^2+d(i)*(xint-X(1,i)).^3;
        fprintf('Polinomio %d de la interpolación por spline cúbico %d:',i)
        fprintf('\n')
        fprintf('p%d(x):(%.2f)+(%.2f)*(x-(%.2f))+(%.2f)*(x-(%.2f))^2+(%.2f)*(x-(%.2f))^3\n',i,
        fprintf('\n')
```

```
        if i > 1
            xint = [xpre xint];
            yint = [ypre yint];
        end
        fprintf('\n')
    end
    plot (xint,yint,'bo',xint,yint,'r',X(1,:),X(2,:),'g*');
    grid on
    xlabel('Posición X')
    ylabel('Posición Y')
    title('Monorraíl de SpringField')
    legend('Splines natures(cúbicos)','Líneas','Paradas')
end
```