Tania Ortiz Mendez

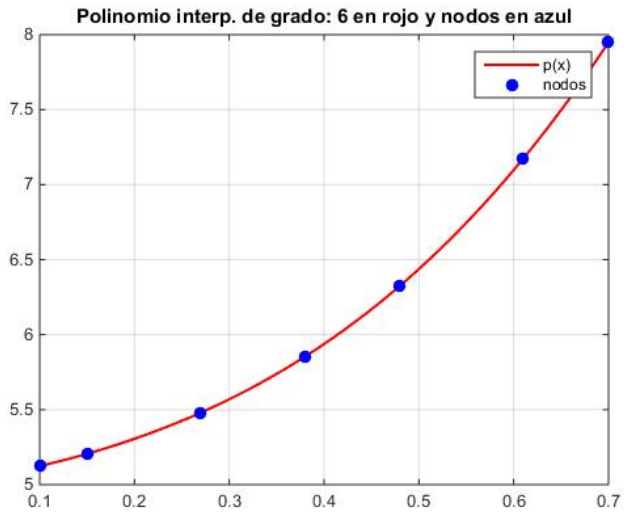
EJERCICIOS

INTERPOLACIÓN

EJERCICIO 1

a)

```
opt=0;
xn=[0.1 0.15 0.27 0.38 0.48 0.61 0.7];
f0=[5.1234 5.2057 5.4801 5.8541 6.3255 7.1737 7.9493];
[xd,pd] =polinterdifdiv(xn,opt,f0)
```



```
function [xd,pd] =polinterdifdiv(xn,opt,f0)
```

En esta función se calcula el polinomio de interpolación de grado **n** utilizando **(n+1) nodos x0,x1,..xn** y empleando diferencias divididas.

Se distinguen los dos casos siguientes:

- Que se conozca la expresión analítica de la función **f(x)** a interpolar
- Que solo se conozca el valor de la función en los nodos **xi.**

Se calculan los valores de tal polinomio en un vector de puntos **xd** uniformemente espaciados con una distancia entre ellos de **0.01.** Tales valores del polinomio se obtienen en el vector **pd.** También se realiza un gráfico de tal polinomio, representando precisamente los puntos **(xd,pd).**

Argumentos de entrada:

xn: Vector con los **(n+1)** nodos

opt: Este parámetro tomará el valor 1 si se conoce la expresión analítica de **f** o cualquier otro valor si solo se conocen los valores de **f** en los nodos

f0: Identificador de la función **f(x)** a interpolar(si se conoce su expresión analítica), o bien el vector conteniendo el valor de **f** en los **n+1** nodos.

Argumentos de salida:

xd: Vector conteniendo los puntos donde evaluamos el polinomio de interpolación (va a ser un vector de puntos entre **min(xn)** y **max(xn)** uniformemente espaciados con una distancia entre ellos de **0.01**. A veces los nodos **xn** pueden no estar ordenados de menor a mayor)

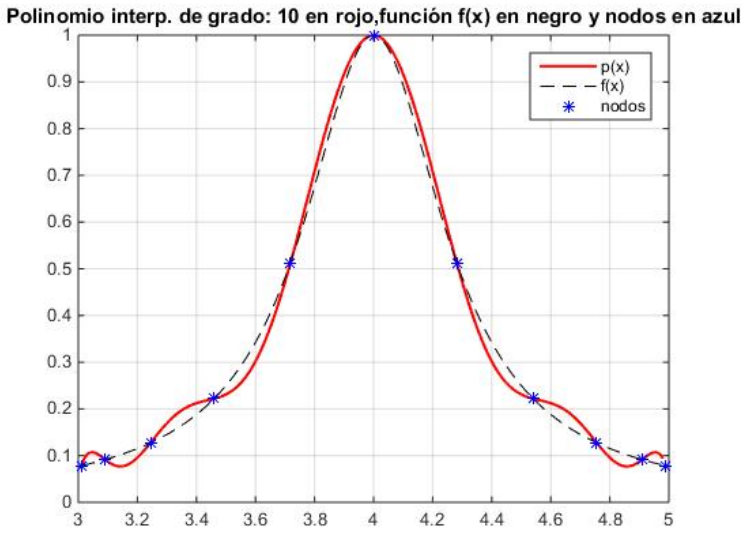
pd: vector conteniendo los valores del polinomio de interpolación en **xd**.

```
s=length(xn); El polinomio interpolador será de grado s-1
F=[]; Inicializamos el vector F, de s componentes donde se van
a calcular las diferencias divididas, empezando siempre por la
última componente y siguiendo hacia arriba
 if (opt==1)
   F=f0(xn);
  else F=f0;
 end
Empezamos construyendo la tabla de diferencias divididas (la
primera componente de F,f(x0),la mantenemos, porque es el
término independiente del polinomio)
for k=1:s-1
 F(s:-1:k+1)=(F(s:-1:k+1)-F(s-1:-1:k))./(xn(s:-1:k+1)-xn(s-k:-1:1))
end
En F tenemos los coeficientes a0,a1,...an del polinomio de
interpolación. Ahora evaluamos en xd tal polinomio mediante la
variante de Horner y lo hacemos llamando a la subfuncion
varHorner
xd=min(xn):0.01:max(xn);
 Vector xd donde vamos a evaluar el polinomio de interpolación
(Como los nodos guardados en x pueden no estar ordenados en
forma creciente, se calcula su máximo y su mínimo)
pd=varHorner(F,xn,xd);
Dibujamos el polinomio obtenido y los nodos
```

```
plot(xd,pd,'r','Linewidth',1.5);
grid on
hold on
 if (opt==1)
    plot(xd,f0(xd),'--k');
    Dibujamos la función f(x) en línea discontinua
    plot(xn,f0(xn),'b*') Dibujo de nodos
    title(['Polinomio interp. de grado: ',num2str(s-1),' en
      rojo,función f(x)',' en negro y nodos en azul'])
    legend('p(x)','f(x)','nodos')

  else
    plot(xn,f0,'.b','MarkerSize',25) Dibujo de nodos
    title(['Polinomio interp. de grado: ',num2str(s-1),' en
    rojo y nodos en azul'])
    legend('p(x)','nodos')
 end
hold off


  function b=varHorner(p,x,x1)
Esta subfunción evalua,por la variante de Horner, en el vector
x1 el polinomio a0+a1*(x-x0)+a2*(x-x0)*(x-x1)+...+an*(x- x0)..
...(x-xn-1), cuyos coeficientes
(a0,a1,...an) están en el vector p y los nodos (x0,x1,...,xn)
en x
    b=[];
    s=length(p);
    b=p(s);
     for i=s-1:-1:1
       b=b.*(x1-x(i))+p(i);
     end
  end
end
```

b)

```
opt=1;
a=3;
b=5;
grado=10;
x = Chebyshev(a,b,grado)

f0=@(x)(1./(1+12.*(x-4).^2));
[xd,pd] = polinterdifdiv(x,opt,f0)
```

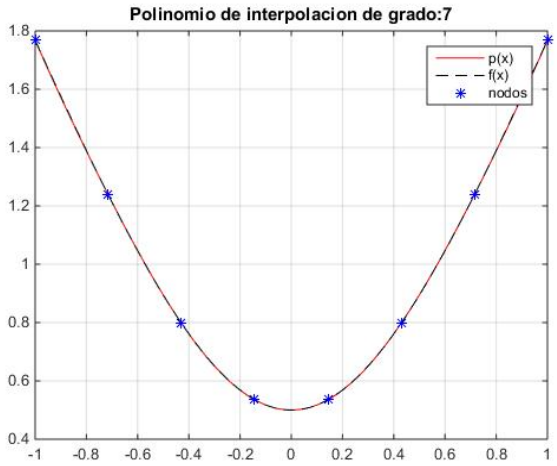


```
function x = Chebyshev(a,b,grado_pol)
    c = (b-a)/2;
    d = (b+a)/2;
    n=grado_pol+1; a0+a1grg Para sacar los nodos de Chebyshev
se usan (grado +1) iraciones para sacar los nodos necesarios
     for k = 1:n
        t(k) = cos((2*(k-1)+1)*pi/(2*n));
        x(k) = c*t(k)+d;
     end
end
```

EJERCICIO 2

a)

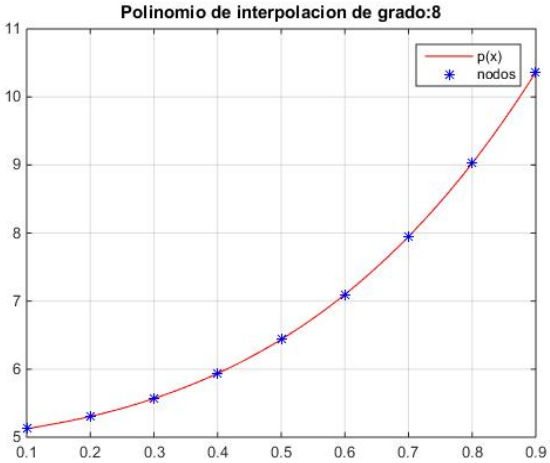
```
f0=@(x) ((4*x.^2)+1)./(2.*sqrt(1+x.^2));
a=(-1);
b=(1);
n=7;
opt=1;
[p1,x1]= NewtonDifFinProgresivas(n,a,b,opt,f0)
```



---

b)

```
opt=0;
a=0.1;
b=0.9;
f0=[5.1234 5.3057 5.5687 5.9378 6.4370 7.0978 7.9493 9.0253 10.3627];
n=length(f0)-1;
[p1,x1]= NewtonDifFinProgresivas(n,a,b,opt,f0)
```



```
function [ p1,x1 ]=NewtonDifFinProgresivas(n,a,b,opt,f0)
h=(b-a)/n;
x=a:h:b;
s=length(x);
F=[];

  if (opt==1)
    F=f0(x);
  else
    F=f0;
  end

  for k=1:s-1
    F(s:-1:k+1)=(F(s:-1:k+1)-F(s-1:-1:k));
  end

x1=a:0.01:b;
t1=(x1-a)./h;
p1=Horner(F,t1);

plot(x1,p1,'r')
grid on
hold on

  if (opt==1)
    plot(x1,f0(x1),'--k')
    plot(x,f0(x),'b*')
    title(['Polinomio de interpolacion de grado:',num2str(s-1)])
    legend('p(x)','f(x)','nodos')
  else
    plot(x,f0,'b*')
    title(['Polinomio de interpolacion de grado:',num2str(s-1)])
    legend('p(x)','nodos')
  end

hold off
```

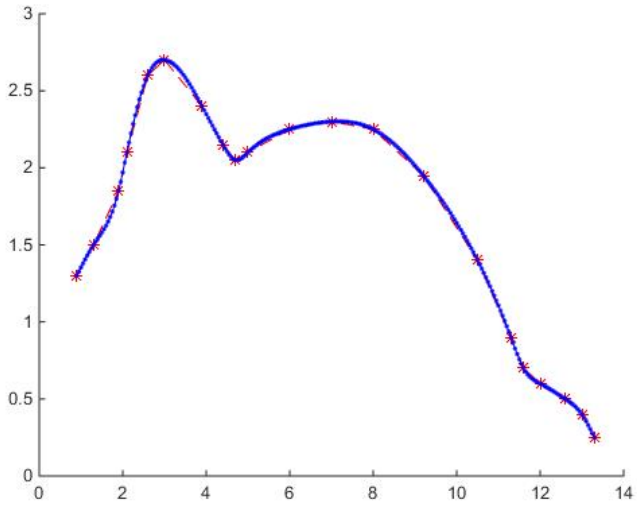
```
    function [b] = Horner ( F,t1 )
       b=[];
       s=length(F);
       b=F(s);
       for i=s-1:-1:1
         b=b.*((t1-(i-1))./((i-1)+1))+F(i);
       end
    end
end
```

EJERCICIO 3

a)

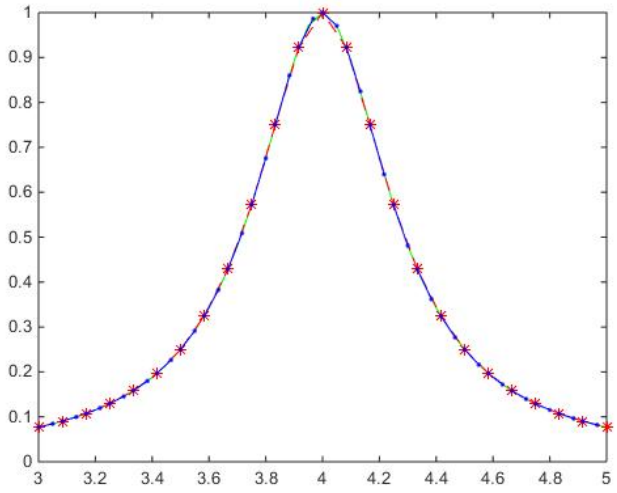
```
x = [0.9 1.3 1.9 2.1 2.6 3.0 3.9 4.4 4.7 5.0 6.0 7.0 8.0 9.2
10.5 11.3 11.6 12.0 12.6 13.0 13.3 ];

f0 = [ 1.3 1.5 1.85 2.1 2.6 2.7 2.4 2.15 2.05 2.1 2.25 2.3
2.25 1.95 1.4 0.9 0.7 0.6 0.5 0.4 0.25];

opt=0;

[ ppr ] = SplineCubico1( x, opt, f0);
```

Natural

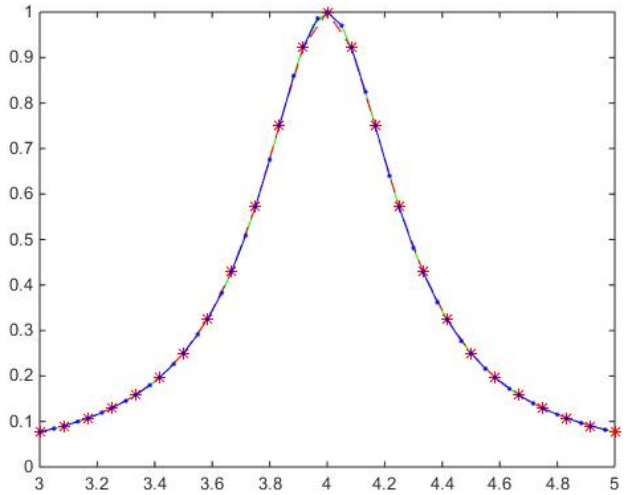


b)

```
f0 = @(x)1./((1+12*(x-4).^2));

a=3;

b=5;

opt=1;

h = (b-a)/24;

x=a:h:b;

[ ppr ] = SplineCubico1( x, opt, f0)
```




```
f0 = @(x)1./((1+12*(x-4).^2));
df = @(x)(-24*(x-4))./(1+12*(x-4).^2).^2;
a=3;
b=5;
opt=1;
h = (b-a)/24;
x=a:h:b;
ypr = [df(a) df(b)];
[ ppr ] = SplineCubico1( x, opt, f0, ypr )
```



```
function [ ppr ] = SplineCubico ( x, opt, f0, ypr)
s=length(x);
h(1:s-1)=x(2:s)-x(1:s-1);
  if opt==1
    y=f0(x);
    xn=x(1):0.05:x(end);
    plot(xn,f0(xn),'g')
  else y=f0;
  end;

  if nargin==3
    [M, ppr] = SplineNatural1( h,y,s );
  elseif nargin==4
    [ M, ppr ] = SplineCondContorno( h,y,ypr,s );
  end

hold on
  plot(x,y,'--r*')
hold off
  for j=1:s-1
    xko=x(j):0.05:x(j+1);
    f= y(j)+(ppr(j)*(xko-x(j)))+((M(j)*(xko-x(j)).^2)/2) +
      (((M(j+1)-M(j))*(xko-x(j)).^3)/(6*h(j)));

    hold on
      plot(xko,f,'b.-')
    hold off
  end
```

```
function [M, ppr] = SplineNatural1( h,y,s )

diag1=2*(h(1:s-2)+h(2:s-1));
diag2=h(2:s-2);
diag3=diag2;

      Otra forma
         diag1(1:s-2)=2*(h(1:s-2)+h(2:s-1))
         diag2(1:s-3)=h(2:s-2)
         diag3=diag2

      Otra forma
         for i=1:s-2
            diag1(i)=2*(h(i)+h(i+1));
         end;

         for i=1:s-3
            diag2(i)=h(i+1);
            diag3(i)=diag2(i);
         end;

d = 6*(((y(3:s)-y(2:s-1))./h(2:s-1))-((y(2:s-1)-y(1:s-
2))./h(1:s-2)));

 d=6*(((y((1:s-2)+2)-y((1:s-2)+1))./h((1:s-2)+1))-((y((1:s-
2)+1)-y(1:s-2))./h(1:s-2)))

      Otra forma
         for i=1:s-2;
            d(i)=6*(((y(i+2)-y(i+1))./h(i+1))-((y(i+1)-
y(i))./h(i)));
         end
```

```
[M]=Crout(diag1 ,diag2,diag3,d);
M=[0 M 0];  Para Spline Natural M0 y Mn son cero

ppr=(((y(2:s)-y(1:s-1)))./h(1:s-1))-((M(1:s-1).*h(1:s-1))./2)-
(((M(2:s)-M(1:s-1)).*h(1:s-1))./6);

 ppr=(((y((1:s-1)+1)-y(1:s-1)))./h(1:s-1))-((M(1:s-1).*h(1:s-
1))./2)-(((M((1:s-1)+1)-M(1:s-1)).*h(1:s-1))./6);

      Otra forma
         for i=1:s-1
            ppr(i)=(((y(i+1)-y(i)))/h(i))-((M(i)*h(i))./2)-
(((M(i+1)-M(i))*h(i))/6);
         end
end
```

```
function [ M , ppr ] = SplineCondContorno( h,y,ypr,s )

diag1=2*(h(1:s-2)+h(2:s-1));

  diag1=2*(h(1:s-2)+h((1:s-2)+1))

      Otra Forma
         for i=1:s-2
            diag1(i)=2*(h(i)+h(i+1))
         end

diag1=[2*h(1) diag1 2*h(end)];
diag2=h(1:end);
diag3=diag2;

d0=6*(((y(2)-y(1))/h(1))-ypr(1));
dn=6*(ypr(2)-((y(s)-y(s-1))/h(s-1)));

d = 6*(((y(3:s)-y(2:s-1))./h(2:s-1))-((y(2:s-1)-y(1:s-
2))./h(1:s-2)));

 d=6*(((y((1:s-2)+2)-y((1:s-2)+1))./h((1:s-2)+1))-((y((1:s-
2)+1)-y(1:s-2))./h(1:s-2)))

      Otra forma
         for i=1:s-2;
            d(i)=6*(((y(i+2)-y(i+1))./h(i+1))-((y(i+1)-
y(i))./h(i)));
         end

d=[d0 d dn];

[M]=Crout(diag1,diag2,diag3,d);
```

```
ppr=(((y(2:s)-y(1:s-1)))./h(1:s-1))-((M(1:s-1).*h(1:s-1))./2)-
(((M(2:s)-M(1:s-1)).*h(1:s-1))./6);

 ppr=(((y((1:s-1)+1)-y(1:s-1)))./h(1:s-1))-((M(1:s-1).*h(1:s-
1))./2)-(((M((1:s-1)+1)-M(1:s-1)).*h(1:s-1))./6);

      Otra forma
         for i=1:s-1
            ppr(i)=(((y(i+1)-y(i)))/h(i))-((M(i)*h(i))./2)-
(((M(i+1)-M(i))*h(i))/6);
         end

end
```

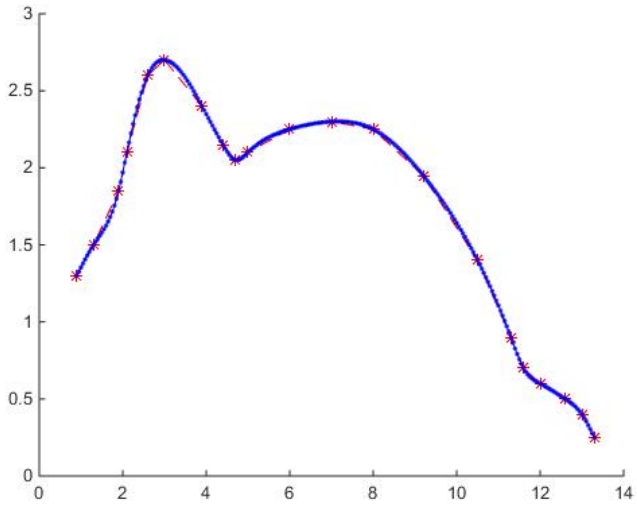
```
function [x]=Crout(a,b,c,d)
  Ponerle al vector b un cero delante para q cuadren las
formulas que te dan

  for i=length(b)+1:-1:2
         b(i)=b(i-1);
      end
        b(1)=0;

n=length(a);
  Factorizacion
      alfa(1)=a(1);
        for i=1:n-1
           gamma(i)=c(i)/alfa(i);
           alfa(i+1)=a(i+1)-(b(i+1)*gamma(i));
        end
    Obtener vector z
    z(1)=d(1)/alfa(1);
       for i=2:n
          z(i)=(d(i)-(b(i)*z(i-1)))/alfa(i);
       end

     Obtener vector x
    x(n)=z(n);
       for i=n-1:-1:1
          x(i)=z(i)-(gamma(i)*x(i+1));
       end
end

end
```

EJERCICIO 3

a)

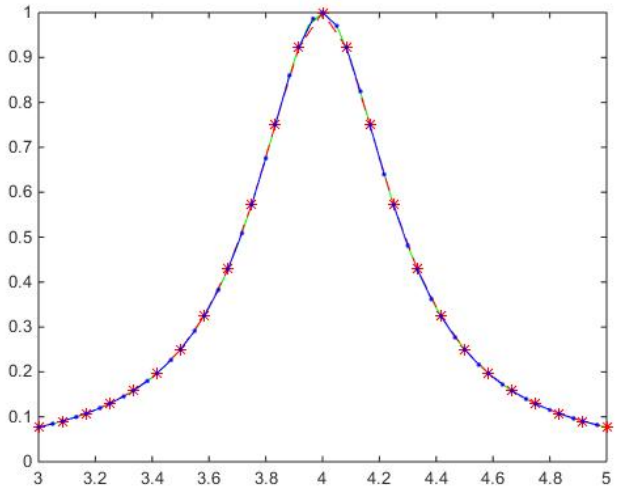
```
x = [0.9 1.3 1.9 2.1 2.6 3.0 3.9 4.4 4.7 5.0 6.0 7.0 8.0 9.2
10.5 11.3 11.6 12.0 12.6 13.0 13.3 ];

f0 = [ 1.3 1.5 1.85 2.1 2.6 2.7 2.4 2.15 2.05 2.1 2.25 2.3
2.25 1.95 1.4 0.9 0.7 0.6 0.5 0.4 0.25];

opt=0;

[ ppr ] = SplineCubico1( x, opt, f0);
```

Natural



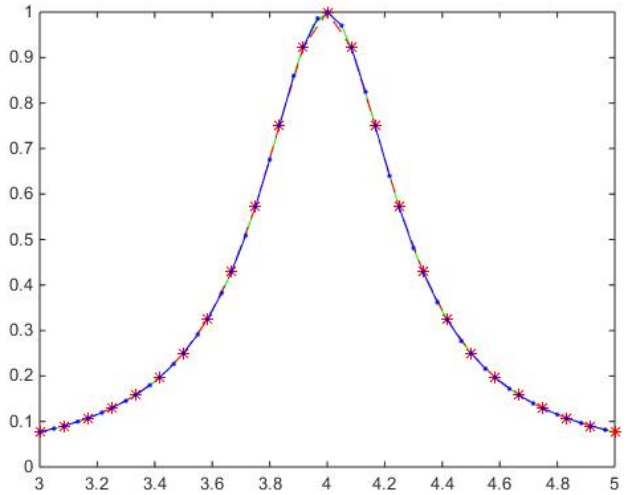
b1)

```
f0 = @(x)1./((1+12*(x-4).^2));

a=3;

b=5;

opt=1;

h = (b-a)/24;

x=a:h:b;

[ ppr ] = SplineCubico1( x, opt, f0)
```



b2)

```
f0 = @(x)1./((1+12*(x-4).^2));
df = @(x)(-24*(x-4))./(1+12*(x-4).^2).^2;
a=3;
b=5;
opt=1;
h = (b-a)/24;
x=a:h:b;
ypr = [df(a) df(b)];
[ ppr ] = SplineCubico1( x, opt, f0, ypr )
```



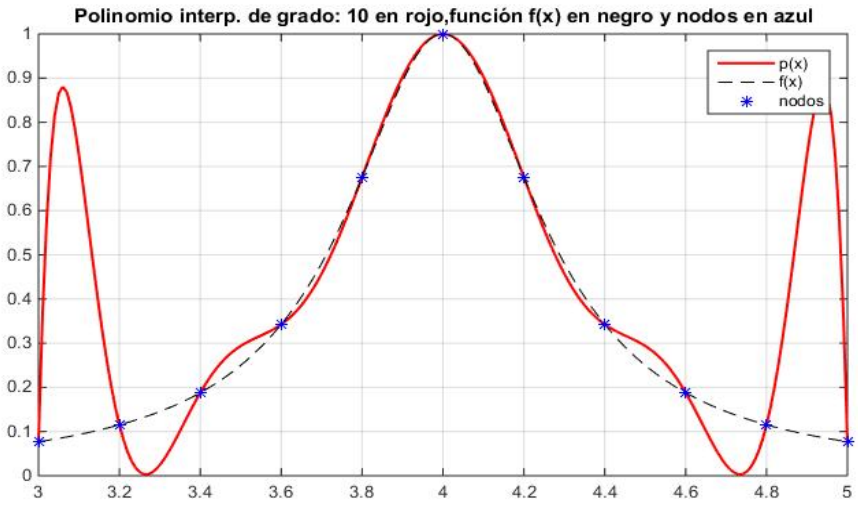
```
function [ ppr ] = SplineCubico ( x, opt, f0, ypr)
s=length(x);
h(1:s-1)=x(2:s)-x(1:s-1);
  if opt==1
    y=f0(x);
    xn=x(1):0.05:x(end);
    plot(xn,f0(xn),'g')
  else y=f0;
  end;

  if nargin==3
    [M, ppr] = SplineNatural1( h,y,s );
  elseif nargin==4
    [ M, ppr ] = SplineCondContorno( h,y,ypr,s );
  end

hold on
  plot(x,y,'--r*')
hold off
  for j=1:s-1
    xko=x(j):0.05:x(j+1);
    f= y(j)+(ppr(j)*(xko-x(j)))+((M(j)*(xko-x(j)).^2)/2) +
      (((M(j+1)-M(j))*(xko-x(j)).^3)/(6*h(j)));

    hold on
      plot(xko,f,'b.-')
    hold off
  end
```

```
function [M, ppr] = SplineNatural1( h,y,s )

diag1=2*(h(1:s-2)+h(2:s-1));
diag2=h(2:s-2);
diag3=diag2;

      Otra forma
         diag1(1:s-2)=2*(h(1:s-2)+h(2:s-1))
         diag2(1:s-3)=h(2:s-2)
         diag3=diag2
d = 6*(((y(3:s)-y(2:s-1))./h(2:s-1))-((y(2:s-1)-y(1:s-
2))./h(1:s-2)));

 d=6*(((y((1:s-2)+2)-y((1:s-2)+1))./h((1:s-2)+1))-((y((1:s-
2)+1)-y(1:s-2))./h(1:s-2)))

 [M]=Crout(diag1 ,diag2,diag3,d);
M=[0 M 0];  Para Spline Natural M0 y Mn son cero

ppr=(((y(2:s)-y(1:s-1)))./h(1:s-1))-((M(1:s-1).*h(1:s-1))./2)-
(((M(2:s)-M(1:s-1)).*h(1:s-1))./6);

 ppr=(((y((1:s-1)+1)-y(1:s-1)))./h(1:s-1))-((M(1:s-1).*h(1:s-
1))./2)-(((M((1:s-1)+1)-M(1:s-1)).*h(1:s-1))./6);

end
```

```
function [ M , ppr ] = SplineCondContorno( h,y,ypr,s )

diag1=2*(h(1:s-2)+h(2:s-1));

  diag1=2*(h(1:s-2)+h((1:s-2)+1))

diag1=[2*h(1) diag1 2*h(end)];
diag2=h(1:end);
diag3=diag2;

d0=6*(((y(2)-y(1))/h(1))-ypr(1));
dn=6*(ypr(2)-((y(s)-y(s-1))/h(s-1)));

d = 6*(((y(3:s)-y(2:s-1))./h(2:s-1))-((y(2:s-1)-y(1:s-
2))./h(1:s-2)));

 d=6*(((y((1:s-2)+2)-y((1:s-2)+1))./h((1:s-2)+1))-((y((1:s-
2)+1)-y(1:s-2))./h(1:s-2)))

d=[d0 d dn];

[M]=Crout(diag1,diag2,diag3,d);

ppr=(((y(2:s)-y(1:s-1)))./h(1:s-1))-((M(1:s-1).*h(1:s-1))./2)-
(((M(2:s)-M(1:s-1)).*h(1:s-1))./6);

 ppr=(((y((1:s-1)+1)-y(1:s-1)))./h(1:s-1))-((M(1:s-1).*h(1:s-
1))./2)-(((M((1:s-1)+1)-M(1:s-1)).*h(1:s-1))./6);

end
```

```
function [x]=Crout(a,b,c,d)
  Ponerle al vector b un cero delante para q cuadren las
formulas que te dan

  for i=length(b)+1:-1:2
         b(i)=b(i-1);
      end
        b(1)=0;

n=length(a);
  Factorizacion
      alfa(1)=a(1);
        for i=1:n-1
           gamma(i)=c(i)/alfa(i);
           alfa(i+1)=a(i+1)-(b(i+1)*gamma(i));
        end
    Obtener vector z
    z(1)=d(1)/alfa(1);
       for i=2:n
          z(i)=(d(i)-(b(i)*z(i-1)))/alfa(i);
       end

     Obtener vector x
    x(n)=z(n);
       for i=n-1:-1:1
          x(i)=z(i)-(gamma(i)*x(i+1));
       end
end

end
```

EJERCICIO 4




```
opt=1;
a=3;
b=5;
grado=10;
h=(b-a)/grado;
x=a:h:b;

f0=@(x)(1./(1+12.*(x-4).^2));
[xd,pd] = polinterdifdiv(x,opt,f0);
```

```
opt=1;
a=3;
b=5;
grado=10;
x = Chebyshev(a,b,grado)
f0=@(x)(1./(1+12.*(x-4).^2));
[xd,pd] = polinterdifdiv(x,opt,f0)
```