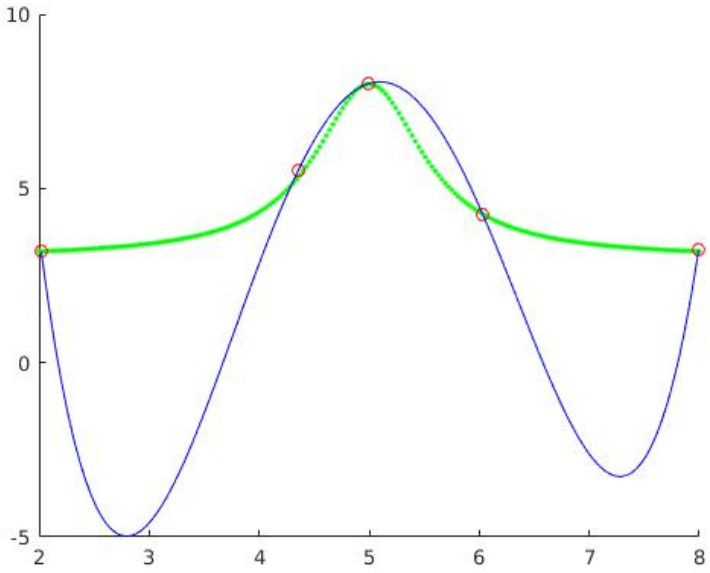
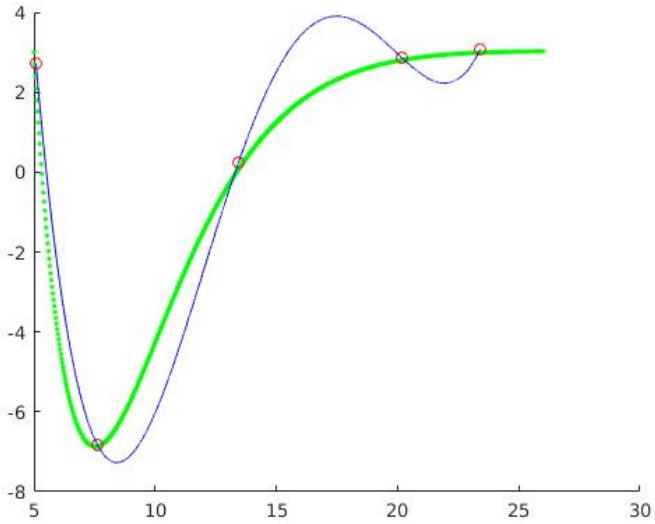
Apellidos, Nombre: NIEVES SÁNCHEZ, VÍCTOR
Apellidos, Nombre:

1. Curvas 1D. Polinomio Newton

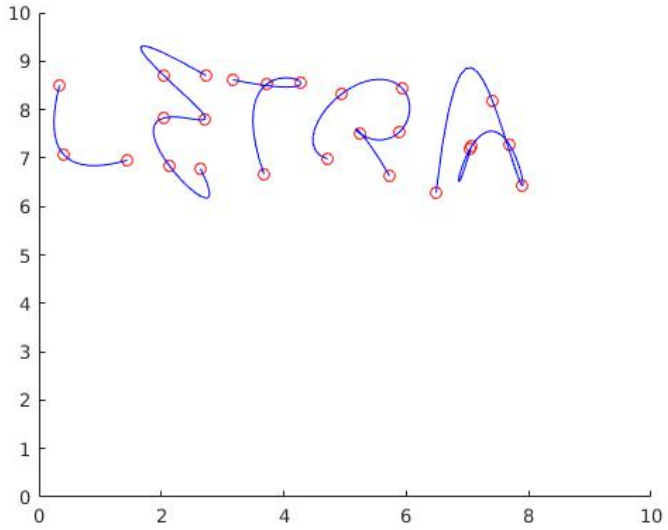
f(x):



g(x):



2. Curvas 2D. Polinomio Newton



3.1. Curvas 2D. Spline cúbico

Código:

```
function a=tri(r)
  for k=1:length(r)
     if k==1
        M(k,1)=1;
        M(k,2)=-2;
        M(k,3)=1;
     elseif k==length(r)
        M(k,length(r)-2)=1;
        M(k,length(r)-1)=-2;
        M(k,length(r))=1;
     else
        M(k,k-1)=1;
        M(k,k)=4;
        M(k,k+1)=1;
     end
  end
  a=M\r;
end
```

Volcado:

```
>>  r=[1:10]';z=tri(r)

z =
   0.7892
   0.1667
   0.5441
   0.6570
   0.8280
   1.0312
   1.0472
   1.7799
  -0.1667
   7.8868
```

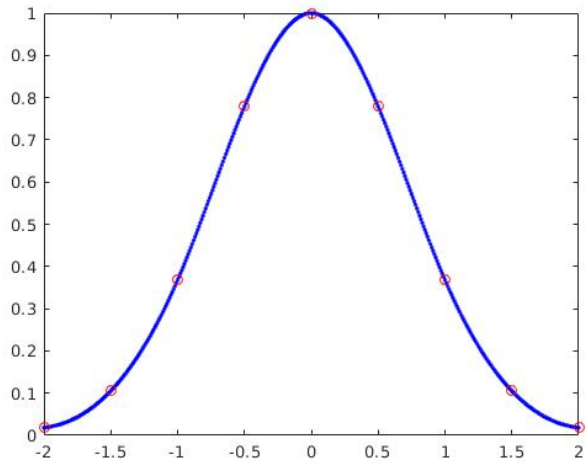
3.2. Codificar la rutina yy=spline3(xi,yi,xx) para calcular el spline cúbico.

Código:

```
function yy=spline3(xi,yi,xx)
    n=length(xi);
    h=xi(2)-xi(1);
    F=zeros(n,1);
    for k=2:n-1
        F(k,1)=6./h.^2.*(yi(k+1)-2.*yi(k)+yi(k-1));
    end
    z=tri(F);
    z(end+1)=0;
    C=zeros(n+1,1);
    for k=1:n
        C(k,1)=yi(k)-(h.^2./6).*z(k);
    end

    for x=1:length(xx)
        w=(xx(x)-xi(1))./h;
        k=floor(w);
        u=w-k;
        yy(x)=(h.^2/6).*z(k+2).*u.^3+(h.^2/6).*z(k+1).*(1-u).^3+C(k+2).*u+C(k+1).*(1-u);
    end
end
```

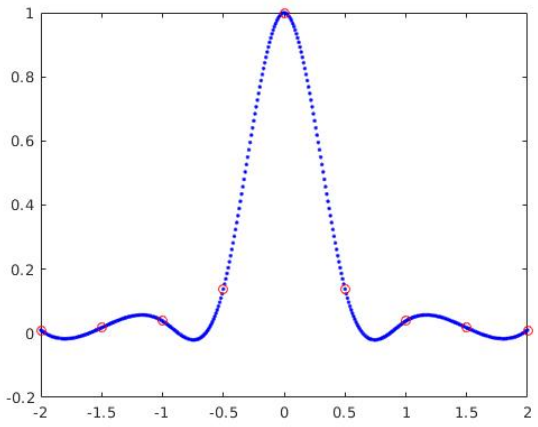
Volcado:

```
>> xi=[-2:0.5:2]';yi=exp(-xi.^2);xx=-2:0.01:2;yy=spline3(xi,yi,xx);plot(xx,yy,'b.',xi,yi,'ro')
```



Gráfica de interpolación de la función 1/(1+25X^2).

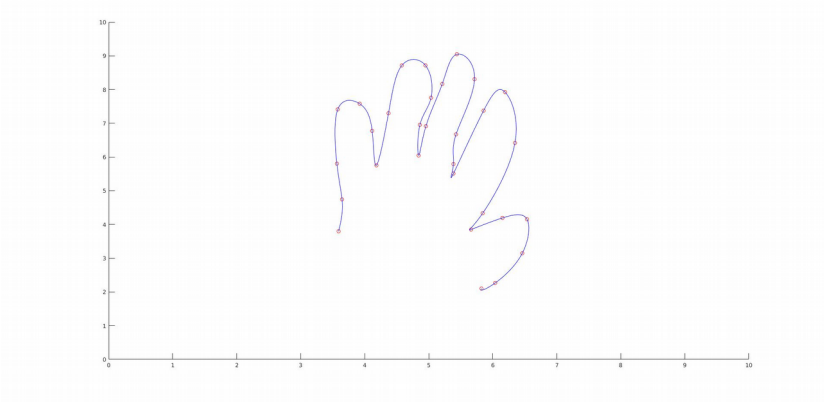
```
>> xi=[-2:0.5:2]';yi=1./(1+25*xi.^2);xx=-2:0.01:2;yy=spline3(xi,yi,xx);plot(xx,yy,'b.',xi,yi,'ro')
```



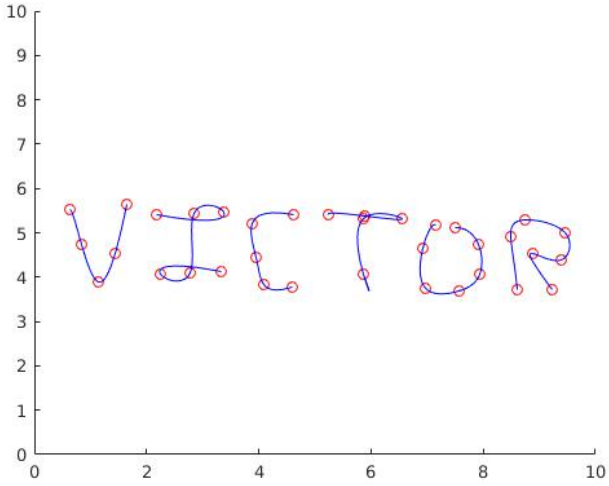
3.3. En la rutina DibujaCurva2D(nletras) sustituir la función polNewton por la función spline3.
Código:

```
function DibujaCurva2D_b(nletras)
% nletras es el nÃºmero de caracteres que queremos dibujar

% InterpolaciÃ³n de Newton (o spline cÃºbico) en cada una de las coordenadas

% Arranca la interface grÃ¡fica.
% Con el boton de la izda del ratÃ³n, aÃ±adimos puntos de control.
% Cada punto aÃ±adido se dibuja en la pantalla (un circulo rojo).
% Con el boton de la drcha del ratÃ³n, aÃ±adimos el Ãºltimo punto.
% Calcula y dibuja el polinomio de Newton que interpola cada una de las coordenadas de los
% puntos de control (lÃnea azul).

axis([0 10 0 10])
hold on
% Initially, the list of points is empty.
for k=1:nletras
xy = [];
n = 0;
% Loop, picking up the points.
disp('Left mouse button picks points.')
disp('Right mouse button picks last point.')
but = 1;
while but == 1
   [xi,yi,but] = ginput(1);
   plot(xi,yi,'ro')
   n = n+1;
   xy(:,n) = [xi;yi];
end
% Interpolate with a spline curve and finer spacing.
t = 1:n;
ts = 1: 0.1: n;
%Interpolate with a spline curve and finer spacing.
xs = spline3(t,xy(1,:),ts);
ys = spline3(t,xy(2,:),ts);
% Plot the interpolated curve.
plot(xs,ys,'b-');
end
hold off
return
```

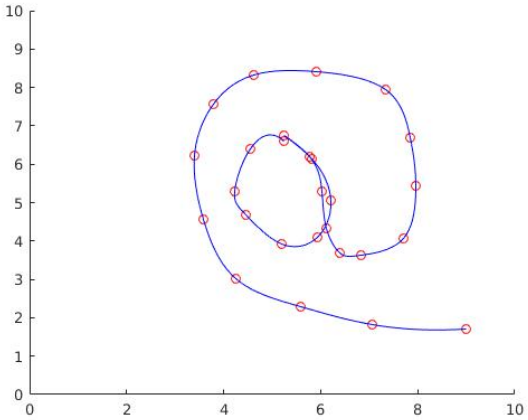
3.3. Dibujar curvas 2D con splines cúbicos: caracteres y perfiles.
Mano:



Nombre:



@:

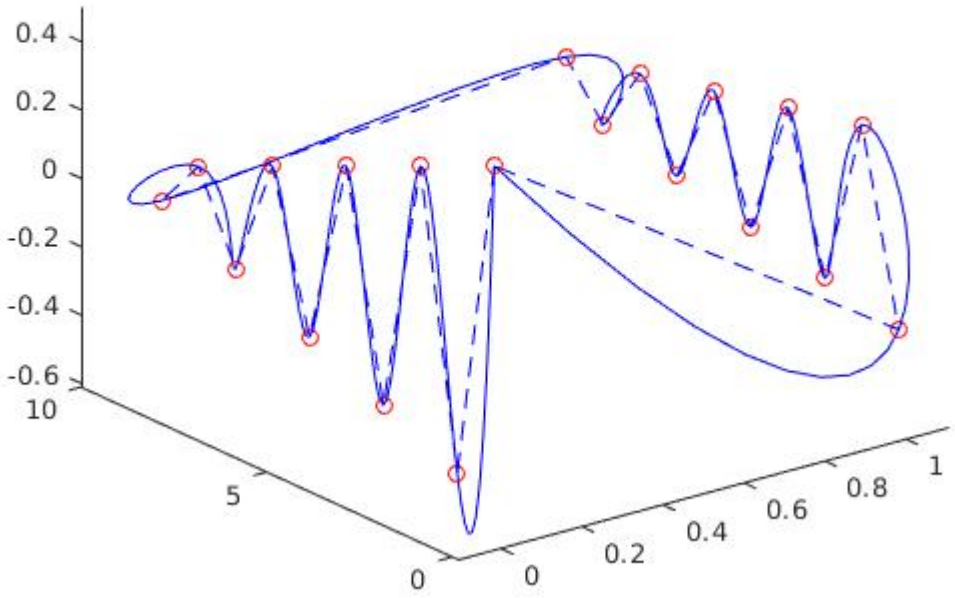


4. Dibujar curvas 3D

Código:

```
function DibujaCurva3D()
   P=zeros(21,3);
   P(1,:)=[0 0 0.5];
   P(2,:)=[0 1 -0.45];
   P(3,:)=[0 2 0.4];
   P(4,:)=[0 3 -0.35];
   P(5,:)=[0 4 0.3];
   P(6,:)=[0 5 -0.25];
   P(7,:)=[0 6 0.2];
   P(8,:)=[0 7 -0.15];
   P(9,:)=[0 8 0.1];
   P(10,:)=[0 9 -0.05];
   P(11,:)=[1 9 0.05];
   P(12,:)=[1 8 -0.1];
   P(13,:)=[1 7 0.1];
   P(14,:)=[1 6 -0.15];
   P(15,:)=[1 5 0.15];
   P(16,:)=[1 4 -0.2];
   P(17,:)=[1 3 0.2];
   P(18,:)=[1 2 -0.25];
   P(19,:)=[1 1 0.25];
   P(20,:)=[1 0 -0.3];
   P(21,:)= P(1,:);
   plot3(P(:,1),P(:,2),P(:,3),'ro', P(:,1),P(:,2),P(:,3),'b--');
   hold on
   % Initially, the list of points is empty.
   xyz = [];
   n = 0;
   for k=1:21
      xi = P(k,1);
      yi = P(k,2);
      zi = P(k,3);
      plot3(xi,yi,zi,'ro')
      n = n+1;
      xyz(:,n) = [xi;yi;zi];
   end
   for k=1:20
      t = 1:n;
      ts = 1: 0.1: n;
      xs = spline3(t,xyz(1,:),ts);
      ys = spline3(t,xyz(2,:),ts);
      zs = spline3(t,xyz(3,:),ts);
      plot3(xs,ys,zs,'b-')
   end
end
```

Gráfica:



Anexo: código

```
function yy=polNewton(x,y,xx)
% Calcula diferencias divididas a partir de x e y
% Entrada: x, N abcisas de interpolacion
%          y, N ordenadas de interpolacion
%         xx, opcional. Si se da, se evalua el polinomio en esos puntos
% Salida:  si no se da xx, devuelve las N diferencias divididas
%          si se da xx, devuelve la evaluación del polinomio en xx.

N=length(y); y=reshape(y,N,1); % aseguro vector columna.
DD = zeros(N,N);      % Reservo matriz de Diferencias Divididas
DD(:,1) = y;    % 1era columna = valores de y

for k=2:N, %  Barremos columnas de DD (diferencias ordenadas orden k)
  for j=k:N, % En cada columna k barremos de la diagonal(k) hacia abajo
    dif =  (DD(j,k-1)-DD(j-1,k-1));  % resta de dif divididas orden k-1
    dx =  x(j)-x(j-k+1);           % resta de abscisas
    DD(j,k) =  dif/ dx;              % Diferencia DIVIDIDA
  end
end

c = diag(DD)';  % extraemos la diagonal del cuadro de diferencias div

if nargin==2,  % Solo dos argumentos de entrada, devolver dif div
 out=c;
else    % Si llegamos aqui hay un tercer arg -> evaluar polinomio en xx
 pp = c(N);  % Regla de Horner con diferencias divididas
 for k=N-1:-1:1,
   pp = c(k) + pp.*(xx-x(k));
 end
 yy=pp; % devuelve valores obtenidos polinomio en out
end
return
```

```
function DibujaCurva1D

% Interpolación de Newton 1D

% Arranca la interface gráfica.
% Dibuja la gráfica de la función f(x)=1/(1+25x^2)
% Con el boton de la izda del ratón, añadimos puntos de control sobre la función.
% Cada punto añadido se dibuja en la pantalla (un circulo rojo).
% Con el boton de la drcha del ratón, añadimos el último punto.
% Calcula y dibuja el polinomio de Newton que interpola esos puntos de control (línea
azul). Utiliza la rutina polNewton.

hold on
% Initially, the list of points is empty.
xy = [];
xx=-1:0.01:1;
fx=1./(1+25*xx.^2);
plot(5+3*xx,3+5*fx,'g.');
n = 0;
% Loop, picking up the points.
disp('Left mouse button picks points.')
```

```
disp('Right mouse button picks last point.')
but = 1;
while but == 1
    [xi,yi,but] = ginput(1);
    plot(xi,yi,'ro')
    n = n+1;
    xy(:,n) = [xi;yi];
end
% Interpolate with a spline curve and finer spacing.
%t = 1:n;
ts = min(xy(1,:)): 0.001: max(xy(1,:));
xys = polNewton(xy(1,:),xy(2,:),ts);
 % Plot the interpolated curve.
plot(ts,xys,'b-');
hold off
return

function DibujaCurva2D(nletras)
% nletras es el número de caracteres que queremos dibujar

% Interpolación de Newton (o spline cúbico) en cada una de las coordenadas

% Arranca la interface gráfica.
% Con el boton de la izda del ratón, añadimos puntos de control.
% Cada punto añadido se dibuja en la pantalla (un circulo rojo).
% Con el boton de la drcha del ratón, añadimos el último punto.
% Calcula y dibuja el polinomio de Newton que interpola cada una de las coordenadas de los
% puntos de control (línea azul).

axis([0 10 0 10])
hold on
% Initially, the list of points is empty.
for k=1:nletras
xy = [];
n = 0;
% Loop, picking up the points.
disp('Left mouse button picks points.')
disp('Right mouse button picks last point.')
but = 1;
while but == 1
    [xi,yi,but] = ginput(1);
    plot(xi,yi,'ro')
    n = n+1;
    xy(:,n) = [xi;yi];
end
% Interpolate with a spline curve and finer spacing.
t = 1:n;
ts = 1: 0.1: n;
%Interpolate with a spline curve and finer spacing.
xs = polNewton(t,xy(1,:),ts);   % Sustituir por xs=spline3(t,xy(1, : ),ts));
ys = polNewton(t,xy(2,:),ts);   % Sustituir por xs=spline3(t,xy(2, : ),ts));
% Plot the interpolated curve.
plot(xs,ys,'b-');
end
hold off
return
```