```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                  %
%Interpolación Cúbica de Akima     %
%                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Parámetros de Entrada:

%x,y:Vectores que guardan los números de los pares ordenado (xi,yi)
%    para i=0,1,2,....,n donde x0<x1<x2<...<xn.

%newx:Vector de elementos para los cuales se estimarán mediante
la Interpolación
%     Cúbica de Akima.

function[newy]=ICAkima(x,y,newx)
    n1=length(x);%Obtenemos el número de elementos de x.
    n2=length(newx);%Obtenemos el número de elementos de newx.
    newy=zeros(n2,1);%Vector que guardará los nuevos valores de y.
    xadd=zeros(n1+4,1);
    yadd=zeros(n1+4,1);
    %Vectores necesarios para calcular los coeficientes de los
    polinomios.
    h=zeros(n1-1,1);
    s=zeros(n1+3,1);
    dy=zeros(n1,1);
    a=zeros(n1-1,1);
    b=zeros(n1-1,1);
    c=zeros(n1-1,1);
    d=zeros(n1-1,1);

    %Calculamos los nuevos puntos extra necesarios
    xadd(n1+4)=2*x(n1)-x(n1-2);
    xadd(n1+3)=x(n1)+x(n1-1)-x(n1-2);
    xadd(1)=2*x(1)-x(3);
    xadd(2)=x(1)+x(2)-x(3);
    yadd(n1+3)=((((2*(y(n1)-y(n1-1)))/(x(n1)-x(n1-1)))-((y(n1-1)-y
    (n1-2))/(x(n1-1)-x(n1-2))))*(xadd(n1+3)-x(n1)))+y(n1);
    yadd(n1+4)=((((2*(yadd(n1+3)-y(n1)))/(xadd(n1+3)-x(n1)))-((y(
    n1)-y(n1-1))/(x(n1)-x(n1-1))))*(xadd(n1+4)-xadd(n1+3)))+yadd(
    n1+3);
    yadd(2)=-((((2*(y(2)-y(1)))/(x(2)-x(1)))-((y(3)-y(2))/(x(3)-x(
    2))))*(x(1)-xadd(2)))+y(1);
    yadd(1)=-((((2*(y(1)-yadd(2)))/(x(1)-xadd(2)))-((y(2)-y(1))/(x
    (2)-x(1))))*(xadd(2)-xadd(1)))+yadd(2);
    for i=(1:n1-1)
        xadd(i+2)=x(i);
        yadd(i+2)=y(i);
        h(i)=x(i+1)-x(i);
    end
    xadd(n1+2)=x(n1);
    yadd(n1+2)=y(n1);

    %Calculamos los coeficientes del polinomio.
    for i=(1:n1+3)
        s(i)=(yadd(i+1)-yadd(i))/(xadd(i+1)-xadd(i));
    end
    for i=(3:n1+2)
        w1=abs(s(i+1)-s(i));
        w2=abs(s(i-1)-s(i-2));
        if(w1<=10^-8 && w2<=10^-8)
            dy(i-2)=(s(i-1)+s(i))/2;
        else
```

```
57                  dy(i-2)=(w1*s(i-1)+w2*s(i))/(w1+w2);
58              end
59          end
60          for i=(1:n1-1)
61              a(i)=y(i);
62              b(i)=dy(i);
63              c(i)=(3*s(i+2)-dy(i+1)-2*dy(i))/(h(i));
64              d(i)=-(2*s(i+2)-dy(i+1)-dy(i))/(h(i)^2);
65          end
66          %Encontramos el nuevo punto para determinar a que polinomio
            pertenece.
67          for i=(1:n2)
68              distancia=abs(min(x)-max(x));
69              for j=(1:n1)
70                  if(abs(newx(i)-x(j))<distancia)
71                      distancia=abs(newx(i)-x(j));
72                      indx=j;
73                  end
74              end
75              %Aproximamos los nuevos puntos y
76              if(newx(i)<x(indx))
77                  indx=indx-1;
78                  newy(i)=a(indx)+b(indx)*(newx(i)-x(indx))+c(indx)*(
                    newx(i)-x(indx))^2+d(indx)*(newx(i)-x(indx))^3;
79              elseif(newx(i)>x(indx))
80                  newy(i)=a(indx)+b(indx)*(newx(i)-x(indx))+c(indx)*(
                    newx(i)-x(indx))^2+d(indx)*(newx(i)-x(indx))^3;
81              elseif((newx(i)-x(indx))<10^-8)
82                  newy(i)=y(indx);
83              end
84          end
85
86      end
87
88      %Parámetros de Salida:
89
90      %newy:Nuevos puntos estimados mediante Interpolación de Akima.
```