```
function [S, Sp, Spp] = SplineCubica(X, Y, fpa, fpb, x)
    %Synopsis:  [S] = SplineCubica(X, Y, fpa, fpb, x)
    %           [S, Sp] = SplineCubica(X, Y, fpa, fpb, x)
    %           [S, Sp, Spp] = SplineCubica(X, Y, fpa, fpb, x)
    %           [S, Sp, Spp, C] = SplineCubica(X, Y, fpa, fpb, x)
    %                     ^---matrice 4 coloane - fiecare coloana
    %                     reprezentand coeficientii a(j), b(j), c(j),
 d(j)

    %Input: (X, Y) - setul de date
    %       fpa, fpb - panta lui S in capete
    %       x - variabila curenta

    %Output: S = S(x)
    %        Sp = S'(x)
    %        Spp = S''(x)

    n = length(X) - 1;
    %Determinam coeficientii a(j)
    for j = 1:n
        a(j) = Y(j);
    end
    %Calculam coeficientii b(j) - rezolvand sistemul din curs
    A(1, 1) = 1;
    A(n+1, n+1) = 1;
    for i = 2:n
        A(i, i) = 4;
        A(i, i-1) = 1;
        A(i, i+1) = 1;
    end
    %Construim vectorul termenilor liberi - L
    L(1) = fpa;
    L(n+1) = fpb;
    %Diferenta dintre doua noduri consecutive - functioneaza doar
 pentru
    %discretizare echidistanta
    %h = X(2) - X(1); - modificam pasul
    for i = 2:n
        h = X(i+1) - X(i);
        L(i) = (3/h)*(Y(i+1) - Y(i-1));
    end
    b = GaussPivTotal(A, L);
    %Aflam in continuare ceilalti coef: c(j) si d(j);

    for  i =  1:n
        h = X(i+1) - X(i);
        c(i) = (3/h^2) * (Y(i+1) - Y(i)) - (b(i+1) + 2*b(i))/h;
        d(i) = (-2/h^3) * (Y(i+1) - Y(i)) + (1/h^2) * (b(i+1) + b(i));
    end

    %Voi parcurge fiecare subinterval in parte cautant intervalul care
 il
```

```
    %contine pe x si calculam in acelasi timp valoare lui S;
   for k = 1:length(x)
        for i = 1:n
            if x(k) >= X(i) && x(k) <=X(i+1)
                S(k) = a(i) + b(i)*(x(k) - X(i)) + c(i)*(x(k) -
 X(i))^2 + d(i)*(x(k) - X(i))^3;
                if nargout >= 2
                    Sp(k) = b(i) + 2*c(i)*(x(k) - X(i)) + 3*d(i)*(x(k)
 - X(i))^2;
                    if nargout >= 3
                        Spp(k) = 2*c(i) + 6*d(i)*(x(k) - X(i));
                    end
                end
            end

        end
   end
 if nargout == 4
    for j = 1:n
        C(j, 1) = a(j);
        C(j, 2) = b(j);
        C(j, 3) = c(j);
        C(j, 4) = d(j);
    end
 end
end
```

```
Not enough input arguments.

Error in SplineCubica (line 17)
    n = length(X) - 1;
```

Published with MATLAB® R2019a