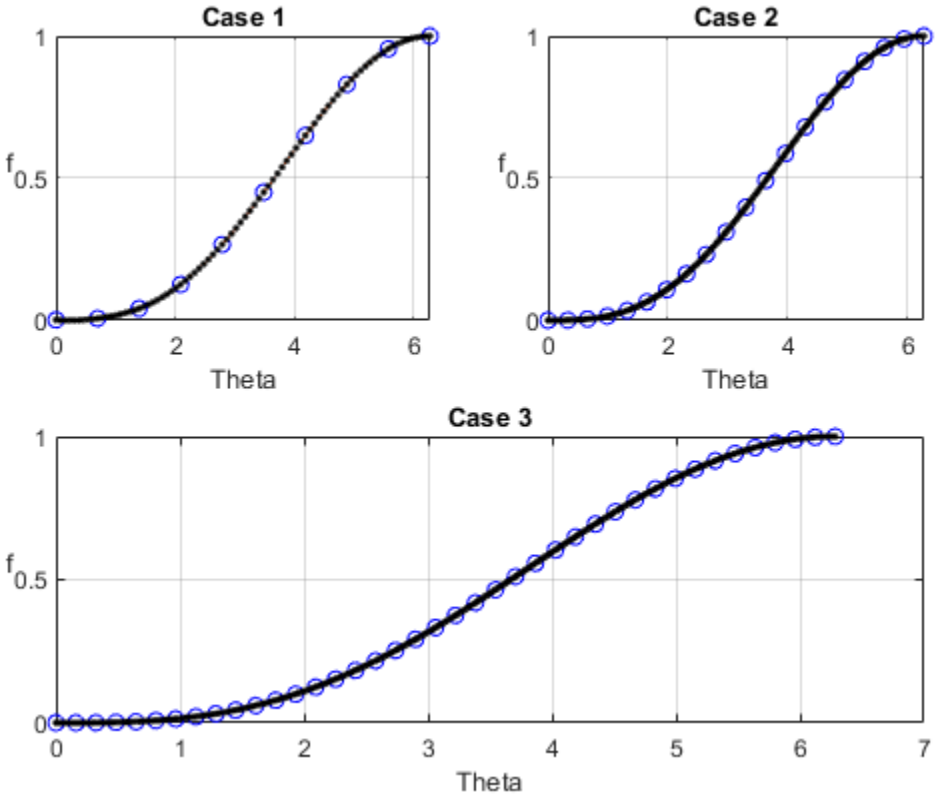
```
clear, clc;
close all;

% make data points for different cases
n1 = 10;
n2 = 20;
n3 = 40;

% exact case
xexact = linspace(0,2*pi(),100);
yexact = sin(xexact./4).^3;

%getting splines/x/y values
[s1,x1,y1] = clampedSpline(n1);
[s2,x2,y2] = clampedSpline(n2);
[s3,x3,y3] = clampedSpline(n3);

%Plot all 3 cases against each other
figure;
subplot(2,2,1)
cubicplot(x1,y1,xexact,yexact,s1,n1, 'Case 1')
subplot(2,2,2)
cubicplot(x2,y2,xexact,yexact,s2,n2, 'Case 2')
subplot(2,2,[3 4])
cubicplot(x3,y3,xexact,yexact,s3,n3, 'Case 3')

%Funtion that take in number of data points and return splines with domain
%and range
function [splines,x,y] = clampedSpline(n)
    %making uniform grid based on given size n
    x = linspace(0,2*pi,n);
    y = sin(x./4).^3;

    % Set up matrices for cramer's
    % Use cramer sub-routine to solve for k1 and kn
    Al = [(x(2)-x(1)),((x(2)-x(1))^2);(x(3)-x(1)),((x(3)-x(1))^2)];
    bl = [y(2);y(3)];
    Ar = [(x(n-1)-x(n)),((x(n-1)-x(n))^2);(x(n-2)-x(n)),((x(n-2)-x(n))^2)];
    br = [(y(n-1)-y(n));(y(n-2)-y(n))];
    k1 = cramer(Al,bl,1);
    kn = cramer(Ar,br,1);

    % making tridiagonal for Thomas3 and solve for k2 to kn-1
    a = (1/6)*ones(1,n);
    b = (4/6)*ones(1,n);
    c = (1/6)*ones(1,n);
    e = ones(1,n);
    for i = 2:n-1
        e(i) = (y(i+1)-y(i-1))/(2*(x(i+1)-x(i)));
    end
    e(1) = k1;
    e(n) = kn;
```

```
    k = THOMAS3(a,b,c,e,n);
    k(1) = k1;
    k(n) = kn;

    %solving for m and d
    for i = 1:n-1
        m(i) = (3*(y(i+1)-y(i))/(x(i+1)-x(i))^2)-((k(i+1)+2*k(i))/(x(i+1)-
x(i)));
        d(i) = ((k(i+1)+k(i))/(x(i+1)-x(i))^2)-((2*(y(i+1)-y(i)))/((x(i+1)-
x(i))^3));
    end

    %set up 10 points per subdivision
    %solve for splines
    splines = cell(n-1,1);
    for i = 1:n-1
        xspline = linspace(x(i),x(i+1),10);
        yspline = y(i)+k(i)*(xspline-x(i))+m(i)*(xspline-
x(i)).^2+d(i)*((xspline-x(i)).^3);
        splines{i} = [xspline;yspline];
    end
end

%Plot function
%plot the original/exact function and data points
%plot splines
function cubicplot(x,y,xexact,yexact,splines,n,str)
    figure(1);
    plot(x, y, 'bo',xexact,yexact,'--')
    hold on; grid on;
    for i = 1:n-1
        plot(splines{i}(1,:),splines{i}(2,:),'.k')
    end
    title(str);
    xlabel('Theta');
    ylabel('f','Rotation',0);
end

%take matrix a,b, and column values then use cramer's rule
function cram = cramer(a,b,col)
    deterA = (a(1,1).*a(2,2))-(a(1,2).*a(2,1));
    ai = a;
    ai(:,col) = b;
    deterAI = (ai(1,1).*ai(2,2))-(ai(1,2).*ai(2,1));
    cram = deterAI/deterA;
end

%Sub-routine to solve for tridiagonal matrix system
function x = THOMAS3(a,b,c,d,n)

    %initial condition
    bbar(1) = b(1);
    cbar(1) = c(1);
    dbar(1) = d(1);
```

```
    %making upper triangle
    for i = 2:n
        multiplier = a(i)./bbar(i-1);
        abar(i) = a(i) - bbar(i-1).*multiplier;
        bbar(i) = b(i) - cbar(i-1).*multiplier;
        cbar(i) = c(i);
        dbar(i) = d(i) - dbar(i-1).*multiplier;
    end

    %initialize x of size n
    x = ones(1,n);

    %initialize end condition
    x(n) = dbar(n)/bbar(n);

    % Upward substitution AKA zip it up
    for i = n-1:-1:1
        x(i) = (dbar(i)-(cbar(i)*x(i+1)))/bbar(i);
    end
end
```



Published with MATLAB® R2021b