1a. P1.m

1b. Polyarea relies on a linear approximation of the given shape, which limits the overall order of accuracy to order two.

2a. myarea works by splitting the shape into two sets of points that can be interpolated to create a top and bottom curve. It uses a piecewise cubic hermite interpolating polynomial (pchip function) to interpolate these points to give two piecewise polynomials for the top and bottom curves. These curves are then integrated using the integral function for each polynomial in the piecewise functions over their specific breaks. The function finally subtracts the bottom integral from the top to give the final area of the shape.

2b. myarea.m

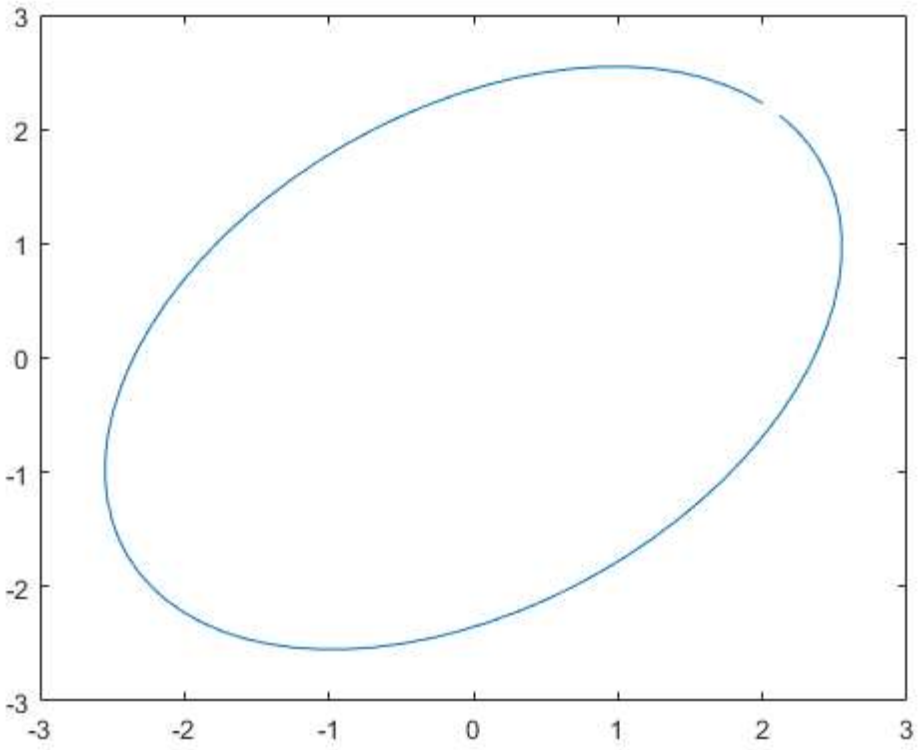
2c. P2.m

The order of accuracy of myarea is order four since it uses the pchip function to interpolate the given points. Since pchip is cubic, and the integral function used to find the area has a higher order of accuracy, the overall order of accuracy will be four.

3a. P3.m

```
function S = P1(iter)
iter = 4;
k = zeros(iter,1);
n = zeros(iter,1);
enum = zeros(iter,1);
alpha = zeros(iter,1);
for l = 1:iter
    k(l) = l;
    n(l) = 2^(l-1)*10;
    theta = zeros(n(l),1);
    x = zeros(n(l),1);
    y = zeros(n(l),1);

    for i = 1:n(l)
        theta(i) = 2*i*pi/n(l);
        x(i) = (3*sqrt(2)/2)*cos(theta(i))-sqrt(2)*sin(theta(i));
        y(i) = (3*sqrt(2)/2)*cos(theta(i))+sqrt(2)*sin(theta(i));
    end
    plot(x,y);
    enum(l) = abs(6*pi-polyarea(x,y));
end
for l = 2:iter
    alpha(l) = log2(abs(enum(l-1)/enum(l)));
end

R = table(k,n,enum,alpha, 'VariableNames',{'k','n','error','accuracy'});
S = table(R,'VariableNames',{'Results'});
```

```
ans =

  4×1 table

              Results
    k    n      error      accuracy
    ____________________________________

    1    10       1.216          0
    2    20     0.30854     1.9786
    3    40     0.07742     1.9947
    4    80    0.019373     1.9987
```



Published with MATLAB® R2021b

Contents


```
function area = myarea(x,y)
```

```
%MYAREA - computes the area of a region with a curved boundary.
% input
% x - a vector for the x values of points on the boundary
% y - a vector for the y values of points on the boundary
% output
% area - area enclosed by a curved boundary defined by vectors x and y.
```

Find endpoints where curves will intersect

```
[mx,id1] = max(x);
[mn,id2] = min(x);
maxin = max(id1,id2);
minin = min(id1,id2);
polyleng = maxin-minin;
```

Create vectors for x and y values for top and bottom curves

```
polyx1 = zeros(polyleng+1,1);
polyy1 = zeros(polyleng+1,1);
polyx2 = zeros(length(x)-polyleng+1,1);
polyy2 = zeros(length(x)-polyleng+1,1);

for i = minin:maxin
    polyx1(i-minin+1) = x(i);
    polyy1(i-minin+1) = y(i);
end
for i = maxin:length(x)
    polyx2(length(polyx2)+maxin-i) = x(i);
    polyy2(length(polyx2)+maxin-i) = y(i);
end
for i = 1:minin
    polyx2(minin-i+1) = x(i);
    polyy2(minin-i+1) = y(i);
end
```

Create piecewise functions using piecewise cubic hermite interpolation

```
pp1 = pchip(polyx1,polyy1);
pp2 = pchip(polyx2,polyy2);
xq = mn:0.0001:mx;
```

```
plot(xq,ppval(pp1,xq),'red');
plot(xq,ppval(pp2,xq),'green');
```

Integrate top and bottom piecewise functions

```
[B1,C1] = unmkpp(pp1);
[B2,C2] = unmkpp(pp2);

Integral1 = 0;
for i = 1:length(B1)-1
    polyindv1 = @(x) C1(i,1).*(x-B1(i)).^3+C1(i,2).*(x-B1(i)).^2+C1(i,3).*(x-B1(i))+C1(i,4);
    a = B1(i);
    b = B1(i+1);
    I = integral(polyindv1,a,b);
    Integral1 = Integral1 + I;
end
Integral2 = 0;
for i = 1:length(B2)-1
    polyindv2 = @(x) C2(i,1).*(x-B2(i)).^3+C2(i,2).*(x-B2(i)).^2+C2(i,3).*(x-B2(i))+C2(i,4);
    a = B2(i);
    b = B2(i+1);
    I = integral(polyindv2,a,b);
    Integral2 = Integral2 + I;
end
```

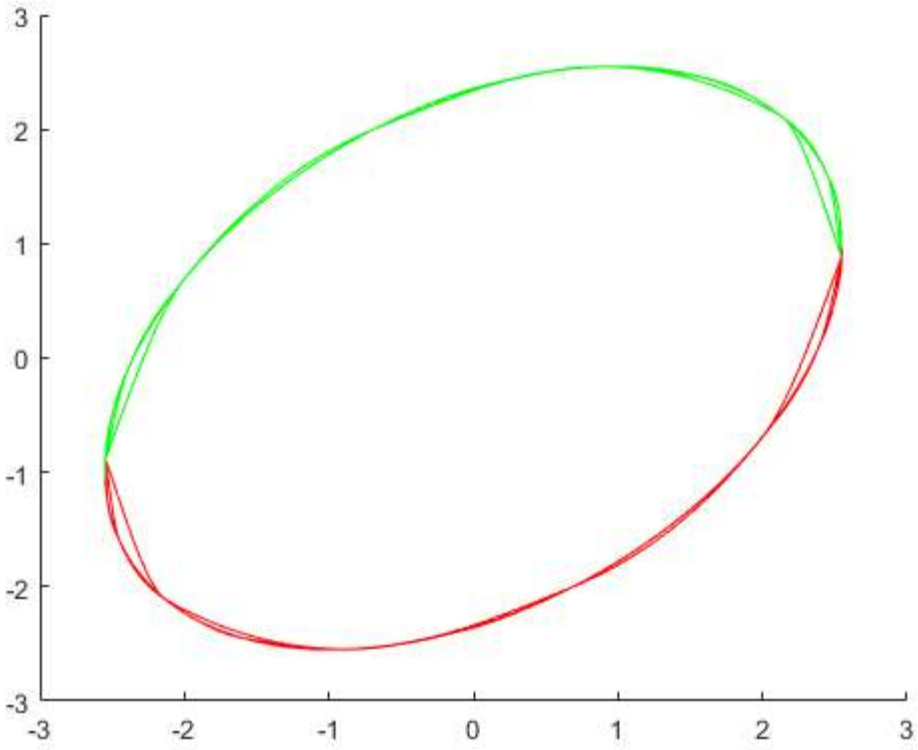
Find the total area

```
area = Integral2-Integral1;
```

Published with MATLAB® R2021b

```
function S = P2(iter)
iter = 4;
k = zeros(iter,1);
n = zeros(iter,1);
enum = zeros(iter,1);
alpha = zeros(iter,1);
for l = 1:iter
    k(l) = l;
    n(l) = 2^(l-1)*10;
    theta = zeros(n(l),1);
    x = zeros(n(l),1);
    y = zeros(n(l),1);

    for i = 1:n(l)
        theta(i) = 2*i*pi/n(l);
        x(i) = (3*sqrt(2)/2)*cos(theta(i))-sqrt(2)*sin(theta(i));
        y(i) = (3*sqrt(2)/2)*cos(theta(i))+sqrt(2)*sin(theta(i));
    end
    enum(l) = abs(6*pi-myarea(x,y));
end
for l = 2:iter
    alpha(l) = log2(abs(enum(l-1)/enum(l)));
end

R = table(k,n,enum,alpha, 'VariableNames',{'k','n','error','accuracy'});
S = table(R,'VariableNames',{'Results'});
```

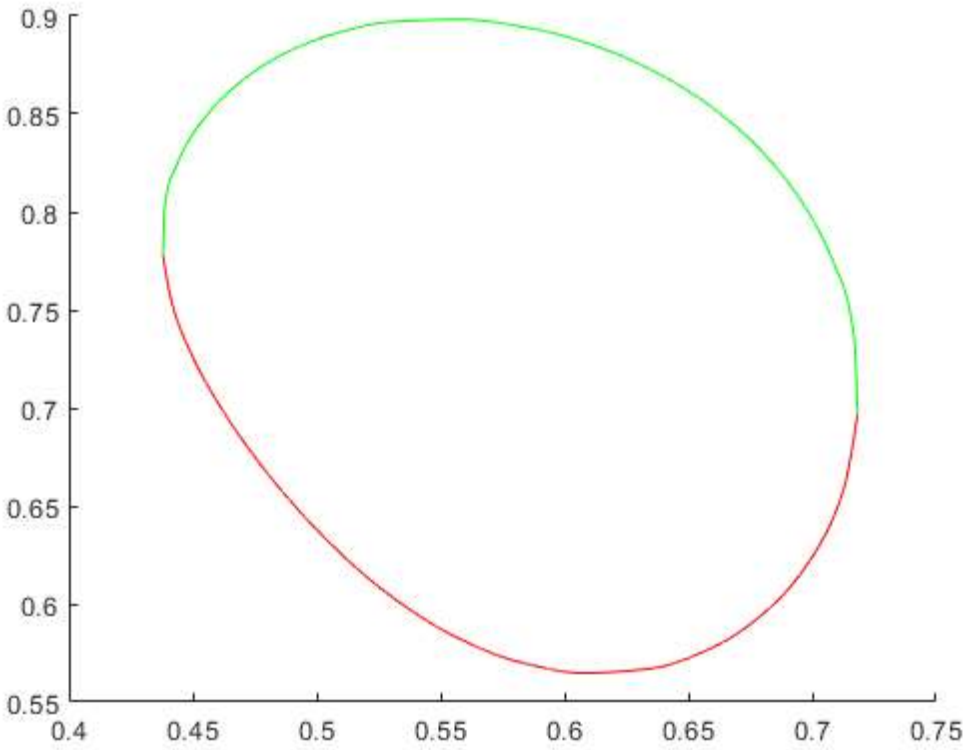
```
ans =

  4×1 table

               Results
    k    n       error       accuracy
    ____________________________________

    1    10       0.33616          0
    2    20      0.030214     3.4758
    3    40     0.0022574     3.7425
    4    80    0.00012505     4.1741
```



Published with MATLAB® R2021b

```
function S = P3
[x,y]=markers_Euler(32,0.1,0.0001);
    enum = abs(0.15^2*pi-polyarea(x,y));
    enum2 = abs(0.15^2*pi-myarea(x,y));
R = table(enum,enum2, 'VariableNames',{'polyerror','myerror'});
S = table(R,'VariableNames',{'Results'});
```

```
ans =

  table

          Results
    polyerror     myerror
    ____________________

    0.0004771    1.8951e-05
```



Published with MATLAB® R2021b