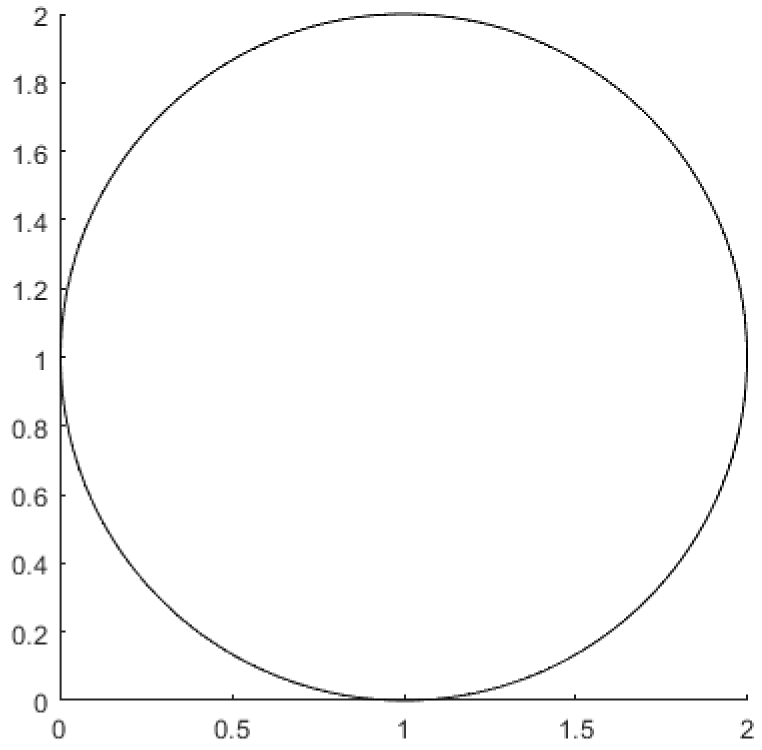
Section 3: Function to provide pi value of specified precision

```
function monteCarloValue = getMonteCarloPiVal(desiredSigFigs)
    % Validate that input is a single integer
    validateattributes(desiredSigFigs, {'double'}, {'scalar', 'integer',
'nonzero', 'positive'});

    % Create parameter arrays for a circle to draw on graph
    theta = linspace(0, 2*pi(), 256);
    circleX = cos(theta)+1;
    circleY = sin(theta)+1;

    % Initialize point and value arrays for Monte Carlo calculations
    generatedPoints=[rand(2,1)];
    insideCircle = [[0.5;0.5]];
    outsideCircle = [[1;1]];
    piValues = [4];
    piVal = 0;

    % Create plot and setup circle, axes
    figure;

    hold on
    plot(circleX, circleY, '-k');

    xlim([0,2]);
    ylim([0,2]);
    axis square;

    % While the provided pi value does not possess the desired number of
    % sig figs:
    %   - generate a new point and add it to the array of random points,
    %   - re-calculate a pi value based on the updated array,
    %   - check if sig figs are met or if maximum attempts have been
    %   expended
    %   - update the plot of generated values with the newest point and
    %   annotation of calculated pi value
    while ~meetsSigFigs(desiredSigFigs, piValues)
        newestPoint = 2*rand(2,1);
        generatedPoints = cat(2, generatedPoints, newestPoint);
        piVal = getPiValFromCoords(generatedPoints);
        piValues(numel(piValues)+1) = piVal;
        if meetsSigFigs(desiredSigFigs, piValues)
            break;
        elseif numel(generatedPoints(1,:)) > 10^8
            disp('maximum tries expended')
            break;
        end

        delete(findall(gcf, 'type', 'annotation'));
        if distanceToCenter(newestPoint) <= 1
            outsideCircle = cat(2, outsideCircle,
[newestPoint(1);newestPoint(2)]);
            plot(newestPoint(1,1), newestPoint(2,1), 'xb');
        else
            insideCircle = cat(2, insideCircle,
[newestPoint(1);newestPoint(2)]);
            plot(newestPoint(1,1), newestPoint(2,1), 'xr');
        end
        annotation('textbox', [0,0.1,.3,.3], 'String', ['Pi value: ',
string(piVal)], 'FitBoxToText', 'on');
        annotation('textbox', [0,0,.3,.3], 'String', ['Sample size:  ',
string(numel(piValues))], 'FitBoxToText', 'on');
        drawnow;
    end

    % On successful generation, perform final update of chart and write
output in console, assign
    % generated value to return variable
    delete(findall(gcf, 'type', 'annotation'));
    annotation('textbox', [0,0.1,.3,.3], 'String', ['Pi value: ',
string(piVal)], 'FitBoxToText', 'on');
    annotation('textbox', [0,0,.3,.3], 'String', ['Sample size:  ',
string(numel(piValues))], 'FitBoxToText', 'on');
    drawnow;

    fprintf('Pi value with %d significant figure(s): %f \n', desiredSigFigs,
piVal);
    fprintf('Number of points required for %d significant figure(s): %d \n',
desiredSigFigs, numel(generatedPoints(1,:)));
    monteCarloValue = piVal;
end
```

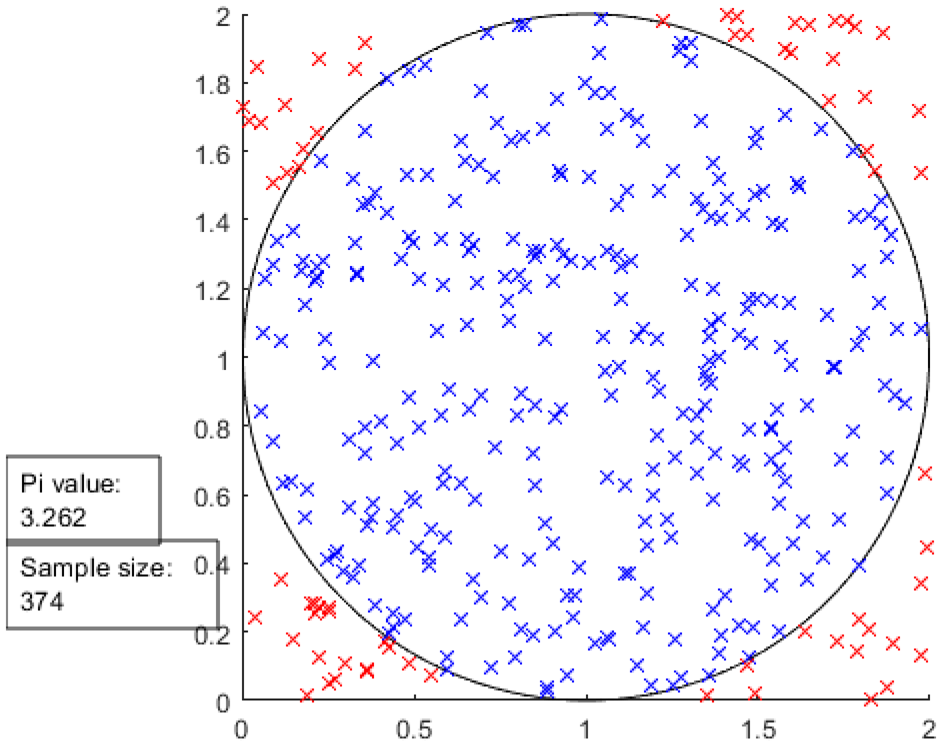


Helper functions

```
% Function to check whether the provided pi value matches pi up to the
% desired quantity of significant figures - checks if the last 30 values
% generated have the same value in the desired significant figure place
% and if they are returns true
function hasFigs = meetsSigFigs(sigfigs, valueArray)
    hasFigs = false;
    minVal = numel(valueArray) - 30;
    difference = 10^(-sigfigs+1);
    if minVal <= 0
        return
    else
        for val = (minVal+1):numel(valueArray)
            if abs(valueArray(minVal) - valueArray(val)) > difference
                hasFigs = false;
                break;
            else
                hasFigs = true;
            end
        end
    end
end
```

```matlab
% Function that accepts an array of randomly generated coordinates
% and provides a pi value based on the number that fall within
% a circle, as well as two arrays of points inside and outside the circle
function valueToReturn = getPiValFromCoords(coordArray)
    intCount = 0;
    for coord = 1:numel(coordArray(1,:))
        if distanceToCenter([coordArray(1,coord),coordArray(2,coord)]) <= 1
            intCount = intCount + 1;
        end
    end
    proportion = intCount/numel(coordArray(1,:));
    valueToReturn = proportion*4;
end


% Function to calculate distance between point argument (treated as [x;x])
% and center of square, taken to be [0.5;0.5]
function dist = distanceToCenter(point)
    dist = sqrt((1-point(1))^2 + (1-point(2))^2);
end
```

```
Pi value with 3 significant figure(s): 3.262032
Number of points required for 3 significant figure(s): 374

ans =

    3.2620
```



Published with MATLAB® R2024a