Abelardo Riojas
ISC 4221C Fall 2019
Prof. Quaife

Lab 6 Report

1. Computing Integrals with Monte Carlo
    a. Write a code that takes as its inputs three vertices of a triangle and returns its area. Demonstrate that your code works. The answer should be exact up to round-off error

```
A = [4;0];
B = [3;4];
C = [0;1];

area = triArea(A,B,C)

function area = triArea(A,B,C)
    A = [A;0]; %append 0 to the ends of all points
    B = [B;0];
    C = [C;0];
    AB = B - A;
    AC = C - A ;
    BC = B - C;
    area = 0.5 * norm(cross(AC, BC));
end
```

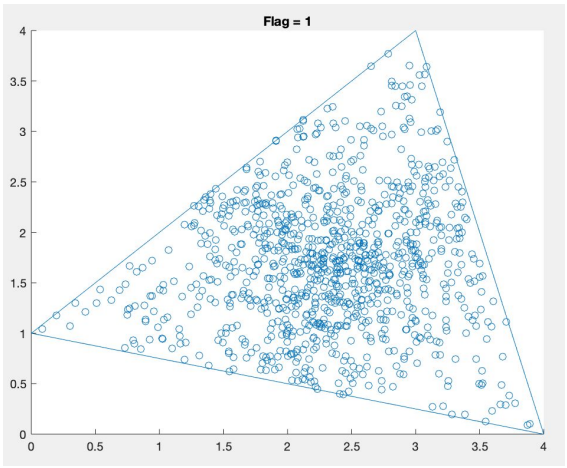
Output from the command line:

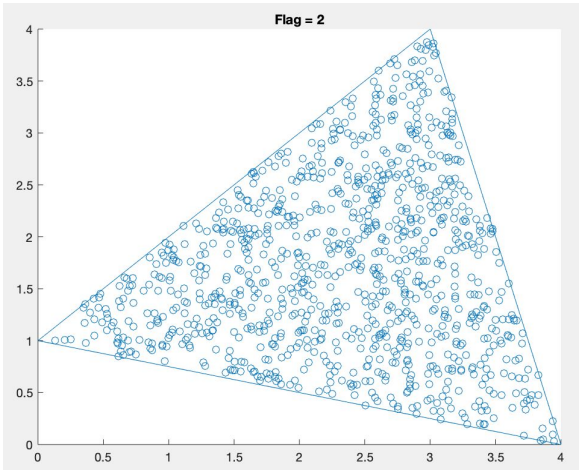
```
area =

    7.5000
```

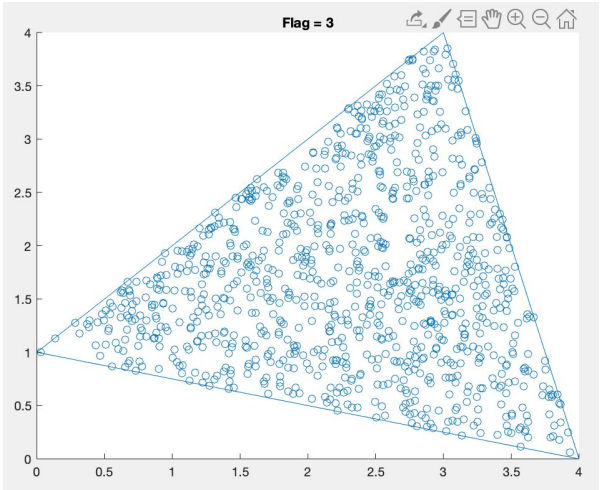
    b. Random points in the triangle

```
function pts = triSample(A,B,C,n,flag)
pts = [];
    for i = 1:n
        r1 = rand();
        r2 = rand();
        if flag == 1
            r3 = rand();
            alpha = r1/(r1+r2+r3);
            beta =  r2/(r1+r2+r3);
            gamma = r3/(r1+r2+r3);
            point = alpha*A + beta*B + gamma*C;
```

```
        elseif flag == 2
            if r1+r2 > 1
                r1 = 1-r1;
                r2 = 1-r2;
            end
            alpha = 1 - r1 - r2;
            beta = r1;
            gamma = r2;
            point = alpha*A + beta*B + gamma*C;
        else
            alpha = 1 - sqrt(r1);
            beta = sqrt(r1)*r2;
            gamma = sqrt(r1)*(1-r2);
            point = alpha*A + beta*B + gamma*C;
        end
    pts = [pts point];
    end
end
```







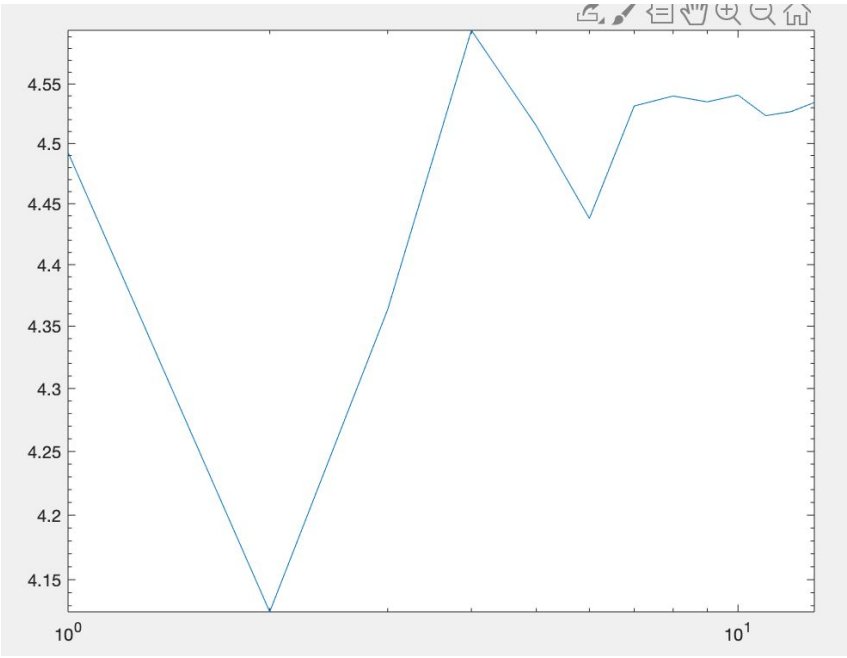
The difference between them is that the first method does not give an even distribution of samples from inside the triangle.

c. Write a code that uses your routines for computing triangle areas and selecting random points inside the triangle to develop a Monte-Carlo method to approximate integrals over triangles.

```
function integral = monteTri(A,B,C,n,f,flag)
sum = 0;
samples = triSample(A,B,C,n,flag);
    for i = 1:n
        sum = sum + f(samples(1,i),samples(2,i));
    end
integral = abs(triArea(A,B,C))*(1/n)*sum;
end
```

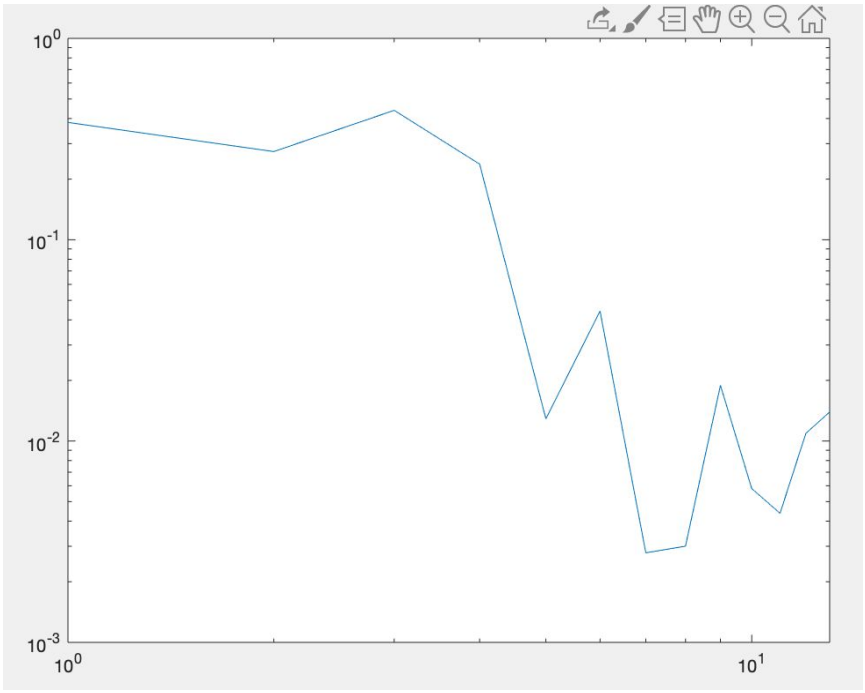
d. Use your code to approximate the above integral with the three different choices for choosing random points. Approximate the integral 100 times and compute the mean value of the approximation. Compare it to the exact value of the integral by plotting the error vs. N on a loglog axis.

```
f = @fxy;
error = [];
for i = 6:18
    val = 0;
    for j = 1:100
        val = val + monteTri(A,B,C,2^i,f,1);
    end
    val = val/100;
    err = abs(72.5 - val);
    error = [error;err];
end
loglog(error);
```

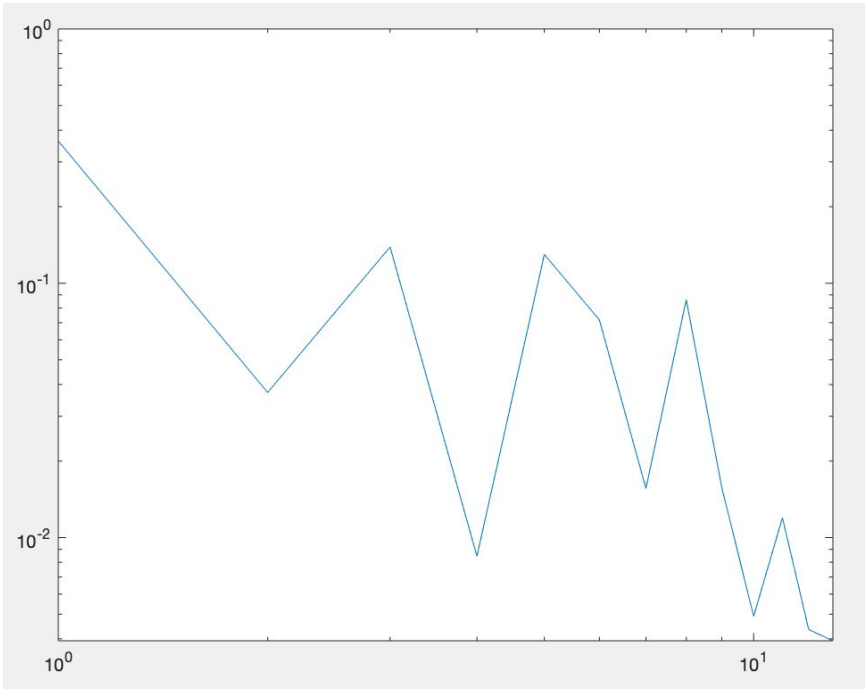


Method 1

Method 2



Method 3



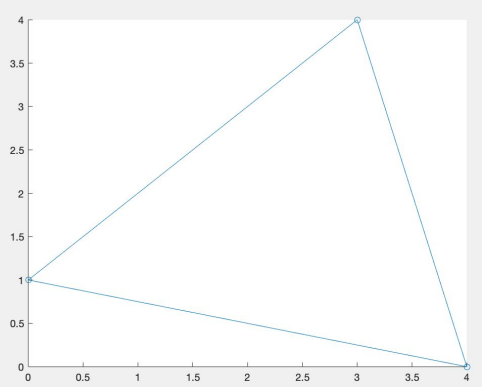
For Monte Carlo methods, it can be shown that the error is approximately equal to $n^{-½}$. Both methods that converge (2 and 3) have error that approaches $1000^{-½} \sim .0316$.

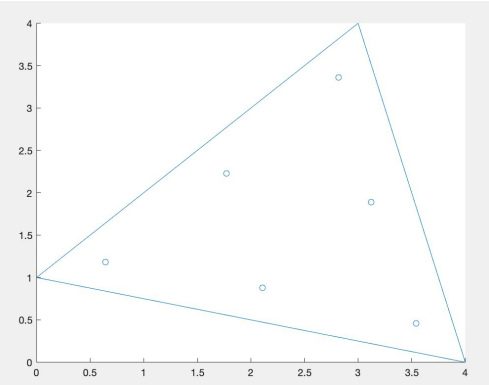
Error for method 2 with 2^{18} samples = .0096
Error for method 3 with 2^{18} samples = .0039

2. Computing Integrals with Quadrature
    a. Write a code that takes as its inputs the three vertices of the triangle and a flag for the quadrature rule that is to be used.

```
function [q,Xvec,Yvec] = quadrature(ax,ay,bx,by,cx,cy,flag)
A = [ ax; ay ];
B = [ bx; by ];
C = [ cx; cy ];
    if flag == 1
        xvec = [1;0;0];
        yvec = [0;0;1];
        wvec = [1/3;1/3;1/3];
    else
        a = 0.816847572980459;
        b = 0.091576213509771;
        c = 0.108103018168070;
        d = 0.445948490915965;
        u = 0.109951743655322;
        v = 0.223381589678011;


        xvec = [ a; b; b; c; d; d ];
        yvec = [ b; a; b; d; c; d ];
        wvec = [ u; u; u; v; v; v ];
    end

% coordinates of TABC

% The linear translation of quadrature points
Xvec = ax * xvec + bx * yvec + cx * ( 1 - xvec - yvec );
Yvec = ay * xvec + by * yvec + cy * ( 1 - xvec - yvec );
Fvec = f(Xvec, Yvec);
% compute area
Area = triArea(A,B,C);
% the integral
q = Area * wvec' * Fvec;

end
```

b. Use your code to compute the quadrature points in the above mentioned triangle. Plot the quadrature points and include the edges of the triangle in the plot. There should be two plots—one for each set of quadrature points.





c. Use the routines you have developed to approximate the example integral with the quadrature formula. Use both sets of quadrature points. If implemented correctly, one of the errors should be much smaller than the other. Explain why the error is so small.

Method 1: Integral ~ 105.0
Method 2: Integral = 72.5

The error is so small because using 6 points we can devise a rule through deg = 4 to get the exact value of the integral.

$$
\begin{aligned}
f(x,y) = \quad & a \\
& + bx + cy \\
& + dx^2 + exy + fy^2 \\
& + gx^3 + hx^2y + ixy^2 + jy^3 \\
& + kx^4 + lxy^3 + mx^2y^2 + nxy^3 + oy^4
\end{aligned}
$$