Cálculo de π

Rubén Romero Ortega 174178

Monte Carlo methods, or Monte Carlo experiments, are a broad class of computational algorithms that rely on repeated **random sampling** to obtain numerical results.

https://en.wikipedia.org/wiki/Monte_Carlo_method#Integration

Area method

This method enables you to estimate areas (**definite integrals**) by generating a uniform sample of points and counting how many fall into a planar region.

If you generate a 2-D point (an ordered pair) uniformly at random within the unit square, then the probability that the point is inside the quarter circle is equal to the ratio of the area of the quarter circle divided by the area of the unit square. That is, P(point inside circle) = Area(quarter circle) / Area(unit square) = $\frac{\pi}{4}$

It is easy to use a Monte Carlo simulation to estimate the probability P:

1. generate *n* random points uniformally distributed inside the unit square, and
2. count the proportion that fall in the quarter circle

Genera 10000 puntos aleatorios

```
nRand=10000;
```

Encuentra los puntos que se encuentran dentro del área de interés (distancia al origen menor a 1) y fuera de ella

```
%Crear vector aleatorio x y y
xRand=rand(1,nRand);
yRand=rand(1,nRand);

%Calcular radio como operacion vectorial
rRand=xRand.^2+yRand.^2;

%Vector booleano
nIn= rRand<1;
nOut=rRand>=1;
totalPuntosInternos=sum(nIn);
totalPuntosInternos
```

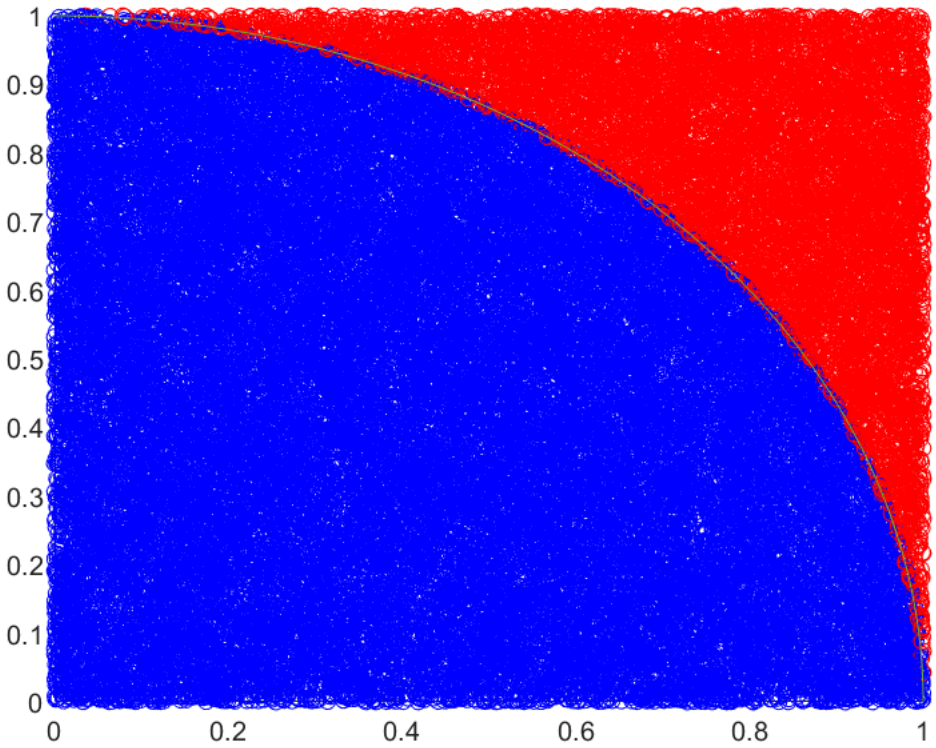
```
totalPuntosInternos = 7877
```

```
piaprox=4*totalPuntosInternos/nRand;
piaprox
```

```
piaprox = 3.1508
```

Grafica los puntos que caen adentro en azul y los que caen afuera en rojo

```
angles=linspace(0,pi/2,200);
xCir=cos(angles);
yCir=sin(angles);
hold on
xRandIn=xRand(nIn);
yRandIn=yRand(nIn);
xRandOut=xRand(nOut);
yRandOut=yRand(nOut);
scatter(xRandIn,yRandIn,'blue');
scatter(xRandOut,yRandOut,'red');
plot(xCir,yCir);
```



Encuentra la proporción de puntos que están adentro (p) y de puntos que están afuera (q)

```
p=sum(nIn)/nRand;
q=1-p;
```

Encuentra la desviación estándar (proceso de Bernoulli)

```
std=sqrt(p*q/nRand);
std
```

```
std = 0.0041
```

Calcula el intervalo de confianza (95%) para una proporción

```
alpha=.05;
```

```
za2=1.96;
min=p-za2*std;
max=p+za2*std;
min
```

```
min = 0.7797
```

```
max
```

```
max = 0.7957
```

Repite el mismo experimento 100 veces, calcula la varianza, el error estándar y el intervalo de confianza.

```
aux=zeros(1,100);
for cont=1:100
    %Crear vector aleatorio x y y
    xRand=rand(1,nRand);
    yRand=rand(1,nRand);

    %Calcular radio como operacion vectorial
    rRand=xRand.^2+yRand.^2;

    %Vector booleano
    nIn= rRand<1;
    p=sum(nIn)/nRand;
    aux(cont)=p;
    %Meter al aux
end
M=mean(aux)
```

```
M = 0.7857
```

```
V=var(aux)
```

```
V = 1.4103e-05
```

```
errorRelativo=abs(M-pi)/pi
```

```
errorRelativo = 0.7499
```

```
za2=1.96;
min=M-za2*sqrt(V);
max=M+za2*sqrt(V);
min
```

```
min = 0.7783
```

```
max
```

```
max = 0.7930
```

Fórmula de Leibniz

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots$$

Calcula el valor de π usando la fórmula anterior. Detén el cálculo cuando la diferencia relativa entre el valor actual calculado y el anterior calculado sea menor a 0.0000001 o se hayan realizado 200 mil iteraciones.

```
aux=0;
intentos=1;
dif=1;

while(intentos<200000 || dif>0.0000001)

    aux=aux+(-1)^(intentos+1)*1/(2*intentos-1);
    dif=1/(2*intentos-1);
    intentos=intentos+1;
end
piaprox2=aux*4;
errorRelativo=abs(pi-piaprox2);
piaprox2
```

```
piaprox2 = 3.1416
```

```
intentos
```

```
intentos = 5000002
```

```
errorRelativo
```

```
errorRelativo = 2.0000e-07
```