Ujjwal

April 28, 2022

Objectives : **plot** $log(avg(\delta_N))$ **vs** $log(N)$ **and find linear function that best approximate the data using least square method**

updated Calculation
for particular N number of random points , 100 iteration is performed on Error function .and the average is noted.

[4]:
```
using Distributions
using Plots
using LinearAlgebra
using Printf
```

[5]:
```
lowerLimit = 0
upperLimit = pi;
```

[6]:
```
function approxIntegral(n)
        v = rand(Uniform(lowerLimit,upperLimit),n) # create random vector v of␣
 ↪n elements within given limits
        f_appliedTo_v = sin.(v)# apply sin function elementwise to vector v    ␣
 ↪
        integral = sum(f_appliedTo_v) # sum the elements of vector obtained␣
 ↪after applying function
        ans = (upperLimit - lowerLimit)/n * integral # calclulate ans
        return ans
end
```

[6]:
```
approxIntegral (generic function with 1 method)
```

[7]:
```
trueVal = 2.0 # represents actual value of integral
function Error(n) # function with input : n number of points          output :
 ↪error
        approxVal = approxIntegral(n) # generate approxval by calling functon␣
 ↪approxIntegral
        errorObt = abs(approxVal - trueVal)/trueVal # calculate error
        return errorObt
end
```

[7]:
```
Error (generic function with 1 method)
```

let the number of sample be 10 for $N_{specific}$

[8]:
```
samples = 100 # 100 samples having n number of random points is selected
function avgError(n) # function which calculated average of error of 100 samples
    sum = 0
    for sample in 1:samples
        sum = sum + Error(n)
    end
    return sum/100
end
```

[8]:
```
avgError (generic function with 1 method)
```

If N be the number of randomly choosen points for Monte Carlo Integration , then the average error (100 samples) introduced on choosing N be denoted by $avg(\delta N)$
let the relation between $avg(\delta N)$ and $ N $ be modeled by following equation

$$avg(\delta N) = C.N^{\alpha}$$

$$log(avg(\delta_N)) = logC + \alpha.log(N)$$

Linear Regression Theory
To fit a collection of data

$$(x_i, y_i)$$

to a straight line that minimizes the squares of the differences between the predicted y values and the actual y_i , we solve the following system:

$$\left(\sum_{i=1}^{n} x_i^2\right) a + \left(\sum_{i=1}^{n} x_i\right) b = \sum_{i=1}^{n} x_i y_i$$

$$\left(\sum_{i=1}^{n} x_i\right) a + nb = \sum_{i=1}^{n} y_i.$$

writing equation in matrix form

$$\begin{bmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & n \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} y_i \end{bmatrix}$$

solving matrix gives a and b and the equation of best fit line is given by $y = ax + b$

Linear Regression Implimentation

for graphing convenience N is choosen as follows

$$\begin{bmatrix} 1000 & 2000 & 3000 & .... & 100000 \end{bmatrix}$$

[9]:
```
N =  1000 : 1000 : 100000
avgErrors = avgError.(N)
```

```
# taking log of data
X = log.(N)
Y = log.(avgErrors);
```

$$X = log(N)$$
$$Y = log(avg(\delta_N))$$

[10]:
```
#fit actuall error in straight line
A = [transpose(X)*(X) sum(X); sum(X) size(X)[1]]
B = [sum(X.*Y);sum(Y)]
#solve matrix for a and b
linApp = A\B
slope = linApp[1]
yIntercept = linApp[2]
# interpreting the result
alpha = slope
C = exp(yIntercept);
```

Average Error $avg(\delta_N)$ at given specific points N_specific = [100 300 1000 3000 10000 30000 100000 300000]

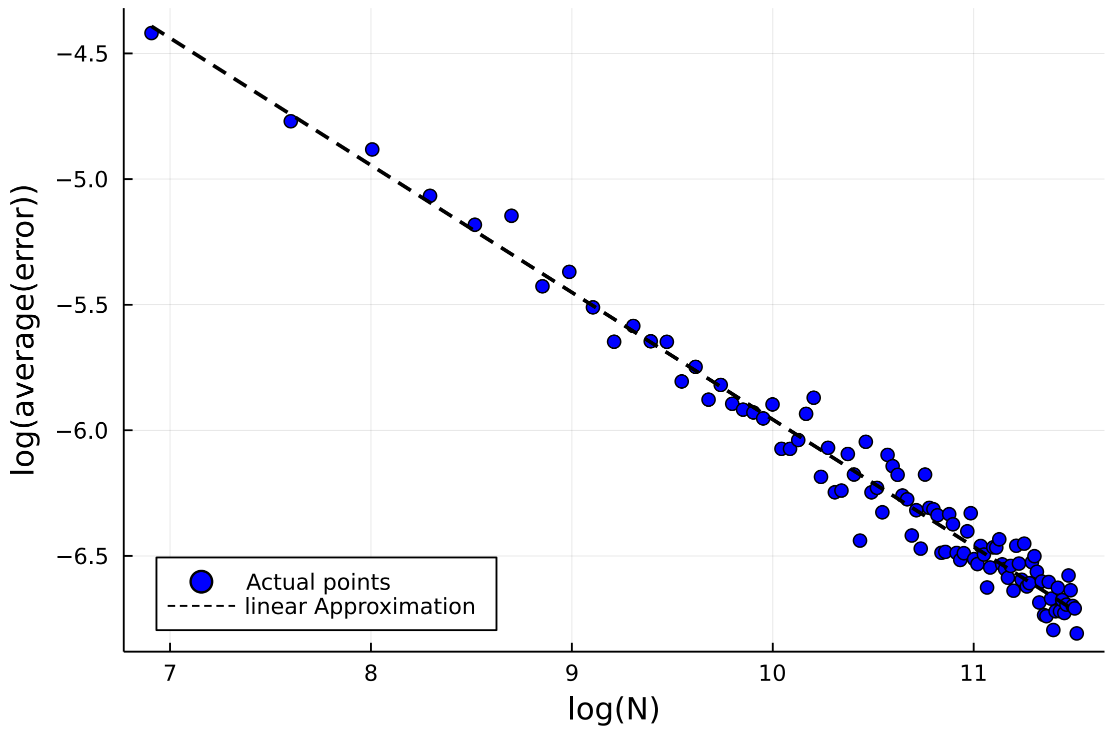
[11]:
```
N_specific = [100 300 1000 3000 10000 30000 100000 300000]
# error at N_specific
Error_specific = avgError.(N_specific)
println(N_specific)
Error_specific = round.(Error_specific,digits = 5)
println(Error_specific)
```

```
[100 300 1000 3000 10000 30000 100000 300000]
[0.03646 0.01859 0.01221 0.0064 0.00421 0.00212 0.00114 0.00075]
```

plotting $log(avg(\delta_N))$ vs $log(N)$ and approximate linear function

[12]:
```
#plot actuall  points
scatter(X,Y,label = "Actual points",xaxis = "log(N)",yaxis =␣
 ↪"log(average(error))",color = :blue)
# plot approximation
f(points) = slope * points  + yIntercept
plot!(f,linewidth = 2.0,linestyle = :dash,label = "linear Approximation",color␣
 ↪= :black,legend = :bottomleft)
```

[12]:



final Result

[13]:
```
# display result
@printf "alpha(slope of Line),  = %.2f " alpha
@printf "yIntercept = %.2f " yIntercept
```

```
alpha(slope of Line),  = -0.51 yIntercept = -0.90
```

created by : Ujjwal