# PS6

May 7, 2017

### 0.1 PS 6

### 0.2 Estimation of Variance For Multiple Segments

Previously we've found that $\sigma_I$ for single segment can be estimated using the following relation.

$$\sigma_I = \frac{1}{12\sqrt{5}} \frac{(b-a)^3}{\sqrt{N}} g'' \left( \frac{a+b}{2} \right)$$

where $g = \frac{f}{w}$. For multiple segments, we sum up the variance for each segment and find the standard deviation.

Then the relation becomes

$$\sigma_I = \frac{1}{12\sqrt{5}} \frac{(b-a)^3}{M^2\sqrt{N}} \sqrt{\sum_i \frac{1}{M} g'' \left( x_{mid}^{(i)} \right)}$$

,

or,

$$\sigma_I = \frac{1}{12\sqrt{5}} \frac{(b-a)^3}{M^2\sqrt{N}} g''_{rms}$$

,

where $M$ is the number of segments. However this formula is not practical since $g$ is calculated using different weight functions. Therefore we would like to write this down in terms of $f$. Going back to the relation for one segment, instead of $g''$ we can write,

$$g'' = \frac{f'' - 2f'w'w + 2fw'^2}{kw^3}$$

.

This is a complicated expression and we can simplify this by assuming that $w$ is almost constant for a very small region.

$$g'' \simeq \frac{f''}{(b-a)}$$

.

Note that all functions are evaluated at the middle point unless specified otherwise.

Then the relation for single segment becomes,

$$\sigma_I = \frac{1}{12\sqrt{5}} \frac{(b-a)^4}{\sqrt{N}} f'' \left( \frac{a+b}{2} \right)$$

Let's test this result.

```
In [83]: import numpy as np
         from numpy.polynomial import Polynomial as P
         #import plotly
         #import plotly.plotly as py
         #import plotly.figure_factory as ff
         import matplotlib.pyplot as plt
         #Integrand function
         def f(x,H):
             return (x-5)*np.exp(-(x/2-3))+100+H

         #Calculates the coefficients of linear weight function.
         def findw(f,H,lower,upper,normalize):
             #Find the linear function.
             slope=(f(upper,H)-f(lower,H))/(upper-lower)
             a=slope
             b=-slope*upper+f(upper,H)
             #Normalization.
             A=(a/2)*(upper**2)+b*upper-(a/2)*(lower**2)-b*lower
             if normalize:
                 a/=A
                 b/=A
             return [a,b]
         #Performs integration.
         def integrate(f,lower,upper,N,C):
             H=C
             w=findw(f,H,lower,upper,True)
             #Generate uniform random inputs.
             inputs=np.random.rand(N)
             a=w[0]/2
             b=w[1]
             c=-(a*lower**2+b*lower)

             SUM=0
             SUM2=0

             inverse_inputs=[]
             for i in inputs:
                 p=[(-b-np.sqrt(b**2-4*a*(c-i)))/(2*a),(-b+np.sqrt(b**2-4*a*(c-i)))
                 if p[0]>=lower and p[0]<=upper:
                     inverse_inputs.append(p[0])
                 else :
                     inverse_inputs.append(p[1])

             inverse_inputs=np.array(inverse_inputs)
             #Calculate f(inverse(x))/w(inverse(x)).
             outputsF=f(inverse_inputs,H)
             outputsW=w[0]*(inverse_inputs)+w[1]
             outputs=outputsF/outputsW
```

```
    SUM=outputs.sum()
    SUM2=(outputs*outputs).sum()
    var=SUM2/N-(SUM/N)**2
    var=var/N
    #Store generated points for variance calculation.
    Vsum=outputs.sum()
    return Vsum/N-H*(upper-lower),(upper-lower)**2*var




def theoretical_sigma(f,lower,upper,N,C):

    w=findw(f,C,lower,upper,True)

    flower=f(lower,C)
    fmiddle=f((lower+upper)/2,C)
    fupper=f(upper,C)

    wlower=w[0]*lower+w[1]
    wmiddle=w[0]*(lower+upper)/2+w[1]
    wupper=w[0]*upper+w[1]

    gupper=fupper/wupper
    gmiddle=fmiddle/wmiddle
    glower=flower/wlower



    #sigma=np.abs(0.03727*(upper-lower)**3/np.sqrt(N)*(4*(gupper-2*gmiddle
    sigma=np.abs(1/np.sqrt(720)*(upper-lower)**4/np.sqrt(N)*(4*(fupper-2*f
    #sigma=np.abs(1/np.sqrt(720)*(upper-lower)**3/np.sqrt(N)*(4*(gupper-2

    #return (upper-lower)*np.sqrt(var)
    return sigma

low=5
up=5.4
#Divide the region into 10 pieces.
l=[low,up]
#Real value of the integral
I_real=-.12002
#N values
N=[100,500,1000,5000,10000,50000,100000]
#Integration results.
results=[]
#Standart deviation values
sigmas=[]
```

```
    theory_sigmas=[]
    print('N','   sigma_exp','        sigma_theory','    sigma_theory/sigma_exp

    for k in N:
        I=0
        sigma=0
        S=0
        for i in range (0,len(l)-1):
            temp_sigma=[]
            for j in range(0,10):
                I=0
                sigma=0
                temp,temp2=integrate(f,l[i],l[i+1],k,S)
                I+=temp
                sigma+=temp2
                temp_sigma.append(np.sqrt(sigma))
        results.append(I)
        sigmas.append(np.mean(temp_sigma))
        theory_sigmas.append(theoretical_sigma(f,low,up,k,S))
        sigma=np.mean(temp_sigma)
        print(k,sigma,theoretical_sigma(f,low,up,k,S),theoretical_sigma(f,low,
```

```
N    sigma_exp        sigma_theory     sigma_theory/sigma_exp
100 0.000133999449559 0.00013544336242 1.01077551338
500 6.16092621007e-05 6.05721130945e-05 0.983165696669
1000 4.29084294641e-05 4.28309519199e-05 0.998194351433
5000 1.91442741979e-05 1.91545840068e-05 1.00053853224
10000 1.34911874644e-05 1.3544336242e-05 1.00393951813
50000 6.06141598961e-06 6.05721130945e-06 0.999306320476
100000 4.28676333724e-06 4.28309519199e-06 0.999144308896
```

Let's apply the same relation for each segment and add it up.
This gives,

$$\sigma_I = \frac{1}{12\sqrt{5}} \frac{(b-a)^4}{M^3\sqrt{N}} \sqrt{\sum_i \frac{1}{M} f'' \left( \frac{a_i + b_i}{2} \right)}$$

so,

$$\sigma_I = \frac{1}{12\sqrt{5}} \frac{(b-a)^4}{M^3\sqrt{N}} f''_{rms}$$

```
In [84]: def theoretical_sigma(f,lower,upper,N,M,C):

        l=np.arange(lower,upper+(upper-lower)/M,(upper-lower)/M)
    # sigma=np.abs(0.03727*(upper-lower)**3/np.sqrt(N)*(4*(gupper-2*gmiddle+gl
        sigma2=0
        for i in range(0,len(l)-1):
```

```
                lo=l[i]
                hi=l[i+1]
                flower=f(lo,C)
                fmiddle=f((lo+hi)/2,C)
                fupper=f(hi,C)
                sigma2+=(4*(fupper-2*fmiddle+flower)/(hi-lo)**2)**2
            sigma2/=M
           # print(sigma2)
            return  np.abs(1/np.sqrt(720)*(upper-lower)**4/(M**3*np.sqrt(N))*np.sc


        print('N','   sigma_exp','       sigma_theory','    sigma_theory/sigma_exp

        #Divide the region into 10 pieces.
        l=np.arange(2,10.01,0.8)
        #Real value of the integral
        I_real=-16.6728
        #N values
        N=[100,1000,10000,100000,1000000]
        #Integration results.
        results=[]
        #Standart deviation values
        sigmas=[]
        for k in N:
            I=0
            sigma=0
            S=0
            for i in range (0,len(l)-1):
                temp,temp2=integrate(f,l[i],l[i+1],int(0.1*k),0)
                I+=temp
                sigma+=temp2
            results.append(I)
            sigmas.append(np.sqrt(sigma))
            print(k,np.sqrt(sigma),theoretical_sigma(f,2,10,k,10,S),theoretical_si
```

```
N    sigma_exp        sigma_theory     sigma_theory/sigma_exp
100 0.0503844789728 0.0596157803733 1.18321716506
1000 0.0201257805041 0.0188521650468 0.936717214171
10000 0.00602545361828 0.00596157803733 0.989399041965
100000 0.00189355222028 0.00188521650468 0.995597842238
1000000 0.000601032078808 0.000596157803733 0.991890158201
```