PS4

April 23, 2017

0.1 Estimation Of Variance For Single Segment

Let's try to estimate σ_I analytically for one segment.

$$\sigma_I^2 = \left\langle \frac{f^2}{w^2} \right\rangle - \left\langle \frac{f}{w} \right\rangle^2$$

Let $g = \frac{f}{w}$, then,

$$\sigma_{f/w}^2 = \langle g^2 \rangle - \langle g \rangle^2$$

We can expand g using Taylor Series around $\frac{b+a}{2}$,

$$g(x) = g\left(\frac{b+a}{2}\right) + g'\left(\frac{b+a}{2}\right)\left(x - \frac{b+a}{2}\right) + \frac{1}{2}g''\left(\frac{b+a}{2}\right)\left(x - \frac{b+a}{2}\right)^2$$

We can also aprrroximate first and second derivative very accurately for a small region,

$$g'\left(\frac{b+a}{2}\right) = \frac{g(b) - g(a)}{b - a}$$

$$g''\left(\frac{b+a}{2}\right) = \frac{g(b) - 2g(\frac{b+a}{2}) + g(a)}{(\frac{b-a}{2})^2}$$

To simplify equations we can define the following variables,
$p = g(\frac{b+a}{2})$
$q = \frac{g(b)-g(a)}{b-a}$
$r = \frac{g(b)-2g(\frac{b+a}{2})+g(a)}{2(\frac{b-a}{2})^2}$
$u = (x - \frac{b+a}{2})$
Then,

$$g(x) = ru^2 + qu + p$$

Since we forced w to be equal to f at the boundaries, q is 0.
Our job reduces to calculate

$$\langle (ru^2 + p)^2 \rangle - \langle ru^2 + p \rangle^2$$

Expanding the terms,

$$\sigma^2_{f/w} = \left\langle r^2u^4 + 2rpu^2 + p^2 \right\rangle - \left\langle ru^2 + p \right\rangle^2$$

$$= (r^2 \left\langle u^4 \right\rangle + 2rp \left\langle u^2 \right\rangle + p^2) - (r^2 \left\langle u^2 \right\rangle^2 + 2rp \left\langle u^2 \right\rangle + p^2)$$

$$= r^2 \left\langle u^4 \right\rangle - r^2 \left\langle u^2 \right\rangle^2$$

$$= r^2(\left\langle u^4 \right\rangle - \left\langle u^2 \right\rangle^2)$$

This could further be simplified as,

$$\sigma^2_{f/w} = r^2\sigma^2_{u^2}$$

Finally σ_I can be estimated as,

$$\sigma_I = \frac{(b-a)}{\sqrt{N}} r\sigma_{u^2} = \frac{(b-a)}{\sqrt{N}} \frac{g(b) - 2g(\frac{b+a}{2}) + g(a)}{2(\frac{b-a}{2})^2} \sqrt{\left(\frac{2(b-a)^4}{5 \cdot 2^5} - \frac{4(b-a)^4}{9 \cdot 2^6}\right)}$$

$$\sigma_I = \sqrt{\left(\frac{2(b-a)^2}{5 \cdot 2^5} - \frac{4(b-a)^2}{9 \cdot 2^6}\right)} \frac{2}{\sqrt{N}} (g(b) - 2g(\frac{b+a}{2}) + g(a))$$

or

$$\sigma_I = \frac{0.03727(b-a)^3}{\sqrt{N}} g''\left(\frac{a+b}{2}\right)$$

In [16]:
```
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))+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+


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

    low=4.6
    up=5.2
    #Divide the region into 10 pieces.
    l=[low,up]
    #Real value of the integral
    I_real=-.12002
    #N values
    N=[100,1000,10000,100000,1000000]

    #Integration results.
    results=[]
    #Standart deviation values
    sigmas=[]
    for k in N:
        I=0
        sigma=0
        S=1000
        for i in range (0,len(l)-1):
            temp,temp2=integrate(f,l[i],l[i+1],k,S)
            I+=temp
            sigma+=temp2
        results.append(I)
        sigmas.append(np.sqrt(sigma))
        print(k,I,I-I_real,np.sqrt(sigma),theoretical_sigma(f,low,up,k,S)/np.s
```

```
100 -0.121168988625 -0.00114898862507 0.000830959217088 1.03652094663 -1.382725651
1000 -0.120463231735 -0.00044323173513 0.000278923898527 0.976499590964 -1.5890776
10000 -0.119894524822 0.000125475178084 8.62560747757e-05 0.998546057823 1.4546822
100000 -0.120008293845 1.17061545557e-05 2.7295876215e-05 0.997839639497 0.4288616
1000000 -0.120011109322 8.89067807795e-06 8.63024146321e-06 0.998009891122 1.03017
```

In [ ]: