Numerical Integration

Shivam Kumaran , SC17B122 22/10/2020

1 Problem

1. Consider the following integration:

$$\int_{-1}^{1} \sqrt{1-x^2}dx$$

$$\int_{0}^{2} \frac{1}{\sqrt{x}}dx$$

$$\int_{0}1 \int_{0}^{2} (xy^2)dxdy$$

1. Calculate the true value of the integration
2. Evaluate the integration using Trapezoidal and Simpson's 1/3 rule for different value of interval
3. Estimate and tabulate the percentage of true error for every interval (n)
4. Find the optimum n for error less than 0.01%
5. Plot percentage of true error versus step size

2 Solution

2.1 Code

```
'''
defining both Simpson and Trapezoidal rule
pass argument kind= 'tpz' - for trapezoidal integral
and kind= 'simp' for simpson's rule
'''
def integral(f,a_0,b_0,n,kind='simp'):
    h = (b_0-a_0)/n
    integ = 0
    for i in range(n):
        a = a_0+i*h
        b = a_0 +(i+1)*h
        if(kind=='simp'):
```

```
            integ += (((b-a)/2)/3)*(f(a)+4*(f((a+b)/2))+f(b))
        elif(kind=='tpz'):
            integ += ((b-a)/2)*(f(a)+f(b))
    return(integ)
```

```
def analysis_simp(f,integ_f,a,b,acc=0.01 , growth = 'const'):
'''
If the convergence is extremely slow, having constant
growth of step size will result in computational bottle-neck
for such case I have used geometric growth of the step size .
which can be passed as an argument to the function
growth = 'const' for Arithematic growth of interval
growth = 'exp' for geometrc growth of interval
'''
    n_max = int(2**100)
    n=2
    err=1
    index = []
    f_err = []
    t = integ_f(a,b)
    while((err-acc)>1e-6 and n<n_max):
        approx = integral(f,a,b,n,kind='simp')
        index.append(n)
        err = abs((approx-t)/t)*100
        f_err.append(err)
        if (growth=='const'):
            n = n+2 # keep this even number
        if(growth=='exp'):
            n = n*2
        print(n ,err)
    return(index,f_err)

def analysis_tpz(f,integ_f,a,b,acc=0.01,growth = 'const'):
    n_max = int(2**60)
    n=1
    err=1
    index = []
    f_err = []
    t = integ_f(a,b)
```

```
    while((err-acc)>1e-6 and n<n_max):
        approx = integral(f,a,b,n,kind='tpz')
        index.append(n)
        err = abs((approx-t)/t)*100
        f_err.append(err)
        if (growth=='const'):
            n = n+1 # keep this even number
        if(growth=='exp'):
            n = n*2
        print(n ,err)
    return(index,f_err)
```

3 Results

```
def analysis(f1,integ_f1,a,b,interval, simp_growth='const'):
    #display the true value of integral
    t = integ_f1(a,b)

    # Diplay values of integral calculated for different intervals

    print('true value of integral:' , t)
    print('Interval , Value_tpz , Err tpz , Value_simp ,  error_simp')
    for n in interval:
        if(n%2==0):
            f_a_simp = integral(f1,a,b,n ,kind ='simp')
            e_simp = 100*abs(f_a_simp-t)/t
            f_a_tpz = integral(f1,a,b,n,kind='tpz')
            e_tpz = 100*abs(f_a_tpz-t)/t
            print('{} \t\t {:.4f} \t {:.4f} \t {:.4f} \t
                {:.4f}'.format(n , f_a_tpz ,e_tpz , f_a_simp ,e_simp))
        else:
            f_a_tpz = integral(f1,a,b,n,kind='tpz')
            e_tpz = 100*abs(f_a_tpz-t)/t
            print('{} \t\t {:.4f} \t {:.4f}'.format(n , f_a_tpz ,e_tpz))


    # estimate the percetage of error till error goes to 0.01 percent
    index_simp,err_simp = analysis_simp(f1,integ_f1,a,b , growth=simp_growth)
```

```
    print('Simpson: Took {} intervals to converge error to
            {:.4f}'.format(index_simp[-1],err_simp[-1]))
    index_tpz, err_tpz = analysis_tpz(f1,integ_f1,a,b , growth=simp_growth)
    print('Trapezoidal: Took {} intervals to converge error to
            {:.4f}'.format(index_tpz[-1],err_tpz[-1]))

    #plotting error vs step size
    step_size_simp = (b-a)/np.array(index_simp)
    step_size_tpz = (b-a)/np.array(index_tpz)
    plt.style.use('seaborn-darkgrid')
    plt.xlabel('Step Size')
    plt.ylabel('Error percentage')
    plt.plot(step_size_simp , err_simp)
    plt.plot(step_size_tpz , err_tpz)
    plt.legend(['Simpson ','Trapezoidal'])
    plt.savefig('Problem_{}.png'.format(f1))
    plt.show()
```

3.1 Function (a)

$$\int_{-1}^{1} \sqrt{1-x^2}dx$$

```
def f1(x):
    return ((1-x**2)**0.5)

def integ_f1(a,b):
    from math import asin
    fa = 0.5*(asin(a))+0.5*(a*((1-a**2)**0.5))
    fb = 0.5*(asin(b))+0.5*(b*((1-b**2)**0.5))
    return (fb-fa)
```

```
-------------------------------------------------------
true value of integral: 1.5707963267948966

--------------------------------------------------------
Interval , Value_tpz , Err tpz , Value_simp ,  error_simp
1   0.0000   100.0000
```

2	1.0000	36.3380	1.4880	5.2688
3	1.2571	19.9719		
4	1.3660	13.0361	1.5418	1.8461
5	1.4238	9.3557		
6	1.4588	7.1314	1.5551	1.0016
7	1.4818	5.6673		
8	1.4979	4.6436	1.5606	0.6495
9	1.5096	3.8949		
10	1.5185	3.3277	1.5635	0.4642
11	1.5255	2.8860		
12	1.5310	2.5341	1.5653	0.3529
13	1.5355	2.2482		
14	1.5392	2.0124	1.5664	0.2799
15	1.5423	1.8151		
16	1.5449	1.6480	1.5672	0.2290
17	1.5472	1.5051		
18	1.5491	1.3817	1.5678	0.1919
19	1.5508	1.2743		
20	1.5523	1.1801	1.5682	0.1638
21	1.5536	1.0970		
22	1.5547	1.0232	1.5686	0.1419
23	1.5558	0.9573		
24	1.5567	0.8982	1.5688	0.1246
25	1.5575	0.8449		
26	1.5583	0.7967	1.5691	0.1104
27	1.5590	0.7530		
28	1.5596	0.7130	1.5692	0.0988
29	1.5602	0.6765		
30	1.5607	0.6430	1.5694	0.0891
31	1.5612	0.6122		
32	1.5616	0.5838	1.5695	0.0809
33	1.5620	0.5575		
34	1.5624	0.5331	1.5696	0.0738
35	1.5628	0.5104		
36	1.5631	0.4893	1.5697	0.0678
37	1.5634	0.4697		
38	1.5637	0.4513	1.5698	0.0625
39	1.5640	0.4340		
40	1.5642	0.4179	1.5699	0.0578
41	1.5645	0.4027		
42	1.5647	0.3884	1.5700	0.0538

```
43   1.5649   0.3750
44   1.5651   0.3623   1.5700   0.0501
45   1.5653   0.3503
46   1.5655   0.3389   1.5701   0.0469
47   1.5656   0.3282
48   1.5658   0.3180   1.5701   0.0440
49   1.5660   0.3083
-------------------------------------------------------------
Simpson: Took 130 intervals to converge error to 0.0099
Trapezoidal: Took 483 intervals to converge error to 0.0100
```

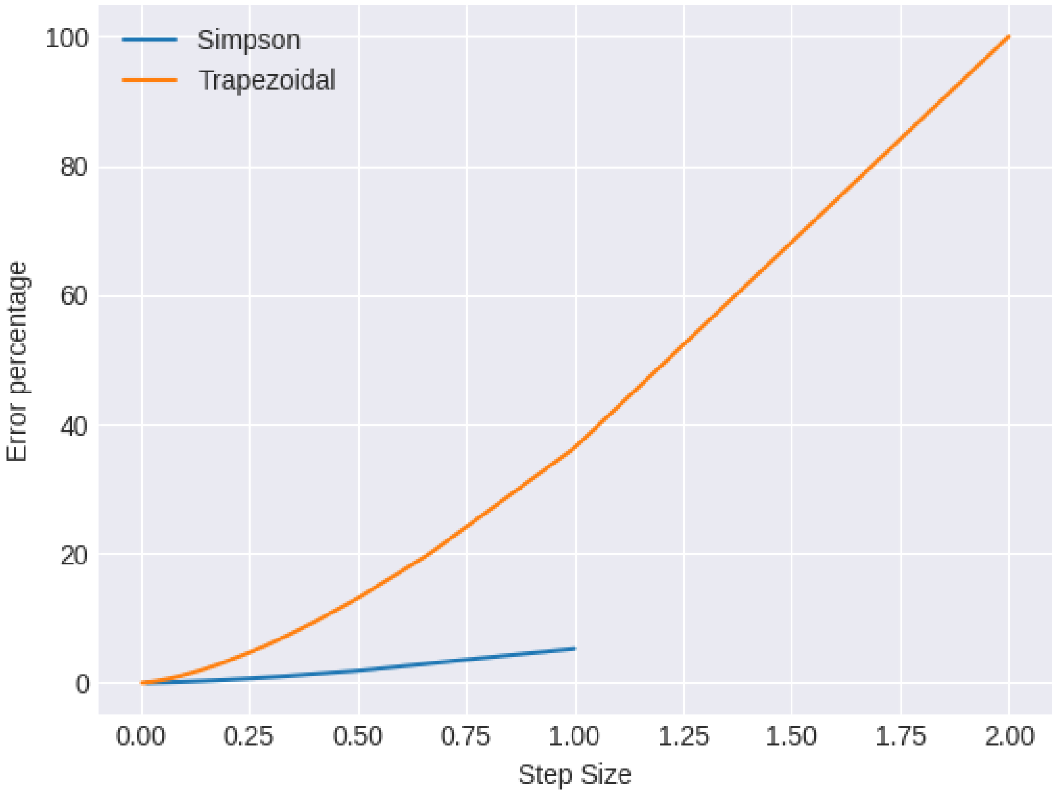

Figure 1: Step size vs Error

3.2 Function (b)

$$\int_0^2 \frac{1}{\sqrt{x}} dx$$

```python
def f2(x):
    # redifining function at x = 0
    if(x!=0):
        val = 1/(x**0.5)
    else:
        val=0
    return(val)
def integ_f2(a,b):
    fa = 2*(a**0.5)
    fb = 2*(b**0.5)
    return(fb-fa)
```

```
true value of integral: 2.8284271247461903
-----------------------------------------------------------
Interval , Value_tpz , Err tpz , Value_simp ,  error_simp
1    0.7071    75.0000
2    1.3536    52.1447    1.9383    31.4699
3    1.6295    42.3867
4    1.7921    36.6386    2.1991    22.2514
5    1.9025    32.7376
6    1.9837    29.8671    2.3146    18.1681
7    2.0466    27.6406
8    2.0973    25.8482    2.3834    15.7340
9    2.1393    24.3649
10    2.1747    23.1111    2.4304    14.0729
11    2.2052    22.0329
12    2.2318    21.0929    2.4651    12.8468
13    2.2553    20.2638
14    2.2762    19.5254    2.4920    11.8938
15    2.2949    18.8624
16    2.3119    18.2626    2.5137    11.1256
17    2.3273    17.7166
18    2.3415    17.2169    2.5317    10.4893
19    2.3545    16.7572
20    2.3665    16.3325    2.5470    9.9511
21    2.3776    15.9385
22    2.3880    15.5717    2.5601    9.4880
23    2.3977    15.2292
24    2.4068    14.9083    2.5715    9.0840
25    2.4153    14.6069
```

26	2.4233	14.3230	2.5816	8.7277
27	2.4309	14.0551		
28	2.4381	13.8017	2.5906	8.4102
29	2.4448	13.5615		
30	2.4513	13.3335	2.5986	8.1250
31	2.4574	13.1165		
32	2.4633	12.9099	2.6059	7.8670
33	2.4689	12.7127		
34	2.4742	12.5242	2.6126	7.6321
35	2.4793	12.3439		
36	2.4842	12.1712	2.6186	7.4171
37	2.4889	12.0056		
38	2.4934	11.8465	2.6242	7.2193
39	2.4977	11.6936		
40	2.5018	11.5464	2.6294	7.0365
41	2.5059	11.4047		
42	2.5097	11.2681	2.6342	6.8669
43	2.5134	11.1362		
44	2.5170	11.0089	2.6387	6.7090
45	2.5205	10.8859		
46	2.5239	10.7669	2.6428	6.5615
47	2.5272	10.6517		
48	2.5303	10.5401	2.6467	6.4234
49	2.5334	10.4320		

```
-----------------------------------------------------
Simpson: Took 33554432 intervals to converge error to 0.0077
Trapezoidal: Took 67108864 intervals to converge error to 0.0089
```

Since the convergence is very slow, Geometric step size growth is used in this case.

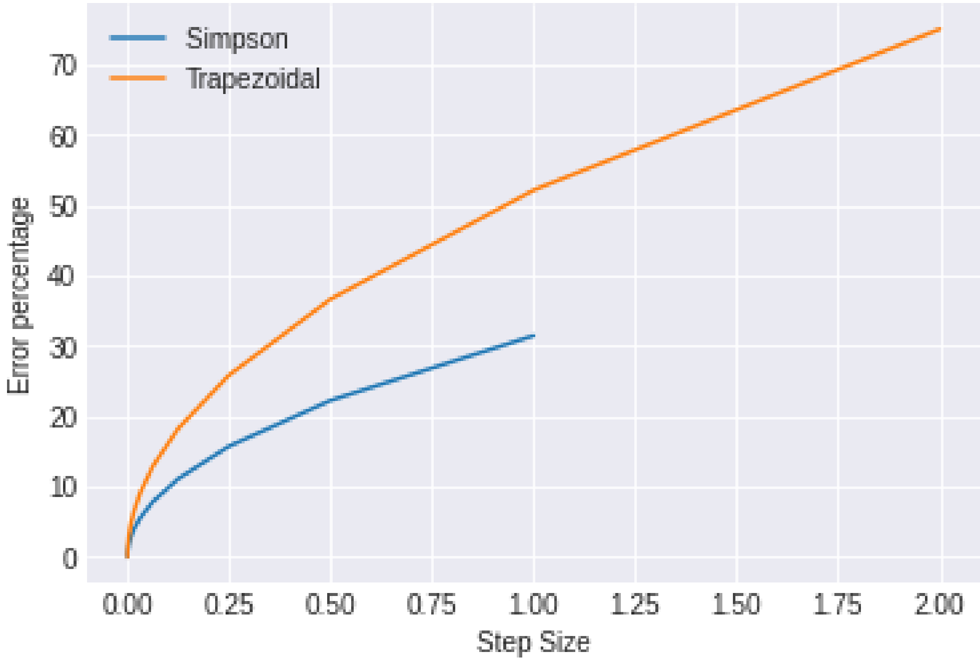

Figure 2: Step size vs Error : for simpson we have only even number of intervals

Better way to calculate this We see a singularity at point x=0 , due to which the convergence of both the method is extremely slow , hence we try to do some appropriate transform to get rid of this ssingularity.

$$x = t^2$$

$$d = 2tdt$$

$$\int_0^2 \frac{1}{\sqrt{x}}dx = \int_0^{\sqrt{2}} \frac{1}{t}(2t)dt = \int_0^{\sqrt{2}} 2dt.$$

```
def f2_p(x):
    return(2)
def integ_f2_p(a,b):
    return(2*(b-a))
```

```
--------------------------------------------------------
true value of integral: 2.8284271247461903
--------------------------------------------------------
Interval , Value_tpz , Err tpz , Value_simp ,  error_simp
1   2.8284   0.0000
2   2.8284   0.0000   2.8284   0.0000
3   2.8284   0.0000
4   2.8284   0.0000   2.8284   0.0000
---------------------------------------------------------
```

```
Simpson: Took 2 intervals to converge error to 0.0000
Trapezoidal: Took 1 intervals to converge error to 0.0000
```

Hence, by proper transform we can reduce the number of computation required drastically

3.3 Function (c)

$$\int_0^1 \int_0^2 (xy^2)dxdy$$

```python
def f3(y,):
    def f3_a(x):
        return (x)
    # using trapezoidal rule for inner function
    return (y**2)*integral(f3_a,0,2,2,'tpz')

def integ_f3(a,b):
    def integ_f3_x(xa,xb):
        fa=(xa**2)/2
        fb = (xb**2)/2
        return(fb-fa)
    return ((b**3)/3-(a**3))*integ_f3_x(0,2)
```

```
--------------------------------------------------
true value of integral: 0.6666666666666666
----------------------------------------------------
Interval , Value_tpz , Err tpz , Value_simp ,  error_simp
1   1.0000   50.0000
2   0.7500   12.5000   0.6667   0.0000
3   0.7037   5.5556
4   0.6875   3.1250   0.6667   0.0000
5   0.6800   2.0000
6   0.6759   1.3889   0.6667   0.0000
7   0.6735   1.0204
8   0.6719   0.7813   0.6667   0.0000
9   0.6708   0.6173
10   0.6700   0.5000   0.6667   0.0000
11   0.6694   0.4132
12   0.6690   0.3472   0.6667   0.0000
```

```
13   0.6686   0.2959
14   0.6684   0.2551   0.6667   0.0000
15   0.6681   0.2222
16   0.6680   0.1953   0.6667   0.0000
17   0.6678   0.1730
18   0.6677   0.1543   0.6667   0.0000
19   0.6676   0.1385
20   0.6675   0.1250   0.6667   0.0000
21   0.6674   0.1134
22   0.6674   0.1033   0.6667   0.0000
23   0.6673   0.0945
24   0.6672   0.0868   0.6667   0.0000
25   0.6672   0.0800
26   0.6672   0.0740   0.6667   0.0000
27   0.6671   0.0686
28   0.6671   0.0638   0.6667   0.0000
29   0.6671   0.0595
30   0.6670   0.0556   0.6667   0.0000
31   0.6670   0.0520
32   0.6670   0.0488   0.6667   0.0000
33   0.6670   0.0459
34   0.6670   0.0433   0.6667   0.0000
35   0.6669   0.0408
36   0.6669   0.0386   0.6667   0.0000
37   0.6669   0.0365
38   0.6669   0.0346   0.6667   0.0000
39   0.6669   0.0329
40   0.6669   0.0313   0.6667   0.0000
41   0.6669   0.0297
42   0.6669   0.0283   0.6667   0.0000
43   0.6668   0.0270
44   0.6668   0.0258   0.6667   0.0000
45   0.6668   0.0247
46   0.6668   0.0236   0.6667   0.0000
47   0.6668   0.0226
48   0.6668   0.0217   0.6667   0.0000
49   0.6668   0.0208
-------------------------------------------------------------
Simpson: Took 2 intervals to converge error to 0.0000
Trapezoidal: Took 71 intervals to converge error to 0.0099
-------------------------------------------------------------
```

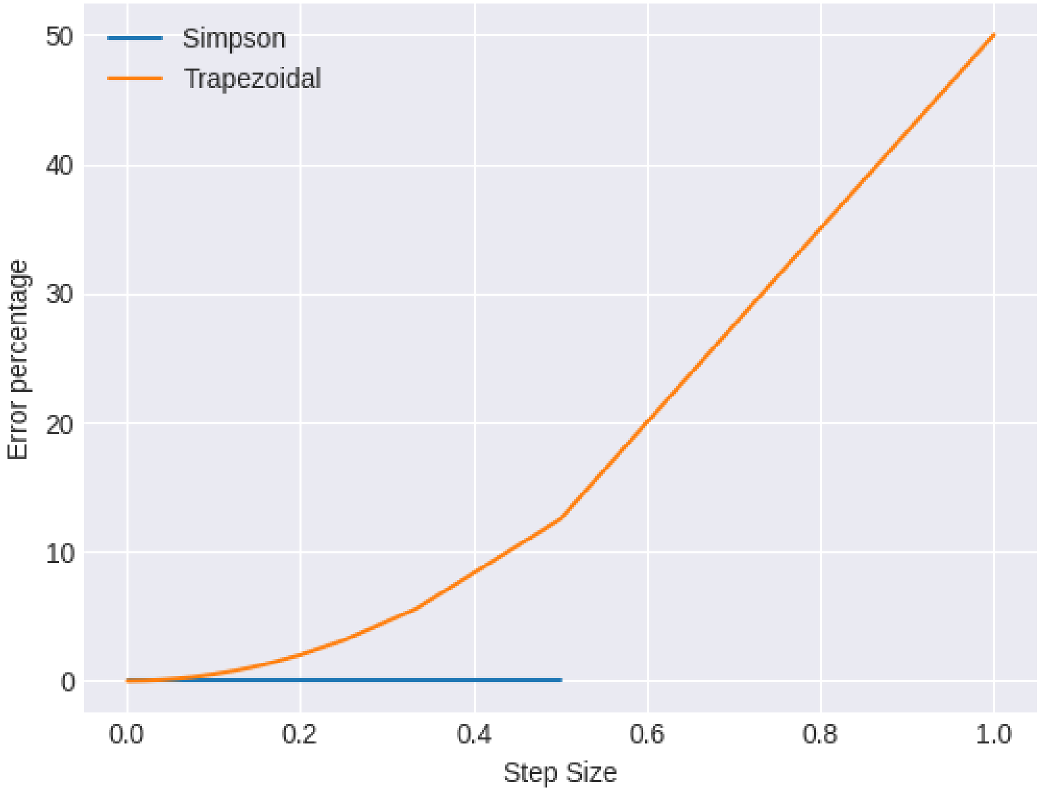

Figure 3: Step size vs Error