Math 105B Lab Report 8

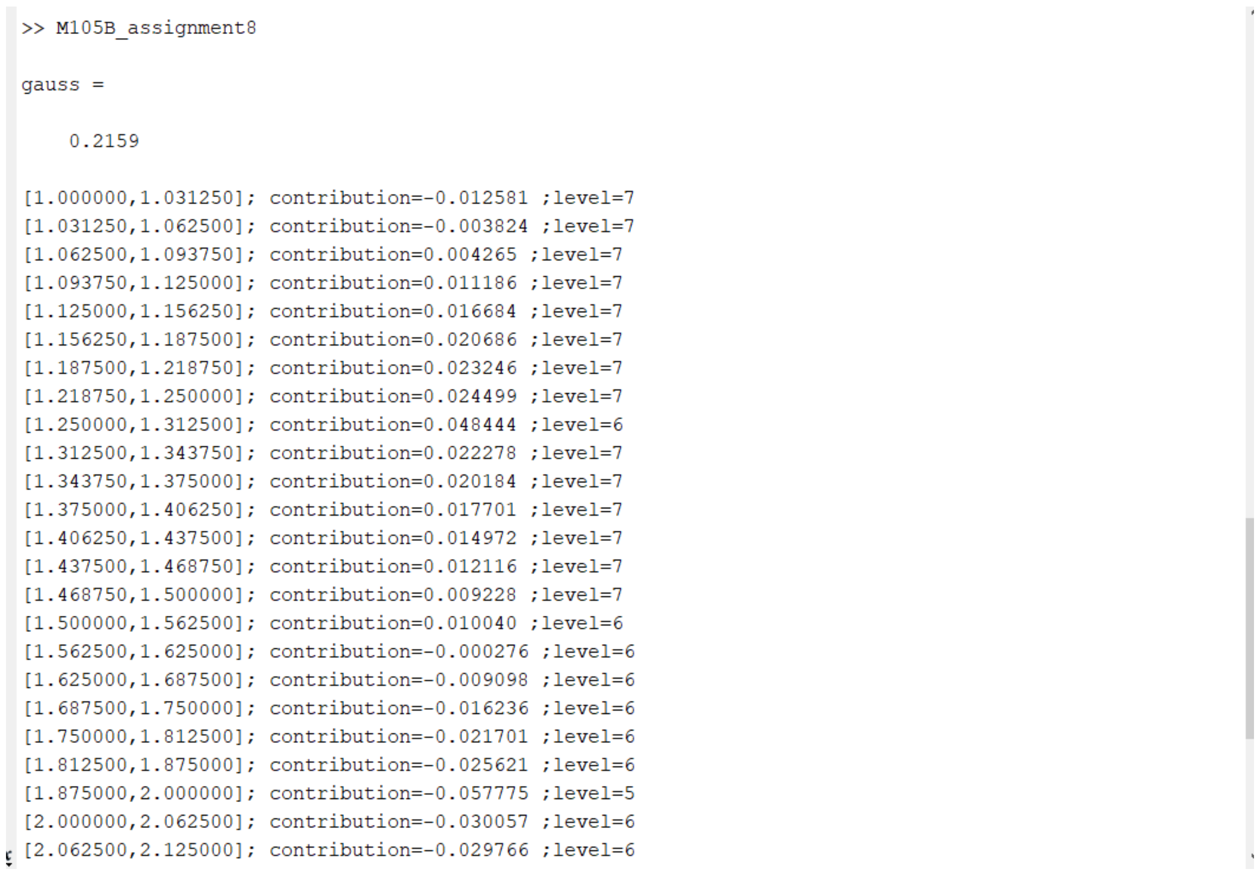
Jingyang Qiu ID:47674821

Purpose/Objective: In this lab, we will code and test an adaptive two-point Gaussian integration algorithm. The first step is to write a function that changes the variable so that the bounds are -1 and 1, given two endpoints a and b. We will then write a recursive function that repeatedly checks whether it satisfies the tolerance test, if not, it splits the interval into two subintervals and calls the function again. Last but not least, we will test this algorithm using a specific example.

Introduction: We are given an integral sin(10/x)/x over the interval [1,3]. The basic two-point Gauss function aims to use change of variables so we could use the formula g(-sqrt(3)/3)+g(sqrt(3)/3). The recursive function asks us to increase the level by 1 each time and check whether the tolerance condition is satisfied. Lastly, we are asked to use a maximum depth of 100 subintervals and the tolerance of 10^-7.

Algorithm: One thing I tried to do but didn't work was defining a new function handle in the basic gauss function. It would cause error later on in the recursive function. I also tried to follow the pseudocode given, but I found out that the input "sum" is not actually needed. So I changed a couple things accordingly, like defining s1 = Adaptive(f,a,c,tol,level,N) and s2 = Adaptive(f,c,b,tol,level,N), then adding them up.

Results:

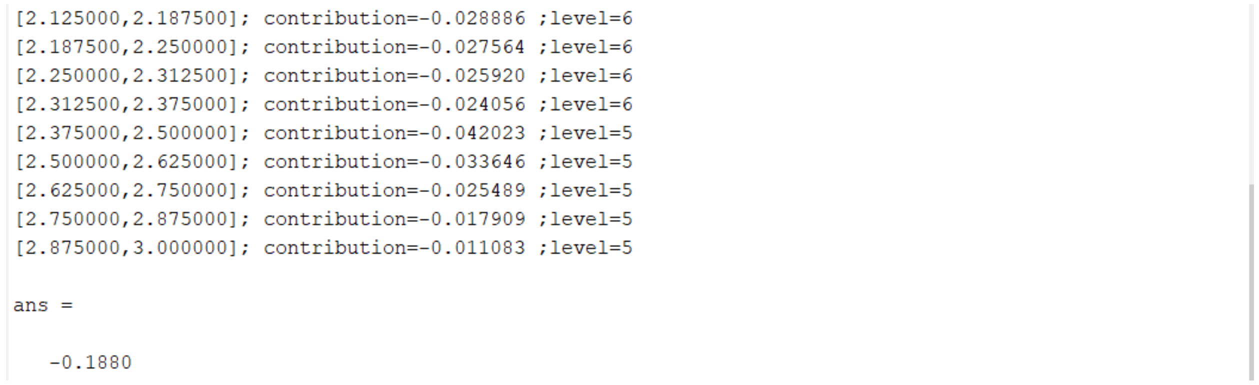


```
>> M105B_assignment8

gauss =

    0.2159

[1.000000,1.031250]; contribution=-0.012581 ;level=7
[1.031250,1.062500]; contribution=-0.003824 ;level=7
[1.062500,1.093750]; contribution=0.004265 ;level=7
[1.093750,1.125000]; contribution=0.011186 ;level=7
[1.125000,1.156250]; contribution=0.016684 ;level=7
[1.156250,1.187500]; contribution=0.020686 ;level=7
[1.187500,1.218750]; contribution=0.023246 ;level=7
[1.218750,1.250000]; contribution=0.024499 ;level=7
[1.250000,1.312500]; contribution=0.048444 ;level=6
[1.312500,1.343750]; contribution=0.022278 ;level=7
[1.343750,1.375000]; contribution=0.020184 ;level=7
[1.375000,1.406250]; contribution=0.017701 ;level=7
[1.406250,1.437500]; contribution=0.014972 ;level=7
[1.437500,1.468750]; contribution=0.012116 ;level=7
[1.468750,1.500000]; contribution=0.009228 ;level=7
[1.500000,1.562500]; contribution=0.010040 ;level=6
[1.562500,1.625000]; contribution=-0.000276 ;level=6
[1.625000,1.687500]; contribution=-0.009098 ;level=6
[1.687500,1.750000]; contribution=-0.016236 ;level=6
[1.750000,1.812500]; contribution=-0.021701 ;level=6
[1.812500,1.875000]; contribution=-0.025621 ;level=6
[1.875000,2.000000]; contribution=-0.057775 ;level=5
[2.000000,2.062500]; contribution=-0.030057 ;level=6
[2.062500,2.125000]; contribution=-0.029766 ;level=6
```



```
[2.125000,2.187500]; contribution=-0.028886 ;level=6
[2.187500,2.250000]; contribution=-0.027564 ;level=6
[2.250000,2.312500]; contribution=-0.025920 ;level=6
[2.312500,2.375000]; contribution=-0.024056 ;level=6
[2.375000,2.500000]; contribution=-0.042023 ;level=5
[2.500000,2.625000]; contribution=-0.033646 ;level=5
[2.625000,2.750000]; contribution=-0.025489 ;level=5
[2.750000,2.875000]; contribution=-0.017909 ;level=5
[2.875000,3.000000]; contribution=-0.011083 ;level=5

ans =

   -0.1880
```

Conclusion: We can see that our basic gauss function is very inaccurate, since the actual integral value is approximately -0.188. Our adaptive function ended up yielding very accurate results since the tolerance we chose was so small and 100 levels was definitely sufficient.