Graded Homework 4 Foundations of Computational Math 2 Spring 2025

Due date: 11:59PM on Monday, April 14, 2025

Written Exercises

There are no written exercises in this assignment.

Programming Exercise

In this assignment you will implement numerical quadrature methods and compare their observed behavior to theoretical predicitons.

The Codes: Consider the **composite quadrature methods** based on the following basic methods:

- The closed Newton-Cotes method that uses one point (the left endpoint), i.e., the Left Rectangle Rule.
- The closed Newton-Cotes method that uses two points (the two endpoints), i.e., the Trapezoidal Rule.
- The closed Newton-Cotes method that uses three points (the two endpoints and the midpoint), i.e., Simpson's First Rule.
- The open Newton-Cotes method that uses one point (the midpoint), i.e., the Midpoint Rule.
- The open Newton-Cotes method that uses two points (the points at 1/3 and 2/3 across the interval of integration).
- The two-point Gauss-Legendre method.

All of these basic methods are defined and discussed to various levels of detail. The composite forms of the Midpoint Rule, Trapezoidal Rule and Simpson's First Rule are derived along with the composite error in the notes. Error expressions for the basic version (not composite) of all are in the notes.

1.i. For each of these basic methods implement a composite quadrature method that uses a set of uniform subintervals of the interval of integration $[a, b]$. Your code should be as efficient as possible in terms of computational complexity, i.e., the number of evaluations of the function $f(x)$, and storage.

1.ii. For the Composite Midpoint Rule and the Composite Trapezoidal Rule implement an efficient adaptive step refinement code that uses a set of uniform subintervals of the interval of integration $[a, b]$ and global uniform subinterval size refinement with a factor $\alpha = 1/k$. For the Composite Trapezoidal Rule use $\alpha = 1/2$. For the Composite Midpoint Rule use $\alpha = 1/3$. **Your implementations should be as efficient as possible in terms of function evaluations. If possible, implement these two methods and choices of α with complete reuse.**

Some Integrals: The integrals used to empirically evaluate your codes should include

$$\int_0^3 e^x dx = e^3 - 1 \tag{1}$$

$$\int_0^{\frac{\pi}{3}} e^{\sin(2x)} \cos(2x) dx = \frac{1}{2}\left(-1 + e^{\frac{\sqrt{3}}{2}}\right) \tag{2}$$

$$\int_{-2}^1 \tanh(x) dx = \ln\left(\frac{\cosh(1)}{\cosh(2)}\right) \tag{3}$$

$$\int_0^{3.5} x \cos(2\pi x) dx = -\frac{1}{2\pi^2} \tag{4}$$

$$\int_{0.1}^{2.5} \left(x + \frac{1}{x}\right) dx = \frac{2.5^2 - 0.1^2}{2} + \ln(2.5/0.1) \tag{5}$$

Note all of these have symbolic solutions that may be used to assess true error, predict expected behavior and analyze observed behavior. You are encouraged to

use additional functions for validation and to demonstrate any trends you consider important.

The Tasks:

2.i. Derive the error expression for the Composite Left Rectangle Rule, the Composite Open Newton-Cotes two-point Method, and the Composite Gauss-Legendre two-point Method.

2.ii. Provide systematic evidence of the correct execution of your codes. This should use more problems than those listed above.

2.iii. Describe the implmentation and efficiency considerations of all of your imoplementations. Make sure that you include in the description of your codes sufficient information how your code is designed to exploit the efficient reuse of function evaluations to avoid redundant work given the α values required above. For example, how did it change the organization of the computation or data structure when efficient interval refinement is used or not used.

2.iv. For each integral in the problems list, estimate the subinterval size needed to satisfy various error demands using the error expressions for the composite quadrature methods. The relative error requires knowledge of the size of the integral. You may use your knowledge of the exact answer for each to make these predictions.

2.v. Run the codes for various error requirements, summarize appropriately and concisely your observations, compare the observed performance to predictions and explain them based on knowledge of the methods and the particular problems. Comment on any behavioral differences observed between the problems and explain them based on differences in the functions being integrated. Discuss and compare the number of function evaluations. Your discussions should include comparing for all of the composite methods, i.e., without adaptive refinement, the accuracy actually achieved using the true error computed from your analytical solutions to the integrals versus the computational complexity. Identify any trends observed.

2.vi. You should include the use the composite error expressions to select a subinterval size, H_m, that is expected to achieve a specified error. Are the

subinterval sizes used based on your a priori analysis for the composite methods conservative, i.e., do you get greater accuracy than you expect?

2.vii. Repeat the experiments and explanations for the composite codes with adaptive refinement using the α value given above for each. Compare the results with the methods above that used a subinterval size chosen by a priori error analysis. Specifically, how did the accuracy per function evaluation ratio behave for the two sets of methods.

Other Test Problems and Intermediate Solutions

As with earlier assignments, you are encouraged to submit intermediate partial results or early versions of your results. You may discuss those with Yue if you wish. Independently of early or partial submissions, after you have submitted you solutions you should make an appointment with the Yue. She will ask you to demonstrate your code on some test problems as a final evaluation of your codes correctness. You should also be prepared to explain the design and operation of your code to her.