

Homework Assignment 6 Numerical Integration

Due: Friday, October 26, 2006 on the hour before class

IMPORTANT NOTE ON THE USE OF T-SQUARE

We all experienced significant problems with t-square the last two times assignments were due. The support staff has identified that the problems originated with the tool "Assignments Auto-Submission". As a result, they have disabled the auto-submission feature. It will therefore be *your responsibility* to submit. So, as previously, you can upload your zip-file using the save as draft, but you will have to go back and click the final "Submit" button otherwise your assignment will not be submitted. Remember, though, once you hit the "Submit" button, you can not go back and upload an updated version. Because of the large number of students in ME2016, we cannot accept revised versions after you have submitted.

Description and Outcomes

In this assignment, you will implement and test two numerical integration algorithms: Trapezoidal and Simpson 1/3 integration. After implementing the two integration algorithms you will compare their performance to the built-in Matlab function `quad` ().

The learning objectives of this assignment are:

- to increase your understanding of numerical integration methods
- to learn about convergence rates and round-off errors
- to learn how to use numerical integration algorithms for solving engineering problems
- to learn how to create numerically efficient programs in Matlab
- to learn how to compare and interpret the performance of different numerical integration algorithms

Background

In this assignment, the focus is on the implementation of several numerical integration methods. The goal is to implement these algorithms efficiently in Matlab and to compare their performance – that is: How does the absolute error change as a function of the number of segments? You need to implement the Trapezoidal and Simpson 1/3 integration methods in such a way that they are both fast and flexible to use. Program execution speed is crucial in this assignment because we will be using the algorithms many times and for different numbers of segments (up to 10^6 segments for a single integral).

Tasks

Task 1: Create the functions `trapezoidal_integration_XYZ` and `simpson_integration_XYZ` (where XYZ are your initials)

To make it easier to perform the integration for different numbers of segments, start by implementing the trapezoidal and Simpson 1/3 integration rules as separate functions. Each function should have the following inputs and outputs:

Inputs:

- 1) a function handle, *fhandle*
- 2) the lower integration bound, *a*
- 3) the upper integration bound, *b*
- 4) the number of segments, *n*

Outputs:

1) the integral, *integral*

Be careful when implementing the “multiple-application” version of the Simpson Rule: The number of segments should be a multiple of 2 for Simpson 1/3. If the user provides a number of segments that does not fit this requirement, you can end the integration scheme with one application of the Simpson 3/8 rule as described in example 21.6 of Chapra and Canale (pg. 602).

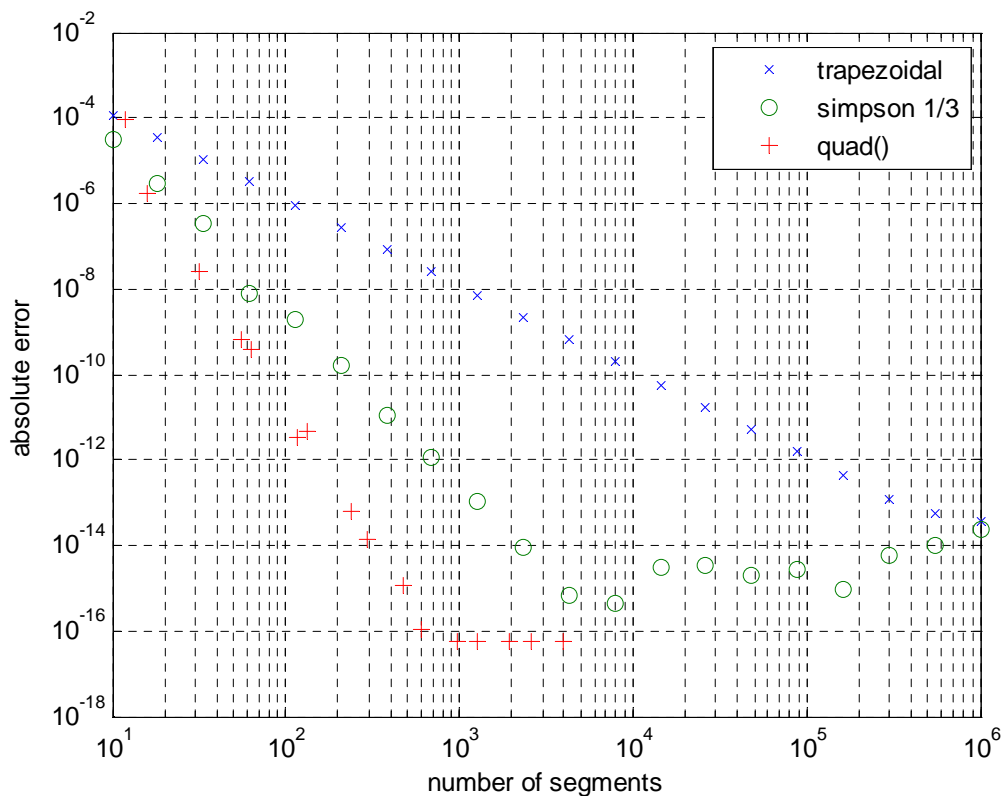
Task 2: Write a script `integration_accuracy_XYZ` (where `XYZ` are your initials) for comparing the accuracy of your integration functions

The script should solve for the integral using three methods: 1) the trapezoidal rule, 2) Simpson’s 1/3 rule, and the Matlab function `quad()`. You should use as a test case the following integral (It is best to implement this as an anonymous function as you did in HW4):

$$I = \int_0^2 \cos(x) \cos(2x) dx$$

In order to compare the performance of the different integration algorithms, you should run all three of them with different values for the number of integration segments. This will allow you to interpret the convergence rate of the algorithms. To help you make this interpretation, you should generate a figure that plots the true error as a function of the number of segments on a log-log scale. Since the purpose is to *test* our implementations, we have chosen an integral for which we know the solution – this will allow you to compute the absolute error exactly:

$$I = \frac{\sin(2)}{2} + \frac{\sin(6)}{6}$$



Hints:

- A log-log scale means that both the number of segments and the error are represented on a logarithmic scale; Matlab has a special plot function for this purpose: `loglog`.
- To make the tendencies in the plot clear, it is also important that you choose the number of segments in such a way that they show up in an equally spaced fashion on a logarithmic scale. Investigate the Matlab function `logspace` – compare to `linspace`. A good range of values for the number of segments would go from 10 through 10^6 in 20 steps.
- It is absolutely critical that you implement your functions in a computationally efficient fashion. Due to the large number of segments that you are considering, your script may take a very long time to run if you don't implement it efficiently (As a reference, my script takes about 5 seconds on a 2GHz Pentium 4). The following Matlab constructs are rather inefficient and should be avoided to the extent possible:
 - for loops
 - function calls (each time Matlab calls a function, it needs to copy all the input variables into a new workspace – this can take a long time, especially if you execute it 10^6 times...)

You can avoid these constructs by taking advantage of Matlab's vector operators.

- You can find more information about the Matlab function `quad` at the following URL: <http://www.mathworks.com/access/helpdesk/help/techdoc/ref/quad.html>

Note that this function does not take the number of segments as an input but uses the desired accuracy instead. I suggest using desired accuracies between 10^{-1} and 10^{-16} (using `logspace` is best). After computing the integral, you should again compute the true absolute error rather than relying on the error being exactly equal to the desired accuracy.

Task 3: Interpret the graph

Question 1: Describe the most important characteristics of the graph.

Question 2: What is the order of convergence of all three integration methods? (Note: use curve fitting to determine the order of convergence)

Question 3: Does the order of convergence match the theoretical results that we derived in class? Show your work.

Question 4: Why does the error for the Simpson rule seem to increase for large numbers of segments?

Question 5: Which method would you use and how many segments would you select? Justify your answers.

Report

Turn in a brief report in MS Word containing the following:

1. A copy of all your Matlab code.
2. A copy of all your figures you generated.
3. Your answers to all the questions in task 3.
4. A statement of collaboration (see below) – include this even if you did not collaborate with anybody

Evaluation Criteria

A detailed grade sheet for this assignment is provided as a separate file on the course web-site. Print out this sheet, fill in your name, and staple it to the front of your submission.

Submission

1. **Hardcopy in class:** At the *beginning* of class, hand in a hardcopy that includes the grade sheet and your report. Make sure you staple everything together so that we don't lose anything. (no staple = incorrect submission)
2. **T-square:** save all your files in a zip-file called HW6.FamilyName.FirstName.zip (replace FamilyName and FirstName with YOUR names) and submit this file in the Assignments section of T-square. The zip-file should contain: your Matlab functions and script, and your report. The deadline for electronic submission is ***on the hour before the beginning of class***. (For instance, if your class starts at 2:05PM, then the assignment is due at 2:00PM).

NOTE: it is best to use the "Save Draft" feature on t-square. This will allow you to resubmit as many times as you want anytime before the deadline. At the deadline, t-square will automatically submit the current draft — no action is required on your behalf.

Late Submission

Remember that the deadline for this assignment will be strictly enforced. After the deadline has passed you will receive a late-penalty. Don't wait until the last minute to get started! We repeat here the policy that is included in the syllabus:

Late Submission

T-square will be set up such that late submissions cannot be submitted electronically. We will accept late homework but with the following penalties. Submit by noon Saturday and receive 20% off of your grade. Submit by noon Sunday and receive 40% off. After noon on Sunday, no more partial credit will be awarded. If you are submitting late, e-mail the electronic version and bring a hardcopy to the office of your instructor (Section A: E-Jiang.Ding@ipst.gatech.edu, IPST 378; Sections C&D: chris.paredis@gatech.edu, MARC 256). If you plan on submitting late, a quick e-mail by the deadline would be appreciated so that we can make appropriate plans for grading your assignment.

Collaboration

We would like to re-emphasize the policy on collaboration. Collaboration is encouraged. Discussing the assignments with your peers will help you to develop a deeper understanding of the material. However, "discussing the assignment" does not mean solving it together; it does not mean asking your friend to debug your code for you. I encourage you to discuss how to approach the problem, which Matlab functions to use, or how to interpret the results, but I do expect each student to turn in a report and Matlab functions that reflect the student's individual work. Do not copy code from another student. Do not copy parts of other electronic documents. In general, an activity is acceptable if it promotes learning by you and your peers. For example, you learn from discussing alternate solution approaches with your friend, but you don't learn from blindly copying your friend's code. To avoid any confusion, each homework solution should explicitly identify the students with whom you collaborated and what the extent of the collaboration was. Any copying on homework and/or exams will be dealt with severely and reported to the Dean of Students – No exceptions. If you have questions about this collaboration policy, do not hesitate to ask your instructor.