Homework 7: Numerical Integration (Quadrature)

Assigned: Wed Nov 16. Due: Wed Nov 30

This homework asks you to make changes to Moler's program *quadtx* and the recursive function that it calls, *quadtxstep* to make it even more accurate. (It may be useful to call *quadgui* to demo *quadtx*.) At present, *quadtxstep* has the following recursive specification:

- compute Simpson's Rule on $[a, b]$ and assign the value to variable $Q1$ (denoted S in Section 6.2 of Moler's text)
- compute Simpson's Rule on the left and right halves of $[a, b]$ and assign the sum of the two values to variable $Q2$ (denoted S_2 in Section 6.2 of Moler's text, or $S_L + S_R$ in my notes)
- if $|Q1 - Q2|$ is less than the desired tolerance, terminate, returning the extrapolated value $Q = Q2 + \frac{Q2-Q1}{15} = \frac{16Q2-Q1}{15}$ (this is sometimes known as Weddle's rule and is more accurate than either $Q1$ or $Q2$)
- otherwise, make recursive calls on the left and right halves of the interval and return the sum of the two approximate integrals

In addition to the approximate integral (the first output argument), *quadtxstep* also computes the number of function evaluations that were made (the second output argument).

Modify a copy of *quadtx* and *quadtxstep* so *quadtxstep* does the following:

- compute Weddle's Rule on $[a, b]$ and assign the value to variable $W1$
- compute Weddle's Rule on the left and right halves of $[a, b]$ and assign the sum of the two values to variable $W2$
- if $|W1 - W2|$ is less than the desired tolerance, terminate, returning a *better extrapolated value* than either $W1$ or $W2$ (see below)
- otherwise, make recursive calls on the left and right halves of the interval and return the sum of the two approximate integrals

The new code should also be updated to correctly return the number of function evaluations that were made.

Before starting to make the changes, work through questions 1-3. The code modification begins in question 4.

1. For what degree polynomials does the *unmodified* program *quadtx* quit immediately after the original 5 function evaluations? Check this experimentally by calling it to integrate the polynomial x^n from 0 to 1 for $n = 2, 3, 4, 5$, using the default tolerance *tol* (10^{-6}). Explain why this happens.

2. For the next higher value of n, *quadtx* takes many more function evaluations. Nonetheless, *no matter what you choose for the tolerance*, for example 10^{-3}, 10^{-6}, 10^{-9}, it gives the *exactly correct answer* for the first output argument, with no discretization error (there will usually be a rounding error around the size of the machine precision). Why is this, and why didn't it quit much earlier?

3. How big do you have to make n before the accuracy *does* depend on the tolerance? Why?

4. What are the formulas for $W1$ and $W2$ that you need for the modified code? Note that $W1$ involves 5 function values on $[a, b]$ and $W2$ involves 5 function values on each of the left and right halves of $[a, b]$, making 9 altogether as the middle one is in both. There is more than one correct way to program this; choose one that makes sense to you and put comments in the code to explain it. However, make sure you don't evaluate the function twice at the same point. For now just return $W2$ when $|W1 - W2|$ is less than the tolerance, instead of the better extrapolated value.

5. For what degree polynomials does the modified code quit immediately after the original 9 function evaluations? The answer should be higher than the answer to question 1. If it is the same or lower, there is a bug in your modified code.

6. How many function values are required by the new code and the old code respectively to integrate the *humps* function (which is built into Matlab) from 0 to 1 with a tolerance of 10^{-12}? The new code should require substantially fewer function evaluations than the original code. If not, there is a bug in your modified code.

7. What are the correct weights to use for the *better extrapolated value*, to make the error terms in $W1$ and $W2$ cancel? Return this instead of $W2$ when $|W1 - W2|$ is less than the tolerance.

8. If you were able to get the code for the previous question working, answer the second and third question for the modified code.