Name: _____

CSC 427/627, MAT 423/623 Programming Project

Chapter 5 – Numerical Integration

Due Wednesday, May 6, 2020

You may write all code in the programming language of your choice. Please save all code in a single folder, compress it, and email it to me. You also need to turn in a printout of the results and all commentary.

The goal of this project is to implement the various quadrature methods introduced in chapter 5 for approximating the definite integral $\int_a^b f(x)dx$ for a given continuous function f defined over the interval $[a, b]$. We will implement the various composite methods as well as the Gaussian quadrature method.

First write functions that implement the six methods specified in Section 5 of this project. Pay close attention to the *Programming Hints* in the textbook. Each function will input a function, endpoints, and an integer n that dictates the accuracy of the method. Each function will output the real number corresponding to the approximation of the definite integral. Errors will be calculated as a means to compare the various methods.

1 Experiment: Estimating the Degree of Precision for the Various Methods

We check to see if the various methods are "exact" when approximating $\int_a^b x^p dx$ for various powers p. Fill in the following chart indicating whether the method was exact or not for the various methods and bounds $[a, b]$. We consider two different sets of bounds to account for even and odd properties of the polynomials that can lead to higher accuracy due to cancellations.

1. $a = -1$, $b = 1$. Then $\int_a^b x^p dx = \frac{(-1)^p+1}{p+1}$. Place a check-mark in each cell where the output for the given approximation method is essentially correct (up to floating-point errors).

	Method							
p	$R_4(x^p)$	$M_4(x^p)$	$T_4(x^p)$	$T_4^C(x^p)$	$S_4(x^p)$	$G_1(x^p)$	$G_2(x^p)$	$G_4(x^p)$
0	2	2	2	2	2	2	2	2
1	0.5	0	0	0	0	0	0	0
2	0.75	0.625	0.75	0.6667	0.6667	0	0.6667	0.6667
3	0.5	0	0	0	0	0	0	0
4	0.5625	0.3203	0.5625	0.4479	0.4167	0	0.2222	0.4
5	0.5	0	0	0	0	0	0	0
6	0.5156	0.1782	0.5156	0.3932	0.3542	0	0.0741	0.2857
7	0.5	0	0	0	0	0	0	0
8	0.5039	0.1001	0.5039	0.3796	0.3385	0	0.0247	0.2106
9	0.5	0	0	0	0	0	0	0

2. $a = 0$, $b = 1$. Then $\int_a^b x^p dx = \frac{1}{p+1}$. Place a check-mark in each cell where the output for the given approximation method is essentially correct (up to floating-point errors).

	Method							
p	$R_4(x^p)$	$M_4(x^p)$	$T_4(x^p)$	$T_4^C(x^p)$	$S_4(x^p)$	$G_1(x^p)$	$G_2(x^p)$	$G_4(x^p)$
0	1	1	1	1	1	1	1	1
1	0.625	0.5	0.5	0.5	0.5	0.5	0.5	0.5
2	0.4688	0.3281	0.3438	0.3333	0.333	0.25	0.333	0.333
3	0.3906	0.2422	0.2656	0.25	0.25	0.125	0.25	0.25
4	0.3457	0.1897	0.2207	0.2015	0.2005	0.0625	0.1944	0.2
5	0.3174	0.1539	0.1924	0.1704	0.168	0.0312	0.1528	0.1667
6	0.2985	0.1278	0.1735	0.1493	0.1453	0.0156	0.1204	0.1429
7	0.2853	0.1077	0.1603	0.1345	0.1292	0.0078	0.0949	0.125
8	0.276	0.0918	0.151	0.1238	0.1174	0.0039	0.0748	0.1111
9	0.2693	0.0788	0.1143	0.1161	0.1087	0.002	0.0590	0.1

3. Based on your results for parts 1 and 2, fill in the following chart with the value of N representing the degree of precision of the specified method using the definition found on page 275.

	Method							
	$R_4(x^p)$	$M_4(x^p)$	$T_4(x^p)$	$T_4^C(x^p)$	$S_4(x^p)$	$G_1(x^p)$	$G_2(x^p)$	$G_4(x^p)$
N	0	1	1	`1	1	2	3	9

2 Experiment: Rates of Convergence for the various Composite Methods when Integrating a Smooth Function

We approximate the definite integral

$$\int_0^{\pi} x^2 \cos(x) dx = -2\pi.$$

For each method R_n, M_n, T_n, T_n^C, and S_n, fill out the following table:

n	h	Error: $E_h = \lvert I_n(f) - I(f) \rvert$	Rate $= \ln(\lvert E_{2h}/E_h \rvert)/\ln(2)$
4	0.7854		N/A
8	0.3927		
16	0.1963		
32	0.0982		
64	0.0491		
128	0.0245		
256	0.0119		
512	0.0061		

3 Experiment: Rates of Convergence for the various Composite Methods when Integrating a Non-Smooth Function

We approximate the definite integral

$$\int_{-1}^{1} \sqrt{1-x^2}dx = \frac{\pi}{2}.$$

For each method R_n, M_n, T_n, T_n^C, and S_n, fill out the following table:

n	h	Error: $E_h = \lvert I_n(f) - I(f)\rvert$	Rate $= \ln(\lvert E_{2h}/E_h\rvert)/\ln(2)$
4	0.5		N/A
8	0.25		
16	0.125		
32	0.0625		
64	0.03125		
128	0.015625		
256	0.0078125		
512	0.00390625		

4 Experiment: Comparing Accuracy for Gauss Quadrature Methods

We now compare the results of the Gauss Quadrature Methods compared to the various composite methods. For each value of n, approximate the integral using the appropriate Gauss quadrature method and record which method was the most accurate (R_n, M_n, T_n, T_n^C, S_n, or G_n). Note that for $n = 1$, T_1^C and S_1 are not defined. For $n \geqslant 4$, the errors have already been computed in parts 2 and 3 for the composite methods.

1. $\int_0^{\pi} x^2 \cos(x)dx = -2\pi.$

n	$G_n(f)$	Error: $E_n = \lvert G_n(f) - I(f)\rvert$	Optimal Method
1			
2			
4			
8			
16			

2. $\int_{-1}^{1} \sqrt{1-x^2}dx = \frac{\pi}{2}$.

n	$G_n(f)$	Error: $E_n = \lvert G_n(f) - I(f)\rvert$	Optimal Method
1			
2			
4			
8			
16			

5 Quadrature Methods

In all of the methods we assume the function f is defined externally. You can input the function f using function handles if allowed in your programming language. Otherwise, you can input an integer that specifies which function you wish to use for the various parts of the project. Do not hard-code the function f inside a method.

For the external function f:
Input: x, (optional – integer specifying which function using cases).
Output: $f(x)$.

Implement the following methods as separate functions. Each function should input f, n, a, and b and return the corresponding approximation to $\int_a^b f(x)dx$. We assume all of the composite methods utilize a uniformly spaced mesh/grid defined by $x_i = a + ih$ for all $i = 0, 1, \ldots, n$, where $h = \frac{b-a}{n}$.

1. **Right-Endpoint Method**:

$$R_n(f) = h\left[f(x_1) + f(x_2) + \cdots f(x_n)\right]$$

2. **Midpoint Method**:

$$M_n(f) = h\left[f\left(\frac{x_0+x_1}{2}\right) + f\left(\frac{x_1+x_2}{2}\right) + \cdots + f\left(\frac{x_{n-1}+x_n}{2}\right)\right]$$

3. **Trapezoidal Method**:

$$T_n(f) = \frac{h}{2}\left[f(x_0) + 2f(x_1) + 2f(x_2) + \cdots + 2f(x_{n-1}) + f(x_n)\right]$$

See Algorithm 2.5 for pseudocode for implementing this method.

4. **Corrected Trapezoidal Method**:

$$T_n^C(f) = T_n(f) - \frac{h}{24}\left[3f(x_n) - 4f(x_{n-1}) + f(x_{n-2}) + 3f(x_0) - 4f(x_1) + f(x_2)\right]$$

This code will first call the trapezoidal method and then add some correction terms. Instead of using the exact value for the derivatives at the endpoints, a high-order derivative approximation is used. The code should print an error statement if $n < 2$.

5. **Simpson's Method**:

$$S_n(f) = \frac{h}{3}\left[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \cdots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)\right]$$

See Algorithm 5.1 for pseudocode for implementing this method. The code should print an error statement if n is not even or $n < 2$.

6. **Gaussian Quadrature**:

$$G_n(f) = \sum_{i=1}^{n} \omega_i^{(n)} F\left(z_i^{(n)}\right)$$

for

$$F(z) = \frac{1}{2}(b-a) f\left(a + \frac{b-a}{2}(z+1)\right),$$

where $-1 \leqslant z \leqslant 1$. The values of $\omega_i^{(n)}$ and $z_i^{(n)}$ for various values of n can be found in Table 5.5 on page 289 of the textbook. The code should include either a switch statement or several if-else statements to decide which values for ω_i and z_i to use given the input n. The code should print an error statement if $n \neq 1, 2, 4, 8, 16$.