Programming Assignment 04

Composite Simpson's Rule

Nick Handelman
2/23/2018

1 Project Description and Results

This program (Appendices A and B) implements the Composite Simpson's Rule (1). The implementation is an approximation since the error term is not included.

$$\int_a^b f(x)dx = \frac{h}{3}[f(a) + 2\sum_{j=1}^{n/2-1} f(x_{2j}) + 4\sum_{j=1}^{n/2} f(x_{2j-1}) + f(b)] - \frac{(b-a)h^4}{180} f^{(4)}(\mu), h = \frac{b-a}{n} \quad (1)$$

In the program, I combined the two sums into one for loop and added an extra term to account for the second sum's last term (j=n/2). (2) is a close representation of the approximation as implemented.

$$\int_a^b f(x)dx \approx \frac{h}{3}[f(a) + \sum_{j=1}^{n/2-1} [2f(x_{2j}) + 4f(x_{2j-1})] + 4f(x_{n-1}) + f(b)], h = \frac{b-a}{n} \quad (2)$$

The program ensures that n is a valid value (positive and even) before running the computation. I tested it on $f(x) = e^x$ with $a = -1$, $b = 1$ and several values of n. The outputs are given in Table 1. The exact answer is $e - \frac{1}{e}$ which is, to 10 decimal places, very approximately 2.3504023872 [1].

n	2	4	6	8	10	20
output	2.3620537565	2.3511948318	2.3505614868	2.3504530172	2.3504231806	2.3504036915
n	50	100	300	362		
output	2.3504024207	2.3504023893	2.3504023873	2.3504023872		

Table 1: Output Values

My implementation of the Composite Simpson's Rule requires $n = 362$ to approximate $\int_{-1}^{1} e^x dx$ to the tenth decimal place, as compared to [1]'s approximation.

References

[1] https://www.wolframalpha.com/input/?i=e-1%2Fe

Appendix A proj4.m

```
func = @(x)e^x;
a = −1;
b = 1;
n = 6;
approx_integral = simpson(func, a, b, n);

if(isnan(approx_integral))
    printf('Invalid Input. n must be positive and even.\n');
else
    fprintf('Integral(%i, %i) of %s = %.15d. Approximated with %i intervals.\n', a, b, sub
endif
```

Appendix B simpson.m

```
%composite simpson's rule
%integral(a,b)f(x)dx = h/3[f(a) + 2sum(n/2−1, j=1)f(xsub(2j)) + 4sum(n/2, j=1)f(xsub(2j−1)
%func: function being integrated
%a,b: lower, upper limits of integration
%n: number of intervals, must be even and positive
function approx_integral = simpson(func, a, b, n)
if(n<=0 || mod(n,2) == 1)
    approx_integral = NaN;
    return;
endif

h = (b−a)/n;
sum_of_evens = 0;
sum_of_odds = 0;

%sum of evens and sum of odds
for j=1:n/2−1
    x = a + 2*j*h;
    sum_of_evens = sum_of_evens + func(x); %sum(n/2−1, j=1)f(xsub(2j))
    sum_of_odds = sum_of_odds + func(x−h); %sum(n/2, j=1)f(xsub(2j−1))
endfor

%add last term of sum of odds
sum_of_odds = sum_of_odds + func(a + (n−1)*h); %f(xsub(2(n/2)−1))

approx_integral = h/3*(func(a) + 2*sum_of_evens + 4*sum_of_odds + func(b));

endfunction
```