The “Background” section of the assignment notes uses descriptions like “monotonically increasing”, “steady rate”, and “wildly varying” to describe functions. Since it may be unrealistic to expect the students to identify that the unknown function is piecewise and composed of different polynomials, it would be nice if they included the aforementioned descriptions in their analysis. At the very least, it should be recognized that the portion of the function in the [1, 3] range is linear. Thus, a 1st or 2nd order interpolant is able to perfectly approximate it, yielding an error-free result, as is the case with both the trapezoid and Simpson’s rules. Furthermore, it appears that portions of the functions with steep derivatives need to be broken down into many intervals. An exception is the range [5, 7], for which the Simpson’s rule does not need additional intervals (this may clue students that it is indeed parabolic, since the trapezoid rule does).

Also, since we have to actually break down an interval into two subintervals to perform our check for the stopping criteria, the algorithm simply returns those two intervals when it stops. This is why regions of the function like [1, 3] are split into two. Theoretically, only 1 interval is needed there, but we would have to turn our stopping criteria check into an elseif statement:

if s == sLeft + sRight

...

elseif abs(s - sLeft - sRight) < 15\*tol

...

else

...

end

However, floating point comparisons “s == sLeft + sRight” are to be avoided at all costs. So, we cannot include this first condition, but it is unnecessary since the correction term

goes to 0 if the above condition is satisfied.

Although we don’t know the result of the actual estimate, it seems the output of the MATLAB integral function and the output of the adaptive Simpson’s algorithm match quite closely. The adaptive trapezoid rule seems to deviate, however. Values obtained with tol = 1e-3:

Adaptive trapezoid: 35.197324924757694  
Adaptive Simpson’s: 35.197222225391329  
MATLAB integral: 35.197224470392428

Regarding computational load, we cannot count multiplications and divisions directly because the function is unknown. Since the Simpson’s rule requires 2 additional multiplications, 1 additional division, and 1 additional call to our unknown function, we know that the computation for each interval is more time-consuming (note: my current implementation of “f.m” re-creates the unknown function on each call in an efficient manner. Even so, this requires many more operations than computing the function directly, so the additional call to the unknown function is the only significant factor here). An analysis with the MATLAB tic and toc functions can be performed as follows:

fHandle = @(x) f(x);

a = 1;

b = 9;

tic

for ii = 1:100

% s = integralSimpsons(fHandle, a, b);

s = integralTrap(fHandle, a, b);

end

avg = toc/100

The time it takes to execute 100 iterations of a for loop is negligible; our average over 100 iterations is accurate to 5 decimal places. To further improve accuracy, store the result of avg in an array, perform the test numerous times, and take the mean of this array. Integrating the unknown function with either rule takes (on my machine):

Trapezoid: 5.6236e-04 sec  
Simpson’s: 7.7657e-04 sec

Using tol = 1e-2, we obtain 274 intervals for the adaptive trapezoid rule and 24 intervals for the adaptive Simpson’s rule. The overall time spent in each function is:

Trapezoid: 0.1541 sec  
Simpson’s: 0.0186 sec

Although the Simpson’s rule requires more operations, the adaptive algorithm that utilizes it results in far fewer intervals. Therefore, it is less computationally demanding than the adaptive trapezoid rule and it also more accurate.