Assignment 3

June Knauth | CMPT360 | 20210921

In this assignment, I developed a pseudocode to evaluate the Adaptive Simpson's Rule.

I used the following resources:

https://en.wikipedia.org/wiki/Simpson's_rule

https://web.stanford.edu/group/sisl/k12/optimization/MO-unit4-pdfs/4.2simpsonintegrals.pdf

https://github.com/scijs/integrate-adaptive-simpson/blob/master/index.js

https://www.math.utk.edu/~ccollins/refs/Handouts/rich.pdf

https://en.wikipedia.org/wiki/Adaptive_Simpson's_method

Here is the final pseudocode, adapted from the above:

```
# Evaluates Simpson's Rule for a given interval. Returns midpoint and function values.
func eval_simpson(f, a, fa, b, fb):
    midpt = (a+b) / 2
    fmidpt = f(midpt)
    return (midpt, fmidpt, abs(b-a) / 6 * (fa + 4*fmidpt + fb))

# The recursive function for Adaptive Simpson's
func rec_adsimp(f, a, fa, b, fb, err_tol, whole, midpt, fmidpt)
    # Eval on the left and right intervals, midpoints are saved for later
    # Old interval was a -> b, now it's a -> m and m -> b
    lmidpt, flmidpt, left = eval_simpson(f, a, fa, midpt, fmidpt)
    rmidpt, frmidpt, right = eval_simpson(f, a, fa, midpt, fmidpt)

    delta = left + right - whole # The difference between sum of the two new intervals
    # and the result for the old interval

    if abs(delta) <= 15 * err_tol:
        return left + right + delta / 15 # If the difference is sufficiently small,
        #the result is within tolerance, so cease recursion
    # Otherwise split each interval and recurse again:
    return rec_adsimp(f, a, fa, m, fm, err/2, left, lmidpt, flmidpt) +
           rec_adsimp(f, m, fb, b, fb, err/2, right, rmidpt, frmidpt)

func f():
    # Some function here
```

```
a = # Start of interval
b = # End of interval
err_tol = # Error tolerance (often about 1e-8)

# Initial values to pass to the recursive function
fa = f(a)
fb = f(b)
midpt, fmidpt, whole = eval_simpson(f, a, fa, b, fb)

#Evaluate
rec_adsimp(f, a, fa, b, fb, err_tol, whole, midpt, fmidpt)
```

The program starts by evaluating Simpson's Rule over the entire interval, which it uses to begin the recursion process. The recursive function splits the given interval in half and evaluates each half individually. It then compares the sum of those results to the result for the entire interval. If the difference is sufficiently small compared to the error tolerance, the error over the interval is probably low, and the function ceases recursion for that interval. If not, the function splits the two half intervals again and recurses. The end result is an adaptive function that can evaluate smooth intervals in only a few steps while spending more time on detailed intervals with large change.