

1. Composite Simpson's Rule

I've included this scratchwork to demonstrate how I programmed my second function for approximating integrals since this is still Composite Simpson's Rule, but written differently than the formula given to us in the project pdf as well as the sources I used.

ex: $h = \frac{1-0}{6} = \frac{1}{6}$
 $n=6$ $b=1$ $a=0$ $f(x) = \sum_{j=1}^{n/2} \left(\int_{x_{2j-2}}^{x_{2j}} f(x) dx \right) = \frac{h}{3} \sum_{j=1}^{n/2} [f(x_{2j-2}) + 4f(x_{2j-1}) + f(x_{2j})]$

ignore exact $f(x_j)$, understand recursion
 $\sum_{j=1}^{n/2}$ goes from 1 to 3
 $j=1$ $\text{sum} = 0$ before loop starts
approximation without error function

$j=1$ $\text{sum} = \text{sum} + \left[\frac{h}{3} (f(x_0) + 4f(x_1) + f(x_2)) \right]$
 $j=2$ $\text{sum} = \text{sum} + \left[\frac{h}{3} (f(x_2) + 4f(x_3) + f(x_4)) \right]$
 $j=3$ $\text{sum} = \text{sum} + \left[\frac{h}{3} (f(x_4) + 4f(x_5) + f(x_6)) \right]$

- end points evaluated once (endpoints are even)
- inner even points are evaluated 2x
- odd points evaluated once
- we can factor out $\frac{h}{3}$ and apply after summation is complete

$x_0 = a + (2 \cdot 1 - 2)h = a = 0$ $x_1 = a + (2 \cdot 1 - 1)h = a + h = \frac{1}{6}$ $x_2 = a + (2 \cdot 1)h = a + 2h = \frac{2}{6}$
 $x_2 = a + (2 \cdot 2 - 2)h = \frac{2}{6}$ $x_3 = a + (2 \cdot 2 - 1)h = \frac{3}{6}$ $x_4 = a + (2 \cdot 2)h = \frac{4}{6}$
 $x_4 = \frac{4}{6}$ $x_5 = \frac{5}{6}$ $x_6 = \frac{6}{6}$

2. Error Analysis

We can set an appropriate upper bound for our error by noticing that the fourth derivative has 2 critical points on $[0,1]$; in other words, the fifth derivative has 2 zeros on our interval. After evaluating the zeros of the fifth derivative we find that the largest absolute value of the fourth derivative occurs at $\gamma = 0$. This lets us set a "lazy" error bound as we can see that our absolute error is less than our bound by about a whole power of 10 in each approximation. None the less, our error and bounds decrease as n increases. Python has its own $\text{erf}(z)$ function as well to verify proper calculations of our absolute error.

Between both methods, we get very similar approximations, with variances only in the last few decimal places so we can assume their error bounds and absolute errors are almost identical. As we've discussed in class, we expect the error in Composite Simpson's Rule to be $O(h^4)$. This means that as h is decreased (or increased) by a certain factor (the reciprocal of that factor is applied to n), the error will decrease (or increase) by that factor to the power of 4. This algorithm behaves exactly as we hope; each time h is reduced by $\frac{1}{2}$, the error bound reduces exactly by $(\frac{1}{2})^4$, although the absolute errors do not reduce by exactly $\frac{1}{16}$.

To determine n such that our error is guaranteed to be less than 10^{-12} , we must set up an inequality with our error bound and 10^{-12} . With some algebra we can rearrange the inequality to give us n by first converting h^4 in our error bound to $((b-a)/n)^4$. After some calculations we find that n must be greater than or equal to 524 to guarantee an error less than 10^{-12} .

Your input: find the local and global minima and maxima of $f = \frac{(32x^4 - 96x^2 + 24)e^{-x^2}}{\sqrt{\pi}}$ on the interval $[0, 1]$

CRITICAL POINTS

$$(x, f(x)) = \left(0, \frac{24}{\sqrt{\pi}}\right) \approx (0, 13.5405500051462)$$

$$(x, f(x)) = \left(\frac{\sqrt{10 - 2\sqrt{10}}}{2}, \frac{-216 + 2(10 - 2\sqrt{10})^2 + 48\sqrt{10}}{\sqrt{\pi}e^{\frac{5}{2} - \frac{\sqrt{10}}{2}}}\right)$$

$$\approx (0.958572464613819, -8.37198799030836)$$

$$f^4(x) = \frac{8}{\sqrt{\pi}} \cdot e^{x^2} \cdot (4x^4 - 12x^2 + 3)$$

critical points on $[0; 1]$
at $x = 0, \frac{\sqrt{10-2\sqrt{10}}}{2} (x_1, x_2)$

$$|f(x_1)| > |f(x_2)|$$

$$|E(f)| = \left| \frac{-h^4(b-a)}{180} f^4(\xi_x) \right|$$

$$\leq \underbrace{\left| \frac{-h^4(b-a)}{180} f^4(0) \right|}_{\text{bound on error}} < 10^{-12}$$

find n that
← guarantees this

$$\rightarrow \frac{1}{180} \cdot \left(\frac{b-a}{n}\right)^4 \cdot (b-a) f^4(0) < 10^{-12} \rightarrow \frac{(b-a)^5 f^4(0)}{180 n^4} < 10^{-12}$$

$$\rightarrow \frac{f^4(0)}{n^4} < 1.8 \times 10^{-10} \rightarrow n^4 > \frac{f^4(0)}{1.8 \times 10^{-10}} \approx \frac{13.54055}{1.8 \times 10^{-10}} = \frac{24}{\sqrt{\pi} \cdot 1.8 \times 10^{-10}}$$

$$\rightarrow n > \sqrt[4]{\frac{24}{\sqrt{\pi} \cdot 1.8 \times 10^{-10}}} \approx 523.710101$$

n must be 524

- even
- guarantees error less than 10^{-12}

```

C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python37_64\python.exe
sub intervals: 2 length of sub intervals: 0.5
approximation: 0.843102830042981
error bound: 0.004701579862897969 abs error: 0.0004020370932661388
sub intervals: 4 length of sub intervals: 0.25
approximation: 0.8427360513893569
error bound: 0.0002938487414311231 abs error: 3.525843964202746e-05
sub intervals: 8 length of sub intervals: 0.125
approximation: 0.8427030358459557
error bound: 1.8365546339445192e-05 abs error: 2.2428962408449493e-06
sub intervals: 16 length of sub intervals: 0.0625
approximation: 0.8427009335720539
error bound: 1.1478466462153245e-06 abs error: 1.406223389954775e-07
sub intervals: 256 length of sub intervals: 0.00390625
approximation: 0.8427007929518632
error bound: 1.75147498506977e-11 abs error: 2.148281552649678e-12
sub intervals: 512 length of sub intervals: 0.001953125
approximation: 0.8427007929498498
error bound: 1.0946718656686063e-12 abs error: 1.3489209749195652e-13
sub intervals: 522 length of sub intervals: 0.0019157088122605363
approximation: 0.842700792949837
error bound: 1.013168755202985e-12 abs error: 1.2212453270876722e-13
sub intervals: 524 length of sub intervals: 0.0019083969465648854
approximation: 0.8427007929498412
error bound: 9.977888633495525e-13 abs error: 1.2634338020234281e-13
recursive method approximations
2 sub intervals: 0.843102830042981
4 subintervals: 0.8427360513893569
8 sub intervals: 0.8427030358459556
16 subintervals: 0.8427009335720541
256 subintervals: 0.842700792951863
512 subintervals: 0.8427007929498492
522 subintervals 0.84270079294984
524 subintervals: 0.8427007929498375

```

3. Resources

<https://www.youtube.com/watch?v=3MNGU5w1mH8&list=PLQ9rYUh73b8SiBfAZJ88HGHiYSSMocQ8K&index=11>

<https://personal.math.ubc.ca/~pwalls/math-python/integration/simpsons-rule/>