

# Adaptive Integration

Sunday, October 11, 2020

5:39 PM

This is ch 4.6 in Burden + Faires, but we'll follow ch 5.7 in Driscoll and Brauer  
Alg. 4.3 is hard to follow

Idea:

Software packages ( Mathematica's NIntegrate  
Matlab's integral (vs "trapz")  
Python's scipy.integrate.quad )

are not going to ask the user for #nodes,  
instead they're going to ask for an accuracy (w/ a sensible default)

How can we do this?

Key Mathematical idea: a posteriori error estimate

"a priori" vs "a posteriori" come up a lot in other subjects too, so let's review them:

a priori = estimate before you've done work

- + always valid
- may require knowledge you don't have
- can't exploit it when you get lucky,  
i.e., it's always pessimistic

Ex: Simpson's Rule

$$\int_a^b f(x) dx = \frac{h}{3} (f(x_0) + 4f(x_1) + f(x_2)) - \frac{h^5}{90} f^{(4)}(\xi)$$

for some  $\xi \in (a, b)$ ,  $h = \frac{b-a}{2}$

our "a priori"  
error estimate

a posteriori = estimate after you've done work

- + sometimes easier, sometimes more useful  
(no unknown things involved)
- + can adapt to your specific situation, and it takes  
it into account if you got lucky
- not useful for prediction or planning, only useful  
for verifying/certifying
- in some cases (like for our usage in integration)  
it is a heuristic or based on unverifiable assumptions.

so, let's make an a posteriori error estimate for integration,

ie., a practical way to evaluate the error, so we know when we have enough nodes

Start w/ composite Simpson's rule (though you could do a similar derivation for other rules)  
Write  $S(n)$  to be Simpson's rule w/  $n$  nodes, or  $S(h)$

Recall non-composite Simpson has error  $-\frac{h^5}{90} f^{(4)}(\xi)$  ( $h = \frac{b-a}{2}$ )  $\xi \in (a, b)$

and composite Simpson has error  $-\frac{b-a}{180} h^4 f^{(4)}(\eta)$   $\eta \in (a, b)$

So composite Simpson's Rule is  $O(h^4) = O(n^{-4})$

Apply Richardson extrapolation

$$R(h/2) = S(h/2) + \frac{S(h/2) - S(h)}{15} \leftarrow = 4^4 - 1$$

$$\text{or } R(2n) = S(2n) + \frac{S(2n) - S(n)}{15}$$

Assumption:  $R(2n)$  is so much more accurate than  $S(2n)$  that the error in  $R(2n)$  is negligible, and so

$$\begin{aligned} E_{\text{error}} &= \int_a^b f(x) dx - S(2n) \approx \underbrace{R(2n) - S(2n)}_{\text{our a posteriori error estimate}} \\ &= \boxed{\frac{S(2n) - S(n)}{15}} = \hat{E} \end{aligned}$$

So, basic strategy: double # of  $n$  until  $|\hat{E}|$  is small.

Details:

- How small should  $|\hat{E}|$  be? ie., pick a tolerance "tol" and require  $|\hat{E}| < \text{tol}$ ?

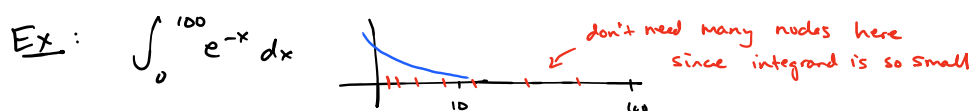
well, it often makes sense to ask for a relative error.

In practice, we usually do both:

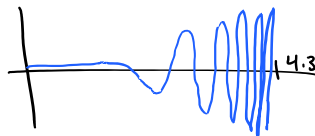
$$\text{Stop when } |\hat{E}| < \text{tol}_{\text{absolute}} + \text{tol}_{\text{relative}} \cdot S(n)$$

- Doubling  $n$  is going to lead to a lot of nodes very quickly.

Observation: we often don't need dense nodes everywhere.



Ex:  $f(x) = (x+1)^2 \cdot \cos\left(\frac{2x+1}{x-4.3}\right)$



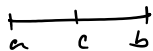
want a lot of nodes here

ie., we want to be **ADAPTIVE**

A nice, popular way to do this is **Divide and Conquer** (this is a general class of techniques beyond just integration)  
Idea:

1) estimate  $\hat{E}$  and stop if  $|\hat{E}|$  is small enough

2) Split  $[a, b]$  in two,  $[a, c]$  and  $[c, b]$  where  $c = \frac{a+b}{2}$



note that  $\int_a^b f(x) dx = \int_a^c f(x) dx + \int_c^b f(x) dx$

Recurse, integrating on  $[a, c]$  and estimating its error  $\hat{E}_{left}$   
and on  $[c, b]$  and estimating its error  $\hat{E}_{right}$

That's the basic idea.

Usually, we use composite Simpson's rule w/  $n=4$ , and are careful to reuse any  $f(\text{node})$  computations. See pseudocode

**FUNCTION** **Adaptive Integration** ( $f, a, b, \text{tol}_{abs}, \text{tol}_{rel}$ )

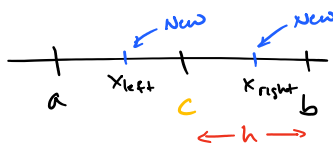
$c = \frac{a+b}{2}$

**return**  $\hat{I}$ , nodes = Recursive Integral ( $a, f(a), b, f(b), c, f(c)$ ) // and, implicitly,  $\text{tol}_{abs}, \text{tol}_{rel}$

**FUNCTION** **Recursive Integral** ( $a, f(a), b, f(b), c, f(c)$ )

$x_{left} = \frac{a+c}{2}$

$x_{right} = \frac{c+b}{2}$



nodes =  $\{a, x_{left}, c, x_{right}, b\}$

$h = \frac{b-a}{2}$

$S_2 = h/3 (f(a) + 4f(c) + f(b))$  // regular (non-composite) Simpson's rule

$S_4 = h/2 \cdot 1/3 (f(a) + 4f(x_{left}) + 2f(c) + 4f(x_{right}) + f(b))$  // composite Simpson

$\hat{E} = 1/15 (S_4 - S_2)$

// often use  $1/10$  instead of  $1/15$  to be more conservative

if  $|\hat{E}| < \text{tol}_{abs} + \text{tol}_{rel} \cdot |S_4|$

// error is acceptable

**return**  $S_4$ , nodes

else

// error is too large, so bisect

$\hat{I}_{\text{left}}, \text{nodes}_{\text{left}} = \text{RecursiveIntegral}(a, f(a), c, f(c), x_{\text{left}}, f(x_{\text{left}}))$

$\hat{I}_{\text{right}}, \text{nodes}_{\text{right}} = \text{RecursiveIntegral}(c, f(c), b, f(b), x_{\text{right}}, f(x_{\text{right}}))$

return  $\hat{I}_{\text{left}} + \hat{I}_{\text{right}}, \text{nodes}_{\text{left}} \cup \text{nodes}_{\text{right}}$  // as lists, the first entry  
in  $\text{nodes}_{\text{right}}$  is a duplicate  
of last entry in  $\text{nodes}_{\text{left}}$

end

### Summary

All professional integration packages are adaptive so

- ① they don't waste time where extra nodes aren't needed
- ② they automatically generate nodes until a tolerance is reached,  
and give a (heuristic) "guarantee" on the final error.