Numerical Analysis Hilary Term 2020
Lecture 3: Newton-Cotes Quadrature (continued)

See Chapter 7 of Süli and Mayers.

Motivation: we've seen oscillations in polynomial interpolation—the Runge phenomenon–for high-degree polynomials.

Idea: split a required integration interval $[a, b] = [x_0, x_n]$ into n equal intervals $[x_{i-1}, x_i]$ for $i = 1, \ldots, n$. Then use a **composite rule**:

$$\int_a^b f(x)\,\mathrm{d}x = \int_{x_0}^{x_n} f(x)\,\mathrm{d}x = \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f(x)\,\mathrm{d}x$$

in which each $\displaystyle\int_{x_{i-1}}^{x_i} f(x)\,\mathrm{d}x$ is approximated by quadrature.

Thus rather than increasing the degree of the polynomials to attain high accuracy, instead increase the number of intervals.

Trapezium Rule:

$$\int_{x_{i-1}}^{x_i} f(x)\,\mathrm{d}x = \frac{h}{2}[f(x_{i-1}) + f(x_i)] - \frac{h^3}{12} f''(\xi_i)$$

for some $\xi_i \in (x_{i-1}, x_i)$

Composite Trapezium Rule:

$$\begin{aligned}\int_{x_0}^{x_n} f(x)\,\mathrm{d}x &= \sum_{i=1}^{n} \left[\frac{h}{2}[f(x_{i-1}) + f(x_i)] - \frac{h^3}{12} f''(\xi_i)\right] \\ &= \frac{h}{2}[f(x_0) + 2f(x_1) + 2f(x_2) + \cdots + 2f(x_{n-1}) + f(x_n)] + e_h^{\mathrm{T}}\end{aligned}$$

where $\xi_i \in (x_{i-1}, x_i)$ and $h = x_i - x_{i-1} = (x_n - x_0)/n = (b - a)/n$, and the error e_h^{T} is given by

$$e_h^{\mathrm{T}} = -\frac{h^3}{12} \sum_{i=1}^{n} f''(\xi_i) = -\frac{nh^3}{12} f''(\xi) = -(b - a)\frac{h^2}{12} f''(\xi)$$

for some $\xi \in (a, b)$, using the Intermediate-Value Theorem n times. Note that if we halve the stepsize h by introducing a new point halfway between each current pair (x_{i-1}, x_i), the factor h^2 in the error should decrease by four.

Another composite rule: if $[a, b] = [x_0, x_{2n}]$,

$$\int_a^b f(x)\,\mathrm{d}x = \int_{x_0}^{x_{2n}} f(x)\,\mathrm{d}x = \sum_{i=1}^{n} \int_{x_{2i-2}}^{x_{2i}} f(x)\,\mathrm{d}x$$

in which each $\displaystyle\int_{x_{2i-2}}^{x_{2i}} f(x)\,\mathrm{d}x$ is approximated by quadrature.

Simpson's Rule:

$$\int_{x_{2i-2}}^{x_{2i}} f(x)\,\mathrm{d}x = \frac{h}{3}[f(x_{2i-2}) + 4f(x_{2i-1}) + f(x_{2i})] - \frac{(2h)^5}{2880} f''''(\xi_i)$$

for some $\xi_i \in (x_{2i-2}, x_{2i})$.

Composite Simpson's Rule:

$$\begin{aligned}\int_{x_0}^{x_{2n}} f(x)\,\mathrm{d}x = & \sum_{i=1}^{n}\left[\frac{h}{3}[f(x_{2i-2}) + 4f(x_{2i-1}) + f(x_{2i})] - \frac{(2h)^5}{2880}f''''(\xi_i)\right] \\ = & \frac{h}{3}[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_4) + \cdots \\ & + 2f(x_{2n-2}) + 4f(x_{2n-1}) + f(x_{2n})] + e_h^{\mathrm{S}}\end{aligned}$$

where $\xi_i \in (x_{2i-2}, x_{2i})$ and $h = x_i - x_{i-1} = (x_{2n} - x_0)/2n = (b-a)/2n$, and the error e_h^{S} is given by

$$e_h^{\mathrm{S}} = -\frac{(2h)^5}{2880}\sum_{i=1}^{n} f''''(\xi_i) = -\frac{n(2h)^5}{2880}f''''(\xi) = -(b-a)\frac{h^4}{180}f''''(\xi)$$

for some $\xi \in (a,b)$, using the Intermediate-Value Theorem n times. Note that if we halve the stepsize h by introducing a new point half way between each current pair (x_{i-1}, x_i), the factor h^4 in the error should decrease by sixteen (assuming f is smooth enough).

Adaptive (or automatic) procedure: if S_h is the value given by Simpson's rule with a stepsize h, then

$$S_h - S_{\frac{1}{2}h} \approx -\frac{15}{16}e_h^{\mathrm{S}}.$$

This suggests that if we wish to compute $\int_a^b f(x)\,\mathrm{d}x$ with an absolute error ε, we should compute the sequence $S_h, S_{\frac{1}{2}h}, S_{\frac{1}{4}h}, \ldots$ and stop when the difference, in absolute value, between two consecutive values is smaller than $\frac{16}{15}\varepsilon$. That will ensure that (approximately) $|e_h^{\mathrm{S}}| \le \varepsilon$.

Sometimes much better accuracy may be obtained: for example, as might happen when computing Fourier coefficients, if f is periodic with period $b-a$ so that $f(a+x) = f(b+x)$ for all x.

Matlab:

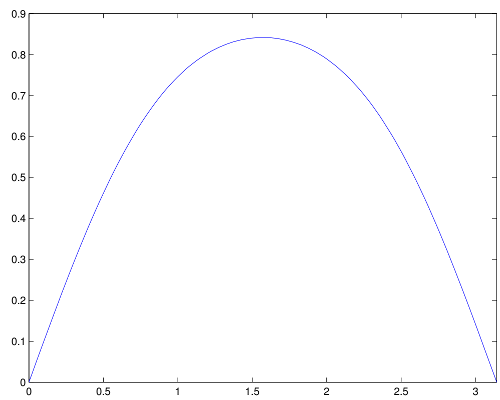
```
>> help adaptive_simpson
 ADAPTIVE_SIMPSON  Adaptive quadrature with Simpson's rule
    S = ADAPTIVE_SIMPSON(F, A, B, TOL, NMAX) computes an approximation
    to the integral of F on the interval [A, B] . It will take a
    maximum of NMAX steps and will attempt to determine the integral
    to a tolerance of TOL.  If omitted, NMAX will default to 100.

    The function uses an adaptive Simpson's rule, as described
    in lectures.

>> format long g   % see more than 5 digits
>> f = @(x) sin(x);
>> s = adaptive_simpson(f, 0, pi, 1e-7)
Step 1 integral is 2.0943951024.
Step 2 integral is 2.0045597550, with error estimate 0.089835.
Step 3 integral is 2.0002691699, with error estimate 0.0042906.
```

```
Step 4 integral is 2.0000165910, with error estimate 0.00025258.
Step 5 integral is 2.0000010334, with error estimate 1.5558e-05.
Step 6 integral is 2.0000000645, with error estimate 9.6884e-07.
Step 7 integral is 2.0000000040, with error estimate 6.0498e-08.
Successful termination at iteration 7.
s =
          2.00000000403226

>> g = @(x) sin(sin(x));
>> fplot(g, [0 pi])
```




```
>> s = adaptive_simpson(g, 0, pi, 1e-7)
Step 1 integral is 1.7623727094.
Step 2 integral is 1.8011896009, with error estimate 0.038817.
Step 3 integral is 1.7870879453, with error estimate 0.014102.
Step 4 integral is 1.7865214631, with error estimate 0.00056648.
Step 5 integral is 1.7864895607, with error estimate 3.1902e-05.
Step 6 integral is 1.7864876112, with error estimate 1.9495e-06.
Step 7 integral is 1.7864874900, with error estimate 1.2118e-07.
Step 8 integral is 1.7864874825, with error estimate 7.5634e-09.
Successful termination at iteration 8.
s =
           1.7864874824541


>> s = adaptive_simpson(g, 0, pi, 1e-7, 3)
Step 1 integral is 1.7623727094.
Step 2 integral is 1.8011896009, with error estimate 0.038817.
Step 3 integral is 1.7870879453, with error estimate 0.014102.
*** Unsuccessful termination: maximum iterations exceeded ***
The integral *might* be 1.7870879453.
s =
          1.78708794526495
```