Math 471 (Numerical methods)

Chapter 6 . Numerical Diferentiation and Integration

Overlap §6.2–6.5 of Bradie

\$6.2 Numerical differentiation So far, we have learned centered difference for 1st and 2nd derivatives

$$\begin{aligned} f'(x) =& \frac{f(x+h)-f(x-h)}{2h} + C_1 \cdot f^{(3)}(\xi_1)h^2 \\ f''(x) =& \frac{f(x+h)-2f(x)+f(x-h)}{h^2} + C_2 \cdot f^{(4)}(\xi)h^2 \end{aligned} \qquad \text{for some } \xi_1, \xi_2 \in [x-h, x+h]$$

Here, C_1, C_2 are universal constants that are independent of h, x and $f(x)$. The above formulas are made equalities, the error terms being given explicitly, and they can be proved with Taylor series expansion (proper number of terms should be used in each Taylor series to yield the results).

In a less precise way, we can use an **error bound** in an inequality,

$$\left| f'(x) - \frac{f(x+h)-f(x-h)}{2h} \right| \le C_1 M_3 h^2$$

$$\left| f''(x) - \frac{f(x+h)-2f(x)+f(x-h)}{h^2} \right| \le C_2 M_4 h^2$$

where $M_k = \max_{x\in[a,b]} |f^{(k)}(x)|$ for some $[a,b]$ that covers $[x-h, x+h]$. The left hand side of the above equations is called the **truncation error**, indicating the inaccuracy caused by using a finite h instead of letting $h \to 0$ for computing derivatives. The error bounds on the right hand side are both $O(h^2)$.

The derivation of these finite difference schemes is based on Taylor series and the stencils that are involved in the schemes.

Example. If the value of $f(x)$ is only available for $x \le x_0$, then centered difference involving $f(x_0-h)$ and $f(x_0+h)$ can not be directly used to compute $f'(x_0)$. Alternatively, one can use a biased 2-point stencil

$$f(x_0) \quad \text{and} \quad f(x_0-h) \quad \text{for small } h > 0.$$

We already knew this leads to the backward difference

$$f'(x_0) \approx \frac{f(x_0)-f(x_0-h)}{h}$$

but how is it established using Taylor series? To this end, expand $f(x_0 - h)$ using $f(x_0), f'(x_0), f''(x_0)...$

$$f(x_0 - h) = f(x_0) - f'(x_0)h + f''(x_0)h^2/2 + ... \tag{1}$$

We don't know how many terms will be needed so the infinite series is used (later, we can always switch to a finite series with ξ once things become clearer). Then, use undetermined coef a, b to combine $f(x_0)$ and $f(x_0-h)$, as well as the Taylor series version (1)

$$af(x_0) + bf(x_0 - h) = (a + b)f(x_0) - bf'(x_0)h + bf''(x_0)h^2/2 + ... \tag{2}$$

In order for the RHS to become an approximation of $f'(x_0)$, the coef need to satisfy

$$a + b = 0 \quad \text{and} \quad -bh = 1$$

Since there are only two unknowns, we shall not introduce a third condition. The above system yields solution

$$a = 1/h \quad b = -1/h.$$

Plug into (2),

$$\begin{aligned}\frac{1}{h}f(x_0) - \frac{1}{h}f(x_0 - h) &= f'(x_0) - f''(x_0)h/2 + ... \\ &= f'(x_0) - f''(\xi)h/2 \text{ for some } \xi \in [x_0 - h, x_0]\end{aligned}$$

Now, not only we have the backward difference scheme, but we have an expression for the truncation error and the rate of convergence is $O(h)$.

Example. In numerical solutions (for the unknown $f(x)$) of $f''(x) = g(x)$ with boundary conditions $f'(0) = k_1, \;\; f'(L) = k_2$, the first derivative $f'(0)$ should be approximated using $f(x)$ with $x \geq 0$. So, again, a biased stencil is used. It is easy to check that a 2-point stencil at x_0 and $x_0 + h$ leads to the forward difference

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0)}{h} - f''(\xi)h/2$$

which, like its backward counterpart, is of accuracy $O(h)$. In general, we can use any 2 or 3 or 4 ... points that are located near and on the postive side of $x_0 = 0$. For instance, let's use $f(x_0), f(x_0+h), f(x_0+2h)$. First, write down the Taylor series of $f(x_0+h)$ and $f(x_0 + 2h)$ about $x = x_0$

$$f(x_0 + h) = f(x_0) + f'(x_0)h + f''(x_0)h^2/2 + f^{(3)}(x_0)h^3/6 + ... \tag{3}$$

$$f(x_0 + 2h) = f(x_0) + f'(x_0)(2h) + f''(x_0)(2h)^2/2 + f^{(3)}(x_0)(2h)^3/6 + ... \tag{4}$$

Then, use **undetermined coefficients** a, b, c to combine $f(x_0), f(x_0+h), f(x_0+2h)$ and their Taylor series, that is, $a * f(x_0) + b * (3) + c * (4)$,

$$
\begin{aligned}
&af(x_0) + bf(x_0+h) + cf(x_0+2h) = \\
&(a+b+c)f(x_0) + (b+2c)f'(x_0)h + (b+4c)f''(x_0)h^2/2 + (b+8c)f'''(x_0)h^3/6 + ...
\end{aligned}
\tag{5}
$$

To make the RHS an approximation of the derivative $f'(x_0)$, the coef should satisfy

$$
\begin{aligned}
a+b+c =& 0 \\
(b+2c)h =& 1 \\
b+4c =& 0
\end{aligned}
$$

With 3 unknowns and 3 equations, this system yields

$$
a = -\frac{3}{2h}, \quad b = \frac{4}{2h}, \quad c = -\frac{1}{2h}
$$

Plug into (5)

$$
\begin{aligned}
\frac{-3f(x_0) + 4f(x_0+h) - f(x_0+2h)}{2h} &= f'(x_0) - \frac{h^2}{3}f'''(x_0) + ... \\
&= f'(x_0) - \frac{h^2}{3}f'''(\xi)
\end{aligned}
$$

So, not only we obtain a biased 3-point stencil scheme for $f'(x_0)$, we reveal that the truncation error from this scheme is of $O(h^2)$, more accurate than the forward difference!

Note: with 4-point stencil, one could get even higher accuracy ....

Note: the grid points in numerical schemes do not necessary space equally. For instance, we can use $f(x_0), f(x_0+h_1), f(x_0+h_2)$ with any $h_2 > h_1 > 0$ to approximate $f'(x_0)$

§6.3 Extrapolation

The key idea of extrapolation is to repeatedly perform a numerical scheme with 2 or more different resolutions, and then simply combine these results in a way such that the final answer is of higher accuracy.

Let $F(f)$ is the exact quantity we want to approximate and $F_h(f)$ is the approximation with resolution h (or, $F_n(f)$ with computational complexity $O(n)$, etc.). For example

$$
F(f) = f'(x_0), \quad F_h(f) = \frac{f(x_0+h) - f(x_0)}{h}
$$

Another example,

$$F(f) = \int_a^b f(x)dx, \quad F_h(f) = h\left(\frac{f(a)}{2} + f(a+h) + f(a+2h) + ... + f(b-h) + \frac{f(b)}{2}\right)$$

which is the composite Trapezoidal rule with resolution $h = (b-a)/n$.

If $|F(f) - F_h(f)| = O(h^p)$, then by choosing coef a, b carefully, $aF_h(f) + bF_{2h}(f)$ can yield better accuracy

$$|F(f) - (aF_h(f) + bF_{2h}(f))| = O(h^{p'}) \quad \text{for} \quad p' > p.$$

To find a, b, we use undetermined coef and Taylor series, just like in the previous section.

Example. We can show by using Taylor series

$$\begin{aligned} f'(x_0) =& F_h(f) + O(h) \\ =& \frac{f(x_0+h) - f(x_0)}{h} - f''(x_0)h/2 - f'''(x_0)h^2/6 + ... \end{aligned} \tag{6}$$

Now, replace every occurrance of h with $2h$,

$$\begin{aligned} f'(x_0) =& F_{2h}(f) + O(2h) \\ =& \frac{f(x_0+2h) - f(x_0)}{2h} - f''(x_0)2h/2 - f'''(x_0)(2h)^2/6 + ... \end{aligned} \tag{7}$$

Then, combine them with coef a, b, i.e. perform $a * (6) + b * (7)$

$$(a+b)f'(x_0) = aF_h(f) + bF_{2h}(f) - f''(x_0)(\frac{ah}{2} + \frac{2bh}{2}) - f'''(x_0)(\frac{ah^2}{6} + \frac{4bh^2}{6})... \tag{8}$$

Now, in order to find 2 undetermined coef, we need to 2 equations. Obviously, the LHS of the above equation should give $f'(x_0)$, therefore $a + b = 1$ is one equation. Then, we can make the $O(h)$ term on the RHS vanish by letting $\frac{ah}{2} + \frac{2bh}{2} = 0$. Thus,

$$a + b = 1, \qquad \frac{ah}{2} + \frac{2bh}{2} = 0$$

$$\implies a = 2, \quad b = -1$$

Plug into (8)

$$f'(x_0) = 2F_h(f) - F_{2h}(f) + f'''(x_0)\frac{h^2}{3} + ...$$

This is of accuracy $O(h^2)$. In fact, using the definition of F_h, F_{2h} as in (6), (7), we see that $2F_h(f) - F_{2h}(f)$ is the same as the biased 3-point stencil scheme discussed in the previous section.

§6.4 Newton-Cotes Quadratures for Intergration

The derivation of Newton-Cotes formula for approximation of $\int_a^b f(x)dx$ always starts with approximation of the integrand $f(x)$ itself. Lagrange form of polynomial interpolation is in order,

$$f(x) \approx \sum_{i=0}^{n} f(x_i) l_i(x) \tag{9}$$

where $\{x_i\}_{i=0}^n$ are the data points and $\{l_i(x)\}_{i=0}^n$ are the Lagrange polynomials (the "atoms" used to build general polynomial interpolations)

$$l_k(x) = \frac{\prod_{i \neq k}(x - x_i)}{\prod_{i \neq k}(x_k - x_i)}$$

Integrating (9) immediately yields the Newton-Cotes formula

$$\int_a^b f(x)dx \approx \int_a^b \sum_{i=0}^{n} f(x_i) l_i(x) dx = \sum_{i=0}^{n} f(x_i) \left(\int_a^b l_i(x) dx \right)$$

The very right hand side above is a linear combination of *discrete* values of $f(x)$ extracted at $x_0, x_1, ..., x_n$. The weights are often denoted by

$$w_i = \int_a^b l_i(x)dx = \int_a^b \frac{\prod_{i \neq j}(x - x_j)}{\prod_{i \neq k}(x_i - x_j)}$$

so that

$$\int_a^b f(x) \approx \sum_{i=0}^{n} w_i f(x_i).$$

Example. Let $n = 1$, $x_0 = a$, $x_1 = b$. This is a **closed form** of N-C formula because the end points are included. Then, the two Lagrange polynomials are

$$l_0(x) = \frac{x - a}{b - a}, \qquad l_1(x) = \frac{x - b}{a - b}$$

and upon intergration, the weights are

$$w_0 = \int_a^b l_0(x)dx = \frac{b - a}{2}, \qquad w_1 = \int_a^b l_1(x)dx = \frac{b - a}{2}$$

and therefore

$$\int_a^b f(x)dx \approx \frac{b - a}{2}(f(a) + f(b))$$

which is the Trapezoidal rule from Calc I.

Example. If $n = 2$ in a closed N-C quadrature, then the stencil includes $x_0 = a,\ x_1 = (a+b)/2,\ \ x_2 = b$. The 3 Lagrange polynomials $l_0(x),\ l_1(x),\ l_2(x)$ then follow and so do the weights w_0, w_1, w_2. It turns out

$$w_0 = w_2 = \frac{b-a}{6}, \quad w_1 = \frac{4(b-a)}{6}$$

which gives us the Simpson's rule

$$\int_a^b f(x)dx \approx \frac{b-a}{6}\left(f(a) + 4f((a+b)/2) + f(b)\right)$$

Example. The **open form** N-C quadratures do NOT use end points $x = a,\ \ x = b$. The number of subintervals therefore become $n+2$ in order to generate $n+1$ *interior* data points

$$x_0 = a + h,\, x_1 = a + 2h,\, ...,\, x_n = a + (n+1)h = b - h$$

where

$$h = \frac{b-a}{n+2}.$$

When $n = 0$, only one data point $x_0 = (b+a)/2$ is used. There is only one Lagrange polynomial as well

$$l_0(x) = 1.$$

It gives the only one weight $w_0 = \int_a^b l_0(x) = b - a$. Thus,

$$\int_a^b f(x) \approx (b-a)f((a+b)/2)$$

which is the midpoint rule.

What about <u>accuracy</u>? To answer this question, we state the following theorem without proof

Theorem. The Newton-Cotes quadrature with $(n+1)$ data points (either closed or open form) has error

$$\begin{cases} C_1(b-a)^{n+3} f^{(n+2)}(\xi) & \text{when n is even} \\ C_2(b-a)^{n+2} f^{(n+1)}(\xi) & \text{when n is odd} \end{cases}$$

where C_1, C_2 are universal constants independent of $a, b, f(x)$ but depend on n.

To further understand this theorem, let's start with the case when n is odd. Since the $(n+1)$th derivative of f is in the error, it becomes zero wheneven $f(x)$ is a polynomial of degree n or less. It in general does not vanish if $f(x)$ is a polynomial of degree $n+1$, e.g.

$$\frac{d^{n+1}}{dx^{n+1}}(a_0 + a_1 x + a_2 x^2 + ... + a_{n+1}x^{n+1}) = (n+1)!a_{n+1}.$$

This is consistent with the fact that the N-C formula is derived from integrating the n-th degree polynomial interpolation of $f(x) \approx \sum f(x_i)l_i(x)$. Such approximation becomes exact whenever $f(x)$ itself is a polynomial of degree n or less — this is the uniqueness theorem from last chapter. So, the N-C quadrature also becomes exact whenever $f(x)$ itself is a polynomial of degree n or less.

The testing of N-C formula on polynomials is a very restricted notion because $f(x)$ can be anything. But nevertheless, it gives some sense of accuracy. The more polynomials that can be computed EXACTLY by an N-C formular, the better precision one should expect. To this end, we introduce the notion of **degree of precision**

<u>Definition</u>. The degree of precision of a numerical integration is the maximum degree of polynomials that the quadrature can compute EXACTLY. In other words, the degree of precision is k if

$$\int_a^b f(x) = \sum w_i f(x_i) \qquad \text{for any } f(x) = a_0 + a_1 x + ...a_k x^k$$

$$\text{but} \int_a^b g(x) \neq \sum w_i g(x_i) \qquad \text{for some } g(x) = b_0 + b_1 x + ...b_{k+1}x^{k+1}$$

So what happens to the N-C formula is: when n is odd, the degree of precision is as high as guaranteed by polynomial interpolation of the integrand $f(x)$ itself; when n is even, we see from the theorem that, the degree of precision is one higher than guaranteed by polynomial interpolation of $f(x)$.

§6.5 Composite Newton-Cotes Quadratures

A disadvantage of using a single N-C quadrature to compute an integral is that, increasingly high degree polynomials tend to generate wild oscillations especially when the derivatives of $f(x)$ are not well bounded. This leads to large error in the N-C formulas as well. A remedy for this phonomenon is inspired by **piecewise** interpolation, in which case the original interval is partitioned into subintervals and on each and every subinterval, an interpolation is performed. This way, lower degree polynomials are used for each subinterval and spurious oscillations tend to fade away. The same principle applies to N-C formulas.

Let $[A, B]$ be the entire interval over which the integral is to be appproximated. Partition once on the integral

$$\int_A^B f(x)dx = \sum_{j=1}^{N} \int_{X_{j-1}}^{X_j} f(x)dx \tag{10}$$

where $X_j = A + jH$ are the interface points. Here, $H = (B - A)/N$ is the mesh size of the top level partition. Then, perform a second partition on each subinterval $[X_{j-1}, X_j]$ so that

$$\begin{aligned} x_{i,j} &= X_{j-1} + i\frac{H}{n} && \text{for the closed form} \\ x_{i,j} &= X_{j-1} + (i+1)\frac{H}{n+2} && \text{for the open form} \end{aligned}$$

where $h = H/n$ or $H/(n+2)$ is the mesh size of the second level partition. On this level, N-C formula is constructed using data points $x_{0,j}, x_{1,j}, ..., x_{n,j}$ and should look like

$$\int_{X_{j-1}}^{X_j} \approx \sum_{i=0}^{n} w_i f(x_{i,j}). \tag{11}$$

Note that the weights here only have one index i because shifting $[X_{j-1}, X_j]$ with changing j does not affect w_i. It is the relative position of x_i within each subinterval that determines w_i. (In fact, scaling the intervals only affects w_i by the same scaling factor, so that we can simply calulate $\int_0^1 l_i(z)dz$ once and mutiply it with the length of the actual interval to get w_i)

Substitute (11) into (10) and arrive at the final form of composite N-C quadrature.

Example. With a Trapezoidal rule on each subinterval $[X_{j-1}, X_j]$,

$$\int_{X_{j-1}}^{X_j} f(x)dx \approx \frac{X_j - X_{j-1}}{2} \left(f(X_{j-1}) + f(X_j)\right)$$

and with $X_j = A + j(B - A)/N$ and $X_j - X_{j-1} = H = (B - A)/N$, the composite Trapezoidal rule reads

$$\int_A^B f(x) = \frac{H}{2} \left(f(A) + 2\sum_{j=1}^{N} f(A + jH) + f(B) \right).$$

What about the error? The theorem from previous section is useful. The above case corresponds to $n = 1$ in that theorem. So, applying Trapezoidal rule on each $[X_{j-1}, X_j]$

introduces an error $C(X_j - X_{j-1})^3 f''(\xi_j)$ and therefore, after summing up, the total error is

$$C\left|\sum_{j=1}^{N}(X_j - X_{j-1})^3 f''(\xi_j)\right| \leq CNH^3M_2$$

Here, we used $X_j - X_{j-1} = H$ and a uniform bound on the 2nd derivative $\max_{x\in[A,B]} |f''(x)| = M_2$. But $NH = B - A$ by definition. So

$$\text{total error of trap. rule } \leq C(B-A)M_2H^2.$$

Since H is the resolution (of the top level partition) and can be refined and refined, we see that the order of convergence of the composite Trapezoidal rule is $O(H^2)$. And it only requires $f(x)$ to have bounded derivatives up to 2nd order.