Lecture 22

Integration: Midpoint and Simpson's Rules

Midpoint rule

If we use the endpoints of the subintervals to approximate the integral, we run the risk that the values at the endpoints do not accurately represent the average value of the function on the subinterval. A point which is much more likely to be close to the average would be the midpoint of each subinterval. Using the midpoint in the sum is called the *midpoint rule*. On the i-th interval $[x_{i-1}, x_i]$ we will call the midpoint $\bar{x}_i$, i.e.

$$\bar{x}_i = \frac{x_{i-1} + x_i}{2}.$$

If $\Delta x_i = x_i - x_{i-1}$ is the length of each interval, then using midpoints to approximate the integral would give the formula

$$M_n = \sum_{i=1}^{n} f(\bar{x}_i)\Delta x_i \,.$$

For even spacing, $\Delta x_i = h = (b-a)/n$, and the formula is

$$M_n = h \sum_{i=1}^{n} f(\bar{x}_i) = h\,(\hat{y}_1 + \hat{y}_2 + \ldots + \hat{y}_n), \tag{22.1}$$

where we define $\hat{y}_i = f(\bar{x}_i)$.

While the midpoint method is obviously better than L_n or R_n, it is not obvious that it is actually better than the trapezoid method T_n, but it is.

Simpson's rule

Consider Figure 22.1. If f is not linear on a subinterval, then it can be seen that the errors for the midpoint and trapezoid rules behave in a very predictable way, they have opposite sign. For example, if the function is concave up then T_n will be too high, while M_n will be too low. Thus it makes sense that a better estimate would be to average T_n and M_n. However, in this case we can do better than a simple average. The error will be minimized if we use a weighted average. To find the proper weight we take advantage of the fact that for a quadratic function the errors EM_n and ET_n are exactly related by

$$|EM_n| = \frac{1}{2}|ET_n| \,.$$

Thus we take the following weighted average of the two, which is called Simpson's rule:

$$S_{2n} = \frac{2}{3}M_n + \frac{1}{3}T_n \,.$$

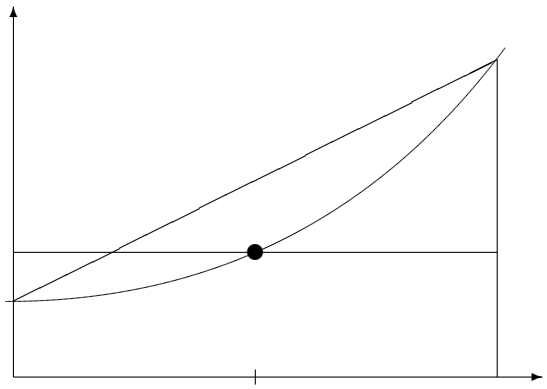

Figure 22.1: Comparing the trapezoid and midpoint method on a single subinterval. The function is concave up, in which case T_n is too high, while M_n is too low.

If we use this weighting on a quadratic function the two errors will exactly cancel.

Notice that we write the subscript as $2n$. That is because we usually think of $2n$ subintervals in the approximation; the n subintervals of the trapezoid are further subdivided by the midpoints. We can then number all the points using integers. If we number them this way we notice that the number of subintervals must be an even number.

The formula for Simpson's rule if the subintervals are evenly spaced is the following (with n intervals, where n is even):

$$S_n = \frac{h}{3}\left(f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \ldots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)\right) .$$

Note that if we are presented with data $\{x_i, y_i\}$ where the x_i points are evenly spaced with $x_{i+1} - x_i = \Delta x$, it is easy to apply Simpson's method:

$$S_n = \frac{h}{3}\left(y_0 + 4y_1 + 2y_2 + 4y_3 + \ldots + 2y_{n-2} + 4y_{n-1} + y_n\right). \tag{22.2}$$

Notice the pattern of the coefficients. The following program will produce these coefficients for n intervals, if n is an even number. Try it for $n = 6, 7, 100$.

```
function w = mysimpweights(n)
    % Computes the weights for Simpson's rule
    % Input:  n -- the number of intervals, must be even
    % Output: w -- a (column) vector with the weights, length n+1
    if rem(n,2) ~= 0
        error('n must be even for Simpsons rule')
    end
    w = 2*ones(n+1,1); % column vector, starts all 2's
    w(1) = 1; w(n+1) = 1; % set ends to 1's
    w(2:2:n)=4; % set even # entries to 4.
end
```

Simpson's rule is incredibly accurate. We will consider just how accurate in the next section. The one drawback is that the points used must either be evenly spaced, or at least the odd number points must lie exactly at the midpoint between the even numbered points. In applications where you can choose the spacing, this is not a problem. In applications such as experimental or simulated data, you might not have control over the spacing and then you cannot use Simpson's rule.

Error bounds

The trapezoid, midpoint, and Simpson's rules are all approximations. As with any approximation, before you can safely use it, you must know how good (or bad) the approximation might be. For these methods there are formulas that give upper bounds on the error. In other words, the worst case errors for the methods. These bounds are given by the following statements:

- Suppose f'' is continuous on [a,b]. Let $K_2 = \max_{x\in[a,b]} |f''(x)|$. Then the errors ET_n and EM_n of the Trapezoid and Midpoint rules, respectively, applied to $\int_a^b f\,dx$ satisfy

$$|ET_n| \le K_2 \frac{b-a}{12} h^2 \quad \text{and}$$
$$|EM_n| \le K_2 \frac{b-a}{24} h^2 \,.$$

- Suppose $f^{(4)}$ is continuous on [a,b]. Let $K_4 = \max_{x\in[a,b]} |f^{(4)}(x)|$. Then the error ES_n of Simpson's rule applied to $\int_a^b f\,dx$ satisfies

$$|ES_n| \le K_4 \frac{b-a}{180} h^4.$$

In practice K_2 and K_4 are themselves approximated from the values of f at the evaluation points.

The most important thing in these error bounds is the dependence on h. To emphasize this dependence, we sometimes use the **order notation** $O(\ )$. The trapezoid and midpoint method errors are $O(h^2)$, so the methods have order 2. The Simpson's rule error is $O(h^4)$, so it has order 4. If h is just moderately small, then there is a huge advantage with Simpson's method.

In MATLAB there is a built-in command for definite integrals: `integral(f,a,b)` where the `f` is a function and `a` and `b` are the endpoints. The command uses "adaptive Simpson quadrature", a form of Simpson's rule that checks its own accuracy and adjusts the grid size where needed. Here is an example of its usage:

```
≫ f = @(x) x.^(1/3).*sin(x.^3)
≫ I = integral(f,0,1)
```

Exercises

22.1 Using formulas (22.1) and (22.2), for the integral $\int_1^2 \sqrt{x}\,dx$ calculate M_4 and S_4 (by hand, but use a calculator and a lot of digits). Find the percentage error of these approximations, using the exact value. Compare with exercise 21.1.

22.2 Write a well-commented MATLAB **function** program `mymidpoint` that calculates the midpoint rule approximation for $\int f$ on the interval $[a, b]$ with n subintervals. The inputs should be f, a, b and n. Use your program on the integral $\int_1^2 \sqrt{x}\,dx$ to obtain M_4 and M_{100}. Compare these with exercise 22.1 and the true value of the integral.

22.3 Write a well-commented MATLAB **function** program `mysimpson` that calculates the Simpson's rule approximation for $\int f$ on the interval $[a, b]$ with n subintervals. It should call the program `mysimpweights` to produce the coefficients. Use your program on the integral $\int_1^2 \sqrt{x}\,dx$ to obtain S_4 and S_{100}. Compare these with exercise 22.1, exercise 22.2, and the true value of the integral.