5.1 Evaluating a polynomial

We will now look at evaluating a polynomial at a point. Your first thought may be: this is easy: just substitute the values, and presto. Unfortunately, this is not the case.

5.1.1 Execution time

First, there is the cost of performing this evaluation: how do we efficiently evaluate a polynomial?

Horrible evaluation

Suppose that the coefficients of a polynomial are stored in an array `coeffs` of length `n` such that `coeffs(1)` is the coefficient of t^{n-1} and `coeffs(n + 1)` is the constant coefficient. In this case, we could implement

```
function [result] = polyval( coeffs, t )
    result = 0;
    n = length(coeffs);

    for k = 1:length(coeffs)
        result = result + coeffs(k)*pow( t, n - k );
    end
end
```

If you were to implement powering as follows:

```
function [result] = pow( t, n )
    if n < 0
        result = 1.0/pow( t, -n );
    else
        result = 1.0;

        for i = 1:n
            result = result*t;
        end
    end
end
```

then there is a special place in hell reserved for you: evaluating a polynomial of degree n would have a run-time of $\Theta(n^2)$.

Less horrible evaluation

If you were to implement powering as follows:

```
function [result] = pow( t, n )
    if n < 0
        result = 1.0/pow( t, -n );
    elseif n == 0
        result = 1.0;
    elseif n == 1
        result = t;
    else
        result = pow( t, n/2 );
        result = result*result;

        if (n % 2) == 1
            result = result*t;
        end
    end
end
```

In this case, evaluating a polynomial of degree n would have a run-time of $\Theta(n \ln(n))$ but would also require $\Theta(\ln(n))$ additional memory due to the size of the call stack.

Getting there

Suppose, instead, we used:

```
function [result] = polyval( coeffs, t )
    result = 0;
    term = 1.0;
    n = length(coeffs);

    for k = 1:length(coeffs)
        result = result + coeffs(n - k + 1)*term;
        term = term*t;
    end
end
```

The run time is now $\Theta(n)$, but there are still many unnecessary calculations, for we have $2n$ multiplications. Fortunately, the memory required is only $\Theta(1)$.

Horner's rule

Suppose, instead, we used Horner's rule for evaluating polynomials:

$$\begin{aligned} p(t) &= a_n t^n + a_{n-1} t^{n-1} + \cdots + a_2 t^2 + a_1 t + a_0 \\ &= \left(a_n t^{n-1} + a_{n-1} t^{n-2} + \cdots + a_2 t + a_1 \right) t + a_0 \\ &= \left(\left(a_n t^{n-2} + a_{n-1} t^{n-3} + \cdots + a_2 \right) t + a_1 \right) t + a_0 \\ &\vdots \\ &= \left(\left(\left(\left(a_n t + a_{n-1} \right) t + \cdots \right) t + a_2 \right) t + a_1 \right) t + a_0 \end{aligned}$$

For example, the polynomial $3.2t^3 + 4.7t^2 - 1.9t + 8.0$ can be written as $(3.2t + 4.7)t - 1.9)t + 8.0$.

From a descriptive point-of-view, this looks ugly; however, from an implementation point-of-view it is elegant:

```
% Let n = length( coeffs )
% The polynomial is therefore:
%                n-1               n-2
%    coeffs[1]*t    + coeffs[2] t    + ... + coeffs[n-1] t + coeffs[n]

function [result] = polyval( coeffs, t )
    result = 0.0;

    for k = 1:length(coeffs)
        result = result*t + coeffs(k);
    end
end
```

The run time is now $\Theta(n)$ and there are only n multiplications and n additions.

Practice questions

1. Evaluate the polynomial $x^2 - 3x + 1$ at the points $x = 0.1$ and $x = -0.1$ using Horner's rule.

5.1.2 Numerical stability

Second, how do we evaluate the polynomial

$$p(t) = (a_2 t + a_1)t + a_0$$

so as to avoid numerical instability? For example, a_2t^2 is large relative to a_1t, but is comparable to $-a_0$, then unfortunately, the following may occur if we are using four significant digits:

> Suppose t = 1000. and a_2 = 1 but a_1 = 0.1, so $a_2t + a_1$= 1000. Suppose now that a_0 = –1000000. In this case, our polynomial, evaluates to 0.000; however, the correct answer is 100, so the result is not so accurate.

The solution to this is to find the coefficients relative to a point close to where we intend to evaluate our polynomial. For example, suppose that we were evaluating our polynomial in the vicinity of t = 1000.0. In this case, we could rewrite the polynomial as

$$p(t) = (\tilde{a}_2 (t - 1000.) + \tilde{a}_1)(t - 1000.) + \tilde{a}_0 .$$

In this case, we can rewrite this as

$$p(t) = (1.000(t - 1000.) + 2000.)(t - 1000.) + 100.$$

This new polynomial, when evaluated at t = 1000. does indeed evaluate to 100.0, but not only that, it does reasonably well for all values around t = 1000. Now Horner's rule may be written as:

```
function [result] = polyval( coeffs, t, t0 )
    result = 0.0;
    dt = t - t0;

    for k = 1:length(coeffs)
        result = result*dt + coeffs(k);
    end
end
```

Note that if we are evaluating results close to the value of t_0, then

$$p(t) = (\tilde{a}_2 (t - t_0) + \tilde{a}_1)(t - t_0) + \tilde{a}_0 ,$$

then $t - t_0$ is small, so we are always multiplying the previous result by a small value and only then adding on the next coefficient. Thus, assuming that $t - t_0$ is small relative to the coefficients, this will often be quite reasonable.

In general, rewriting a polynomial in the form where we are multiplying by t to one where we multiply by $t - t_0$ is nothing more than solving a system of linear equations. This is not the most difficult task, so it can be done quite simply.

Instead, more likely, we are going to be asked: given a set of points (t_0, y_0), (t_1, y_1), (t_2, y_2), …, find a polynomial that passes through these points.

Practice questions

1. Evaluate the polynomial $(x - 5)^2 - 3(x - 5) + 1$ at the points x = 5.1 and x = 4.9 using Horner's rule.

5.1.3 Differentiation

We will look at a number of means of approximating the derivative, including:

1. a simple centered divided difference formula,
2. a backward divided difference formula, and
3. using interpolating polynomials to get better approximations.

Centered divided difference

Suppose we want to evaluate the derivative of a function *f* at a point *x*. One way of doing this is to sample the function as follows:

$$f^{(1)}(x) \approx \frac{f(x+h) - f(x-h)}{2h}.$$

This is based on the limit

$$f^{(1)}(x) = \lim_{h \to 0} \frac{f(x+h) - f(x-h)}{2h},$$

so if *h* is small enough, this should be a reasonable approximation.

Alternatively, suppose that we have a signal *y* at some time *t*, so we don't actually know what *y* is in the future; that is, at time *t* + *h*. In this case, we must use the approximation

$$y^{(1)}(t) = \lim_{h \to 0} \frac{y(t) - y(t-h)}{h},$$

and so our approximation is

$$y^{(1)}(t) \approx \frac{y(t) - y(t-h)}{h}.$$

Now, how good is each of these approximations? Using Taylor series, we have

$$f(x+h) = f(x) + f^{(1)}(x)h + \frac{1}{2} f^{(2)}(x)h^2 + \frac{1}{6} f^{(3)}(\xi_+)h^3$$
$$f(x-h) = f(x) - f^{(1)}(x)h + \frac{1}{2} f^{(2)}(x)h^2 - \frac{1}{6} f^{(3)}(\xi_-)h^3$$

Subtracting these two, we have

$$f(x+h) - f(x-h) = 2f^{(1)}(x)h + \frac{1}{6}\left(f^{(3)}(\xi_+) + f^{(3)}(\xi_-)\right)h^3.$$

Dividing both sides by 2*h*, we get

$$f^{(1)}(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{1}{12}\left(f^{(3)}(\xi_+) + f^{(3)}(\xi_-)\right)h^2.$$

Because $x-h<\xi_{-}<x$ and $x<\xi_{+}<x+h$, by the intermediate value theorem, there must be a point $x-h<\xi<x+h$ such that $\frac{1}{2}\left(f^{(3)}\left(\xi_{+}\right)+f^{(3)}\left(\xi_{-}\right)\right)=f^{(3)}\left(\xi\right)$, so we will simply write

Now, assuming that the third derivative is not too wild, then this is approximately

$$f^{(1)}\left(x\right)=\frac{f\left(x+h\right)-f\left(x-h\right)}{2h}-\frac{1}{6}f^{(3)}\left(\xi\right)h^{2}$$

for some value $x-h<\xi<x+h$.

Note that if we reduce *h* by half, the error drops by approximately one quarter. If we reduce *h* by a factor of 10, the error drops by a factor of 100.

We can see this with this example:

$$\sin^{(1)}\left(1\right)\approx\frac{\sin\left(1+0.01\right)-\sin\left(1-0.01\right)}{0.02}=0.5402933008747350,$$

and subtracting the approximation from the exact solution, we get 0.000009005. Now we have to find $-\frac{1}{6}f^{(3)}\left(\xi\right)0.01^{2}$. We don't know the exact value of ξ, but the cosine function on that interval is monotonic, so we need only look at the endpoints and ensure that our error appears in that interval:

$$-\frac{1}{6}f^{(3)}\left(0.99\right)0.01^{2}\approx0.000009145\text{ and }-\frac{1}{6}f^{(3)}\left(1.01\right)0.01^{2}\approx0.000008864$$

and the error does fall between these two values.

We also see that

$$\sin^{(1)}\left(1\right)\approx\frac{\sin\left(1+0.001\right)-\sin\left(1-0.001\right)}{0.002}=0.5403022158177500$$

and subtracting the approximation from the exact solution, we get 0.00000009005. Thus, you can see that by dividing *h* by 10, the error drops by a factor of 100. Additionally, the two end-points of the error bounds are

$$-\frac{1}{6}f^{(3)}\left(0.999\right)0.001^{2}\approx0.00000009019\text{ and }-\frac{1}{6}f^{(3)}\left(1.001\right)0.001^{2}\approx0.00000008991.$$

Now, this is important: while we see here that yes, the estimate of the error is very successful, we will not know this in general. However, what is critical here is that the formula does work.

Practice questions

1. Estimate the derivative of $f(x) = xe^{x}$ at $x = -2$ using $h = 0.1$ and then $h = 0.01$ using this formula. Demonstrate that the estimator of the error is reasonable by estimating the average value of the appropriate derivative by evaluating that derivative at $x = -1.9$ and $x = -2.1$ and again at $x = -1.99$ and $x = -2.01$.

2. Show that the error of $f^{(1)}(x) \approx \frac{f(x+h)-f(x-h)}{2h}$ is O(h^2).

3. You are participating in a project where you are attempting to detect the source of pollution along a river. Your sensors are placed 0.25 km apart, and the most recent reading give concentrations of HCO_3 at three of the sensors as being 82.9 mg/L, 109.5 mg/L and 97.2 mg/L, with the last reading being the furthest upstream. What is an estimate as to the gradient of the concentration (mg/L/m) at the middle sensor in the downstream direction? (–0.0172 mg/L/m)

Backward divided difference

On the other hand, the one-sided derivative is not so successful: we have the Taylor series

$$y(t-h) = y(t) - y^{(1)}(t)h + \frac{1}{2}y^{(2)}(\tau_-)h^2$$

and solving this for the derivative yields

$$y^{(1)}(t) = \frac{y(t)-y(t-h)}{h} + \frac{1}{2}y^{(2)}(\tau_-)h\,.$$

This is an interesting contrast to the previous: now the error depends on the average value of the second derivatives multiplied by *h*: divide *h* by 2 and the error drops by 2, divide *h* by 10, the error drops by 10:

$$\sin^{(1)}(1) \approx \frac{\sin(1)-\sin(1-0.01)}{0.01} = 0.5445006207375990\,,$$

which, if we subtract this from the exact answer, has an error of 0.004198, whereas

$$\sin^{(1)}(1) \approx \frac{\sin(1)-\sin(1-0.001)}{0.001} = 0.5407229512751026$$

which, if we subtract this from the exact answer, has an error of 0.0004206. Thus, you can see that by dividing *h* by 10, the error drops by a factor of 10. Now, to see that the error fall within the expected intervals, we have

$$\frac{1}{2}f^{(2)}(0.99)0.01 \approx 0.004180 \text{ and } \frac{1}{2}f^{(2)}(1)0.01 \approx 0.004207$$

and

$$\frac{1}{2}f^{(2)}(0.999)0.001 \approx 0.0004205 \text{ and } \frac{1}{2}f^{(2)}(1)0.001 \approx 0.0004207\,.$$

Consequently, the first formula is better than the second; however, the first only works if we have information about the system both before and after the point in question: something we simply do not have in a real-time system.

Implementation

Suppose you have an embedded system where periodically, a sensor returns a reading and you may sporadically need an approximation of the derivative. This means you would require a data structure that can estimate the rate of change of the given data. Such a data structure would only have to store the last three entries, so an array of capacity three would be appropriate. In calculating the derivative, the formula also requires the value between the samples, or the Δt. We could pass this value with the constructor.

As for the required member functions:

1. The interrupt handling routine would have to add the next datum to this data structure, so we would require some sort of function that would pass that new datum. We will use a *circular array* to ensure that each insertion is $\Theta(1)$.
2. We will need a function that calculates the rate of change based on the given data.

```
class Data_collation {
    private:
        double delta_t;
        double buffer[3];
        size_t last;

    public:
        Data_collection( double dt );
        void store_datum( double x );
        double derivative() const;
};

Data_collection::Data_collection( double dt ):
delta_t{dt},
buffer{0, 0, 0},
last{0} {
    // Empty constructor
}

void Data_collection::store_datum( double x ) {
    last = (last + 1) % 3;
    buffer[last] = x;
}

//        3 y(t) - 4 y(t - h) + y(t – 2h)
// Return -------------------------------
//                       2 h

double Data_collection::derivative() const {
    return (
        3.0*buffer[last] - 4.0*buffer[(last + 2) % 3] + buffer[(last + 1) % 3]
    )/(2.0*delta_t);
}
```

Practice questions

1. Estimate the derivative of $y(t) = te^t$ at $t = -2$ using $h = 0.1$ and then $h = 0.01$ using this formula. Demonstrate that the estimator of the error is reasonable by estimating the average value of the appropriate derivative by evaluating that derivative at $x = -2$ and $x = -2.1$ and again at $x = -2$ and $x = -2.01$.

2. Show that the error of $y^{(1)}(t) \approx \frac{y(t) - y(t-h)}{h}$ is $O(h)$.

3. You want to estimate the derivative at time t but only by using historic data (at time $t - h$, $t - 2h$, etc.). Thus, you could use $y^{(1)}(t) \approx \frac{y(t-h) - y(t-2h)}{h}$. How does the size of the coefficient of the error term differ from the previous approximation?

4. You have a sensor that probes a voltage at a rate of 10 Hz. Its last three readings are 7.325 V, 7.331 V and 7.339 V, with the last being the most recent. Estimate the rate of change of voltage at the current time. (0.08 V/s)

5. Based on the result, what is the best estimator of the voltage in the next half second?
(7.339 V + 0.08 V/s · 0.5 s = 7.397 V)

Using interpolating polynomials

Instead, let's look at a different way of finding these formulas using polynomial interpolation.

If we interpolate three points $(x_0 - h, f_{-1})$, (x_0, f_0) and $(x_0 + h, f_1)$ we get a polynomial in x of degree two, if we differentiate this with respect to x, we get a polynomial of degree 1, and if we evaluate this at the point x_0, we get the formula

$$\frac{f_1 - f_{-1}}{2h},$$

which is the same formula we previously saw. Similarly, if we interpolate the two points $(t_0 - h, y_{-1})$ and (t_0, y_0), differentiate the interpolating polynomial at t, and evaluate the resulting linear polynomial $t = t_0$, we get the formula

$$\frac{y_0 - y_{-1}}{h}.$$

This is the formula from the approximation of the derivative.

Suppose now we find the interpolating polynomial of the three points two points $(t_0 - 2h, y_{-2})$, $(t_0 - h, y_{-1})$ and (t_0, y_0), differentiate the interpolating polynomial at t, and evaluate the resulting linear polynomial $t = t_0$, we get the formula

$$\frac{3y_0 - 4y_{-1} + y_{-2}}{2h}.$$

If you wish to see this, the following Maple code finds the interpolating polynomial, differentiates it, and then evaluates that derivative at the point t_0:

```
> curve := CurveFitting:-PolynomialInterpolation( [t[0], t[0] - h, t[0] - 2*h],
                                                  [y[0], y[-1],    y[-2]    ], tau );
```

$$curve := \frac{1}{2}\frac{\left(y_{-2}-2y_{-1}+y_0\right)\tau^2}{h^2} + \frac{1}{2}\frac{\left(hy_{-2}-4hy_{-1}+3hy_0-2t_0y_{-2}+4t_0y_{-1}-2t_0y_0\right)\tau}{h^2} + \frac{1}{2}\frac{2h^2y_0-ht_0y_{-2}+4ht_0y_{-1}-3ht_0y_0+t_0^2y_{-2}-2t_0^2y_{-1}+t_0^2y_0}{h^2}$$

```
> dcurve := diff( curve, tau );  # Differentiate the curve w.r.t. tau
```

$$dcurve := \frac{\left(y_{-2}-2y_{-1}+y_0\right)\tau}{h^2} + \frac{1}{2}\frac{hy_{-2}-4hy_{-1}+3hy_0-2t_0y_{-2}+4t_0y_{-1}-2t_0y_0}{h^2}$$

```
> eval( dcurve, tau = t[0] );    # Evaluate that derivative at t[0]
```

$$\frac{\left(y_{-2}-2y_{-1}+y_0\right)t_0}{h^2} + \frac{1}{2}\frac{hy_{-2}-4hy_{-1}+3hy_0-2t_0y_{-2}+4t_0y_{-1}-2t_0y_0}{h^2}$$

```
> simplify( %, size );           # Simplify the previous result making it compact
```

$$\frac{1}{2}\frac{y_{-2}-4y_{-1}+3y_0}{h}$$

That is,

$$y^{(1)}(t) \approx \frac{3y(t) - 4y(t-h) + y(t-2h)}{2h}.$$

Question: what is the error? In this equation, we have $y(t - h)$ and $y(t - 2h)$, so let us write these two Taylor series down:

$$\begin{aligned} y(t-h) &= y(t) - y^{(1)}(t)h \quad + \frac{1}{2}y^{(2)}(t)h^2 \quad - \frac{1}{6}y^{(3)}(\tau_{-1})h^3 \\ y(t-2h) &= y(t) - y^{(1)}(t)(2h) + \frac{1}{2}y^{(2)}(t)(2h)^2 - \frac{1}{6}y^{(3)}(\tau_{-2})(2h)^3 \\ &= y(t) - 2y^{(1)}(t)h \quad + 2y^{(2)}(t)h^2 \quad - \frac{4}{3}y^{(3)}(\tau_{-2})h^3 \end{aligned}$$

Now, we will multiply the first by four and subtract from that the second to get

$$4y(t-h) - y(t-2h) = 3y(t) - 2y^{(1)}(t)h - \frac{4}{6}y^{(3)}(\tau_{-1})h^3 + \frac{4}{3}y^{(3)}(\tau_{-2})h^3.$$

Now, if we look at $-\frac{4}{6}y^{(3)}(\tau_{-1}) + \frac{4}{3}y^{(3)}(\tau_{-2})$, we note that there must be some value $t - 2h < \tau < t$ such that $-\frac{4}{6}y^{(3)}(\tau_{-1}) + \frac{4}{3}y^{(3)}(\tau_{-2}) = \frac{2}{3}y^{(3)}(\tau)$ and dividing $\frac{2}{3}y^{(3)}(\tau)h^3$ by $2h$ allows us to rewriting this equation and get the simplified form

$$y^{(1)}(t) = \frac{3y(t) - 4y(t-h) + y(t-2h)}{2h} + \frac{1}{3}y^{(3)}(\tau)h^2$$

where $t - 2h < \tau < t$. Compare the errors:

$$f^{(1)}(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{1}{6}f^{(3)}(\xi)h^2$$

where $x - h < \xi < x + h$.

Thus, the *centered* approximation is twice as accurate as our *backward* approximation.

To demonstrate this, let us again take our example of

$$\sin^{(1)}(1) \approx \frac{3\sin(1) - 4\sin(1-0.01) + \sin(1-0.02)}{0.02} = 0.5403205256788960,$$

which, if we subtract this from the exact answer, has an error of –0.00001822, whereas

$$\sin^{(1)}(1) \approx \frac{3\sin(1) - 4\sin(1-0.001) + \sin(1-0.002)}{0.002} = 0.5403024861792130$$

which, if we subtract this from the exact answer, has an error of –0.0000001803.

Notice again that our approximations of the errors are indeed very close:

$$\frac{1}{3}f^{(3)}(0.98)0.01 \approx -0.001857 \text{ and } \frac{1}{3}f^{(3)}(1)0.01 \approx -0.00001801$$

and

$$\frac{1}{3}f^{(3)}(0.998)0.001 \approx -0.0000001807 \text{ and } \frac{1}{3}f^{(3)}(1)0.001 \approx -0.0000001801.$$

Practice questions

1. Estimate the derivative of $y(t) = te^t$ at $t = -2$ using $h = 0.1$ and then $h = 0.01$ using this formula. Demonstrate that the estimator of the error is reasonable by estimating the average value of the appropriate derivative by evaluating that derivative at $x = -2$ and $x = -2.2$ and again at $x = -2$ and $x = -2.02$.

2. Show that the error of $y^{(1)}(t) \approx \dfrac{3y(t) - 4y(t-h) + y(t-2h)}{2h}$ is $O(h^2)$.

3. You want to estimate the derivative at time t but only by using historic data (at time $t - h$, $t - 2h$, etc.). Thus, you could use $y^{(1)}(t) \approx \dfrac{3y(t-h) - 4y(t-2h) + y(t-3h)}{2h}$. Show that now the error is $O(h)$.

4. Disappointed with the $O(h)$ error in the previous question, you find the interpolating polynomial of the points $(t - 3h, y(t - 3h))$, $(t - 2h, y(t - 2h))$ and $(t - h, y(t - h))$. You differentiate this interpolating polynomial and evaluate it at t to get the formula $y^{(1)}(t) \approx \dfrac{5y(t-h) - 8y(t-2h) + 3y(t-3h)}{2h}$. Show that now the error is $O(h^2)$.

5. You experiment and find that the formula $y^{(1)}(t) \approx \dfrac{8y(t) - 9y(t-h) + y(t-3h)}{6h}$ has an error equal to $O(h^2)$. Prove this. Be sure to note the $y(t - \mathbf{3}h)$.

4. You have a sensor that probes a voltage at a rate of 10 Hz. Its last three readings are 7.325 V, 7.331 V and 7.339 V, with the last being the most recent. Estimate the rate of change of voltage at the current time. (0.09 V/s)

5. Based on the result, what is the best estimator of the voltage in the next half second?
(7.339 V + 0.08 V/s · 0.5 s = 7.384 V)

Higher derivatives

We will look at both centered divided difference and backward divided difference approximations, in both cases, using the same approach: finding interpolating polynomials, differentiating those twice, and then evaluating them at the appropriate point.

Centered divided-difference approximations

Suppose we want to estimate a higher derivative. Again, if we find the interpolating quadratic passing through the three points $x - h$, x and $x + h$, we get the approximation

$$f^{(2)}(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$

whereas if we find the interpolating quadratic passing through the three points t, $t - h$ and $t - 2h$, we get the approximation

$$y^{(2)}(t) \approx \frac{y(t) - 2y(t-h) + y(t-2h)}{h^2}.$$

Note that these are the same formulas, only the first is centered while the second is not. Using Taylor series, we can once again estimate the error.

$$f(x+h) = f(x) + f^{(1)}(x)h + \frac{1}{2}f^{(2)}(x)h^2 + \frac{1}{6}f^{(3)}(x)h^3 + \frac{1}{24}f^{(4)}(\xi_+)h^4$$
$$f(x-h) = f(x) - f^{(1)}(x)h + \frac{1}{2}f^{(2)}(x)h^2 - \frac{1}{6}f^{(3)}(x)h^3 + \frac{1}{24}f^{(4)}(\xi_-)h^4$$

Adding these two, we get

$$f(x+h) + f(x-h) = 2f(x) + f^{(2)}(x)h^2 + \frac{1}{24}f^{(4)}(\xi_+)h^4 + \frac{1}{24}f^{(4)}(\xi_-)h^4$$

and solving for the second derivative, we have

$$f^{(2)}(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} - \frac{1}{12}f^{(4)}(\xi)h^2$$

where $x - h < \xi < x + h$.

To demonstrate this, let us again take our example of

$$\sin^{(2)}(1) \approx \frac{\sin(0.99) - 2\sin(1) + \sin(1.01)}{0.01^2} = -0.8414639725730639,$$

which, if we subtract this from the exact answer, has an error of –0.000007012, whereas

$$\sin^{(2)}(1) \approx \frac{\sin(0.999) - 2\sin(1) + \sin(1.001)}{0.001^2} = -0.8414709146853168$$

which, if we subtract this from the exact answer, has an error of –0.00000007012.

Notice again that our approximations of the errors are indeed very close:

$$\frac{1}{12} f^{(4)}(0.99)0.01^2 \approx -0.000006967 \text{ and } \frac{1}{12} f^{(3)}(1.01)0.01^2 \approx -0.000007057$$

and

$$\frac{1}{12} f^{(4)}(0.999)0.001^2 \approx -0.00000007008 \text{ and } \frac{1}{12} f^{(3)}(1.001)0.001^2 \approx -0.00000007017 .$$

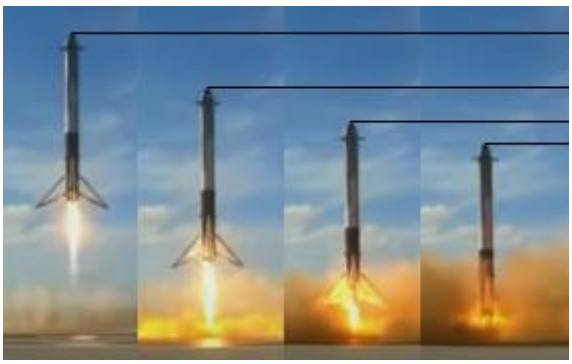
Practice questions

1. Estimate the second derivative of $y(t) = te^t$ at $t = -2$ using $h = 0.1$ and then $h = 0.01$ using this formula. Demonstrate that the estimator of the error is reasonable by estimating the average value of the appropriate derivative by evaluating that derivative at $x = -1.9$ and $x = -2.1$ and again at $x = -1.99$ and $x = -2.01$.

2. Show that the error of $f^{(2)}(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$ is $O(h^2)$.

3. Suppose that the concentration of a gas along a tube are 0.53 μg/L, 0.52 μg/L , 0.57 μg/L and 0.62 μg/L. Each of the readings are taken 10 cm apart. Calculate the concentration pressure at the two interior points (μg/L/m^2). The local rate of change in the concentration is proportional to this concentration pressure. Do you expect the concentration at each of the interior points to increase, stay the same, or decrease in the next few seconds?
(6 μg/L/m^2 and 0 μg/L/m^2, and thus the concentration at the first interior point should increase, while the concentration at the second should stay approximately unchanged.)

4. Here are four photographs of the Falcon Heavy booster rockets landing taken from the YouTube channel of SpaceX. The photographs are taken one second apart. Assuming the booster rockets are 45 m tall, estimate the vertical speed and acceleration of the booster in the two middle frames.



(This author estimates the vertical speed and acceleration in the second frame as –11.9 m/s and –5.2 m/s^2, respectively; while in the third frame as –7.7 m/s and –3.3 m/s^2, respectively.)

Backward divided-difference approximations

Similarly, but considering $t - h$ and $t - 2h$, we have:

$$y(t-h) = y(t) - y^{(1)}(t)h \quad + \frac{1}{2}y^{(2)}(t)h^2 \quad - \frac{1}{6}y^{(3)}(\tau_{-1})h^3$$
$$y(t-2h) = y(t) - y^{(1)}(t)2h + \frac{1}{2}y^{(2)}(t)4h^2 - \frac{1}{6}y^{(3)}(\tau_{-2})8h^3$$

which now becomes

$$y(t-h) = y(t) - \ y^{(1)}(t)h + \frac{1}{2}y^{(2)}(t)h^2 - \frac{1}{6}y^{(3)}(\tau_{-1})h^3$$
$$y(t-2h) = y(t) - 2y^{(1)}(t)h + 2y^{(2)}(t)h^2 - \frac{4}{3}y^{(3)}(\tau_{-2})h^3$$

Now we must be more careful: we want the formula for the second derivative, so we will subtract twice the first from the second:

$$y(t-2h) - 2y(t-h) = -y(t) + y^{(2)}(t)h^2 - \frac{4}{3}y^{(3)}(\tau_{-2})h^3 + \frac{1}{3}y^{(3)}(\tau_{-1})h^3.$$

Solving for the second derivative and estimating with the average value of the third derivative in the vicinity of t, we have

$$y^{(2)}(t) = \frac{y(t) - 2y(t-h) + y(t-2h)}{h^2} + y^{(3)}(\tau)h.$$

Thus, we see that this formula has an error that is $O(h^2)$ for the approximating the second derivative at the center, but only $O(h)$ for approximating the derivative at either end-point.

To demonstrate this, let us again take our example of

$$\sin^{(2)}(1) \approx \frac{\sin(0.98) - 2\sin(0.99) + \sin(1)}{0.01^2} = -0.8360190117405884,$$

which, if we subtract this from the exact answer, has an error of –0.005452, whereas

$$\sin^{(2)}(1) \approx \frac{\sin(0.999) - 2\sin(1) + \sin(1.001)}{0.001^2} = -0.8409301917791019$$

which, if we subtract this from the exact answer, has an error of –0.0005408.

Notice again that our approximations of the errors are indeed very close:

$$f^{(3)}(0.99)0.01 \approx -0.005487 \text{ and } f^{(3)}(1.01)0.01 \approx -0.005319$$

and

$$f^{(3)}(0.999)0.001 \approx -0.0005411 \text{ and } f^{(3)}(1.001)0.001 \approx -0.0005395.$$

Thus, despite the fact the formula is the same, only shifted, the error is significantly higher.

As an example of finding a cubic interpolating polynomial (interpolating the points t, $t - h$, $t - 2h$ and $t - 3h$), if we find such a polynomial, differentiate it, and evaluate at it at the point t, we get the approximation

$$y^{(2)}(t) = \frac{2y(t) - 5y(t-h) + 4y(t-2h) - y(t-3h)}{h^2} + \frac{11}{12}y^{(4)}(\tau)h^2.$$

Implementation

We need a good approximation of the second derivative, but this requires four taps and not just three. Consequently, we must change our buffer capacity of 3. As you may have noted, the buffer capacity was hard-coded. This is described as a *magic number*, as it is not entirely obvious where the "3" comes from. Instead, we will now assign to the constant member variable `BUFFER_CAPACITY` the capacity of the internal buffer (now modified to 4), update the other functions, and then add a function for calculating the second derivative.

```
class Data_collation {
    private:
        size_t const BUFFER_CAPACITY{4};
        double delta_t;
        double buffer[BUFFER_CAPACITY];
        size_t last;

    public:
        Data_collection( double dt );
        void store_datum( double x );
        double derivative() const;
        double second_derivative() const;
};

Data_collection::Data_collection( double dt ):
delta_t{dt},
buffer{},
last{0} {
```

```
    // Empty constructor
}

void Data_collection::store_datum( double x ) {
    last = (last + 1) % BUFFER_CAPACITY;
    buffer[last] = x;
}

//        3 y(t) - 4 y(t - h) + y(t – 2h)
// Return -------------------------------
//                       2 h
double Data_collection::derivative() const {
    size_t const last1{(last + (BUFFER_CAPACITY - 1)) % BUFFER_CAPACITY};
    size_t const last2{(last + (BUFFER_CAPACITY - 2)) % BUFFER_CAPACITY};

    return (3.0*buffer[last] - 4.0*buffer[last1] + buffer[last2])/(2.0*delta_t);
}

//        2 y(t) - 5 y(t - h) + 4y(t – 2h) – y(t - 3h)
// Return -------------------------------------------
//                               2
//                              h
double Data_collection::second_derivative() const {
    size_t const last1{(last + (BUFFER_CAPACITY - 1)) % BUFFER_CAPACITY};
    size_t const last2{(last + (BUFFER_CAPACITY - 2)) % BUFFER_CAPACITY};
    size_t const last3{(last + (BUFFER_CAPACITY - 3)) % BUFFER_CAPACITY};

    return (
        2.0*buffer[last] - 5.0*buffer[last1] + 4.0*buffer[last2] - buffer[last3]
    )/(delta_t* delta_t);
}
```

Practice questions

1. Estimate the second derivative of $y(t) = te^t$ at $t = -2$ using $h = 0.1$ and then $h = 0.01$ using this formula. Demonstrate that the estimator of the error is reasonable by estimating the average value of the appropriate derivative by evaluating that derivative at $x = -2$ and $x = -2.2$ and again at $x = -2$ and $x = -2.02$.

2. Show that the error of $y^{(2)}(t) = \dfrac{y(t) - 2y(t-h) + y(t-2h)}{h^2}$ is O(h).

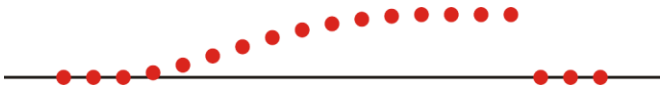
3. You want to estimate the second derivative at time t but only by using historic data (at time $t - h$, $t - 2h$, etc.). Thus, you could use $y^{(2)}(t) = \dfrac{y(t-h) - 2y(t-2h) + y(t-3h)}{h^2}$. How does the error coefficient change?

4. Suppose you have programmed an app to estimate how quickly a runner is both running and accelerating. Your application takes three photographs in rapid succession (0.25 s). It then measures the leading edge of the chest, and based on the height of the individual (a value that must be entered separately), it estimates the relative distances. The distances travelled the second and third photographs relative to the first are 1.17 m and 2.21 m. What is a reasonable estimate of the acceleration of the individual at the moment the last photograph is taken?
(-3.25 m/s^2)

5. Here are three photographs of the Falcon Heavy taken from the YouTube channel of SpaceX. The three photographs are one second apart. Is the acceleration positive or negative, and why? If the Falcon Heavy is 70 m tall, estimate the vertical speed and acceleration of the Falcon Heavy in the third frame.

(This author estimates the vertical speed and acceleration as 28.90 m/s and 3.8 m/s^2, respectively.)

6. While thrusters on launch rockets may be on the order of meganewtons (the Falcon 9 thrusters produce 7.607 MN of force), on a trip to Mars, smaller thrusters will be used to correct the direction of travel, and as such will only produce force on the order of newtons. The force of a thruster can be calculated as a function of the fuel that is injected into the nozzle, and while such a thruster requires a ramp-up time to go to maximum thrust, these engines can be cut off almost instantaneously by cutting off the fuel supply. Suppose that the following are estimates of the thrust (newtons) taken at 10 Hz:



The readings are:

0.0	0.1	0.2	0.3	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8
0	0	0	0.31	0.67	1.02	1.43	1.81	2.13	2.36	2.51	2.54	2.56	3.57	3.57	3.58	3.58	0	0	0

Use the three-value backward divided-difference formulas for both the rate of change and acceleration to determine the first and second derivatives of the force at time $t = 0.8$ and $t = 1.7$. Why is the approximation at the first time reasonable, but entirely erroneous at the second?

5.1.4 Integration

We will look at two approaches to integration:

1. calculation of an integral through polynomial interpolation, and
2. composite applications of the above formulas.

5.1.4.1 Use of polynomial interpolation

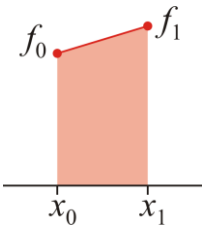
Suppose we have an interval (a, b) on which a function f is defined, and we'd like to approximate the integral

$$\int_a^b f(x)\,\mathrm{d}x\,.$$

Suppose we can only sample this function at a finite number of points in approximating this integral. We could choose the endpoints as our two sample points: $x_0 = a$ and $x_1 = b$ with $f_0 = f(x_0)$ and $f_1 = f(x_1)$ where $h = x_1 - x_0$.

Trapezoidal rule

In this case, given the information we have, the best approximation is to use the trapezoidal rule, which determines the integral by calculating the area of the trapezoid



Thus, the approximation is

$$\int_{x_0}^{x_1} f(x)\,\mathrm{d}x \approx \left(\frac{1}{2} f_0 + \frac{1}{2} f_1\right) h\,.$$

Thus, we are multiplying a weighted average of the value of the function by the width of the interval:

$$f(x_0) = f(x) + f^{(1)}(x)(x_0 - x) + \frac{1}{2} f^{(2)}\left(\xi(x)\right)(x_0 - x)^2\,.$$

Integrating this from a to b yields

$$\int_{x_0}^{x_1} f(x_0)\,\mathrm{d}x = \int_{x_0}^{x_1} f(x)\,\mathrm{d}x + \int_{x_0}^{x_1} f^{(1)}(x)(x_0 - x)\,\mathrm{d}x + \int_{x_0}^{x_1} \frac{1}{2} f^{(2)}\left(\xi(x)\right)(x_0 - x)^2\,\mathrm{d}x\,.$$

Simplifying this, we have:

$$\begin{aligned}
f(x_0)h &= \int_{x_0}^{x_1} f(x)\,\mathrm{d}x + \int_{x_0}^{x_1} f^{(1)}(x)(x_0 - x)\,\mathrm{d}x + \int_{x_0}^{x_1} \frac{1}{2} f^{(2)}(\xi(x))(x_0 - x)^2\,\mathrm{d}x \\
&= \int_{x_0}^{x_1} f(x)\,\mathrm{d}x + a\int_{x_0}^{b} f^{(1)}(x)\,\mathrm{d}x - \int_{x_0}^{x_1} x f^{(1)}(x)\,\mathrm{d}x + \frac{1}{2} f^{(2)}(\xi)\int_{x_0}^{x_1} (x_0 - x)^2\,\mathrm{d}x \\
&= \int_{x_0}^{x_1} f(x)\,\mathrm{d}x + x_0 f(x_1) - x_0 f(x_0) - \left(\left(x f(x) \right|_{x_0}^{b} - \int_{x_0}^{b} f(x)\,\mathrm{d}x \right) + \frac{1}{6} f^{(2)}(\xi) h^3 \\
&= \int_{x_0}^{x_1} f(x)\,\mathrm{d}x + x_0 f(x_1) - x_0 f(x_0) - x_1 f(x_1) + x_0 f(x_0) + \int_{x_0}^{x_1} f(x)\,\mathrm{d}x + \frac{1}{6} f^{(2)}(\xi) h^3
\end{aligned}$$

If we rewrite this, we have

$$h\left(f(x_0) + f(b) \right) = 2\int_{x_0}^{x_1} f(x)\,\mathrm{d}x + \frac{1}{6} f^{(2)}(\xi) h^3$$

Dividing both sides by two yields:

$$\int_{x_0}^{x_1} f(x)\,\mathrm{d}x = \frac{1}{2}\left(f(x_0) + f(x_1) \right) h - \frac{1}{12} f^{(2)}(\xi) h^3 .$$

Let's integrate a function we know:

$$\int_{0.9}^{1.1} \sin(x)\,\mathrm{d}x \approx \frac{1}{2}\left(\sin(0.9) + \sin(1.1) \right)(1.1 - 0.9) = 0.1674534269688919 .$$

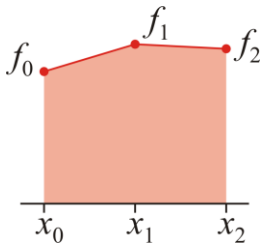
When we subtract this from the exact solution, we get an error of 0.0005604. When we look at the errors

$$-\frac{1}{12}\sin^{(2)}(0.9)0.2^3 = 0.0005222 \text{ and } -\frac{1}{12}\sin^{(2)}(1.1)0.2^3 = 0.0005941,$$

we see that the error does fall between these two values.

Now, with this formula, we cannot reduce *h*, because we want to specifically calculate that integral from x_0 to x_1. In order to reduce the error, we will look at other techniques.

If we wanted to approximate an integral over a longer interval with one additional sample, we can use the trapezoid rule either once or twice:



Using it once, where *h* is the step size between the intervals requires us to calculate

$$\int_{x_0}^{x_2} f(x)\,dx \approx \frac{1}{2}(f_0 + f_2)2h$$
$$= (f_0 + f_2)h$$

while applying it twice allows us to use

$$\int_{x_0}^{x_2} f(x)\,dx \approx \frac{1}{2}(f_0 + f_1)h + \frac{1}{2}(f_1 + f_2)h$$
$$= \frac{1}{2}(f_0 + 2f_1 + f_2)h$$

The error is simply the sum of the errors applied to each sub-interval; after simplification, this becomes:

$$-\frac{1}{12}f^{(2)}(\xi)(2h)^3 = -\frac{2}{3}f^{(2)}(\xi)h^3 \text{ and } -\frac{1}{12}\left(f^{(2)}(\xi_1) + f^{(2)}(\xi_2)\right)h^3 = -\frac{1}{6}f^{(2)}(\xi)h^3.$$

We note the coefficient drops by a factor of four. Let's apply this to the integral above:

$$\int_{0.9}^{1.1} \sin(x)\,dx \approx \frac{1}{2}(\sin(0.9) + 2\sin(1) + \sin(1.1))0.1 = 0.1678738119652356.$$

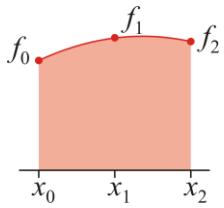
When we subtract this from the exact solution, we get an error of 0.0001400. When we look at the errors

$$-\frac{1}{6}\sin^{(2)}(0.9)0.1^3 = 0.0001306 \text{ and } \frac{1}{6}\sin^{(2)}(1.1)0.1^3 = 0.0001485,$$

we see that the error does fall between these two values.

Simpson's rule

We could do better, however, as this integral almost certain underestimates the actual accumulated signal, for we expect the system to be differentiable. For example, suppose we are periodically sampling the angular speed at which an antenna is rotating. In that case, the integral is the change in position of the antenna, and the antenna has inertia, so its angular velocity cannot change abruptly. Consequently, we could find an interpolating quadratic function and integrate it:



Rather than working this out, This is an appropriate point to demonstrate the power of symbolic computation. Here is an example of Maple code where we:

1. find an interpolating polynomial between points (a, f_0), $(a + h, f_1)$, $(a + 2h, f_2)$ with respect to the variable x,
2. integrate that interpolating polynomial from $x = a$ to $x = a + 2h$, and
3. simplify the resulting expression.

```
> curve :=
CurveFitting:-PolynomialInterpolation( [a, a + h, a + 2*h], [f0, f1, f2], x );
```

$$curve := \frac{1}{2}\frac{(f2 - 2f1 + f0)x^2}{h^2} + \frac{1}{2}\frac{(-2af0 + 4af1 - 2af2 - 3f0h + 4f1h - f2h)x}{h^2} + \frac{1}{2}\frac{a^2f0 - 2a^2f1 + a^2f2 + 3af0h - 4af1h + af2h + 2f0h^2}{h^2}$$

```
> int( curve, x = a..a + 2*h );
```

$$\frac{1}{6}\frac{(f2 - 2f1 + f0)\left((a + 2h)^3 - a^3\right)}{h^2} + \frac{1}{4}\frac{(-2af0 + 4af1 - 2af2 - 3f0h + 4f1h - f2h)\left((a + 2h)^2 - a^2\right)}{h^2} + \frac{a^2f0 - 2a^2f1 + a^2f2 + 3af0h - 4af1h + af2h + 2f0h^2}{h}$$

```
> simplify( %, size );
```

$$\frac{1}{3}h(f0 + 4f1 + f2)$$

Thus, we see the approximation of the integral is

$$\frac{1}{3}\left(f_0 + 4f_1 + f_2\right)h,$$

and this is a weighted average of the values of the polynomial multiplied by the width of the interval.

Beyond the scope of this course, the error for Simpson's rule is

$$-\frac{1}{90} f^{(4)}(\xi) h^5 .$$

Let's apply this to the integral above:

$$\int_{0.9}^{1.1} \sin(x)\,dx \approx \frac{1}{3}\left(\sin(0.9) + 4\sin(1) + \sin(1.1)\right)0.1 = 0.16801394029735 02 .$$

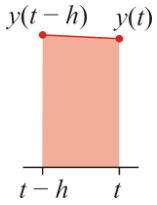
When we subtract this from the exact solution, we get an error of 0.00000009345. When we look at the errors

$$-\frac{1}{90}\sin^{(4)}(0.9)0.1^5 = 0.00000008703 \text{ and } -\frac{1}{90}\sin^{(4)}(1.1)0.1^5 = 0.00000009902 ,$$

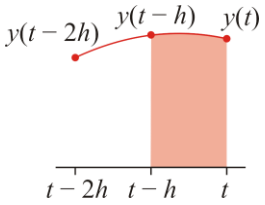
we see that the error does fall between these two values.

Half Simpson's rule (Simpson's rule over one time interval)

Now, suppose you reading data from a sensor and summing that data. For example, to calculate the total energy used, you would integrate a sample of the power usage. On the other hand, to find the total distance travelled, you could integrate the speed. Given the most recent reading, you would like to estimate the integral over the last interval, from $t - h$ to t. The most straight-forward is to use the trapezoidal rule.



The issue here is that it is not a very good approximation, but Simpson's rule assumes you are approximating the integral over two time steps. Suppose you have already approximated the previous integral up to time $t - h$. In this case, Simpson's rule won't help. Suppose, however, you find the interpolating quadratic polynomial over the past two time intervals, but integrated it only over the last time interval, this may indeed give a better approximation of the integral:

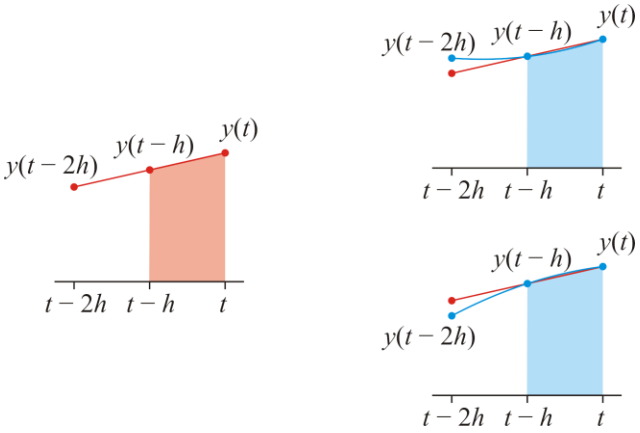


Using Maple, we can find that the approximation of the integral to be

$$\int_{t-h}^{t} y(t)\,dt \approx \frac{5y(t) + 8y(t-h) - y(t-2h)}{12} h .$$

The error will still be on the same magnitude of that of Simpson's rule, but what is interesting here is that this is one of our first approximations that contains a negative weight coefficient in our weighted average: $\frac{5}{12} + \frac{8}{12} - \frac{1}{12} = 1$.

You may wonder why a coefficient would have such a weighted average. This is perhaps best explained with a simple case of looking at the deviations from where the *y* values are in a straight line.



If the point $y(t - 2h)$ increases, the interpolating quadratic polynomial goes down on the right-hand sub-interval, so a positive increase in $y(t - 2h)$ results in a decrease in the estimated integral. Alternatively, if the point $y(t - 2h)$ decreases, the interpolating polynomial increases on the right-hand sub-interval, thus increasing the approximation of the integral.

The error of this formula is not half that of Simpson's rule, but rather, lower order:

$$\int_{t-h}^{t} y(t)\,dt = \frac{5y(t)+8y(t-h)-y(t-2h)}{12}h - \frac{1}{24}y^{(3)}(\tau)h^4.$$

Implementation

Adding to our data collection data structure, we can now estimate the integral; however, this will require keeping track of the integral calculated to this point.

```
class Data_collation {
    private:
        size_t const BUFFER_CAPACITY{4};
        double delta_t;
        double buffer[BUFFER_CAPACITY];
        size_t last;
        double running_sum;
        double partial_sum;
        bool even;

    public:
        Data_collection( double dt );
        void store_datum( double x );
        double derivative() const;
        double second_derivative() const;
        double integral() const;
};

Data_collection::Data_collection( double dt ):
delta_t{dt},
buffer{},
last{0},
running_sum{0.0},
even{true} {
    // Empty constructor
}

void Data_collection::store_datum( double x ) {
    last = (last + 1) % BUFFER_CAPACITY;
    buffer[last] = x;

    // The integral must be calculated regardless as to whether or not the
    // value of the integral is queried.
    //                   5 y(t) + 8 y(t - h) - y(t - 2*h)
    //    running_sum += ------------------------------- h
    //                                   12

    size_t const last1{(last + (BUFFER_CAPACITY - 1)) % BUFFER_CAPACITY};
    size_t const last2{(last + (BUFFER_CAPACITY - 2)) % BUFFER_CAPACITY};

    if
    running_sum += (5*buffer[last] + 8*buffer[last1] - buffer[last2])/12.0*delta_t;
}
```

```
//        3 y(t) - 4 y(t - h) + y(t – 2h)
// Return -------------------------------
//                       2 h
double Data_collection::derivative() const {
    size_t const last1{(last + (BUFFER_CAPACITY - 1)) % BUFFER_CAPACITY};
    size_t const last2{(last + (BUFFER_CAPACITY - 2)) % BUFFER_CAPACITY};

    return (3.0*buffer[last] - 4.0*buffer[last1] + buffer[last2])/(2.0*delta_t);
}

//        2 y(t) - 5 y(t - h) + 4y(t – 2h) – y(t - 3h)
// Return -------------------------------------------
//                               2
//                              h
double Data_collection::second_derivative() const {
    size_t const last1{(last + (BUFFER_CAPACITY - 1)) % BUFFER_CAPACITY};
    size_t const last2{(last + (BUFFER_CAPACITY - 2)) % BUFFER_CAPACITY};
    size_t const last3{(last + (BUFFER_CAPACITY - 3)) % BUFFER_CAPACITY};

    return (
        2.0*buffer[last] - 5.0*buffer[last1] + 4.0*buffer[last2] - buffer[last3]
    )/(delta_t* delta_t);
}

// Return the stored value of the integral
//  - this value will be updated each time a new reading is provided
double Data_collection::integral() const {
    return running_sum;
}
```

Simpson's $^3/_{8^{th}}$ rule

The final approximation we will use is termed Simpson's $^3/_8$th rule, which finds an interpolating cubic and integrates it. We will show this again using Maple:

```
> curve :=
CurveFitting:-PolynomialInterpolation( [a, a + h, a + 2*h, a + 3*h],
                                       [f0, f1, f2, f3], x );
```

$$curve := \frac{1}{6}\,\frac{(f3 - 3f2 + 3f1 - f0)\,x^3}{h^3} + \frac{1}{6}\,\frac{(3af0 - 9af1 + 9af2 - 3af3 + 6f0h - 15f1h + 12f2h - 3f3h)\,x^2}{h^3} + \frac{1}{6}\,\frac{1}{h^3}\left(\left(-3a^2f0 + 9a^2f1 - 9a^2f2 + 3a^2f3 - 12af0h + 30af1h - 24af2h + 6af3h - 11f0h^2 + 18f1h^2 - 9f2h^2 + 2f3h^2\right)x\right) + \frac{1}{6}\,\frac{1}{h^3}\left(a^3f0 - 3a^3f1 + 3a^3f2 - a^3f3 + 6a^2f0h - 15a^2f1h + 12a^2f2h - 3a^2f3h + 11af0h^2 - 18af1h^2 + 9af2h^2 - 2af3h^2 + 6f0h^3\right)$$

```
> int( curve, x = a..a + 3*h );
```

$$\frac{1}{24}\,\frac{(f3 - 3f2 + 3f1 - f0)\left((a + 3h)^4 - a^4\right)}{h^3} + \frac{1}{18}\,\frac{1}{h^3}\left((3af0 - 9af1 + 9af2 - 3af3 + 6f0h - 15f1h + 12f2h - 3f3h)\left((a + 3h)^3 - a^3\right)\right) + \frac{1}{12}\,\frac{1}{h^3}\left(\left(-3a^2f0 + 9a^2f1 - 9a^2f2 + 3a^2f3 - 12af0h + 30af1h - 24af2h + 6af3h - 11f0h^2 + 18f1h^2 - 9f2h^2 + 2f3h^2\right)\left((a + 3h)^2 - a^2\right)\right) + \frac{1}{2}\,\frac{1}{h^2}\left(a^3f0 - 3a^3f1 + 3a^3f2 - a^3f3 + 6a^2f0h - 15a^2f1h + 12a^2f2h - 3a^2f3h + 11af0h^2 - 18af1h^2 + 9af2h^2 - 2af3h^2 + 6f0h^3\right)$$

```
> simplify( %, size );
```

$$\frac{3}{8}\,h\,(f0 + 3f1 + 3f2 + f3)$$

Thus, we see the approximation of the integral is

$$\frac{3}{8}\left(f_0 + 3f_1 + 3f_2 + f_3\right)h,$$

and this is a weighted average of the values of the polynomial multiplied by the width of the interval.

Beyond the scope of this course, the error for Simpson's $^3/_8$th rule is

$$-\frac{3}{80} f^{(4)}(\xi) h^5 .$$

The coefficient is more than three times larger than the error of Simpson's rule assuming we keep the same step size. If we are decreasing the same step size (taking three steps from *a* to *b* instead of two), then the error does reduce slightly.

Let's apply this to the integral above:

$$\int_{0.9}^{1.1} \sin(x)\,dx \approx \frac{3}{8}\left(\sin(0.9) + 3\sin\left(0.9\overline{6}\right) + 3\sin\left(1.0\overline{3}\right) + \sin(1.1)\right)0.1 = 0.1680138468450871 .$$

When we subtract this from the exact solution, we get an error of 0.00000004153, which is very close to 4/9th the error of Simpson's rule (the ratio is 0.44441). When we look at the errors

$$-\frac{3}{80}\sin^{(4)}(0.9)0.1^5 = 0.00000003868 \text{ and } -\frac{3}{80}\sin^{(4)}(1.1)0.1^5 = 0.00000004401,$$

we see that the error does fall between these two values.

A fair comparison of Simpson's and Simpson's $^3/_8$th rule

To fairly compare these, however, we should consider six intervals of width *h*, and consider applying Simpson's rule three times, or Simpson's $^3/_8$th rule twice. In this case, the error of Simpson's rule applied three times would be

$$-\frac{3}{90} f^{(4)}(\xi) h^5$$

while the error of Simpson's $^3/_8$th rule twice would be

$$-\frac{3}{40} f^{(4)}(\xi) h^5 .$$

Thus, the error of Simpson's $^3/_8$th rule is 125% larger than that of Simpson's rule for the same step size. To verify this, note that the quartic function has a 4th derivative that is constant. Thus, if we try to approximate $\int_0^6 x^4 dx = 1555.2$ with $h = 1$, we have the two approximations applying Simpson's rule three times:

$$\frac{1}{3}\left(0^4 + 4\cdot 1^4 + 2^4\right)\cdot 1 + \frac{1}{3}\left(2^4 + 4\cdot 3^4 + 4^4\right)\cdot 1 + \frac{1}{3}\left(4^4 + 4\cdot 5^4 + 6^4\right)\cdot 1$$
$$= \frac{1}{3}\left(0^4 + 4\cdot 1^4 + 2\cdot 2^4 + 4\cdot 3^4 + 2\cdot 4^4 + 4\cdot 5^4 + 6^4\right)\cdot 1 = 1556$$

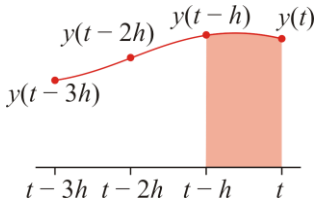
and Simpson's $^3/_8$th rule twice

$$\frac{3}{8}\left(0^4+3\cdot1^4+3\cdot2^4+3^4\right)\cdot1+\frac{3}{8}\left(3^4+3\cdot4^4+3\cdot5^4+6^4\right)\cdot1$$
$$=\frac{3}{8}\left(0^4+3\cdot1^4+3\cdot2^4+2\cdot3^4+3\cdot4^4+3\cdot5^4+6^4\right)\cdot1=1557$$

We note that the error of the first is 0.8, while the error of the second is 1.8, which is exactly 125% larger than the first. Thus,

A third Simpson's $^3/_8^{\text{th}}$ (Simpson's $^3/_8^{\text{th}}$ rule over one time interval)

As with Simpson's rule, we could find the interpolating cubic polynomial over three time intervals, but only integrate that in the most recent interval:



Using Maple, we can find this approximation using

$$\int_{t-h}^{t} y(t)\,\mathrm{d}t \approx \frac{9y(t)+19y(t-h)-5y(t-2h)+y(t-3h)}{24}h\,.$$

With no other information, the error would be a third the error of Simpson's $^3/_8^{\text{th}}$ rule. The error is

$$\int_{t-h}^{t} y(t)\,\mathrm{d}t = \frac{9y(t)+19y(t-h)-5y(t-2h)+y(t-3h)}{24}h-\frac{19}{720}y^{(4)}(\tau)h^5\,,$$

which does have a lower-order error than half Simpson's rule.

Implementation

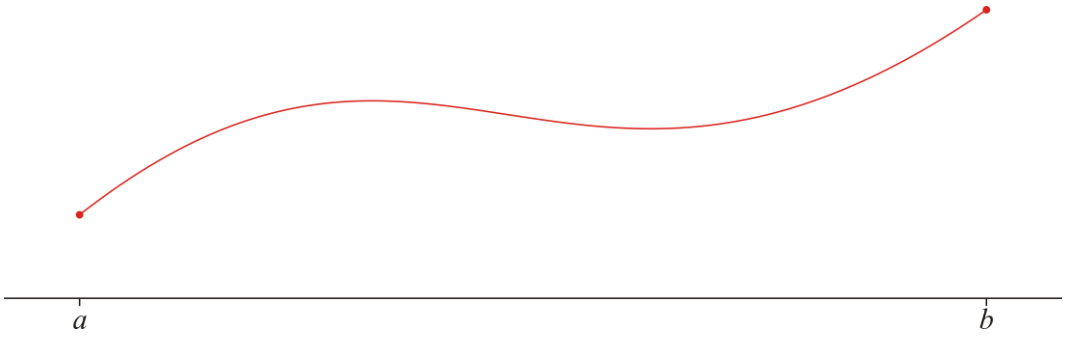
As with tracking the integral using Simpson's rule, we would now update the integral with Simpson's $^{3}/_{8}^{th}$ rule.

```
class Data_collation {
    private:
        size_t const BUFFER_CAPACITY{4};
        double delta_t;
        double buffer[BUFFER_CAPACITY];
        size_t last;
        double running_sum;

    public:
        Data_collection( double dt );
        void store_datum( double x );
        double derivative() const;
        double second_derivative() const;
        double integral() const;
};

Data_collection::Data_collection( double dt ):
delta_t{dt},
buffer{},
last{0},
integral{0.0} {
    // Empty constructor
}

void Data_collection::store_datum( double x ) {
    last = (last + 1) % BUFFER_CAPACITY;
    buffer[last] = x;

    // The integral must be calculated regardless as to whether or not the
    // value of the integral is queried.
    //                   9 y(t) + 19 y(t - h) - 5 y(t - 2*h) + y(t - 3h)
    //    running_sum += ------------------------------------------------ h
    //                                        24

    size_t const last1{(last + (BUFFER_CAPACITY - 1)) % BUFFER_CAPACITY};
    size_t const last2{(last + (BUFFER_CAPACITY - 2)) % BUFFER_CAPACITY};
    size_t const last3{(last + (BUFFER_CAPACITY - 3)) % BUFFER_CAPACITY};
    running_sum += (
        9*buffer[last] + 19*buffer[last1] - 5*buffer[last2] +buffer[last3]
    )/24.0*delta_t;
}
```

As before, the integral function would simply return the `integral` member variable.

5.1.4.2 Composition rules

Unlike differentiation, one cannot *change* the bounds of an integral, for that is exactly what we're trying to approximate. Unfortunately, we cannot continue using higher degree polynomials, as they become more sensitive to noise. Instead, however, we can sub-divide the interval [*a*, *b*] into *n* sub-intervals, and then apply any of the previous rules to each sub-interval and then add them. To demonstrate these techniques, we will show integrating the following generic function from *a* to *b*:



Composite trapezoidal rule

Suppose we divide the interval into *n* sub-intervals, and apply the trapezoidal rule to each. In this case, let us denote $h = \frac{b-a}{n}$ so the interval is divided into the points $a = x_0, x_1, x_2, \ldots, x_n = b$ where $x_k = a + kh$. If we denote $f_k = f(x_k)$, then the approximation is

$$\left(\frac{1}{2}f_0 + \frac{1}{2}f_1\right)h + \left(\frac{1}{2}f_1 + \frac{1}{2}f_2\right)h + \cdots + \left(\frac{1}{2}f_{n-1} + \frac{1}{2}f_n\right)h.$$

Rearranging this, we get

$$\int_a^b f(x)\,\mathrm{d}x \approx \left(\frac{1}{2}f_0 + f_1 + f_2 + \cdots + f_{n-1} + \frac{1}{2}f_n\right)h.$$

Question: what is the error? It must be the sum of the individual errors:

$$-\frac{1}{12}f^{(2)}(\xi_1)h^3 - \frac{1}{12}f^{(2)}(\xi_2)h^3 - \frac{1}{12}f^{(2)}(\xi_3)h^3 - \cdots - \frac{1}{12}f^{(2)}(\xi_n)h^3.$$

Recall, that each sub-interval is of width *h*. Collecting similar terms, we have

$$-\frac{1}{12}\left(f^{(2)}(\xi_1) + f^{(2)}(\xi_2) + f^{(2)}(\xi_3) + \cdots + f^{(2)}(\xi_n)\right)h^3 = -\frac{h^3}{12}\sum_{k=1}^{n} f^{(2)}(\xi_k)$$

where $x_{k-1} < \xi_k < x_k$. The problem is that that while the coefficient $-\frac{h^3}{12}$ may go to zero as *h* gets smaller, simultaneously, *n* becomes larger, and so the sum $\sum_{k=1}^{n} f^{(2)}(\xi_k)$ grows. To contrast the two, we observe that

$$\sum_{k=1}^{n} f^{(2)}(\xi_k)$$

is a sum of points evaluated at *n*, and by assuming the second derivative is continuous, there must be a point $a < \xi < b$ such that it equals the average of these values

$$f^{(2)}(\xi) = \frac{1}{n}\sum_{k=1}^{n} f^{(2)}(\xi_k)$$

so

$$nf^{(2)}(\xi) = \sum_{k=1}^{n} f^{(2)}(\xi_k).$$

Substituting this into our formula, we get

$$-\frac{h^3}{12}\sum_{k=1}^{n} f^{(2)}(\xi_k) = -\frac{h^3 n f^{(2)}(\xi)}{12}$$

and because $h = \frac{b-a}{n}$, it follows that $hn = b-a$, and thus we have

$$\int_a^b f(x)\,\mathrm{d}x = \left(\frac{1}{2}f_0 + \left(\sum_{k=1}^{n-1} f_k\right) + \frac{1}{2}f_n\right)h - f^{(2)}(\xi)\frac{b-a}{12}h^2$$

or

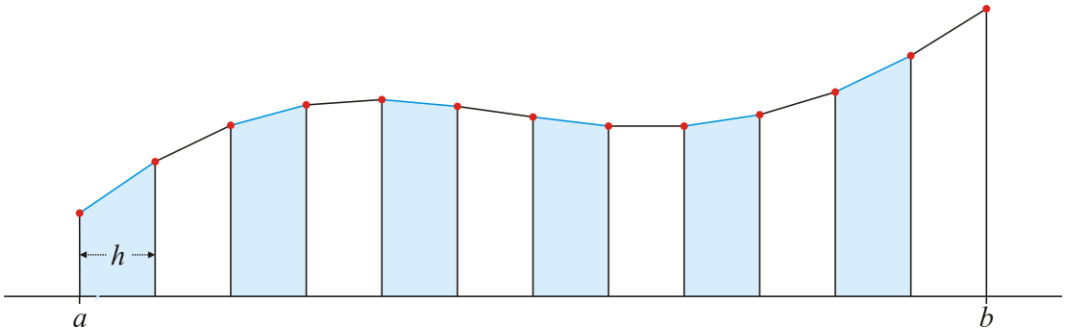
$$\int_a^b f(x)\,\mathrm{d}x = \frac{1}{2}\left(f_0 + 2\left(\sum_{k=1}^{n-1} f_k\right) + f_n\right)h - f^{(2)}(\xi)\frac{b-a}{12}h^2.$$

If you wanted to be explicit with respect to the function evaluations (as $f_k = f(a+kh)$), you could write this as

$$\int_a^b f(x)\,\mathrm{d}x = \frac{1}{2}\left(f(a) + 2\left(\sum_{k=1}^{n-1} f(a+kh)\right) + f(b)\right)h - f^{(2)}(\xi)\frac{b-a}{12}h^2.$$

You may note that $\frac{1}{2}\left(1+\underbrace{2+2+2+\cdots+2+2}_{n-1 \text{ times}}+1\right)h = b-a$, so this is a weighted average of these evaluations of the function on the interval [*a*, *b*] multiplied by the width of the interval.

The components of $-f^{(2)}(\xi)\frac{b-a}{12}$ are either fixed or bounded (assuming the second derivative is not too wild). Thus, halve *h* and the error goes down by a factor of four; increase the number of intervals by ten and the error goes down by a factor of 100.



As an example, let us approximate an integral slightly larger than the one previously given. We know, for example, that the integral $\int_1^3 \sin(x)\,dx = \cos(1) - \cos(3) \approx 1.530294802468585$. The minimum and maximum values of the second derivative of sin(*x*) on that interval are –1 and -0.1411200080598672, respectively. If we let n = 12, in which case h = 1/6, our approximation is

$$\int_1^3 \sin(x)\,dx \approx \frac{1}{2}\left(\sin(1) + 2\left(\sum_{k=1}^{11} \sin\left(1 + \frac{2}{12}k\right)\right) + \sin(3)\right)\left(\frac{2}{12}\right) = 1.526750812326977\,.$$

Subtracting this from the exact solution yields an error of 0.003544. The minimum and maximum values of the approximation of the error are 0.0006533 and 0.004630, respectively, and the error of our approximation falls between these two.

If we now increase n = 120, in which case, h = 1/60, our approximation is

$$\int_1^3 \sin(x)\,dx \approx \frac{1}{2}\left(\sin(1) + 2\left(\sum_{k=1}^{119} \sin\left(1 + \frac{2}{120}k\right)\right) + \sin(3)\right)\left(\frac{2}{120}\right) = 1.530259378813789\,.$$

Subtracting this from the exact solution yields an error of 0.00003542, and as you can see, this is approximately one-hundredth the previous error. Of course, the bounds on the error would not change.

We can now apply this to Simpson's rule and Simpson's $^3/_8{}^{\text{th}}$ rule.

Practice questions

1. Approximate the integral $\int_0^3 x^4\,dx$ using the composite trapezoidal rule with n = 6.

2. Find the exact error of your approximation and find the bounds of the error using the given formulas.

Composite Simpson's rule

We will want to keep the number of function evaluations the same. Thus, we will assume we are dividing the interval into an even number of sub-intervals for Simpson's rule, and into a multiple of three sub-intervals for Simpson's $^3/_8^{\text{th}}$ rule.

In the case of Simpson's rule, we must assume that n is even, and thus each sub-interval to which we will apply Simpson's rule will be of width $2h$:

$$\int_a^b f(x)\,\mathrm{d}x = \frac{1}{6}\left(f_0 + 4f_1 + 2f_2 + 4f_3 + 2f_4 + \cdots + 2f_{n-2} + 4f_{n-1} + f_n \right)(2h) - f^{(4)}(\xi)\frac{b-a}{2880}(2h)^4$$

or

$$\int_a^b f(x)\,\mathrm{d}x = \frac{1}{3}\left(f_0 + 4\sum_{k=1}^{\frac{n}{2}} f_{2k-1} + 2\sum_{k=1}^{\frac{n}{2}-1} f_{2k} + f_n \right)h - f^{(4)}(\xi)\frac{b-a}{180}h^4 .$$

If you wanted to be explicit with respect to the function evaluations (as $f_k = f(a+kh)$), you could write this as

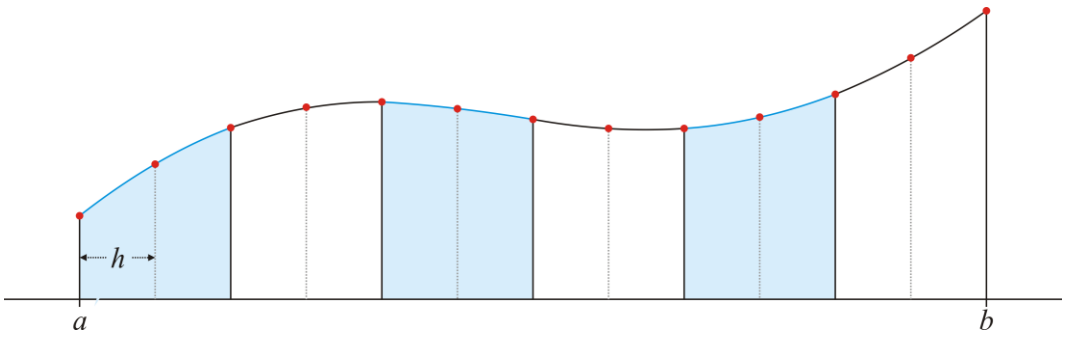
$$\int_a^b f(x)\,\mathrm{d}x = \frac{1}{3}\left(f(a) + 4\left(\sum_{k=1}^{\frac{n}{2}} f\left(a+(2k-1)h\right) \right) + 2\left(\sum_{k=1}^{\frac{n}{2}-1} f(a+2kh) \right) + f(b) \right)h - f^{(4)}(\xi)\frac{b-a}{180}h^4 .$$

What is important, however, is the weights in front of the coefficients:

1, 4, 2, 4, 2, 4, …, 4, 2, 4, 1.

You may note that $\frac{1}{3}\left(1+\underbrace{4+2+4+2+4+\cdots+4+2+4}_{n-1 \text{ times}}+1\right)h = b-a$, so this is a weighted average of these evaluations of the function on the interval $[a, b]$ multiplied by the width of the interval.

Viewing this, we are finding interpolating and integrating interpolating quadratics over intervals of width $2h$:



Approximating our integral of the sine function from 1 to 3, we have that this equals

$$\int_1^3 \sin(x)\,\mathrm{d}x \approx \frac{1}{3}\left(\sin(1) + 4\sum_{k=1}^{\frac{n}{2}} \sin\left(1+\frac{1}{12}(2k-1)\right) + 2\sum_{k=1}^{\frac{n}{2}-1}\left(1+\frac{1}{12}2k\right) + \sin(3) \right)\left(\frac{1}{12}\right) = 1.530301384130549 .$$

Subtracting this from the correct answer yields an error of –0.000006582. The minimum and maximum values of the approximation of the error are –0.000001210 and –0.000008573, respectively, and the error of our approximation falls between these two. If we increase the number of intervals by a factor of 10 to 120, the approximation is now 1.530294803124597 with an error of –0.0000000006560; again, a drop by almost exactly a factor of 10^4.

Practice questions

1. Approximate the integral $\int_0^3 x^4 \mathrm{d}x$ using the composite Simpson's rule with $n = 6$.

2. Find the exact error of your approximation and find the bounds of the error using the given formulas.

Composite Simpson's $^3/_{8}^{th}$ rule

In the case of Simpson's $^3/_8^{\text{th}}$ rule, we must assume n is a multiple of three, and thus each sub-interval will be of width $3h$.

$$\int_a^b f(x)\,\mathrm{d}x = \frac{1}{8}\left(f_0 + 3f_1 + 3f_2 + 2f_3 + 3f_4 + 3f_5 + 2f_6 + \cdots + 2f_{n-3} + 3f_{n-2} + 3f_{n-1} + f_n \right)(3h) - f^{(4)}(\xi)\frac{b-a}{6480}(3h)^4$$

or

$$\int_a^b f(x)\,\mathrm{d}x = \frac{3}{8}\left(f_0 + 3\sum_{k=1}^{\frac{n}{3}} f_{3k-2} + 3\sum_{k=1}^{\frac{n}{3}} f_{3k-1} + 2\sum_{k=1}^{\frac{n}{3}-1} f_{3k} + f_n \right)h - f^{(4)}(\xi)\frac{b-a}{80}h^4.$$

If you wanted to be explicit with respect to the function evaluations (as $f_k = f(a+kh)$), you could write this as

$$\int_a^b f(x)\,\mathrm{d}x = \frac{3}{8}\left(f(a) + 3\left(\sum_{k=1}^{\frac{n}{3}} f\left(a+(3k-2)h\right) \right) + 3\left(\sum_{k=1}^{\frac{n}{3}} f\left(a+(3k-1)h\right) \right) + 2\left(\sum_{k=1}^{\frac{n}{3}-1} f\left(a+3kh\right) \right) + f(b) \right)h - f^{(4)}(\xi)\frac{b-a}{80}h^4$$

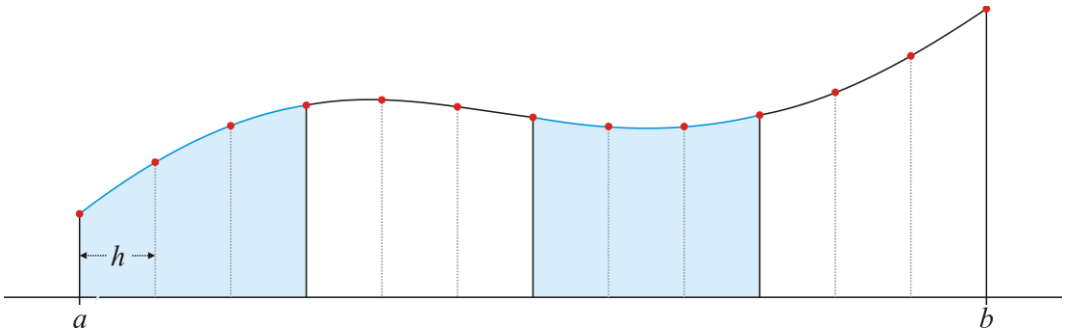
.

What is important, however, is the weights in front of the coefficients:

$$1, 3, 3, 2, 3, 3, 2, \ldots, 2, 3, 3, 1.$$

You may note that $\frac{3}{8}\left(1+\underbrace{3+3+2+3+3+2+\cdots+2+3+3}_{n-1 \text{ times}}+1\right)h = b-a$, so this is a weighted average of these evaluations of the function on the interval $[a, b]$ multiplied by the width of the interval.

Viewing this, we are finding interpolating and integrating interpolating cubic polynomials over intervals of width $3h$:



Again, applying this out our integral, we get an approximation

$$\int_1^3 \sin(x)\,\mathrm{d}x \approx \frac{3}{8}\left(\sin(1) + 3\sum_{k=1}^{\frac{n}{3}} \sin\left(1+\frac{1}{12}(3k-1)\right) + 3\sum_{k=1}^{\frac{n}{3}} \sin\left(1+\frac{1}{12}(3k-2)\right) + 2\sum_{k=1}^{\frac{n}{3}-1}\left(1+\frac{1}{12}3k\right) + \sin(3) \right)\left(\frac{2}{12}\right)$$

$$= 1.530309660494875$$

The error when we subtract this from the actual integral is –0.00001486. Our approximations for the limits of the error span form –0.000002722 to –0.00001929, and we see that this error falls into that interval.

Again, if we increase n by a factor of 10 to 120, we get the approximation 1.530294803944662 which has an error of –0.000000001476 which is smaller by a factor of 10^4 and this again falls between the two end-points of the error estimates.

You will notice that the error is approximately twice that of the approximation using Simpson's rule and that the intervals of possible error do not overlap.

Practice questions

1. Approximate the integral $\int_0^3 x^4 \mathrm{d}x$ using the composite Simpson's rule with $n = 6$.

2. Find the exact error of your approximation and find the bounds of the error using the given formulas.

3. Compare the answers for the composite trapezoidal, Simpson's and Simpson's $^3/_8^{\text{th}}$ rules.

5.1.3.3 Integration rules implemented in C++

For those keen to implement these in C++, consider the following:

```
#include <cassert>
#include <iostream>
#include <cmath>

int main();
double trapezoidal_rule( double f(double),
                         double a, double b,
                         unsigned int n );
double simpsons_rule( double f(double),
                      double a, double b,
                      unsigned int n );
double simpsons_rule( double f[], size_t capacity, double h );
double simpsons_3_8_rule( double f(double),
                          double a, double b,
                          unsigned int n );

double trapezoidal_rule( double f(double),
                         double a, double b,
                         unsigned int n ) {
    assert( n > 0 );
    double h{(b - a)/n};

    double sum{0.0};

    for ( unsigned int k{1}; k < n; ++k ) {
        sum += f( a + k*h );
    }

    return (0.5*h)*(f(a) + 2.0*sum + f(b));
}

double simpsons_rule( double f(double),
                      double a, double b,
                      unsigned int n ) {
    assert( n > 0 );
    assert( (n % 2) == 0 ); // 'n' must be even
    double h{(b - a)/n};

    double sum_2{0.0};
    double sum_4{0.0};

    for ( unsigned int k{1}; k < n/2; ++k ) {
        sum_4 += f( a + (2*k - 1)*h );
        sum_2 += f( a +  2*k     *h );
    }

    return (h/3.0)*(f(a) + 4.0*(sum_4 + f( a + (n - 1)*h ))
                         + 2.0*sum_2 + f(b));
}
```

```
double simpsons_rule( double f[], std::size_t capacity, double h ) {
    double sum{0.0};

    for ( std::size_t k{1}; k < capacity - 1; ++k ) {
        sum += f[k];
    }

    sum *= 2.0;
    sum += f[0] + f[capacity - 1];

    return sum*h/3;
}

double simpsons_3_8_rule( double f(double),
                          double a, double b,
                          unsigned int n ) {
    assert( n > 0 );
    assert( (n % 3) == 0 ); // 'n' must be a multiple of 3
    double h{(b - a)/n};

    double sum_2{0.0};
    double sum_3{0.0};

    for ( unsigned int k{1}; k < n/3; ++k ) {
        sum_3 += f( a + (3*k - 2)*h ) + f( a + (3*k - 1)*h );
        sum_2 += f( a +  3*k     *h );
    }

    return (3.0*h/8.0)*(f(a) + 3.0*(sum_3 + f( a + (n-2)*h ) + f( a + (n-1)*h ))
                            + 2.0*sum_2 + f(b));
}

int main() {
    std::cout.precision( 16 );

    std::cout << trapezoidal_rule( std::sin, 1, 3, 12 )  << std::endl;
    std::cout << trapezoidal_rule( std::sin, 1, 3, 120 ) << std::endl;
    std::cout << simpsons_rule( std::sin, 1, 3, 12 )     << std::endl;
    std::cout << simpsons_rule( std::sin, 1, 3, 120 )    << std::endl;
    std::cout << simpsons_3_8_rule( std::sin, 1, 3, 12 ) << std::endl;
    std::cout << simpsons_3_8_rule( std::sin, 1, 3, 120 ) << std::endl;
    std::cout << (std::cos(1.0) - std::cos(3.0)) << std::endl;

    return 0;
}
```

The output of this program is

```
1.526750812326977
1.530259378813789
1.530301384130549
1.530294803124598
1.530309660494876
1.530294803944662
1.530294802468585
```

Appendix: Finding a formula

Finding interpolating polynomials requires you to find the polynomial that interpolates, for example, the points (t_0, y_0), $(t_0 - h, y_{-1})$ and $(t_0 - 2h, y_{-2})$. We want to find a polynomial $at^2 + bt + c$ that passes through all three of these points; that is, we require that all three of these are true:

$$\begin{aligned} at_0^2 + bt_0 + c &= y_0 \\ a(t_0 - h)^2 + b(t_0 - h) + c &= y_{-1} \\ a(t_0 - 2h)^2 + b(t_0 - 2h) + c &= y_{-2} \end{aligned}$$

This is a system of linear equations in a, b and c, and thus we must solve

$$\left(\begin{array}{ccc|c} t_0^2 & t_0 & 1 & y_0 \\ (t_0 - h)^2 & t_0 - h & 1 & y_{-1} \\ (t_0 - 2h)^2 & t_0 - 2h & 1 & y_{-2} \end{array}\right)$$

We must now perform Gaussian elimination and backward substitution, so we start by adding $-\frac{(t_0 - h)^2}{t_0^2}$ times Row 1 onto Row 2, and adding $-\frac{(t_0 - 2h)^2}{t_0^2}$ times Row 1 onto Row 3:

$$\left(\begin{array}{ccc|c} t_0^2 & t_0 & 1 & y_0 \\ 0 & \frac{h(t_0 - h)}{t_0} & \frac{h(2t_0 - h)}{t_0^2} & y_{-1} - \frac{y_0 (t_0 - h)^2}{t_0^2} \\ 0 & \frac{2h(t_0 - 2h)}{t_0} & \frac{4h(t_0 - h)}{t_0^2} & y_{-2} - \frac{y_0 (t_0 - 2h)^2}{t_0^2} \end{array}\right)$$

Now, we must add $-\frac{\frac{2h(t_0 - 2h)}{t_0}}{\frac{h(t_0 - h)}{t_0}} = -\frac{2(t_0 - 2h)}{(t_0 - h)}$ times Row 2 onto Row 3:

$$\left(\begin{array}{ccc|c} t_0^2 & t_0 & 1 & y_0 \\ 0 & \frac{h(t_0 - h)}{t_0} & \frac{h(t_0 - 2h)}{t_0^2} & y_{-1} - y_0 \frac{(t_0 - h)^2}{t_0^2} \\ 0 & 0 & \frac{2h^2}{t_0(t_0 - h)} & y_{-2} - y_2 \frac{2(t_0 - 2h)}{t_0 - h} - y_0 \frac{2h - t_0}{t_0} \end{array}\right).$$

The next step is to solve for c, then b and then a:

$$c = \frac{y_{-2}\dfrac{t_0(t_0-h)}{t_0(t_0-h)} - y_2\dfrac{t_0 2(t_0-2h)}{t_0(t_0-h)} - y_0\dfrac{(2h-t_0)(t_0-h)}{t_0(t_0-h)}}{\dfrac{2h^2}{t_0(t_0-h)}}$$
$$= \frac{y_{-2}t_0(t_0-h) - y_2 t_0 2(t_0-2h) - y_0(2h-t_0)(t_0-h)}{2h^2}.$$

You can now substitute this into the second equation to solve for b:

$$b = \frac{y_{-2}(h-2t_0) + y_{-1}4(t_0-h) + y_0(3h-2t_0)}{2h^2}.$$

You can now substitute the values for both b and c into the first equation to solve for a:

$$a = \frac{y_0 - 2y_{-1} + y_{-2}}{2h^2}.$$

Consequently, the best-fitting polynomial that passes through the three points (t_0, y_0), $(t_0 - h, y_{-1})$ and $(t_0 - 2h, y_{-2})$ is the polynomial

$$\frac{y_0 - 2y_{-1} + y_{-2}}{2h^2}t^2 + \frac{y_{-2}(h-2t_0) + y_{-1}4(t_0-h) + y_0(3h-2t_0)}{2h^2}t + \frac{y_{-2}t_0(t_0-h) - y_2 t_0 2(t_0-2h) - y_0(2h-t_0)(t_0-h)}{2h^2}.$$

You can now, for example, differentiate this polynomial with respect to t:

$$\frac{y_0 - 2y_{-1} + y_{-2}}{h^2}t + \frac{y_{-2}(h-2t_0) + y_{-1}4(t_0-h) + y_0(3h-2t_0)}{2h^2}$$

and evaluate it at the point $t = t_0$:

$$\frac{y_0 - 2y_{-1} + y_{-2}}{h^2}t_0 + \frac{y_{-2}(h-2t_0) + y_{-1}4(t_0-h) + y_0(3h-2t_0)}{2h^2}$$

Finding a common denominator, we get

$$\frac{2y_0t_0 - 4y_{-1}t_0 + 2y_{-2}t_0 + y_{-2}(h-2t_0) + y_{-1}4(t_0-h) + y_0(3h-2t_0)}{2h^2},$$

and expanding this and the collecting on t_0, we have

$$\frac{2t_0(y_0 - 2y_{-1} + y_{-2} - y_{-2} + 2y_{-1} - y_0) + y_{-2}h - 4y_{-1}h + 3hy_0}{2h^2}.$$

If you were to simplify the numerator, you would find that

$$\frac{3y_0h - 4y_{-1}h + y_{-2}h}{2h^2}.$$

We can cancel out an h in both the numerator and the denominator to get

$$\frac{3y_0 - 4y_{-1} + y_{-2}}{2h}.$$

Note also if we differentiate the interpolating polynomial twice, we get

$$2\frac{y_0 - 2y_{-1} + y_{-2}}{2h^2}.$$

which simplifies to

$$\frac{y_0 - 2y_{-1} + y_{-2}}{h^2},$$

which you may recall is our approximation of the second derivative:

$$y^{(1)}(t_0) \approx \frac{y(t_0) - 2y(t_0 - h) + y(t_0 - 2h)}{h^2}.$$

Finally, let us integrate this interpolating polynomial from $t = t_0 - 2h$ to $t = t_0$. Note that this is simply a polynomial in t, and we have that

$$\int_{t_0-2h}^{t_0} \left(at^2 + bt + c\right) \mathrm{d}t = \frac{1}{3}\left({t_0}^3 - \left(t_0 - 2h\right)^3\right)a + \frac{1}{2}\left({t_0}^2 - \left(t_0 - 2h\right)^2\right)b + 2hc.$$

Substituting the above coefficients of the interpolating polynomial into this expression, we get

$$\frac{1}{3}\left({t_0}^3 - \left(t_0 - 2h\right)^3\right)\frac{y_0 - 2y_{-1} + y_{-2}}{2h^2} + \frac{1}{2}\left({t_0}^2 - \left(t_0 - 2h\right)^2\right)\frac{y_{-2}\left(h - 2t_0\right) + y_{-1}4\left(t_0 - h\right) + y_0\left(3h - 2t_0\right)}{2h^2}$$
$$+ 2h\frac{y_{-2}t_0\left(t_0 - h\right) - y_2 t_0 2\left(t_0 - 2h\right) - y_0\left(2h - t_0\right)\left(t_0 - h\right)}{2h^2}$$

If you expand and simplify this expression, you will get the very familiar expression:

$$\frac{1}{6}\left(y_0 + 4y_{-1} + y_{-2}\right)\left(2h\right).$$

Exercise: Carefully expand the above expression and demonstrate that when all other terms cancel, the only terms that remain are those in the formula for Simpson's rule. It is easiest to note that the above expression has terms containing t_0, ${t_0}^2$, and ${t_0}^3$, and if you were to collect terms, you would not all of these cancel, leaving simply those terms independent of t_0, which would then simplify to the expression above.

Acknowledgments

Dhviti Jignesh Patel
Kinderdeep Singh Virdi
Priyanka Hariharan
Deven Rajesh Pattni
Michel Georges Najarian
Priyanka Hariharan
Aditya Arora