Lecture X : Numerical integration

I. WHY DO WE NEED NUMERICAL INTEGRATION?

Although symbolic integration is nowadays available through such numerical packages as `Mathematica`, `Maple` and `Maxima`, there are still several cases where numerical integration is useful. Numerical integration comes in handy in cases where

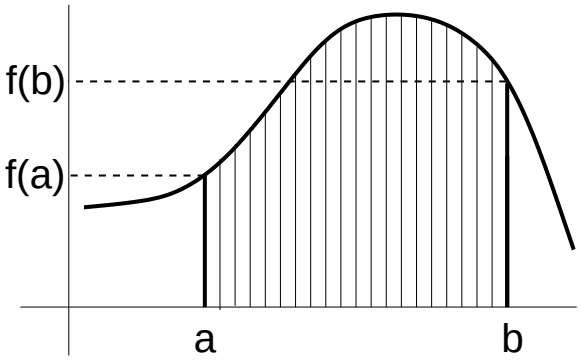
- the data over which to integrate is available at discrete points.
- the result of the integration cannot be evaluated in closed form.
- the result contains special functions which are not available.
- the result is too complicated to be typed up in a code without making errors.

II. TRAPEZOIDAL RULE

Suppose we'd like to integrate a smooth function in one dimension

$$I = \int_a^b f(x)dx. \tag{1}$$

We recall from elementary calculus that integrating a function geometrically corresponds to finding the area under the curve defined by the integrand, as seen in the following figure.

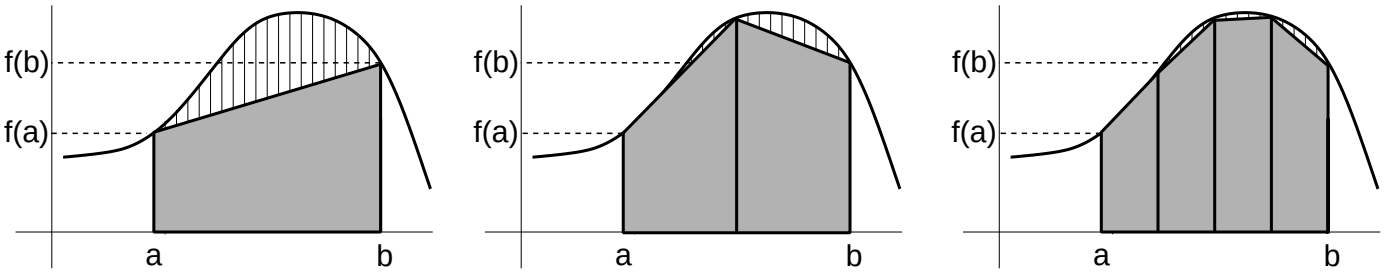


In the event that we do not know how to integrate the function analytically, we can make a *polynomial* approximation to the curve between the upper and lower limits, a and b. We may then easily find the area of the resulting curve either by integration or, in the case of a linear approximation, by simple geometry.

The simplest polynomial approximation is a linear approximation and is called the trapezoidal rule. If we, for instance, join points a and b in the above figure with a single line as seen in the rightmost panel in the figure below, the resulting shape is a trapezoid with corners a, b, $f(a)$ and $f(b)$. The area of this trapezoid is then a rather crude approximation to the integral in Eq. 1

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

We can improve the quality of the approximation by dividing the $[a, b]$ interval into subintervals and drawing separate trapezoids for every interval as seen in the middle panel of the figure below. It is obvious from the rightmost panel that the greater the number of trapezoids (or equivalently subintervals) the better the approximation to the integrand.



Let's calculate the area shaded in gray in the second figure

$$I \approx \sum_{i=1}^{2} A_i = \frac{1}{2}\left[f(a) + f\left(\frac{b+a}{2}\right)\right] \cdot \frac{b-a}{2} + \frac{1}{2}\left[f\left(\frac{b+a}{2}\right) + f(b)\right] \cdot \frac{b-a}{2} \tag{3}$$

and in the fourth figure

$$I \approx \sum_{i=1}^{4} A_i = \frac{b-a}{4}\left\{\frac{1}{2}f(a) + f\left(\frac{b-a}{4} + a\right) + f\left(\frac{2(b-a)}{4} + a\right) + f\left(\frac{3(b-a)}{4} + a\right) + \frac{1}{2}f(b)\right\} \tag{4}$$

Rearranging terms in Eq. 3 a little bit, we obtain Generalizing this sum to an arbitrary number, N of partitions

$$I \approx \sum_{i=1}^{N} A_i = h \cdot \left(\frac{f(a) + f(b)}{2} + \sum_{i=1}^{N-1} f(x_i)\right) \tag{5}$$

where $h = \frac{b-a}{N}$, $x_i = \frac{i(b-a)}{N} + a$, $x_0 = a$ and $x_N = b$.

This is very easy to code up in `Octave` so let's try that. We'll later translate it to `C` code in the lab.

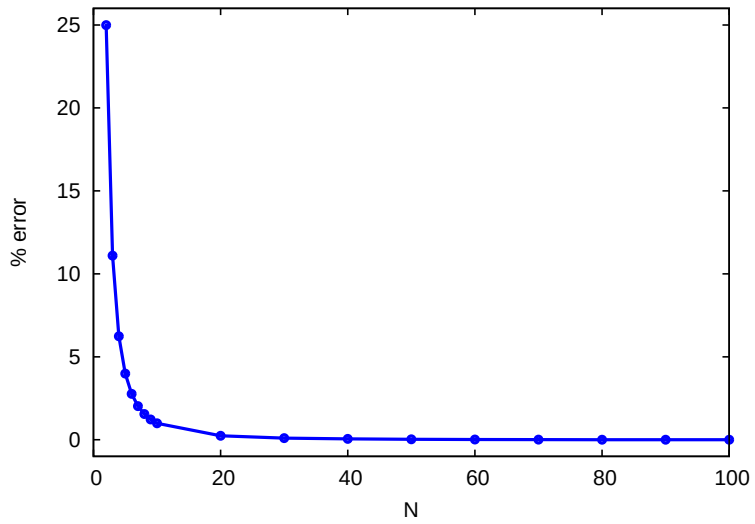
```
## Trapezoid integration of a given set of data assuming that the
## x-coordinates of the data points are equally spaced.

I=[];  ## Collect the approximations for different number of steps

lower=0;  ## Lower limit
upper=1;  ## Upper limit
Ns=[2 3 4 5 6 7 8 9 10:10:100];  ## Number of partitions
for N=Ns
  h=(upper-lower)/N;
  x=lower:h:upper;
  f=x.^3;
  r=sum(f)-0.5*(f(1)+f(N+1));
  r*=h;
  I=[I;r];
endfor
figure(1)
perr=100*(I-I(length(I)))/I(length(I));
plot(Ns,perr,'b*-');
th=Ns(find(perr<1)(1))
integral=I(length(I))
```

- In the beginning, we define the upper and lower limits of the integral in the variables `upper` and `lower`.
- At every step of the `for` loop, we apply the trapezoidal rule with an increasing number of partitions which are contained in the array `Ns`. The initial `Ns` are incremented one by one while the later ones go in steps of 10. This is because the error is expected to be larger in the earlier steps while it is supposed to decrease as we increase the number of steps.

- For each `N`, we define the step size or in other words the partition length, `h`.
- This also defines the collection of points, `x`, which we'd like to evaluate the function at.
- In this case, the function whose integral we'd like to evaluate is $f(x) = x^3$ and the trapezoid rule approximation to this integral is just a simple sum of the elements of the array `f` containing the function evaluated at all points of `x` minus half the sum of its first and second elements.
- After calculating the approximation to the integral for several partition numbers (collecting them in the array `I`), we plot the percent error of each of these steps with respect to the last element of `I`.
- At the same time, we find the first element of `Ns` for which the error is smaller than 1% and assign this to the variable `th`. Notice here the usage of `find`. `find(perr<1)` is an array containing the indices of elements of `perr` that are smaller than one and `find(perr<1)(1)` is just the first element of that array. As you see here, we don't need to form an intermediate array as a result of `find` but can use the outcome of that statement directly.
- Finally, our most converged result for the trapezoid rule is going to be the last element of the array `I`, which we assign to the variable `integral`.

When we run the script above, called `trapezoid`, we obtain the following result and graph.

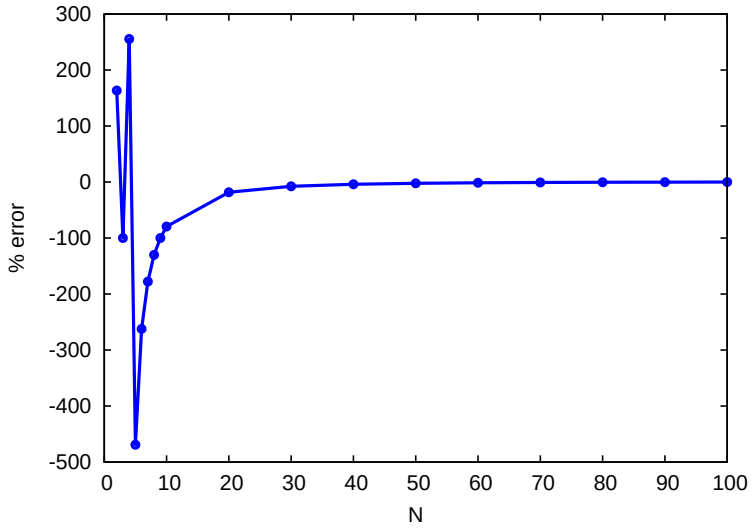
```
octave:1> trapezoid
th = 10
integral = 0.250025000000000
```



As you see here, for such a smooth function as x^3, the exact answer is reached within 10 partitions to within 1%. For a 100 partitions, we more or less obtain the exact answer.

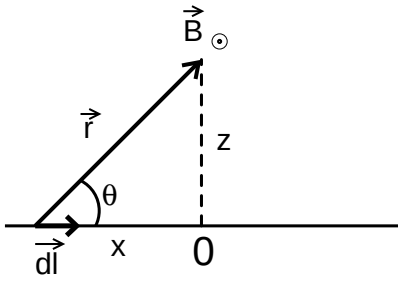
If instead, we try a more oscillatory function such as $\frac{\sin x}{x}$ from with `lower=pi` and `upper=10*pi`, the answer is

```
 octave:1> trapezoid
th = 70
integral = -0.310573949616896
```



You see now that because the function is highly oscillatory in the interval bounded by the upper and lower limits of integration, the required percent error is achieved only after 70 partitions. This is also reflected in the plot of the percent error where the initial few partition number give really awful approximations to the integral.

III. MAGNETIC FIELD PRODUCED BY A CURRENT-CARRYING WIRE



The infinitesimal magnetic field at a given point generated by a current carrying a current I is given by the Biot-Savart law as

$$d\vec{B} = \frac{\mu_0 I}{4\pi}\frac{\vec{dl}\times r}{r^3} \tag{6}$$

where $\vec{dl}$ is the infinitesimal distance in the same direction as the current, $\vec{r}$ is the vector that extends from the location of $\vec{dl}$ to the point where we'd like to evaluate the B-field, μ_0 is the permeability constant and I is the current. In order to evaluate the total field at the point of interest, we need to integrate $d\vec{B}$ in Eq. 6 over the entire wire. For a straight wire as depicted in the figure to the left, this is a tractable integral and rather straightforward integral if we are interested in finding the B-field at a distance z over the center point. Rearranging $d\vec{B}$ in terms of x and z, we have for the magnitude of $\vec{B}$

$$B = \frac{\mu_0 I}{4\pi}\int_{-a}^{a}\frac{z\,dx}{(x^2+z^2)^{3/2}} \equiv \frac{\mu_0 I}{4\pi}\mathcal{I} \tag{7}$$

where $\mp a$ are the limits of the wire. We also know on the other hand that the B-field at a vertical distance z from an infinite wire is given by $B = \frac{\mu_0 I}{2\pi z}$. Using our `trapezoid` code from above, we may evaluate B for, say $z = 1$ m where a goes from 1 to 100 seeing how fast the finite wire approaches the infinite wire. For an infinite wire and $z = 1$, $\mathcal{I}$ as defined in Eq. 7 is expected to be $\mathcal{I} = 2$. The results are given in the following table.

a	I
1	1.4142
5	1.9612
10	1.9901
100	1.9999

So a wire length of 100 m for a z of 1 m is, for all intents and purposes, an infinite wire. Of course as the wire length increases so does the number of partitions in order to get an accurate result (Why?). In this case, the integral is actually tractable. In lab, we are going to deal with a case that doesn't have a closed-form solution, namely the magnetic field generated at an arbitrary point carried by a circular ring carrying a current.