

Integrating discrete data

Matlab's `quad` and `quadl` functions, as well as all the programs we have written for Newton–Cotes and Gaussian quadrature, take as their first input argument a *formula* for the integrand f .

However, in many practical situations, we do not have a formula for the integrand, and in fact the value of f may only be known at a finite set of data points, say

$$(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n) \quad \text{where } a = x_1 < x_2 < \dots < x_n = b$$

How can we approximate the integral $\int_a^b f(x) dx$ in this case?

The most obvious approach is to integrate the *piecewise linear* (spline) function that interpolates the data, i.e. calculate the sum of the areas of the trapezoids bounded by the piecewise linear interpolant and the x -axis between $x = a$ and $x = b$.

This is done by Matlab's `trapz` command. It differs from the other integration commands we've met in that its input arguments should be *arrays* of x_i and corresponding $f(x_i)$ values.

To obtain a possibly better estimate for the integral of the underlying function, we could instead fit a *cubic spline* interpolant through the data points, generate denser sets of x and y values (taken from the interpolating curve), and use `trapz` with these larger arrays of x and y values as input.

As an example, download `v_car.txt`. This file contains data on the velocity of a car as it travels along a race track (in feet per second), measured at a sequence of times (seconds). The first column lists the times and the second column contains the corresponding velocities. How long is the race track?

Cumulative integrals

So far we have been computing definite integrals between fixed endpoints a and b , but often we may need to calculate integrals of the form

$$\int_a^x f(u) du$$

where the upper limit of integration can vary. For example, if $v(t)$ is the car's velocity at time t , then its displacement as a function of time (relative to its position at $t = 0$) is given by $\int_0^t v(\tau) d\tau$.

The only command Matlab has for automatically computing such “cumulative integrals” is `cumtrapz`. It takes the same kind of input arguments as `trapz`, i.e. arrays of x_i and $f(x_i)$ values, and outputs a sequence of values approximating the cumulative integral from x_1 up to $x_1, x_2, x_3, \dots, x_n$.

To obtain potentially more accurate approximations of the cumulative integral, we could do as above: fit a cubic spline interpolant through the data points and generate denser sets of x and y values to use as input for `cumtrapz`.

An alternative method is to fit a curve to the data with the *least-squares method*. This curve will have a formula that we can then feed into `quad` and `quadl`. Although there are no versions of `quad` and `quadl` that automatically perform cumulative integration, we can easily generate cumulative integral values by writing a for-loop.

Similarly, if a formula for the integrand f is known from the start, we can use a for-loop together with `quad` or `quadl` to compute $\int_a^x f(u) du$ for a sequence of x -values.

Multidimensional integrals

Finally, Matlab has commands `dblquad` and `triplequad` to compute double and triple integrals. For example, to approximate $\int_4^6 \int_0^1 (y^2 e^x + x \cos y) dx dy$, type

```
f=@(x,y) y.^2*exp(x) + x.*cos(y);
dblquad(f,0,1,4,6)
```