In MATLAB the function

$$f(x) = \left((x-3/10)^2 + \frac{1}{100}\right)^{-1} + \left(\left(x - \frac{9}{10}\right)^2 + 1/25\right)^{-1} - 6$$

is implemented as a routine called `humps`. It is often used for testing other routines, such as quadrature. Figure 8 shows the plot of `humps` on the interval $[0, 8]$.

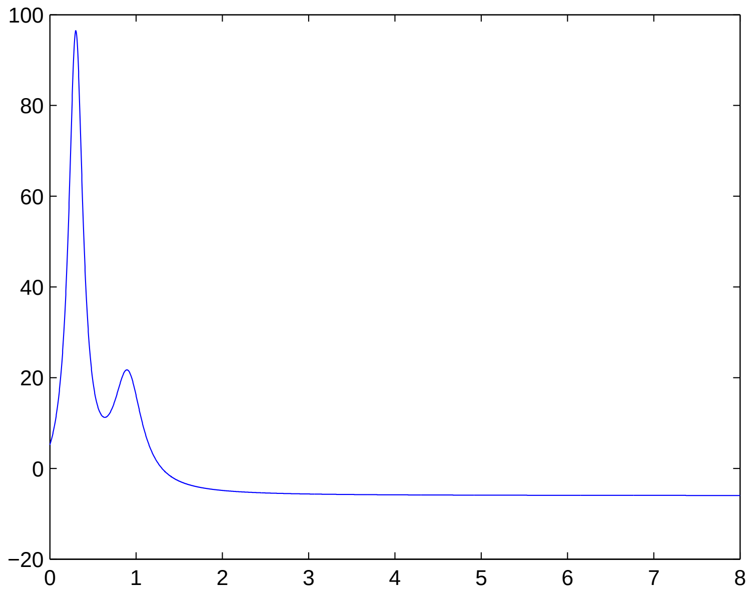

Figure 8. The plot of `humps` on $[0, 8]$

Notice that the function changes rapidly on [0,2] while on the rest of the interval its behavior is quite sedate: for $x > 3$ the function is nearly constant.

Suppose now that we want to use composite trapezoid rule to approximate the integral $\int_0^8 f(x)\,dx$ with a rather large tolerance 10^{-3}. If we use constant step size, we must subdivide the interval $[0, 8]$ into at least 578 subintervals. This already is a lot of functional evaluations for such a large tolerance. However things quickly become worse if we shrink tolerance. To achieve absolute error of 10^{-4} the minimum number of subdivisions must be 1826; whereas for 10^{-5} it is 5774. As the linear portion of Figure 9 implies, the cost of composite trapezoid rule obeys the asymptotic law:

$$\#\text{ of evaluations} \sim \text{tolerance}^{-1/2}, \quad \text{as tolerance} \to 0$$

This suggests that composite trapezoid rule *with even subdivisions* is not computationally effective.

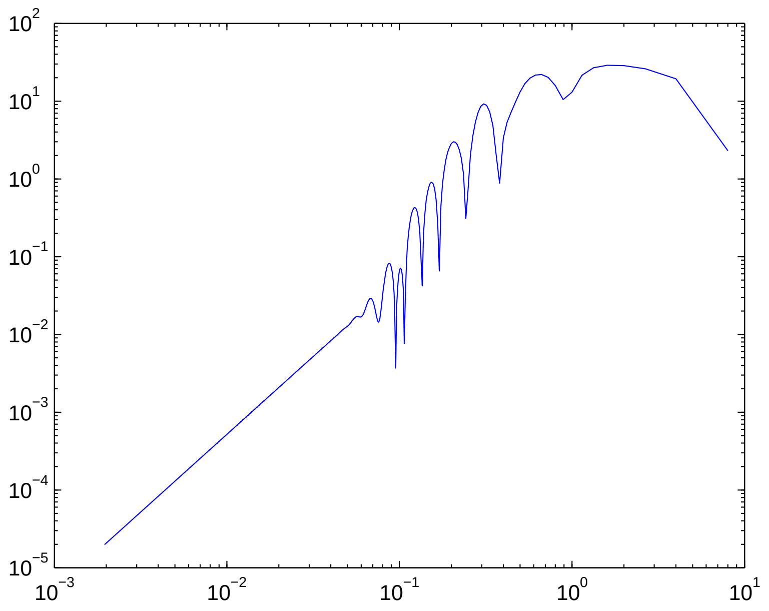

FIGURE 9. Log-log plot of the error of the composite trapezoid rule for $\int_0^8 f(x)\,dx$. Notice that for small subdivisions the plot is linear with slope 2, as predicted by theory.

One way to remedy the situation is to use a higher accuracy quadrature rule, say, the Simpson's rule. Yet, even if we use composite Simpson's rule, the computation will be inefficient for as long as the subdivision is even. The real problem is that bad behavior of the integrand on a small portion of the domain leads to a lot of extra computation on the portions of the domain where the integral is well-behaved. To compute integrals cost effectively we need to subdivide domains in a more flexible manner.

The idea of *adaptive quadrature* is based on additivity of integration:

$$\int_a^b f(x)\,dx = \int_a^c f(x)\,dx + \int_c^b f(x)\,dx.$$

Suppose we have a quadrature rule and a way to estimate error. If the error is too large when we apply the rule on the interval $[a, b]$, we can subdivide the interval into two halves and apply the rule on the smaller subintervals. If the error for either subinterval is too large, that subinterval is subdivided into two halves and the process repeats recursively, until desired tolerance is achieved.

Any quadrature rule can be turned into adaptive quadrature. The key to the enterprize is estimation of error which we discuss next.

6.1. **Error estimation.** Let T_1 be the result of applying the trapezoid rule to f on $[a, b]$:

$$T_1 = \frac{1}{2}\,(f(a) + f(b))\,(b - a).$$

Denote by c the midpoint of $[a, b]$ and let

$$T_2 = \frac{1}{2}\,(f(a) + f(c))\,\frac{b-a}{2} + (f(c) + f(b))\,\frac{b-a}{2}$$

be the estimate of the integral resulting from applying the trapezoid rule to $[a, c]$ and $[c, b]$ and adding the results. It stands to reason that T_2 should be a more accurate approximation to the integral $\int_a^b f(x)\,dx$ than T_1. Now, we know from the previous handout that

$$\int_a^b f(x)\,dx = T_1 - \frac{f''(\xi_1)}{12}(b-a)^3$$

and

$$\int_a^b f(x)\,dx = T_2 - \frac{f''(\xi_2)}{48}(b-a)^3.$$

If we assume that $f''(\xi_1) = f''(\xi_2)$, we get a system of two equations in two unknowns, from which follows that:

$$\int_a^b f(x)\,dx = \frac{1}{3}\,(4\,T_2 - T_1) \tag{33}$$

and

$$\frac{f''(\xi_1)}{12}(b-a)^3 = \frac{4}{3}\,(T_2 - T_1). \tag{34}$$

Equation (33) shows that the best estimate of the integral is not T_2 by itself, as one might expect, but rather a linear combination of T_1 and T_2; the process of combining T_1 and T_2 into an approximation that is more accurate than either T_1 or T_2 is an example of *Richardson's extrapolation.* Meanwhile, Equation (34) gives us a way to gauge the error of the trapezoid rule T_1.

We must bear in mind that we derived both Equation (33) and (34) by making a big assumption about the values of f''. This assumption is not always fulfilled, so our error estimate has to be taken with a grain of salt. Nevertheless, as the next section demonstrates, Equation (34) can be successfully used for error control.

7. Adaptive trapezoid rule in MATLAB

Since MATLAB supports recursive programming, we can implement adaptive quadrature rule using recursion. I prefer the implementation where the quadrature routine has a recursive integrator subroutine, as shown below.

```
function [Q,fcnt] = atrapz(f,a,b,tol)

% ATRAPZ   Numerically evaluate integral, adaptive trapezoid rule.

fa = f(a);
fb = f(b);
fcnt = 2;
Q = .5*(fa + fb)*(b-a);
Q = quadstep(a,b,fa,fb,Q);


function q = quadstep(a,b,fa,fb,T1)
    l = b - a;
    c = a + .5*l;
    fc = f(c);
    fcnt = fcnt + 1;
    Tleft = .25*(fa+fc)*l;
    Tright = .25*(fc+fb)*l;
    T2 = Tleft + Tright;
    E = 4*abs(T2 - T1)/3;
    if (b-a)*E < tol
        q = (4*T2 - T1)/3;
    else
        q = quadstep(a,c,fa,fc,Tleft) + quadstep(c,b,fc,fb,Tright);
    end
end

end
```

Notice that `quadstep` estimates the absolute error of the trapezoid method applied to a subinterval $[a, b]$ using Equation (34). The subdivision criterion, however, is based on the magnitude of

$$\frac{4}{3}\left|T_2 - T_1\right|(b-a).$$

The multiplication by $(b-a)$ improves the efficiency of the method. If the interval $[a, b]$ is very small, it does not need to be subdivided even if the error is relatively large.

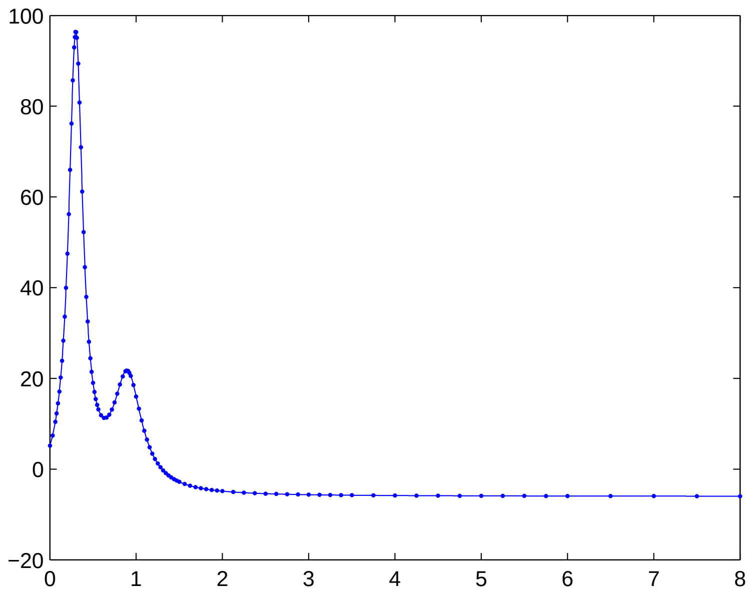

FIGURE 10. Adaptive trapezoid method applied to $\int_0^8 f(x)\,dx$ with tolerance 10^{-3}. The dots mark the resulting subdivision of $[0, 8]$.

Figure 10 shows the subdivision of $[0, 8]$ resulting from applying the adaptive trapezoid method to $\int_0^8 f(x)\,dx$ with tolerance 10^{-3}. The required tolerance is achieved using only 103 functional evaluations. For tolerance 10^{-6} adaptive trapezoid rule requires 607 functional evaluations; compare that with 5774 evaluations that composite rule with even subdivision requires to achieve tolerance 10^{-5}.

In order to gauge the efficiency of `atrapz` we can plot the number of functional evaluations against tolerance. Figure 11 shows the log-log plot which is almost a straight line with slope $-.2477$. Similar plots for different functions and intervals all result in lines with slopes close to $-.25$. This suggests that the functional count for adaptive trapezoid rule obeys the power law:

$$\# \text{ of evaluations} \sim \text{tolerance}^{-1/4}\,.$$

In words, the number of functional evaluations of adaptive trapezoid rule is roughly the square root of the number of functional evaluations of the composite trapezoid rule with even subdivisions.

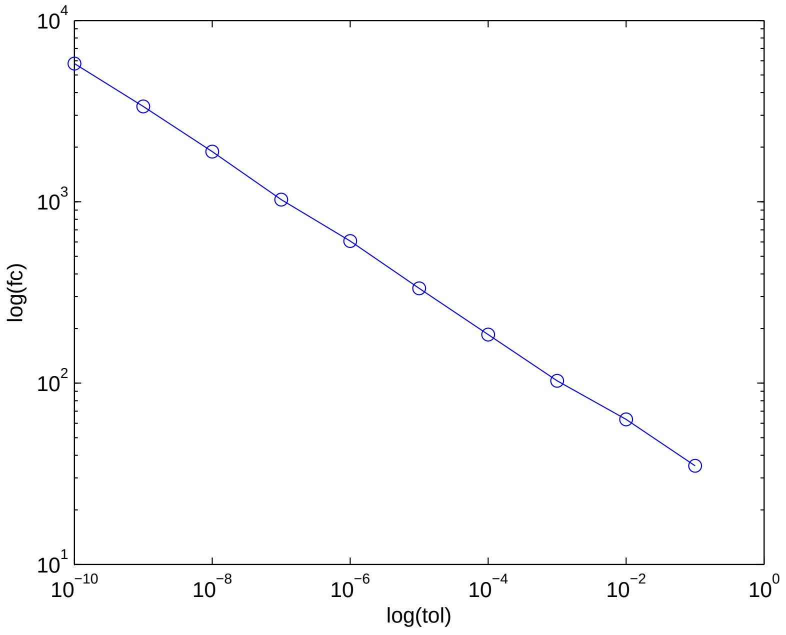

Figure 11. Efficiency of the adaptive trapezoid rule. The log-log plot of the number of functional evaluations against tolerance is nearly a straight line with slope $-.2477$.

Exercises

(1) Implement adaptive Simpson's rule and test it by integrating `humps` on $[0, 8]$ with tolerance ranging from 10^{-1} to 10^{-10}. Present your results in tabular form where for each tolerance level you compute the approximate value of the integral, the absolute error, and the number of functional evaluations. Additionally, make a log-log plot of the number of functional evaluations (cost) against the tolerance, similar to Figure 11. What can you say about the efficiency of the adaptive Simpson's rule?