Error estimates for quadrature rules

We have seen that by increasing the number of panels the error in quadrature becomes smaller. It is desirable to have a technique of estimating roughly the magnitude of the error so that we know how large an s we should pick so that we have an error below a given tolerance *tol*.

Let T_s be the approx to I by the composite trapezoidal rule with s panels and $E_s = I - T_s$ the respective error. Then $E_{2s} = I - T_{2s}$ is the error with $2s$ panels. If f'' does not vary a lot in $[a, b]$, we expect $E_{2s} \approx \frac{1}{4}E_s$. Thus,

$$\begin{aligned} & I - T_s = E_s \approx 4E_{2s} \\ & I - T_{2s} = E_{2s} \approx \tfrac{1}{4}E_s \\ & \Rightarrow E_{2s} \approx \tfrac{1}{3}(T_{2s} - T_s) \end{aligned}$$

This tells us that, if we have T_s and T_{2s} calculated we can also have a rough estimate to the error in T_{2s} calculated.

1. do $s+1$ func evals on $t_0, \ldots, t_s$
2. calculate $T_s = \frac{h}{2}(f(t_0) + 2\sum_{i=1}^{s-1} f(t_i) + f(t_s))$, $h = \frac{b-a}{s}$
3. do another s func evals $f(\frac{t_{i-1}+t_i}{2}), i = 1, \ldots, s$
4. calculate
$$\begin{aligned} T_{2s} &= \tfrac{h}{4}(f(t_0) + 2\sum_{i=1}^{s-1} f(t_i) \\ &\quad + 2\sum_{i=1}^{s-1} f(\tfrac{t_{i-1}+t_i}{2}) + f(t_s)) \\ &= \tfrac{1}{2}T_s + \tfrac{h}{2}\sum_{i=1}^{s-1} f(\tfrac{t_{i-1}+t_i}{2}) \end{aligned}$$
5. estimate the error $E_{2s} \approx \frac{1}{3}(T_{2s} - T_s) = Est$
6. If $|Est| \le tol$ accept T_{2s} as approx, otherwise set $s = 2s$,reset t_i's and repeat from 3.

Each time we double the number of panels we reuse the current func eval. The above technique can be applied to all NC rules. The fact that the nodes of an NC rule are equidistant helps in reusing the current func evals.

Adaptive quadrature

We have seen how a nested repetition of a compound quadrature rule gives us an approx to an integral within a certain tolerance.

If the function we integrate varies a lot in some part of (a, b) and very little in another part, we expect that this blind nested repetition of a compound rule wastes function evaluations in the part of little change.

Adaptive quadrature tries to avoid the waste of func evals, by smartly choosing the parts of (a, b) where the rule will be applied in a nested format. It dynamically refines the nodes in the parts of (a, b) where the accuracy has not reached a certain level and assigns an appropriate portion of the total error to each subinterval of (a, b).

Let $(Q, E) = LQ(f, a, b)$ be the routine that uses a simple quadrature rule in (a, b) for the function f and returns the approx Q and an error estimate E. Then the adapt quad routine may be implemented as follows:

```
function Q = AQ(f,a,b,tol)
(Q,E) = LQ(f,a,b)
if |E|> tol then
   m =(a+b)/2
   Q = AQ(f,a,m,tol/2) + AQ(f,m,b,tol/2)
endif
```

We often set a maximum recursion level in order to avoid non-convergent recursions.

Let Q=LQ(f,a,b) be a simple quad rule without error estimate.

```
function Q = AQ(f,a,b,tol)
Q0= ∞
Q = LQ(f,a,b)
if |Q − Q0| > |Q|tol then
    Q = AQs(f,a,b,tol,Q,1)
endif

function Q =AQs(f,a,b,tol,Q0,lv)
m =(a+b)/2
Q1 = LQ(f,a,m)
Q2 = LQ(f,m,b)
Q = Q1+Q2
if |Q − Q0| > |Q|tol and lv ≤ 10 then
    tol = tol/2, lv = lv+1
    Q1 = AQs(f,a,m,tol, Q1, lv)
    Q2 = AQs(f,m,b,tol, Q2, lv)
    Q = Q1+Q2
endif
```


Gauss quadrature

NC rules with nodes $x_0, \ldots, x_n$ have poly degree (pd) $\geq n$. The rules we've looked at have pd either exactly n (i.e. trapezoidal) or $n+1$ (i.e. midpt, Simpson). What is the highest pd we can achieve with $n+1$ nodes?

Consider a general rule of the form
$Q_n = \sum_{i=0}^{n} w_i f(x_i)$. There are $2n+2$ free parameters,
$n+1$ weights $w_i, i = 0, \ldots, n$
$n+1$ nodes $x_i, i = 0, \ldots, n$

In order to construct a quad rule with pd at least d, the following conditions must be satisfied:

$$\int_a^b x^k dx = \sum_{i=0}^{n} w_i x_i^k, k = 0, \ldots, d.$$

In order to solve for the $2n+2$ parameters, we choose $d = 2n+1$, so that we have a total of $2n+2$ conditions.

$$\begin{array}{ll} k=0 & \int_a^b dx = \sum_{i=0}^{n} w_i \\ k=1 & \int_a^b x dx = \sum_{i=0}^{n} w_i x_i \\ \vdots & \\ k=2n+1 & \int_a^b x^{2n+1} dx = \sum_{i=0}^{n} w_i x_i^{2n+1} \end{array} \quad (1)$$

Nonlinear systems of equations do not always have a unique solution, and even if they do, it might be complex or imaginary. Gauss showed that the nonlinear system (1) has a unique real solution.

The following can be shown:

- Equations (1) have a unique real solution for $w_i, x_i, i = 0 \ldots, n$ which gives rise to a quadrature rule of pd exactly $2n+1$ (i.e. the rule is not exact for a polynomial of degree $2n+2$).
- The points $x_i, i = 0, \ldots, n$, include neither a nor b.
- The weights w_i are positive.
- The set of points of an $(n+1)$-point Gauss rule is disjoint from the set of points of an $(m+1)$-point Gauss rule, for $n \neq m$, except if n, m are even, in which case only the midpt is a shared point.

- Open rules can be programmed to compute integrals with singularities on the end points.

- The weights and points for Gauss rules are usually taken from tables, so we do not have to solve (1). The points are usually given for $[-1, 1]$, then we have to map them to the interval and scale the range of integration appropriately.

- If we apply a compound Gauss rule with s panels, no panel share any function evals with any other panel. If we apply a compound Gauss rule with s panels and then the same with $2s$ panels, no function evals from the s-panel rule can be reused in the $2s$-panel rule.

Comparison between NC and Gauss rules

	NC	Gauss
$w_i > 0$	for $n \geq 10$ some $w_i < 0$	yes
w_i rational	yes	no
endpts included	yes/no	no
pd	$n \leq pd \leq 2n + 1$ in most cases $pd = n$ or $n + 1$	$pd = 2n+1$
interpolatory	yes	yes
reuse of func evals when doubling panels	often yes	mostly no

Infinite (and semi-infinite) integrals

Infinite integrals are those definite integrals whose at least one endpt is infinity

$$I = \int_a^\infty f(x)dx.$$

Assuming that I exists, how do we approximate I?

a) **Transformation:** Transform the infinite interval into a finite one by a change of variable. Note: not all changes of variable work out well.

Example: $I = \int_0^\infty e^{-x} \cos^2(x^2)dx$.

b) **Truncation:** Find a "sufficiently large" b and use

$$I = \int_a^\infty f(x)dx \approx \int_a^b f(x)dx = I_b$$

and apply a quadrature rule to I_b.

i) Keep using larger and larger values of b and stop when the answers seem to converge, i.e. when the relative difference in absolute value between successive answers is small.

ii) Find a b st $|I - I_b| \leq$ a certain tolerance. Note that

$$\int_a^\infty f(x)dx = \int_a^b f(x)dx + \int_b^\infty f(x)dx$$
$$\Rightarrow |I - I_b| = |\int_b^\infty f(x)dx|$$

Now try to work an upper bound on

$$|\int_b^\infty f(x)dx| \leq g(b)$$

in terms of b.
To compute I within accuracy ϵ
- choose b st $g(b) \leq \epsilon/2$
- compute I_b within accuracy $\epsilon/2$, say Q is the result.

Then

$$|I - Q| = |I - I_b + I_b - Q|$$
$$\leq |I - I_b| + |I_b - Q|$$
$$\leq \epsilon/2 + \epsilon/2 = \epsilon$$

Example: $f(x) = e^{-x} \cos x$

