Numerical integration: Gaussian quadrature rules

Matlab's built-in numerical integration function `[Q,fcount]=quad(f,a,b,tol)` is essentially our `simp_compextr` code with some further efficiency-enhancing features.
Note that `quad` requires scalar functions to be defined with elementwise operations, so $f(x) = \frac{2}{1+x^2}$ should be entered as `f=inline('2./(1+x.^2)', 'x')` or `f=@(x) 2./(1+x.^2)`
The default tolerance for `quad` is 10^{-6}.

Matlab has another efficient integration command called `quadl`, with the same input and output arguments. The method underlying `quadl` is a "Gaussian quadrature rule".

Recall that each Newton–Cotes quadrature rule came from integrating the Lagrange polynomial that interpolates the integrand f at n *equally spaced* nodes in the interval $[a, b]$. Thus, in general, we expect the degree of exactness of the rule to be $n - 1$ (though, as we've seen, some rules turn out to have a higher-than-expected degree of exactness).

Is it possible to find quadrature rules that use n nodes but have degree of exactness higher than $n-1$?

Any (non-composite) quadrature rule can be written in the form

$$(\star) \qquad \sum_{i=1}^{n} w_i \, f(x_i)$$

where the x_i are nodes and the w_i are non-zero constants called "quadrature weights".
If the n nodes are required to be equally spaced, then once you are told whether or not to include the endpoints (i.e. whether the rule should be closed or open), the positions of the nodes are all *fixed*, so it only remains to find the n weights w_i. These are given by

$$w_i = \int_a^b L_i(x)\, dx \qquad \text{where } L_i(x) = \prod_{k=1,\, k\neq i}^{n} \frac{x - x_i}{x_k - x_i}$$

But what if the nodes are not required to be equally spaced? If we put no restrictions on the positions of the nodes, other than that they must lie in the interval $[a, b]$, then $(\star)$ would contain $2n$ free parameters: $w_1, \ldots, w_n$ and $x_1, \ldots, x_n$. In general, $2n$ parameters could potentially define a polynomial of degree up to $2n-1$. Therefore, the highest degree of exactness we can expect to achieve with any quadrature rule is $2n - 1$.

To illustrate the procedure of finding such a quadrature rule with degree of exactness $2n - 1$, let us consider how to choose the w_i and x_i when $n = 2$ and the interval of integration is $[-1, 1]$. In other words, we want to determine w_1, w_2 and x_1, x_2 such that

$$w_1 f(x_1) + w_2 f(x_2) \quad \text{gives the exact value of} \quad \int_{-1}^{1} f(x)\, dx$$

whenever f is a polynomial of degree $2(2) - 1 = 3$ or less.

It can be proved using the theory of orthogonal polynomials that, on the integration interval $[-1,1]$, to achieve the maximum degree of exactness $2n-1$ with n nodes and n weights we should choose the nodes $x_1,\ldots,x_n$ to be the roots (zeros) of the degree-n *Legendre polynomial* $P_n(x)$; the weights are then given by $w_i = \displaystyle\int_{-1}^{1} \prod_{k=1,\,k\neq i}^{n} \frac{x-x_i}{x_k-x_i}\,dx$, and so $\displaystyle\sum_{i=1}^{n} w_i\, f(x_i)$ is an approximation of $\displaystyle\int_{-1}^{1} f(x)\,dx$.

The Legendre polynomials can be defined via the recursive relation

$$P_{k+1}(x) = \frac{2k+1}{k+1}\,x\,P_k(x) - \frac{k}{k+1}\,P_{k-1}(x) \qquad \text{with } \; P_0(x)=1, \;\; P_1(x)=x.$$

The first few are $P_2(x) = \frac{3}{2}(x^2-\frac{1}{3})$, $P_3(x) = \frac{5}{2}(x^3-\frac{3}{5}x)$ and $P_4(x) = \frac{1}{8}(35x^4-30x^2+3)$.

Each $P_k(x)$ is a polynomial of degree k, and has k roots that all lie in the interval $(-1,1)$.

So, to find the quadrature rule with maximum degree of exactness using n nodes and n weights, in principle we need to:

- Find the Legendre polynomial $P_n(x)$.
- Find the roots of $P_n(x)$ in $(-1,1)$; these will be our nodes $x_1,\ldots,x_n$ (and so the quadrature rule will be an "open" rule).
- Find the Lagrange polynomial that interpolates the integrand $f(x)$ at $x_1,\ldots,x_n$.
- Integrate the Lagrange polynomial to determine the weights $w_1,\ldots,w_n$.

Fortunately, the roots of the Legendre polynomials and their corresponding weights have been extensively tabulated, so we can simply use these tables without redoing the calculations. The only issue to be careful about is that the tabulated values were computed by taking $[-1,1]$ as the integration interval, whereas in any given problem the integration interval may not be $[-1,1]$. Therefore we need to transform the tabulated values to analogous values on a general interval $[a,b]$. This is done through the following change of variable:

$$\begin{array}{rccc} & -1 \le \tilde{x} \le 1 & \longleftrightarrow & a \le x \le b \\ \text{if} \quad & \dfrac{\tilde{x}-(-1)}{1-(-1)} & = & \dfrac{x-a}{b-a} \\ \text{so that} \quad & x & = & \dfrac{b-a}{2}\,\tilde{x} + \dfrac{a+b}{2} \end{array}$$

$$\begin{aligned} \int_a^b f(x)\,dx &= \int_{-1}^{1} f\left(\frac{b-a}{2}\,\tilde{x}+\frac{a+b}{2}\right)\frac{dx}{d\tilde{x}}\,d\tilde{x} \\ &= \int_{-1}^{1} f\left(\frac{b-a}{2}\,\tilde{x}+\frac{a+b}{2}\right)\frac{b-a}{2}\,d\tilde{x} \\ &= \frac{b-a}{2}\int_{-1}^{1} f\left(\frac{b-a}{2}\,\tilde{x}+\frac{a+b}{2}\right)d\tilde{x} \\ &\approx \frac{b-a}{2}\sum_{i=1}^{n} w_i\, f\left(\frac{b-a}{2}\,\tilde{x}_i+\frac{a+b}{2}\right) \end{aligned}$$

If we define $h = b-a$ (the length of the interval) and $c = \frac{a+b}{2}$ (the midpoint of the interval), then the roots $\tilde{x}_i$ in $[-1,1]$ are transformed to the nodes x_i in $[a,b]$ via $\boxed{x_i = \frac{h}{2}\,\tilde{x}_i + c}$, and the quadrature formula for approximating $\int_a^b f(x)\,dx$ will be $\boxed{\frac{h}{2}\text{ times}}$ the formula for approximating the equivalent integral over $[-1,1]$.

The quadrature rules defined above, using the roots of Legendre polynomials as their nodes, are called **Gauss–Legendre rules**. They have degree of exactness $2n-1$ (and order $2n$). Gauss–Legendre rules are open rules, and because the nodes are often positioned at irrational points in the interval, when we code the adaptive composite rules by repeatedly halving the interval, many extra function evaluations may need to be performed.

In order to make use of previously computed function values more efficiently, one could try to define a *closed* Gaussian quadrature rule. Such a rule would have $x_1 = a$ and $x_n = b$, and it turns out that the appropriate choice of the $n-2$ interior nodes should be the (transformed) roots of $P'_{n-1}(x)$ in $(-1, 1)$. These roots and their associated weights are also available in tables, and the same transformation as above would convert them to the corresponding nodes and quadrature formula on $[a, b]$. The rules defined in this way are called **Gauss–Lobatto rules**. Note that because x_1 and x_n are now fixed, the formula $\sum_{i=1}^{n} w_i f(x_i)$ has only $2n-2$ free parameters, so the degree of exactness attainable by a Gauss–Lobatto rule is $2n-3$ (order $2n-2$).

Gauss–Legendre nodes and coefficients

n	Roots of $P_n(x)$	Coefficients c_i
2	$-1/\sqrt{3}$	1
	$1/\sqrt{3}$	1
3	$-\sqrt{3/5}$	$5/9$
	0	$8/9$
	$\sqrt{3/5}$	$5/9$
4	$-\sqrt{\left(15+2\sqrt{30}\right)/35}$	$\left(18-\sqrt{30}\right)/36$
	$-\sqrt{\left(15-2\sqrt{30}\right)/35}$	$\left(18+\sqrt{30}\right)/36$
	$\sqrt{\left(15-2\sqrt{30}\right)/35}$	$\left(18+\sqrt{30}\right)/36$
	$\sqrt{\left(15+2\sqrt{30}\right)/35}$	$\left(18-\sqrt{30}\right)/36$

Gauss–Lobatto nodes and coefficients

n	x_1, roots of $P'_{n-1}(x)$, x_n	Coefficients c_i
2	-1	1
	1	1
3	-1	$1/3$
	0	$4/3$
	1	$1/3$
4	-1	$1/6$
	$-1/\sqrt{5}$	$5/6$
	$1/\sqrt{5}$	$5/6$
	1	$1/6$
5	-1	$1/10$
	$-\sqrt{3/7}$	$49/90$
	0	$32/45$
	$\sqrt{3/7}$	$49/90$
	1	$1/10$

Matlab's `quadl` employs a composite 4-node Gauss–Lobatto rule with some further efficiency-enhancing features.