

# Quadrature

M.Sc. in Mathematical Modelling & Scientific Computing,  
Practical Numerical Analysis

Michaelmas Term 2019, Lecture 3

# Quadrature

Suppose we want to compute

$$I(f) = \int_a^b \mu(x)f(x)dx$$

where  $\mu(x)$  is a non-negative weight function (we will consider  $\mu(x) \equiv 1$  for now). Unfortunately most integrals do not have closed form solutions. For example, what is

$$I(f) = \int_{-1}^1 \exp(-x^2)dx ?$$

# Quadrature

Suppose we want to compute

$$I(f) = \int_a^b \mu(x)f(x)dx$$

where  $\mu(x)$  is a non-negative weight function (we will consider  $\mu(x) \equiv 1$  for now). Unfortunately most integrals do not have closed form solutions. For example, what is

$$I(f) = \int_{-1}^1 \exp(-x^2)dx ?$$

The idea of quadrature is to approximate  $I(f)$  so

$$I(f) \approx I_n(f) = \sum_{k=0}^n w_k f(x_k) .$$

Here

- ▶  $n$  is the degree of the quadrature
- ▶  $x_k$  are the quadrature nodes
- ▶  $w_k$  are the quadrature weights

# Relation to Interpolation

One of the reasons we gave for interpolation of  $f(x)$  by  $p(x)$  was

“I might want to know  $\int_a^b f(x)dx$  — a good approximation should be  $\int_a^b p(x)dx$  .”

We looked at different sorts of interpolants in 1D:

- ▶ Lagrange interpolant on uniform meshes;
- ▶ Lagrange interpolant on Chebyshev meshes;
- ▶ Splines, in particular linear splines.

What sort of quadrature rules do these lead to?

# Integral of Linear Lagrange Interpolant

The linear Lagrange interpolant of  $f(x)$  on  $[a, b]$  can be written as

$$p_1(x) = \frac{x-a}{b-a}f(b) + \frac{b-x}{b-a}f(a) .$$

# Integral of Linear Lagrange Interpolant

The linear Lagrange interpolant of  $f(x)$  on  $[a, b]$  can be written as

$$p_1(x) = \frac{x-a}{b-a}f(b) + \frac{b-x}{b-a}f(a) .$$

Then we can write

$$\begin{aligned} I(f) &= \int_a^b f(x)dx \approx \int_a^b p_1(x)dx \\ &= \frac{b-a}{2}(f(a) + f(b)) . \end{aligned}$$

# Integral of Linear Lagrange Interpolant

The linear Lagrange interpolant of  $f(x)$  on  $[a, b]$  can be written as

$$p_1(x) = \frac{x-a}{b-a}f(b) + \frac{b-x}{b-a}f(a) .$$

Then we can write

$$\begin{aligned} I(f) &= \int_a^b f(x)dx \approx \int_a^b p_1(x)dx \\ &= \frac{b-a}{2}(f(a) + f(b)) . \end{aligned}$$

This is the trapezium rule!

# Integral of Quadratic Lagrange Interpolant

The quadratic Lagrange interpolant of  $f(x)$  at the points  $a$ ,  $(a + b)/2$ , and  $b$  can be written as

$$\begin{aligned} p_2(x) = & \frac{x-a}{b-a} \frac{2x-(a+b)}{b-a} f(b) - \frac{2(x-a)}{b-a} \frac{2(x-b)}{b-a} f\left(\frac{a+b}{2}\right) \\ & + \frac{x-b}{b-a} \frac{2x-(a+b)}{b-a} f(a) . \end{aligned}$$



# Integral of Quadratic Lagrange Interpolant

The quadratic Lagrange interpolant of  $f(x)$  at the points  $a$ ,  $(a + b)/2$ , and  $b$  can be written as

$$\begin{aligned} p_2(x) = & \frac{x-a}{b-a} \frac{2x-(a+b)}{b-a} f(b) - \frac{2(x-a)}{b-a} \frac{2(x-b)}{b-a} f\left(\frac{a+b}{2}\right) \\ & + \frac{x-b}{b-a} \frac{2x-(a+b)}{b-a} f(a) . \end{aligned}$$

Then we can write

$$\begin{aligned} I(f) = \int_a^b f(x) dx & \approx \int_a^b p_2(x) dx \\ & = \frac{b-a}{6} \left( f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right) . \end{aligned}$$

# Integral of Quadratic Lagrange Interpolant

The quadratic Lagrange interpolant of  $f(x)$  at the points  $a$ ,  $(a + b)/2$ , and  $b$  can be written as

$$\begin{aligned} p_2(x) = & \frac{x-a}{b-a} \frac{2x-(a+b)}{b-a} f(b) - \frac{2(x-a)}{b-a} \frac{2(x-b)}{b-a} f\left(\frac{a+b}{2}\right) \\ & + \frac{x-b}{b-a} \frac{2x-(a+b)}{b-a} f(a) . \end{aligned}$$

Then we can write

$$\begin{aligned} I(f) = \int_a^b f(x) dx & \approx \int_a^b p_2(x) dx \\ & = \frac{b-a}{6} \left( f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right) . \end{aligned}$$

This is Simpson's rule!

# Newton-Cotes

The Newton-Cotes quadrature rules are the extensions of the trapezium rule and Simpson's rule to interpolants of higher degrees.

# Newton-Cotes

The Newton-Cotes quadrature rules are the extensions of the trapezium rule and Simpson's rule to interpolants of higher degrees.

Let  $p_n(x)$  be the Lagrange interpolant of degree  $n$  of  $f(x)$  at the uniformly spaced nodes  $x_k = a + k(b - a)/n$ ,  $0 \leq k \leq n$ . Then the Newton-Cotes rule is

$$I(f) \approx I(p_n) = \sum_{k=0}^n f(x_k) \int_a^b L_{n,k}(x) dx .$$

The integral on the right-hand-side can be computed exactly since the integrand is just a polynomial of degree  $n$ .

# Error in Newton-Cotes Quadrature

We wrote down a formula for the error in the Lagrange interpolant as:

$$f(x) - p_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{k=0}^n (x - x_k) . \quad (1)$$

# Error in Newton-Cotes Quadrature

We wrote down a formula for the error in the Lagrange interpolant as:

$$f(x) - p_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{k=0}^n (x - x_k). \quad (1)$$

Integrating this gives an error bound for Newton-Cotes quadrature of the form

$$\left| \int_a^b f(x) dx - \int_a^b p_n(x) dx \right| \leq \frac{\max_{\xi \in [a,b]} |f^{(n+1)}(\xi)|}{(n+1)!} \left| \int_a^b \prod_{k=0}^n (x - x_k) dx \right|.$$

# Error in Newton-Cotes Quadrature

We wrote down a formula for the error in the Lagrange interpolant as:

$$f(x) - p_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{k=0}^n (x - x_k). \quad (1)$$

Integrating this gives an error bound for Newton-Cotes quadrature of the form

$$\left| \int_a^b f(x) dx - \int_a^b p_n(x) dx \right| \leq \frac{\max_{\xi \in [a,b]} |f^{(n+1)}(\xi)|}{(n+1)!} \left| \int_a^b \prod_{k=0}^n (x - x_k) dx \right|.$$

We already saw that uniformly spaced points are bad for polynomial interpolation and it follows that Newton-Cotes quadrature does not work well for large degrees.

## Possible Remedies

Since Newton-Cotes is not an effective quadrature rule for high degrees we need an alternative. Possibilities are:



## Possible Remedies

Since Newton-Cotes is not an effective quadrature rule for high degrees we need an alternative. Possibilities are:

- ▶ Smaller degrees on sub-intervals — composite rules;

# Possible Remedies

Since Newton-Cotes is not an effective quadrature rule for high degrees we need an alternative. Possibilities are:

- ▶ Smaller degrees on sub-intervals — composite rules;
- ▶ Integrate better interpolants (using different nodes) — Gauss quadrature/ Clenshaw-Curtis.

# Possible Remedies

Since Newton-Cotes is not an effective quadrature rule for high degrees we need an alternative. Possibilities are:

- ▶ Smaller degrees on sub-intervals — composite rules;
- ▶ Integrate better interpolants (using different nodes) — Gauss quadrature/ Clenshaw-Curtis.

In either case adaptivity can help improve efficiency.

# Composite Trapezium Rule

Here the idea is to split the range of integration into subintervals and to apply the trapezium rule on each subinterval.

# Composite Trapezium Rule

Here the idea is to split the range of integration into subintervals and to apply the trapezium rule on each subinterval. Hence (with  $x_k = a + kh$ ,  $h = (b - a)/m$ )

$$\begin{aligned}\int_a^b f(x)dx &= \sum_{k=1}^m \int_{x_{k-1}}^{x_k} f(x)dx \\ &\approx \frac{h}{2} \sum_{k=1}^m (f(x_{k+1}) + f(x_k)) \\ &= \frac{h}{2} \left( f(a) + 2 \sum_{k=1}^{m-1} f(x_k) + f(b) \right) =: I_m(f) .\end{aligned}$$

# Composite Trapezium Rule

Here the idea is to split the range of integration into subintervals and to apply the trapezium rule on each subinterval. Hence (with  $x_k = a + kh$ ,  $h = (b - a)/m$ )

$$\begin{aligned}\int_a^b f(x)dx &= \sum_{k=1}^m \int_{x_{k-1}}^{x_k} f(x)dx \\ &\approx \frac{h}{2} \sum_{k=1}^m (f(x_{k+1}) + f(x_k)) \\ &= \frac{h}{2} \left( f(a) + 2 \sum_{k=1}^{m-1} f(x_k) + f(b) \right) =: I_m(f) .\end{aligned}$$

This is the composite trapezium rule and can also be thought of as the integral of the linear spline approximation of  $f(x)$ .

# Error in Composite Trapezium Rule

We can simply integrate the error given by Equation (1) on each subinterval and sum to get the error in the composite trapezium rule as

$$|I(f) - I_m(f)| \leq \frac{h^2(b-a)}{12} \max_{\xi \in [a,b]} |f''(\xi)|.$$

# Error in Composite Trapezium Rule

We can simply integrate the error given by Equation (1) on each subinterval and sum to get the error in the composite trapezium rule as

$$|I(f) - I_m(f)| \leq \frac{h^2(b-a)}{12} \max_{\xi \in [a,b]} |f''(\xi)|.$$

If  $f$  is a periodic analytic function we see geometric convergence.



# Error in Composite Trapezium Rule

We can simply integrate the error given by Equation (1) on each subinterval and sum to get the error in the composite trapezium rule as

$$|I(f) - I_m(f)| \leq \frac{h^2(b-a)}{12} \max_{\xi \in [a,b]} |f''(\xi)|.$$

If  $f$  is a periodic analytic function we see geometric convergence.

Note that the points  $x_k$  do not have to be equally spaced. In this case the error bound becomes

$$|I(f) - I_m(f)| \leq \sum_{k=1}^m \frac{(x_k - x_{k-1})^3}{12} \max_{\xi \in [x_{k-1}, x_k]} |f''(\xi)|,$$

and this can be used as the basis for an adaptive quadrature rule.

# Composite Simpson's Rule

There is an analagous composite Simpson's rule

$$I_n(f) = \frac{h}{3} \left( f(a) + 2 \sum_{k=1}^{n/2-1} f(x_{2k}) + 4 \sum_{k=1}^{n/2} f(x_{2k-1}) + f(b) \right)$$

where  $n$  must be even.

# Composite Simpson's Rule

There is an analagous composite Simpson's rule

$$I_n(f) = \frac{h}{3} \left( f(a) + 2 \sum_{k=1}^{n/2-1} f(x_{2k}) + 4 \sum_{k=1}^{n/2} f(x_{2k-1}) + f(b) \right)$$

where  $n$  must be even. The approximation error is

$$|I(f) - I_n(f)| \leq \frac{h^4(b-a)}{180} \max_{\xi \in [a,b]} |f^{(4)}(\xi)| .$$

## Clenshaw-Curtis Quadrature

We already saw that using Chebyshev points was great for interpolation on  $[-1, 1]$  and we could scale to other intervals. The idea of Clenshaw-Curtis rules is to integrate polynomial interpolants over  $[-1, 1]$  based on Chebyshev points (and we can perform a change of variable first if we wish to integrate over other intervals).

## Clenshaw-Curtis Quadrature

We already saw that using Chebyshev points was great for interpolation on  $[-1, 1]$  and we could scale to other intervals. The idea of Clenshaw-Curtis rules is to integrate polynomial interpolants over  $[-1, 1]$  based on Chebyshev points (and we can perform a change of variable first if we wish to integrate over other intervals).

Such quadrature rules inherit the accuracy of the interpolant so  $|I(f) - I_n(f)| \sim \mathcal{O}(\rho^{-n})$  as  $n \rightarrow \infty$ .

# Clenshaw-Curtis Quadrature

We already saw that using Chebyshev points was great for interpolation on  $[-1, 1]$  and we could scale to other intervals. The idea of Clenshaw-Curtis rules is to integrate polynomial interpolants over  $[-1, 1]$  based on Chebyshev points (and we can perform a change of variable first if we wish to integrate over other intervals).

Such quadrature rules inherit the accuracy of the interpolant so  $|I(f) - I_n(f)| \sim \mathcal{O}(\rho^{-n})$  as  $n \rightarrow \infty$ .

Unfortunately the nice representation of the Chebyshev interpolant we had via the second barycentric interpolation formula is not helpful here. It helps to re-write the interpolant as

$$p_n(x) = \sum_{k=0}^n \alpha_k T_k(x),$$

where  $T_k(x)$  is the degree  $k$  first kind Chebyshev polynomial.

# Clenshaw-Curtis Quadrature: Coefficient Space

The Chebyshev polynomials are defined as

$$T_k(x) = \cos(k \cos^{-1}(x))$$

for  $x \in [-1, 1]$ .

# Clenshaw-Curtis Quadrature: Coefficient Space

The Chebyshev polynomials are defined as

$$T_k(x) = \cos(k \cos^{-1}(x))$$

for  $x \in [-1, 1]$ . Thus we have

$$\begin{aligned} \int_{-1}^1 T_k(x) dx &= \int_{-1}^1 \cos(k \cos^{-1}(x)) dx \\ &= \int_0^\pi \cos(k\theta) \sin(\theta) d\theta \\ &= \begin{cases} 0 & \text{for } k \text{ odd} \\ \frac{2}{1-k^2} & \text{for } k \text{ even} . \end{cases} \end{aligned}$$



## Clenshaw-Curtis Quadrature: Coefficient Space

The Chebyshev polynomials are defined as

$$T_k(x) = \cos(k \cos^{-1}(x))$$

for  $x \in [-1, 1]$ . Thus we have

$$\begin{aligned} \int_{-1}^1 T_k(x) dx &= \int_{-1}^1 \cos(k \cos^{-1}(x)) dx \\ &= \int_0^\pi \cos(k\theta) \sin(\theta) d\theta \\ &= \begin{cases} 0 & \text{for } k \text{ odd} \\ \frac{2}{1-k^2} & \text{for } k \text{ even} . \end{cases} \end{aligned}$$

Thus we may write the quadrature rule as

$$I(f) \approx I(p_n) = \sum_{\substack{k=0 \\ k \text{ even}}}^n \frac{2\alpha_k}{1-k^2} ,$$

where the  $\alpha_k$  can be found using a Vandermonde matrix approach. 

# Clenshaw-Curtis Quadrature: Value Space

An alternative approach is to find the weights,  $w_k$ , such that

$$I(p_n) = \sum_{k=0}^n w_k f(x_k).$$

# Clenshaw-Curtis Quadrature: Value Space

An alternative approach is to find the weights,  $w_k$ , such that  $I(p_n) = \sum_{k=0}^n w_k f(x_k)$ .

We can do this by ensuring that the quadrature rule integrates each  $T_k$  exactly, i.e. we require

$$\sum_{k=0}^n w_k T_j(x_k) = \int_{-1}^1 T_j(x) dx ,$$

for  $0 \leq j \leq n$ . Thus we again solve a Vandermonde type system,  $V^T \mathbf{w} = \mathbf{b}$  where  $b_j = \int_{-1}^1 T_j(x) dx$ .

# Clenshaw-Curtis Quadrature: Value Space

An alternative approach is to find the weights,  $w_k$ , such that  $I(p_n) = \sum_{k=0}^n w_k f(x_k)$ .

We can do this by ensuring that the quadrature rule integrates each  $T_k$  exactly, i.e. we require

$$\sum_{k=0}^n w_k T_j(x_k) = \int_{-1}^1 T_j(x) dx ,$$

for  $0 \leq j \leq n$ . Thus we again solve a Vandermonde type system,  $V^T \mathbf{w} = \mathbf{b}$  where  $b_j = \int_{-1}^1 T_j(x) dx$ .

(In practice this system can be solved in  $\mathcal{O}(n \log n)$  operations via an FFT/DCT.)

# Gauss Quadrature

So far we have fixed the nodes of a quadrature rule and then chosen the weights to intergrate the corresponding polynomial interpolant exactly.

# Gauss Quadrature

So far we have fixed the nodes of a quadrature rule and then chosen the weights to integrate the corresponding polynomial interpolant exactly.

An alternative is to choose both the weights and nodes in the formula

$$I_n(f) = \sum_{k=0}^N w_k f(x_k) .$$

# Gauss Quadrature

So far we have fixed the nodes of a quadrature rule and then chosen the weights to integrate the corresponding polynomial interpolant exactly.

An alternative is to choose both the weights and nodes in the formula

$$I_n(f) = \sum_{k=0}^N w_k f(x_k) .$$

Here there are  $2n + 2$  unknowns ( $n + 1$  nodes and  $n + 1$  weights) and so these can be chosen to integrate all polynomials of degree  $2n + 1$  exactly. This is the idea behind Gauss quadrature.

# Gauss Quadrature: Derivation

Orthogonal polynomials on an interval  $[a, b]$  with respect to a weight function  $\mu(x)$  are defined to be the polynomials  $P_0(x), P_1(x), \dots$  such that  $P_k(x)$  is a polynomial of degree  $k$  and the orthogonality property

$$\int_a^b \mu(x) P_j(x) P_k(x) dx = 0,$$

holds, whenever  $j \neq k$ .



# Gauss Quadrature: Derivation

Orthogonal polynomials on an interval  $[a, b]$  with respect to a weight function  $\mu(x)$  are defined to be the polynomials  $P_0(x), P_1(x), \dots$  such that  $P_k(x)$  is a polynomial of degree  $k$  and the orthogonality property

$$\int_a^b \mu(x) P_j(x) P_k(x) dx = 0,$$

holds, whenever  $j \neq k$ .

Now define  $\mathbb{P}_k$  to be the set of polynomials of degree  $k$ . Then, for  $f_{2n+1} \in \mathbb{P}_{2n+1}$  we may write

$$f_{2n+1}(x) = q_n(x) P_{n+1}(x) + r_n(x)$$

for some  $q_n, r_n \in \mathbb{P}_n$ .

# Gauss Quadrature: Derivation

Then we have

$$I(f_{2n+1}) = \int_a^b \mu(x) f_{2n+1}(x) dx$$

# Gauss Quadrature: Derivation

Then we have

$$\begin{aligned} I(f_{2n+1}) &= \int_a^b \mu(x) f_{2n+1}(x) dx \\ &= \int_a^b \mu(x) q_n(x) P_{n+1}(x) dx + \int_a^b \mu(x) r_n(x) dx \end{aligned}$$

# Gauss Quadrature: Derivation

Then we have

$$\begin{aligned} I(f_{2n+1}) &= \int_a^b \mu(x) f_{2n+1}(x) dx \\ &= \int_a^b \mu(x) q_n(x) P_{n+1}(x) dx + \int_a^b \mu(x) r_n(x) dx \\ &= \sum_{k=0}^n \alpha_k \int_a^b \mu(x) P_k(x) P_{n+1}(x) dx + \int_a^b \mu(x) r_n(x) dx \end{aligned}$$

# Gauss Quadrature: Derivation

Then we have

$$\begin{aligned} I(f_{2n+1}) &= \int_a^b \mu(x) f_{2n+1}(x) dx \\ &= \int_a^b \mu(x) q_n(x) P_{n+1}(x) dx + \int_a^b \mu(x) r_n(x) dx \\ &= \sum_{k=0}^n \alpha_k \int_a^b \mu(x) P_k(x) P_{n+1}(x) dx + \int_a^b \mu(x) r_n(x) dx \\ &= 0 + \int_a^b \mu(x) r_n(x) dx . \end{aligned}$$

## Gauss Quadrature: Derivation

Now let the  $x_k$  be the  $n + 1$  roots of  $P_{n+1}(x)$ , then the quadrature rule gives

$$I_n(f_{2n+1}) = \sum_{k=0}^n w_k f_{2n+1}(x_k)$$

## Gauss Quadrature: Derivation

Now let the  $x_k$  be the  $n + 1$  roots of  $P_{n+1}(x)$ , then the quadrature rule gives

$$\begin{aligned} I_n(f_{2n+1}) &= \sum_{k=0}^n w_k f_{2n+1}(x_k) \\ &= \sum_{k=0}^n w_k q_n(x_k) P_{n+1}(x_k) + \sum_{k=0}^n w_k r_n(x_k) \end{aligned}$$

## Gauss Quadrature: Derivation

Now let the  $x_k$  be the  $n + 1$  roots of  $P_{n+1}(x)$ , then the quadrature rule gives

$$\begin{aligned} I_n(f_{2n+1}) &= \sum_{k=0}^n w_k f_{2n+1}(x_k) \\ &= \sum_{k=0}^n w_k q_n(x_k) P_{n+1}(x_k) + \sum_{k=0}^n w_k r_n(x_k) \\ &= \sum_{k=0}^n w_k r_n(x_k) . \end{aligned}$$



## Gauss Quadrature: Derivation

Now let the  $x_k$  be the  $n + 1$  roots of  $P_{n+1}(x)$ , then the quadrature rule gives

$$\begin{aligned} I_n(f_{2n+1}) &= \sum_{k=0}^n w_k f_{2n+1}(x_k) \\ &= \sum_{k=0}^n w_k q_n(x_k) P_{n+1}(x_k) + \sum_{k=0}^n w_k r_n(x_k) \\ &= \sum_{k=0}^n w_k r_n(x_k) . \end{aligned}$$

Thus, if we choose the weights  $w_k$  such that polynomial interpolants through the  $x_k$  are integrated exactly, we have

$$I(f_{2n+1}) = I_n(f_{2n+1}) .$$

# Gauss Quadrature: Examples

Different ranges of integration and different weight functions lead to different sets of orthogonal polynomials which define Gauss quadrature rules. Common examples are:

Name	Interval	Weight
Gauss-Legendre	$[-1, 1]$	1
Gauss-Chebyshev	$[-1, 1]$	$1/\sqrt{(1-x^2)}$
Gauss-Jacobi	$[-1, 1]$	$(1+x)^\alpha(1-x)^\beta$
Gauss-Laguerre	$[0, \infty)$	$\exp(-x)$
Gauss-Hermite	$(-\infty, \infty)$	$\exp(-x^2)$

# Gauss Quadrature: Computing the Nodes and Weights

Orthogonal polynomials satisfy a recurrence relation

$$\gamma_k P_{k-1}(x) + \beta_k P_k(x) + \gamma_{k+1} P_{k+1}(x) = x P_k(x) .$$

# Gauss Quadrature: Computing the Nodes and Weights

Orthogonal polynomials satisfy a recurrence relation

$$\gamma_k P_{k-1}(x) + \beta_k P_k(x) + \gamma_{k+1} P_{k+1}(x) = x P_k(x).$$

We can write this in matrix form as

$$\begin{pmatrix} \beta_0 & \gamma_1 & & & \\ \gamma_1 & \beta_1 & \gamma_2 & & \\ & \gamma_2 & \beta_2 & \gamma_3 & \\ & & \ddots & \ddots & \ddots \\ & & & \gamma_n & \beta_n \end{pmatrix} \begin{pmatrix} P_0(x) \\ P_1(x) \\ P_2(x) \\ \vdots \\ P_n(x) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ \gamma_{n+1} P_{n+1}(x) \end{pmatrix} = x \begin{pmatrix} P_0(x) \\ P_1(x) \\ P_2(x) \\ \vdots \\ P_n(x) \end{pmatrix}$$

or equivalently  $T\mathbf{P}(x) + \gamma_{n+1}P_{n+1}(x)\mathbf{e}_{n+1} = x\mathbf{P}(x)$ .

# Gauss Quadrature: Computing the Nodes and Weights

Orthogonal polynomials satisfy a recurrence relation

$$\gamma_k P_{k-1}(x) + \beta_k P_k(x) + \gamma_{k+1} P_{k+1}(x) = x P_k(x).$$

We can write this in matrix form as

$$\begin{pmatrix} \beta_0 & \gamma_1 & & & \\ \gamma_1 & \beta_1 & \gamma_2 & & \\ & \gamma_2 & \beta_2 & \gamma_3 & \\ & & \ddots & \ddots & \ddots \\ & & & \gamma_n & \beta_n \end{pmatrix} \begin{pmatrix} P_0(x) \\ P_1(x) \\ P_2(x) \\ \vdots \\ P_n(x) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ \gamma_{n+1} P_{n+1}(x) \end{pmatrix} = x \begin{pmatrix} P_0(x) \\ P_1(x) \\ P_2(x) \\ \vdots \\ P_n(x) \end{pmatrix}$$

or equivalently  $T\mathbf{P}(x) + \gamma_{n+1}P_{n+1}(x)\mathbf{e}_{n+1} = x\mathbf{P}(x)$ .

Thus the roots of  $P_{n+1}(x)$  are the solution of a tridiagonal eigenvalue problem.

# Gauss Quadrature: Computing the Nodes and Weights

Orthogonal polynomials satisfy a recurrence relation

$$\gamma_k P_{k-1}(x) + \beta_k P_k(x) + \gamma_{k+1} P_{k+1}(x) = x P_k(x).$$

We can write this in matrix form as

$$\begin{pmatrix} \beta_0 & \gamma_1 & & & \\ \gamma_1 & \beta_1 & \gamma_2 & & \\ & \gamma_2 & \beta_2 & \gamma_3 & \\ & & \ddots & \ddots & \ddots \\ & & & \gamma_n & \beta_n \end{pmatrix} \begin{pmatrix} P_0(x) \\ P_1(x) \\ P_2(x) \\ \vdots \\ P_n(x) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ \gamma_{n+1} P_{n+1}(x) \end{pmatrix} = x \begin{pmatrix} P_0(x) \\ P_1(x) \\ P_2(x) \\ \vdots \\ P_n(x) \end{pmatrix}$$

or equivalently  $T\mathbf{P}(x) + \gamma_{n+1}P_{n+1}(x)\mathbf{e}_{n+1} = x\mathbf{P}(x)$ .

Thus the roots of  $P_{n+1}(x)$  are the solution of a tridiagonal eigenvalue problem. Once we have computed the nodes  $x_k$  we can compute the weights as

$$w_k = 2v_{1,k}^2$$

where  $v_{1,k}$  is the first entry of the eigenvector corresponding to eigenvalue (node)  $x_k$ .

## Aside: Richardson Extrapolation

Suppose we wish to approximate a quantity  $A$  by  $A_h$  where the error formula for  $A_h$  is known to be a polynomial in  $h$ :

$$A - A_h = a_1 h + a_2 h^2 + a_3 h^3 + \dots,$$

## Aside: Richardson Extrapolation

Suppose we wish to approximate a quantity  $A$  by  $A_h$  where the error formula for  $A_h$  is known to be a polynomial in  $h$ :

$$A - A_h = a_1 h + a_2 h^2 + a_3 h^3 + \dots ,$$

or equivalently

$$A = A_h + a_1 h + a_2 h^2 + a_3 h^3 + \dots .$$



## Aside: Richardson Extrapolation

Suppose we wish to approximate a quantity  $A$  by  $A_h$  where the error formula for  $A_h$  is known to be a polynomial in  $h$ :

$$A - A_h = a_1 h + a_2 h^2 + a_3 h^3 + \dots,$$

or equivalently

$$A = A_h + a_1 h + a_2 h^2 + a_3 h^3 + \dots.$$

We could also compute the approximation with a different value  $H$  and write

$$A = A_H + a_1 H + a_2 H^2 + a_3 H^3 + \dots.$$

## Aside: Richardson Extrapolation

Suppose we wish to approximate a quantity  $A$  by  $A_h$  where the error formula for  $A_h$  is known to be a polynomial in  $h$ :

$$A - A_h = a_1 h + a_2 h^2 + a_3 h^3 + \dots,$$

or equivalently

$$A = A_h + a_1 h + a_2 h^2 + a_3 h^3 + \dots.$$

We could also compute the approximation with a different value  $H$  and write

$$A = A_H + a_1 H + a_2 H^2 + a_3 H^3 + \dots.$$

We can then eliminate the first order error term by combining these results to get

$$A = \frac{HA_h - hA_H}{H - h} - a_2 hH - a_3 hH(h + H) + \dots.$$

## Aside: Richardson Extrapolation

Suppose we wish to approximate a quantity  $A$  by  $A_h$  where the error formula for  $A_h$  is known to be a polynomial in  $h$ :

$$A - A_h = a_1 h + a_2 h^2 + a_3 h^3 + \dots,$$

or equivalently

$$A = A_h + a_1 h + a_2 h^2 + a_3 h^3 + \dots.$$

We could also compute the approximation with a different value  $H$  and write

$$A = A_H + a_1 H + a_2 H^2 + a_3 H^3 + \dots.$$

We can then eliminate the first order error term by combining these results to get

$$A = \frac{HA_h - hA_H}{H - h} - a_2 hH - a_3 hH(h + H) + \dots.$$

Hence  $(HA_h - hA_H)/(H - h)$  is a higher order approximation to  $A$ .

# Romberg Integration

Romberg integration combines Richardson extrapolation with the composite trapezium rule.

# Romberg Integration

Romberg integration combines Richardson extrapolation with the composite trapezium rule. The idea is as follows:

Let  $T_h$  be the composite trapezium rule approximation to  $I(f)$  then we have

$$I(f) = T_h + a_2 h^2 + a_4 h^4 + \dots$$

$$I(f) = T_{h/2} + a_2 \frac{h^2}{4} + a_4 \frac{h^4}{16} + \dots$$

# Romberg Integration

Romberg integration combines Richardson extrapolation with the composite trapezium rule. The idea is as follows:

Let  $T_h$  be the composite trapezium rule approximation to  $I(f)$  then we have

$$I(f) = T_h + a_2 h^2 + a_4 h^4 + \dots$$

$$I(f) = T_{h/2} + a_2 \frac{h^2}{4} + a_4 \frac{h^4}{16} + \dots$$

Eliminating the  $\mathcal{O}(h^2)$  terms gives

$$I(f) = \frac{1}{3}(4T_{h/2} - T_h) - a_4 \frac{h^4}{4} + \dots$$

# Romberg Integration

Now recurse!

Let

$$T_h^{(2)} = \frac{1}{3}(4T_{h/2} - T_h)$$

so that

$$I(f) = T_h^{(2)} + a_4^{(2)}h^4 + \dots$$

Similarly

$$I(f) = T_{h/2}^{(2)} + a_4^{(2)}\frac{h^4}{16} + \dots$$

and thus we have

$$I(f) = \frac{1}{15}(16T_{h/2}^{(2)} - T_h^{(2)}) + a_6^{(3)}h^6 + \dots$$

# Romberg Integration

Note that if we compute  $T_h$  and  $T_{h/2}$  independently we will be recomputing lots of function values so we can be more clever.



# Romberg Integration

Note that if we compute  $T_h$  and  $T_{h/2}$  independently we will be recomputing lots of function values so we can be more clever.

Let  $R_{k,1}$  be the composite trapezium rule approximation with  $2^{k-1}$  subintervals. Let  $h_1 = b - a$  and  $h_k = h_1/2^{k-1}$ . Then

$$R_{1,1} = \frac{h_1}{2}(f(a) + f(b))$$

# Romberg Integration

Note that if we compute  $T_h$  and  $T_{h/2}$  independently we will be recomputing lots of function values so we can be more clever.

Let  $R_{k,1}$  be the composite trapezium rule approximation with  $2^{k-1}$  subintervals. Let  $h_1 = b - a$  and  $h_k = h_1/2^{k-1}$ . Then

$$\begin{aligned}R_{1,1} &= \frac{h_1}{2}(f(a) + f(b)) \\R_{2,1} &= \frac{h_2}{2}(f(a) + 2f(a + h_2) + f(b)) \\&= \frac{1}{2}(R_{1,1} + h_1 f(a + h_2))\end{aligned}$$

# Romberg Integration

Note that if we compute  $T_h$  and  $T_{h/2}$  independently we will be recomputing lots of function values so we can be more clever.

Let  $R_{k,1}$  be the composite trapezium rule approximation with  $2^{k-1}$  subintervals. Let  $h_1 = b - a$  and  $h_k = h_1/2^{k-1}$ . Then

$$\begin{aligned}R_{1,1} &= \frac{h_1}{2}(f(a) + f(b)) \\R_{2,1} &= \frac{h_2}{2}(f(a) + 2f(a + h_2) + f(b)) \\&= \frac{1}{2}(R_{1,1} + h_1 f(a + h_2)) \\&\vdots \\R_{k,1} &= \frac{1}{2} \left( R_{k-1,1} + h_{k-1} \sum_{i=1}^{2^{k-2}} f(a + (2i-1)h_k) \right) .\end{aligned}$$

# Romberg Integration

Applying Richardson extrapolation then gives

$$R_{k,2} = \frac{4R_{k,1} - R_{k-1,1}}{3} .$$

# Romberg Integration

Applying Richardson extrapolation then gives

$$R_{k,2} = \frac{4R_{k,1} - R_{k-1,1}}{3}.$$

This will have an error of size  $\mathcal{O}(h^4)$ . (In fact this yields the composite Simpson rule.)

# Romberg Integration

Applying Richardson extrapolation then gives

$$R_{k,2} = \frac{4R_{k,1} - R_{k-1,1}}{3}.$$

This will have an error of size  $\mathcal{O}(h^4)$ . (In fact this yields the composite Simpson rule.)

We can then continue with extrapolation to get a general formula

$$R_{k,n} = \frac{4^{n-1}R_{k,n-1} - R_{k-1,n-1}}{4^{n-1} - 1}$$

which has an error of size  $\mathcal{O}(h^{2^n})$ .

# Romberg Integration Example

Results for Romberg integration are often displayed in a so-called Romberg table of the form shown below for the integration

$$I = \int_0^{\pi} \sin(x) dx .$$

$R_{k,1}$	$R_{k,2}$	$R_{k,3}$	$R_{k,4}$
0.0000000000000000			
1.570796326794897	2.094395102393195		
1.896118897937040	2.004559754984421	1.998570731823836	
1.974231601945551	2.000269169948388	1.999983130945986	2.000005549979671
1.993570343772340	2.000016591047936	1.999999752454573	2.000000016288042
1.998393360970145	2.000001033369413	1.999999996190845	2.000000000059674
1.999598388640037	2.000000064530001	1.999999999940707	2.000000000000229
1.999899600184202	2.000000004032257	1.999999999999074	2.000000000000000