CHAPTER 8: Numerical Integration & Differentiation

Outline

- Numerical Integration
  - General form: $Q = \sum w_i f_i \approx \int f \, dx$.
  - Conditioning
  - Newton-Cotes: (midpoint, trapezoidal, Simpson)
  - Gauss quadrature & degree of quadrature rule
  - Composite trapezoidal rule
  - Richardson extrapolation
  - Tensor-product integration

- Numerical Differentiation
  - Conditioning
  - Finite differences
  - Derivative matrices

Numerical Differentiation Techniques

- Three common approaches for deriving formulas
  - Taylor series
  - Taylor series + Richardson extrapolation
  - Differentiate Lagrange interpolants
    - Readily programmed, see, e.g., Fornberg’s spectral methods text.

Using Taylor Series to Derive Difference Formulas

Taylor Series:

$$(1)\quad f_{j+1} \;=\; f_j + hf_j' + \frac{h^2}{2}f_j'' + \frac{h^3}{3!}f_j''' + \frac{h^4}{4!}f^{(4)}(\xi_+)$$

$$(2)\quad f_j \;=\; f_j$$

$$(3)\quad f_{j-1} \;=\; f_j - hf_j' + \frac{h^2}{2}f_j'' - \frac{h^3}{3!}f_j''' + \frac{h^4}{4!}f^{(4)}(\xi_-)$$

Approximation of $f_j' := f'(x_j)$:

$$\frac{1}{h}\left[(1)-(2)\right]: \quad \frac{f_{j+1}-f_j}{h} \;=\; f_j' + \frac{h}{2}f_j'' + h.o.t.$$

or

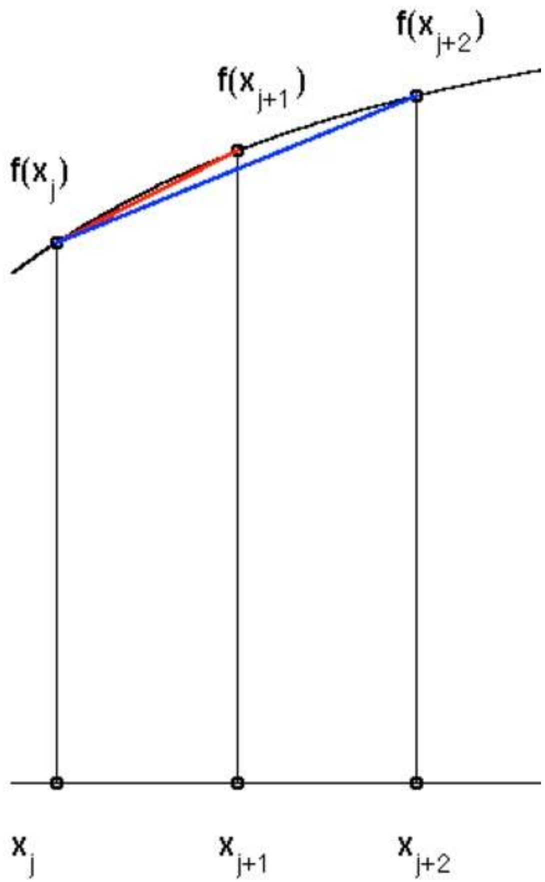
$$\frac{1}{2h}\left[(1)-(3)\right]: \quad \frac{f_{j+1}-f_{j-1}}{2h} \;=\; f_j' + \frac{h^2}{3!}f_j''' + h.o.t.$$

Richardson Extrapolation

- Formula for computing derivative at x_j

$$\delta_h : \quad \frac{f_{j+1} - f_j}{h} = f'_j + c_1 h + c_2 h^2 + c_3 h^3 + \cdots$$

$$\delta_{2h} : \quad \frac{f_{j+2} - f_j}{2h} = f'_j + c_1 2h + c_2 4h^2 + c_3 8h^3 + \cdots$$



$$\begin{aligned} 2\delta_h - \delta_{2h} &= \frac{4f_{j+1} - 4f_j}{2h} - \frac{f_{j+2} - f_j}{2h} \\ &= \frac{-3f_j + 4f_{j+1} - f_{j+2}}{2h} \\ &= f'_j + \tilde{c}_2 h^2 + \tilde{c}_3 h^3 + \cdots \end{aligned}$$

- Formula is improved from O(h) to O(h^2)

Stop Here

Finite Difference Example

- Suppose we wish to estimate $\left.\frac{d^2 f}{dx^2}\right|_{x_j}$ using function values $f_{j\pm 1} := f(x_j \pm h)$

- As before, to evaluate derivative *at* x_j, use the Taylor series expansion about x_j:

$$
\begin{aligned}
(1) \quad f_{j+1} &= f_j + hf_j' + \frac{h^2}{2}f_j'' + \boxed{\frac{h^3}{3!}f_j'''} + \frac{h^4}{4!}f_j^{(4)} + \frac{h^5}{5!}f_j^{(5)} + \frac{h^6}{6!}f^{(6)}(\xi_+) \\
(2) \quad f_j &= f_j \\
(3) \quad f_{j-1} &= f_j - hf_j' + \frac{h^2}{2}f_j'' - \boxed{\frac{h^3}{3!}f_j'''} + \frac{h^4}{4!}f_j^{(4)} - \frac{h^5}{5!}f_j^{(5)} + \frac{h^6}{6!}f^{(6)}(\xi_-)
\end{aligned}
$$

- If we add (1) & (3), we have:

$$f_{j-1} + f_{j+1} = 2f_j + h^2 f_j'' + \frac{h^4}{12}f_j^{(4)} + \frac{h^6}{6!}\left(f^{(6)}(\xi_-) + f^{(6)}(\xi_+)\right)$$

- Subtract $2 \times (2)$ and divide by h^2:

$$\frac{f_{j-1} - 2f_j + f_{j+1}}{h^2} = f_j'' + \underbrace{\frac{h^2}{12}f_j^{(4)} + O(h^4)}_{\textit{truncation error: } O(h^2)}$$

- This is the *central difference formula* for f_j'' with *uniform spacing*, $x_j = x_0 + j \cdot h$

- If the spacing is *nonuniform*, accuracy is only $O(h)$. (WHY?) *noncentral.m*

Richardson Example

- To get a higher order (e.g., $O(h^4)$) approximation to f_j'', we can derive a formula with a "5-point" stencil involving f_{j-2}, f_{j-1}, f_j, f_{j+1}, and f_{j+2}

- A cleaner approach for the uniform grid case is to once again apply Richardson extrapolation.

- Define

$$\frac{\delta_h^2 f_j}{\delta x^2} \;:=\; \frac{f_{j-1} - 2f_j + f_{j+1}}{h^2} \;=\; f_j'' + c_2 h^2 + O(h^4),$$

  where c_2 is a constant independent of h (per preceding slide)

- With this definition, the formula for "$2h$" is

$$\frac{\delta_{2h}^2 f_j}{\delta x^2} \;:=\; \frac{f_{j-2} - 2f_j + f_{j+2}}{(2h)^2} \;=\; f_j'' + c_2 (2h)^2 + O(h^4),$$

- To annihilate the $O(h^2)$ term we take $4\times$ the first equation minus $1\times$ the second to yield

$$4\frac{\delta_h^2 f_j}{\delta x^2} - \frac{\delta_{2h}^2 f_j}{\delta x^2} \;:=\; 3f_j'' + O(h^4)$$

- Dividing by 3 gives the desired $O(h^4)$ formula

$$\frac{4}{3}\frac{\delta_h^2 f_j}{\delta x^2} - \frac{1}{3}\frac{\delta_{2h}^2 f_j}{\delta x^2} \;:=\; f_j'' + O(h^4)$$

Richardson Example, continued

- Notice that if we had a *noncentered* FD formula for f''_j then the leading order error term in $\frac{\delta^2_{2h} f_j}{\delta x^2}$ would be $O(h)$ and not $O(h^2)$.

- Consequently, the Richardson extrapolation weights would not be $\frac{4}{3}$ and $-\frac{1}{3}$, they should instead be 2 and -1

- If we use the wrong weights the convergence in this case is only $O(h)$

- If we use, however, 2 and -1, we can at least recover $O(h^2)$ accuracy.

rich_central.m

Finite Difference Properties

- Assuming that f is $k+p$ times differentiable in the neighborhood of x_j, then

$$\frac{d^k f_j}{dx^k} = \underbrace{\frac{\delta^k f_j}{\delta x^k}}_{FD\ approx.} + \underbrace{O(h^p)}_{truncation\ error} = \frac{\delta^k f_j}{\delta x^k} + c_p h^p \frac{d^{k+p} f_j}{dx^p} + O(h^{p+1})$$

- This formula will be *exact* whenever $f \in \mathbb{P}_{k+p-1}$ because $f^{(k+p)} \equiv 0$

- For example, on a uniform grid,

$$\frac{\delta_h^2 f_j}{\delta x^2} = f_j'' + \frac{h^2}{12} f_j^{(4)} + O(h^4) = f_j'' \qquad \forall f \in \mathbb{P}_3$$

- On a nonuniform grid,

$$\frac{\delta_h^2 f_j}{\delta x^2} = f_j'' + c_1 h f_j''' + O(h^2) = f_j'' \qquad \forall f \in \mathbb{P}_2$$

- Consequence is that we can derive FD formulas by differentiating the *unique* polynomial interpolant that pass through the relevant (x_{j+k}, f_{j+k}) pairs, which is particularly useful in the nonuniform case

Differentiation via Lagrange Polynomials

- Recall the polynomial interpolation matrix

$$\mathbf{J}_{ij} = l_j(\tilde{x}_i)$$

where the Lagrange cardinal functions satisfy $l_j(x_k) = \delta_{jk}$ for nodal points x_k, $k = 1, \ldots, n$

- The $(n-1)$th-order polynomial approximation to $f(\tilde{x}_i)$ is given by

$$\tilde{\mathbf{p}} = \mathbf{J}\mathbf{f}$$

- We can also define the *derivative matrix*,

$$\mathbf{D}_{ij} = l_j'(\tilde{x}_i)$$

which would yield $p'(\tilde{x}_i)$ as an approximation to $f'(\tilde{x}_i)$:

$$\tilde{\mathbf{p}}' = \mathbf{D}\mathbf{f}$$

- It is generally easier, however, to define a derivative matrix $\hat{\mathbf{D}}$, based on the **nodes** x_j rather than the (arbitrary) target interpolation points $\tilde{x}_i$

- One can then (exactly) interpolate the approximation to $f'(\tilde{x}_i)$, which is a polynomial of degree $n-2$ using $\mathbf{J}$.

- Thus, the arbitrary interpolation matrix is $\mathbf{D} = \mathbf{J}\hat{\mathbf{D}}$

Derivative Matrices via Lagrange Interpolants

- Consider

$$
\begin{aligned}
p(x) &= \sum_{j=1}^{n} l_j(x) f_j \\
p'(x) &= \sum_{j=1}^{n} \frac{dl_j}{dx} f_j \\
p'(x_i) &= \sum_{j=1}^{n} \left.\frac{dl_j}{dx}\right|_{x_i} f_j \;=\; \sum_{j=1}^{n} d_{ij} f_j \;\Longrightarrow\; \mathbf{p}' = \hat{D}\mathbf{f}.
\end{aligned}
$$

- Recall Lagrange cardinal polynomial,

$$
l_j(x) \;=\; \alpha_j (x - x_1)(x - x_2)\cdots(x - x_{j-1})(x - x_{j+1})\cdots(x - x_n),
$$

with

$$
\alpha_j \;:=\; \left[(x_j - x_1)(x_j - x_2)\cdots(x_j - x_{j-1})(x_j - x_{j+1})\cdots(x_j - x_n)\right]^{-1}.
$$

Derivative Matrices via Lagrange Interpolants

- Recall Lagrange cardinal polynomial,

$$l_j(x) \;=\; \alpha_j (x-x_1)(x-x_2)\cdots(x-x_{j-1})(x-x_{j+1})\cdots(x-x_n),$$

with

$$\alpha_j \;:=\; \left[(x_j-x_1)(x_j-x_2)\cdots(x_j-x_{j-1})(x_j-x_{j+1})\cdots(x_j-x_n)\right]^{-1}.$$

- Define the linear functions $g_i(x) := (x - x_i)$, such that

$$l_j(x) \;=\; \alpha_j \left[g_1\, g_2 \,\cdots\, g_{j-1}\, g_{j+1} \,\cdots\, g_n\right].$$

Note that $g_i' \equiv 1$ and $g_i(x_i) \equiv 0$.

- Differentiate $l_j(x)$ term-by-term:

$$\begin{aligned}
\frac{dl_j}{dx} \;=\;& \alpha_j \,[\; g_1'\; g_2 \;\cdots\; g_{j-1}\; g_{j+1} \;\cdots\; g_n \\
&+\; g_1\; g_2' \;\cdots\; g_{j-1}\; g_{j+1} \;\cdots\; g_n \\
&\qquad\qquad\vdots \\
&+\; g_1\; g_2 \;\cdots\; g_{j-1}\; g_{j+1} \;\cdots\; g_n' \;].
\end{aligned}$$

Derivative Matrices, continued

- If we now evaluate this expression at $x = x_i \neq x_j$, then every row drops out except for the ith one:

$$
\begin{aligned}
\left.\frac{dl_j}{dx}\right|_{x_i} &= \alpha_j \left[\, g_1 \; g_2 \; \cdots \; g_{i-1} \; g_i' \; g_{i+1} \; \cdots \; g_{j-1} \; g_{j+1} \; \cdots \; g_n \right] \\
&= \alpha_j \left[\, g_1 \; g_2 \; \cdots \; g_{i-1} \cdot 1 \cdot g_{i+1} \; \cdots \; g_{j-1} \; g_{j+1} \; \cdots \; g_n \right] \\
&= \frac{\alpha_j \left[\, g_1 \; g_2 \; \cdots \; g_{i-1} \cdot 1 \cdot g_{i+1} \; \cdots \; g_{j-1} \; g_j \; g_{j+1} \; \cdots \; g_n \right]}{g_j} \\
&= \frac{\alpha_j \left[\alpha_i^{-1}\right]}{g_j} \\
&= \frac{\alpha_j}{\alpha_i \; g_j} \\
&= \frac{\alpha_j}{\alpha_i \left(x_i - x_j\right)} \;\; =: \;\; D_{ij}
\end{aligned}
$$

- Here, because we are evaluating the functions at x_i, we have $g_i = 0$ (which is not present), and $g_j(x_i) = x_i - x_j$.

Derivative Matrices, continued

- To find D_{ii}, we use the fact that $\mathbf{Dp} = 0$ if $\mathbf{p} = [1\,1\,\ldots\,1]^T$ because the derivative of $p(x) := 1$ is identically zero.

- So, for each row i, we have

$$\sum_{j=1}^{n} D_{ij} \;=\; 0 \quad\Longrightarrow\quad D_{ii} \;=\; -\sum_{j\neq i} D_{ij}.$$

- **In summary:**

$$\begin{aligned}
\alpha_j &:= \left[(x_j-x_1)(x_j-x_2)\cdots(x_j-x_{j-1})(x_j-x_{j+1})\cdots(x_j-x_n)\right]^{-1} \\
D_{ij} &= \frac{\alpha_j}{\alpha_i\,(x_i-x_j)}, \quad i\neq j, \\
D_{ii} &= -\sum_{i\neq j} D_{ij}.
\end{aligned}$$

- **As usual, this approach is stable for large n only if the x_is are Chebyshev, Gauss-Legendre, or other similar set of points.**

deriv_conv.m

Example: ODE-IVP

- We can use approximations to $\frac{df}{dt}$ to solve (numerically) the following ordinary differential equation (ODE), which is an *initial value problem* (IVP) for an unknown function $f(t)$,

$$\frac{df}{dt} = \lambda f(t),$$

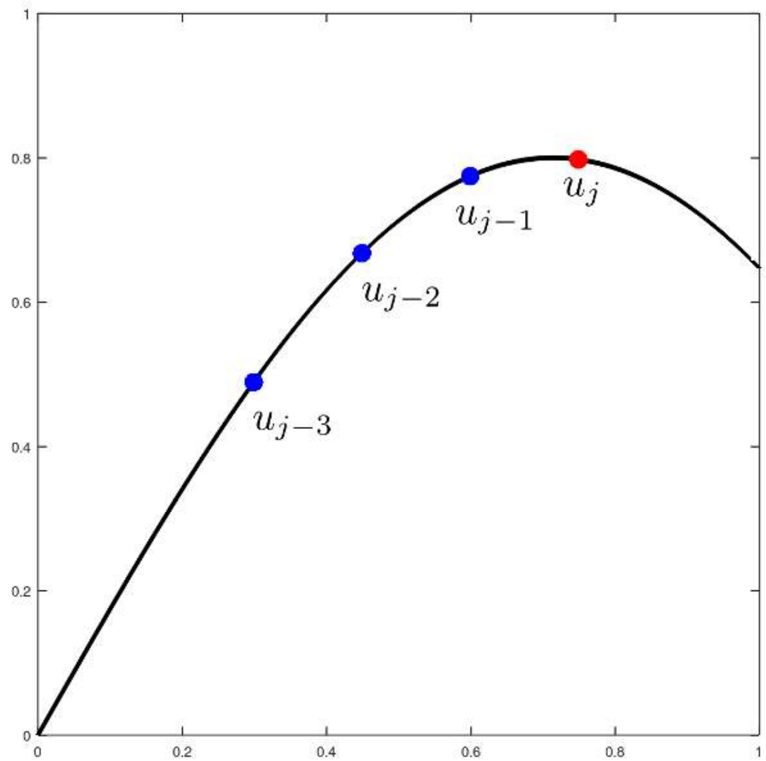
  with initial condition $f(t = 0) = f_0$

- For $\lambda = -2$ and $f_0 = 1$, the solution is $f(t) = e^{-2t}$, so the solution at $t = 1$ is $f(1) = e^{-2}$.

- Here, we will use kth-order *backward difference formulas*, BDFk, to approximate $\frac{df}{dt}$:

$$\begin{aligned}
\text{BDF1:} \quad & \frac{f_j - f_{j-1}}{h} = \lambda f_j + O(h) \\
\text{BDF2:} \quad & \frac{3f_j - 4f_{j-1} + f_{j-2}}{2h} = \lambda f_j + O(h^2) \\
\text{BDF3:} \quad & \frac{11f_j - 18f_{j-1} + 9f_{j-2} - 2f_{j-3}}{6h} = \lambda f_j + O(h^3)
\end{aligned}$$

kth-order Backward Difference Formula



- BDFk combines k *known* values from prior timesteps with the *unknown* value f_j to approximate the derivative at the new step (t_j).

- This approximation is then equated to the *rhs* of the ODE

- These schemes are *implicit* because the unknown appears on the *rhs*

ODE-IVP Example, continued

- Upon rearranging and dropping the $O(h)$ error term, BDF1 leads to the update formula

$$(1 - \lambda h) f_j \;=\; f_{j-1}$$

- For BDF1, we can start with f_0 and compute f_1, f_2, and so on.

- For BDF2, we have

$$(\frac{3}{2} - \lambda h) f_j \;=\; \frac{4}{2} f_{j-1} \;-\; \frac{1}{2} f_{j-2}$$

- For BDF2, we also need f_{-1} to get started, which we typically do not have.

- As a substitute, we can perform one step of BDF1 to get f_1 and then use f_1 and f_0 to move forward.

- And for BDF3,

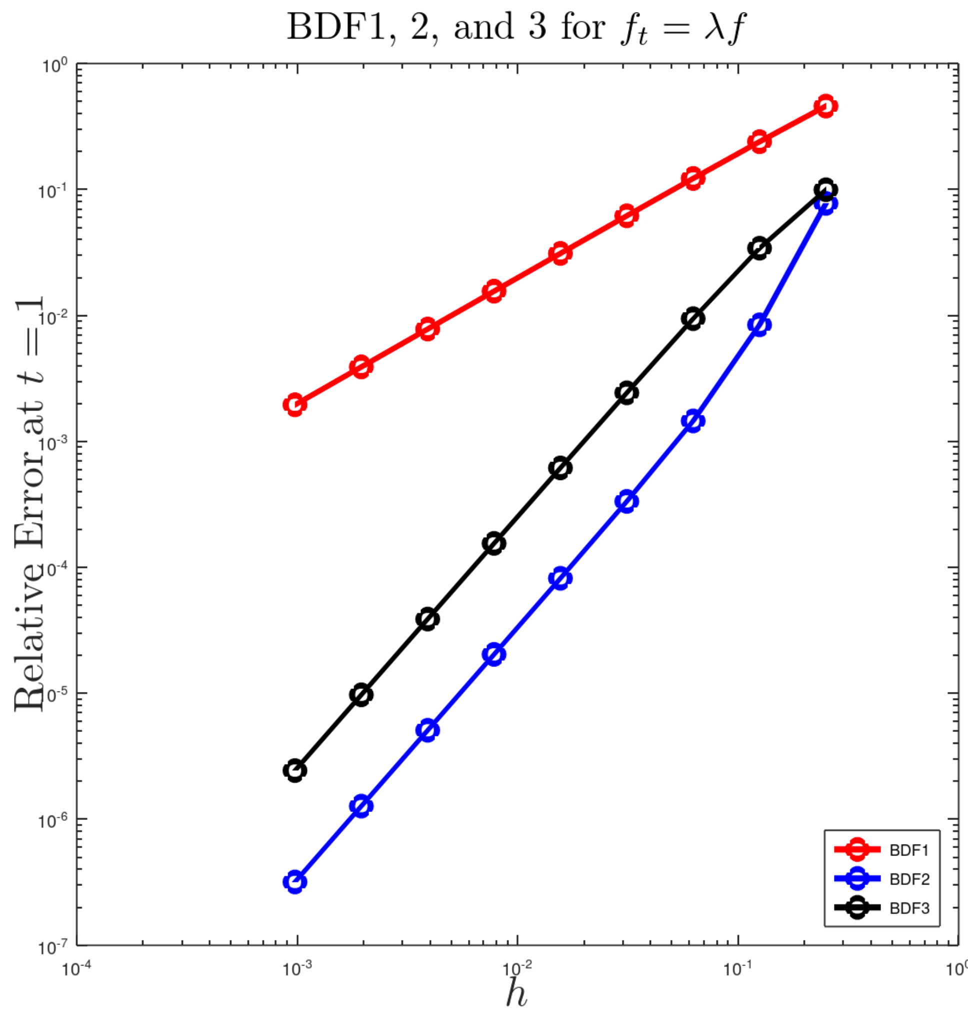
$$(\frac{11}{6} - \lambda h) f_j \;=\; \frac{18}{6} f_{j-1} \;-\; \frac{9}{6} f_{j-2} \;+\; \frac{2}{6} f_{j-3}$$

- For BDF3 we need to similarly bootstrap with one step of BDF1 followed by one step of BDF2.

ODE-IVP Example, continued

```
lambda = -2;  T =  1;   %% Final time

for bdf=1:3;
  kk=0;
  for k=2:10; kk=kk+1;

    nsteps=2^k; h=T/nsteps; lh = lambda*h;

    f3=0; f2=0; f1=0; f0=1;   %% INITIAL CONDITION

    for j=1:nsteps t=h*j;

        if j<2 || bdf==1;
           b0=1; b1=1; b2=0; b3=0;
        elseif j<3 || bdf==2;
           b0=1.5; b1=2; b2=-.5; b3=0;
        else
           b0=11/6; b1=3; b2=-9/6; b3=2/6;
        end;

        f3=f2; f2=f1; f1=f0;  % Shift old values off stack

        f0=(b1*f1+b2*f2+b3*f3)/(b0-lh);

    end;

    fe = exp(lambda*t);  %% Exact solution at time t

    ek (kk,bdf) = abs(fe-f0)/abs(fe);
    hk (kk,bdf) = h;

  end;
end;

loglog(hk(:,1),ek(:,1),'ro-',lw,2,...
       hk(:,2),ek(:,2),'bo-',lw,2,...
       hk(:,3),ek(:,3),'ko-',lw,2)
legend('BDF1','BDF2','BDF3','location','southeast')
axis square;
xlabel('$h$',intp,ltx,fs,24);
ylabel('Relative Error at $t=1$',intp,ltx,fs,24);
title('BDF1, 2, and 3 for $f_t = \lambda f$',intp,ltx,fs,22);
```

ode_ivp.m



- BDF3 shows only $O(h^2)$ error. WHY?
- Error on first step is already $O(h^2)$!

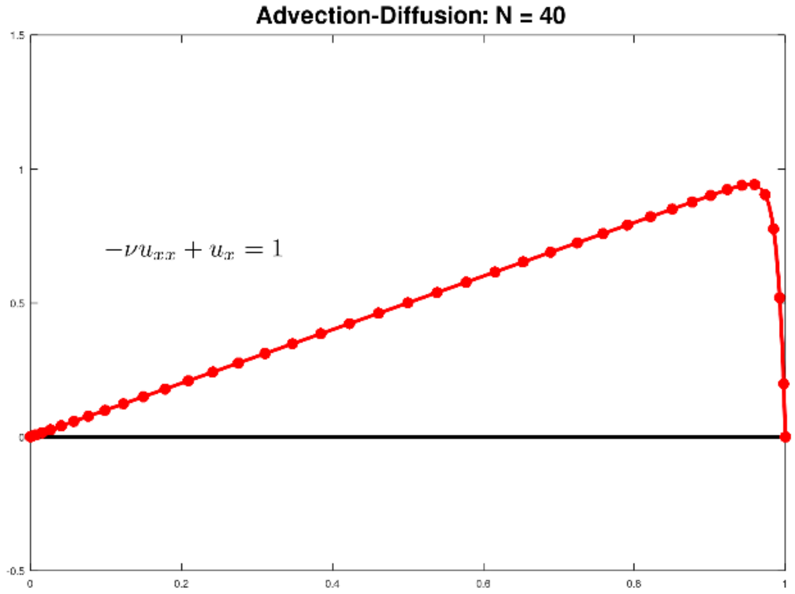
Example: BVP-ODE

- Here, we consider a *boundary value problem* (BVP), which has a single independent variable, x, and thus is also characterized as an ODE

- We'll take the example of an unknown function $\tilde{u}(x)$ for $x \in [0, 1]$ with prescribed boundary values $\tilde{u}(0) = 0$ and $\tilde{u}(1) = 0$.

- Let $u(x)$ satisfy

$$-\frac{d^2\tilde{u}}{dx^2} = f(x), \quad \tilde{u}(0) = \tilde{u}(1) = 0$$



BVP-ODE Example, continued

- We will approximate $\tilde{u}(x)$ by a Lagrange polynomial interpolant with unknown basis coefficents, u_j, $j = 0, \ldots, N$

$$\tilde{u}(x) \approx u(x) := \sum_{j=0}^{N} l_j(x) u_j$$

- In this *collocation* approach, we set $-u_i'' = f_i$, $i = 1, \ldots, N-1$ and $u_0 = u_N = 0$

- Discounting the boundary conditions, we have $N-1$ unknowns, $u_1, \ldots, u_{N-1}$, and $(n-1)$ equations associated with f_i, $i = 1, \ldots, N-1$.

- Because $u(x) \in \mathbb{P}_N$, we can compute its second derivative *exactly* with the derivative matrix described earlier.

- Define $\bar{\mathbf{D}}_{ij} = l_j(x_i)$ to be the $(N+1) \times (N+1)$ derivative matrix that is evaluated at *all* nodes, including x_0 and x_N.

- If $\bar{\mathbf{u}} = [u_0 \;\; u_1 \;\; \cdots u_N]^T$ is the vector of basis coeffients representing an Nth-order polynomial, then the vector

$$\bar{\mathbf{u}}'' = \bar{\mathbf{D}}^2 \bar{\mathbf{u}}$$

  is the vector of basis coeffients representing the second derivative of u

- Unfortunately, $\bar{\mathbf{D}}^2$ is not invertible. (WHY?)

- ANS: $\bar{\mathbf{D}}\mathbf{1} = 0$

- Fortunately, we only need rows 1 to $N-1$ of $\bar{\mathbf{D}}^2$ because the differential equation applies only at those rows.

- Moreover, because $u_0 = u_N = 0$, we do not need column 0 nor column N of $\bar{\mathbf{D}}^2$

- Let $\mathbf{D}_{2,ij} := \bar{\mathbf{D}}^2_{ij}$ define the matrix comprising rows $i = 1 : N-1$ and columns $j = 1 : N-1$ of $\bar{\mathbf{D}}^2$, and let $\mathbf{u} = [u_1 \ \cdots u_{N-1}]^T$ be the vector of *interior* basis coeffients

- If $\mathbf{f} = [f_1 \ \cdots f_{N-1}]^T$ is the *rhs* data, then the collocation system for our 2-point BVP is

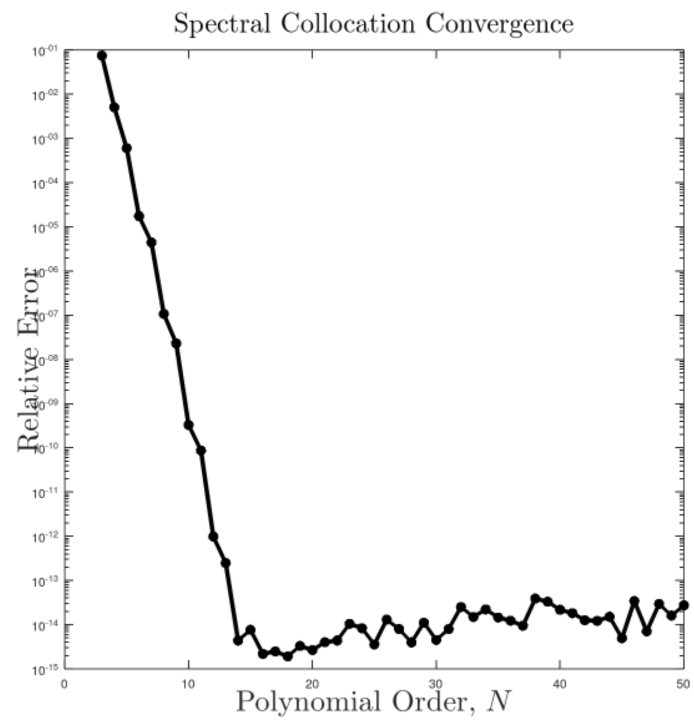
$$-\mathbf{D}_2\mathbf{u} \;=\; \mathbf{f}$$

- Note that $\mathbf{D}_2$ is not the square of (say) $\mathbf{D}$. It is the *interior* of the square of $\bar{\mathbf{D}}$

bvp_ode.m, bvp_ode2.m

BVP-ODE Example, continued

```
kk=0;
for N=3:50; kk=kk+1;

    [z,w]=zwglc(N); xb=.5*(z+1); % Chebyshev nodes

    Dh = deriv_mat(xb); D2 = Dh*Dh;
    A  = -D2(2:end-1,2:end-1);

    x = xb(2:end-1);
    k = 1;
    f = sin(k*pi*x);
    ue= f/(k*k*pi*pi); % Exact solution: f/(k*pi)^2;

    u = A\f;

    er(kk) = max(abs(ue-u))/max(abs(ue));
    en(kk) = N;

end;

semilogy(en,er,'k-',lw,2,en,er,'k.',ms,14);
axis square;
xlabel('Polynomial Order, $N$',intp,ltx,fs,24);
ylabel('Relative Error',intp,ltx,fs,24);
title('Spectral Collocation Convergence',intp,ltx,fs,22);
```



bvp_ode.m, bvp_ode2.m

Numerical Quadrature

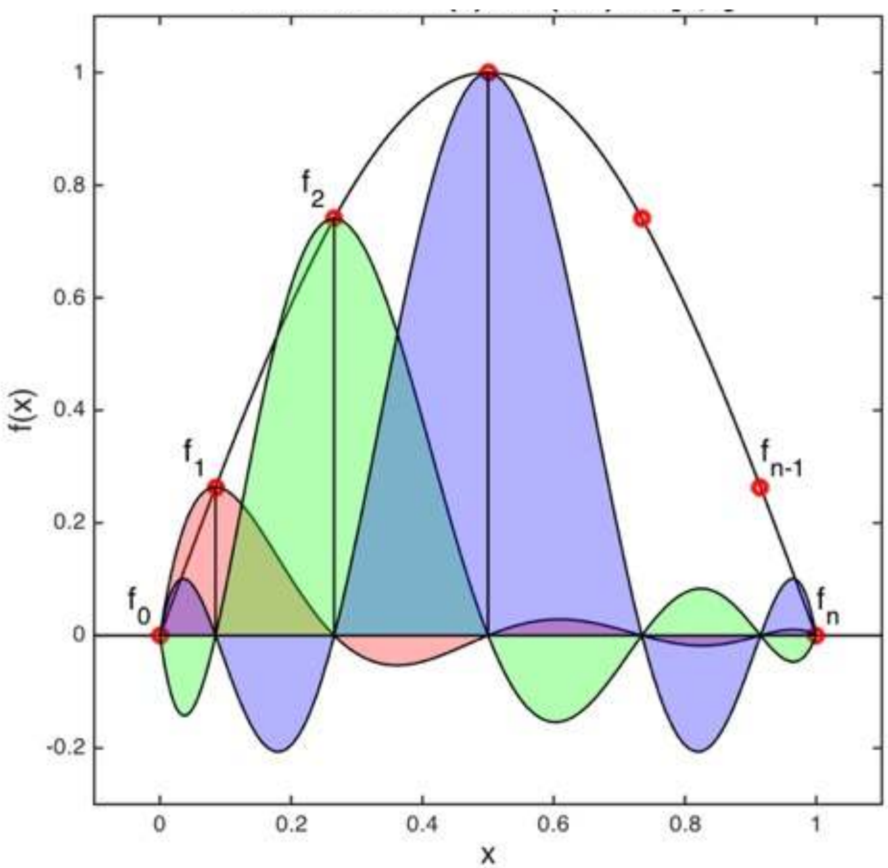
Numerical Quadrature

Numerical Quadrature

- *Quadrature* is the term used for approximating definite integrals of the form

$$\mathcal{I} = \int_a^b f(x)\,dx$$

- A *quadrature rule* is a weighted sum of a finite number of sample values of integrand function



Numerical Quadrature

- *Quadrature* is the term used for approximating definite integrals of the form

$$\mathcal{I} = \int_a^b f(x)\,dx$$

- A *quadrature rule* is a weighted sum of a finite number of sample values of integrand function

- Computational work is measured by number of evaluations of integrand function

- To obtain desired accuracy at low cost,
  - How should sample points be chosen?
  - How should weights be chosen?

Quadrature Rules

- An n-point quadrature rule has the form

$$Q_n(f) = \sum_{i=1}^{n} w_i f(x_i)$$

- Sorted points x_i are called nodes or quadrature points

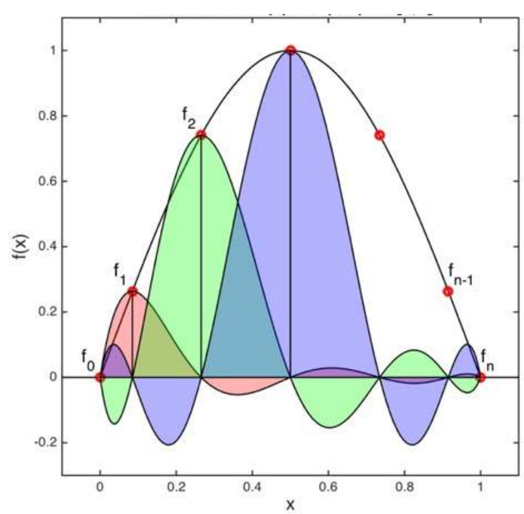
- Multipliers w_i are called weights

- Quadrature rule is
  - *open* if $a < x_1$ and $x_n < b$
  - *closed* if $a = x_1$ and $x_n = b$

Quadrature Rules, continued

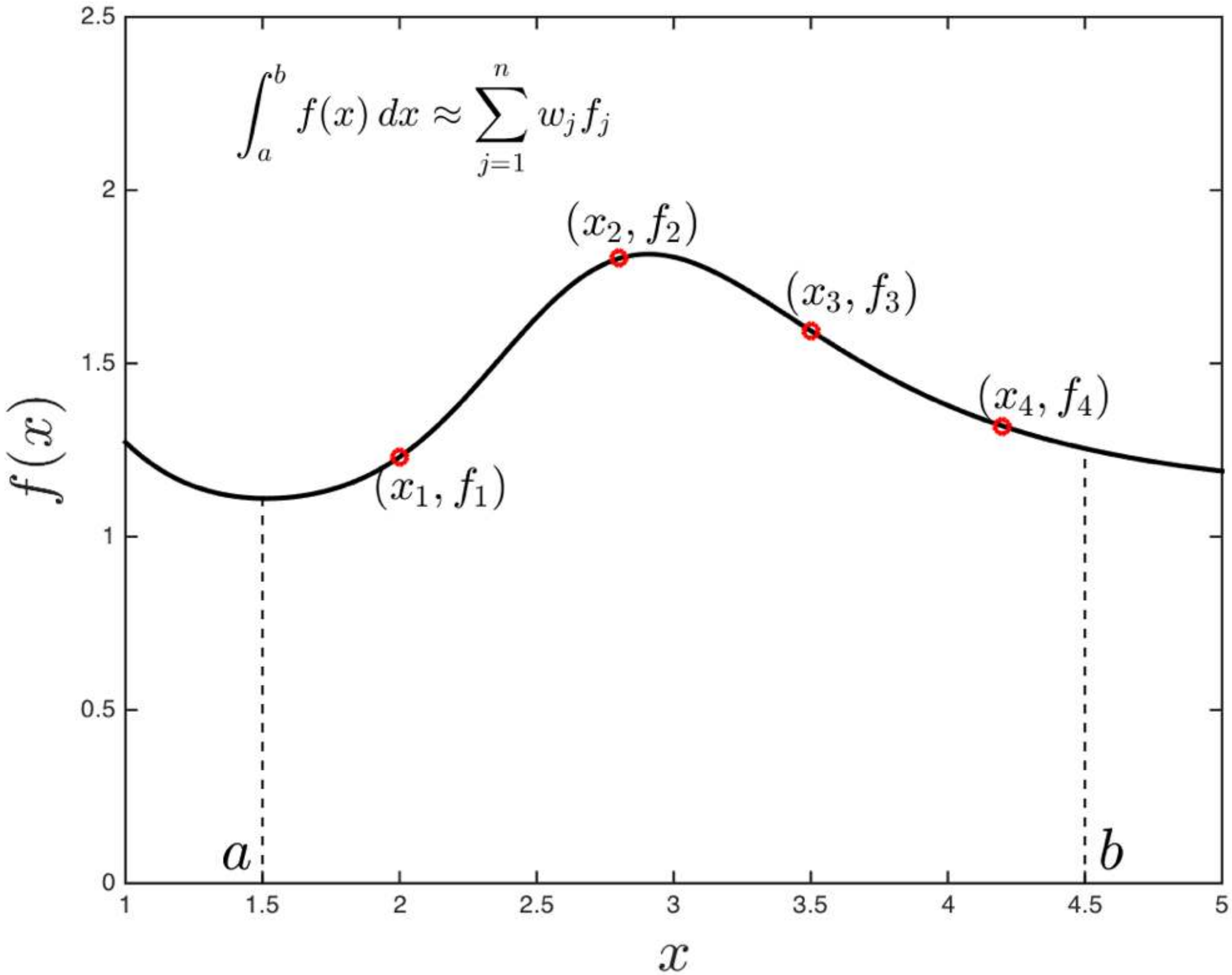
- Quadrature rules are based on polynomial interpolation
- Integrand function f is sampled at finite set of points
- Polynomial interpolating those points is determined
- Integral of interpolant is taken as estimate for integral of original function
- In practice, interpolating polynomial is not determined explicitly, but is used to determine weights corresponding to nodes
- If Lagrange interpolation is used, then the weights are

$$w_i = \int_a^b l_i(x)\,dx, \quad i = 1, \ldots, n$$



Quadrature Overview

- Choose *nodes* x_j, and *weights*, w_j, to approximate $\int_a^b f(x)\,dx$.



Quadrature Overview

$$I(f) \;:=\; \int_a^b f(t)\,dt \;\approx\; \sum_{i=1}^{n} w_i\, f_i, \;=:\; Q_n(f) \qquad f_i \;:=\; f(t_i)$$

- Idea is to minimize the number of function evaluations.
- *Small n is good.*
- Several strategies:
  - global rules
  - composite rules
  - composite rules + extrapolation
  - adaptive rules

Global (Interpolatory) Quadrature Rules

- Generally, approximate $f(t)$ by polynomial interpolant, $p(t)$.

$$f(t) \;\approx\; p(t) \;=\; \sum_{i=1}^{n} l_i(t)\, f_i$$

$$I(f) \;=\; \int_a^b f(t)\,dt \;\approx\; \int_a^b p(t)\,dt \;=:\; Q_n(f)$$

$$Q_n(f) \;=\; \int_a^b \left(\sum_{i=1}^{n} l_i(t)\, f_i \right) dt \;=\; \sum_{i=1}^{n} \left(\int_a^b l_i(t)\,dt \right) f_i \;=\; \sum_{i=1}^{n} w_i\, f_i.$$

$$\boxed{w_i \;=\; \int_a^b l_i(t)\,dt}$$

- We will see two types of global (interpolatory) rules:
  - Newton-Cotes — interpolatory on uniformly spaced nodes.
  - Gauss rules — interpolatory on optimally chosen point sets.

Method of Undetermined Coefficients

- Alternative derivation of quadrature rule uses *method of undetermined coefficients*
- To derive n-point rule on $[a, b]$ take nodes $x_1, \ldots, x_n$ as given and consider weights $w_1, \ldots, w_n$ as coefficients to be determined
- Force quadrature rule to be exact for first n polynomial basis functions (e.g., x^j, $j = 0, \ldots, n-1$)
- By linearity, rule will be exact for all $f \in \mathbb{P}_{n-1}$
- Thus, we obtain system of *moment equations* that determines weights for quadrature rule

Finding Weights: Method of Undetermined Coefficients

Find w_i for $[a,b] = [1,2]$, $n = 3$, with $t_1 = 1$, $t_2 = 3/2$, and $t_3 = 2$

- First approach: $f = 1,\ t,\ t^2$.

$$
\begin{aligned}
I(1) &= \sum_{i=1}^{3} w_i \cdot 1 = 1 \\
I(t) &= \sum_{i=1}^{3} w_i \cdot t_i = \left.\frac{1}{2} t^2 \right|_1^2 \\
I(t^2) &= \sum_{i=1}^{3} w_i \cdot t_i^2 = \left.\frac{1}{3} t^3 \right|_1^2
\end{aligned}
$$

Results in 3×3 matrix for the w_is.

Finding Weights: Method of Undetermined Coefficients

- Second approach: Choose f so that some of the coefficients multiplying the w_is vanish.

$$
\begin{aligned}
I_1 &= w_1(1-\frac{3}{2})(1-2) = \int_1^2 (t-\frac{3}{2})(t-2)\,dt \\
I_2 &= w_2(\frac{3}{2}-1)(\frac{3}{2}-2) = \int_1^2 (t-1)(t-2)\,dt \\
I_3 &= w_3(2-1)(2-\frac{3}{2}) = \int_1^2 (t-1)(t-\frac{3}{2})\,dt
\end{aligned}
$$

Corresponds to the Lagrange interpolant approach with 3 basis functions that span $\mathbb{P}_2(x)$:

$$
\begin{aligned}
h_1(x) &= (x-\frac{3}{2})(x-2) \\
h_2(x) &= (x-1)(x-2) \\
h_3(x) &= (x-1)(x-\frac{3}{2})
\end{aligned}
$$

Method of Undetermined Coefficients

Example 2: Find w_i for $[a, b] = [0, 1]$, $n = 3$, but using $f_i = f(t_i) = f(i)$, with t_i = -2, -1, and 0. (The t_i's are outside the interval (a, b).)

- Result should be exact for $f(t) \in \mathbb{P}_0$, $\mathbb{P}_1$, and $\mathbb{P}_2$.
- Take f=1, f=t, and f=t^2.

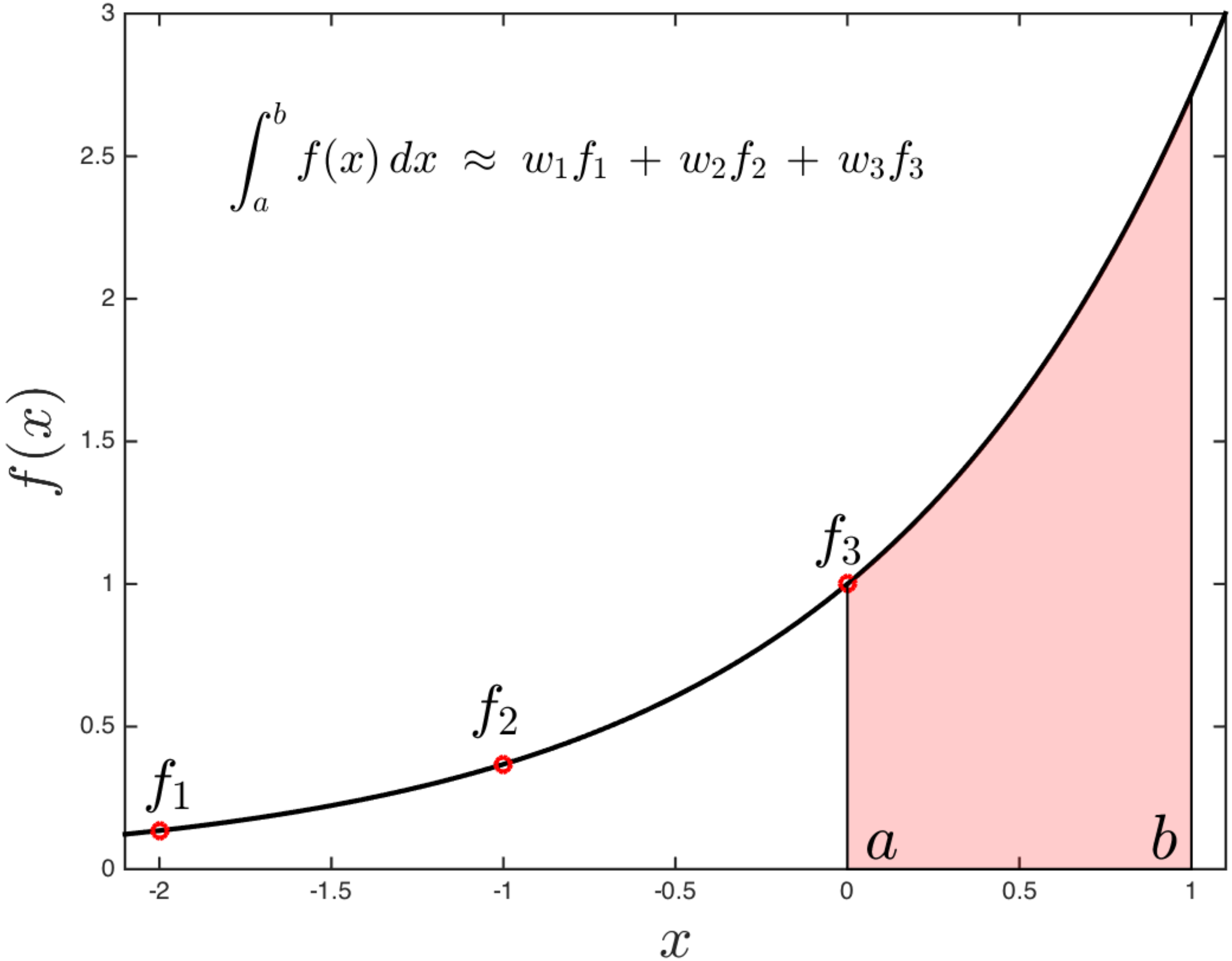
$$\begin{aligned} \sum w_i &= 1 = \int_0^1 1\,dt \\ -2w_{-2} - w_{-1} &= \frac{1}{2} = \int_0^1 t\,dt \\ 4w_{-2} + w_{-1} &= \frac{1}{3} = \int_0^1 t^2\,dt \end{aligned}$$

- Find

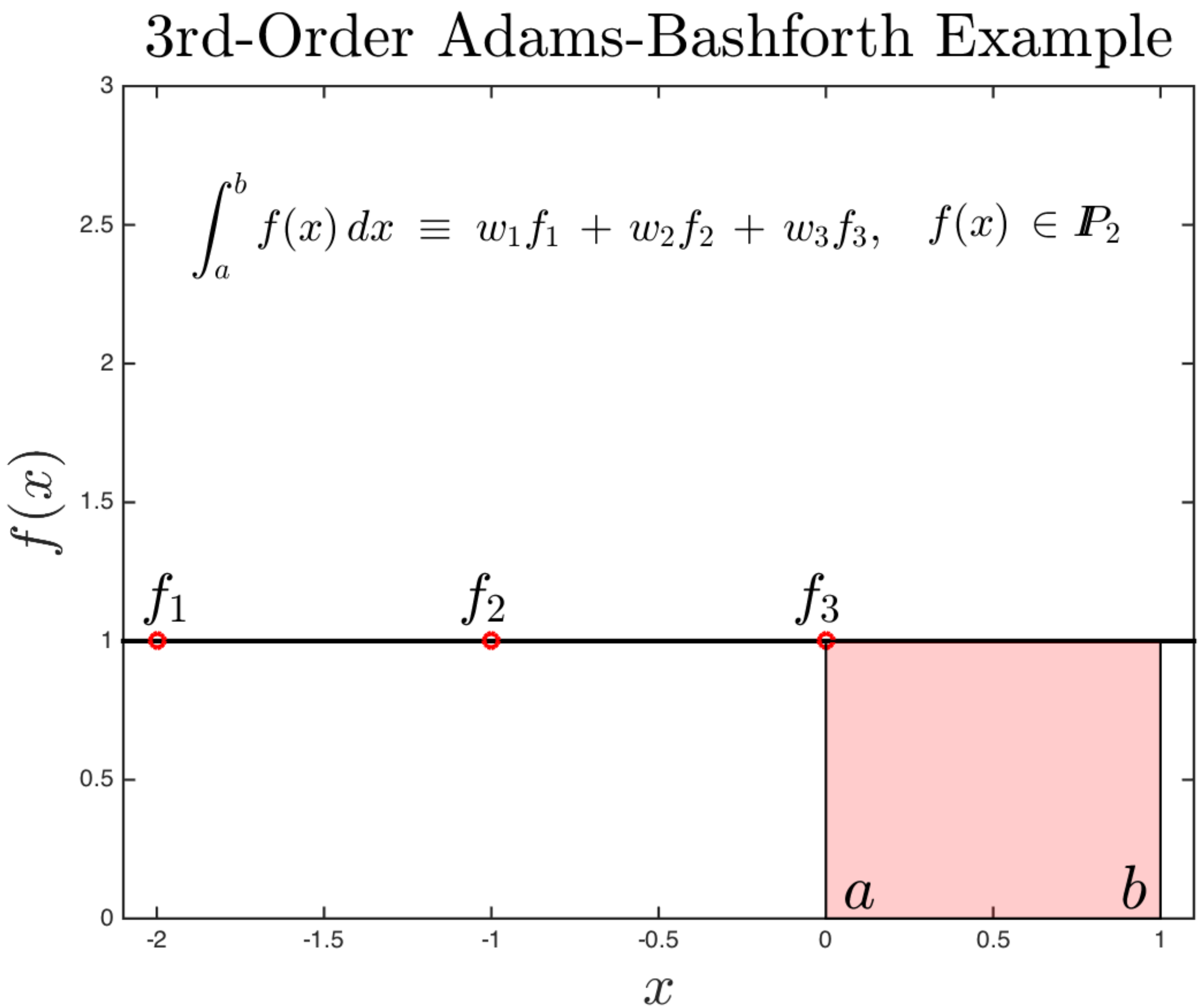
$$w_{-2} = \frac{5}{12} \qquad w_{-1} = -\frac{16}{12} \qquad w_0 = \frac{23}{12}.$$

- This example is useful for finding integration coefficients for explicit time-stepping methods that will be seen in later chapters.

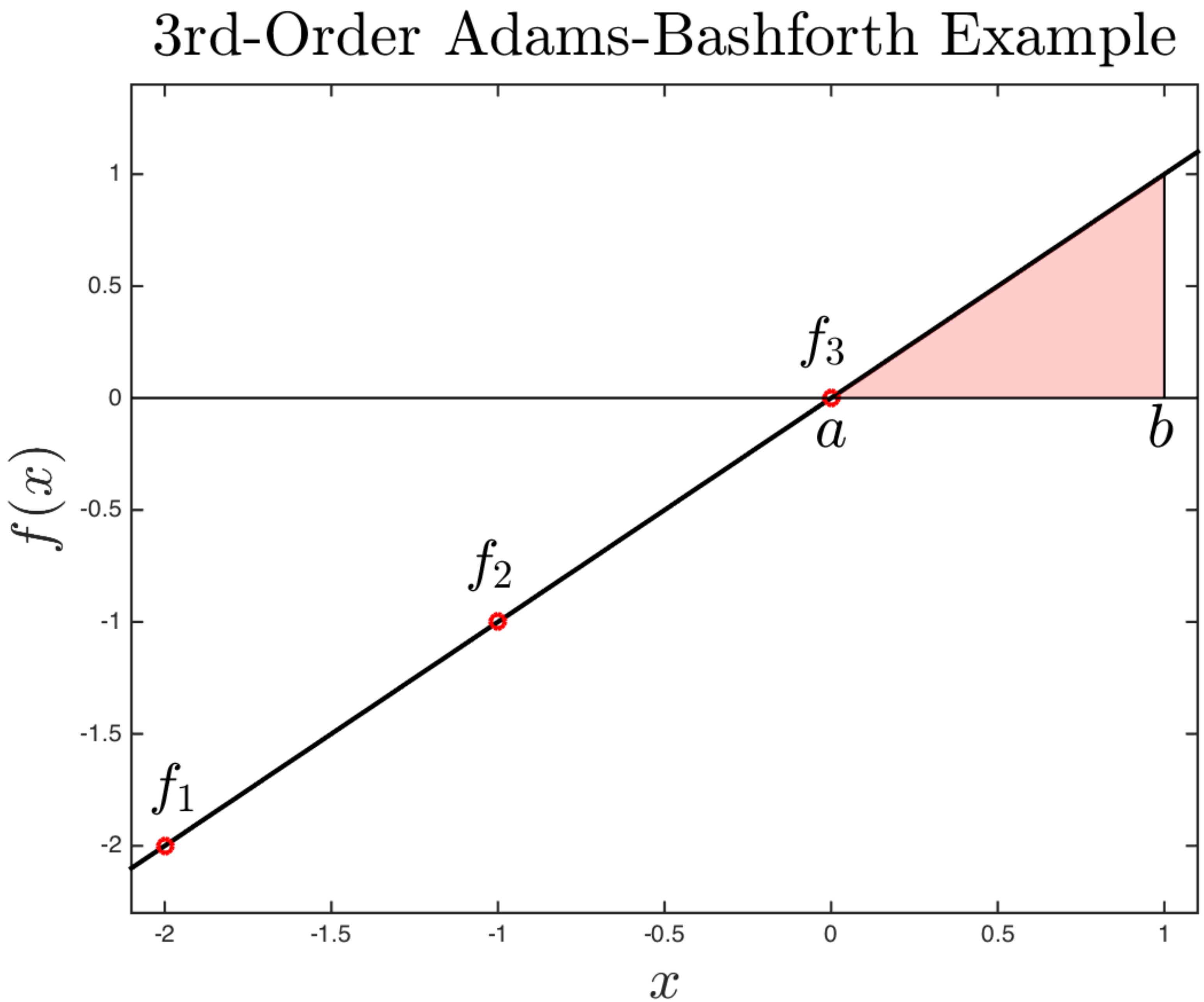
3rd-Order Adams-Bashforth Example



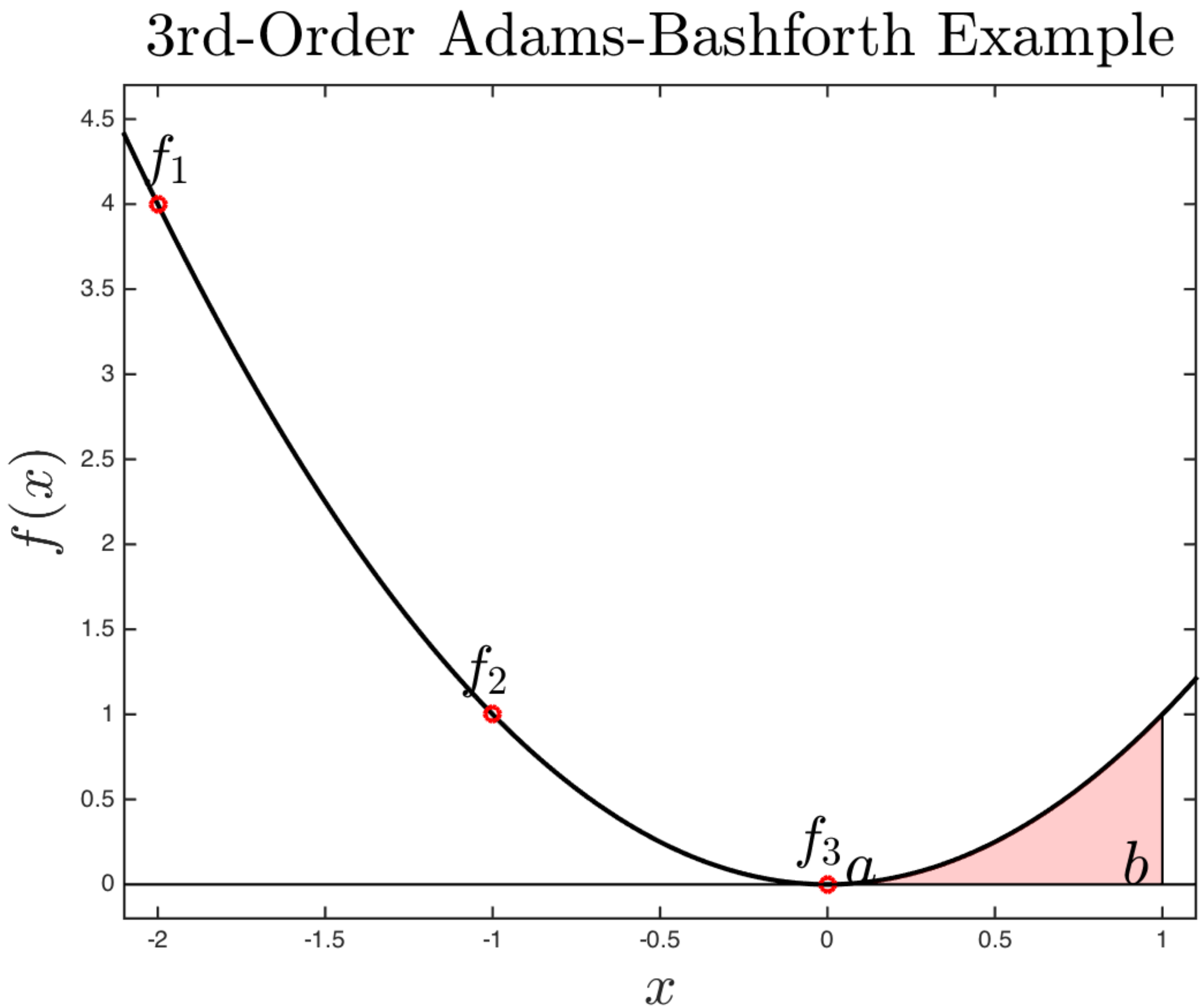
Method of Undetermined Coefficients



Method of Undetermined Coefficients



Method of Undetermined Coefficients



Method of Undetermined Coefficients

Example 2: Find w_i for $[a,b] = [0,1]$, $n = 3$, but using $f_i = f(t_i) = f(i)$, with t_i = -2, -1, and 0. (The t_i's are outside the interval (a,b).)

- $\boxed{\text{Result should be exact for } f(t) \in \mathbb{P}_0,\ \mathbb{P}_1, \text{ and } \mathbb{P}_2.}$
- Take $f{=}1$, $f{=}t$, and $f{=}t^2$.

$$
\begin{aligned}
\sum w_i &= 1 = \int_0^1 1\,dt \\
-2w_{-2} - w_{-1} &= \frac{1}{2} = \int_0^1 t\,dt \\
4w_{-2} + w_{-1} &= \frac{1}{3} = \int_0^1 t^2\,dt
\end{aligned}
$$

- Find

$$
w_{-2} = \frac{5}{12} \qquad w_{-1} = -\frac{16}{12} \qquad w_0 = \frac{23}{12}.
$$

Scale weights by h if (uniform) interval width is h.

Accuracy of Quadrature Rules

- Quadrature rule is of *degree* d if it is exact for every polynomial of degree d, but not exact for some polynomial of degree $d+1$

- By construction, n-point interpolatory rule is of degree at least $n-1$

- Rough error bound,

$$|I(f) - Q_n(f)| \leq \frac{1}{4}h^{n+1}\|f^{(n)}\|_\infty,$$

  where $h = \max\{x_{i+1} - x_i\}$, shows that $Q_n(f) \longrightarrow I(f)$ as $n \longrightarrow \infty$, provided $f^{(n)}$ remains well behaved

- Higher accuracy can be obtained by decreasing h

- If $f^{(n)}$ remains well behaved, can also increase n

Conditioning

- Absolute condition number of integration:

$$I(f) \;=\; \int_a^b f(t)\,dt$$

$$I(\hat{f}) \;=\; \int_a^b \hat{f}(t)\,dt$$

$$\left|I(f) - I(\hat{f})\right| \;=\; \left|\int_a^b (f - \hat{f})\,dt\right| \;\leq\; |b-a|\;||f - \hat{f}||_\infty$$

- Absolute condition number is $|b - a|$.

Conditioning

- Absolute condition number of quadrature:

$$\left|Q_n(f) - Q_n(\hat{f})\right| \;\leq\; \left|\sum_{i=1}^{n} w_i \left(f_i - \hat{f}_i\right)\right| \;\leq\; \sum_{i=1}^{n} |w_i| \max_i \left|f_i - \hat{f}_i\right|$$

$$\leq\; \sum_{i=1}^{n} |w_i| \; ||f - \hat{f}||_\infty$$

$$C \;=\; \sum_{i=1}^{n} |w_i|$$

- If $Q_n(f)$ is interpolatory, then $\sum w_i \;=\; (b-a)$:

$$Q_n(1) \;=\; \sum_{i=1}^{n} w_i \cdot 1 \;\equiv\; \int_a^b 1\, dt \;=\; (b-a).$$

- If $w_i \geq 0$, then $C \;=\; (b-a)$.
- Otherwise, $C > (b-a)$ and can be arbitrarily large as $n \longrightarrow \infty$.

Stopped Here

Newton-Cotes Quadrature

Newton-Cotes quadrature rules use equally spaced nodes in $[a, b]$

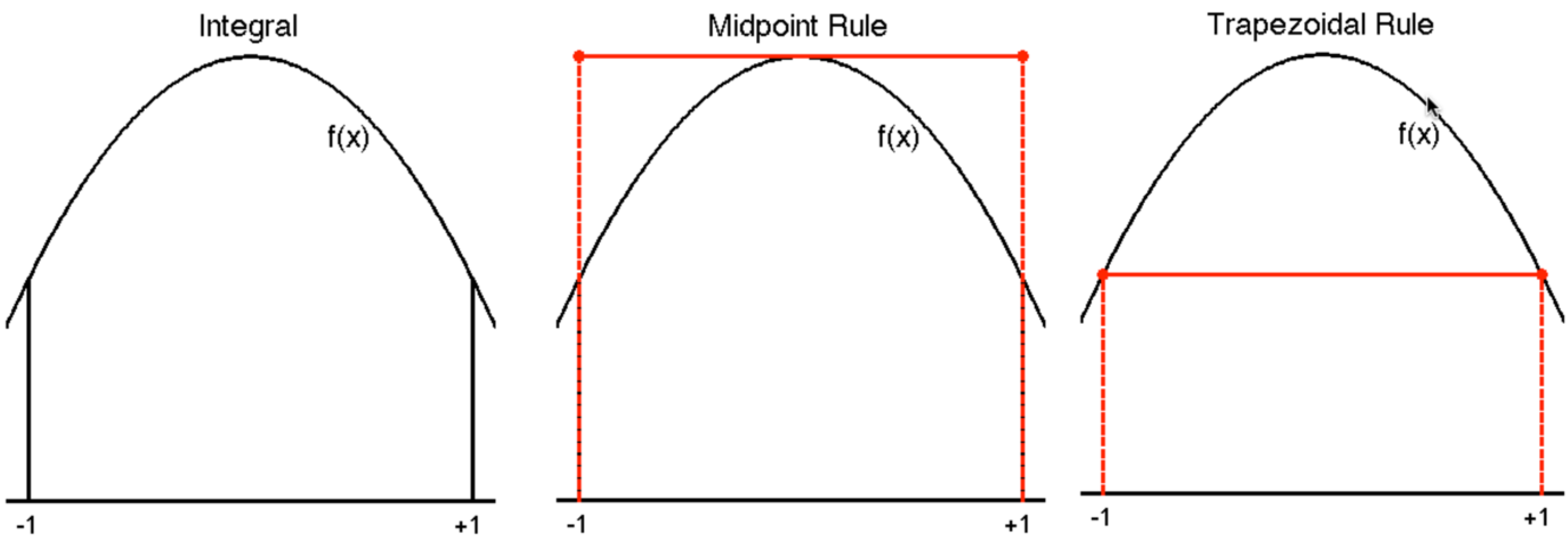
- *Midpoint rule* $\quad M(f) := (b-a)f(m), \quad m := (a+b)/2$

- *Trapezoidal rule* $\quad T(f) := \dfrac{b-a}{2} f\,(f(a) + f(b))$

- *Simpson's rule* $\quad S(f) := \dfrac{b-a}{6}\,(f(a) + 4f(m) + f(b))$

Example

$$
\begin{aligned}
f(x) &= 2 - x^2 \\
I(f) &= \int_{-1}^{1} f(x)\,dx = 2x - \left.\frac{x^3}{3}\right|_{-1}^{1} = 3\frac{1}{3}. \\
M(f) &= 2 \cdot f(0) = 2 \cdot 2 = 4. \qquad (|\text{error}|=2/3) \\
T(f) &= 2 \cdot \frac{f(-1) + f(1)}{2} = 2. \qquad (|\text{error}|=4/3)
\end{aligned}
$$



- Error for *midpoint rule* is generally ½ that of *trapezoidal rule.*

Quadrature Accuracy Example

- Recall the interpolatory-quadrature error bound,

$$|I(f) - Q_n(f)| \leq \frac{1}{4} h^{n+1} \, \|f^{(n)}\|_\infty$$

- Consider *trapezoidal rule*, $\displaystyle\int_a^b f(x)\,dx \approx h\frac{f(a)+f(b)}{2}$, with $b = a + h$

- Here, $n = 2$ as we are evaluating $f(x)$ at two points, so we would expect that the error is $O(h^3)$

- Let's take $f = \cos(x)$, for which $|f''| \leq 1$, and choose $a = 2$.

- The exact answer is $I = \sin(2+h) - \sin(2)$.

- `demo_trap`

Error for Midpoint Rule

- Define $m = \frac{a+b}{2}$

- Midpoint rule is $M(f) = (b-a)\,f(m) \approx I(f)$

- Assuming sufficiently smooth (i.e., differentiable) f, Taylor series about m,

$$f(x) \;=\; f(m) + (x-m)f'_m + \frac{(x-m)^2}{2}f''_m + \frac{(x-m)^3}{3!}f'''_m + \cdots$$

- Integrate from a to b,

$$\begin{aligned} I(f) \;&=\; (b-a)f(m) \;+\; 0\cdot f'_m \;+\; \frac{h^3}{12}f''_m \;+\; 0 \;+\; \frac{h^5}{1920}f^{(4)}_m \;+\; 0 \;+\; O(h^7) \\ &=\; M + \underbrace{\frac{h^3}{12}f''_m}_{E(f)} + \underbrace{\frac{h^5}{1920}f^{(4)}_m}_{F(f)} + O(h^7) \end{aligned}$$

- The leading-order error for the midpoint rule, $E(f)$, is $O(h^3)$

Error for Trapezoidal Rule

- Here, we need the Taylor series expansions for $f_a := f(a)$ and $f_b := f(b)$:

$$f(x) \;=\; f(m) + (x-m)f'_m + \frac{(x-m)^2}{2}f''_m + \frac{(x-m)^3}{3!}f'''_m + \cdots$$

$$f_a \;=\; f_m - \frac{h}{2}f'_m + \frac{h^2}{8}f''_m - \frac{h^3}{8\cdot 3!}f'''_m + \frac{h^4}{16\cdot 4!}f_m^{(4)} + \cdots$$

$$f_b \;=\; f_m + \frac{h}{2}f'_m + \frac{h^2}{8}f''_m + \frac{h^3}{8\cdot 3!}f'''_m + \frac{h^4}{16\cdot 4!}f_m^{(4)} + \cdots$$

- Apply to trapezoidal rule, $T(f)$:

$$
\begin{aligned}
T(f) &= h\left(\frac{f_a + f_b}{2}\right) \\
&= h f_m + \frac{h^3}{8} f''_m + \frac{h^5}{16 \cdot 4!} f_m^{(4)} + O(h^7) \\
&= M + \frac{h^3}{8} f''_m + \frac{h^5}{16 \cdot 4!} f_m^{(4)} + O(h^7) \\
&= (I(f) \; - \; E \; - \; F \; + \; O(h^7)) \; + \; 3E \; + \; 5F \; + \; O(h^7) \\
&= I(f) \; + \; 2E \; + \; 4F \; + \; \cdots
\end{aligned}
$$

- The leading-order error for the trapezoidal rule, $-2E(f)$, is $O(h^3)$

$$
I(f) \; = \; T(f) \; - \; 2E \; - \; 4F \; + \; \cdots
$$

Error, continued

- Can estimate the error by taking difference of midpoint and trapezoidal rules

$$I(f) = M(f) + E + F + \cdots$$

$$I(f) = T(f) - 2E - 4F + \cdots$$

$$T(f) = I(f) + 2E + 4F + \cdots$$

$$M(f) = I(f) - E - F + \cdots$$

$$T - M = 0 + 3E + 5E \approx 3E$$

$$E \approx \frac{T - M}{3}$$

Simpson's Rule

- Can use preceding results to annihilate the leading order E term (for which we now have an estimate!)

$$
\begin{aligned}
sum &= 2M(f) + T(f) \\
&= 3I(f) + 2F + \cdots
\end{aligned}
$$

$$
\begin{aligned}
S(f) &:= \frac{2M(f) + T(f)}{3} = \frac{2}{3}M(f) + \frac{1}{3}T \\
&= I(f) = \frac{2}{3}F + \cdots
\end{aligned}
$$

- *Error Model*:

$$
model = \frac{2}{3}\frac{h^5}{1920}|f_m^{(4)}|
$$

Accuracy of Newton-Cotes Quadrature

- Since n-point Newton-Cotes rule is based on polynomial of degree $n-1$, we expect it to have degree $n-1$

- Thus, we expect midpoint to have degree 0, trapezoid degree 1, and Simpson's to have degree 2

- From Taylor series expansion, error for midpoint rule depends on second and higher derivatives of integrand, which vanish for linear and constant polynomials

- So midpoint rule integrates linear polynomials exactly and degree is 1

- Similarly, Simpson's rule depends on 4th and higher derivatives, which vanish for all $f \in \mathbb{P}_3$, so Simpson's rule of degree 3

Accuracy of Newton-Cotes Quadrature

- In general, Newton-codes with odd number of points gains extra degree beyond that of polynomial interpolant on which it is based because odd functions (about the midpoint) contribute nothing to the integral or to the quadrature

- n-point Newton-Cotes rule is of degree $n-1$ if n is even but of degree n if n is odd

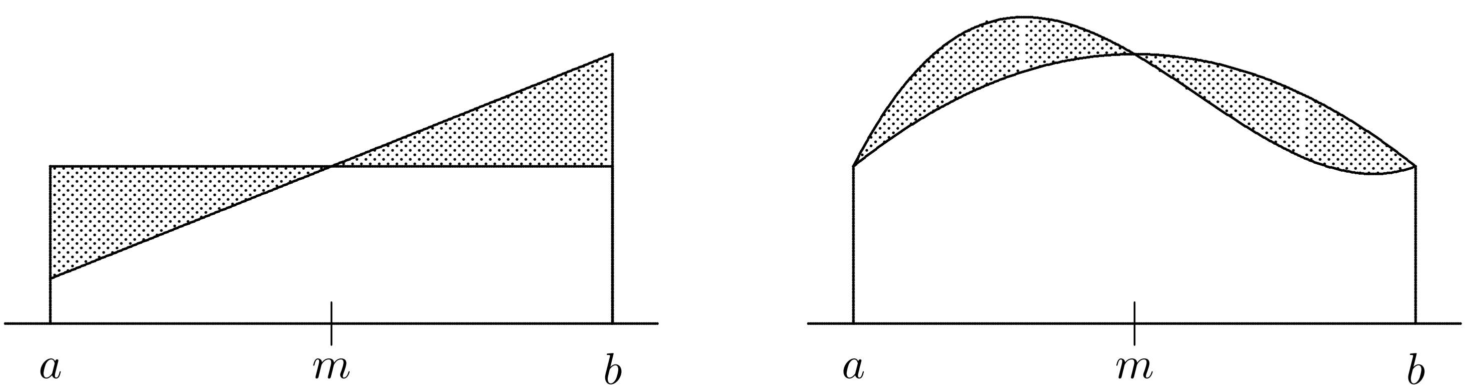


Figure 8.3: Cancellation of errors in midpoint (left) and Simpson (right) rules.

Drawbacks of Newton-Cotes Rules

- Newton-Cotes rules are simple and often effective, but they have drawbacks

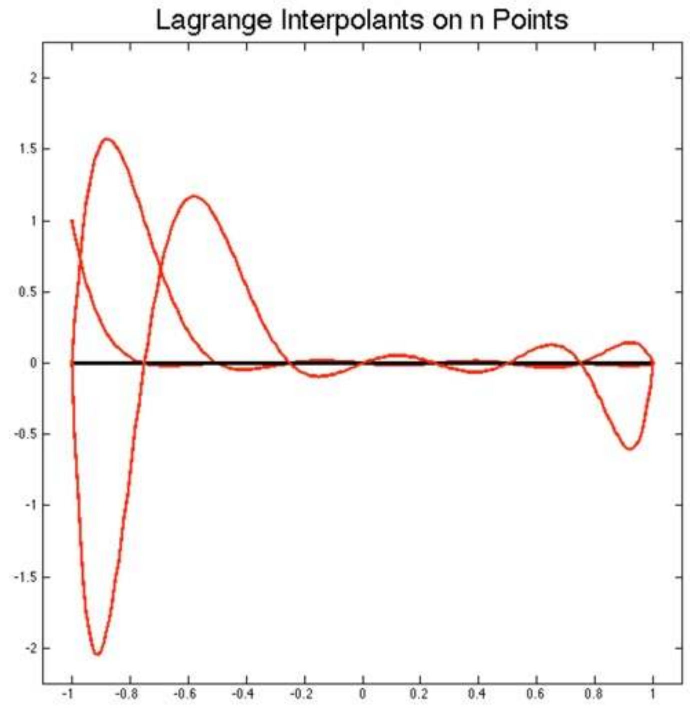
- For large n, behavior can be erratic because of the usual instabilities of high-order polynomial interpolation on uniform grids

- Moreover, for $n \geq 11$, every Newton-Cotes rule has at least one negative weight and $\sum_{i=1}^{n} |w_i| \longrightarrow \infty$ as $n \longrightarrow \infty$, so Newton-Cotes rules become arbitrarily ill-conditioned

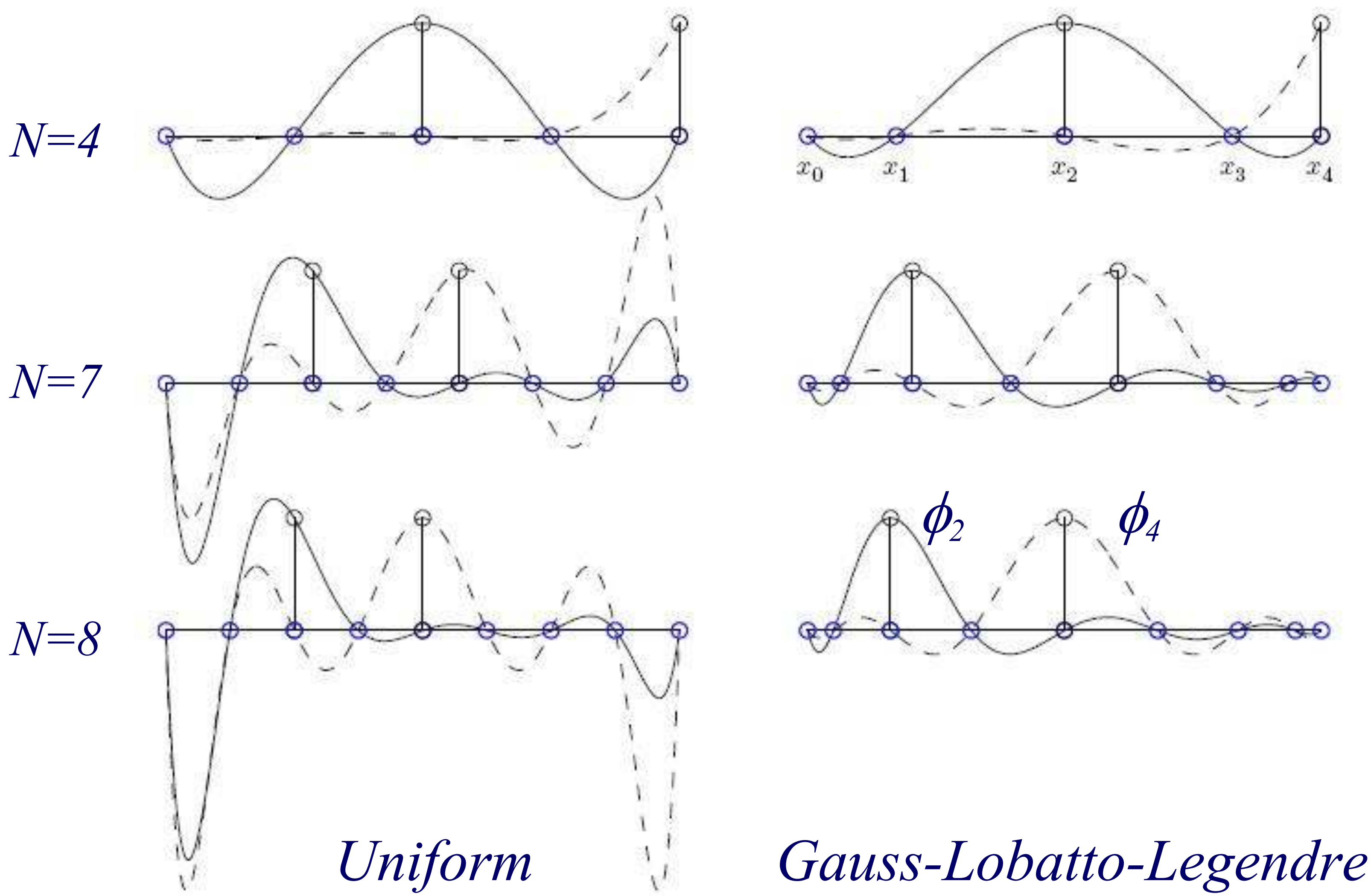
- Finally, Newton-Cotes rules do not realize the highest degree possible with n points

Newton-Cotes Formulae: What Could Go Wrong?

- Demo: newton_cotes.m
- Newton-Cotes formulae are interpolatory.
- For high n, Lagrange interpolants through uniform points are ill-conditioned.
- In quadrature, this conditioning is manifest through negative quadrature weights (bad).



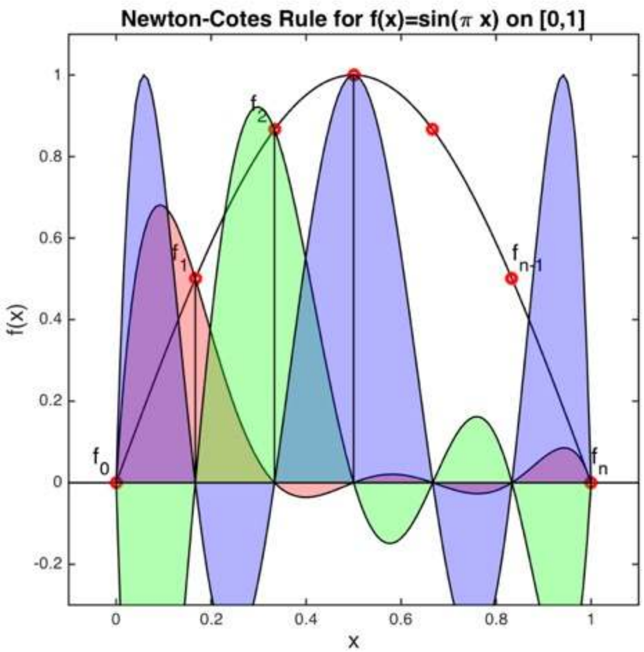
Lagrange Polynomials: Good and Bad Point Distributions



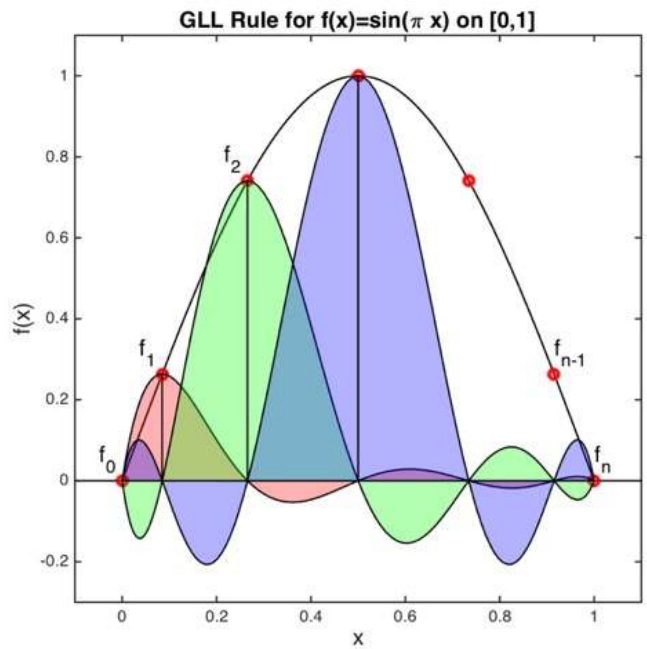
- ***We can see that for N=8 one of the uniform weights is close to becoming negative.***

Quadrature Rules: Stable and Unstable

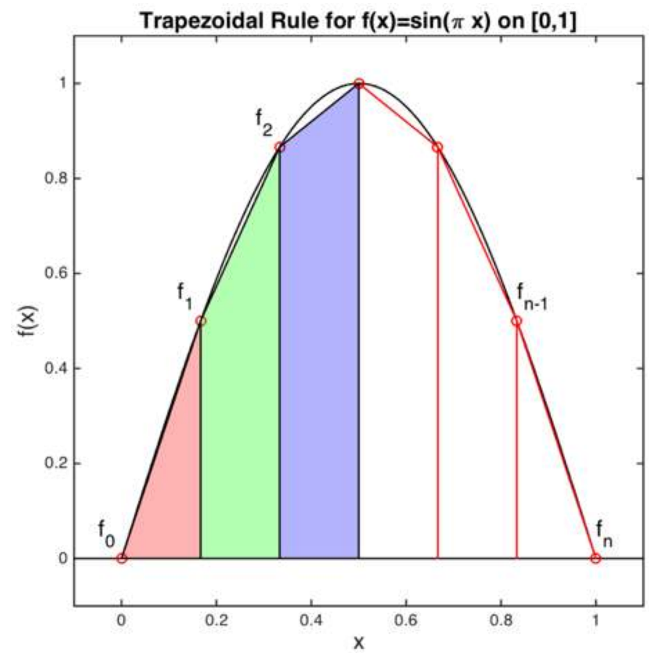
- Left – Unstable; Center – stable & rapid; Right – stable & $O(h^2)$



Newton-Cotes



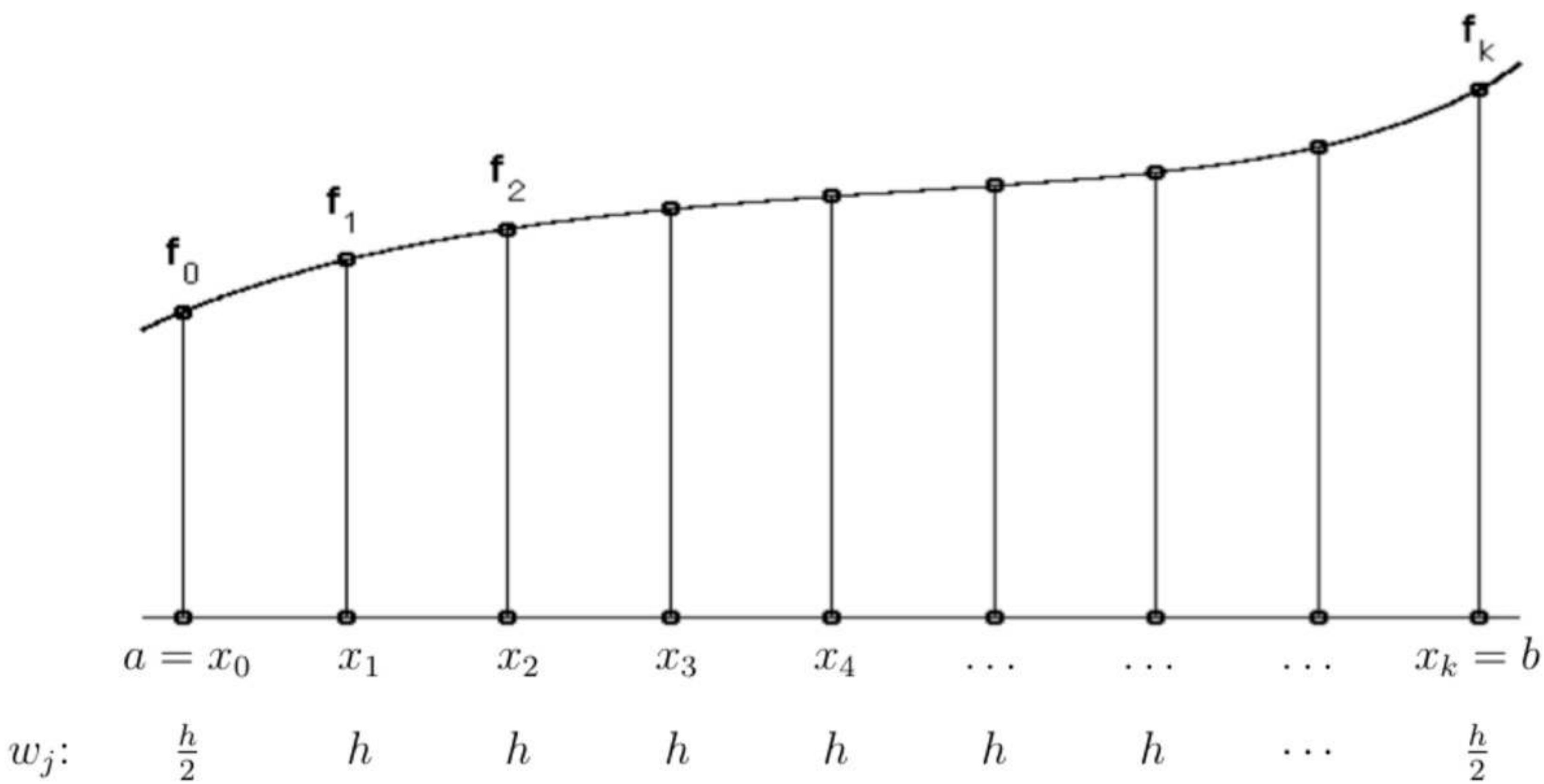
Gauss-Lobatto-Legendre

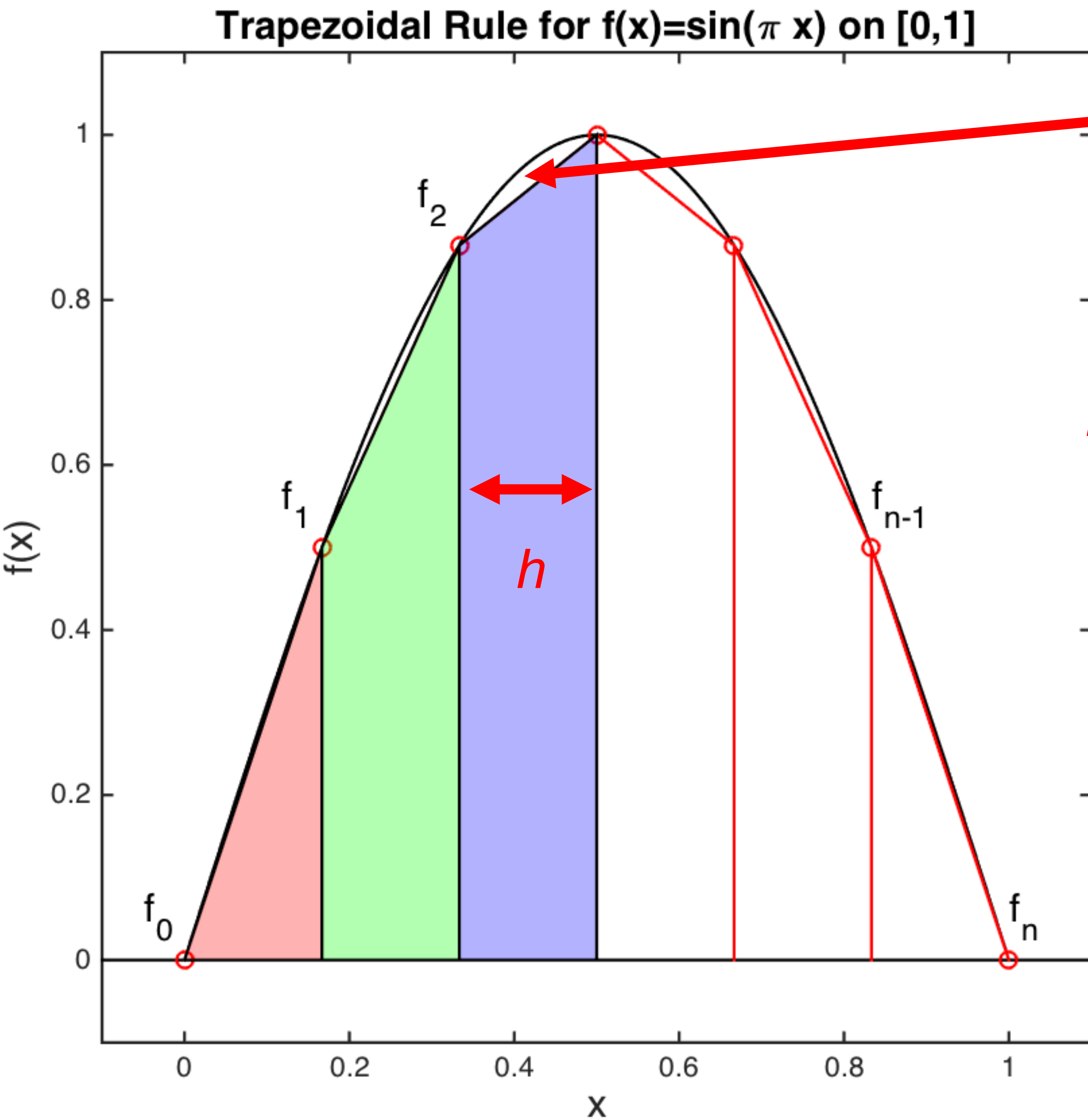


Composite-Trapezoidal

Composite Rules

- Main Idea: Use your favorite rule on each panel and sum across all panels.
- Particularly good if f(x) not differentiable at the knot points.





$O(h^2)$ x h
$= O(h^3)$

Cumulative Error:
$n \times O(h^3) =$
$(b-a)O(h^2)$

trap_v_gll_k

Composite Quadrature Rules

- Composite Trapezoidal (Q_{CT}) and Composite Simpson (Q_{CS}) rules work by breaking the interval [a,b] into panels and then applying either trapezoidal or Simpson method to each panel.

- Q_{CT} is the most common, particularly since Q_{CS} is readily derived via Richardson extrapolation at no extra work.

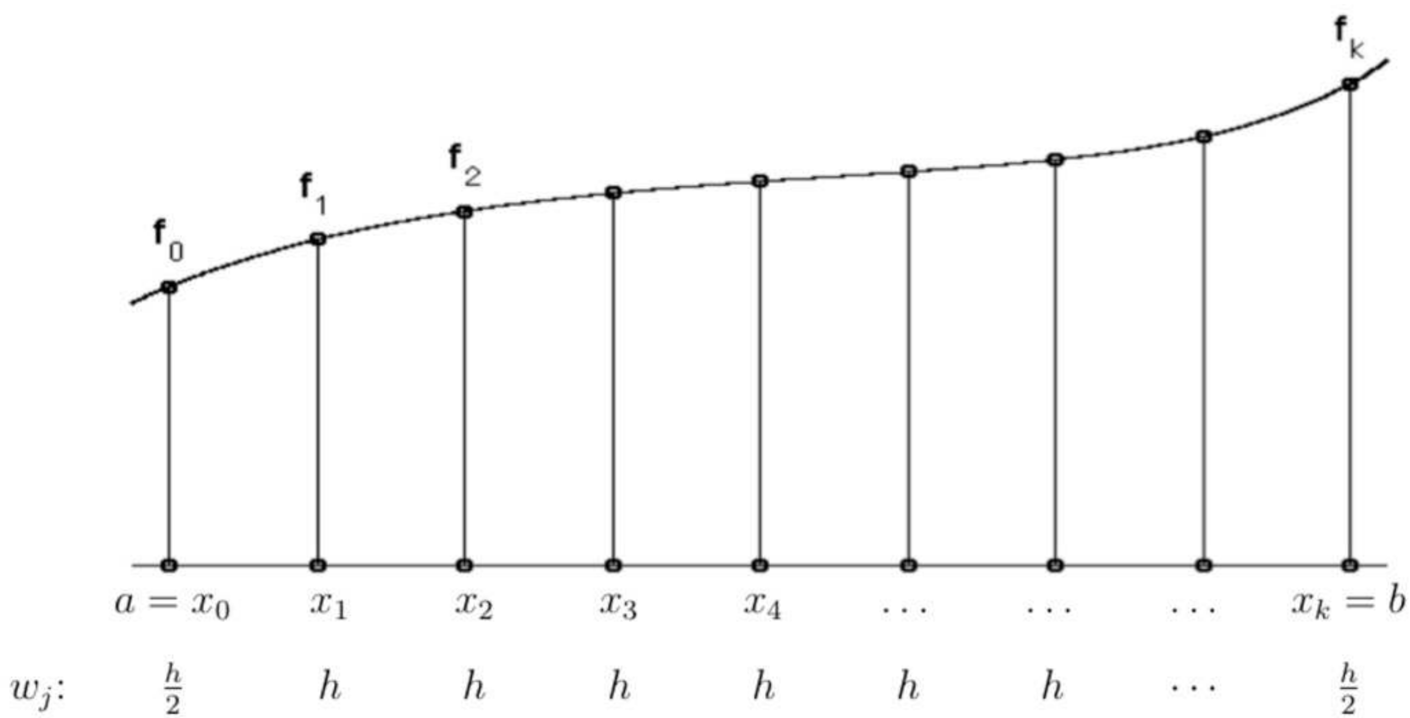
- Q_{CT} can be combined with Richardson extrapolation to get higher order accuracy, not quite competitive with Gauss quadrature, but a significant improvement.
-
- This combination is known as ***Romberg integration.***

- For functions that are periodic on [a,b], Q_{CT} ***is*** a Gauss quadrature rule.

Implementation of Composite Trapezoidal Rule

Assuming uniform spacing $h = (b-a)/k$,

$$
\begin{aligned}
Q_{CT} &:= \sum_{j=1}^{k} Q_j \;=\; \sum_{j=1}^{k} \frac{h}{2}(f_{j-1} + f_j) \\
&= \frac{h}{2} f_0 \,+\, h f_1 \,+\, h f_2 \,+\, \ldots \,+\, \ldots \,+\, h f_{k-1} \,+\, \frac{h}{2} f_k \\
&= \sum_{j=0}^{k} w_j\, f_j, \qquad\qquad w_0 = w_k = \frac{h}{2},\;\; w_j = h,\; \text{otherwise.}
\end{aligned}
$$

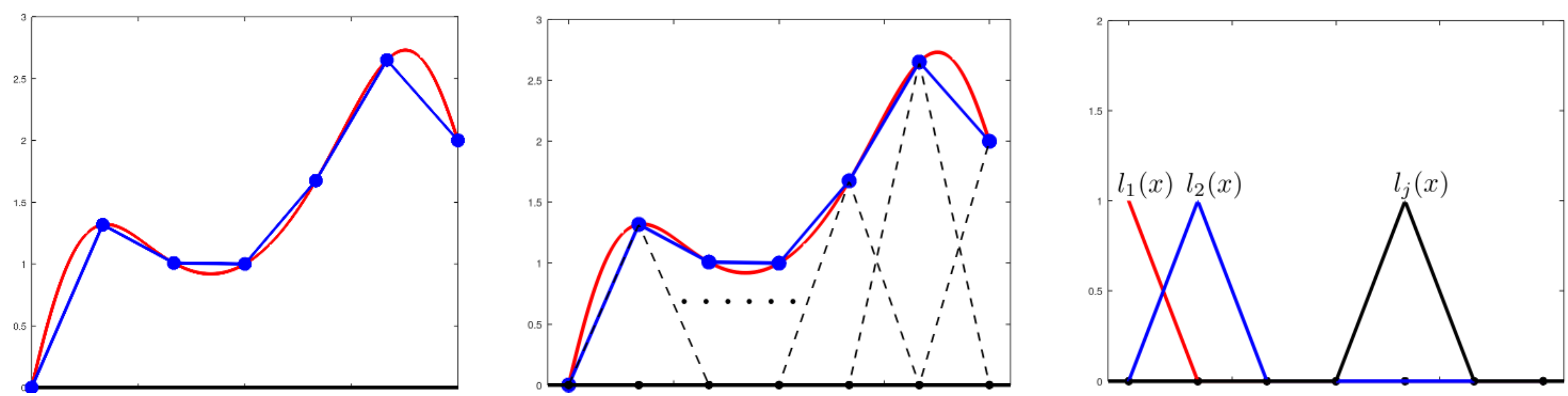


Composite Trapezoidal Rule

- Trapezoidal rule is also interpolatory, $Q_n(f) \;=\; \displaystyle\sum_{j=1}^{n} \int_a^b l_j(x) f_j \, dx$

- Lagrangian interpolants are piecewise linear "hat" functions

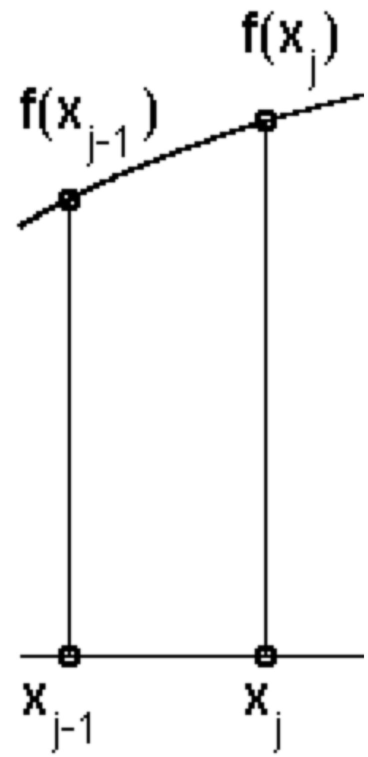
- On uniform grid with spacing h, $\int_a^b l_j(x)\, dx \;=\; h$ for *interior* basis functions, $j = 2, \ldots, n-1$, and $h/2$ for end-point basis functions $l_1(x)$ and $l_n(x)$



Error: Composite Trapezoidal Rule

$$
\begin{aligned}
I_j \; := \; \int_{x_{j-1}}^{x_j} f(x)\,dx \quad &= \quad \frac{h}{2}(f_{j-1} + f_j) \; + \; O(h^3) \\
&=: \quad Q_j \; + \; O(h^3) \\
&= \quad Q_j \; + \; c_j h^3 + \textit{higher order terms.}
\end{aligned}
$$

$$
c_j \leq \frac{1}{4} \max_{[x_{j-1},x_j]} |f''(x)|
$$



$$
I \; = \; \int_a^b f(x)\,dx \;\; = \;\; \underbrace{\sum_{j=1}^{n} Q_j}_{Q_{CT}} \; + \; \sum_{j=1}^{n} c_j h^3 \; + \; h.o.t.
$$

$$
|I - Q_{CT}| \; + \; h.o.t. = h^3 \sum_{j=1}^{n} c_j \; \leq \; h^3 n \max_j |c_j| \; = \; (b-a)h^2 \max_j |c_j|
$$

Composite Rule: Sum trapezoid rule across n panels:

$$
\begin{aligned}
Q_{CT} &:= h\left[\sum_{i=1}^{n-1} f_i + \frac{1}{2}(f_0 + f_n)\right] \\
&= \sum_{i=1}^{n} \frac{h}{2}[f_{i-1} + f_i] \\
&= \sum_{i=1}^{n} \left[\tilde{I}_i \,+\, c_2 h^3\, f''_{i-\frac{1}{2}} \,+\, c_4 h^5\, f^{iv}_{i-\frac{1}{2}} \,+\, c_6 h^7\, f^{vi}_{i-\frac{1}{2}} \,+\, \cdots\right] \\
&= \tilde{I} \,+\, c_2 h^2 \left[h \sum_{i=1}^{n} f''_{i-\frac{1}{2}}\right] \,+\, c_4 h^4 \left[h \sum_{i=1}^{n} f^{iv}_{i-\frac{1}{2}}\right] \,+\, \cdots \\
&= \tilde{I} \,+\, \frac{h^2}{12}\left[\int_a^b f''\,dx \,+\, \text{h.o.t.}\right] \,+\, c_4 h^4 \left[\int_a^b f^{iv}\,dx \,+\, \text{h.o.t.}\right] \,+\, \cdots \\
&= \tilde{I} \,+\, \frac{h^2}{12}\,[f'(b) - f'(a)] \,+\, O(h^4).
\end{aligned}
$$

- Global truncation error is $O(h^2)$ and has a particularly elegant form.
- Can estimate $f'(a)$ and $f'(b)$ with $O(h^2)$ accurate formula to yield $O(h^4)$ accuracy.
- With care, can also precisely define the coefficient for h^4, h^6, and other terms (Euler-Maclaurin Sum Formula).

Euler-Maclaurin Sum Formula

- Let $f \in C^6([a,b])$, and define $h = \frac{b-a}{n}$.
- Then the Euler–Maclaurin formula applied to the trapezoidal rule gives:

$$\int_a^b f(x)\,dx = T_n + \frac{h^2}{12}\left(f'(b) - f'(a)\right) - \frac{h^4}{720}\left(f^{(3)}(b) - f^{(3)}(a)\right) + \frac{h^6}{30240}\left(f^{(5)}(b) - f^{(5)}(a)\right) + R_3,$$

where the composite trapezoidal rule is:

$$T_n = \frac{h}{2}\left(f(a) + 2\sum_{j=1}^{n-1} f(a+jh) + f(b)\right),$$

- R_3 denotes the remainder after three correction terms.
- The coefficients arise from Bernoulli numbers:

$$B_2 = \frac{1}{6}, \quad B_4 = -\frac{1}{30}, \quad B_6 = \frac{1}{42}.$$

Examples.

- Apply (composite) trapezoidal rule for several endpoint conditions, $f'(a)$ and $f'(b)$:
  1. Standard case (nothing special).
  2. Lucky case ($f'(a) = f'(b) = 0$).
  3. Unlucky case ($f'(b) = -\infty$).
  4. Really lucky case ($f^{(k)}(a) = f^{(k)}(b)$, $k = 1, 2, \ldots$).

```
for kase=1:4;
for k=1:10; n=2^k; h=1/n; x=[0:n]'/n;

  if kase==1; f=exp(x);                         end;
  if kase==2; f=exp(x).*(1-cos(2*pi*x));        end;
  if kase==3; f=sqrt(1-x.*x);                   end;
  if kase==4; f=log(2+cos(2*pi*x));             end;

  w=1 + 0*x; w(1)=0.5; w(end)=0.5; w=h*w;

  Ih(k)=w'*f;

  if k>1; Id(k-1)=Ih(k-1)-Ih(k);   end;
  if k>2; Ir(k-2)=Id(k-2)/Id(k-1); end;
  hk(k)=h;

  if k>2; RATIO = Id(k-1)/Id(k-2); [n RATIO]; end;
```

- Functions on $[a, b] = [0, 1]$:

$$
\begin{aligned}
(1)\quad f(x) &= e^x \\
(2)\quad f(x) &= e^x\,(1 - \cos 2\pi x) \\
(3)\quad f(x) &= \sqrt{1 - x^2} \\
(4)\quad f(x) &= \log(2 + \cos 2\pi x).
\end{aligned}
$$

- `quad1.m` example.

Strategies to improve to $O(h^4)$ or higher?

- **Endpoint Correction.**
  - Estimate $f'(a)$ and $f'(b)$ to $O(h^2)$ using available f_i data.
  - **How?**
  - **Q:** What happens if you don't have at least $O(h^2)$ accuracy?
  - **-** Requires knowing the c_2 coefficient. :(

trap_endpoint.m

- **Richardson Extrapolation.**

$$
\begin{aligned}
I_h &= \tilde{I} + c_2 h^2 + O(h^4) \\
I_{2h} &= \tilde{I} + 4c_2 h^2 + O(h^4) \\
&\quad \text{(Reuses } f_i,\ i\text{=even!)} \\
I_R &= \left[\frac{4}{3} I_h - \frac{1}{3} I_{2h}\right] \\
&= I_{\text{SIMPSON}}\,!
\end{aligned}
$$

Composite Trapezoidal + Richardson Extrapolation

Can in fact show that if $f \in C^{2K+1}$ then

$$I \;=\; Q_{CT} + \tilde{c}_2 h^2 + \tilde{c}_4 h^4 + \tilde{c}_6 h^6 + \ldots + \tilde{c}_{2K} h^{2K} + O(h^{2K+1})$$

Suggests the following strategy:

$$\begin{aligned}
(1)\; I &= Q_{CT(h)} + \tilde{c}_2 h^2 + \tilde{c}_4 h^4 + \tilde{c}_6 h^6 + \ldots \\
(2)\; I &= Q_{CT(2h)} + \tilde{c}_2 (2h)^2 + \tilde{c}_4 (2h)^4 + \tilde{c}_6 (2h)^6 + \ldots
\end{aligned}$$

Take 4 $\times$(1)-(2) (*eliminate* $O(h^2)$ *term*):

$$\begin{aligned}
4I - I &= 4Q_{CT(h)} - Q_{CT(2h)} + c'_4 h^4 + c'_6 h^6 + \ldots \\
I &= \frac{4}{3} Q_{CT(h)} - \frac{1}{3} Q_{CT(2h)} + \hat{c}_4 h^4 + \hat{c}_6 h^6 + \ldots \\
&= Q_{S(2h)} + \hat{c}_4 h^4 + \hat{c}_6 h^6 + h.o.t.
\end{aligned}$$

Here, $\quad Q_{S(2h)} \;\equiv\; \dfrac{4}{3} Q_{CT(h)} - \dfrac{1}{3} Q_{CT(2h)}$

Composite Trapezoidal + Richardson Extrapolation

Can in fact show that if $f \in C^{2K+1}$ then

$$\boxed{I \;=\; Q_{CT} + \tilde{c}_2 h^2 + \tilde{c}_4 h^4 + \tilde{c}_6 h^6 + \ldots + \tilde{c}_{2K} h^{2K} + O(h^{2K+1})}$$

Suggests the following strategy:

$$\begin{aligned}
(1)\; I &= Q_{CT(h)} + \tilde{c}_2 h^2 + \tilde{c}_4 h^4 + \tilde{c}_6 h^6 + \ldots \\
(2)\; I &= Q_{CT(2h)} + \tilde{c}_2 (2h)^2 + \tilde{c}_4 (2h)^4 + \tilde{c}_6 (2h)^6 + \ldots
\end{aligned}$$

Take 4 $\times$(1)-(2) (*eliminate* $O(h^2)$ *term*):

$$\begin{aligned}
4I - I &= 4Q_{CT(h)} - Q_{CT(2h)} + c_4' h^4 + c_6' h^6 + \ldots \\
I &= \frac{4}{3} Q_{CT(h)} - \frac{1}{3} Q_{CT(2h)} + \hat{c}_4 h^4 + \hat{c}_6 h^6 + \ldots \\
&= Q_{S(2h)} + \hat{c}_4 h^4 + \hat{c}_6 h^6 + h.o.t.
\end{aligned}$$

Here, $\boxed{Q_{S(2h)} \;\equiv\; \frac{4}{3} Q_{CT(h)} - \frac{1}{3} Q_{CT(2h)}}$

Composite Trapezoidal + Richardson Extrapolation

Can in fact show that if $f \in C^{2K+1}$ then

$$I = Q_{CT} + \boxed{\tilde{c}_2 h^2} + \tilde{c}_4 h^4 + \tilde{c}_6 h^6 + \ldots + \tilde{c}_{2K} h^{2K} + O(h^{2K+1})$$

Original error – O(h²)

Suggests the following strategy:

$$\begin{aligned}
(1)\ I &= Q_{CT(h)} + \tilde{c}_2 h^2 + \tilde{c}_4 h^4 + \tilde{c}_6 h^6 + \ldots \\
(2)\ I &= Q_{CT(2h)} + \tilde{c}_2 (2h)^2 + \tilde{c}_4 (2h)^4 + \tilde{c}_6 (2h)^6 + \ldots
\end{aligned}$$

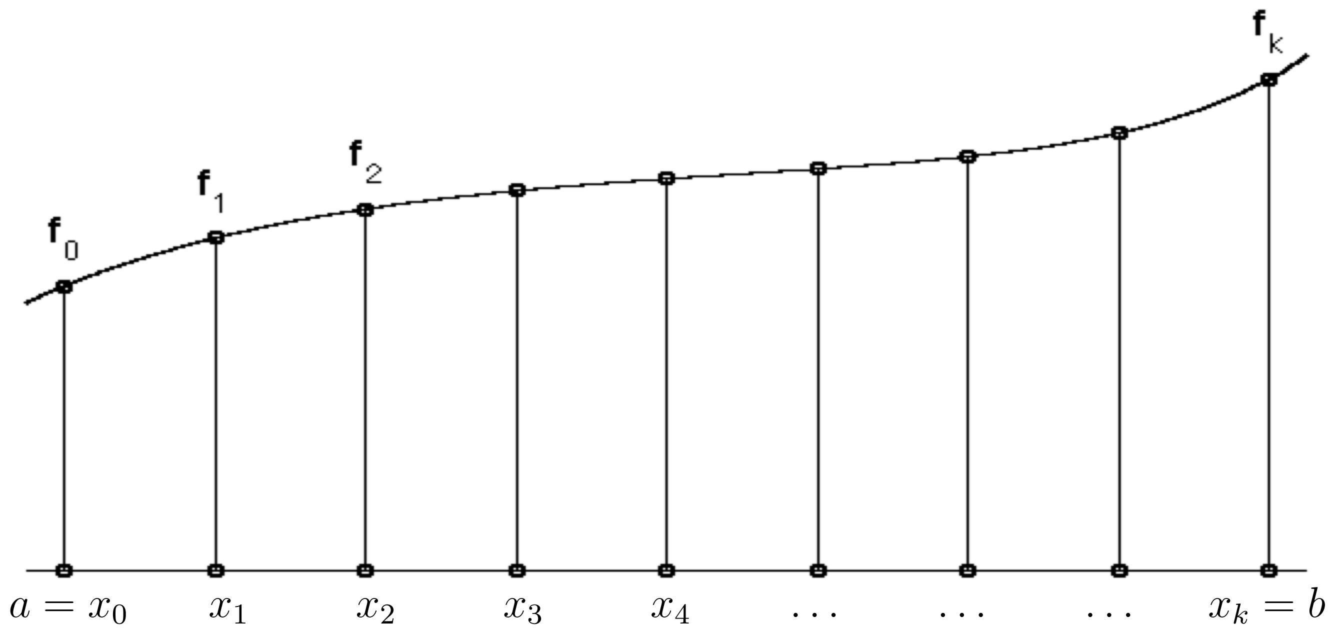
Take 4 $\times$(1)-(2) (*eliminate* $O(h^2)$ *term*):

$$\begin{aligned}
4I - I &= 4Q_{CT(h)} - Q_{CT(2h)} + c_4' h^4 + c_6' h^6 + \ldots \\
I &= \frac{4}{3} Q_{CT(h)} - \frac{1}{3} Q_{CT(2h)} + \hat{c}_4 h^4 + \hat{c}_6 h^6 + \ldots \\
&= Q_{S(2h)} + \boxed{\hat{c}_4 h^4} + \hat{c}_6 h^6 + h.o.t.
\end{aligned}$$

Here, $\quad Q_{S(2h)} \equiv \frac{4}{3} Q_{CT(h)} - \frac{1}{3} Q_{CT(2h)}$

New error – O(h⁴)

Richardson Extrapolation + Composite Trapezoidal Rule



$h:$	w_j:	$\frac{h}{2}$	h	h	h	h	h	h	$\cdots$	$\frac{h}{2}$	
$2h:$	$\tilde{w}_j$:	$\frac{2h}{2}$	0	$2h$	0	$2h$	0	$2h$	$\cdots$	$\frac{2h}{2}$	(k even)
$\frac{4}{3}w_j - \frac{1}{3}\tilde{w}_j$:		$\frac{h}{3}$	$\frac{4h}{3}$	$\frac{2h}{3}$	$\frac{4h}{3}$	$\frac{2h}{3}$	$\frac{4h}{3}$	$\frac{2h}{3}$	$\cdots$	$\frac{2h}{3}$	

- ***Richardson + Composite Trapezoidal = Composite Simpson***
- But we never compute it this way.
- Just use $Q_{CS} = (4\ Q_{CT(h)} - Q_{CT(2h)})\ /\ 3$
- No new function evaluations required!

Trapezoidal + Repeated Richardson Extrapolation
(Romberg Integration)

- We can repeat the extrapolation process to get rid of the $O(h^4)$ term.
- And repeat again, to get rid of $O(h^6)$ term.

$$
\begin{aligned}
T_{k,0} &= \text{Trapezoidal rule with } h = (b-a)/2^k \\
T_{k,j} &= \frac{4^j\, T_{k,j-1} - T_{k-1,j-1}}{4^j - 1}
\end{aligned}
$$

$$
\begin{array}{lllll}
h & T_{0,0} & & & \\
h/2 & T_{1,0} & T_{1,0} & & \\
h/4 & T_{2,0} & T_{2,0} & T_{2,0} & \\
h/8 & T_{3,0} & T_{3,1} & T_{3,2} & T_{3,3} \\
 & O(h^2) & O(h^4) & O(h^6) & O(h^8)
\end{array}
$$

- Idea works just as well if errors are of form $c_1 h + c_2 h^2 + c_3 h^3 + \ldots$, but tabular form would involve 2^j instead of 4^j

Repeated Richardson Extrapolation

(Romberg Integration)

- We can repeat the extrapolation process to get rid of the $O(h^4)$ term.
- And repeat again, to get rid of $O(h^6)$ term.

```
exact = exp(1)-1;
n=16;
x=0:n; x=x'/n; h=x(2)-x(1); f=exp(cos(5*x)); f=exp(-x.*x); f=exp(x);

T=zeros(5,5);

T(1,1)=16*h*(sum(f(1:16:end))-(f(1)+f(n+1))/2); % n must be divisible by 16
T(2,1)= 8*h*(sum(f(1: 8:end))-(f(1)+f(n+1))/2);
T(3,1)= 4*h*(sum(f(1: 4:end))-(f(1)+f(n+1))/2);
T(4,1)= 2*h*(sum(f(1: 2:end))-(f(1)+f(n+1))/2);
T(5,1)= 1*h*(sum(f(1: 1:end))-(f(1)+f(n+1))/2); % Finest approximation

format compact; format longe

T(:,1:1)

for j=2:5; for i=j:5; j1=j-1;
  T(i,j)=((4^j1)*T(i,j-1)-T(i-1,j-1))/(4^j1 - 1);
end;end;
```

Richardson Example

$$I = \int_0^1 e^x \, dx$$

Initial values, all created from same 17 values of f(x).

```
1.859140914229523
1.753931092464825
1.727221904557517
1.720518592164302
1.718841128579994
```

Using these 5 values, we build the table (extrapolate) to get more precision.

None	Round 1	Round 2	Round 3	Round 4
1.859140914229				
1.753931092464	1.718861151876			
1.727221904557	1.718318841921	1.718282687924		
1.720518592164	1.718284154699	1.718281842218	1.718281828794	
1.718841128579	1.718281974051	1.718281828675	1.718281828460	1.718281828459

Richardson Example

Error for Richardson Extrapolation (aka Romberg integration)

1/h	None	Round 1	Round 2	Round 3	Round 4
1	1.4086e-01				
2	3.5649e-02	5.7932e-04			
4	8.9401e-03	3.7013e-05	8.5947e-07		
8	2.2368e-03	2.3262e-06	1.3759e-08	3.3549e-10	
16	5.5930e-04	1.4559e-07	2.1631e-10	1.3429e-12	3.2419e-14
	O(h^2)	O(h^4)	O(h^6)	O(h^8)	O(h^10)

Gauss Quadrature Results

n	Qn	E
2	1.8591e+00	1.4086e-01
3	1.7189e+00	5.7932e-04
4	1.7183e+00	1.0995e-06
5	1.7183e+00	1.1666e-09
6	1.7183e+00	7.8426e-13
7	1.7183e+00	0

Next Up

- Gaussian Quadrature

- Composite Trapezoidal Rule

- Richardson Extrapolation

Gaussian Quadrature

- *Gaussian quadrature rules* are based on polynomial interpolation, but nodes as well as weights are chosen to maximize degree

- With $2n$ parameters, we can attain a degree of $2n-1$

- Gaussian quadrature rules can be found by method of undetermined coefficients, but resulting system of moment equations is nonlinear

- It is relatively easy to show that the standard (open) Gauss nodes on [-1,1] are the roots of $P_n(x)$, the nth-order Legendre polynomial

- The weights are the integrals of the corresponding Lagrange cardinal functions based on these nodes

- The closed Gauss nodes, which include $x = \pm 1$ are the roots of $(1-x^2)P'_{n-1}(x)$

Gaussian Quadrature

- The nodes and weights are extensively tabulated but are also available in routines for most every programming language

- The open points are often referred to as *Gauss* or *Gauss-Legendre* quadrature points

- The closed points are referred to as *Gauss-Lobatto* or *Gauss-Lobatto-Legendre* points

- There are also Gauss-Chebyshev points, etc.

- Finally, there are Gauss-Radau points for the case where -1 is included as a node but $+1$ is not (i.e., closed on left but open on the right)

Example: Gaussian Quadrature, $n = 2$

- Derive two-point Gauss quadrature rule on [-1,1],

$$G_2(f) \;=\; w_1 f(x_1) \,+\, w_2 f(x_2)$$

  with (w_i, x_i) chosen to maximize degree of resulting rule

- Use method of undetermined coefficients

- Four parameters to be determined, so expect to be able to integrate cubics exactly because cubics are determined by 4 parameters

Gauss Quadrature Example, continued

- Requiring rule to integrate first four monomials exactly yields four moment equations

$$w_1 + w_2 = \int_{-1}^{1} 1\,dx = x\Big|_{-1}^{1} = 2$$

$$w_1 x_1 + w_2 x_2 = \int_{-1}^{1} x\,dx = x^2\Big|_{-1}^{1} = 0$$

$$w_1 x_1^2 + w_2 x_2^2 = \int_{-1}^{1} x^2\,dx = \frac{1}{3}x^3\Big|_{-1}^{1} = 2/3$$

$$w_1 x_1^3 + w_2 x_3^2 = \int_{-1}^{1} x^3\,dx = \frac{1}{4}x^4\Big|_{-1}^{1} = 0$$

Gauss Quadrature Example, continued

- In this case, can exploit symmetry, $x_1 = -x_2$, $w_1 = w_2 = 1$, to obtain quadratic equation for x_2

- Solution is $x_1 = -1/\sqrt{3}$, $x_2 = 1/\sqrt{3}$, $w_1 = 1$, $w_2 = 1$

- Resulting two-point Gauss quadrature rule has form

$$G_2(f) = f(-1/\sqrt{3}) + f(1/\sqrt{3})$$

- Remarkably, evaluating f at just two points allows us to *exactly* integrate all polynomials up to and including degree 3

Gauss Quadrature Example, continued

- Degree of 2-point Gauss quadrature rule is $d = 3$

- In general, n-point Gauss quadrature rule has degree $d = 2n - 1$

- For n-point Gauss-Lobatto rule, which has endpoints ± 1 prescribed, degree $d = 2n - 3$ as there are only $2n - 2$ free parameters in this case

Gauss Quadrature, I

Consider
$$I \;:=\; \int_{-1}^{1} f(x)\,dx.$$

Find w_i, x_i $i = 1, \ldots, n$, to maximize degree of accuracy, M.

- Cardinality, $|\,.\,|$:
$$\begin{aligned} |\,\mathbb{P}_M\,| \;&=\; M \,+\, 1 \\ |\,w_i\,| \;+\; |\,x_i\,| \;&=\; 2n \\ M+1 \;&=\; 2n \qquad \Longleftrightarrow \qquad M \;=\; 2n-1 \end{aligned}$$

- Indeed, it is possible to find x_i and w_i such that *all polynomials of degree* $\leq M = 2n-1$ are integrated *exactly.*

- The n nodes, x_i, are the zeros of the nth-order Legendre polynomial.

- The weights, w_i, are the integrals of the cardinal Lagrange polynomials associated with these nodes:
$$w_i \;=\; \int_{-1}^{1} l_i(x)\,dx, \qquad l_i(x) \in \mathbb{P}_{n-1}, \qquad l_i(x_j) = \delta_{ij}.$$

- Error scales like $\quad |I - Q_n| \sim C\dfrac{f^{(2n)}(\xi)}{(2n)!} \quad$ (Q_n exact for $f(x) \in \mathbb{P}_{2n-1}$.)

- n nodes are roots of orthogonal polynomials

Change of Interval

- Gauss rules are prescribed on interval $[-1, 1]$, so usually need to transform to $[a, b]$ for general application

- Suppose $[\xi_i, w_i]$ are the Gauss points and weights associated with interval $[-1, 1]$

- Then, use the *affine* (i.e., linear) transformation,

$$t_i = a + (b - a)(\xi_i + 1)/2,$$

  which satisfies $\xi = -1$ when $t = a$ and $\xi = 1$ when $t = b$

- Here ξ_i, $i = 1, \ldots, n$ are the Gauss points on [-1,1]

- You simply *look up** the (ξ_i, w_i) pairs, use the above formula to get t_i, then evaluate

$$Q_n = \frac{b-a}{2} \sum_{i=1}^{n} w_i f(t_i)$$

(*that is, call a function)

Use of Gauss Quadrature

Table 25.4 ABSCISSAS AND WEIGHT FACTORS FOR GAUSSIAN INTEGRATION

$$\int_{-1}^{+1} f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i)$$

Abscissas $= \pm x_i$ (Zeros of Legendre Polynomials) Weight Factors $= w_i$

$\pm x_i$	w_i
$n=2$	
0.57735 02691 89626	1.00000 00000 00000
$n=3$	
0.00000 00000 00000	0.88888 88888 88889
0.77459 66692 41483	0.55555 55555 55556
$n=4$	
0.33998 10435 84856	0.65214 51548 62546
0.86113 63115 94053	0.34785 48451 37454
$n=5$	
0.00000 00000 00000	0.56888 88888 88889
0.53846 93101 05683	0.47862 86704 99366
0.90617 98459 38664	0.23692 68850 56189
$n=6$	
0.23861 91860 83197	0.46791 39345 72691
0.66120 93864 66265	0.36076 15730 48139
0.93246 95142 03152	0.17132 44923 79170
$n=7$	
0.00000 00000 00000	0.41795 91836 73469
0.40584 51513 77397	0.38183 00505 05119
0.74153 11855 99394	0.27970 53914 89277
0.94910 79123 42759	0.12948 49661 68870

$\pm x_i$	w_i
$n=8$	
0.18343 46424 95650	0.36268 37833 78362
0.52553 24099 16329	0.31370 66458 77887
0.79666 64774 13627	0.22238 10344 53374
0.96028 98564 97536	0.10122 85362 90376
$n=9$	
0.00000 00000 00000	0.33023 93550 01260
0.32425 34234 03809	0.31234 70770 40003
0.61337 14327 00590	0.26061 06964 02935
0.83603 11073 26636	0.18064 81606 94857
0.96816 02395 07626	0.08127 43883 61574
$n=10$	
0.14887 43389 81631	0.29552 42247 14753
0.43339 53941 29247	0.26926 67193 09996
0.67940 95682 99024	0.21908 63625 15982
0.86506 33666 88985	0.14945 13491 50581
0.97390 65285 17172	0.06667 13443 08688
$n=12$	
0.12523 34085 11469	0.24914 70458 13403
0.36783 14989 98180	0.23349 25365 38355
0.58731 79542 86617	0.20316 74267 23066
0.76990 26741 94305	0.16007 83285 43346
0.90411 72563 70475	0.10693 93259 95318
0.98156 06342 46719	0.04717 53363 86512

- There is a lot of software, in most every language, for computing the nodes and weights for all of the Gauss, Gauss-Lobatto, Gauss-Radau rules (Chebyshev, Legendre, etc.)

Let's work out an example for 3-Point Gaussian quadrature applied to

$$I \;:=\; \int_{-1}^{1} e^x \, dx.$$

Table Look-Up: quadrature points, $\xi_i \in (-1, 1)$ and weights, w_i:

$$\xi_1 = -0.774596669241483 \qquad w_1 = 0.555555555555555$$

$$\xi_2 = 0.000000000000000 \qquad w_2 = 0.888888888888890$$

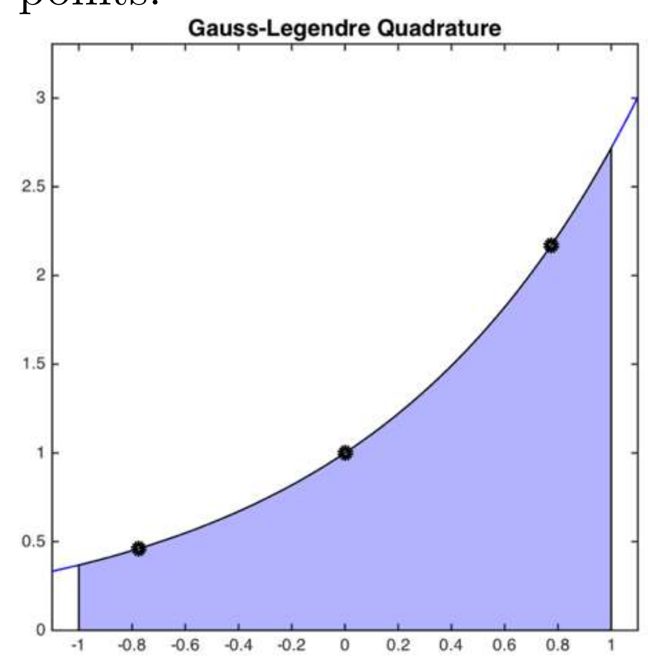
$$\xi_3 = 0.774596669241483 \qquad w_3 = 0.555555555555555$$

Function Eval: evaluate integrand $f(x) = e^x$ at quadrature points:

$$f_1 = e^{-0.774596669241483} = 0.460889634482101\,5$$

$$f_2 = e^{\;0.000000000000000} = 1.000000000000000$$

$$f_3 = e^{+0.774596669241483} = 2.169716837141918\,5$$



Quadrature Rule: sum product of weights $\times$ function, $w_i f_i$:

$$\begin{aligned} Q_{GL} &= 0.555555555555555 * 0.4608896344821015 \\ &+ 0.888888888888888 * 1.0000000000000000 \\ &+ 0.555555555555555 * 2.1697168371419185 \\ &= 2.350336928680011 \end{aligned}$$

Comparison: Compare to exact answer:

$$I = \int_{-1}^{1} e^x \, dx = e^1 - e^{-1} = 2.350402387287603$$

$$|I - Q_{GL}| = 6.545860759255007e - 05$$

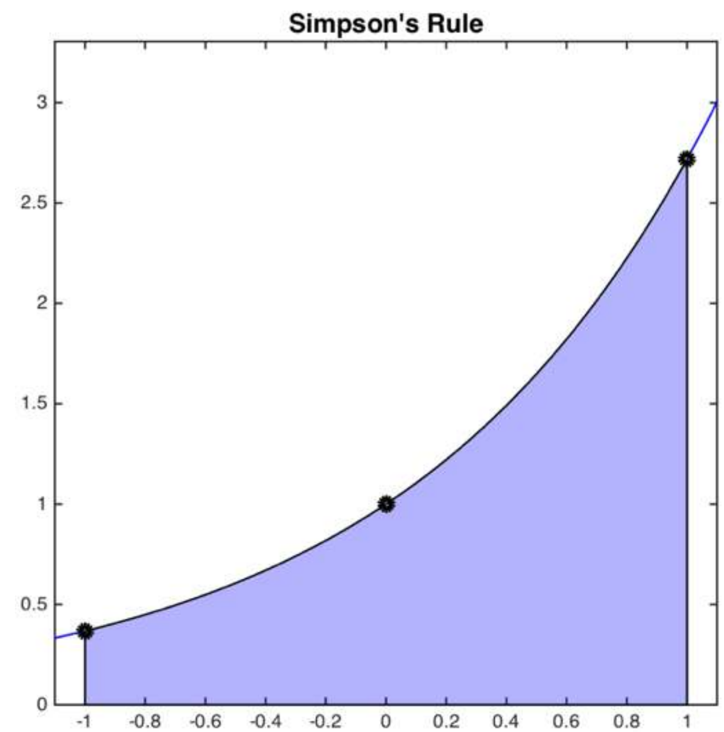
Let's compare to Simpson's Rule:

Quadrature Points and Weights:

$$\begin{aligned} \xi_1 &= -1.0 & w_1 &= 1/3 \qquad \text{(Recall, } b-a=2.\text{)} \\ \xi_2 &= 0.0 & w_2 &= 4/3 \\ \xi_3 &= 1.0 & w_3 &= 1/3 \end{aligned}$$

Function Eval: evaluate integrand $f(x) = e^x$ at quadrature points:

$$\begin{aligned} f_1 &= e^{-1} = 0.3678794411714423 \\ f_2 &= e^{\,0} = 1.0000000000000000 \\ f_3 &= e^{+1} = 2.7182818284590452 \end{aligned}$$



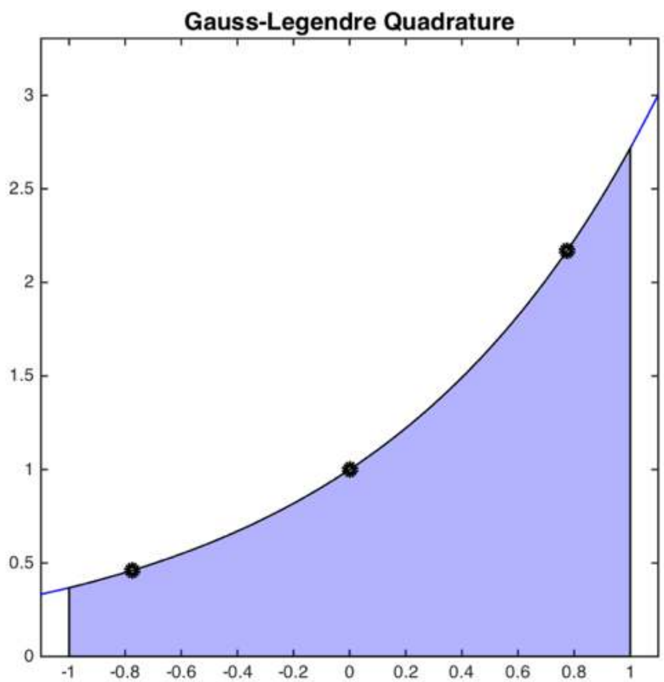
Simpsons Rule: sum product of weights $\times$ function, $w_i f_i$:

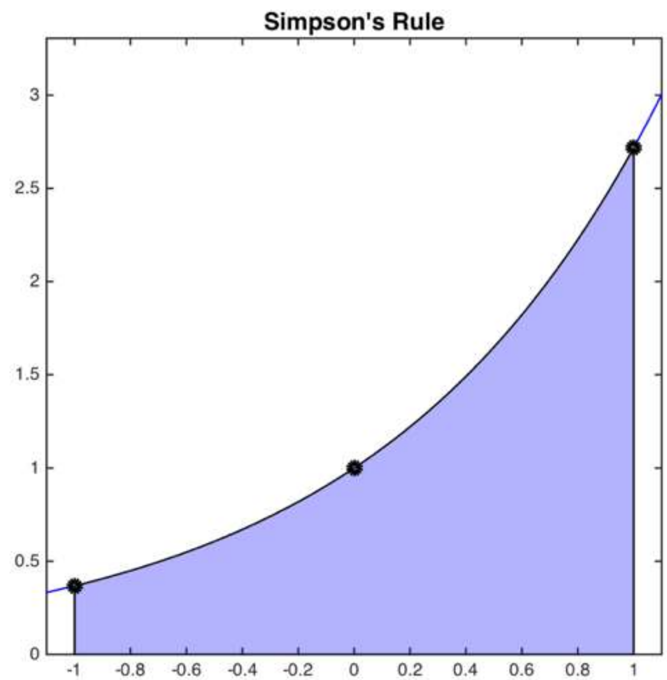
$$\begin{aligned} Q_{simp} &= \frac{1}{3}0.3678794411714423 + \frac{4}{3}1.0000000000000000 + \frac{1}{3}2.7182818284590452 \\ &= 2.362053756543496 \end{aligned}$$

Comparison: Compare to exact answer:

$$|I - Q_{simp}| = 1.165136925589261e - 02$$

Error for Simpson's rule is $\approx$ 180 times larger.





Gaussian Quadrature

- Gaussian quadrature rules have maximal degree and optimal accuracy for the number of nodes used

- Weights are *always positive* and approximate integral always converges to exact integral as $n \longrightarrow \infty$

- Unfortunately, aside from Chebyshev, Gaussian rules of different orders do not have points in common so Gaussian rules are not *progressive*

- If you want to improve the estimate by increasing n to n', you have to re-evaluate f at all n' nodes

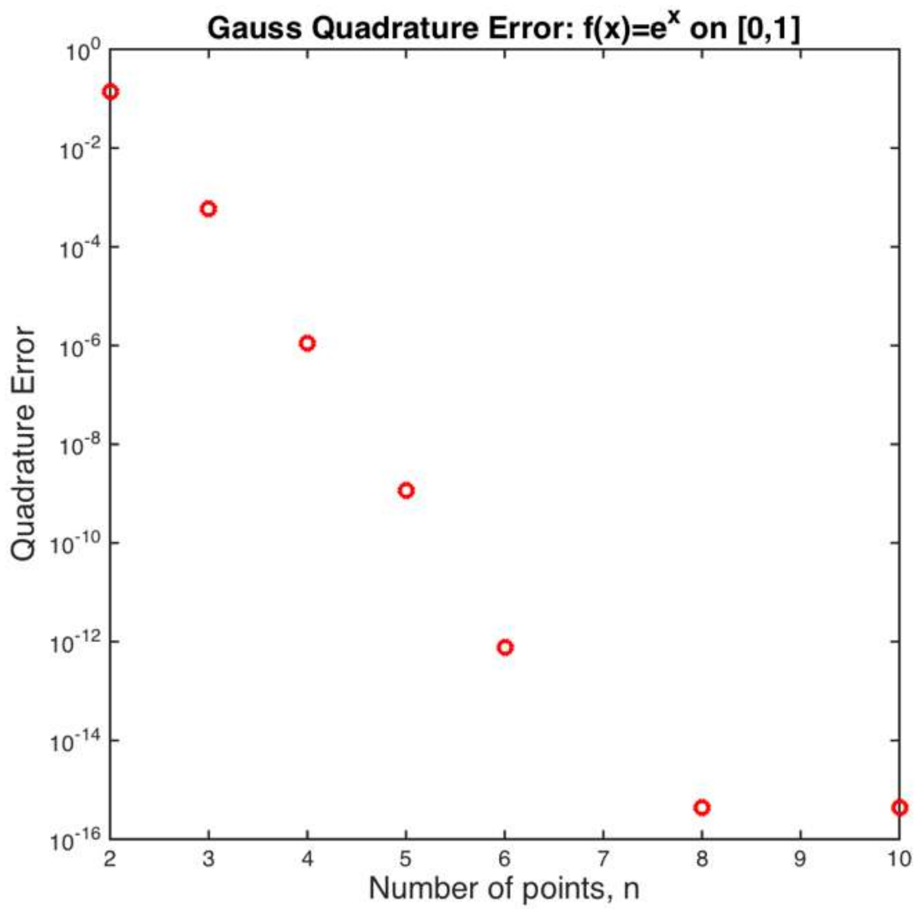
- Thus, estimating error using Gauss rules of different order requires a full re-evaluation

- Gauss-Konrod rules augment the Gauss points with $n' - n$ points, but are suboptimal

Gauss-Lobatto-Legendre Quadrature Example

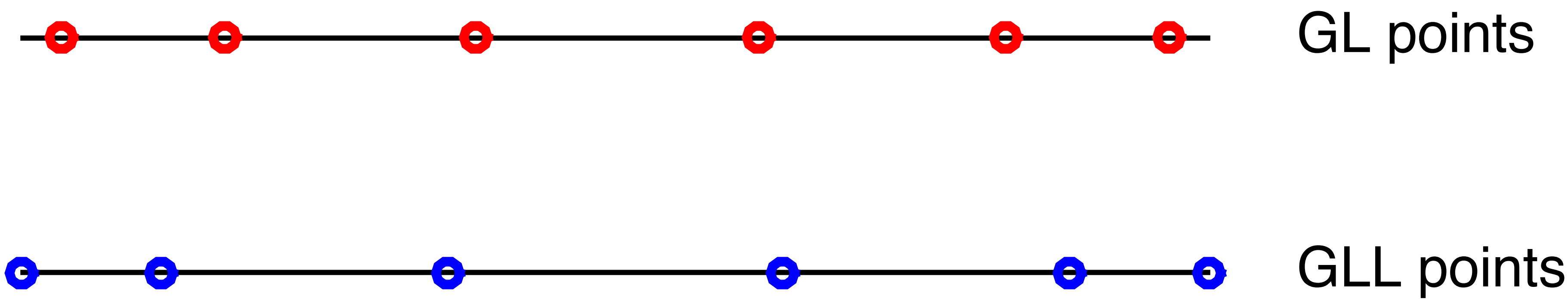
```
format longe; format compact; lw='linewidth',fs='fontsize';

a=0; b=1;

exact = exp(1)-1;  %% Integral from 0 to 1 of e^x

for n=2:10;
  [z,w]=zwgll(n-1);  % Gauss-Lobatto-Legendre pts/wts

  t=a + 0.5*(z+1)*(b-a);
  f=exp(t);

  Q  = w'*f*(b-a)/2.;
  err= abs(Q-exact);
  disp([n Q err ])

  semilogy(n,err,'ro',lw,2); hold on

end;
title('Gauss Quadrature Error: f(x)=e^x on [0,1]',fs,14);
xlabel('Number of points, n',fs,14);
ylabel('Quadrature Error',fs,14);
axis square
```

gauss_quad_demo.m



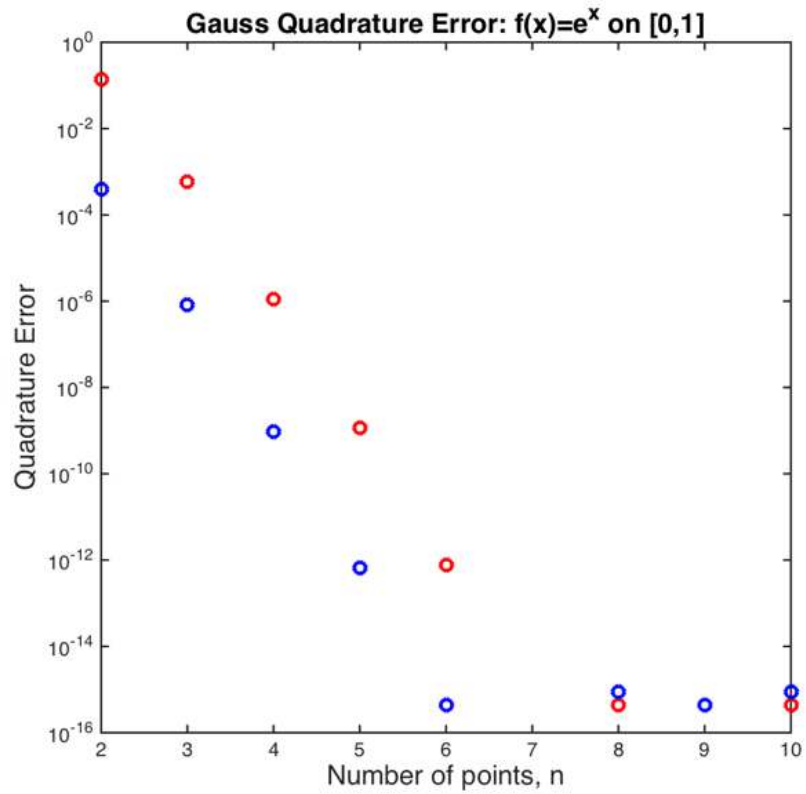
Closed Gauss Rules (Gauss-Lobatto-Legendre)

- Gauss-Legendre quadrature
  - Endpoints not included
  - Open formula
- Gauss-Lobatto-Legendre quadrature
  - +1 and -1 (i.e., a,b) included in function evaluation (like Simpson)
  - Closed formula
- GL is more efficient than GLL.



Gauss-Legendre Quadrature Example

```
a=0; b=1;

exact = exp(1)-1;  %% Integral from 0 to 1 of e^x

for kpass=1:2;
for n=2:10;
  [z,w]=zwgll(n-1);  % Gauss-Lobatto-Legendre pts/wts
  if kpass==2; [z,w]=zwgl(n); end;  % Gauss-Legendre pts/wts

  t=a + 0.5*(z+1)*(b-a);
  f=exp(t);

  Q  = w'*f*(b-a)/2.;
  err= abs(Q-exact);
  disp([n Q err ])

  if kpass==1; semilogy(n,err,'ro',lw,2); hold on; end;
  if kpass==2; semilogy(n,err,'bo',lw,2); hold on; end;

end;
end;
title('Gauss Quadrature Error: f(x)=e^x on [0,1]',fs,14);
xlabel('Number of points, n',fs,14);
ylabel('Quadrature Error',fs,14);
axis square

print -dpng t.png;
!open t.png
```



gauss_quad_demo2.m