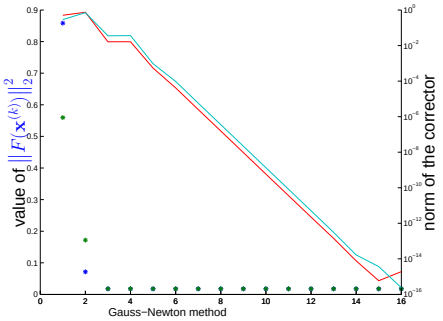
Gauss-Newton method:

initial value $(1.8, 1.8, 0.1)^T$ (red curve),
initial value $(1.5, 1.5, 0.1)^T$ (cyan curve),

convergence in both cases.

Notice: linear convergence.



◇

6.5.3 Trust region method (Levenberg-Marquardt method)

As in the case of Newton's method for non-linear systems of equations, see Sect. 3.4.4: often overshooting of Gauss-Newton corrections occurs.

Remedy as in the case of Newton's method: damping.

Idea: damping of the Gauss-Newton correction in (6.5.7) using a penalty term

instead of $\left\| F(\mathbf{x}^{(k)}) + DF(\mathbf{x}^{(k)})\mathbf{s} \right\|^2$ minimize $\left\| F(\mathbf{x}^{(k)}) + DF(\mathbf{x}^{(k)})\mathbf{s} \right\|^2 + \lambda \|\mathbf{s}\|_2^2$.

$\lambda > 0 \hat{=}$ penalty parameter (how to choose it ? → heuristic)

$$\lambda = \gamma \left\| F(\mathbf{x}^{(k)}) \right\|_2 \quad , \quad \gamma := \begin{cases} 10 & \text{, if } \left\| F(\mathbf{x}^{(k)}) \right\|_2 \geq 10 \, , \\ 1 & \text{, if } 1 < \left\| F(\mathbf{x}^{(k)}) \right\|_2 < 10 \, , \\ 0.01 & \text{, if } \left\| F(\mathbf{x}^{(k)}) \right\|_2 \leq 1 \, . \end{cases}$$

☛ Modified (regularized) equation for the corrector $\mathbf{s}$:

$$\left(DF(\mathbf{x}^{(k)})^T DF(\mathbf{x}^{(k)}) + \lambda \mathbf{I} \right) \mathbf{s} = -DF(\mathbf{x}^{(k)}) F(\mathbf{x}^{(k)}) \, . \tag{6.5.8}$$

6.6 Essential Skills Learned in Chapter 6

You should know:

- several possibilities to solve linear least squares problems
- how to solve non-linear least squares problems

7 Numerical Quadrature

Numerical quadrature

= Approximate evaluation of $\int_\Omega f(\mathbf{x}) \, \mathrm{d}\mathbf{x}$, integration domain $\Omega \subset \mathbb{R}^d$

Continuous function $f : \Omega \subset \mathbb{R}^d \mapsto \mathbb{R}$ only available as `function y = f(x)` (point evaluation)

Special case $d = 1$: $\Omega = [a, b]$ (interval)

☞ Numerical quadrature methods are key building blocks for methods for the numerical treatment of differential equations.

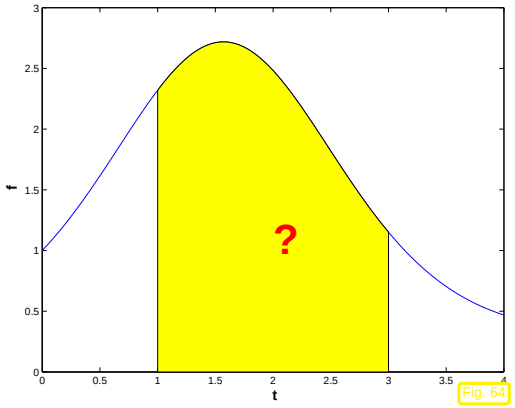


Fig. 64

Numerical quadrature methods

approximate

$$\int_a^b f(t)\,\mathrm{d}t$$

7.1 Quadrature Formulas

n-point quadrature formula on $[a,b]$: (n-point quadrature rule)

$$\int_a^b f(t)\,\mathrm{d}t \;\approx\; Q_n(f) := \sum_{j=1}^{n} \omega_j^n\, f(\xi_j^n)\;. \qquad (7.1.1)$$

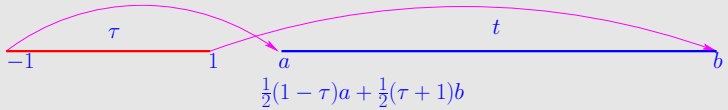
ω_j^n : quadrature weights $\in \mathbb{R}$ (*ger.:* Quadraturgewichte)
ξ_j^n : quadrature nodes $\in [a,b]$ (*ger.:* Quadraturknoten)

Remark 7.1.1 (Transformation of quadrature rules).

Given: quadrature formula $\left(\widehat{\xi}_j, \widehat{\omega}_j\right)_{j=1}^n$ on reference interval $[-1,1]$

Idea: transformation formula for integrals

$$\int_a^b f(t)\,\mathrm{d}t = \tfrac{1}{2}(b-a)\int_{-1}^{1} \widehat{f}(\tau)\,\mathrm{d}\tau\;,\quad \widehat{f}(\tau) := f(\tfrac{1}{2}(1-\tau)a + \tfrac{1}{2}(\tau+1)b)\;. \qquad (7.1.2)$$



quadrature formula for general interval $[a,b]$, $a,b \in \mathbb{R}$:

$$\int_a^b f(t)\,\mathrm{d}t \approx \tfrac{1}{2}(b-a)\sum_{j=1}^{n} \widehat{\omega}_j \widehat{f}(\widehat{\xi}_j) = \sum_{j=1}^{n} \omega_j f(\xi_j) \quad \text{with} \quad \begin{array}{l} \xi_j = \tfrac{1}{2}(1-\widehat{\xi}_j)a + \tfrac{1}{2}(1+\widehat{\xi}_j)b\;, \\ \omega_j = \tfrac{1}{2}(b-a)\widehat{\omega}_j\;. \end{array}$$

A 1D quadrature formula on arbitrary intervals can be specified by providing its weights $\widehat{\omega}_j$ /nodes $\widehat{\xi}_j$ for integration domain $[-1,1]$. Then the above transformation is assumed.

Other common choice of reference interval: $[0,1]$

△

7.1 p. 373

Inevitable for generic integrand:

quadrature error $\quad E(n) := \left| \int_a^b f(t)\,\mathrm{d}t - Q_n(f) \right|$

Given families of quadrature rules $\{Q_n\}_n$ with quadrature weights $\left\{\omega_j^n,\ j=1,\ldots,n\right\}_{n\in\mathbb{N}}$ and quadrature nodes $\left\{\xi_j^n,\ j=1,\ldots,n\right\}_{n\in\mathbb{N}}$ we

should be aware of the *asymptotic* behavior of quadrature error $E(n)$ for $n \to \infty$

Qualitative distinction:
- ▷ algebraic convergence $E(n) = O(n^{-p})$, $p > 0$
- ▷ exponential convergence $E(n) = O(q^n)$, $0 \le q < 1$

Note that the number n of nodes agrees with the number of f-evaluations required for evaluation of the quadrature formula. This is usually used as a *measure for the cost* of computing $Q_n(f)$.

Therefore we consider the quadrature error as a function of n.

7.1 p. 374

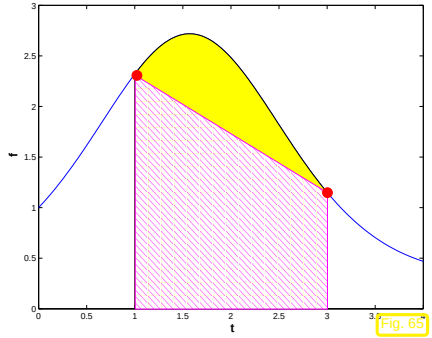
Idea: Equidistant quadrature nodes $t_j := a + hj$, $h := \frac{b-a}{n}$, $j = 0,\ldots,n$: choose the n weights such that the error $E(n) = 0$ for all polynomials f of degree $n-1$.

Example 7.1.2 (Newton-Cotes formulas).

- $n = 1$: Trapezoidal rule

$$\widehat{Q}_{\mathrm{trp}}(f) := \frac{1}{2}(f(0) + f(1)) \tag{7.1.3}$$

$$\left(\int_a^b f(t)\,\mathrm{d}t \approx \frac{b-a}{2}(f(a) + f(b)) \right)$$



- $n = 2$: Simpson rule

$$\frac{h}{6}\left(f(0) + 4f(\tfrac{1}{2}) + f(1)\right) \quad \left(\int_a^b f(t)\,\mathrm{d}t \approx \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right) \right) \tag{7.1.4}$$

◇

Remark 7.1.3 (Error estimates for polynomial quadrature).

Quadrature error estimates directly from L^∞-interpolation error estimates for Lagrangian interpolation with polynomial of degree $n-1$:

$$f \in C^n([a,b]) \;\Rightarrow\; \left| \int_a^b f(t)\,\mathrm{d}t - Q_n(f) \right| \le \frac{1}{n!}(b-a)^{n+1} \left\| f^{(n)} \right\|_{L^\infty([a,b])} . \tag{7.1.5}$$

△

Idea: Gaussian quadrature: Choose the n weights and the n points such that the error $E(n) = 0$ for all polynomials f of degree $2n-1$.

Example 7.1.4 (2-point quadrature rule of order 4).

Necessary & sufficient conditions for order 4 (first wrong integral is $\int_a^b x^4 dx$):

$$Q_n(p) = \int_a^b p(t)\,\mathrm{d}t \;\forall p \in \mathcal{P}_3 \;\Leftrightarrow\; Q_n(t^q) = \frac{1}{q+1}(b^{q+1} - a^{q+1}) , \quad q = 0,1,2,3 .$$

4 equations for weights ω_j and nodes ξ_j, $j = 1,2$ ($a = -1, b = 1$), *cf.* Rem. **??**

$$\begin{aligned} \int_{-1}^1 1\,\mathrm{d}t = 2 = 1\omega_1 + 1\omega_2 \quad , \quad \int_{-1}^1 t\,\mathrm{d}t = 0 = \xi_1\omega_1 + \xi_2\omega_2 \\ \int_{-1}^1 t^2\,\mathrm{d}t = \frac{2}{3} = \xi_1^2\omega_1 + \xi_2^2\omega_2 \quad , \quad \int_{-1}^1 t^3\,\mathrm{d}t = 0 = \xi_1^3\omega_1 + \xi_2^3\omega_2 . \end{aligned} \tag{7.1.6}$$

Solve using MAPLE:

```
> eqns := seq(int(x^k, x=-1..1) = w[1]*xi[1]^k+w[2]*xi[2]^k,k=0..3);
> sols := solve(eqns, indets(eqns, name)):
> convert(sols, radical);
```

➢ weights & nodes: $\left\{\omega_2 = 1, \omega_1 = 1, \xi_1 = 1/3\sqrt{3}, \xi_2 = -1/3\sqrt{3}\right\}$

► quadrature formula: $\int_{-1}^1 f(x)\,dx \approx f\left(\frac{1}{\sqrt{3}}\right) + f\left(-\frac{1}{\sqrt{3}}\right)$ (7.1.7)

◇

Remark 7.1.5 (Computing Gauss nodes and weights).

Compute nodes/weights of Gaussian quadrature by solving an eigenvalue problem! (Golub-Welsch algorithm [18, Sect. 3.5.4])

In codes: ξ_j, ω_j from tables!

Code 7.1.6: Golub-Welsch algorithm

```
1 function [x,w]=gaussquad(n)
2 b = zeros(n-1,1);
3 for i=1:(n-1), b(i)=i/sqrt(4*i*i-1); end
4 J=diag(b,-1)+diag(b,1); [ev,ew]=eig(J);
5 for i=1:n, ev(:,i)./norm(ev(:,i)); end
6 x=diag(ew); w=(2*(ev(1,:).*ev(1,:)))';
```

△

Idea: Clenshaw-Curtis quadrature:

$$\int_{-1}^{1} f(x)\,dx = \int_{0}^{\pi} f(\cos\theta)\,\sin\theta d\theta = \sum_{\text{even } k} \frac{2a_k}{1-k^2} \tag{7.1.8}$$

with a_k the Fourier coefficients of $F(\theta) = f(\cos\theta) = \sum_{k=0}^{\infty} a_k \cos(k\theta)$.
Advantage for the Clenshaw-Curtis is the speed and stability of the fast Fourier transform.

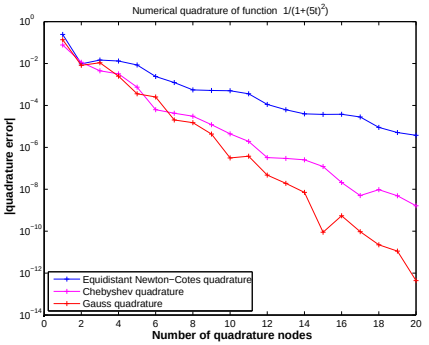
Code 7.1.7: tracking errors on quadrature rules

```
function I = cc(f,n) %clenshaw curtis
x = cos(pi*(0:n)'/n);
fx = feval(f,x)/(2*n);
g = real(fft(fx([1:n+1 n:-1:2])));
a = [g(1); g(2:n) + g(2*n:-1:n+2) ;g(n+1)];
w = 0*a'; w(1:2:end) = 2./(1-(0:2:n).^2);
I = w*a;
```

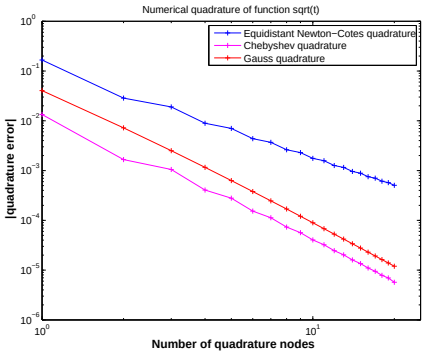
Example 7.1.8 (Error of (non-composite) quadratures).

Code 7.1.9: important polynomial quadrature rules

```
function res = numquad(f,a,b,N,mode)
%Numerical quadrature on [a,b] by polynomial quadrature formula
%f -> function to be integrated (handle), must support vector arguments
%a,b -> integration interval [a,b] (endpoints included)
%N -> Maximal degree of polynomial
%mode: equidistant, Chebychev, Gauss

if (nargin < 5), mode = 'equidistant'; end

res = [];

if strcmp(mode,'Gauss')
  for deg=1:N
    [gx,w] = gaussQuad(deg);
    x = 0.5*(b-a)*gx+0.5*(a+b);
    y = feval(f,x);
    res = [res; deg, 0.5*(b-a)*dot(w,y)];
  end
else
  p = (N+1:-1:1);
  w = (b.^p - a.^p)./p;
  for deg=1:N
    if strcmp(mode,'Chebychev')
      x = 0.5*(b-a)*cos((2*(0:deg)+1)/(2*deg+2)*pi)+0.5*(a+b);
    else
      x = (a:(b-a)/deg:b);
    end
    y = feval(f,x);
    poly = polyfit(x,y,deg);
    res = [res; deg, dot(w(N+1-deg:N+1),poly)];
  end
end
```

quadrature error, $f_1(t) := \frac{1}{1+(5t)^2}$ on $[0,1]$



quadrature error, $f_2(t) := \sqrt{t}$ on $[0,1]$

Asymptotic behavior of quadrature error $\epsilon_n := \left|\int_0^1 f(t)\,\mathrm{d}t - Q_n(f)\right|$ for "$n \to \infty$":

➢ exponential convergence $\epsilon_n \approx O(q^n)$, $0 < q < 1$, for C^∞-integrand f_1 $\rightsquigarrow$: Newton-Cotes quadrature : $q \approx 0.61$, Clenshaw-Curtis quadrature : $q \approx 0.40$, Gauss-Legendre quadrature : $q \approx 0.27$

➢ algebraic convergence $\epsilon_n \approx O(n^{-\alpha})$, $\alpha > 0$, for integrand f_2 with singularity at $t = 0$ $\rightsquigarrow$ Newton-Cotes quadrature : $\alpha \approx 1.8$, Clenshaw-Curtis quadrature : $\alpha \approx 2.5$, Gauss-Legendre quadrature : $\alpha \approx 2.7$

Code 7.1.10: tracking errors on quadrature rules

```
function numquaderrs()
%Numerical quadrature on [0,1]
N = 20;

figure('Name','1/(1+(5t)^2)');
exact = atan(5)/5;
eqdres = numquad(inline('1./(1+(5*x).^2)'),0,1,N,'equidistant');
chbres = numquad(inline('1./(1+(5*x).^2)'),0,1,N,'Chebychev');
gaures = numquad(inline('1./(1+(5*x).^2)'),0,1,N,'Gauss');
semilogy(eqdres(:,1),abs(eqdres(:,2)-exact),'b+-',...
```

```
      chbres(:,1),abs(chbres(:,2)-exact),'m+-',...
      gaures(:,1),abs(gaures(:,2)-exact),'r+-');
set(gca,'fontsize',12);
title('Numerical quadrature of function 1/(1+(5t)^2)');
xlabel('{\bf Number of quadrature nodes}','fontsize',14);
ylabel('{\bf |quadrature error|}','fontsize',14);
legend('Equidistant Newton-Cotes quadrature',...
       'Clenshaw-Curtis quadrature',...
       'Gauss quadrature',3);
eqdp1 = polyfit(eqdres(:,1),log(abs(eqdres(:,2)-exact)),1)
chbp1 = polyfit(chbres(:,1),log(abs(chbres(:,2)-exact)),1)
gaup1 = polyfit(gaures(:,1),log(abs(gaures(:,2)-exact)),1)
print -depsc2 '../PICTURES/numquaderr1.eps';

figure('Name','sqrt(t)');
exact = 2/3;
eqdres = numquad(inline('sqrt(x)'),0,1,N,'equidistant');
chbres = numquad(inline('sqrt(x)'),0,1,N,'Chebychev');
gaures = numquad(inline('sqrt(x)'),0,1,N,'Gauss');
loglog(eqdres(:,1),abs(eqdres(:,2)-exact),'b+-',...
      chbres(:,1),abs(chbres(:,2)-exact),'m+-',...
      gaures(:,1),abs(gaures(:,2)-exact),'r+-');
set(gca,'fontsize',12);
```

```
axis([1 25 0.000001 1]);
title('Numerical quadrature of function sqrt(t)');
xlabel('{\bf Number of quadrature nodes}','fontsize',14);
ylabel('{\bf |quadrature error|}','fontsize',14);
legend('Equidistant Newton-Cotes quadrature',...
       'Clenshaw-Curtis quadrature',...
       'Gauss quadrature',1);
eqdp2 = polyfit(log(eqdres(:,1)),log(abs(eqdres(:,2)-exact)),1)
chbp2 = polyfit(log(chbres(:,1)),log(abs(chbres(:,2)-exact)),1)
gaup2 = polyfit(log(gaures(:,1)),log(abs(gaures(:,2)-exact)),1)
print -depsc2 '../PICTURES/numquaderr2.eps';
```

◇

Equal spacing is a disaster for high-order interpolation and integration **!**

▶ Divide the integration domain in small pieces and use low-order rule on each piece (composite quadrature)

▶ Take into account the eventual non-smoothness of f when dividing the integration domain

7.2 Composite Quadrature

With $a = x_0 < x_1 < \cdots < x_{m-1} < x_m = b$

$$\int_a^b f(t)\,\mathrm{d}t = \sum_{j=1}^{m} \int_{x_{j-1}}^{x_j} f(t)\,\mathrm{d}t\,. \tag{7.2.1}$$

Recall (7.1.5): for polynomial quadrature rule (**??**) and $f \in C^n([a,b])$ quadrature error shrinks with $n+1$st power of length of integration interval.

▶ Reduction of quadrature error can be achieved by

- splitting of the integration interval according to (7.2.1),
- using the intended quadrature formula on each sub-interval $[x_{j-1}, x_j]$.

Note: Increasse in total no. of f-evaluations incurred, which has to be balanced with the gain in accuracy to achieve optimal efficiency,

Idea:

- Partition integration domain $[a,b]$ by mesh (grid, → Sect.**??**) $\mathcal{M} := \{a = x_0 < x_1 < \ldots < x_m = b\}$
- Apply quadrature formulas on sub-intervals $I_j := [x_{j-1}, x_j]$, $j = 1, \ldots, m$, and sum up.

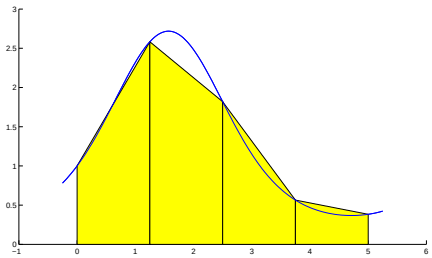
▶ composite quadrature rule

Note: Here we only consider one and the same quadrature formula (local quadrature formula) applied on all sub-intervals.

Example 7.2.1 (Simple composite polynomial quadrature rules).

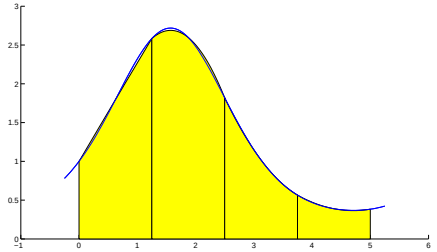
Composite trapezoidal rule, *cf.* (8.4.2)

$$\int_a^b f(t)\mathrm{d}t = \tfrac{1}{2}(x_1 - x_0)f(a) + \sum_{j=1}^{m-1} \tfrac{1}{2}(x_{j+1} - x_{j-1})f(x_j) + \tfrac{1}{2}(x_m - x_{m-1})f(b)\,. \tag{7.2.2}$$



Composite Simpson rule, *cf.* (7.1.4)

$$\int_a^b f(t)\mathrm{d}t = \frac{1}{6}(x_1 - x_0)f(a) + \sum_{j=1}^{m-1} \frac{1}{6}(x_{j+1} - x_{j-1})f(x_j) + \sum_{j=1}^{m} \frac{2}{3}(x_j - x_{j-1})f(\tfrac{1}{2}(x_j + x_{j-1})) + \frac{1}{6}(x_m - x_{m-1})f(b)\ . \tag{7.2.3}$$



Formulas (7.2.2), (7.2.3) directly suggest efficient implementation with minimal number of f-evaluations.

◇

Focus: *asymptotic* behavior of quadrature error for

mesh width $h := \max_{j=1,\ldots,m} |x_j - x_{j-1}| \to 0$

For *fixed* local n-point quadrature rule: $O(mn)$ f-evaluations for composite quadrature ("total cost")

➢ If mesh equidistant ($|x_j - x_{j-1}| = h$ for all j), then total cost for composite numerical quadrature $= O(h^{-1})$.

Theorem 7.2.1 (Convergence of composite quadrature formulas)**.**
For a composite quadrature formula Q based on a local quadrature formula of order $p \in \mathbb{N}$ holds

$$\exists C > 0: \quad \left| \int_I f(t)\,\mathrm{d}t - Q(f) \right| \le C h^p \left\| f^{(p)} \right\|_{L^\infty(I)} \quad \forall f \in C^p(I), \forall \mathcal{M}\ .$$

Proof. Apply interpolation error estimate (**??**). □

Example 7.2.2 (Quadrature errors for composite quadrature rules).

Composite quadrature rules based on

- trapezoidal rule (8.4.2) ➢ local order 2 (exact for linear functions),
- Simpson rule (7.1.4) ➢ local order 3 (exact for quadratic polynomials)

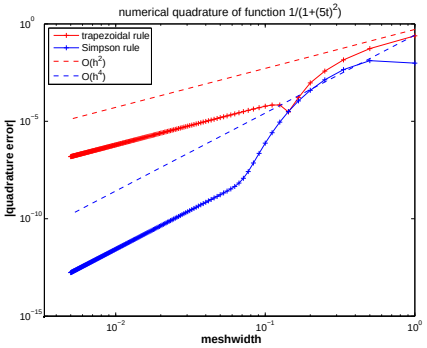
on equidistant mesh $\mathcal{M} := \{jh\}_{j=0}^{n}$, $h = 1/n$, $n \in \mathbb{N}$.

Code 7.2.3: composite trapezoidal rule (7.2.2)

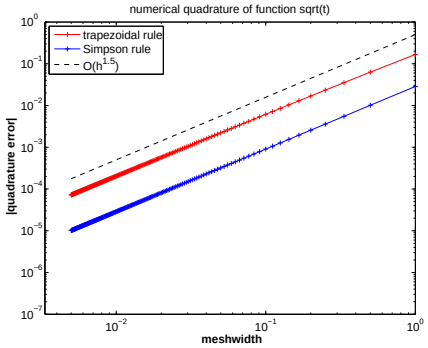
```
function res = trapezoidal(fnct,a,b,N)
%Numerical quadrature based on trapezoidal rule
%fnct handle to y = f(x)
%a,b bounds of integration interval
%N+1 = number of equidistant integration points (can be a vector)
res = [];
for n = N
  h = (b-a)/n;  x = (a:h:b); w = [0.5 ones(1,n-1) 0.5];
  res = [res; h, h*dot(w,feval(fnct,x))];
end
```

Code 7.2.4: composite Simpson rule (7.2.3)

```
function res = simpson(fnct,a,b,N)
%Numerical quadrature based on Simpson rule
%fnct handle to y = f(x)
%a,b bounds of integration interval
%N+1 = number of equidistant integration points (can be a vector)

res = [];
for n = N
  h = (b-a)/n;
  x = (a:h/2:b);
  fv = feval(fnct,x);
  val = sum(h*(fv(1:2:end-2)+4*fv(2:2:end-1)+fv(3:2:end)))/6;
  res = [res; h, val];
end
```



quadrature error, $f_1(t) := \frac{1}{1+(5t)^2}$ on $[0,1]$



quadrature error, $f_2(t) := \sqrt{t}$ on $[0,1]$

Asymptotic behavior of quadrature error $E(n) := \left|\int_0^1 f(t)\,\mathrm{d}t - Q_n(f)\right|$ for meshwidth "$h \to 0$"

☛ algebraic convergence $E(n) = O(h^\alpha)$ of order $\alpha > 0$, $n = h^{-1}$

➢ Sufficiently smooth integrand f_1: trapezoidal rule $\to \alpha = 2$, Simpson rule $\to \alpha = 4$!?

➢ singular integrand f_2: $\alpha = 3/2$ for trapezoidal rule **&** Simpson rule **!**

(lack of) smoothness of integrand limits convergence !

Simpson rule: order = 4 **?** investigate with MAPLE

```
> rule := 1/3*h*(f(2*h)+4*f(h)+f(0))
> err := taylor(rule - int(f(x),x=0..2*h),h=0,6);
```

$$err := \left(\frac{1}{90}\left(D^{(4)}\right)(f)(0)\,h^5 + O\left(h^6\right), h, 6\right)$$

➢ Simpson rule is of order 4, indeed **!**

Code 7.2.5: errors of composite trapezoidal and Simpson rule

```
function comruleerrs()
%Numerical quadrature on [0,1]

figure('Name','1/(1+(5t)^2)');
exact = atan(5)/5;
trres = trapezoidal(inline('1./(1+(5*x).^2)'),0,1,1:200);
smres = simpson(inline('1./(1+(5*x).^2)'),0,1,1:200);
loglog(trres(:,1),abs(trres(:,2)-exact),'r+-',...
       smres(:,1),abs(smres(:,2)-exact),'b+-',...
       trres(:,1),trres(:,1).^2*(trres(1,2)/trres(1,1)^2),'r--',...
       smres(:,1),smres(:,1).^4*(smres(1,2)/smres(1,1)^2),'b--');
set(gca,'fontsize',12);
title('numerical quadrature of function 1/(1+(5t)^2)','fontsize',14);
xlabel('{\bf meshwidth}','fontsize',14);
ylabel('{\bf |quadrature error|}','fontsize',14);
legend('trapezoidal rule','Simpson rule','O(h^2)','O(h^4)',2);
axis([1/300 1 10^(-15) 1]);
trp1 =
  polyfit(log(trres(end-100:end,1)),log(abs(trres(end-100:end,2)-exact)),1
smp1 =
  polyfit(log(smres(end-100:end,1)),log(abs(smres(end-100:end,2)-exact)),1
print -dpsc2 '../PICTURES/compruleerr1.eps';

figure('Name','sqrt(t)');
exact = 2/3;
trres = trapezoidal(inline('sqrt(x)'),0,1,1:200);
smres = simpson(inline('sqrt(x)'),0,1,1:200);
loglog(trres(:,1),abs(trres(:,2)-exact),'r+-',...
       smres(:,1),abs(smres(:,2)-exact),'b+-',...
       trres(:,1),trres(:,1).^(1.5)*(trres(1,2)/trres(1,1)^2),'k--');
set(gca,'fontsize',14);
title('numerical quadrature of function sqrt(t)','fontsize',14);
xlabel('{\bf meshwidth}','fontsize',14);
ylabel('{\bf |quadrature error|}','fontsize',14);
legend('trapezoidal rule','Simpson rule','O(h^{1.5})',2);
axis([1/300 1 10^(-7) 1]);
trp2 =
  polyfit(log(trres(end-100:end,1)),log(abs(trres(end-100:end,2)-exact)),1
smp2 =
  polyfit(log(smres(end-100:end,1)),log(abs(smres(end-100:end,2)-exact)),1
print -dpsc2 '../PICTURES/compruleerr2.eps';
```

◇

Remark 7.2.6 (Removing a singularity by transformation).

Ex. 7.2.2 ➢ lack of smoothness of integrand limits rate of algebraic convergence of composite quadrature rule for meshwidth $h \to 0$.

Idea: recover integral with smooth integrand by "analytic preprocessing"

Here is an example:

For $f \in C^\infty([0,b])$ compute $\int_0^b \sqrt{t} f(t)\,\mathrm{d}t$ via quadrature rule (→ Ex. 7.2.2)

substitution $s = \sqrt{t}$: $$\int_0^b \sqrt{t} f(t)\,\mathrm{d}t = \int_0^{\sqrt{b}} \boxed{2s^2 f(s^2)}\,\mathrm{d}s\,. \tag{7.2.4}$$

Then: Apply quadrature rule to smooth integrand

△

Example 7.2.7 (Convergence of equidistant trapezoidal rule).

Sometimes there are surprises: convergence of a composite quadrature rule is much better than predicted by the order of the local quadrature formula:

Equidistant trapezoidal rule (order 2), see (7.2.2)

$$\int_a^b f(t)\,\mathrm{d}t \approx T_m(f) := h\left(\tfrac{1}{2}f(a) + \sum_{k=1}^{m-1} f(kh) + \tfrac{1}{2}f(b)\right), \quad h := \frac{b-a}{m}\,. \tag{7.2.5}$$

Code 7.2.8: equidistant trapezoidal quadrature formula

```
function res = trapezoidal(fnct,a,b,N)
%Numerical quadrature based on trapezoidal rule
%fnct handle to y = f(x)
%a,b bounds of integration interval
%N+1 = number of equidistant integration points (can be a vector)
res = [];
for n = N
  h = (b-a)/n;  x = (a:h:b); w = [0.5 ones(1,n-1) 0.5];
  res = [res; h, h*dot(w,feval(fnct,x))];
end
```

1-periodic smooth (analytic) integrand

$$f(t) = \frac{1}{\sqrt{1 - a\sin(2\pi t - 1)}}, \quad 0 < a < 1\,.$$

(“exact value of integral”: use T_{500})

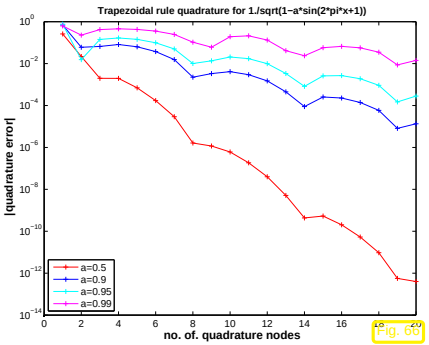


Fig. 66

quadrature error for $T_n(f)$ on $[0,1]$

exponential convergence !!

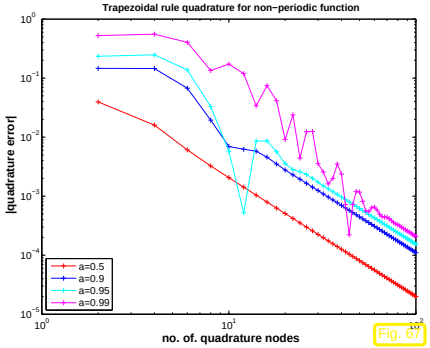


Fig. 67

quadrature error for $T_n(f)$ on $[0,\frac{1}{2}]$

merely algebraic convergence

Code 7.2.9: tracking error of equidistant trapezoidal quadrature formula

```
function traperr()

clear a;
global a;
l = 0; r = 0.5; %integration interval

N = 50;
a = 0.5; res05 = trapezoidal(@issin,l,r,1:N);
ex05 = trapezoidal(@issin,l,r,500); ex05 = ex05(1,2);
a = 0.9; res09 = trapezoidal(@issin,l,r,1:N);
ex09 = trapezoidal(@issin,l,r,500); ex09 = ex09(1,2);
a = 0.95; res95 = trapezoidal(@issin,l,r,1:N);
ex95 = trapezoidal(@issin,l,r,500); ex95 = ex95(1,2);
a = 0.99; res99 = trapezoidal(@issin,l,r,1:N);
ex99 = trapezoidal(@issin,l,r,500); ex99 = ex99(1,2);
figure('name','trapezoidal rule for non-periodic function');
loglog(1./res05(:,1),abs(res05(:,2)-ex05),'r+-',...
       1./res09(:,1),abs(res09(:,2)-ex09),'b+-',...
       1./res95(:,1),abs(res95(:,2)-ex95),'c+-',...
       1./res99(:,1),abs(res99(:,2)-ex99),'m+-');
set(gca,'fontsize',12);
legend('a=0.5','a=0.9','a=0.95','a=0.99',3);
xlabel('{\bf no. of. quadrature nodes}','fontsize',14);
ylabel('{\bf |quadrature error|}','fontsize',14);
title('{\bf Trapezoidal rule quadrature for non-periodic
  function}','fontsize',12);

print -depsc2 '../PICTURES/traperr2.eps';
```

```
clear a;
global a;
l = 0; r = 1; %integration interval
N = 20;
a = 0.5; res05 = trapezoidal(@issin,l,r,1:N);
ex05 = trapezoidal(@issin,l,r,500); ex05 = ex05(1,2);
a = 0.9; res09 = trapezoidal(@issin,l,r,1:N);
ex09 = trapezoidal(@issin,l,r,500); ex09 = ex09(1,2);
a = 0.95; res95 = trapezoidal(@issin,l,r,1:N);
ex95 = trapezoidal(@issin,l,r,500); ex95 = ex95(1,2);
a = 0.99; res99 = trapezoidal(@issin,l,r,1:N);
ex99 = trapezoidal(@issin,l,r,500); ex99 = ex99(1,2);
figure('name','trapezoidal rule for periodic function');
semilogy(1./res05(:,1),abs(res05(:,2)-ex05),'r+-',...
         1./res09(:,1),abs(res09(:,2)-ex09),'b+-',...
         1./res95(:,1),abs(res95(:,2)-ex95),'c+-',...
         1./res99(:,1),abs(res99(:,2)-ex99),'m+-');
set(gca,'fontsize',12);
legend('a=0.5','a=0.9','a=0.95','a=0.99',3);
xlabel('{\bf no. of quadrature nodes}','fontsize',14);
ylabel('{\bf |quadrature error|}','fontsize',14);
title('{\bf Trapezoidal rule quadrature for
  1./sqrt(1-a*sin(2*pi*x+1))}','fontsize',12);
```

```
print -depsc2 '../PICTURES/traperr1.eps';
```

Explanation:

$$f(t) = e^{2\pi i k t} \quad \blacktriangleright \quad \begin{cases} \int_0^1 f(t)\,\mathrm{d}t = \begin{cases} 0 & \text{, if } k \neq 0\,, \\ 1 & \text{, if } k = 0\,. \end{cases} \\ T_m(f) = \frac{1}{m}\sum_{l=0}^{m-1} e^{\frac{2\pi i}{m} l k} \overset{(??)}{=} \begin{cases} 0 & \text{, if } k \notin m\mathbb{Z}\,, \\ 1 & \text{, if } k \in m\mathbb{Z}\,. \end{cases} \end{cases}$$

$\blacktriangleright$

Equidistant trapezoidal rule T_m is exact for trigonometric polynomials of degree $< 2m$ **!**

It takes sophisticated tools from complex analysis to conclude exponential convergence for analytic integrands from the above observation.

$\diamond$

7.2 p. 401

7.3 p. 402

7.3 Monte-Carlo Quadrature

Monte-Carlo integration: instead of step-functions as quadrature,

$$I = \int_0^1 f(t)dt \approx h\sum_{i=1}^{N} f(t_i)$$

where $t_i = (i - \frac{1}{2})h$, $h = \frac{1}{N}$, the $\{f(t_i)\}$ may be reordered in any way. In particular, we can order them randomly:

$$I = \int_0^1 f(t)dt \approx \frac{1}{N}\sum_{i=1}^{N} f(t_i)$$

where $t_i \in (0,1)$ are uniformly distributed and sampled from a **random number generator**. A little more generally,

$$I = \int_a^b f(t)dt = |b-a|\langle f\rangle \approx |b-a|\frac{1}{N}\sum_{i=1}^{N} f(t_i)$$

where $t_i = a + (b-a)\cdot RNG$ (Matlab's `rand` e.g.).

What is really important is the statistical error, in d-dimensions,

$$\text{error} = \frac{k_d}{\sqrt{N}}.$$

The constant $k_d = \sqrt{\text{variance}}$.

The method is very general. For example, $A \subset R^d$,

$$\int_A f(\mathbf{x})dx_1dx_2\cdots d\mathbf{x}_d \approx |A|\langle f\rangle$$

where $|A|$ is the volume of region A.

The error is always $\propto N^{-1/2}$ **!**

But k_d can be reduced significantly. Two such methods: antithetic variates & control variates. An example,

$$I_0(x) = \frac{1}{\pi}\int_0^\pi e^{-x\cos t}dt\,.$$

$I_0(x)$ is a modified Bessel function of order zero.

Code 7.3.1: Plain Monte-Carlo

```
function dexraw
%
   M=100;
   asval=1.266065878;
   Ex = zeros(1,M);
   fprintf('A and S tables: I0(1) = %12.6e\n',asval);
```

```
  fprintf(' sample    variance       MC I0 val \n');
  fprintf(' ------   ------------    ------------ \n');
  for iN=1:5
     N = 10^iN;
     for j=1:M
        x = pi*rand(1,N);
        x = exp(cos(-x));
        z = sum(x);
        Ex(j) = z/N;
     end
     Eav    = sum(Ex);
     VEx    = dot(Ex,Ex);
     Eav    = Eav/M;
     VEx    = VEx/M;
     VEx    = VEx - Eav*Eav;
     fprintf('%6d %15.6e %15.6e\n',N,VEx,Eav)
  end
```

➢ General Principle of Monte Carlo: If, at any point of a Monte Carlo calculation, we can replace an estimate by an exact value, we shall replace an estimate by an exact value, we shall reduce the sampling error in the final result.

➢ Mark Kac: "You use Monte Carlo until you understand the problem."

Antithetic variates: usually only 1-D. Estimator for

$$I = I_a + I_b$$

where

$$I_a \approx \theta_a = \frac{1}{N}\sum_{i=1}^{N} f^{[a]}(x_i) \quad \text{and} \quad I_b \approx \theta_b = \frac{1}{N}\sum_{i=1}^{N} f^{[b]}(x_i)$$

so the variance is

$$\begin{aligned} \text{Var}_{ab} &= \langle(\theta_a + \theta_b - I)^2\rangle \\ &= \langle(\theta_a - I_a)^2\rangle + \langle(\theta_b - I_b)^2\rangle + 2\,\langle(\theta_a - I_a)(\theta_b - I_b)\rangle \\ &= \text{Var}_a + \text{Var}_b + 2\text{Cov}_{ab} \end{aligned}$$

If $\text{Cov}_{ab} = \langle(\theta_a - I_a)(\theta_b - I_b)\rangle < 0$ (negatively correlated), Var_{ab} is reduced.

Our example: break the integral into two pieces $0 < x < \pi/2$ and $x + \pi/2$. The new integrand is $e^{\sin(x)} + e^{-\cos(x)}$, for $0 < x < \pi/2$, and strictly **monotone**.

$$\begin{aligned} I_0(1) &\approx I_+ + I_- \\ &= \frac{1}{4N}\sum_{i=1}^{N} e^{\sin \pi u_i/2} + e^{\sin \pi(1-u_i)/2} \\ &\quad + e^{-\cos \pi u_i/2} + e^{-\cos \pi(1-u_i)/2}. \end{aligned}$$

Code 7.3.2: Antitetic Variates Monte-Carlo

```
function dexatv
%
  M=100;
  asval=1.266065878;
  Ex = zeros(1,M);
  fprintf(' A and S tables:  I0(1) = %12.6e\n',asval);
  fprintf(' sample    variance       MC I0 val \n');
  fprintf(' ------   ------------    ------------ \n');
  pi2 = 0.5*pi;
  for iN=1:5
     N = 10^iN;
     for j=1:M
        up  = pi2*rand(1,N);
        dn  = pi2-up;
        atv = exp(sin(up)) + exp(sin(dn)) + exp(-cos(up)) +
          exp(-cos(dn));
        Ex(j) = sum(atv)/(4*N);
     end
     Eav    = sum(Ex);
     VEx    = dot(Ex,Ex);
     Eav    = Eav/M;
     VEx    = VEx/M;
     VEx    = VEx - Eav*Eav;
     fprintf('%6d %15.6e %15.6e\n',N,VEx,Eav)
  end
```

Control variates Integral

$$I = \int_0^1 f(t)dt$$

can be re-written

$$\begin{aligned} I &= \int_0^1 (f(t) - \phi(t))dt + \int_0^1 \phi(t)dt \\ &\approx \frac{1}{N}\sum_{i=1}^{N}[f(t_i) - \phi(t_i)] + I_\phi \end{aligned}$$

Pick ϕ:

- $\phi(u) \approx g(u)$ is *nearby*, and
- $I_\phi = \int \phi(u)du$ is known.

Variance is reduced if

$$\text{var}(f - \phi) \ll \text{var}(f)$$

To see how it works, our problem

$$\begin{aligned} f(t) &= e^{-cos(\pi t)} \\ &= 1 - cos(\pi t) + \frac{1}{2}(cos(\pi t))^2 + ... \\ \phi(t) &= \text{first three terms} \end{aligned}$$

Code 7.3.3: Control Variates Monte-Carlo

```
function dexatv
%
   M=100;
   asval=1.266065878;
   Ex = zeros(1,M);
   fprintf('  A and S tables:   I0(1) =  %12.6e\n',asval);
   fprintf('  sample      variance         MC  I0 val  \n');
   fprintf('  ------   ------------    ----  ------------  \n');
   for iN=1:5
      N = 10^iN;
      for j=1:M
         x   = pi*rand(1,N);
         ctv = exp(-cos(x)) - 1. + cos(x) - 0.5*cos(x).*cos(x);
         Ex(j) = 1.25 + sum(ctv)/N;
      end
      Eav   = sum(Ex);
      VEx   = dot(Ex,Ex);
      Eav   = Eav/M;
      VEx   = VEx/M;
      VEx   = VEx - Eav*Eav;
      fprintf('%6d  %15.6e  %15.6e\n',N,VEx,Eav)
   end
```

Importance Sampling

Idea: Concentrate the distribution of the sample points in the parts of the interval that are of most importance instead of spreading them out evenly.
Importance Sampling:

$$\theta = \int_0^1 f(x)\,dx = \int_0^1 \frac{f(x)}{g(x)} g(x)\,dx = \int_0^1 \frac{f(x)}{g(x)} dG(x),$$

where g and G satisfy

$$G(x) = \int_0^x g(y)\,dy, \quad G(1) = \int_0^1 g(y)\,dy = 1,$$

and $G(x)$ is a distribution function. Variance

$$\sigma^2_{f/g} = \int_0^1 (f(x)/g(x) - \theta)^2 dG(x)$$

How to select a good sampling function?

How about $g = cf$? g must be simple enough for us to know its integral theoretically.

7.4 Essential Skills Learned in Chapter 7

You should know:

- several (compozite) polynomial quadrature formulas with their convergence order
- what is special about the trapezoidal rule
- Gaussian quadrature rules
- particularities of Monte-Carlo integration
- how to reduce the variance by Monte-Carlo integration

Part II

Integration of Ordinary Differential Equations