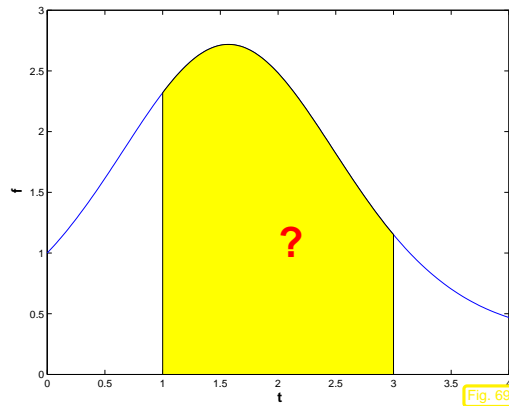


10

Numerical Quadrature

Numerical quadrature

- = Approximate evaluation of $\int_{\Omega} f(\mathbf{x}) \, d\mathbf{x}$, integration domain $\Omega \subset \mathbb{R}^d$
- Continuous function $f: \Omega \subset \mathbb{R}^d \mapsto \mathbb{R}$ only available as function $y = f(\mathbf{x})$ (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.



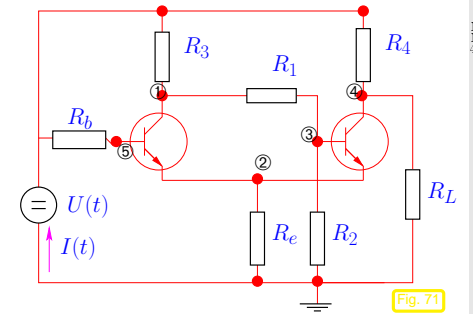
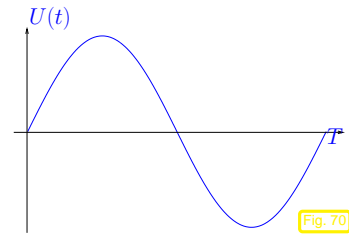
Numerical quadrature methods

approximate

$$\int_a^b f(t) \, dt$$

Example 10.0.1 (Heating production in electrical circuits).

Time-harmonic excitation:



Integrating power $P = UI$ over period $[0, T]$ yields heat production per period:

$$W_{\text{therm}} = \int_0^T U(t)I(t) \, dt, \quad \text{where } I = I(U).$$

function $I = \text{current}(U)$ involves solving non-linear system of equations, see Ex. 1.0.1!

10.1 Quadrature Formulas

n -point quadrature formula on $[a, b]$: $\int_a^b f(t) \, dt \approx Q_n(f) := \sum_{j=1}^n w_j^n f(c_j^n)$. (10.1.1)
(n -point quadrature rule)

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

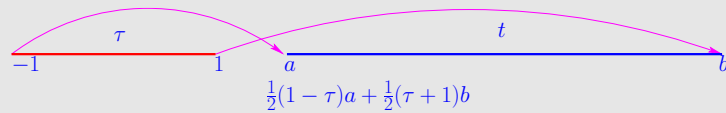
Remark 10.1.1 (Transformation of quadrature rules).

Given: quadrature formula $(\hat{c}_j, \hat{w}_j)_{j=1}^n$ on reference interval $[-1, 1]$



Idea: transformation formula for integrals

$$\int_a^b f(t) \, dt = \frac{1}{2}(b-a) \int_{-1}^1 \hat{f}(\tau) \, d\tau, \quad \hat{f}(\tau) := f\left(\frac{1}{2}(1-\tau)a + \frac{1}{2}(\tau+1)b\right). \quad (10.1.2)$$



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

$$\int_a^b f(t) dt \approx \frac{1}{2}(b-a) \sum_{j=1}^n \hat{w}_j \hat{f}(\hat{c}_j) = \sum_{j=1}^n w_j f(c_j) \quad \text{with} \quad c_j = \frac{1}{2}(1 - \hat{c}_j)a + \frac{1}{2}(1 + \hat{c}_j)b, \\ w_j = \frac{1}{2}(b-a)\hat{w}_j.$$

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

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

△

Inevitable for generic integrand:

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

Given families of quadrature rules $\{Q_n\}_n$ with quadrature weights $\{w_j^n, j = 1, \dots, n\}_{n \in \mathbb{N}}$ and quadrature nodes $\{c_j^n, j = 1, \dots, n\}_{n \in \mathbb{N}}$ we

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

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

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 .



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

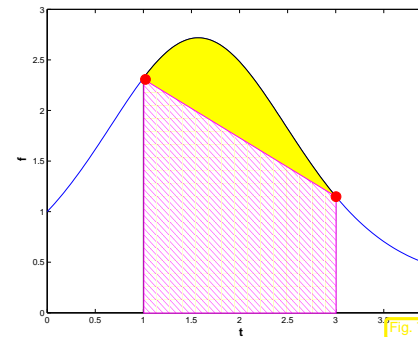


Idea: **Gaussian** quadrature: Choose the n weights and the n points such that the error $E(n) = 0$ for all polynomials f of degree $2n - 1$.

Example 10.1.2 (Newton-Cotes formulas).

• $n = 1$: Trapezoidal rule

$$\hat{Q}_{\text{trp}}(f) := \frac{1}{2}(f(0) + f(1)) \quad (10.1.3) \\ \left(\int_a^b f(t) dt \approx \frac{b-a}{2}(f(a) + f(b)) \right)$$



• $n = 2$: Simpson rule

$$\frac{h}{6} \left(f(0) + 4f\left(\frac{1}{2}\right) + f(1) \right) \quad \left(\int_a^b f(t) dt \approx \frac{b-a}{6} \left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right) \right) \quad (10.1.4)$$

Remark 10.1.3 (Error estimates for polynomial quadrature).

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

$$f \in C^n([a, b]) \Rightarrow \left| \int_a^b f(t) dt - Q_n(f) \right| \leq \frac{1}{n!} (b-a)^{n+1} \|f^{(n)}\|_{L^\infty([a, b])}. \quad (10.1.5)$$

△

Example 10.1.4 (2-point quadrature rule of order 4).

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

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

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

$$\begin{aligned} \int_{-1}^1 1 dt = 2 &= 1w_1 + 1w_2, & \int_{-1}^1 t dt = 0 &= c_1w_1 + c_2w_2 \\ \int_{-1}^1 t^2 dt = \frac{2}{3} &= c_1^2w_1 + c_2^2w_2, & \int_{-1}^1 t^3 dt = 0 &= c_1^3w_1 + c_2^3w_2. \end{aligned} \quad (10.1.6)$$

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: $\{w_2 = 1, w_1 = 1, c_1 = 1/3\sqrt{3}, c_2 = -1/3\sqrt{3}\}$

► quadrature formula: $\int_{-1}^1 f(x) dx \approx f\left(\frac{1}{\sqrt{3}}\right) + f\left(-\frac{1}{\sqrt{3}}\right)$ (10.1.7)

Remark 10.1.5 (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: c_j, w_j from tables!

Code 10.1.6: 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) = ev(:,i)/norm(ev(:,i)); end
6 x=diag(ew); w=(2*(ev(1,:)'.*ev(1,:)))';
```

Idea: Clenshaw-Curtis quadrature:



$$\int_{-1}^1 f(x) dx = \int_0^\pi f(\cos \theta) \sin \theta d\theta = \sum_{\text{even } k} \frac{2a_k}{1-k^2} \quad (10.1.8)$$

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 10.1.7: tracking errors on quadrature rules

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

Example 10.1.8 (Error of (non-composite) quadratures).

Code 10.1.9: important polynomial quadrature rules

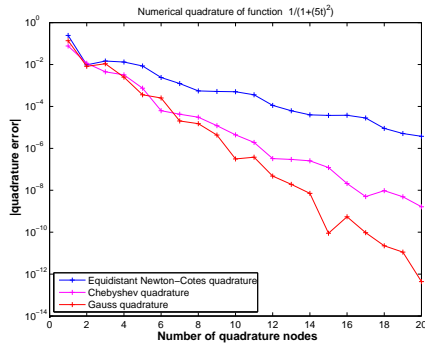
```
1 function res = numquad(f,a,b,N,mode)
2 % Numerical quadrature on [a,b] by polynomial quadrature formula
3 % f -> function to be integrated (handle), must support vector arguments
```

```
4 % a,b -> integration interval [a,b] (endpoints included)
5 % N -> Maximal degree of polynomial
6 % mode: equidistant, Chebychev, Gauss
7
8 if ( nargin < 5 ), mode = 'equidistant'; end
9
10 res = [];
11
12 if strcmp(mode, 'Gauss')
13     for deg=1:N
14         [gx,w] = gaussQuad(deg);
15         x = 0.5*(b-a)*gx+0.5*(a+b);
16         y = feval(f,x);
17         res = [res; deg, 0.5*(b-a)*dot(w,y)];
18     end
19 else
20     p = (N+1:-1:1);
21     w = (b.^p - a.^p) ./ p;
22     for deg=1:N
23         if strcmp(mode, 'Chebychev')
24             x = 0.5*(b-a)*cos((2*(0:deg)+1)/(2*deg+2)*pi)+0.5*(a+b);
25         else
26             x = (a:(b-a)/deg:b);
```

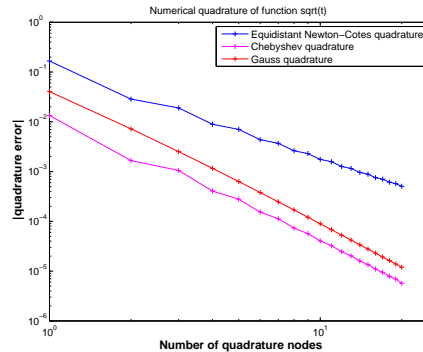
```

27 end
28 y = feval(f,x);
29 poly = polyfit(x,y,deg);
30 res = [res; deg, dot(w(N+1-deg:N+1),poly)];
31 end
32 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 $\epsilon_n := \left| \int_0^1 f(t) dt - Q_n(f) \right|$ for " $n \rightarrow \infty$ ":

- 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 10.1.10: tracking errors on quadrature rules

```

1 function numquaders()
2 % Numerical quadrature on [0,1]
3 N = 20;
4
5 figure('Name','1/(1+(5t)^2)');
6 exact = atan(5)/5;
7 eqdres = numquad(inline('1./(1+(5*x).^2)'),0,1,N,'equidistant');
8 chbres = numquad(inline('1./(1+(5*x).^2)'),0,1,N,'Chebychev');
9 gaures = numquad(inline('1./(1+(5*x).^2)'),0,1,N,'Gauss');
10 semilogy(eqdres(:,1),abs(eqdres(:,2)-exact),'b+-',...

```

Numerical Methods 401-0654

```

11 chbres(:,1),abs(chbres(:,2)-exact),'m+-',...
12 gaures(:,1),abs(gaures(:,2)-exact),'r+-');
13 set(gca,'fontsize',12);
14 title('Numerical quadrature of function 1/(1+(5t)^2)');
15 xlabel('\bf_{Number_of_quadrature_nodes}','fontsize',14);
16 ylabel('\bf_{quadrature_error}','fontsize',14);
17 legend('Equidistant-Newton-Cotes quadrature',...
18 'Clenshaw-Curtis quadrature',...
19 'Gauss quadrature',3);
20 eqdp1 = polyfit(eqdres(:,1),log(abs(eqdres(:,2)-exact)),1)
21 chbp1 = polyfit(chbres(:,1),log(abs(chbres(:,2)-exact)),1)
22 gaup1 = polyfit(gaures(:,1),log(abs(gaures(:,2)-exact)),1)
23 print -depsc2 '../PICTURES/numquaderr1.eps';
24
25 figure('Name','sqrt(t)');
26 exact = 2/3;
27 eqdres = numquad(inline('sqrt(x)'),0,1,N,'equidistant');
28 chbres = numquad(inline('sqrt(x)'),0,1,N,'Chebychev');
29 gaures = numquad(inline('sqrt(x)'),0,1,N,'Gauss');
30 loglog(eqdres(:,1),abs(eqdres(:,2)-exact),'b+-',...
31 chbres(:,1),abs(chbres(:,2)-exact),'m+-',...
32 gaures(:,1),abs(gaures(:,2)-exact),'r+-');
33 set(gca,'fontsize',12);

```

V. Gradinaru D-ITET, D-MATL

10.1 p. 493

Numerical Methods 401-0654

V. Gradinaru D-ITET, D-MATL

10.1 p. 49

Numerical Methods 401-0654

```

34 axis([1 25 0.00001 1]);
35 title('Numerical quadrature of function sqrt(t)');
36 xlabel('\bf_{Number_of_quadrature_nodes}','fontsize',14);
37 ylabel('\bf_{quadrature_error}','fontsize',14);
38 legend('Equidistant-Newton-Cotes quadrature',...
39 'Clenshaw-Curtis quadrature',...
40 'Gauss quadrature',1);
41 eqdp2 = polyfit(log(eqdres(:,1)),log(abs(eqdres(:,2)-exact)),1)
42 chbp2 = polyfit(log(chbres(:,1)),log(abs(chbres(:,2)-exact)),1)
43 gaup2 = polyfit(log(gaures(:,1)),log(abs(gaures(:,2)-exact)),1)
44 print -depsc2 '../PICTURES/numquaderr2.eps';

```

V. Gradinaru D-ITET, D-MATL

Numerical Methods 401-0654

V. Gradinaru D-ITET, D-MATL

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

10.1 p. 494

10.2 p. 49

10.2 Composite Quadrature

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

$$\int_a^b f(t) dt = \sum_{j=1}^m \int_{x_{j-1}}^{x_j} f(t) dt. \quad (10.2.1)$$

Recall (10.1.5): 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 (10.2.1),
 - using the intended quadrature formula on each sub-interval $[x_{j-1}, x_j]$.

Note: Increase 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, \rightarrow Sect.??) $\mathcal{M} := \{a = x_0 < x_1 < \dots < x_m = b\}$
 - Apply quadrature formulas on sub-intervals $I_j := [x_{j-1}, x_j], j = 1, \dots, m$, and sum up.

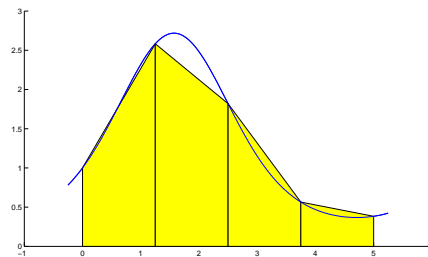
composite quadrature rule

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

Example 10.2.1 (Simple composite polynomial quadrature rules).

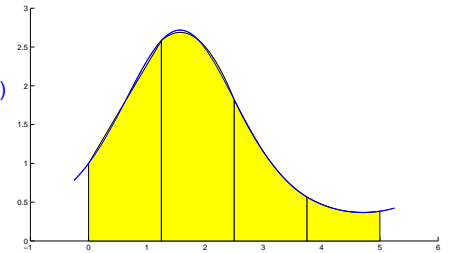
Composite trapezoidal rule, cf. (11.6.2)

$$\int_a^b f(t) dt = \frac{1}{2}(x_1 - x_0)f(a) + \sum_{j=1}^{m-1} \frac{1}{2}(x_{j+1} - x_{j-1})f(x_j) + \frac{1}{2}(x_m - x_{m-1})f(b). \quad (10.2.2)$$



Composite Simpson rule, cf. (10.1.4)

$$\int_a^b f(t) dt = \frac{1}{6}(x_1 - x_0)f(a) + \sum_{j=1}^{m-1} \frac{1}{6}(x_{j+1} - x_{j-1})f(x_j) + \sum_{j=1}^m \frac{2}{3}(x_j - x_{j-1})f\left(\frac{1}{2}(x_j + x_{j-1})\right) + \frac{1}{6}(x_m - x_{m-1})f(b). \quad (10.2.3)$$



Formulas (10.2.2), (10.2.3) directly suggest efficient implementation with minimal number of f -evaluations.

Focus: asymptotic behavior of quadrature error for

$$\text{mesh width } h := \max_{j=1, \dots, m} |x_j - x_{j-1}| \rightarrow 0$$

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 10.2.1 (Convergence of composite quadrature formulas).

For a composite quadrature formula Q based on a local quadrature formula of order $p \in \mathbb{N}$ holds

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

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

Example 10.2.2 (Quadrature errors for composite quadrature rules).

Composite quadrature rules based on

- trapezoidal rule (11.6.2) \triangleright local order 2 (exact for linear functions),
- Simpson rule (10.1.4) \triangleright local order 3 (exact for quadratic polynomials)

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

Code 10.2.3: composite trapezoidal rule (10.2.2)

```

1 function res = trapezoidal(fnct,a,b,N)
2 % Numerical quadrature based on trapezoidal rule
3 % fnct handle to y = f(x)
4 % a,b bounds of integration interval
5 % N+1 = number of equidistant integration points (can be a vector)
6 res = [];
7 for n = N
8     h = (b-a)/n; x = (a:h:b); w = [0.5 ones(1,n-1) 0.5];
9     res = [res; h, h*dot(w, feval(fnct,x))];
10 end

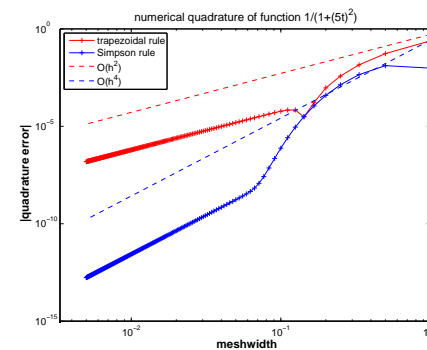
```

Code 10.2.4: composite Simpson rule (10.2.3)

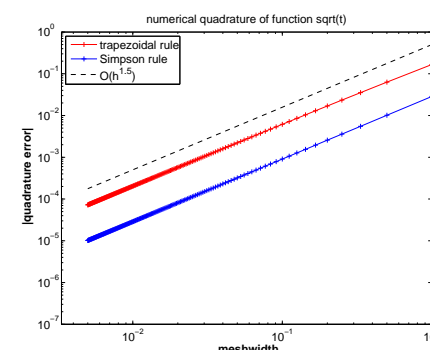
```

1 function res = simpson(fnct,a,b,N)
2 % Numerical quadrature based on Simpson rule
3 % fnct handle to y = f(x)
4 % a,b bounds of integration interval
5 % N+1 = number of equidistant integration points (can be a vector)
6
7 res = [];
8 for n = N
9     h = (b-a)/n;
10    x = (a:h/2:b);
11    fv = feval(fnct,x);
12    val = sum(h*(fv(1:2:end-2)+4*fv(2:2:end-1)+fv(3:2:end)))/6;
13    res = [res; h, val];
14 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 $E(n) := \left| \int_0^1 f(t) dt - Q_n(f) \right|$ for meshwidth " $h \rightarrow 0$ "

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

\triangleright Sufficiently smooth integrand f_1 : trapezoidal rule $\rightarrow \alpha = 2$, Simpson rule $\rightarrow \alpha = 4$!

\triangleright 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);

```

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

\triangleright Simpson rule is of order 4, indeed !

Code 10.2.5: errors of composite trapezoidal and Simpson rule

```

1 function comruleerrs()
2 % Numerical quadrature on [0,1]
3
4 figure('Name','1/(1+(5t)^2)');
5 exact = atan(5)/5;
6 trres = trapezoidal(inline('1./(1+(5*x).^2)'),0,1,1:200);
7 smres = simpson(inline('1./(1+(5*x).^2)'),0,1,1:200);

```

```

8 loglog ( trres (:,1) ,abs(trres (:,2)-exact) , 'r+' , ...
9         smres (:,1) ,abs(smres (:,2)-exact) , 'b+' , ...
10        trres (:,1) ,trres (:,1).^2*(trres (1,2)/trres (1,1)^2) , 'r-' , ...
11        smres (:,1) ,smres (:,1).^4*(smres (1,2)/smres (1,1)^2) , 'b-' );
12 set(gca, 'fontsize',12);
13 title ('numerical_quadrature_of_function_1/(1+(5t)^2)' , 'fontsize',14);
14 xlabel ('\bf_meshwidth' , 'fontsize',14);
15 ylabel ('\bf_|quadrature_error|' , 'fontsize',14);
16 legend ('trapezoidal_rule' , 'Simpson_rule' , 'O(h^2)' , 'O(h^4)' , 2);
17 axis([1/300 1 10^(-15) 1]);
18 trp1 =
19     polyfit(log(trres(end-100:end,1)) , log(abs(trres(end-100:end,2)-exact)) , 1);
20 smp1 =
21     polyfit(log(smres(end-100:end,1)) , log(abs(smres(end-100:end,2)-exact)) , 1);
22 print -dpsc2 ' ../PICTURES/compruleerr1.eps';
23
24 figure('Name','sqrt(t)');
25 exact = 2/3;
26 trres = trapezoidal(inline('sqrt(x)') , 0,1,1:200);
27 smres = simpson(inline('sqrt(x)') , 0,1,1:200);
28 loglog ( trres (:,1) ,abs(trres (:,2)-exact) , 'r+' , ...
29         smres (:,1) ,abs(smres (:,2)-exact) , 'b+' , ...
30         trres (:,1) ,trres (:,1).^1.5*(trres (1,2)/trres (1,1)^2) , 'k-' );

```

```

29 set(gca, 'fontsize',14);
30 title ('numerical_quadrature_of_function_sqrt(t)' , 'fontsize',14);
31 xlabel ('\bf_meshwidth' , 'fontsize',14);
32 ylabel ('\bf_|quadrature_error|' , 'fontsize',14);
33 legend ('trapezoidal_rule' , 'Simpson_rule' , 'O(h^{1.5})' , 2);
34 axis([1/300 1 10^(-7) 1]);
35 trp2 =
36     polyfit(log(trres(end-100:end,1)) , log(abs(trres(end-100:end,2)-exact)) , 1);
37 smp2 =
38     polyfit(log(smres(end-100:end,1)) , log(abs(smres(end-100:end,2)-exact)) , 1);
39 print -dpsc2 ' ../PICTURES/compruleerr2.eps';

```

Remark 10.2.6 (Removing a singularity by transformation).

Ex. 10.2.2 > lack of smoothness of integrand limits rate of algebraic convergence of composite quadrature rule for meshwidth $h \rightarrow 0$.

Idea: recover integral with smooth integrand by “analytic preprocessing”

Numerical
Methods
401-0654

Here is an example:

For $f \in C^\infty([0, b])$ compute $\int_0^b \sqrt{t}f(t) dt$ via quadrature rule (\rightarrow Ex. 10.2.2)

$$\text{substitution } s = \sqrt{t}: \int_0^b \sqrt{t}f(t) dt = \int_0^{\sqrt{b}} 2s^2 f(s^2) ds. \quad (10.2.4)$$

Then:

Apply quadrature rule to smooth integrand

V.
Gradinaru
D-ITET,
D-MATL

Example 10.2.7 (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 (10.2.2)

$$\int_a^b f(t) dt \approx T_m(f) := h \left(\frac{1}{2}f(a) + \sum_{k=1}^{m-1} f(kh) + \frac{1}{2}f(b) \right), \quad h := \frac{b-a}{m}. \quad (10.2.5)$$

10.2
p. 505

10.2
p. 50

Numerical
Methods
401-0654

Code 10.2.8: equidistant trapezoidal quadrature formula

```

1 function res = trapezoidal(fnct,a,b,N)
2 % Numerical quadrature based on trapezoidal rule
3 % fnct handle to y = f(x)
4 % a,b bounds of integration interval
5 % N+1 = number of equidistant integration points (can be a vector)
6 res = [];
7 for n = N
8     h = (b-a)/n; x = (a:h:b); w = [0.5 ones(1,n-1) 0.5];
9     res = [res; h, h*dot(w, feval(fnct,x))];
10 end

```

V.
Gradinaru
D-ITET,
D-MATL

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})

10.2
p. 506

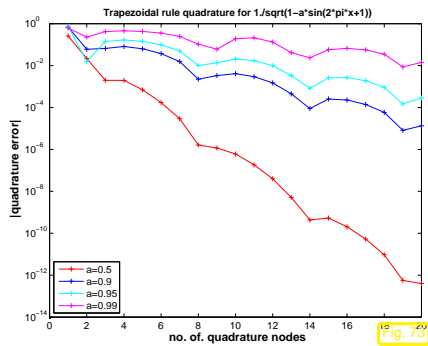
10.2
p. 50

Numerical
Methods
401-0654

V.
Gradinaru
D-ITET,
D-MATL

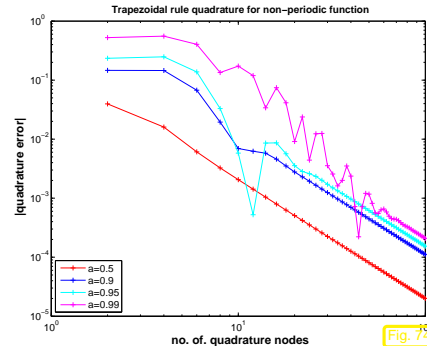
Numerical
Methods
401-0654

V.
Gradinaru
D-ITET,
D-MATL



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

exponential convergence !!



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

merely algebraic convergence

Code 10.2.9: tracking error of equidistant trapezoidal quadrature formula

```

1 function traperr()
2
3 clear a;
4 global a;
5 l = 0; r = 0.5; %integration interval
6
7 N = 50;
8 a = 0.5; res05 = trapezoidal(@issin,l,r,1:N);
9 ex05 = trapezoidal(@issin,l,r,500); ex05 = ex05(1,2);
10 a = 0.9; res09 = trapezoidal(@issin,l,r,1:N);
11 ex09 = trapezoidal(@issin,l,r,500); ex09 = ex09(1,2);
12 a = 0.95; res95 = trapezoidal(@issin,l,r,1:N);
13 ex95 = trapezoidal(@issin,l,r,500); ex95 = ex95(1,2);
14 a = 0.99; res99 = trapezoidal(@issin,l,r,1:N);
15 ex99 = trapezoidal(@issin,l,r,500); ex99 = ex99(1,2);
16 figure('name','trapezoidal_rule_for_non-periodic_function');
17 loglog(1./res05(:,1),abs(res05(:,2)-ex05),'r+',...
18        1./res09(:,1),abs(res09(:,2)-ex09),'b+',...
19        1./res95(:,1),abs(res95(:,2)-ex95),'c+',...
20        1./res99(:,1),abs(res99(:,2)-ex99),'m+',...);
21 set(gca,'fontsize',12);
22 legend('a=0.5','a=0.9','a=0.95','a=0.99',3);
23 xlabel('\bf_no_of_quadrature_nodes','fontsize',14);
24 ylabel('\bf|quadrature_error|','fontsize',14);
25 title('\bf_Trapezoidal_rule_quadrature_for_non-periodic_
26        function','fontsize',12);
27
28 print -depsc2 '../PICTURES/traperr2.eps';

```

```

28 clear a;
29 global a;
30 l = 0; r = 1; %integration interval
31 N = 20;
32 a = 0.5; res05 = trapezoidal(@issin,l,r,1:N);
33 ex05 = trapezoidal(@issin,l,r,500); ex05 = ex05(1,2);
34 a = 0.9; res09 = trapezoidal(@issin,l,r,1:N);
35 ex09 = trapezoidal(@issin,l,r,500); ex09 = ex09(1,2);
36 a = 0.95; res95 = trapezoidal(@issin,l,r,1:N);
37 ex95 = trapezoidal(@issin,l,r,500); ex95 = ex95(1,2);
38 a = 0.99; res99 = trapezoidal(@issin,l,r,1:N);
39 ex99 = trapezoidal(@issin,l,r,500); ex99 = ex99(1,2);
40 figure('name','trapezoidal_rule_for_periodic_function');
41 semilogy(1./res05(:,1),abs(res05(:,2)-ex05),'r+',...
42          1./res09(:,1),abs(res09(:,2)-ex09),'b+',...
43          1./res95(:,1),abs(res95(:,2)-ex95),'c+',...
44          1./res99(:,1),abs(res99(:,2)-ex99),'m+',...);
45 set(gca,'fontsize',12);
46 legend('a=0.5','a=0.9','a=0.95','a=0.99',3);
47 xlabel('\bf_no_of_quadrature_nodes','fontsize',14);
48 ylabel('\bf|quadrature_error|','fontsize',14);
49 title('\bf_Trapezoidal_rule_quadrature_for_
50        1./sqrt(1-a*sin(2*pi*x+1))','fontsize',12);

```

Explanation:

$$f(t) = e^{2\pi ikt} \rightarrow \begin{cases} \int_0^1 f(t) dt = \begin{cases} 0, & \text{if } k \neq 0, \\ 1, & \text{if } k = 0. \end{cases} \\ T_m(f) = \frac{1}{m} \sum_{l=0}^{m-1} e^{\frac{2\pi i}{m}lk} \stackrel{(6.2.2)}{=} \begin{cases} 0, & \text{if } k \notin m\mathbb{Z}, \\ 1, & \text{if } k \in m\mathbb{Z}. \end{cases} \end{cases}$$

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.



10.3 Essential Skills Learned in Chapter 10

You should know:

- several (compozite) polynomial quadrature formulas with their convergence order
- what is special about the trapezoidal rule
- Gaussian quadrature rules

x

Part III

Integration of Ordinary Differential Equations

11

Single Step Methods

11.1 Initial value problems (IVP) for ODEs

Some grasp of the meaning and theory of ordinary differential equations (ODEs) is indispensable for understanding the construction and properties of numerical methods. Relevant information can be found in [52, Sect. 5.6, 5.7, 6.5].

Example 11.1.1 (Growth with limited resources). [1, Sect. 1.1]

$y : [0, T] \mapsto \mathbb{R}$: bacterial population density as a function of time

Model: autonomous **logistic differential equations**

$$\dot{y} = f(y) := (\alpha - \beta y) y \tag{11.1.1}$$

Notation (Newton): $\dot{} \hat{=} (\text{total derivative with respect to time } t)$

- $y \hat{=} \text{population density, } [y] = \frac{1}{\text{m}^2}$
- growth rate $\alpha - \beta y$ with growth coefficients $\alpha, \beta > 0$, $[\alpha] = \frac{1}{\text{s}}$, $[\beta] = \frac{\text{m}^2}{\text{s}}$: decreases due to more fierce competition as population density increases.

Note: we can only compute a solution of (11.1.3), when provided with an **initial value** $y(0)$.

The logistic differential equation arises in autocatalytic reactions (as in haloform reaction, tin pest, binding of oxygen by hemoglobin or the spontaneous degradation of aspirin into salicylic acid and acetic acid, causing very old aspirin in sealed containers to smell mildly of vinegar):



As $\dot{c}_A = -r$ and $\dot{c}_B = -r + 2r = r$ we have that $c_A + c_B = c_A(0) + c_B(0) = D$ is constant and we get two decoupled equations

$$\dot{c}_A = -k(D - c_A)c_A \tag{11.1.3}$$