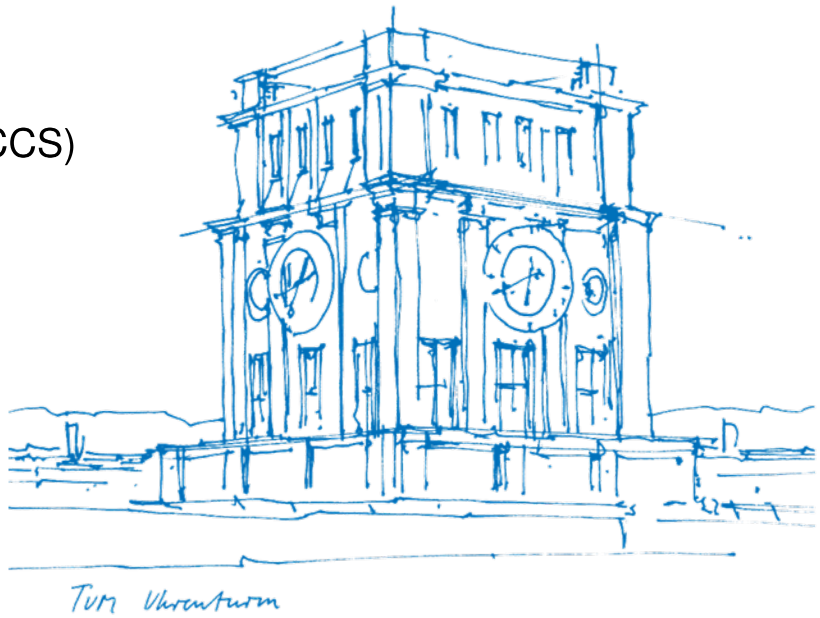
Algorithms for Scientific Computing

Hierarchical Methods and Sparse Grids: Archimedes’ Quadrature in 1D

Felix Dietrich
Technische Universität München
Department of Informatics 5
Chair of Scientific Computing in Computer Science (SCCS)
Summer 2020



Why Numerical Quadrature?

Integration integral part in many applications, e.g.:

- Determine volumes (e.g. of beer/wine barrels)
- Option pricing (expectation values)
- Defuzzification for fuzzy controller
- Radiosity (accumulating light)
- discretization: Finite Volume/Finite Elements

Why Numerical Quadrature?

Integration integral part in many applications, e.g.:

- Determine volumes (e.g. of beer/wine barrels)
- Option pricing (expectation values)
- Defuzzification for fuzzy controller
- Radiosity (accumulating light)
- discretization: Finite Volume/Finite Elements

Often no analytical solution available

$\Rightarrow$ Approximate solution: **"numerical quadrature"**

- Typical approach: approximate/interpolate, then integrate

Why Numerical Quadrature?

Integration integral part in many applications, e.g.:

- Determine volumes (e.g. of beer/wine barrels)
- Option pricing (expectation values)
- Defuzzification for fuzzy controller
- Radiosity (accumulating light)
- discretization: Finite Volume/Finite Elements

Often no analytical solution available

$\Rightarrow$ Approximate solution: **“numerical quadrature”**

- Typical approach: approximate/interpolate, then integrate

Core-problem: representation of functions in several variables

- In **higher-dimensional settings** only stochastic (“Monte Carlo”, etc.) or hierarchical methods available
- Here: focus on hierarchical methods

One-Dimensional Quadrature

Approximations for the definite integral

$$F_1(f, a, b) := \int_a^b f(x)\, dx$$

for $f : [a, b] \to \mathbb{R}$

- We first consider classical methods
  $\to$ Trapezoidal Rule
- Then hierarchical approach
- Assumption in the following: "f is sufficiently smooth"
  (i.e., all desired derivatives of f exist and are continuous)

Trapezoidal Rule, Simpson Rule

Classical methods for numerical quadrature: Newton-Cotes formulas

- $f(x_i)$ at equally spaced points $x_i = a + ih$
- Integrate
$$\int_a^b f(x) \approx \sum w_i f(x_i)$$
- choose weights w_i such that integration is exact for polynomials up to a certain degree
- typically result from an interpolation problem

Trapezoidal Rule, Simpson Rule

Trapezoidal rule

- Interpolate at interval boundaries with linear function

$$F_1 \approx T := (b-a)\frac{f(a)+f(b)}{2}$$

Simpson rule

- Interpolate at interval boundaries and midpoint with quadratic function

$$F_1 \approx S := (b-a)\frac{f(a)+4f\left(\frac{a+b}{2}\right)+f(b)}{6}$$

Quadrature Error

- Error terms are known for the two methods:

$$
\begin{aligned}
|T - F_1| &\leq \frac{M_2}{12}(b-a)^3 \\
|S - F_1| &\leq \frac{M_4}{2880}(b-a)^5
\end{aligned}
$$

- M_2 and M_4 are bounds for the second, resp. fourth, derivative:

$$
\begin{aligned}
M_2 &:= \sup_{x \in [a,b]} |f''(x)|, \\
M_4 &:= \sup_{x \in [a,b]} |f^{(4)}(x)|.
\end{aligned}
$$

$\Rightarrow$ assumes that these derivatives and such bounds exist!

Composite Quadrature Rules

- Error bounds suggest the following improvement:
  - Split interval $[a,b]$ into smaller subintervals
  - Apply simple quadrature rule in each of them
- Simplest case:

  choose uniform grid with n intervals and mesh-width $h = (b-a)/n$
- Composite trapezoidal rule

$$CT := h \cdot \left[\frac{f(a)}{2} + \sum_{i=1}^{n-1} f(a+ih) + \frac{f(b)}{2} \right]$$

- Composite Simpson's rule

$$CS := \frac{h}{6} \left[f(a) + 4f\left(a + \frac{h}{2}\right) + 2f(a+h) + 4f\left(a + \frac{3h}{2}\right) \right.$$
$$\left. + \ldots + 4f\left(b - \frac{h}{2}\right) + f(b) \right]$$

Composite Quadrature Rules – Error

- To measure the error: sum up $n = (b-a)/h$ terms (one for each interval)
- Terms are in $\mathcal{O}(h^3)$ and $\mathcal{O}(h^5)$ resp.

$$
\begin{aligned}
|CT - F_1| &\leq \frac{M_2}{12}(b-a)\cdot h^2, \\
|CS - F_1| &\leq \frac{M_4}{2880}(b-a)\cdot h^4.
\end{aligned}
$$

- Accuracy increases with n
- Doubling the computational effort ($h \rightsquigarrow h/2$) reduces error bound to $1/4$ (CT) and $1/16$ (CS), if f is sufficiently smooth

Composite Quadrature Rules – Summary

Typical non-hierarchical methods

- Summands have (more or less) same weight
- To store: use array
- To implement: use for-loop
- To increase accuracy: discard old result, start all over once again

Composite Quadrature Rules – Summary

Typical non-hierarchical methods

- Summands have (more or less) same weight
- To store: use array
- To implement: use for-loop
- To increase accuracy: discard old result, start all over once again

What we desire:

- Allow “adaptive” choice of h: fine where required, coarse where possible
- Availability of an “error indicator”: where do we need fine resolution?

Archimedes' Hierarchical Approach

We now decompose the area F_1 in a **hierarchical manner**:

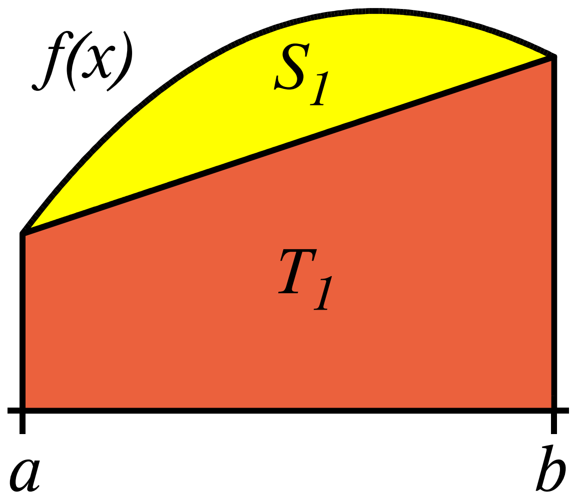
- Start with trapezoid as for trapezoidal rule:

$$T_1(f, a, b) = \frac{b-a}{2}(f(a) + f(b)).$$

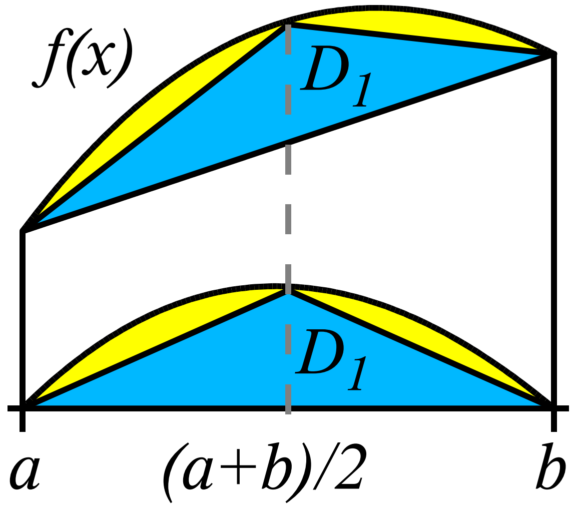
- Let remaining error term (area between trapezoid and curve) be S_1:

$$F_1(f, a, b) = T_1(f, a, b) + S_1(f, a, b).$$

- Hierarchical approach if current approximation too inaccurate:
  - Take trapezoid (intermediate solution)
  - Add approximation for S_1



Decomposition of Remainder S_1



- Decompose remainder S_1 into triangle D_1 with (projected) base $(b-a)$ and height

$$f\left(\frac{a+b}{2}\right) - \frac{f(a)+f(b)}{2}:$$

$$D_1(f,a,b) = \frac{b-a}{2}\left(f\left(\frac{a+b}{2}\right) - \frac{f(a)+f(b)}{2}\right)$$

- We obtain two remainders of similar type

$$S_1(f,a,b) = D_1(f,a,b) + S_1(f,a,\tfrac{a+b}{2}) + S_1(f,\tfrac{a+b}{2},b)$$

- Both are typically much smaller!

Recursive Computation of F_1

- Interprete formulas for F_1 (area below curve), T_1 (trapezoid) and S_1 (remainder) as function definitions

$\Rightarrow$ Obtain recursive method to compute F_1

Recursive Computation of F_1

- Interprete formulas for F_1 (area below curve), T_1 (trapezoid) and S_1 (remainder) as function definitions

$\Rightarrow$ Obtain recursive method to compute F_1

Stopping criterion

- Note: recursion does not terminate so far
- As we're only interested in approximation: implement termination criterion in function S_1, for example
  - Count recursion depth ($t = 0$ for whole interval $[a, b]$, $t = 1$ for the first two subintervals, ...)
  - Stop recursion for certain $t = l$
  - Then we exactly compute the composite trapezoidal quadrature for $n = 2^l$
  - Alternatively, we could have used $b - a \leq h$ for some $h = 2^{-l}$ as stopping criterion

Adaptive Stopping Criterion

- Intuitive assumption (look at drawings):
  triangle D_1 comprises most of S_1
- Later, we'll see that D_1 covers $3/4$ of the area of S_1 for sufficiently smooth functions and asymptotically for small h
- We can hope (but not be sure!):
  Error for the computation of S_1 is about $D_1/3$
  when stopping the recursion
- Hierarchical approach provides a stopping criterion for free

$\Rightarrow$ We can control the error of the quadrature!

- Even better:
  - Take height of triangle (*hierarchical surplus*) instead of area
  - Stop if smaller than some ε

  $\Rightarrow$ We can hope to bound global error (w.r.t. F_1) by $\varepsilon \cdot (b-a)$

Archimedes' Quadrature – Summary

Algorithm:

1. compute area as sum of trapezoid and remaining segment:

$$F_1(f,a,b) = T_1(f,a,b) + S_1(f,a,b) = \frac{b-a}{2}\big(f(a)+f(b)\big) + S_1(f,a,b)$$

2. compute area of triangular surplus:

$$D_1(f,a,b) = \tfrac{b-a}{2}\left(f\left(\tfrac{a+b}{2}\right) - \tfrac{f(a)+f(b)}{2}\right)$$

3. terminate, if surplus or $D_1(f,a,b)$ is smaller than ε
4. otherwise, recursively compute remaining segment:

$$S_1(f,a,b) = D_1(f,a,b) + S_1\left(f,a,\tfrac{a+b}{2}\right) + S_1\left(f,\tfrac{a+b}{2},b\right)$$

Some Remarks

- For polynomials f of degree 2 (i.e., parabolas), we can compute (exactly)

$$D_1 = \frac{3}{4} S_1$$

- When stopping the recursion, we can take $4/3 \cdot D_1$ rather than D_1
  $\Rightarrow$ We obtain the integrand exactly
- Without adaptivity: computes the composite Simpson's rule

Some Remarks

- For polynomials f of degree 2 (i.e., parabolas), we can compute (exactly)

$$D_1 = \frac{3}{4} S_1$$

- When stopping the recursion, we can take $4/3 \cdot D_1$ rather than D_1
  $\Rightarrow$ We obtain the integrand exactly
- Without adaptivity: computes the composite Simpson's rule

Improved Recursive Scheme:

- Currently: 3 evaluations of f to compute the hierarchical surplus
- When calling function S_1, we have already computed f at the interval boundaries

$\Rightarrow$ Extend recursive call $S(f, a, b)$ to $S(f, a, b, f(a), f(b))$ at no extra cost