

# Algorithms of Scientific Computing

## Hierarchical Methods and Sparse Grids

Tobias Neckel, Dirk Pflüger

Technische Universität München

Summer Term 2011



# Overview

# Topics

- Non-hierarchical and hierarchical quadrature and interpolation
- Curse of dimensionality
- Hierarchical basis and subspace decompositions
- High-dimensional function representations: sparse grids
- Hierarchical finite elements
- Multi-recursive and hierarchical algorithms on sparse grids
- Applications
- Multi-grid methods
- More hierarchical bases: wavelets, . . .

# Part I

## Archimedes' Quadrature, One-Dimensional

# Numerical Quadrature

## Why Quadrature?

- Integration integral part in many applications
  - Determine volumes (e.g. of beer/wine barrels)
  - Option pricing (expectation values)
  - Defuzzification for fuzzy controller
  - Optimization
  - Radiosity (accumulating light)
  - ...

# Numerical Quadrature

## Why Quadrature?

- Integration integral part in many applications
    - Determine volumes (e.g. of beer/wine barrels)
    - Option pricing (expectation values)
    - Defuzzification for fuzzy controller
    - Optimization
    - Radiosity (accumulating light)
    - ...
  - Often no analytical solution available
- ⇒ Approximate solution: numerical quadrature

# Numerical Quadrature

## Why Quadrature?

- Integration integral part in many applications
    - Determine volumes (e.g. of beer/wine barrels)
    - Option pricing (expectation values)
    - Defuzzification for fuzzy controller
    - Optimization
    - Radiosity (accumulating light)
    - ...
  - Often no analytical solution available
- ⇒ Approximate solution: numerical quadrature
- Core-problem: representation of functions in several variables
    - In higher-dimensional settings only stochastic or hierarchical methods available
    - Here: focus on hierarchical methods

# Quadrature One-Dimensional

## Approximations for the definite integral

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

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

- First example for a hierarchical method
- We first consider classical methods
- Then hierarchical approach
- Assumption in the following:  $f$  is sufficiently often continuously differentiable



# Trapezoidal Rule, Simpson Rule

## Classical methods for numerical quadrature: Newton-Cotes formulas

- $f(x_i)$  at equally spaced points  $x_i = ih + a$
- Integrate

$$\int_a^b f(x) \approx \sum w_i f(x_i)$$

# Trapezoidal Rule, Simpson Rule

## Trapezoidal rule

- Interpolate in interval boundaries with linear function

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

## Simpson rule

- Interpolate in 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

- It holds for the error term of the two methods

$$|T - F_1| \leq \frac{M_2}{12}(b - a)^3$$

$$|S - F_1| \leq \frac{M_4}{2880}(b - a)^5$$

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

$$M_2 := \sup_{x \in [a, b]} |f''(x)|,$$

$$M_4 := \sup_{x \in [a, b]} |f^{(4)}(x)|.$$

# Composite Quadrature Rules

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

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

## Composite Quadrature Rules – Error

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

$$|CT - F_1| \leq \frac{M_2}{12}(b - a) \cdot h^2,$$

$$|CS - F_1| \leq \frac{M_4}{2880}(b - a) \cdot h^4.$$

- 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

# Archimedes' Hierarchical Approach

We now decompose the area  $F_1$  in a hierarchical manner

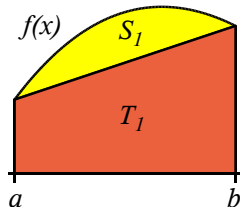
- Start with trapezoid as for trapezoidal rule:

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

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

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

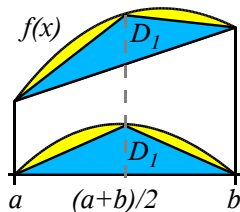
- Hierarchical approach if current approximation too inaccurate:
  - Take trapezoid (intermediate solution)
  - Add approximation for  $S_1$



## Decomposition of Remainder Term $S_1$

- Decompose remainder term  $S_1$  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 remainder terms of similar type

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

- Both are typically much smaller!



# Recursive Computation of $F_1$

- Interpret formulas for  $F_1$  (area below curve),  $T_1$  (trapezoid),  $S_1$  (remainder) as function definitions
- ⇒ Obtain recursive method to compute  $F_1$

# Recursive Computation of $F_1$

- Interpret formulas for  $F_1$  (area below curve),  $T_1$  (trapezoid),  $S_1$  (remainder) as function definitions
- ⇒ 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 it is  $3/4$  of the area 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
- ⇒ We can control the error of the quadrature!
- Even better:
    - Take height of triangle (*hierarchical surplus*) instead of area
    - Stop if smaller than some  $\epsilon$
- ⇒ We can even hope to bound global error (w.r.t.  $F_1$ ) by  $\epsilon(b - a)$

## Some Remarks

- For polynomials  $f$  of degree 2, it holds exactly

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

- When stopping the recursion, we can take  $4/3 \cdot D_1$  rather than  $D_1$
- ⇒ We obtain the integrand exactly
- In total, we just compute the composite Simpson's rule

## Some Remarks

- For polynomials  $f$  of degree 2, it holds exactly

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

- When stopping the recursion, we can take  $4/3 \cdot D_1$  rather than  $D_1$
- ⇒ We obtain the integrand exactly
- In total, we just compute the composite Simpson's rule
- 
- Currently, we have to evaluate  $f$  three times to compute the hierarchical surplus
  - When calling function  $S_1$ , we have already computed  $f$  at the interval boundaries
- ⇒ Extend  $S(f, a, b)$  to  $S(f, a, b, f(a), f(b))$  at no extra cost