Chapter 4
Numerical Differentiation and Integration

Weichung Wang
Institute of Applied Mathematical Sciences
National Taiwan University

Introduction to Computational Mathematics
Fall, 2016 (Version 2016.10.24)

Approximation of Derivatives

Approximation of Definition

- By definition,

$$f'(\bar{x}) = \lim_{h \to 0} \frac{1}{h}(f(\bar{x}+h) - f(\bar{x}))$$

- Two approximations
  - Forward finite difference

$$(\delta_+ f)(\bar{x}) = \frac{f(\bar{x}+h) - f(\bar{x})}{h}$$

  - Backward finite difference

$$(\delta_- f)(\bar{x}) = \frac{f(\bar{x}) - f(\bar{x}-h)}{h}$$

Taylor's Viewpoint w/ Error Upper bound

- Forward: O(h)

$$f(\bar{x}+h) = f(\bar{x}) + hf'(\bar{x}) + \frac{h^2}{2}f''(\xi)$$

$$(\delta_+ f)(\bar{x}) = f'(\bar{x}) + \frac{h}{2}f''(\xi),$$

- Backward: O(h)

$$f(\bar{x}-h) = f(\bar{x}) - hf'(\bar{x}) + \frac{h^2}{2}f''(\eta)$$

- Centered finite difference: O(h^2)

$$(\delta f)(\bar{x}) = \frac{f(\bar{x}+h) - f(\bar{x}-h)}{2h}$$

$$f'(\bar{x}) - (\delta f)(\bar{x}) = -\frac{h^2}{12}[f'''(\xi_-) + f'''(\xi_+)]$$

Geometric Viewpoint

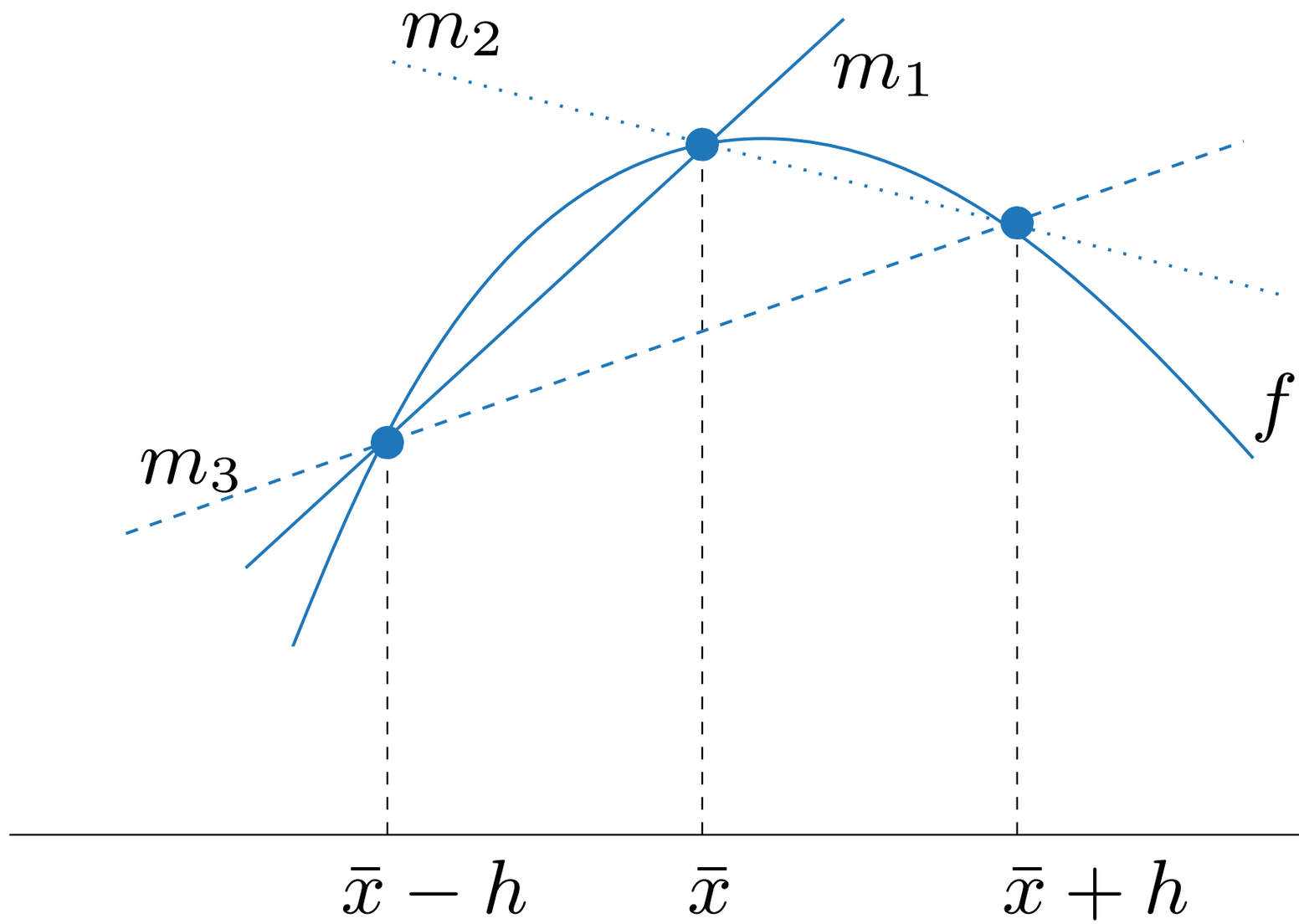


Fig. 4.2. Finite difference approximation of $f'(\bar{x})$: backward (*solid line*), forward (*dotted line*) and centered (*dashed line*). The values $m_1 = (\delta_- f)(\bar{x})$, $m_2 = (\delta_+ f)(\bar{x})$ and $m_3 = (\delta f)(\bar{x})$ denote the slopes of the three straight lines

Another Second Order Approximations

- Consider n+1 equispaced points

$$x_i = x_0 + ih,\ i = 0, \ldots, n, \text{ with } h > 0$$

- How about the 2nd order centered finite difference for the two endpoints?

- Homework: Show that the following finite difference approximations are 2nd order.

$$\frac{1}{2h}\left[-3f(x_0) + 4f(x_1) - f(x_2)\right] \qquad \text{at } x_0,$$

$$\frac{1}{2h}\left[3f(x_n) - 4f(x_{n-1}) + f(x_{n-2})\right] \text{ at } x_n,$$

In Summary

- Taylor is our good friend. Apply his theorem cleverly.

Numerical Integration

Numerical Quadrature on 1D

- Main Idea:
  The integral gives the area under the curve of the function

- The Riemann sum

$$\lim_{n\to\infty} \sum_{i=0}^{n} f(t_i)(x_{i+1} - x_i) = J, \text{ where } x_i \leq t_i \leq x_{i+1}$$

- A quadrature formula approximates the Riemann integral as a discrete sum over a set of n nodes

$$J \approx J_n = \sum_{i=1}^{n} \alpha_i f(x_i)$$

Main Idea

- Composite quadrature $$I(f) = \int_a^b f(x)dx = \sum_{k=1}^{M} \int_{I_k} f(x)dx$$
  - In each sub-interval, approximate the function by a (piecewise) polynomial
  - Only function values on the abscissae are needed!
  - Zero order (Midpoint formula)

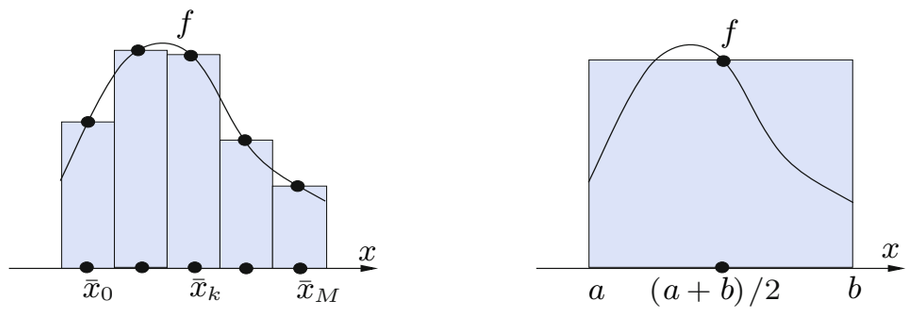


Fig. 4.3. The composite midpoint formula (*left*); the midpoint formula (*right*)

$$I_{mp}^{c}(f) = H\sum_{k=1}^{M} f(\bar{x}_k)$$

- First order (Trapezoidal formula)

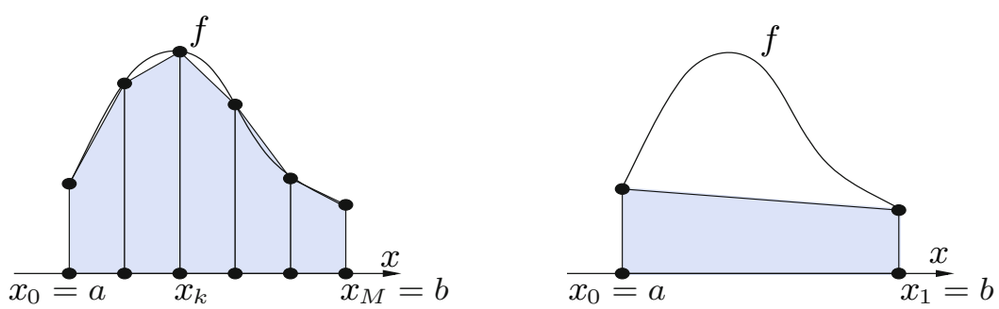
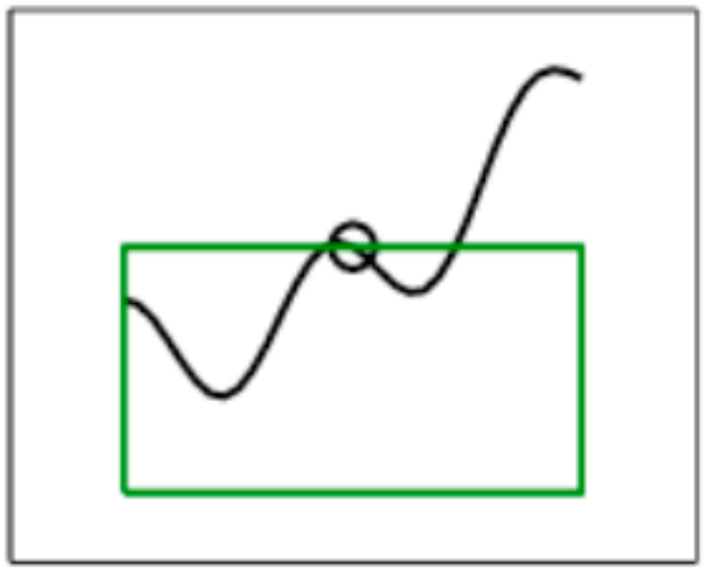


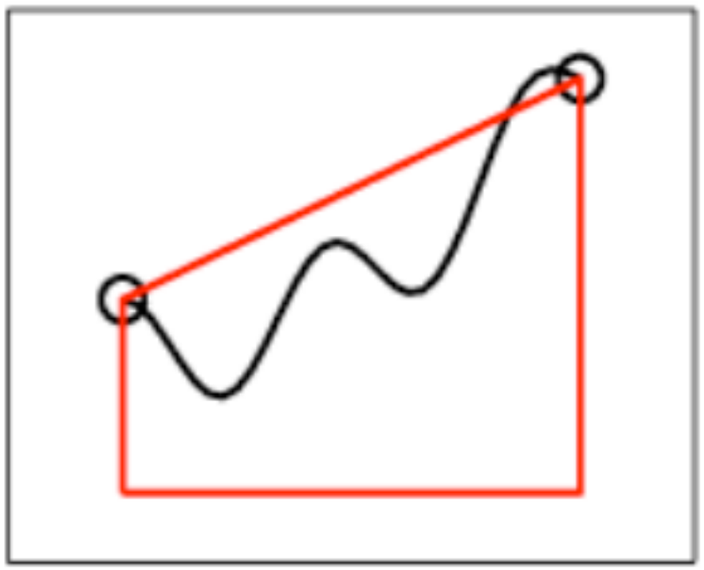
Fig. 4.4. Composite trapezoidal formula (*left*); trapezoidal formula (*right*)

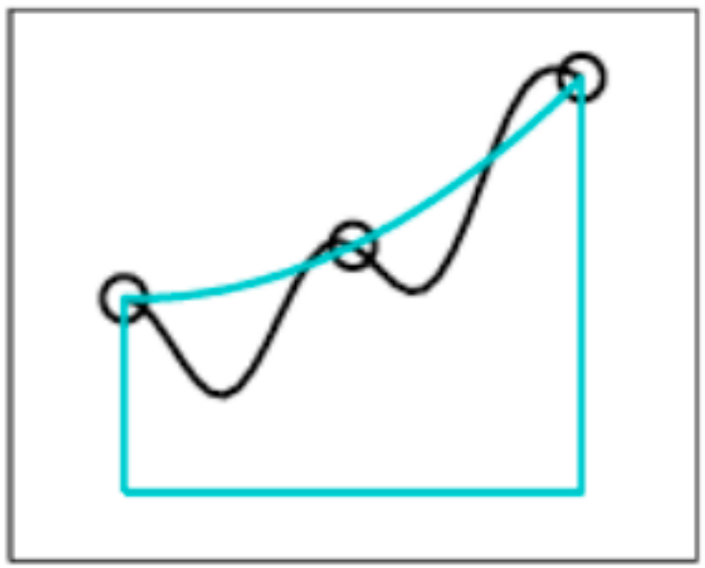
$$I_t^c(f) = \frac{H}{2}\sum_{k=1}^{M}[f(x_{k-1}) + f(x_k)] = \frac{H}{2}[f(a) + f(b)] + H\sum_{k=1}^{M-1} f(x_k)$$

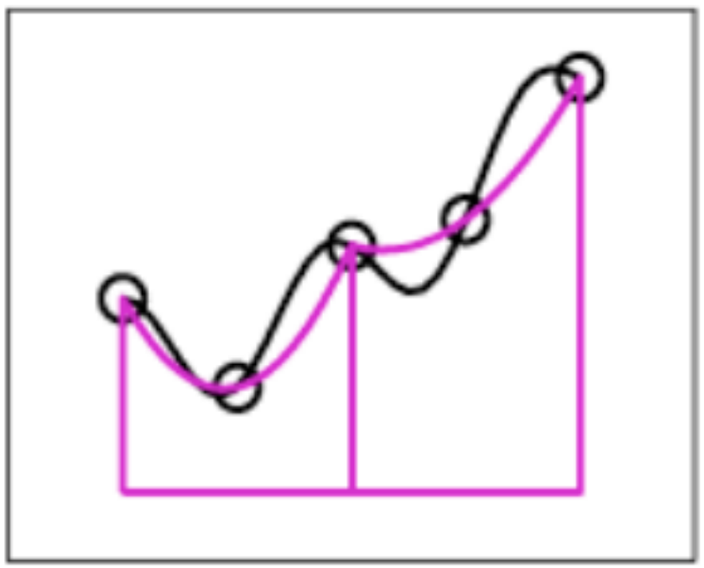
- Second order (Simpson formula)

Interpolatory Quadrature









Midpoint Formula: Error

- Consider

$$I_{mp}(f) = (b-a)f[(a+b)/2]$$

$$I(f) - I_{mp}(f) = \int_a^b [f(x) - f(\bar{x})]dx$$

$$= \int_a^b f'(\bar{x})(x-\bar{x})dx + \frac{1}{2}\int_a^b f''(\eta(x))(x-\bar{x})^2 dx$$

Taylor

=0

$$= \frac{1}{2}f''(\xi)\int_a^b (x-\bar{x})^2 dx$$

Mean Value Theorem for Integers

$$= \frac{(b-a)^3}{24}f''(\xi).$$

- For composite midpoint quadrature, we have

$$I(f) - I_{mp}^c(f) = \frac{b-a}{24}H^2 f''(\xi)$$

- Degree of exactness = 1 (Why?)

Midpoint Formula: Matlab

Program 4.1. midpointc: composite midpoint quadrature formula

```
function Imp=midpointc(a,b,M,f,varargin)
%MIDPOINTC Composite midpoint numerical integration.
% IMP = MIDPOINTC(A,B,M,FUN) computes an approximation
% of the integral of the function FUN via the midpoint
% method (with M equal subintervals).  FUN accepts a
% real vector input x and returns a real vector value.
% FUN can be either an inline function, an anonymous
% function, or it can be defined by an external m-file.
% IMP=MIDPOINT(A,B,M,FUN,P1,P2,...) calls the function
% FUN passing the optional parameters P1,P2,... as
% FUN(X,P1,P2,...).
H=(b-a)/M;
x = linspace(a+H/2,b-H/2,M);
fmp=feval(f,x,varargin{:}).*ones(1,M);
Imp=H*sum(fmp);
```

Trapezoidal Formula

- Quadrature error

$$I(f) - I_t(f) = -\frac{(b-a)^3}{12} f''(\xi)$$

- Composite quadrature error (global error)

$$I(f) - I_t^c(f) = -\frac{b-a}{12} H^2 f''(\xi)$$

- Degree of exactness = 1 (Same as MP formula)
- Matlab
  - `trapz`
  - `cumtrapz`

Simpson Formula

- Let $x_{k-1},\ \bar{x}_k = (x_{k-1} + x_k)/2$ and x_k,

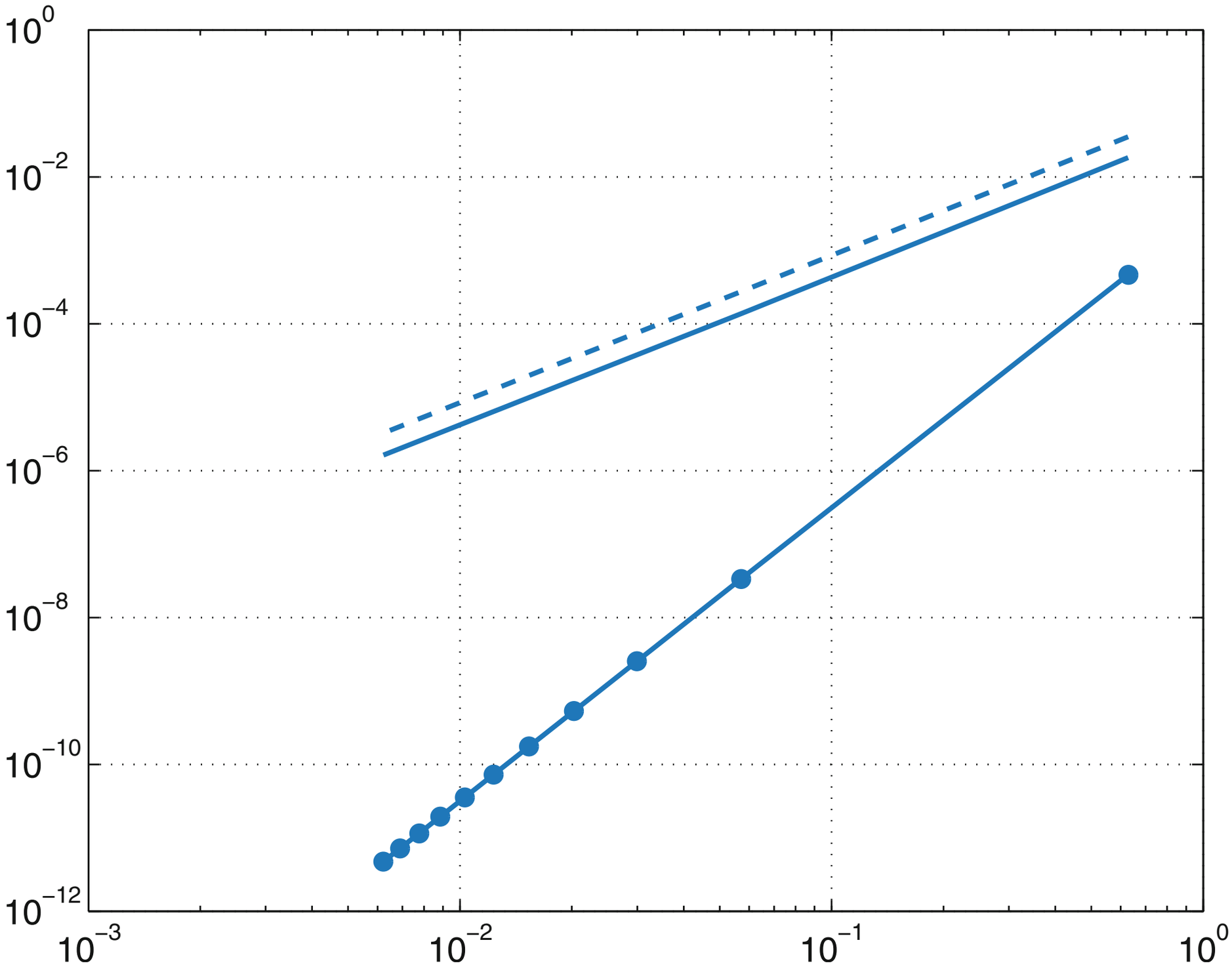
$$\Pi_2 f(x) = \frac{2(x-\bar{x}_k)(x-x_k)}{H^2} f(x_{k-1}) + \frac{4(x_{k-1}-x)(x-x_k)}{H^2} f(\bar{x}_k) + \frac{2(x-\bar{x}_k)(x-x_{k-1})}{H^2} f(x_k).$$

- Composite Simpson quadrature formula

$$I_s^c(f) = \frac{H}{6} \sum_{k=1}^{M} [f(x_{k-1}) + 4f(\bar{x}_k) + f(x_k)]$$

$$I(f) - I_s^c(f) = -\frac{b-a}{180} \frac{H^4}{16} f^{(4)}(\xi),$$

A Comparison



A Comparison

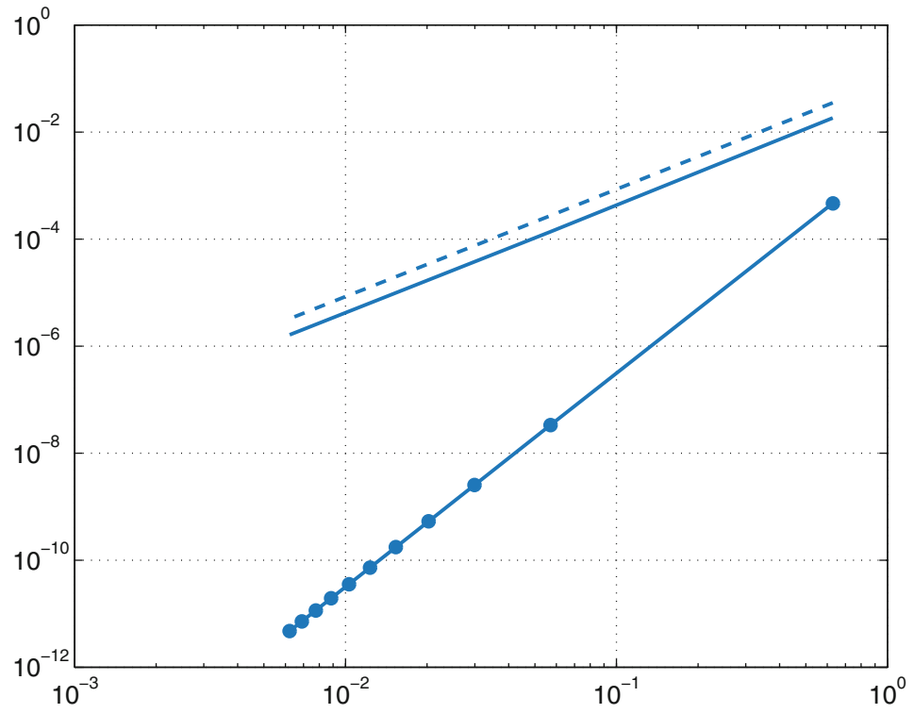


Fig. 4.5. Logarithmic representation of the errors versus H for Simpson (*solid line with circles*), midpoint (*solid line*) and trapezoidal (*dashed line*) composite quadrature formulae

Example 4.3 We want to compare the approximations of the integral $I(f) = \int_0^{2\pi} xe^{-x}\cos(2x)dx = -(10\pi - 3 + 3e^{2\pi})/(25e^{2\pi}) \simeq -0.122122604618968$ obtained by using the composite midpoint, trapezoidal and Simpson formulae. In Figure 4.5 we plot on the logarithmic scale the errors versus H. As pointed out in Section 1.6, in this type of plot the greater the slope of the curve, the higher the order of convergence of the corresponding formula. As expected from the theoretical results, the midpoint and trapezoidal formulae are second-order accurate, whereas the Simpson formula is fourth-order accurate. ■

Guassian Quadratures

Motivation

- To reach higher accuracy, instead of using higher-degree polynomial interpolants (recall Runge's phenomenon), let's try using n **non-equispaced nodes**:

$$J \approx J_n = \sum_{i=1}^{n} w_i f(x_i)$$

- Question: For some choice of the weights and nodes, is it possible to compute

$$\int_a^b p_m(x)dx$$

exactly for any polynomial $p_m(x)$ of degree at most m?

- This **degree of exactness** m would guarantee that the accuracy will be $\varepsilon \sim f^{(m+1)}(\xi)$ since all the lower-order derivatives are done exactly.

Idea: Polynomial Interplant Integration

- Let's first consider polynomials of degree at most n

$$\int_a^b p_n(x)dx = ?$$

- Any polynomial $p_n(x)$ of degree at most n can be expressed in the Lagrange basis:

$$p_n(x) = \sum_{i=0}^{n} p_n(x_i)\varphi_i(x)$$

- Integrating the polynomial interpolant we get:

$$\int_a^b p_n(x)dx = \sum_{i=0}^{n} p_n(x_i)\left[\int_a^b \varphi_i(x)dx\right] = \sum_{i=0}^{n} w_i p_n(x_i),$$

where the **Gauss weights w** are given by

$$w_i = \int_a^b \varphi_i(x)dx.$$

Gauss Integration

- It can be shown that if we choose the **nodes to be zeros** of the Legendre polynomial of degree $n+1$, then even for $m = 2n-1$:

$$\int_a^b p_m(x)dx = \sum_{i=0}^{n} w_i p_m(x_i).$$

- This gives the **Gauss quadrature** based on the **Gauss nodes and weights**, usually pre-tabulated for the standard interval $[-1, 1]$:

$$\int_a^b f(x)dx \approx \frac{b-a}{2} \sum_{i=0}^{m} w_i f(x_i).$$

- Gauss quadrature is **exact** for polynomials of degree up to $2n-1$, and $\varepsilon \sim f^{(2n)}(\xi)$.

Gauss Integration

- See the slides for Burden and Faires
- Matlab (adaptive) Gauss-Legendre-Lobatto quadrature
  - `quadl(fun, a, b, tol)`

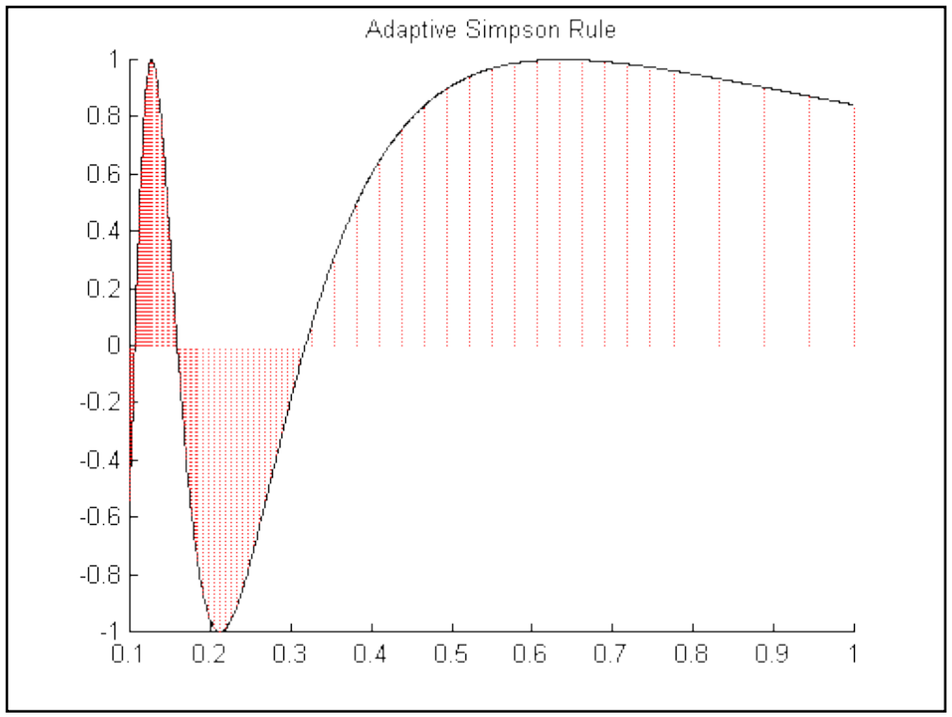
A Short Summary

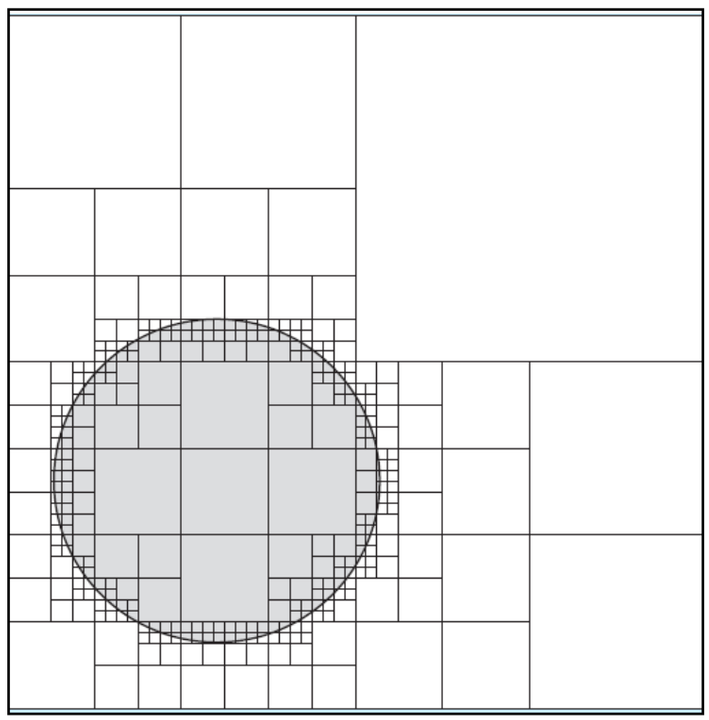
1. A quadrature formula is a formula to approximate the integral of continuous functions on an interval $[a, b]$;
2. it is generally expressed as a linear combination of the values of the function at specific points (called *nodes*) with coefficients which are called *weights*;
3. the *degree of exactness* of a quadrature formula is the highest degree of the polynomials which are integrated exactly by the formula. It is one for the midpoint and trapezoidal rules, three for the Simpson rule, $2n + 1$ for the Gauss-Legendre formula using $n + 1$ quadrature nodes, and $2n - 1$ for the Gauss-Legendre-Lobatto formula using $n + 1$ nodes;
4. the *order of accuracy* of a composite quadrature formula is its order with respect to the size H of the subintervals. The order of accuracy is two for composite midpoint and trapezoidal formulae, four for composite Simpson formula.

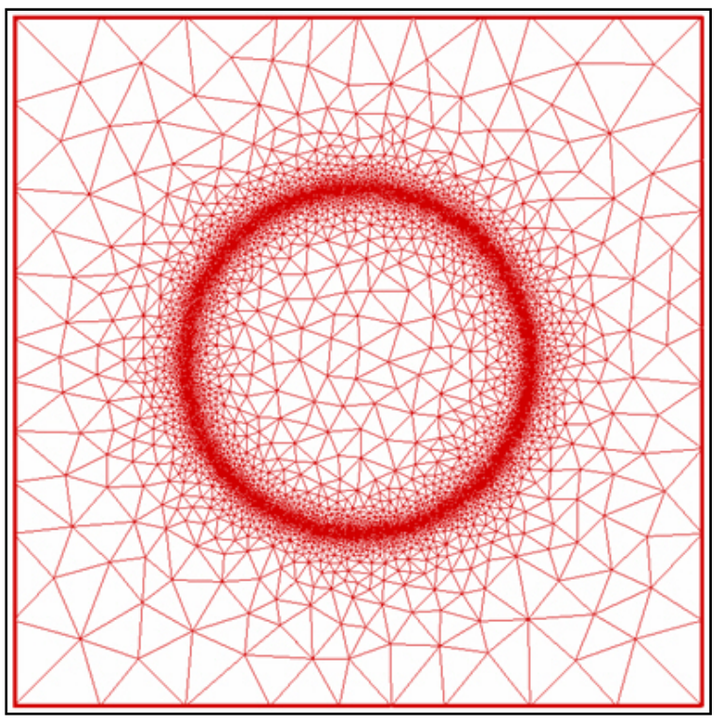
Adaptive (Automatic) Integration

Main Idea

- To approximate the integration within a fixed tolerance ε by a nonuniform distribution of the integration step-sizes



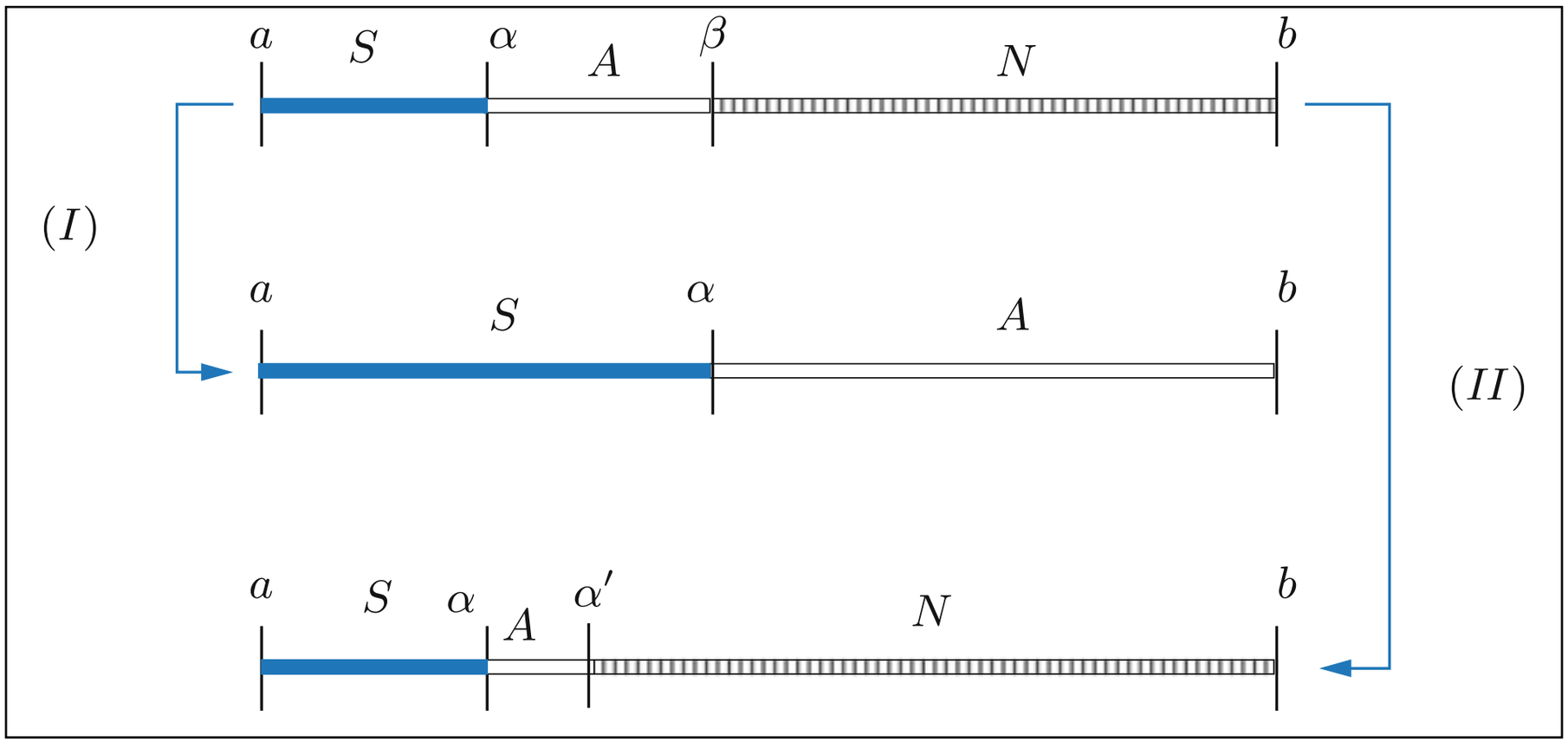




Ingredient 1: Interval Controller

- A: active interval (being computed)
- S: satisfactory interval
- N: not yet examined

- (I): A is satisfactory
- (II) A is not satisfactory



Ingredient 2: Error Estimator for Simpson

- By $[\alpha, \beta] \subset [a, b]$ and results on p.15,

Unknown!

$$E_s(f;\alpha,\beta) = \int_\alpha^\beta f(x)dx - I_s(f) = -\frac{(\beta-\alpha)^5}{2880} f^{(4)}(\xi) < \varepsilon(\beta-\alpha)/(b-a)$$

- Composite Simpson with $H = (\beta-\alpha)/2$ leads to

$$\int_\alpha^\beta f(x)dx - I_s^c(f) = -\frac{(\beta-\alpha)^5}{46080} f^{(4)}(\eta)$$

- Subtracting the two errors, we get

$$\Delta I = I_s^c(f) - I_s(f) = -\frac{(\beta-\alpha)^5}{2880} f^{(4)}(\xi) + \frac{(\beta-\alpha)^5}{46080} f^{(4)}(\eta).$$

- Assume $f^{(4)}(\xi) \simeq f^{(4)}(\eta)$ and solve for $f^{(4)}(\eta)$. We get

$$\int_\alpha^\beta f(x)dx - I_s^c(f) \simeq \frac{1}{15}\Delta I.$$

Lab: Program 4.3

Program 4.3. simpadpt: adaptive Simpson formula

```
function [JSf,nodes]=simpadpt(f,a,b,tol,hmin,varargin)
%SIMPADPT Adaptive Simpson quadrature formula
% JSF = SIMPADPT(FUN,A,B,TOL,HMIN) tries to approximate
% the integral of function FUN from A to B within
% error TOL using recursive adaptive Simpson
% quadrature with H>=HMIN. The function FUN should
% accept a vector argument x and return a vector.
% FUN can be either an inline function, an anonymous
% function, or it can be defined by an external m-file.
% JSF = SIMPADPT(FUN,A,B,TOL,HMIN,P1,P2,...) calls the
% function FUN passing the optional parameters
% P1,P2,... as FUN(X,P1,P2,...).
% [JSF,NODES] = SIMPADPT(...) returns the distribution
% of nodes used in the quadrature process.
A=[a,b]; N=[]; S=[]; JSf = 0; ba = 2*(b - a); nodes=[];
while ~isempty(A),
  [deltaI,ISc]=caldeltai(A,f,varargin{:});
  if abs(deltaI) < 15*tol*(A(2)-A(1))/ba;
     JSf = JSf + ISc;     S = union(S,A);
     nodes = [nodes, A(1) (A(1)+A(2))*0.5 A(2)];
     S = [S(1), S(end)]; A = N; N = [];
  elseif A(2)-A(1) < hmin
     JSf=JSf+ISc;         S = union(S,A);
     S = [S(1), S(end)]; A=N; N=[];
     warning('Too small integration-step');
  else
     Am = (A(1)+A(2))*0.5;
     A = [A(1) Am];    N = [Am, b];
  end
end
nodes=unique(nodes);
return

function [deltaI,ISc]=caldeltai(A,f,varargin)
L=A(2)-A(1);
t=[0; 0.25; 0.5; 0.75; 1];
x=L*t+A(1); L=L/6;
w=[1; 4; 1]; wp=[1;4;2;4;1];
fx=feval(f,x,varargin{:}).*ones(5,1);
IS=L*sum(fx([1 3 5]).*w);
ISc=0.5*L*sum(fx.*wp);
deltaI=IS-ISc;
return
```

High Dimensional Integration

NCTS Short Course on

High-Dimensional Integration: the Quasi-Monte Carlo Way

High dimensional problems are coming to play an ever more important role in applications. They pose immense challenges for practical computation, because of a nearly inevitable tendency for the cost of computation to increase exponentially with dimension. Effective and efficient methods that do not suffer from this "curse of dimensionality" are in great demand. Quasi-Monte Carlo (QMC) methods lift this curse and we will show you how.

Part I Introduction to Quasi-Monte Carlo Methods

covers basic theory and practical usage of QMC methods, with a demo on the software packages.

Room 202, NCTS (Astro-Math Building, National Taiwan University)
November 10 (Thursday) 15:30-18:30

Part II Applications of Quasi-Monte Carlo Methods

illustrates how to apply QMC methods to an option pricing problem from mathematical finance, a maximum likelihood parameter estimation problem from statistics, and a porous media flow problem involving PDEs with a random coefficient from computational physics and uncertainty quantification.

Room 202, NCTS (Astro-Math Building, National Taiwan University)
November 11 (Friday) 9:00-12:00

Speakers:

Frances Kuo
University of New South Wales, Australia

Dirk Nuyens
KU Leuven, Belgium

Peter Kritzer
Johann Radon Institute, Austria

goo.gl/cKexox

Organizers: Frances Kuo, University of New South Wales
Weichung Wang, National Taiwan University

NCTS
National Center for Theoretical Sciences

NCTS Short Course on

High-Dimensional Integration: the Quasi-Monte Carlo Way

High dimensional problems are coming to play an ever more important role in applications. They pose immense challenges for practical computation, because of a nearly inevitable tendency for the cost of computation to increase exponentially with dimension. Effective and efficient methods that do not suffer from this "curse of dimensionality" are in great demand. Quasi-Monte Carlo (QMC) methods lift this curse and we will show you how.

Part I Introduction to Quasi-Monte Carlo Methods

covers basic theory and practical usage of QMC methods, with a demo on the software packages.

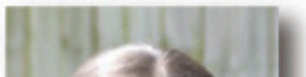
Room 202, NCTS (Astro-Math Building, National Taiwan University)
November 10 (Thursday) 15:30-18:30

Part II Applications of Quasi-Monte Carlo Methods

illustrates how to apply QMC methods to an option pricing problem from mathematical finance, a maximum likelihood parameter estimation problem from statistics, and a porous media flow problem involving PDEs with a random coefficient from computational physics and uncertainty quantification.

Room 202, NCTS (Astro-Math Building, National Taiwan University)
November 11 (Friday) 9:00-12:00

Speakers:



一個高維度積分例子

- PM 2.5 分佈
  - 統計，機率，高維度積分，數值方法，高效能平行計算，幾何
  - 空拍機，軟體，高速電腦，GPU
  - 如何讓我們的生活更好？

思維方式

- 科技在進步，時代在改變
- 能用的工具，關心的問題，可能的解決方案，應具備「時代意義」
- 跨領域
- Open-minded