993SM - Laboratory of Computational Physics
VIII week
November 10, 2025

Maria Peressi
Università degli Studi di Trieste - Dipartimento di Fisica
Sede di Miramare (Strada Costiera 11, Trieste)
e-mail: peressi@units.it

Numerical integration - I

M. Peressi - UniTS - Laurea Magistrale in Physics
Laboratory of Computational Physics - Unit VII

- deterministic methods in 1D
  equispaced points (trapezoidal, Simpson...), others...
- Monte Carlo methods
  (acceptance-rejection, sample mean, importance sampling...)

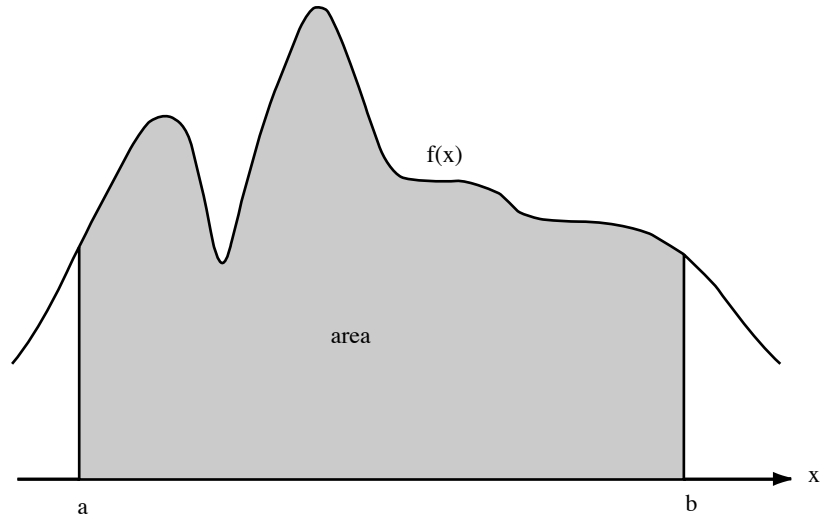
  Error handling:
  sample mean
  block average
  reduction of the variance

Deterministic methods

Deterministic methods

Start from the geometrical interpretation of a definite integral:

$$F = \int_a^b f(x)dx$$



Divide the integration interval into "small" intervals:

$$\Delta x = \frac{b-a}{n},$$

$$x_n = x_0 + n\,\Delta x.$$

(Note: n intervals $\Leftrightarrow$ $n+1$ points)

Deterministic methods:

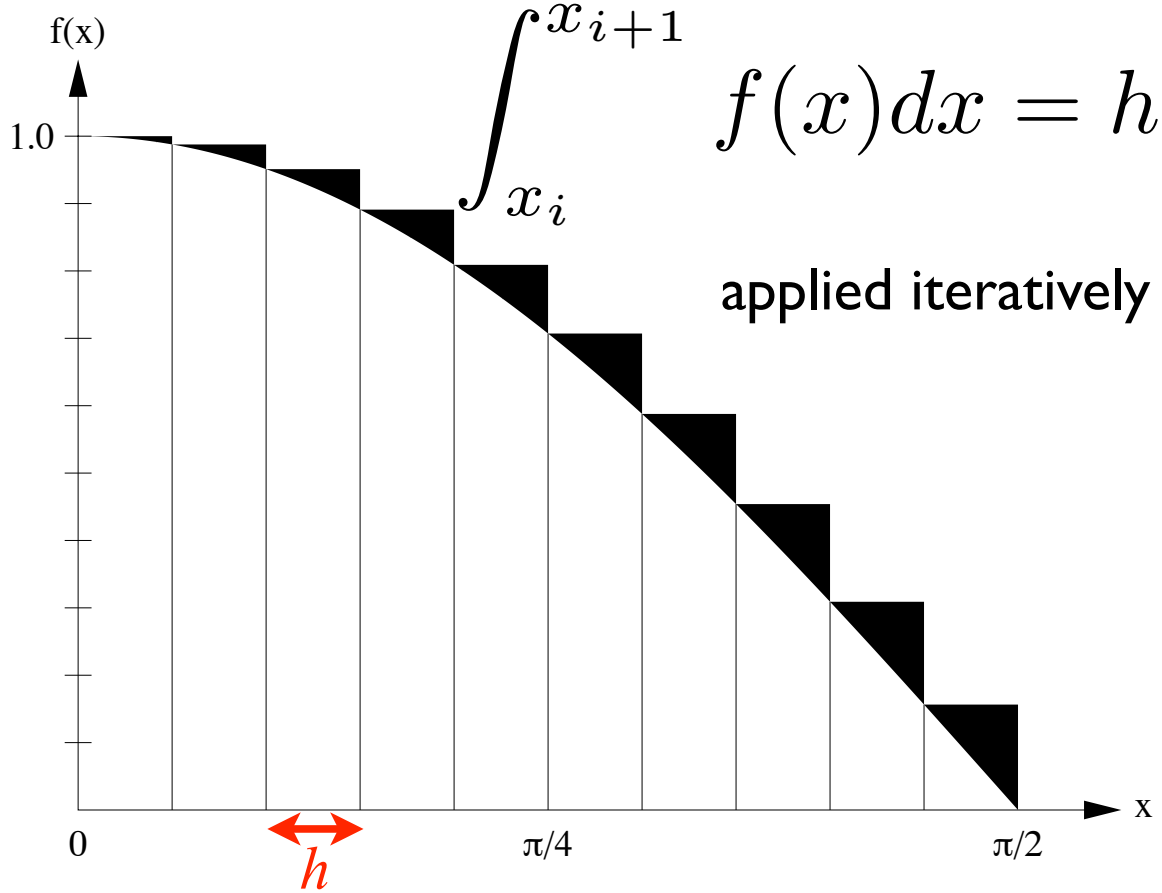
rectangular rule

In one interval:

$$\int_{x_i}^{x_{i+1}} f(x)dx = hf_i$$

with error:

$$\mathcal{O}(h^2 f'), \propto 1/n^2$$



applied iteratively over consecutive intervals:

$$F_n = \sum_{i=0}^{n-1} f(x_i)\Delta x.$$

with a total error:

$$\mathcal{O}(hf'), \propto 1/n$$

: The rectangular approximation for $f(x) = \cos x$ for $0 \leq x \leq \pi/2$.

Deterministic methods:

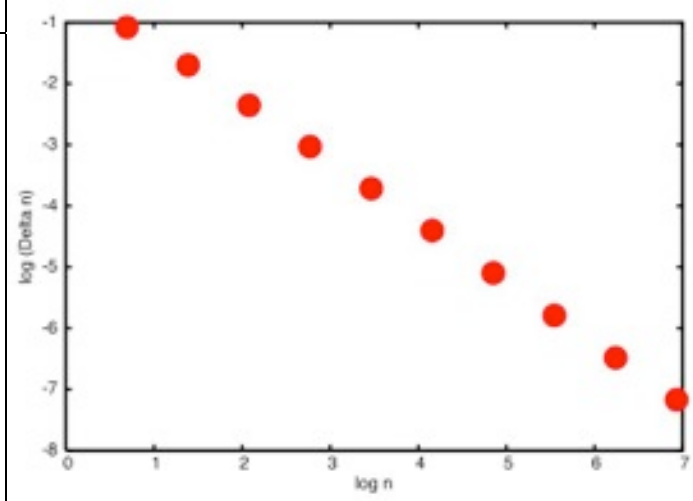
rectangular rule - error

$$I = \int_0^{\pi/2} \cos(x)dx = 1$$

$$F_n = \frac{\pi}{2n} \sum_0^{n-1} \cos x_i; \quad x_i = i\frac{\pi}{2n}$$

$$\Delta_n = F_n - I$$

n	F_n	Δ_n
2	1.34076	0.34076
4	1.18347	0.18347
8	1.09496	0.09496
16	1.04828	0.04828
32	1.02434	0.02434
64	1.01222	0.01222
128	1.00612	0.00612
256	1.00306	0.00306
512	1.00153	0.00153
1024	1.00077	0.00077



Rectangular approximations of the integral of $\cos x$ from $x = 0$ to $x = \pi/2$ as a function of n, the number of intervals. The error Δ_n is the difference between the rectangular approximation and the exact result of unity. Note that the error Δ_n decreases approximately as n^{-1}, that is, if n is increased by a factor of 2, Δ_n decreases by a factor 2.

Deterministic methods: generalities

- sum values of $f(x_i)$ with $x_i \in [a, b]$
- we want to have $F = \int_a^b f(x)dx$ as accurate as possible but with the minimum number of calculations of $f(x_i)$

OK simple algorithms, but if the number of calculations is too high, improve the algorithm!

Deterministic methods:

trapezoidal rule

In one interval:

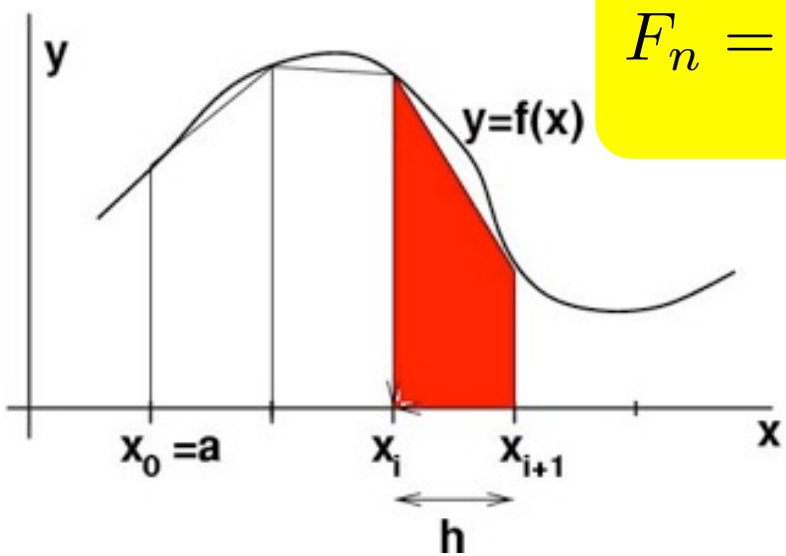
$$\int_{x_i}^{x_{i+1}} f(x)dx = h\left[\frac{1}{2}f_i + \frac{1}{2}f_{i+1}\right]$$

with error:

$$\mathcal{O}(h^3 f'''), \propto 1/n^3$$

Applied iteratively over consecutive intervals:

$$F_n = \left[\frac{1}{2}f(x_0) + \sum_{i=1}^{n-1} f(x_i) + \frac{1}{2}f(x_n)\right]\Delta x.$$

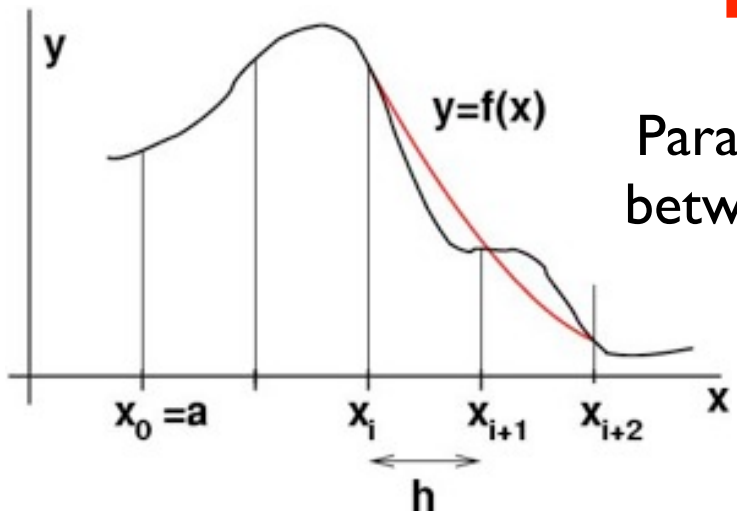


with a total error:

$$\mathcal{O}(h^2 f'''), \propto 1/n^2$$

Deterministic methods:

Simpson's rule



Parabolic interpolation procedure between triplets of adjacent points

In one interval:

$$\int_{x_i}^{x_{i+2}} f(x)dx = h\left[\frac{1}{3}f_i + \frac{4}{3}f_{i+1} + \frac{1}{3}f_{i+2}\right] + \mathcal{O}(h^5 f^{IV}) \; (error \; \propto 1/n^5)$$

Iteratively applied to the whole interval of integration (odd number of points!):

$$\int_{x_0}^{x_n} f(x)dx = h\left[\frac{1}{3}f_0 + \frac{4}{3}f_1 + \frac{2}{3}f_2 + \frac{4}{3}f_3 + \ldots + \frac{2}{3}f_{n-2} + \frac{4}{3}f_{n-1} + \frac{1}{3}f_n\right] + \mathcal{O}(h^4 f^{IV}) \; (error \; \propto 1/n^4)$$

Errors in deterministic methods

Error estimate for numerical integration with deterministic methods

$$\int f(x)dx = F_n + error$$

How to evaluate the error? Consider the Taylor expansion of the integrand function and then integrate:

$$f(x) = f(x_i) + f'(x_i)(x - x_i) + \frac{1}{2}f''(x_i)(x - x_i)^2 + \dots , \quad (*)$$

$$\int_{x_i}^{x_{i+1}} f(x)\,dx = f(x_i)\Delta x + \frac{1}{2}f'(x_i)(\Delta x)^2 + \frac{1}{6}f''(x_i)(\Delta x)^3 + \dots (**)$$

$$\Delta x \equiv x_{i+1} - x_i$$

Error estimate for numerical integration: Rectangular approximation

$$\int_{x_i}^{x_{i+1}} f(x)dx \approx f(x_i)\Delta x$$

Compare with (**):

$$\int_{x_i}^{x_{i+1}} f(x)\,dx = f(x_i)\Delta x + \frac{1}{2}f'(x_i)(\Delta x)^2 + \frac{1}{6}f''(x_i)(\Delta x)^3 + \dots$$

error

(leading order in Δx)

For n intervals $(\Delta x = (b-a)/n)$: error is $n(\Delta x)^2 \sim 1/n$

Error estimate for numerical integration: Trapezoidal approximation

$$\int_{x_i}^{x_{i+1}} f(x)dx \approx \frac{1}{2}\left[f(x_{i+1}) + f(x_i)\right]\Delta x$$

$$f(x_{i+1}) \approx f(x_i) + f'(x_i)\Delta x + \frac{1}{2}f''(x_i)\Delta x^2 + \ldots$$

$$\approx \frac{1}{2}\left[2f(x_i) + f'(x_i)\Delta x + \frac{1}{2}f''(x_i)\Delta x^2 + \ldots\right]\Delta x$$

Compare with (**):

$$\int_{x_i}^{x_{i+1}} f(x)\,dx = f(x_i)\Delta x + \frac{1}{2}f'(x_i)(\Delta x)^2 + \frac{1}{6}f''(x_i)(\Delta x)^3 + \ldots$$

error

(leading order in Δx)

For n intervals: error is $n(\Delta x)^3 \sim 1/n^2$

Error estimate for numerical integration: Simpson approximation

$$\int_{x_i}^{x_{i+2}} f(x)dx \approx \left[\frac{1}{3}f(x_i) + \frac{4}{3}f(x_{i+1}) + \frac{1}{3}f(x_{i+2})\right]\Delta x$$

....

.... (homework!)

$$f(x_{i+1}) \approx f(x_i) + f'(x_i)\Delta x + \frac{1}{2}f''(x_i)\Delta x + \ldots$$

Compare with (**):

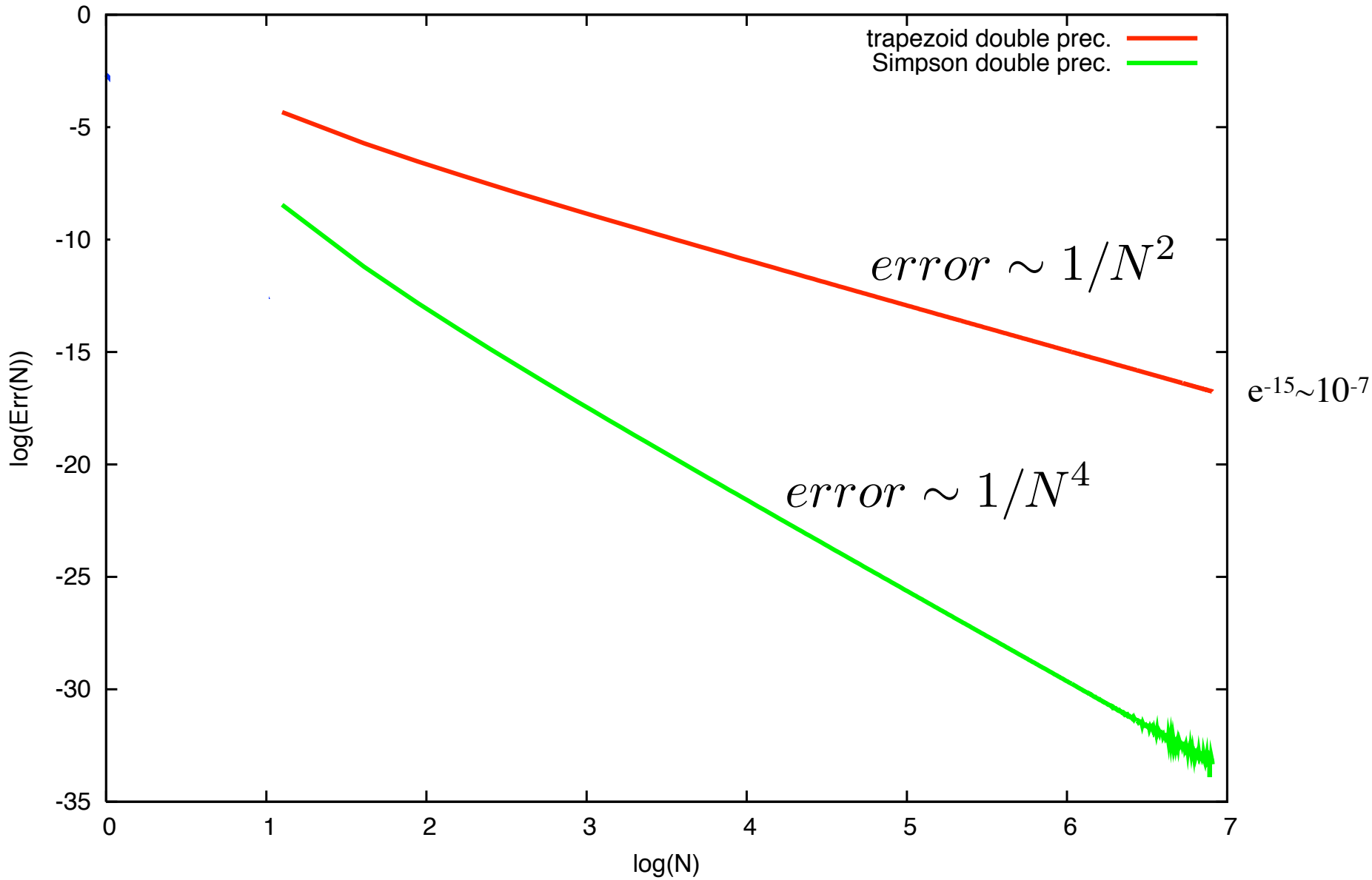
$$\int_{x_i}^{x_{i+2}} f(x)dx = f(x_i)\Delta x + \frac{1}{2!}f'(x_i)(\Delta x)^2 + \frac{1}{3!}f''(x_i)(\Delta x)^3 + \frac{1}{4!}f'''(x_i)(\Delta x)^4 + \frac{1}{5!}f''''(x_i)(\Delta x)^5 + \ldots$$

error

(leading order in Δx)

For n intervals: error is $n(\Delta x)^5 \sim 1/n^4$

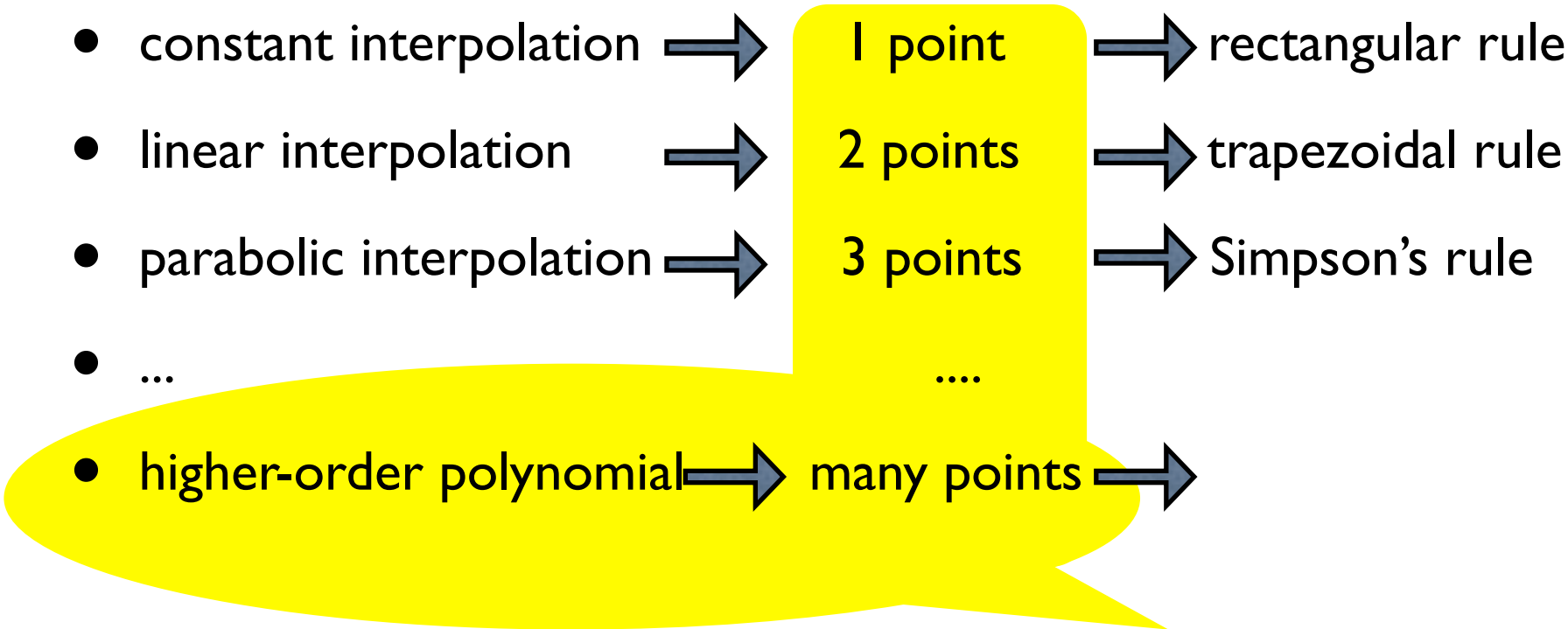
Numerical integration - deterministic methods: comparison of errors in 1D



$e^{-15} \sim 10^{-7}$

Deterministic methods - I

We use a piecewise polynomial interpolation:

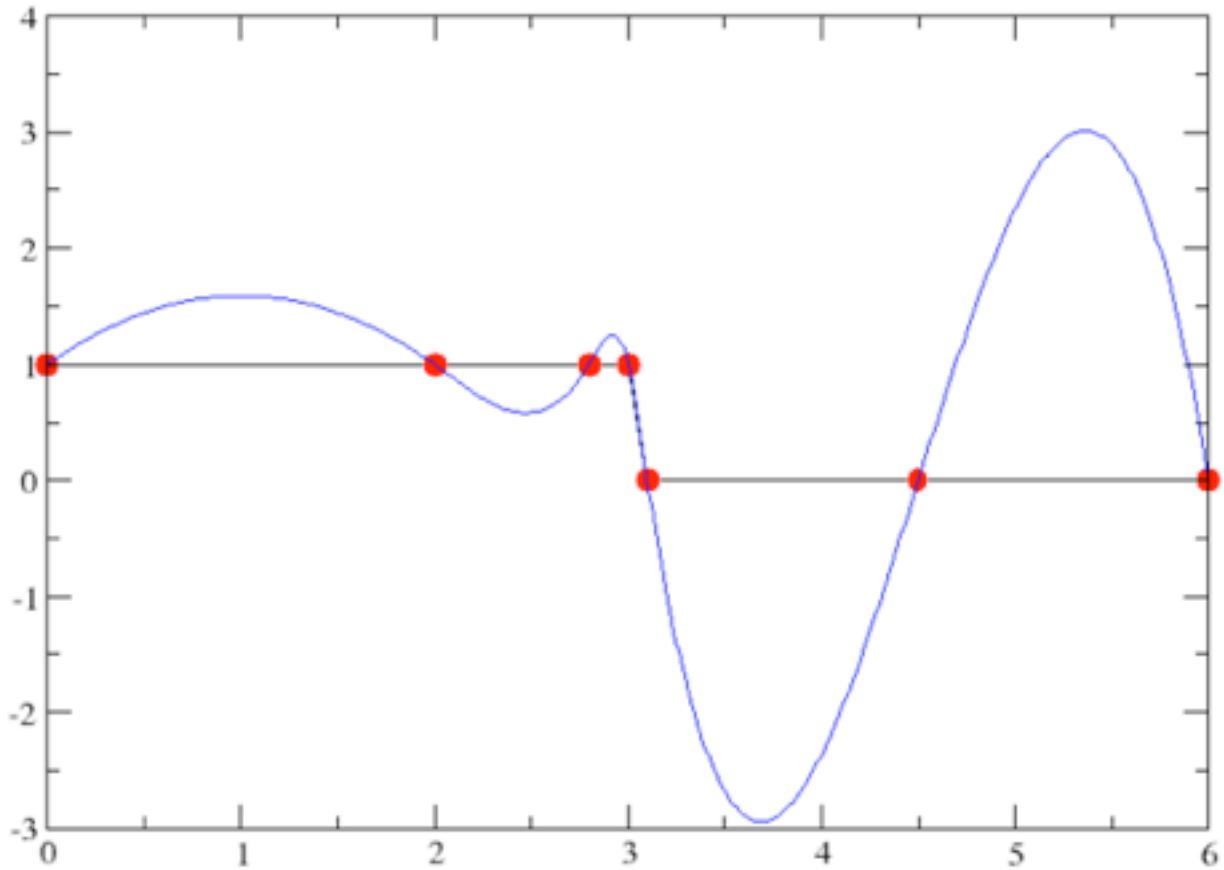


NOT CONVENIENT!

Warning: using higher degrees does not always improve accuracy!

(see also: Runge phenomenon (polynomial interpolation, oscillation at the edges of an interval), Gibbs phenomenon ...)

Deterministic methods - II



• $(x_i, f(x_i))$

Warning:
using high-order piecewise polynomial interpolation: possible strong oscillations between consecutive $(x_i, f(x_i))$, giving a bad interpolation of f(x).

Here: f(x) step function; - linear interp.; - cubic spline

Other deterministic methods

Numerical integration; other deterministic methods:

- in the simplest equally-spaced-point methods, we choose weights to calculate the average of the function:

$$\int_a^b f(x)dx \approx F_N = \sum_{i=1}^{N} v_i f(x_i)$$

rectangular rule: $x_i = a + \frac{b-a}{N} i, \qquad v_i = \frac{b-a}{N} \quad \forall i = 1, \ldots N-1$

trapezoidal rule: $x_i = a + \frac{b-a}{N} i, \qquad v_i = \frac{b-a}{N} \quad \forall i \neq 1, N; \quad v_1 = v_N = \frac{b-a}{2N}$

...

(at variance with these methods, in MC methods such as the 'importance sampling', we choose only points, not weights)

Numerical integration; other deterministic methods:

- in the simplest equally-spaced-point methods, we choose weights to calculate the average of the function:

$$\int_a^b f(x)dx \approx F_N = \sum_{i=1}^{N} v_i f(x_i)$$

idea: choose not only weights but also points: more degrees of freedom!

$$x_i = ?, \quad v_i = ?$$

Another deterministic method: Gaussian quadrature - I

Consider $\int_a^b f(x)dx$ and a function $W(x)$ defined on $[a,b]$

We can always formally write:

$$\int f(x)dx = \int W(x)F(x)dx \approx \sum_{j=1}^{N} w_j F(x_j) = \sum_{j=1}^{N} w_j \frac{f(x_j)}{W(x_j)} = \sum_{j=1}^{N} v_j f(x_j)$$

with $v_j = w_j / W(x_j)$

to be determined, depending on $W(x)$ (*)

(This will be convenient in particular if the resulting $F(x)$ is smooth, but not necessarily)

(*) in general: $w_j \neq W(x_j)$

Another deterministic method: Gaussian quadrature - II

Consider $\int_a^b f(x)dx$ and a function $W(x)$ defined on $[a,b]$

We can always formally write:

$$\int f(x)dx = \int W(x)F(x)dx \overset{(*)}{\approx} \sum_{j=1}^{N} w_j F(x_j) = \sum_{j=1}^{N} w_j \frac{f(x_j)}{W(x_j)} = \sum_{j=1}^{N} v_j f(x_j)$$

For a given $W(x)$, the N points and weights $\{x_j\}, \{w_j\}$ can be chosen to make the approximate relationship (*) an exact equality if $F(x)$ is a 2N-1 degree polynomial.

Another deterministic method: Gaussian quadrature - III

Consider $$\int_{x_1}^{x_2} W(x)F(x)dx = \sum_{j=1}^{N} w_j F(x_j)$$

$F(x)$ a 2N - 1 degree polynomial. Which are the N $\{x_j\}, \{w_j\}$?

If there is a set of polynomials $\{p_N(x)\}$ which are orthogonal in the same interval and for the same weight function $W(x)$:

$$\langle p_N | p_{N'} \rangle_W = \delta_{N,N'} \ , \ i.e., \ \int_{x_1}^{x_2} W(x)p_N(x)p_{N'}(x)dx = \delta_{N,N'}$$

the points $\{x_i\}$ are exactly the roots of the $p_N(x)$ polynomials. The weights $\{w_j\}$ are related to them, but in general $w_j \neq W(x_j)$.

Gauss-Legendre quadrature

Consider $\int_{x_1}^{x_2} W(x)F(x)dx = \sum_{j=1}^{N} w_j F(x_j)$

with $F(x)$ a 2N - 1 degree polynomial.

If: $W(x) = 1$ and $x_1 = -1, \quad x_2 = 1$,

the Legendre polynomials $\{P_N(x)\}$ defined by:

$$(j+1)P_{j+1} = (2j+1)xP_j - jP_{j-1}$$

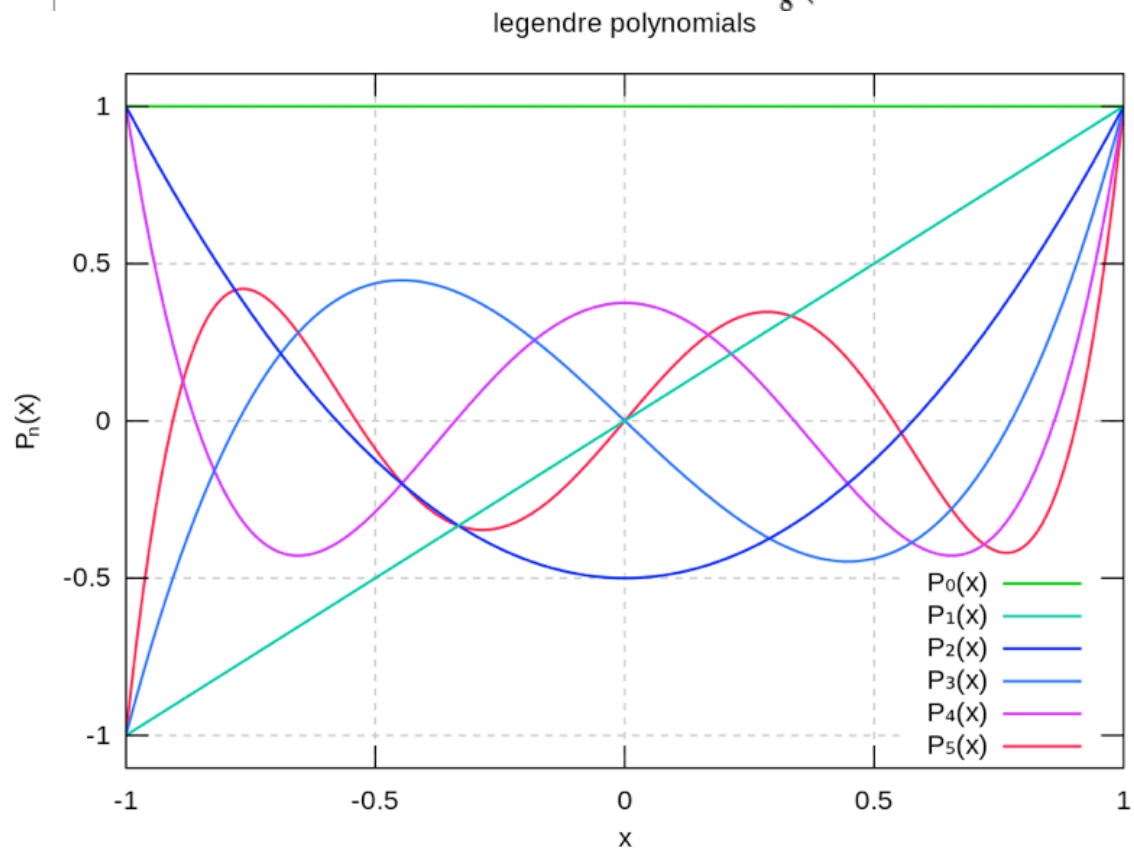
are orthogonal in [-1,1] with $W(x)$=1;

$\{x_j\}, \{w_j\}$ are such that $P_N(x_i) = 0$ and

$$w_i = \frac{2}{(1-x_i^2)[P'_N(x_i)]^2}$$

The first few Legendre polynomials are:

n	$P_n(x)$
0	1
1	x
2	$\frac{1}{2}(3x^2-1)$
3	$\frac{1}{2}(5x^3-3x)$
4	$\frac{1}{8}(35x^4-30x^2+3)$
5	$\frac{1}{8}(63x^5-70x^3+15x)$



polynomials are odd or even in x ⇨ roots are even

Legendre polynomials in Physics: examples of applications

1) For a polynomial expansion of a gravitational or coulombic potential:

$$\frac{1}{|\mathbf{x}-\mathbf{x}'|} = \frac{1}{\sqrt{r^2+r'^2-2rr'\cos\gamma}} = \sum_{\ell=0}^{\infty} \frac{r'^{\ell}}{r^{\ell+1}} P_\ell(\cos\gamma)$$

where r and r' are the lengths of the vectors $\mathbf{x}$ and $\mathbf{x}'$ respectively and γ is the angle between those two vectors.

2) solution of Laplace's equation of the static potential, $\nabla^2\Phi(\mathbf{x}) = 0$, in a charge-free region of space, if the boundary conditions have axial symmetry :

θ is the angle between the position of the observer and the $\hat{\mathbf{z}}$ axis (the zenith angle) ;

the solution for the potential will be

$$\Phi(r,\theta) = \sum_{\ell=0}^{\infty} \left[A_\ell r^\ell + B_\ell r^{-(\ell+1)}\right] P_\ell(\cos\theta)$$

3) solving Schrödinger equation in three dimensions for a central force :

the **associated Legendre polynomials** are derivatives of ordinary Legendre polynomials ($m \geq 0$)

$$P_\ell^m(x) = (-1)^m (1-x^2)^{m/2} \frac{d^m}{dx^m} (P_\ell(x))$$

The Legendre polynomials are closely related to the spherical harmonics

$$Y_{\ell,m}(\theta,\phi) = \sqrt{\frac{(2\ell+1)(\ell-m)!}{4\pi(\ell+m)!}} P_\ell^m(\cos\theta)\, e^{im\phi} \qquad -\ell \leq m \leq \ell.$$

Gaussian quadrature

In practice, we choose W(x) and N and use the set of N points and weights $\{x_j\}, \{w_j\}$ for the approximate integration:

$$\int f(x)dx = \int W(x)F(x)dx \approx \sum_{j=1}^{N} w_j F(x_j) = \sum_{j=1}^{N} w_j \frac{f(x_j)}{W(x_j)} = \sum_{j=1}^{N} v_j f(x_j)$$

Gauss-Legendre quadrature

For: $x_1 = -1, \quad x_2 = 1$

(see slide 10 for the list of the first polynomials)

N	i	x_i	w_i	degree
1	1	0	2	1
2	1	-0.577350269189626	1	3
	2	0.577350269189626	1	
3	1	-0.774596669241483	0.555555555555556	5
	2	0	0.888888888888889	
	3	0.774596669241483	0.555555555555556	
4	1	-0.861136311594053	0.347854845137454	7
	2	-0.339981043584856	0.652145154862546	
	3	0.339981043584856	0.652145154862546	
	4	0.861136311594053	0.347854845137454	

degree of the polynomial exactly integrable

The integration in an interval [a,b] different from [-1,1] ("old") can be easily done performing the scaling:

$$x_{new} = \frac{b-a}{2} x_{old} + \frac{b+a}{2} \quad \text{and} \quad w_{new} = \frac{b-a}{2} w_{old}$$

Gauss-Legendre quadrature

In case of classical, well known, orthogonal polynomials, ready-to-use subroutines exist for the computation of Abscissas and Weights $\{x_j\}, \{w_j\}$

e.g. GAULEG(x_1, x_2, x, w, n) of *Numerical Recipes* which, given x_1, x_2, n, provides as output the arrays $x(n), w(n)$

Some programs:

on https://moodle2.units.it/

int.f90 (trapeziodal and Simpson integration)
gauleg-IIorder.f90
gauleg-others.f90 (generation of points up to 15 points in [-1,1] using GAULEG adapted from "Numerical Recipes" (self-contained) and some tests for easy-to-integrate functions)

In the subdirectory: **gauss-nr90/**
find the original routine from "Numerical Recipes" and related external routines/modules/interfaces and a main program for test (see following slide)

Gauss-Legendre from Numerical Recipes

Use of GAULEG:

In order to use the routines of Numerical Recipes, you have to compile and link the main program with:

- the subroutine **gauleg.f90** which gives points and abscissas
- **nrtype.f90** containing type declarations;
- **nrutil.f90** containing **moduli** and utilities;
- **nr.f90** containing (through the interfaces) the conventions

to call the subroutines with the main program

i) You must first compile these files with the option "-c":
this produces .mod and .o (the objects).

ii) In a second step compile the main program.

iii) Finally you link all the files *.o and produce the executable:

gfortran -c nrtype.f90 nrutil.f90 nr.f90 gauleg.f90
gfortran -c gauleg_nr_test.f90
gfortran -o a.out gauleg_nr_test.o nrtype.o nrutil.o nr.o gauleg.o

gauleg.f90 from Numerical Recipes

```
SUBROUTINE gauleg(x1,x2,x,w)
  USE nrtype; USE nrutil, ONLY : arth,assert_eq,nrerror
  IMPLICIT NONE
  REAL(SP), INTENT(IN) :: x1,x2
  REAL(SP), DIMENSION(:), INTENT(OUT) :: x,w
  REAL(DP), PARAMETER :: EPS=3.0e-14_dp
  INTEGER(I4B) :: its,j,m,n
  INTEGER(I4B), PARAMETER :: MAXIT=10
  REAL(DP) :: xl,xm
  REAL(DP), DIMENSION((size(x)+1)/2) :: p1,p2,p3,pp,z,z1
  LOGICAL(LGT), DIMENSION((size(x)+1)/2) :: unfinished
  n=assert_eq(size(x),size(w),'gauleg')
  m=(n+1)/2
  xm=0.5_dp*(x2+x1)
  xl=0.5_dp*(x2-x1)
  z=cos(PI_D*(arth(1,1,m)-0.25_dp)/(n+0.5_dp))
     ...
     ...
     ...
  x(1:m)=xm-xl*z
  x(n:n-m+1:-1)=xm+xl*z
  w(1:m)=2.0_dp*xl/((1.0_dp-z**2)*pp**2)
  w(n:n-m+1:-1)=w(1:m)
END SUBROUTINE gauleg
```

`nrtype.f90` from Numerical recipes

```
MODULE nrtype
  INTEGER, PARAMETER :: I4B = SELECTED_INT_KIND(9)
  INTEGER, PARAMETER :: I2B = SELECTED_INT_KIND(4)
  INTEGER, PARAMETER :: I1B = SELECTED_INT_KIND(2)
  INTEGER, PARAMETER :: SP = KIND(1.0)
  INTEGER, PARAMETER :: DP = KIND(1.0D0)
  INTEGER, PARAMETER :: SPC = KIND((1.0,1.0))
  INTEGER, PARAMETER :: DPC = KIND((1.0D0,1.0D0))
  INTEGER, PARAMETER :: LGT = KIND(.true.)
  REAL(SP), PARAMETER :: PI=3.141592653589793238462643383279502884197_sp
  REAL(SP), PARAMETER :: PIO2=1.570796326794896619231321691639751442098584_sp
  REAL(SP), PARAMETER :: TWOPI=6.283185307179586476925286766559005768394_sp
  REAL(SP), PARAMETER :: SQRT2=1.414213562373095048801688724209698078569671_sp
  REAL(SP), PARAMETER :: EULER=0.577215664901532860606512090082402431042159_sp
  REAL(DP), PARAMETER :: PI_D=3.141592653589793238462643383279502884197_dp
  REAL(DP), PARAMETER :: PIO2_D=1.570796326794896619231321691639751442098584_dp
  REAL(DP), PARAMETER :: TWOPI_D=6.283185307179586476925286766559005768394_dp
  TYPE sprs2_sp
     INTEGER(I4B) :: n,len
     REAL(SP), DIMENSION(:), POINTER :: val
     INTEGER(I4B), DIMENSION(:), POINTER :: irow
     INTEGER(I4B), DIMENSION(:), POINTER :: jcol
  END TYPE sprs2_sp
  TYPE sprs2_dp
     INTEGER(I4B) :: n,len
     REAL(DP), DIMENSION(:), POINTER :: val
     INTEGER(I4B), DIMENSION(:), POINTER :: irow
     INTEGER(I4B), DIMENSION(:), POINTER :: jcol
  END TYPE sprs2_dp
END MODULE nrtype
```

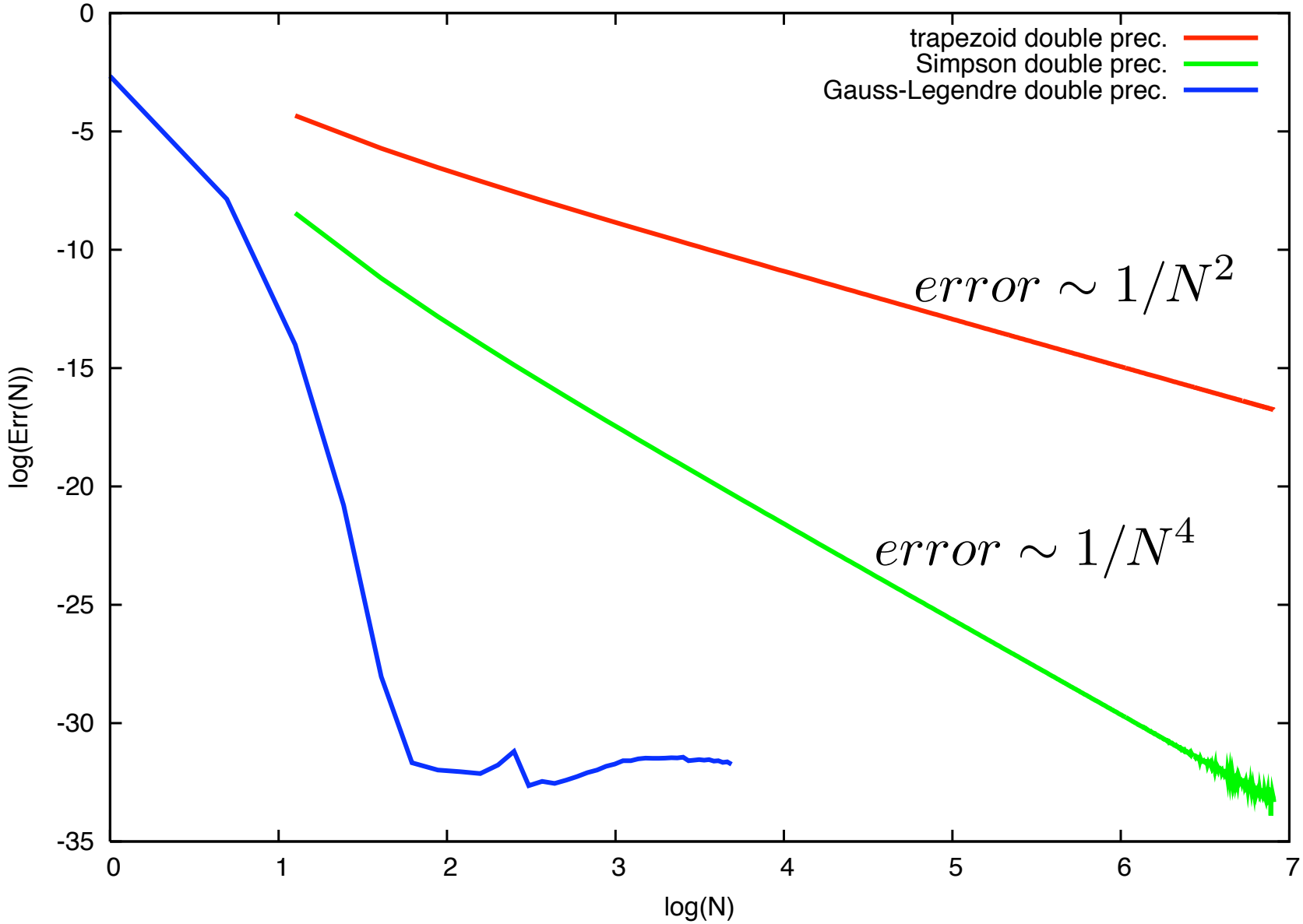
nr.f90 from Numerical Recipes

```
MODULE nr
  INTERFACE
     SUBROUTINE gauleg(x1,x2,x,w)
       USE nrtype
       REAL(SP), INTENT(IN) :: x1,x2
       REAL(SP), DIMENSION(:), INTENT(OUT) :: x,w
     END SUBROUTINE gauleg
  END INTERFACE
  ! ... the original file contains several other  INTERFACES ...
END MODULE nr
```

nrutil.f90 (Here only for: array_copy, arth, assert_eq, nrerror)

```
MODULE nrutil
  USE nrtype
  IMPLICIT NONE
  INTEGER(I4B), PARAMETER :: NPAR_ARTH=16,NPAR2_ARTH=8

    ...
    ...
    ...

  INTERFACE array_copy
     MODULE PROCEDURE array_copy_r, array_copy_d, array_copy_i
  END INTERFACE

    ...
  ! ... l'originale contiene ancora molte altre INTERFACEs....
CONTAINS

  SUBROUTINE array_copy_r(src,dest,n_copied,n_not_copied)
    REAL(SP), DIMENSION(:), INTENT(IN) :: src

    ...


   !  .... and many other FUNCTIONs and SUBROUTINEs ....
 END MODULE nrutil
```

Numerical integration, deterministic methods: comparison of errors in 1D



(double precision needed to appreciate the convergence of Gauss-Legendre numerical estimate)

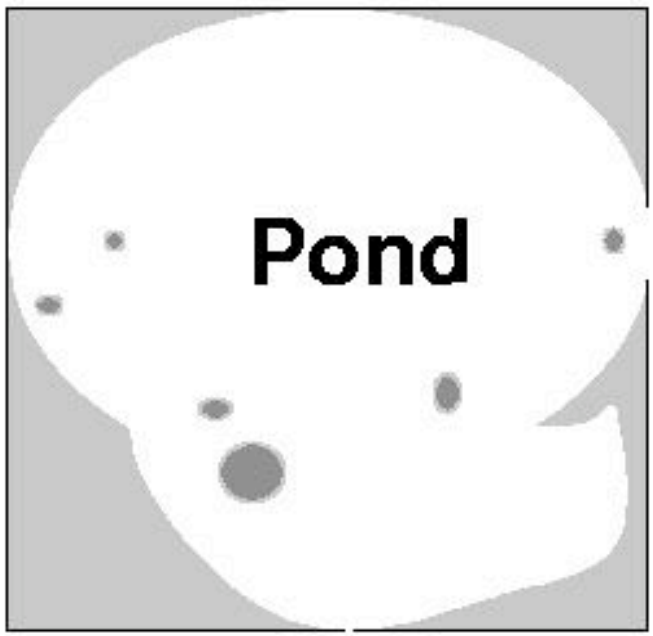
Monte Carlo methods

Monte Carlo methods:

"acceptance-rejection" or "hit or miss"

(to calculate areas)

which is A_{pond} ?



- enclose the pond in a box of Area A_{box}
- throw pebbles uniformly and randomly in the box
- count the number of pebbles felt in the pond with respect to the number felt in the box
- Assuming a uniform distribution, the number of pebbles falling into the ponds is proportional to the area of the pond:

$$\frac{N_{pond}}{N_{pond}+N_{box}} = \frac{A_{pond}}{A_{box}}$$

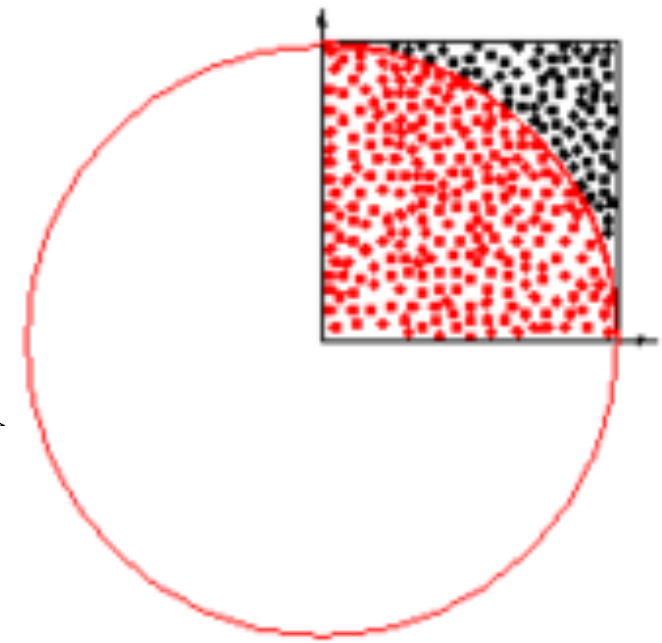
$$\Rightarrow \quad A_{pond} = \frac{N_{pond}}{N_{pond}+N_{box}} A_{box}$$

Monte Carlo methods:

"acceptance-rejection" or "hit or miss"

(to calculate areas)

$$\pi = ???$$



N random points in the unit square
coordinates x_i, y_i

Then, the number of
points N_c lying within the quarter circle (i.e. fulfilling the relation $x^2 + y^2 \leq 1$) is compared to the total number N of points and the fraction will give us an approximate value of π:

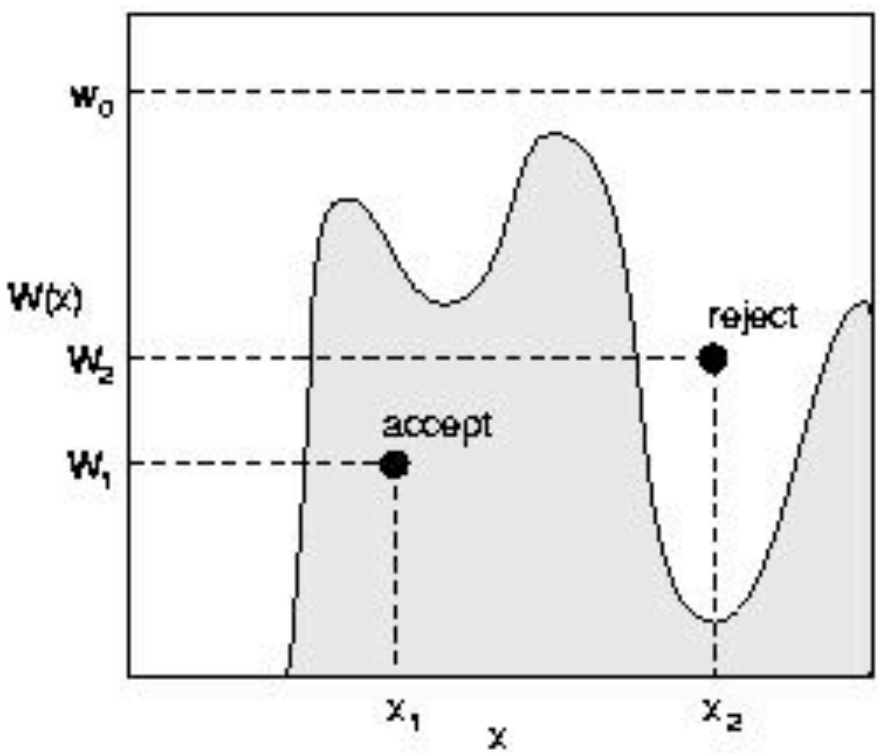
$$\pi(N) = 4\frac{N_c(N)}{N}$$

Monte Carlo methods:

"acceptance-rejection" or "hit or miss"

(to calculate definite integrals)

$$\int W(x)dx = \ ?$$



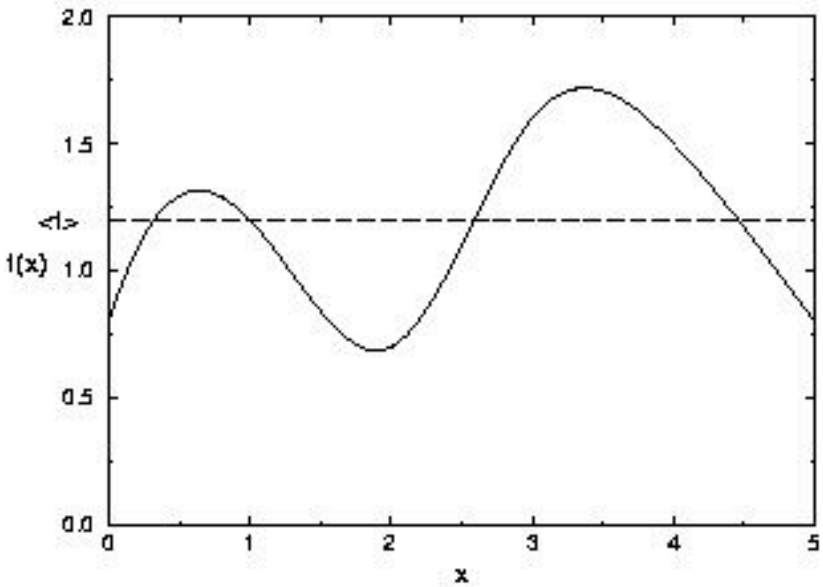
For W(x) positive in the integration interval, the value of the area under W(x) can be obtained by producing random points (i.e. (x,y) random pairs) uniformly distributed in a rectangle containing W(x).

For each point (x,y) compare y with W(x): if y<W(x), the point is accepted. The area under W(x) is the number of points accepted divided by the total number of points generated and multiplied by the area of the rectangle.

(remember: also used to generate random numbers x_i distributed according W(x))

Other simple
Monte Carlo methods

We can always write: $$I = \int_a^b f(x)dx = (b-a)\langle f \rangle$$

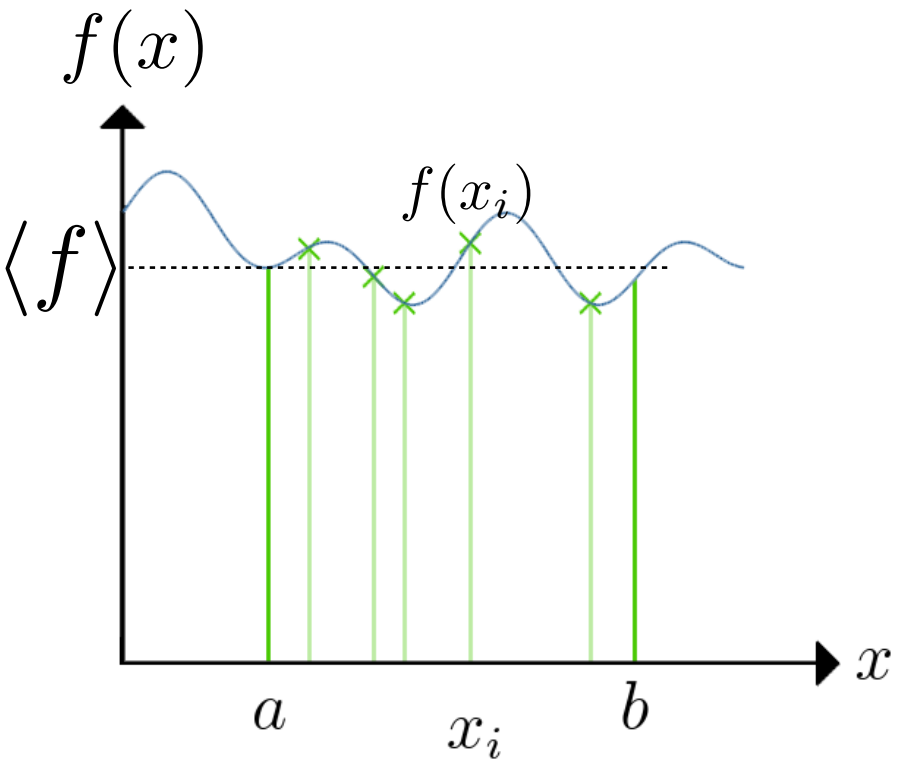


i.e., the value of the integral of f(x) between a and b equals the length of the interval (b-a) times the average value of the function <f> over the same interval.

(If f:[a,b] → R is a continuous function, then there exists a number c in [a,b] such that f(c)=<f>
(mean value theorem for integration))

how to estimate <f> efficiently and accurately?

A simple Monte Carlo method: "sample mean"



$$I = \int_a^b f(x)dx = (b-a)\langle f \rangle$$

The sample mean can be calculated by sampling the function *(if smooth enough...)* with a sequence of N uniform random numbers in [a,b]:

$$\langle f \rangle \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

$$\int_a^b f(x)dx \approx (b-a)\frac{1}{N}\sum_{i=1}^{N} f(x_i) = (b-a)\langle f \rangle$$

Monte Carlo methods:

error estimate

Example: MC estimate of π (exact value known)

We can use either acceptance-rejection or sample mean method: $I = 4\int_0^1 \sqrt{1-x^2} = \pi = 3.1416\ldots$

Since we know the "exact" result *I*, we can calculate the **error** in two ways:

1) the **actual error** from the difference with respect to the exact value:

$\Delta_n = |F_n - I|$ with $F_n = (b-a)\frac{1}{n}\sum_{i=1}^{n} f(x_i), \quad x_i \text{ random}$

2) the numerical error from the **variance of the data** $\{f(x_i)\}$:

$\sigma^2 = \langle f^2 \rangle - \langle f \rangle^2$

where

$\langle f \rangle = \frac{1}{n}\sum_{i=1}^{n} f(x_i)$ and $\langle f^2 \rangle = \frac{1}{n}\sum_{i=1}^{n} f(x_i)^2$

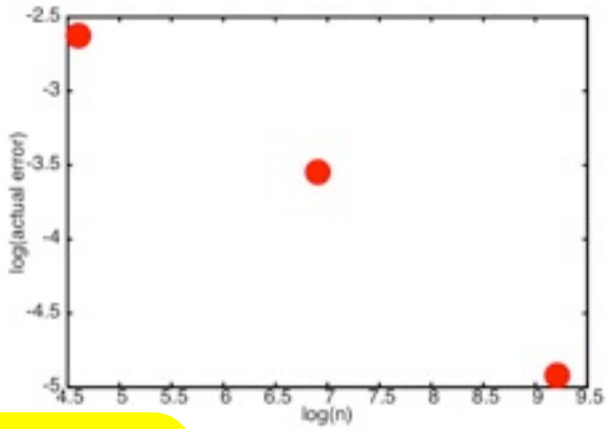
Monte Carlo methods:

error estimate

Results:

$$I = 4\int_0^1 \sqrt{1-x^2} = \pi = 3.1416\ldots$$

n	F_n	actual error	σ_n
10^2	3.0692	0.0724	0.8550
10^3	3.1704	0.0288	0.8790
10^4	3.1489	0.0073	0.8850



1) the actual error Δ_n decreases as $1/n^{1/2}$

2) the numerical error from the variance of the data, σ_n, is roughly constant and is much larger than the actual error

what is the correct error estimate?

Monte Carlo methods:
error estimate

...typically you do not know which is the "actual error" (you do not know the "true" value and you cannot compare your result with that!)....
but we would like to give an error to our numerical estimate...
(to which extent is our numerical estimate reliable?)

Two methods to estimate the error numerically from the variance of the data (**"reduction of variance"**):

I) average of the averages

II) block average

MC error handling: method I
"average of the averages"

make additional runs of n trials each.
Let M_α be the average of each run :

run α	M_α	actual error
1	3.1489	0.0073
2	3.1326	0.0090
3	3.1404	0.0012
4	3.1460	0.0044
5	3.1526	0.0110
6	3.1397	0.0019
7	3.1311	0.0105
8	3.1358	0.0058
9	3.1344	0.0072
10	3.1405	0.0011

$one\ run \equiv n = 10^4\ trials\ each$

Examples of Monte Carlo measurements of the mean value of $f(x) = 4\sqrt{1-x^2}$ in the interval $[0, 1]$. A total of 10 measurements of $n = 10^4$ trials each were made. The mean value M_α and the actual error $|M_\alpha - \pi|$ for each measurement are shown.

Calculate: $\sigma_m{}^2 = \langle M^2 \rangle - \langle M \rangle^2$ with $\langle M \rangle = \frac{1}{m}\sum_{\alpha=1}^{m} M_\alpha$, $\langle M^2 \rangle = \frac{1}{m}\sum_{\alpha=1}^{m} M_\alpha{}^2$

$\Longrightarrow \sigma_m = 0.0068$

σ_m is consistent with the results for the actual errors

MC error handling: method II "block averages"

Instead of doing additional measurements, divide them into "s SUBSETS" and let S_k be the average within each subset :

subset k	S_k
1	3.14326
2	3.15633
3	3.10940
4	3.15337
5	3.15352
6	3.11506
7	3.17989
8	3.12398
9	3.17565
10	3.17878

The variance associated to the average of the subsets ${\sigma_s}^2 = \langle S^2 \rangle - \langle S \rangle^2$ gives $\sigma_s = 0.025$, but

$\sigma_s/\sqrt{s}$, which for our example is approximately $0.025/\sqrt{(10)} \approx 0.008$.

is consistent with the actual error

Monte Carlo methods:

error estimate - variance reduction

summary

$$\sigma_n/\sqrt{n} \approx \sigma_m \approx \sigma_s/\sqrt{s}$$

from the variance of the whole set of data

Note: for uncorrelated data !

(proof)

the variance of the **average of the averages**

from the variance of the **block averages**

the most convenient!
but: change block size
and check that
it does not change

Monte Carlo methods:
summary

We have introduced :

* "acceptance-rejection"

* "sample mean" to estimate $\langle f \rangle \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i)$

both OK for smoothly varying functions, but
not very efficient for rapidly varying functions

How to improve the efficiency of MC integration?

A trick for numerical integration: "reduction of variance"

(Note: same word, but different meaning w.r.t. previous slides on error handling)

Given a function $f(x)$ to integrate, suppose that $g(x)$ exists, whose integral is known and such that:

$$|f(x) - g(x)| << \varepsilon$$

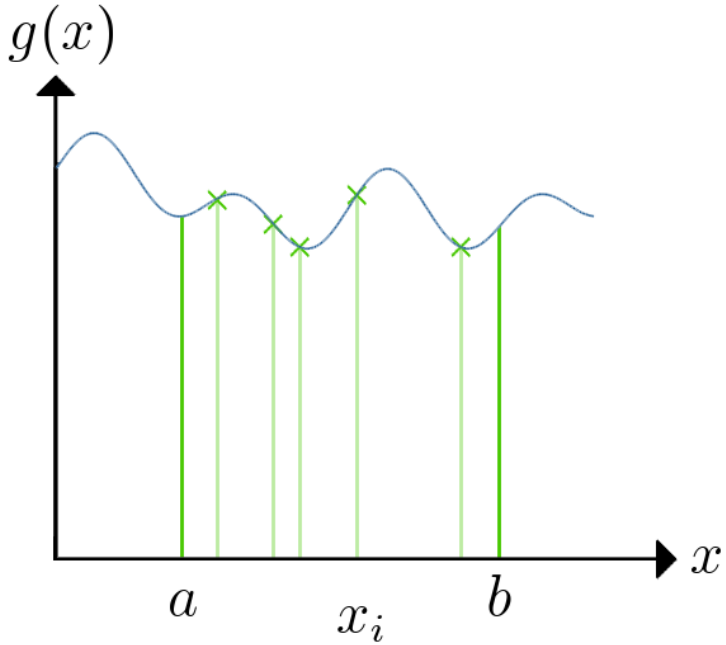
Therefore:

$$F = \int_a^b f(x)dx = \int_a^b \big((f(x)-g(x))+g(x)\big)dx = \int (f(x) - g(x))\, dx + \int g(x)dx$$

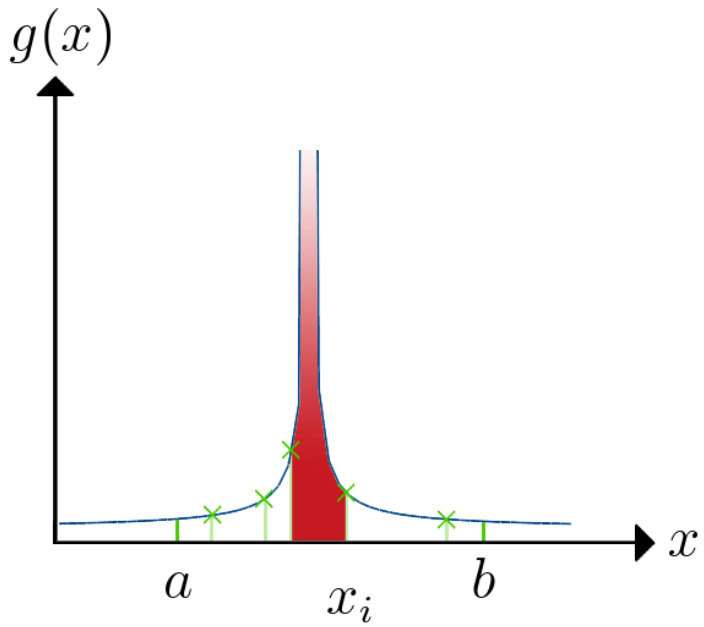
easy to calculate

Another simple Monte Carlo method: "importance sampling"

Mean value: easy to calculate for smoothly varying functions.
But not for functions rapidly varying.



smooth function



function with singularity

How to manage such cases?

Another simple Monte Carlo method: "importance sampling"

Mean value: easy to calculate for smoothly varying functions.
Idea: in order to calculate:

$$\langle f \rangle \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i)$$

consider a distribution function $p(x)$ easy to integrate analytically and close to $f(x)$:

$$F = \int_a^b f(x)dx = \int_a^b \left[\frac{f(x)}{p(x)}\right] p(x)dx = \left\langle \frac{f(x)}{p(x)} \right\rangle \int_a^b p(x)dx$$

where $\left\langle \frac{f(x)}{p(x)} \right\rangle \approx \frac{1}{N} \sum_{i=1}^{N} \left[\frac{f(x_i)}{p(x_i)}\right]$

(particular case: uniform distrib. p(x)=1/(b-a) ...)

with $\{x_i\}$ distributed according to $p(x)$

Monte Carlo methods:
"importance sampling"

Calculate:

$$F = \int_0^1 e^{-x^2}\, dx.$$

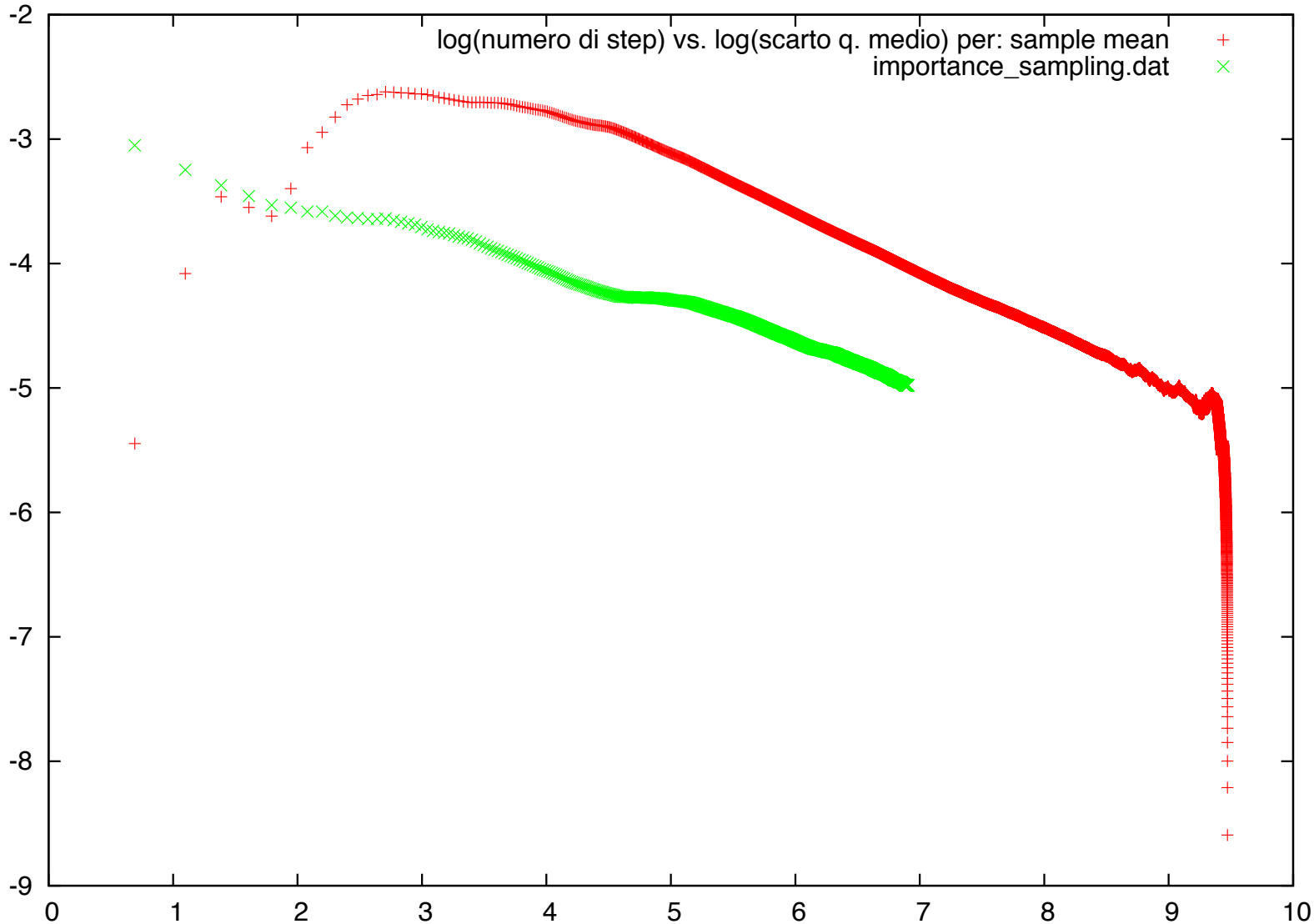
with "sample mean" with random numbers with uniform distribution or using the "importance sampling" with $p(x) = e^{-x}$

	$p(x) = 1$	$p(x) = Ae^{-x}$
n (trials)	4×10^5	8×10^3
F_n	0.7471	0.7469
σ	0.2010	0.0550
$\sigma/\sqrt{n}$	3×10^{-4}	6×10^{-4}
Total CPU time (s)	35	1.35
CPU time per trial (s)	10^{-4}	2×10^{-4}

← efficient !

(pay attention to the normalization of p(x)...)

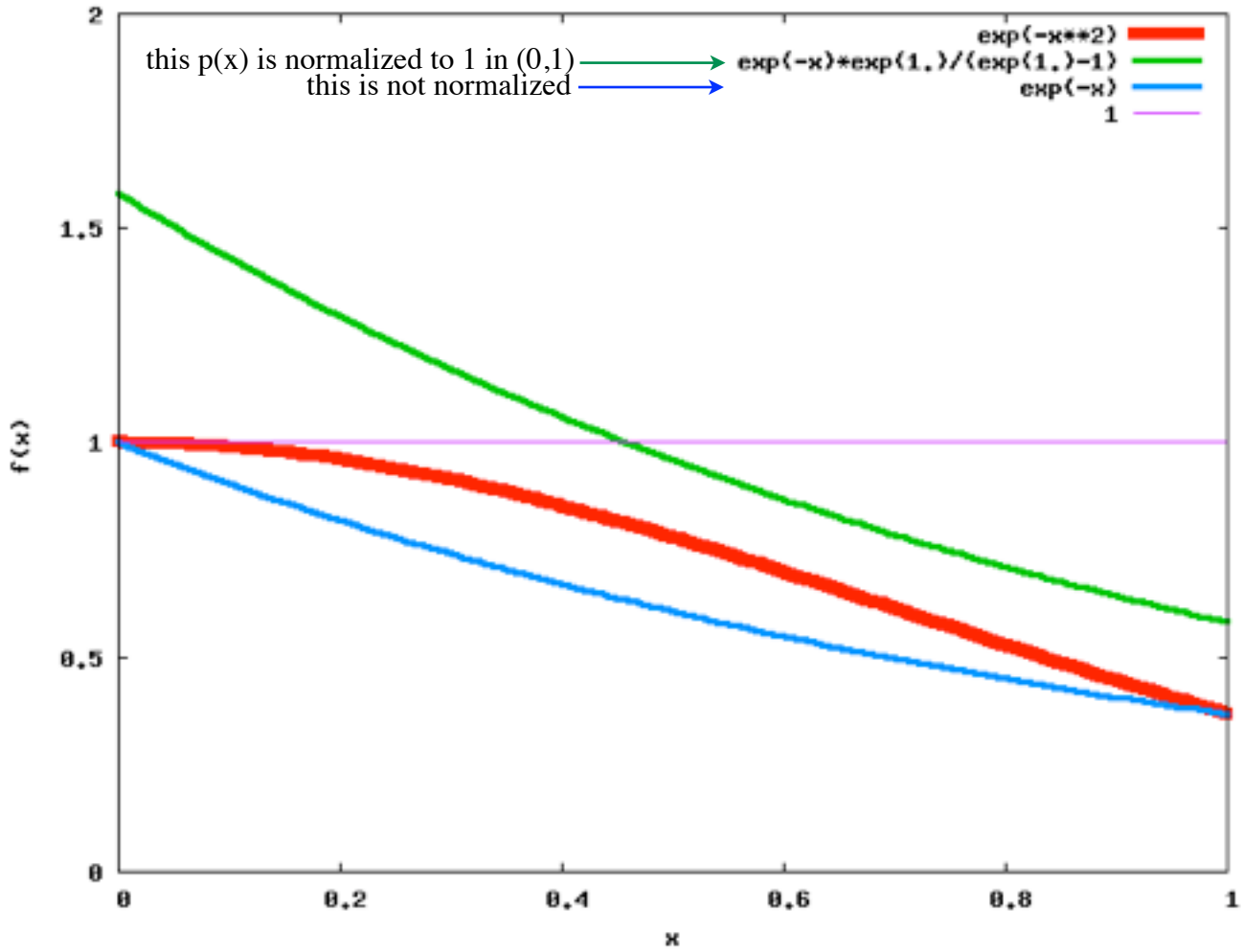
error(MC)~1/√N => see log(error) vs. log(N)
but with different prefactors
for sample means vs importance sampling



Choice of the importance sampling function

$$F = \int_0^1 e^{-x^2}\, dx.$$

$$F = \int_a^b f(x)dx = \int_a^b \left[\frac{f(x)}{p(x)}\right] p(x)dx = \left\langle \frac{f(x)}{p(x)} \right\rangle \int_a^b p(x)dx$$



(pay attention to the normalization of p(x)...)

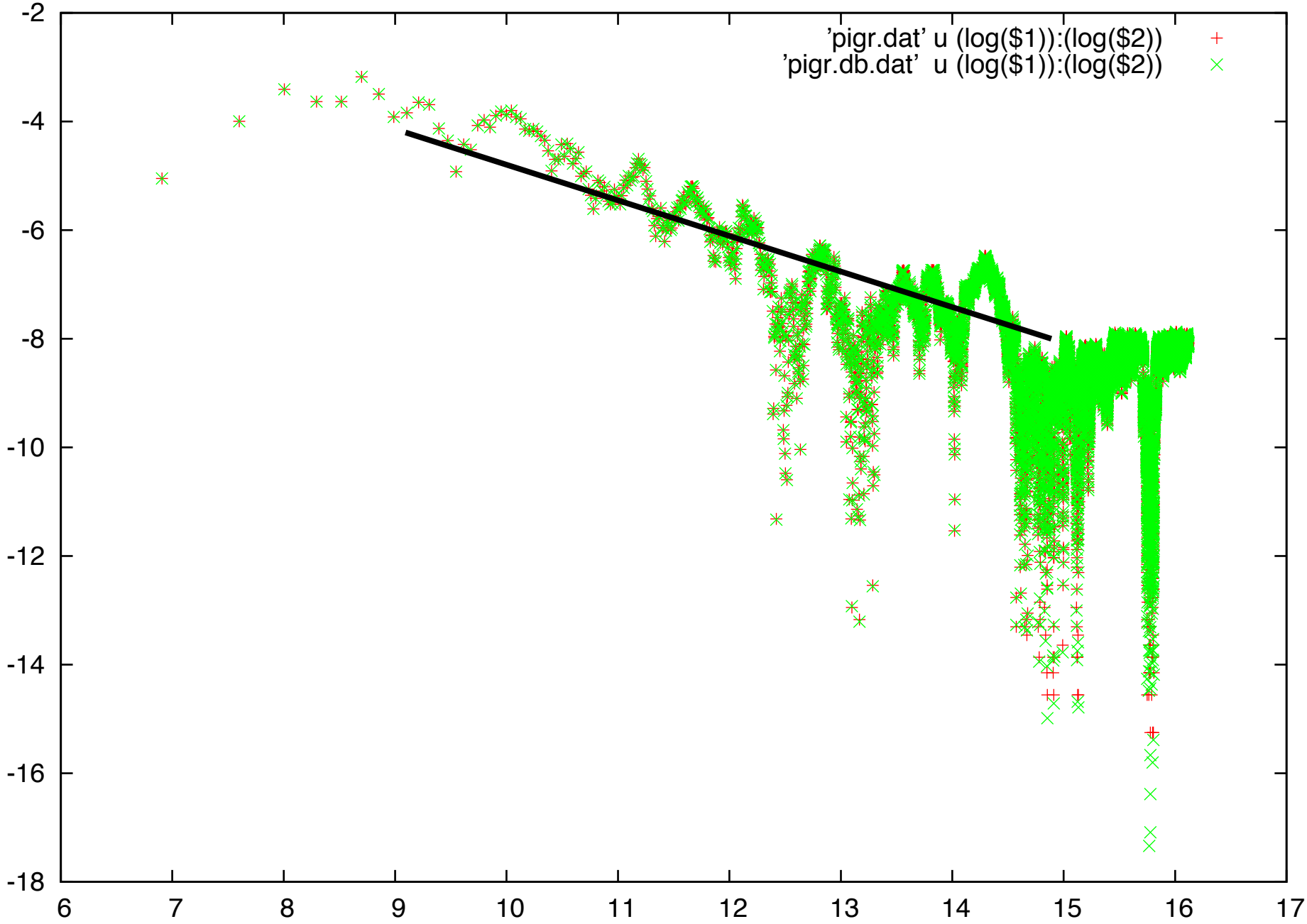
Some programs

on https://moodle2.units.it/

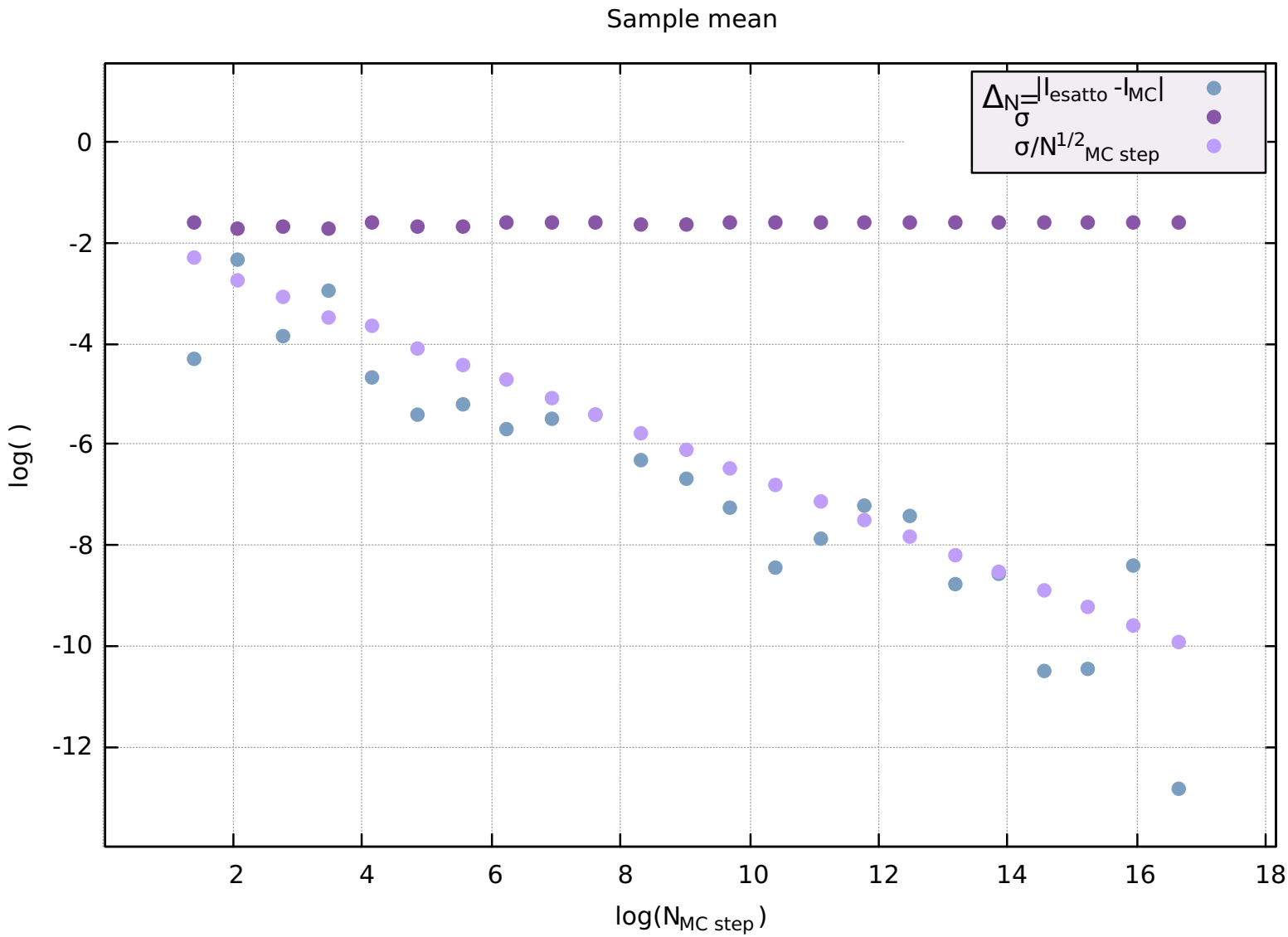
pi.f90 Monte Carlo integration for the calculation of π

For the other exercises: write yourself the codes

error(MC)~1/√N => see log(error) vs. log(N)



error(MC)~1/√N :"true" error and statistical error



(credits: G. Lautizi, a.y. 2019-20)

Summary of numerical integration (MC and deterministic) methods

MC sample mean

$$\int_a^b f(x)dx = (b-a) < f > \approx (b-a)\frac{1}{N}\sum_{i=1}^{N} f(x_i) \quad with\ \{x_i\}\ randomly\ uniformly\ distributed\ in\ [a,b]$$

$$\left(it\ can\ be\ considered\ as\ Importance\ sampling\ with\ p(x) = \frac{1}{b-a}\ in\ [a,b]\right)$$

MC importance sampling

$$\int_a^b f(x)dx = \int_a^b \frac{f(x)}{p(x)}p(x)dx = < \frac{f(x)}{p(x)} > \int_a^b p(x)dx \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}\int_a^b p(x)dx$$

$$with\ \{x_i\}\ \ randomly\ \ distributed\ \ according\ \ p(x)$$

Deterministic, equispaced points

$$\int_a^b f(x)dx \approx \sum_{i=1}^{N} v_i f(x_i) \quad with\ x_i = a + \frac{b-a}{N}i\ ,\ v_i\ to\ be\ determined$$

Deterministic, non equispaced points

$$\int_a^b f(x)dx \approx \sum_{i=1}^{N} v_i f(x_i) \quad with\ \{x_i\}\ ,\ \{v_i\}\ to\ be\ determined$$

Error estimate: comparison between deterministic and MC methods in d-dimension

Error estimate for numerical integration with deterministic methods

(Reminder from previous slides)

$$\int f(x)dx = F_n + error$$

How to evaluate the error? Consider the Taylor expansion of the integrand function and then integrate:

$$f(x) = f(x_i) + f'(x_i)(x - x_i) + \frac{1}{2}f''(x_i)(x - x_i)^2 + \dots ,$$

$$\int_{x_i}^{x_{i+1}} f(x)\,dx = f(x_i)\Delta x + \frac{1}{2}f'(x_i)(\Delta x)^2 + \frac{1}{6}f''(x_i)(\Delta x)^3 + \dots \quad (*)$$

$$\Delta x \equiv x_{i+1} - x_i$$

Error estimate for numerical integration: Rectangular approximation

(Reminder from previous slides)

$$\int_{x_i}^{x_{i+1}} f(x)dx \approx \boxed{f(x_i)\Delta x}$$

Compare ▭ with (*):

$$\int_{x_i}^{x_{i+1}} f(x)\,dx = f(x_i)\Delta x + \frac{1}{2}f'(x_i)(\Delta x)^2 + \frac{1}{6}f''(x_i)(\Delta x)^3 + \dots$$

error

(leading order in Δx)

For n intervals $(\Delta x = (b-a)/n)$: error is $n(\Delta x)^2 \sim 1/n$

(...and similarly for higher-order approximations)

Numerical integration:
multidimensional integrals

$$F = \int_R f(x,y)\, dxdy$$

The rectangular approximation gives $\Delta x \Delta y \sim (\Delta x)^2 \sim 1/n$, being n the number of parts (or pairs of points) of the integration domain:

$$\int_{x_i}^{x_{i+1}} \int_{y_i}^{y_{i+1}} f(x,y) dxdy \approx f(x_i, y_i)\Delta x \Delta y \qquad (*)$$

The Taylor expansion of the integrand function gives:

$$f(x,y) = f(x_i,y_i) + f'_x(x_i,y_i)(x-x_i) + f'_y(x_i,y_i)(y-y_i) + \dots$$

$$\int_{x_i}^{x_{i+1}} \int_{y_i}^{y_{i+1}} f(x,y) dxdy = f(x_i,y_i)\Delta x \Delta y + f'_x(x_i,y_i)\frac{(\Delta x)^2}{2}\Delta y + f'_y(x_i,y_i)\Delta x \frac{(\Delta y)^2}{2} + \dots \quad (**)$$

(*) against (**) => error
(leading order in Δx)

For n intervals: error is $n(\Delta x)^3 \sim 1/n^{1/2}$

Numerical integration: multidimensional integrals

Therefore for rectangular approx.:

d=1: $error \sim 1/n$ d=2: $error \sim 1/n^{1/2}$

In general:

if the error decreases as n^{-a} for $d = 1$, then the error decreases as $n^{-a/d}$ in d dimensions.

Classical formulas with equispaced points: slowly decreasing error for multidimensional integration !

Numerical integration:
error in MC methods

$$\sigma_n/\sqrt{n} \approx \sigma_m \approx \sigma_s/\sqrt{s}$$

(σ n is roughly constant with n ; for uncorrelated points, the variance of the averages goes like $\sim 1/n^{1/2}$)

- The average function value

$$\langle f \rangle \equiv \frac{1}{N} \sum_{i=1}^{N} f(x_i)$$

- The average squared function value

$$\langle f^2 \rangle \equiv \frac{1}{N} \sum_{i=1}^{N} f^2(x_i)$$

- Estimate of the integrand (+/- standard error)

$$\int f \, dV \approx V \langle f \rangle \pm V \sqrt{\frac{\langle f^2 \rangle - \langle f \rangle^2}{N}}$$

Numerical integration:
errors in multidimensional integrals

d	Rect.	Trap.	Simps.	MC
1	$1/n$	$1/n^2$	$1/n^4$	$1/n^{1/2}$
2	$1/n^{1/2}$	$1/n$	$1/n^2$	$1/n^{1/2}$
4	$1/n^{1/4}$	$1/n^{1/2}$	$1/n$	$1/n^{1/2}$
...	...	...	...	

if the error decreases as n^{-a} for $d = 1$, then the error decreases as $n^{-a/d}$ in d dimensions.

the error for all Monte Carlo integration methods decreases as $n^{-1/2}$ independently of the integral.

Monte Carlo convenient for multidimensional integration !

Summary: advantages of MC integration methods

- convergence as ~$N^{1/2}$ in any dimension regardless of the smoothness of the integrand
- simplicity: only two simple steps required (namely, producing a set of sampling points and evaluating the integrand function over such points)
- generality: sampling can be used even on domains that do not have a natural correspondence with the 'standard' domain $[0,1]^d$ and thus are not well-suited to numerical quadrature
- better suited than quadrature for integrands with singularities (importance sampling can handle this problem)
- flexibility: easy to add more points as needed (in the Gaussian quadrature, increasing the accuracy implies doing calculations from scratch)