993SM - Laboratory of Computational Physics
III week
October 7, 2024

Maria Peressi
Università degli Studi di Trieste - Dipartimento di Fisica
Sede di Miramare (Strada Costiera 11, Trieste)
e-mail: peressi@units.it
tel.: +39 040 2240242

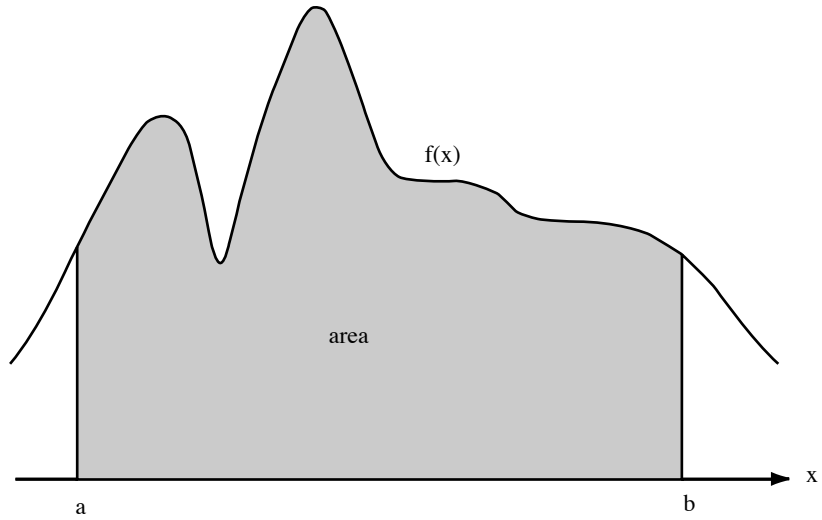
numerical integration

(now only
deterministic methods:
equispaced points & others)

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:

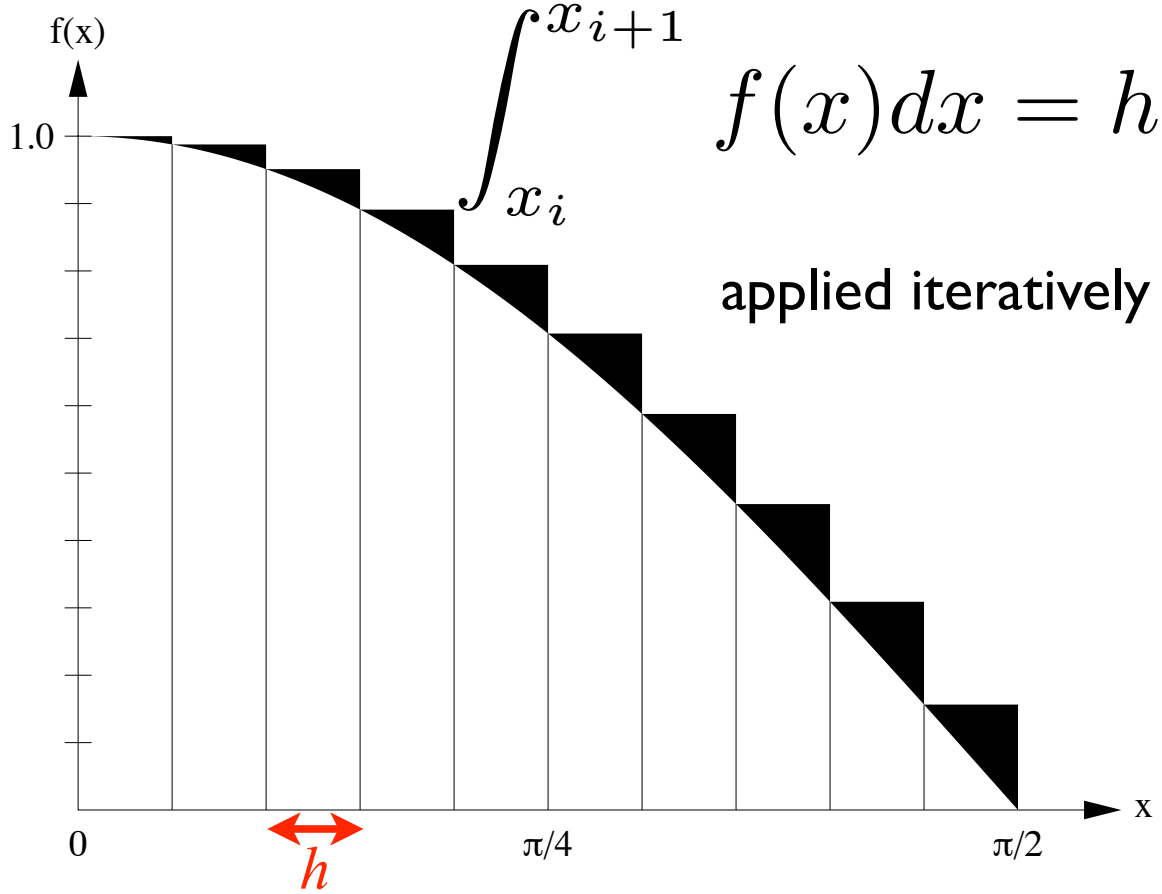
rectangular rule

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:

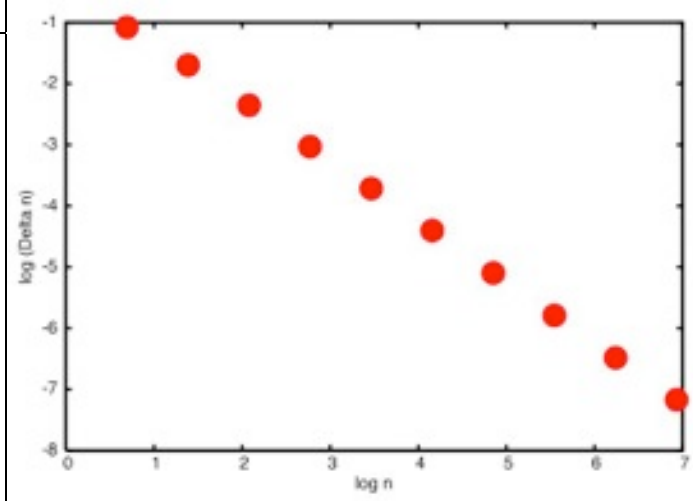
rectangular rule - error

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

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

$$\Delta_n = F_n - I$$

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:

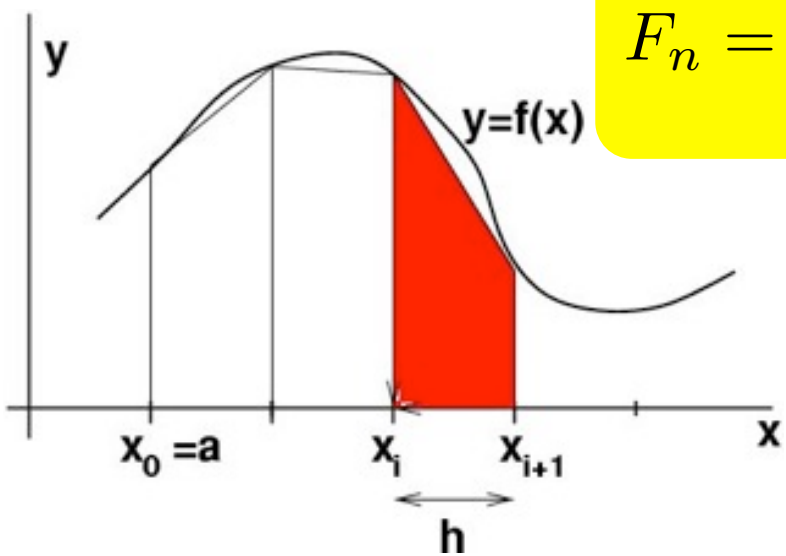
$$\int_{x_i}^{x_{i+1}} f(x)dx = h\left[\frac{1}{2}f_i + \frac{1}{2}f_{i+1}\right]$$

with error:

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

Applied iteratively over consecutive intervals:

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

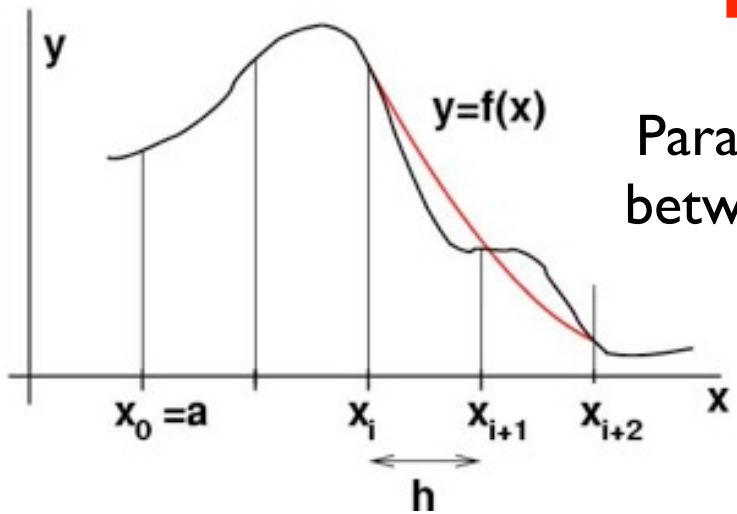


with a total error:

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

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 (**):

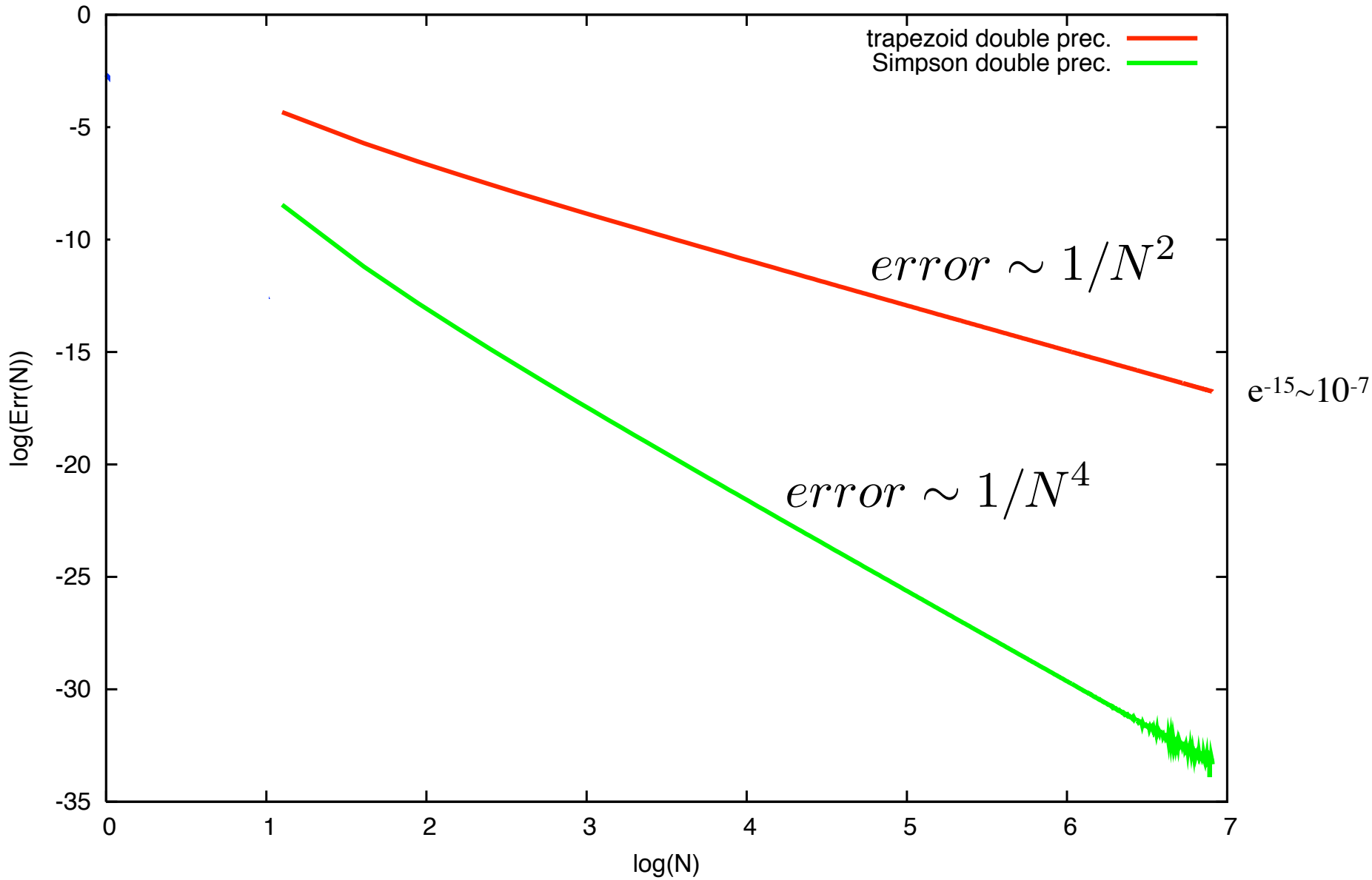
$$\int_{x_i}^{x_{i+2}} f(x)dx = f(x_i)\Delta x + \frac{1}{2!}f'(x_i)(\Delta x)^2 + \frac{1}{3!}f''(x_i)(\Delta x)^3 + \frac{1}{4!}f'''(x_i)(\Delta x)^4 + \frac{1}{5!}f''''(x_i)(\Delta x)^5 + \ldots$$

error

(leading order in Δx)

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

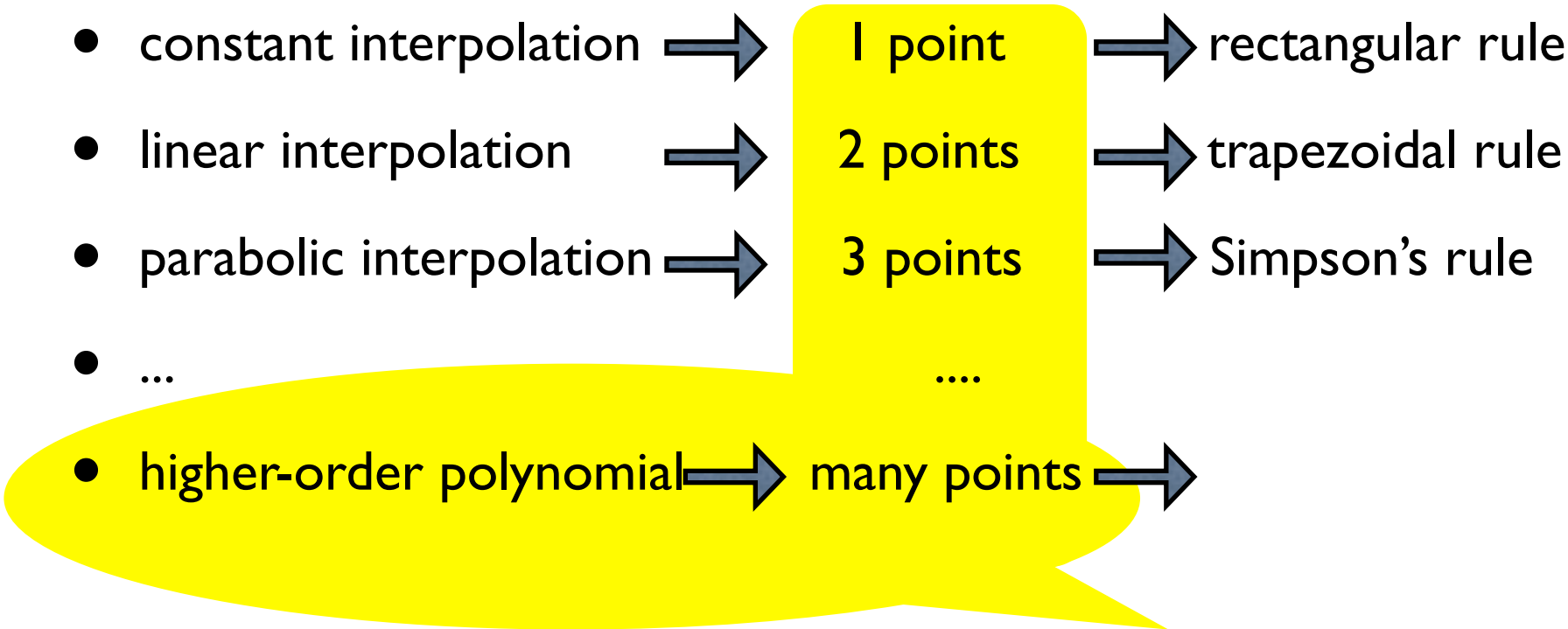
Numerical integration - deterministic methods: comparison of errors in 1D



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

Deterministic methods - I

We use a piecewise polynomial interpolation:

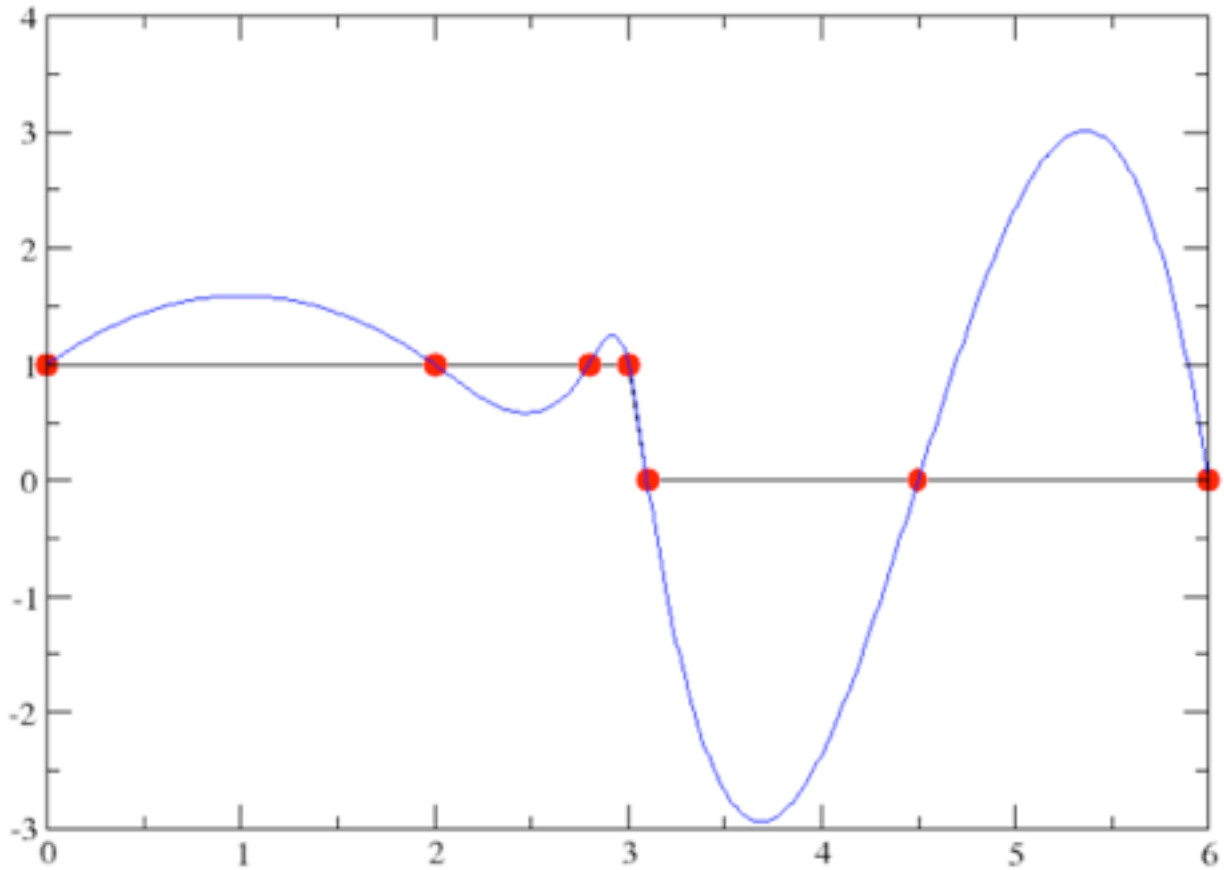


NOT CONVENIENT!

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:

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

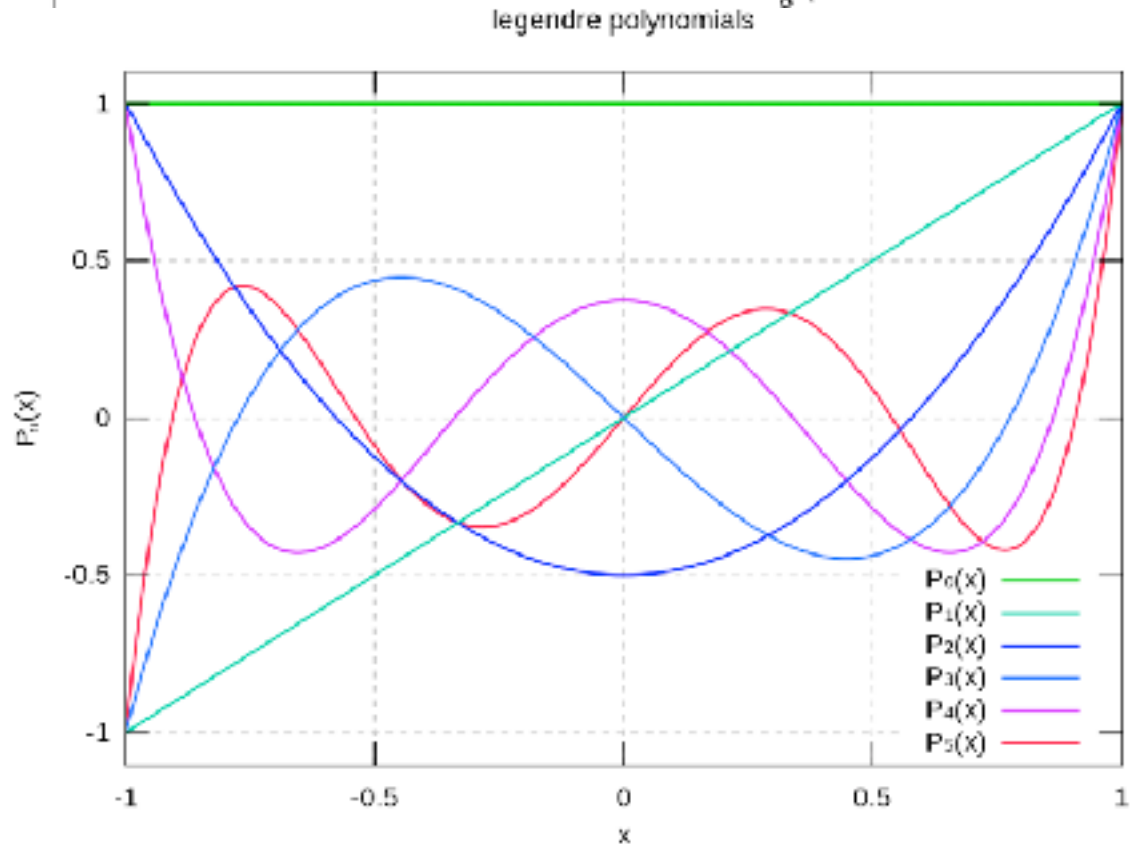
are orthogonal in [-1,1] with $W(x)$=1;

$\{x_j\}, \{w_j\}$ 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
```

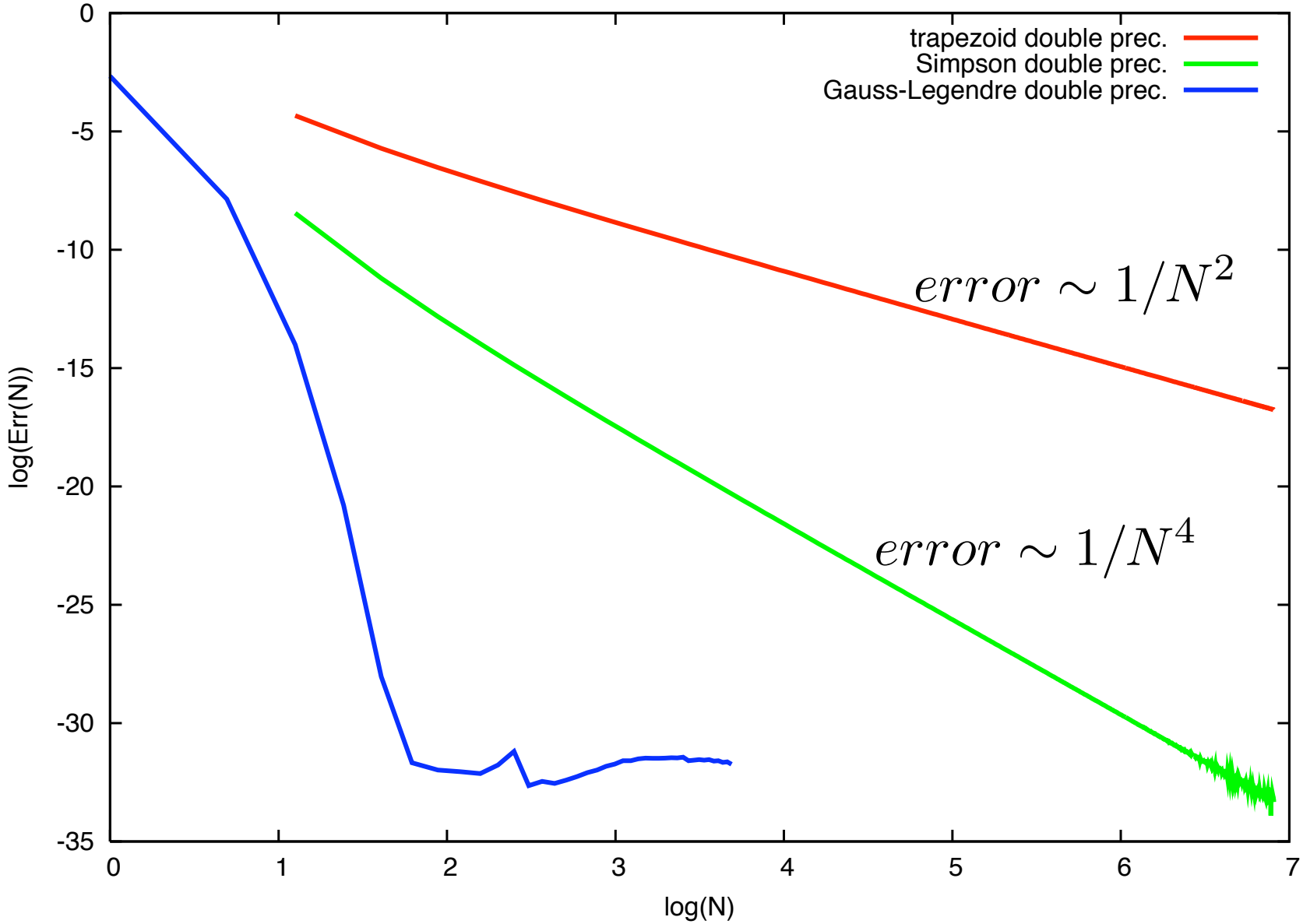
`nr.f90` from Numerical Recipes

```
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)