MA3457/CS4033: Numerical Methods for Calculus and Differential Equations

Course Materials

P A R T III

B'14
2014-2015

Numerical Integration – Conceptual Motivation

When do we need N. I.?

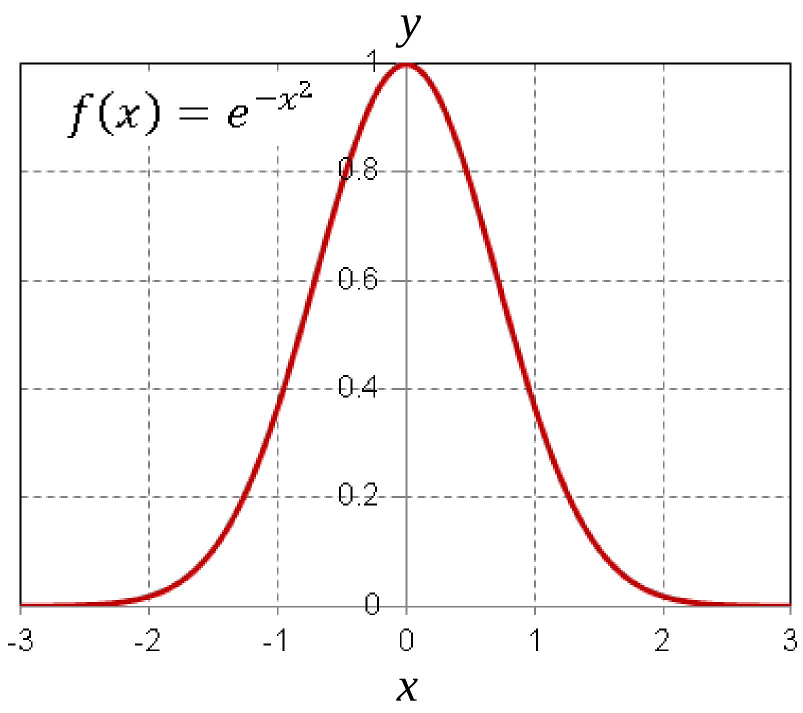
(1) If in an applied problem a Definite Integral turns out to be too complex – its evaluation requires an unreasonably strong effort,

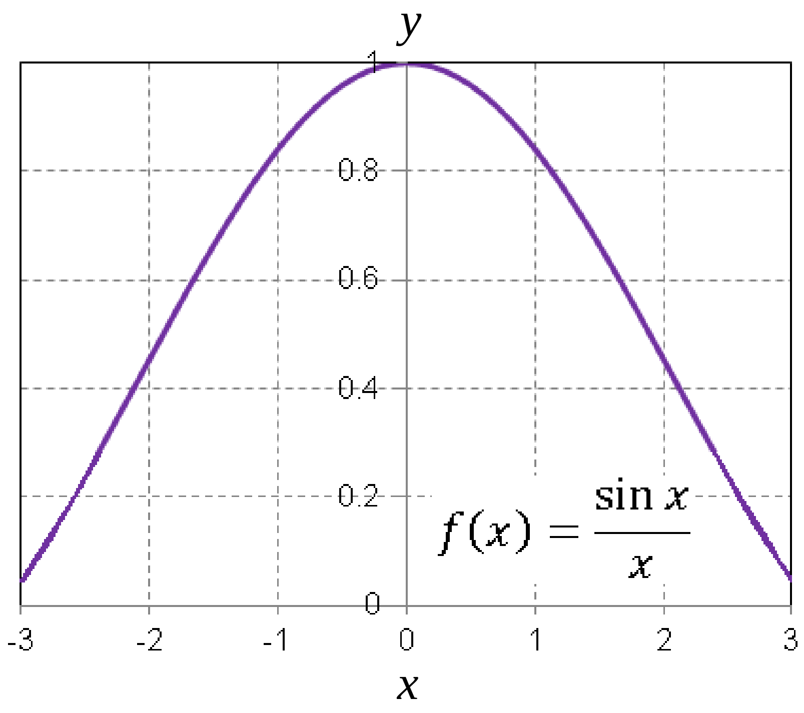
(2) If a Definite Integral still *can be evaluated, but results in* ***a very complex formula*** (which could be difficult for analysis/computation)

(3) If an *indefinite integral of some functions cannot be represented as an elementary function* (and there are many such functions in *popular applications*)

Introductory Illustrations

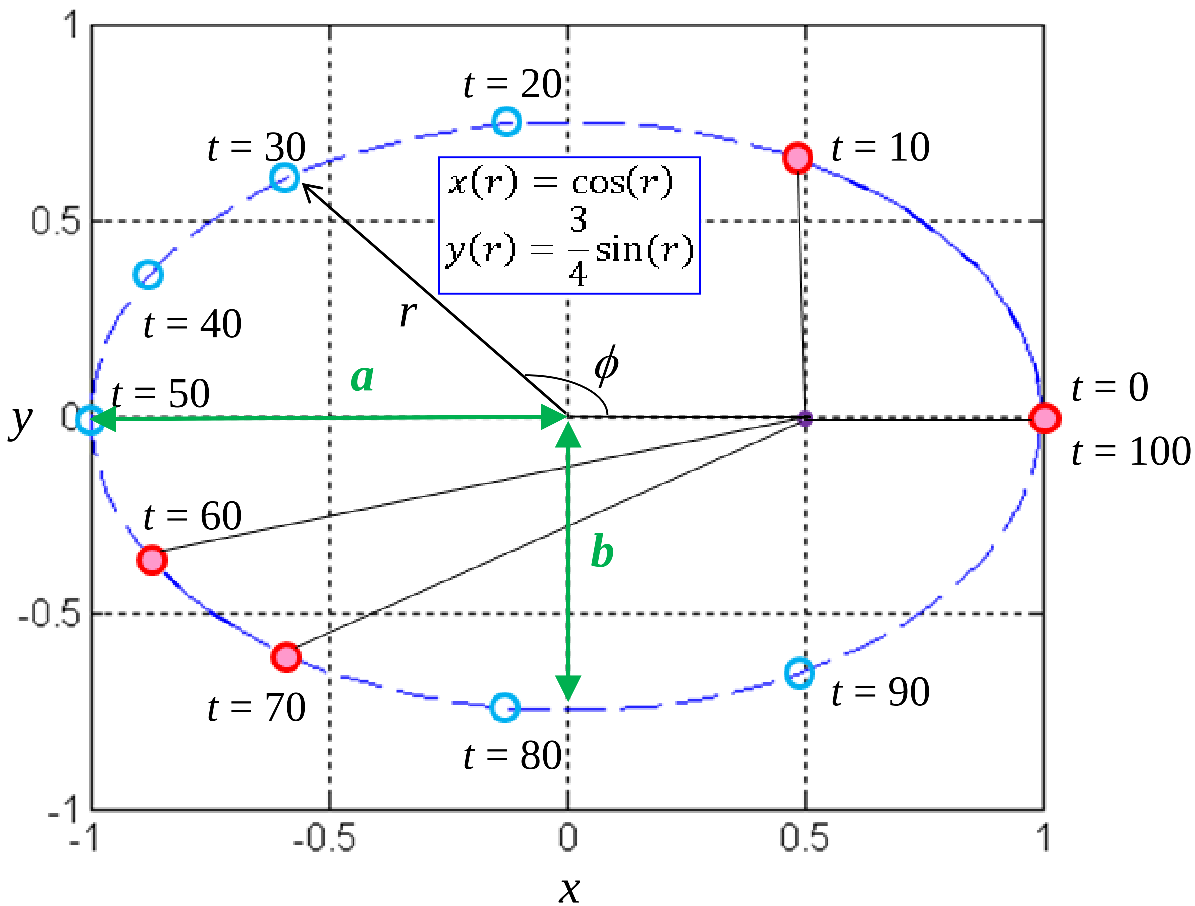
Functions That Cannot Be Obtained by Differentiation of Other Functions





Length of an Elliptical Orbit

Question: *find the distance between certain positions on the orbit*

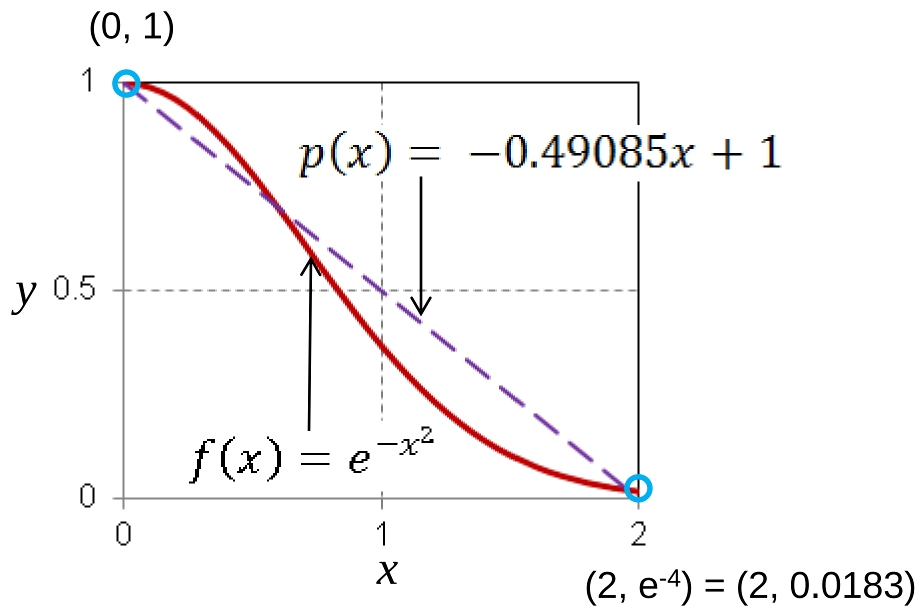


Planetary orbit – Kepler's Law: an elliptical orbit with an eccentricity e = 0.5, 10-day interval (Δt = 10)

$$b^2 = a^2(1 - e^2)$$

Trapezoid Rule

The Function and its Linear Approximation



Trapezoid Rule – MATLAB Script

LIBRARY OF MATLAB PROCEDURES

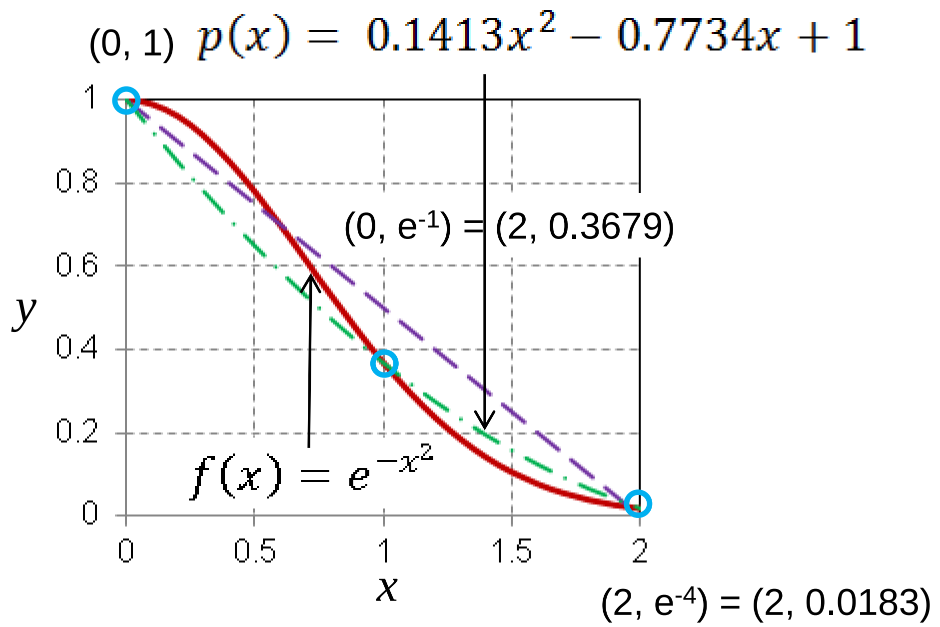
Trap

Performs numerical integration with the use of the Composite Trapezoid Rule

```
function I = Trap(f, a, b, n)
%
% The function finds integral of f using composite trapezoid rule
%
h = (b-a)/n; S = feval(f, a);
%
for i = 1 : n-1
    x(i) = a + h*i
    S = S + 2*feval(f, x(i))
end
%
S = S + feval(f, b); I = h*S/2
```

Simpson's Rule

The Function and its Quadratic Approximation



Simpson's Rule – MATLAB Script

LIBRARY OF MATLAB PROCEDURES

`simp`

Performs numerical integration with the use of the Composite Simpson Rule

```
function I = simp(f, a, b, n)
%
% The function finds integral of f using composite Simpson rule
```

Romberg Integration – MATLAB Script

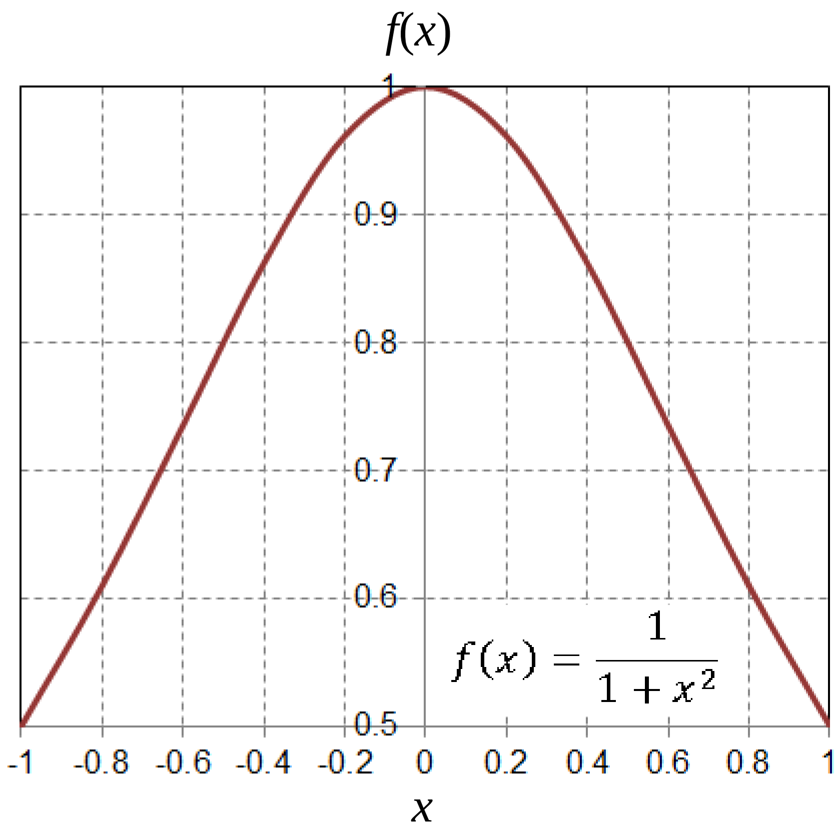
LIBRARY OF MATLAB PROCEDURES

Romb

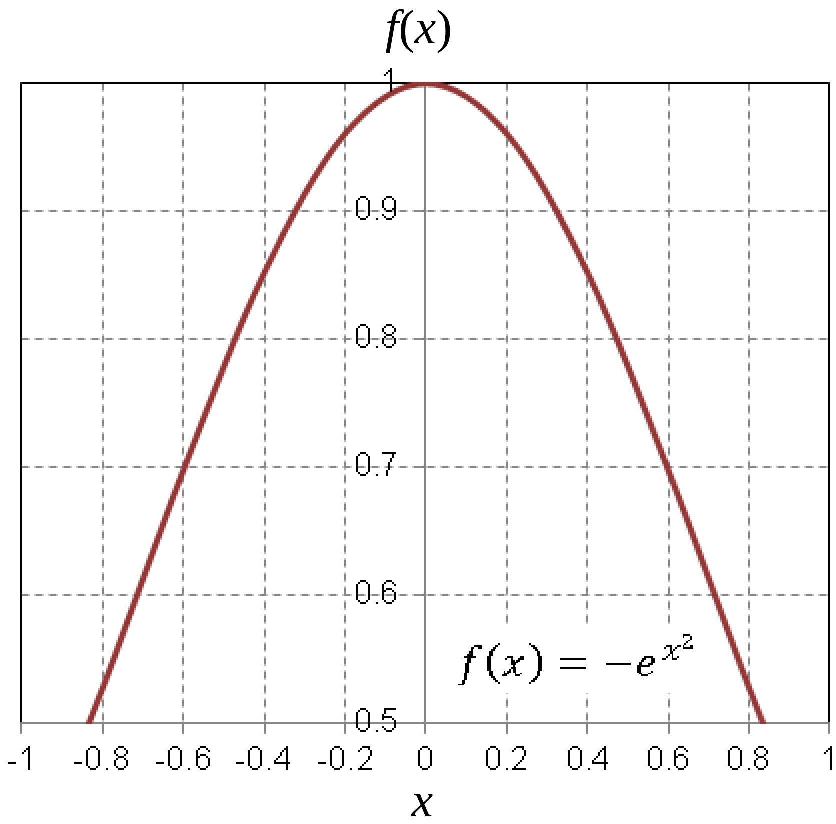
Performs numerical integration with the use of Romberg algorithm

```
function W = Romb(f, a, b, d)
%
% The function finds integral of f on the interval [a, b] using
% d steps of Romberg integration (or accelerated Simpson Rule)
%
T = zeros(d+1, d+1);
%
for k = 1 : d+1
    n = 2^k; T(1, k) = Simp(f, a, b, n);
end
%
for p = 1 : d
    q = 16^p;
    for k = 0 : d-p
        T(p+1, k+1) = (q*T(p, k+2) - T(p, k+1))/(q-1);
    end
end
%
for i = 1 : d+1
    table = T(i, 1 : d-i+2); disp(table)
end
%
W = T(d+1,1);
```

Romberg Integration – Example



Gaussian Quadratures – Example



Gaussian Quadratures – MATLAB Script

LIBRARY OF MATLAB PROCEDURES

`Gauss_quad`

Performs numerical integration with the use of Gaussian quadrature at 2 to 5 points

```
function I = Gauss_quad(f, a, b, k)
%
% The function finds integral of f on the interval [a, b] using
% Gaussian quadrature at k (k = 2, ..., 5) points
%
t = [-0.5773502692  -0.7745966692   -0.8611363116   -0.9061798459;
     0.5773502692    0.0             -0.3399810436   -0.5384693101;
     0.0             0.7745966692    0.3399810436    0.0;
     0.0             0.0             0.8611363116    0.5384693101;
     0.0             0.0             0.0             0.9061798459]
c = [1.0             0.5555555556    0.3478548451    0.2369268850;
     1.0             0.8888888889    0.6521451549    0.4786286705;
     0.0             0.5555555556    0.6521451549    0.5688888889;
     0.0             0.0             0.3478548451    0.4786286705;
     0.0             0.0             0.0             0.2369268850]
%
% Transformation of the interval of integration
x(1 : k) = 0.5*((b-a).*t(1:k,k-1) + b + a);
%
y = feval(f, x);
%
cc(1 : k) = c(1 : k, k-1);
cd = cc';
%
int = y*cd;
I = int*(b-a)/2
```

5/9 → 0.5555555556

8/9 → 0.8888888889

Gaussian Quadratures – Key Features

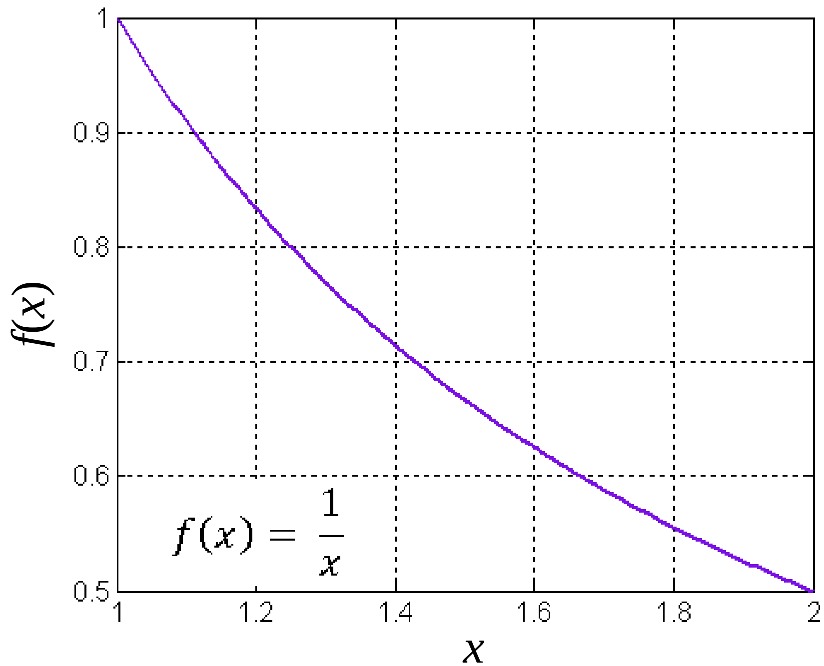
Evaluation of function takes place at ***specified points*** which we choose so that, for a given *n*, the rule would be exact for polynomials up to and including degree (2*n* - 1).

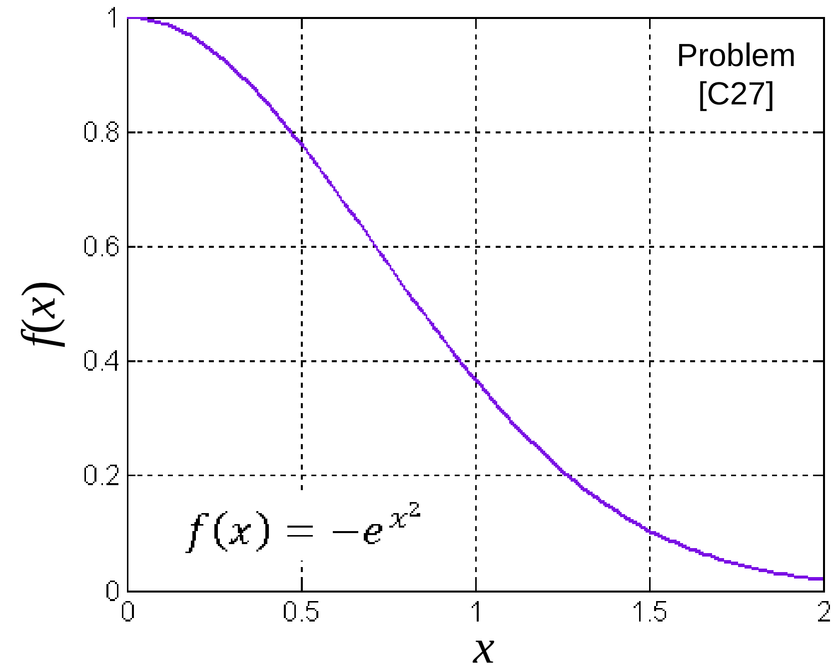
- Information about these points and corresponding coefficients are explicitly included in **`Gauss_quad`** – ***it is always there***, regardless the function, regardless the interval.
- These points between -1 and 1 are always on the ***fixed positions***, and those positions depend only on *n*, and either 2, or 3, or 4, or 5 points are used. (When integrating from a to b, conversion formulas are used.)
- **`Gauss_quad`** is a function in which integration can be performed only for *n* from 2 to 5.

Gaussian Quadratures – Nodes & Coefficients

n	Nodes x_i	Weights c_i	Truncation error
2	$-\sqrt{\frac{1}{3}}$	1	$\sim f^{(4)}(\alpha)$
	$+\sqrt{\frac{1}{3}}$	1	
3	$-\sqrt{\frac{3}{5}}$	$\frac{5}{9}$	
	0	$\frac{8}{9}$	$\sim f^{(6)}(\alpha)$
	$+\sqrt{\frac{3}{5}}$	$\frac{5}{9}$	
4	$-\sqrt{\frac{1}{7}\left(3-4\sqrt{0.3}\right)}$	$\frac{1}{2}+\frac{1}{12}\sqrt{\frac{10}{3}}$	
	$-\sqrt{\frac{1}{7}\left(3+4\sqrt{0.3}\right)}$	$\frac{1}{2}-\frac{1}{12}\sqrt{\frac{10}{3}}$	$\sim f^{(8)}(\alpha)$
	$+\sqrt{\frac{1}{7}\left(3-4\sqrt{0.3}\right)}$	$\frac{1}{2}+\frac{1}{12}\sqrt{\frac{10}{3}}$	
	$+\sqrt{\frac{1}{7}\left(3+4\sqrt{0.3}\right)}$	$\frac{1}{2}-\frac{1}{12}\sqrt{\frac{10}{3}}$	
5	$-\sqrt{\frac{1}{9}\left(5-2\sqrt{\frac{10}{7}}\right)}$	$0.3\left(\frac{-0.7+5\sqrt{0.7}}{-2+5\sqrt{0.7}}\right)$	
	$-\sqrt{\frac{1}{9}\left(5+2\sqrt{\frac{10}{7}}\right)}$	$0.3\left(\frac{0.7+5\sqrt{0.7}}{2+5\sqrt{0.7}}\right)$	
	0	$\frac{128}{225}$	$\sim f^{(10)}(\alpha)$
	$+\sqrt{\frac{1}{9}\left(5-2\sqrt{\frac{10}{7}}\right)}$	$0.3\left(\frac{-0.7+5\sqrt{0.7}}{-2+5\sqrt{0.7}}\right)$	
	$+\sqrt{\frac{1}{9}\left(5+2\sqrt{\frac{10}{7}}\right)}$	$0.3\left(\frac{0.7+5\sqrt{0.7}}{2+5\sqrt{0.7}}\right)$	

Gaussian Quadratures – Examples





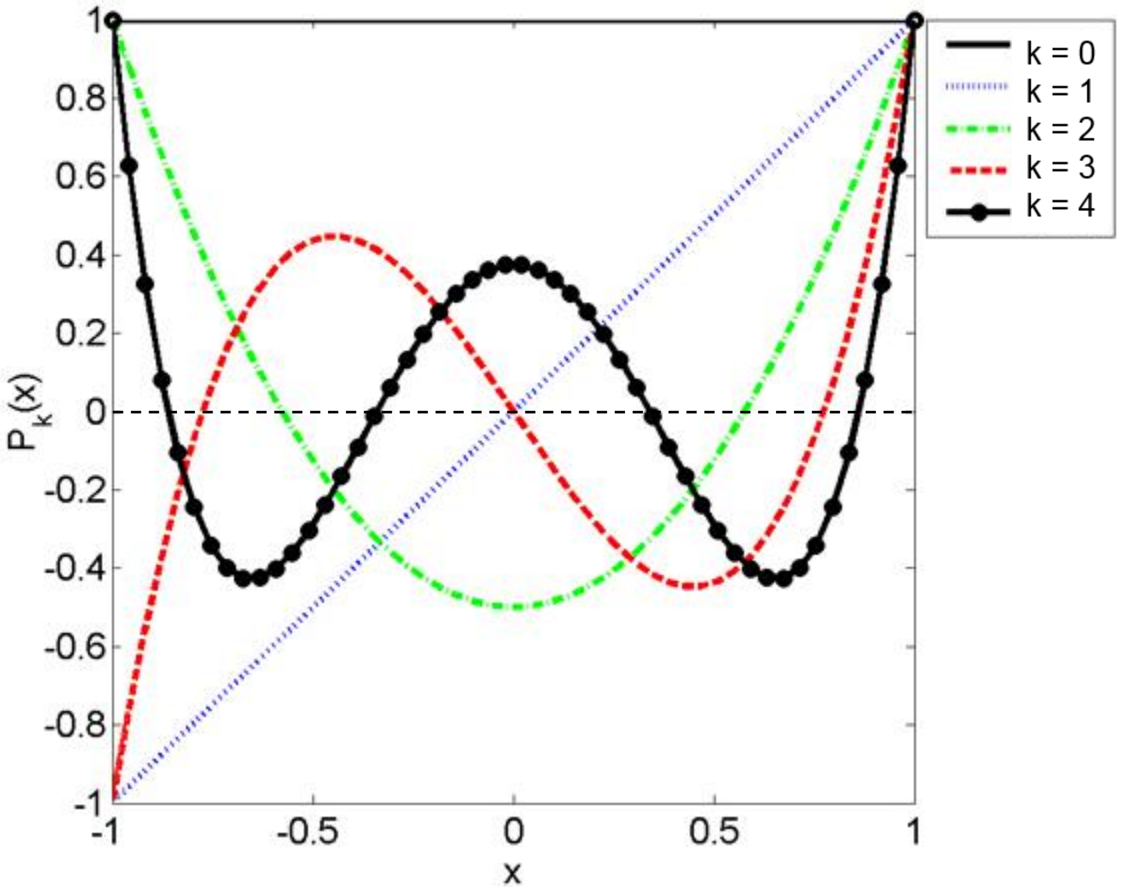
	k =	2	3	4	5	
$\int_1^2$	(1/x)dx =	0.6923	0.6931	0.6931	0.6931	=>

Exact value: ln2 ~ *0.693147*

Problem [C27]:

	k =	2	3	4	5		
$\int_0^2$	(exp(-x^2)dx =	0.9195	0.8789	0.8822	0.8821	=>	*0.8821*

First Five Legendre Polynomials



$P_0(x) = 1$

$P_1(x) = x$

$P_2(x) = x^2 – 1/3$

$P_3(x) = x^3 – (3/5)x$

$P_4(x) = x^4 – (6/7)x^2 + 3/35$

$P_5(x) = x^5 – (10/9)x^3 + 5/21x$

Gaussian Quadratures – Errors

Accuracy of integration with Gaussian Quadratures – a subject of advanced courses; in a very condensed form:

- For Gaussian Quadratures, the error term is not a simple function of step *h*; it does, but ***it is function of not only h***.
- The error in Gaussian Quadratures goes to 0 more rapidly for particular integrands – it gives much better accuracy than other techniques ***for more smooth functions***!
  - ***Composite Trapezoid and Simpson Rules converge as $O(h^2)$ and $O(h^4)$ regardless of the smoothness of f(x)!***

Gaussian Quadratures – Observations

Overall, Gaussian Quadratures

…is a ***special technique*** due to its basic feature – ***an approximation based on the exact values for polynomials***.

…can be successfully used in manual computations, *as a part of analytical manipulations with integrals*, and

…can be used in combinations with other techniques of Num. Int.

…if used separately, **it works *very well with smooth functions*** – and that is ***not a rare situation in applications***, BTW!

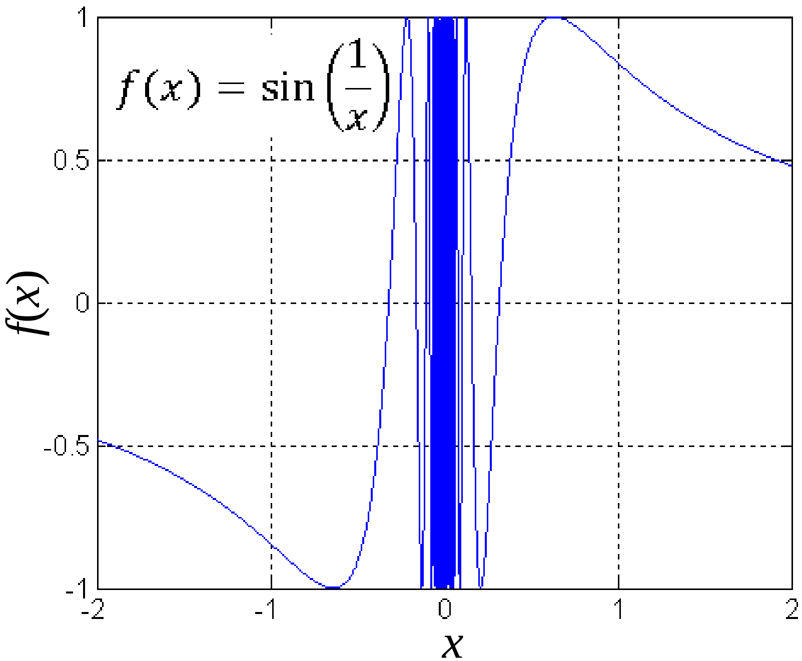
Difficulties in Numerical Integration

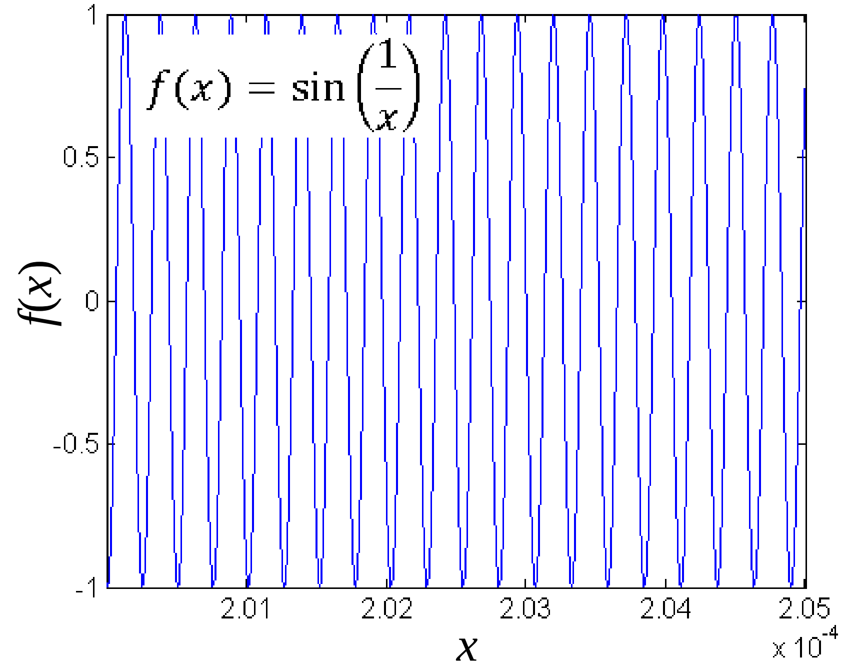
Difficulty	Treatment
a. The function is continuous in the range of integration, but its *derivatives are discontinuous or singular**);	a. The discontinuity/singularity should be *located*, and the integral split into a sum of two/more integrals whose ranges avoid the discontinuities.
b. The *function is discontinuous* in the range of integration;	b. The same as a.
c. The function has *singularities* in the range of integration;	c. Different approaches; e.g., a change of variables, integration by parts, splitting the integral, etc.
d. The *range of integration is infinite*.	d. A method suitable for an infinite range of integration, e.g., Gauss-Laguerre and Gauss-Hermite formulas.

*) Since the derivatives of polynomials are continuous, *polynomials cannot accurately represent functions with discontinuous derivatives*.

Difficulties in Numerical Integration

Example of a Difficult Function



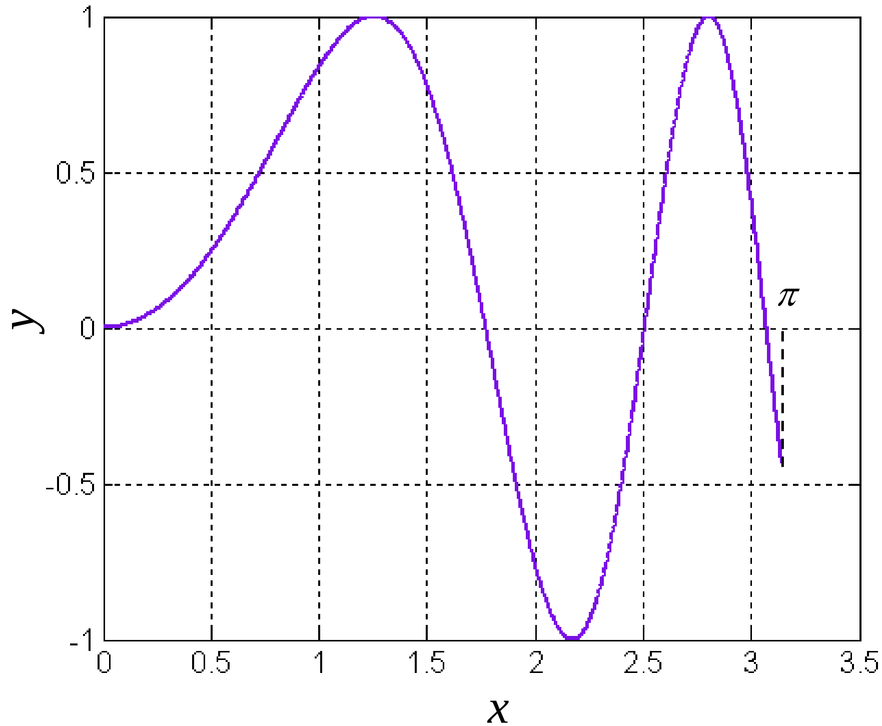


- ***Rapid change of the function for small changes in the independent variable***!
    - Check for the built-in MATLAB function **`quad8`**

Test Functions for Methods of Numerical Integration

$$y = \sin(x^2) \quad (1)$$

$[0, \pi]$



0.7727...

	n = 8	16	24	32	40	96	128	216
Trap	0.6903	0.7539	0.7645	0.7681	0.7697	0.7721	0.7724	0.7727
Simp	0.8368	0.7751	0.7731	0.7728	0.7727			

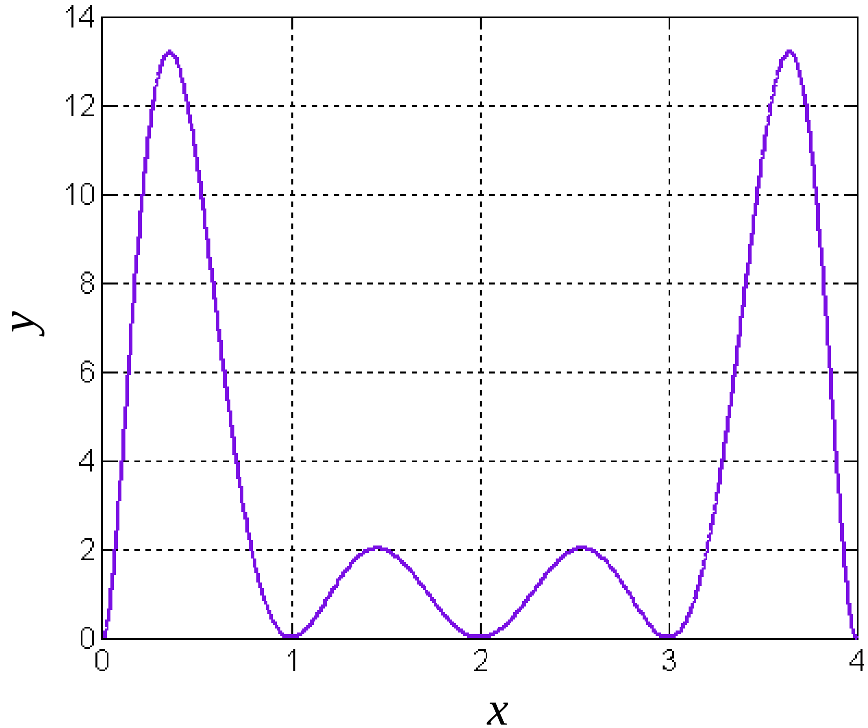
	d = 2	3	4	5
Romb	0.8878	0.7705	0.7726	0.7727

	k = 2	3	4	5
G q	0.4444	1.8504	0.3635	0.7857

Test Functions for Methods of Numerical Integration

$$y = x^2(x-1)^2(x-2)^2(x-3)^2(x-4)^2 \quad (2)$$

[0, 4]



14.7763...

	n = 8	32	64	96	120	144	232
Trap	12.7441	14.7667	14.7757	14.7762	14.7763		
Simp	16.9922	14.8130	14.7787	14.7768	14.7765	14.7764	14.7763

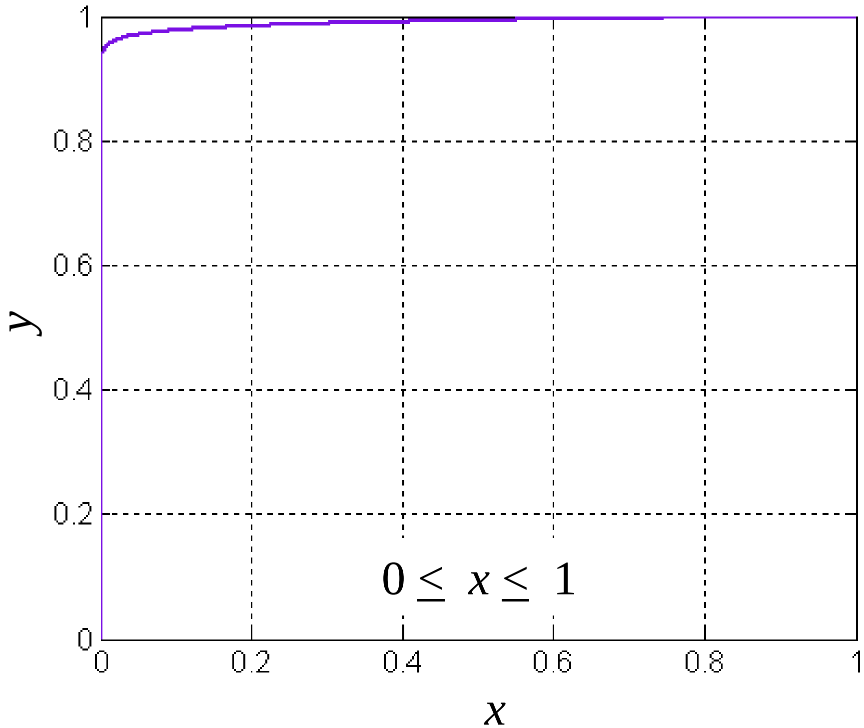
	d = 2	3	4	5	6
Romb	18.1961	15.1273	14.7820	14.7764	14.7763

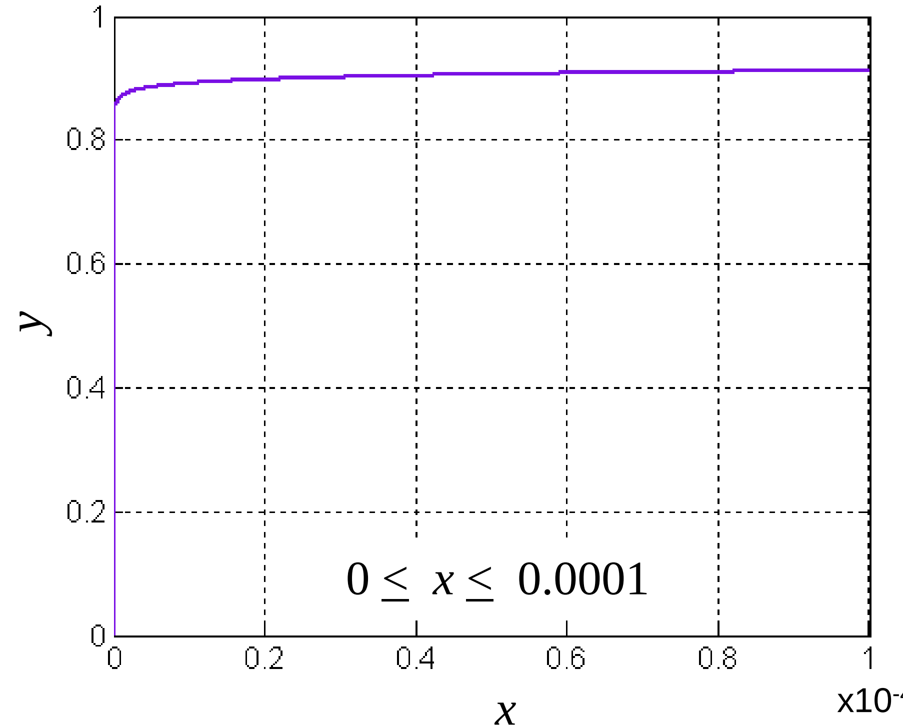
	k = 2	3	4	5
G q	4.2140	26.7605	21.4154	8.7720 (*fails* – needs more points!)

Test Functions for Methods of Numerical Integration

$y = x^{0.01}$ (3)

[0, 1]





	n = 32	64	196	388	1164	2000
Trap	0.9753	0.9827	0.9877	0.9889	0.9897	0.9899
Simp	0.9803	0.9852	0.9885	0.9893	0.9898	0.9899

	d = 2	4	6	8	10
Romb	0.9531	0.9810	0.9879	0.9895	0.9900

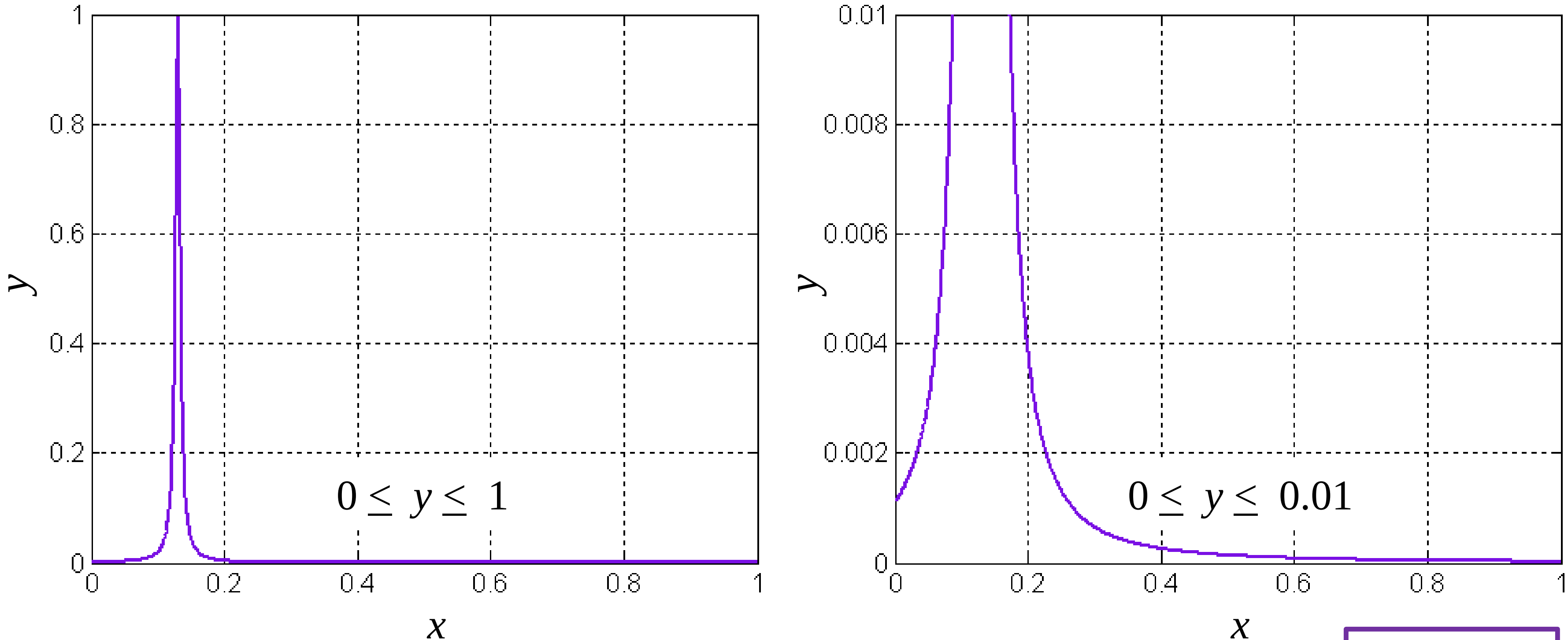
	k = 2	3	4	5
G_q	0.9911	0.9906	0.9904	0.9903

0.9901...

Test Functions for Methods of Numerical Integration

$$y = \frac{1}{1+(230x-30)^2} \qquad (4)$$

$[0, 1]$



0.0135...

	n = 24	50	100	200	300	400
Trap	0.0075	0.0080	0.0153	0.0136	0.0135	
Simp	0.0092	0.0081	0.0178	0.0130	0.0136	0.0135

	d = 2	3	4	5	6	7
Romb	0.0699	0.0138	0.0099	0.0094	0.0144	0.0135

	k = 2	3	4	5
G_q	0.0015	0.0158	0.0011	0.0008

(*fails* – needs more points!)

Numerical Integration – Comparison of Performance

Some Observations

- **`Gauss_quad`** gives best result for very coarse approximations (***of smooth functions!***), is *very quick* – computation happens momentarily [e.g., **`Romb`** for d = 10 took 2½ min (a machine with Duo 1.8 GHz processor)]. It fails with (2), (4) – due to the unlucky choice of the points (nodes).
- **`Simp`** and **`Trap`** perform practically identically with (3): don't need too many points for the result of *not high* accuracy, but ***converge slowly for the a high accurate result***. **`Trap`** may be better for oscillating functions like in (2) - better approximated by straight lines.
- **`Romb`** seems to be strong procedure, but it requires more computational time than other techniques.