



DeepL

Subscribe to DeepL Pro to translate larger documents.

Visit [www.DeepL.com/pro](https://www.DeepL.com/pro) for more information.

University of Applied Sciences Nordhausen

# Thesis Numerical Mathematics

submitted on 30.11.2021

submission closing on

1.12.2021

The original version is in German. The English translation was generated with the translation program DeepL.

Arwed Paul Meinert

## Content

Introduction .....	3
Choice of software .....	3
Project 1: Interpolation .....	3
Project 2: Numer. Differentiation .....	6
Project 3: Numerical integration .....	8
Project 4: Bisection process .....	9
Project 5: Fixed point iteration .....	11
Project 6: Newton method .....	13
Project 7: Differential Equation .....	15
Program Listing .....	19
Used own functions: .....	19
Core routines used: .....	22
Bibliography .....	22
Sources .....	22
Auxiliary means .....	22
Bibliography .....	22
Declaration of independence .....	24

## Introduction

In the subject "Engineering Mathematics IV - Introduction Numerics" with Dr. Gebel we have learned some of the basic numerical algorithms in the summer semester 2021. For the certificate of achievement we should independently implement the algorithms discussed in the lectures and prepare the results. The choice of the software was up to us.

## Choice of software

Regarding the software I have chosen "GNU Octave" in version 6.2.0 (Windows) or version 5.2.0 (Linux) was chosen. The main reasons were that the syntax of Octave is in most cases identical to that of MatLab, which we learned in the lecture "Engineering Mathematics IV - MatLab" last semester. Unlike MatLab, however, Octave is open source (GLP- license) and funded by donations<sup>1</sup>. This makes the reproducibility of my results possible for everyone, without the need for a license or a special operating system.

## Project 1: Interpolation

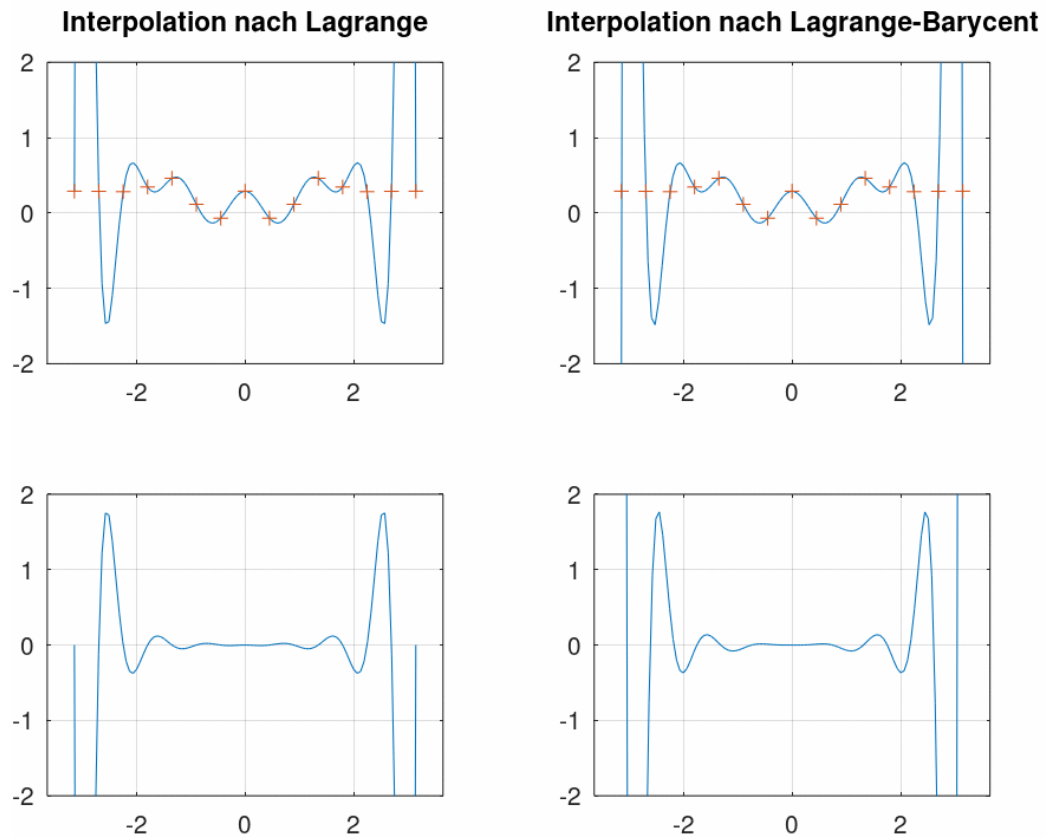
It is the function:  $f(x) = \frac{5 \cos(2x)}{\sqrt{1+x^2}} - x e^{-x} 2 \sin(3x)$  Given. It is supposed to be in the interval

$-\pi \leq x \leq \pi$ . For this purpose, the interpolation polynomial with  $n=15$  nodes is to be calculated. The Lagrangian algorithm and the barycentric algorithm presented in the lectures are used.<sup>2</sup> To make the main program clearer, the function values of the interpolated functions were calculated in two separate functions. In the main program the parameters are defined and the functions are called. Then both interpolation curves and the original function are displayed. The difference of the original function to the interpolation function is plotted under the respective representations. If this value is zero, both functions are identical. Per larger the value, the greater the error between the interpolation curve and the "correct" Value.

---

<sup>1</sup> <https://www.gnu.org/software/octave/about>

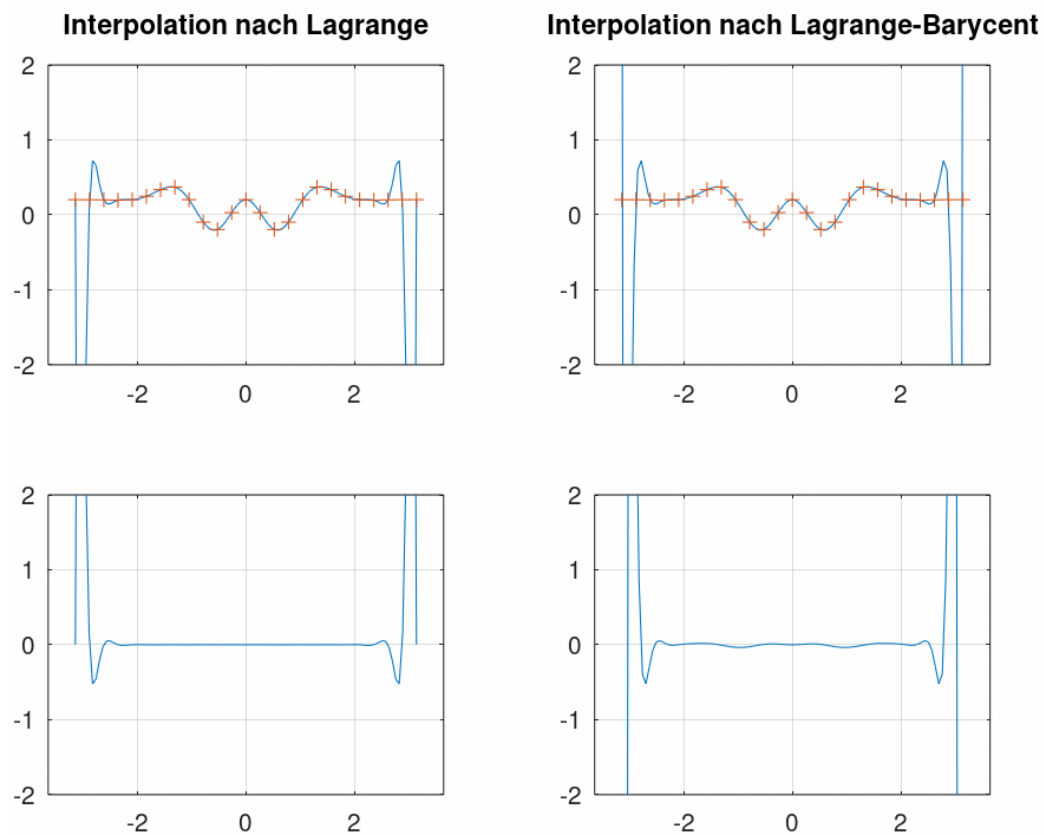
<sup>2</sup> Cf. Bronstein, Taschenbuch der Mathematik, p.941



In the picture on the left the interpolation according to Lagrange is shown. The orange points are the exact function values at the 15 node points in the interval. In blue the interpolated function is shown. Below the plot of the function is an error estimate. It was calculated by subtracting the y-values of the function (in this case with 100 intermediate steps) from those of the interpolation. It can be seen that, as is usual with interpolation using this method, the errors at the edges of the interval are very large. However, the error away from the interval boundaries is small. To estimate the accuracy and for a better comparison of the two interpolations, the average error can be calculated (program 1.1 line 37f.). This is 1.15. However, the value is essentially so large because the interpolation at the edges is very inaccurate.

The two plots on the right show the interpolation according to the barycent algorithm and the error of the interpolation to the "correct" value. Here, too, enormous errors occur at the interval boundaries. However, the average error is only 0.46 and is thus more than half as large as with the Lagrange interpolation.

To get a more accurate interpolation of the original function, the number of nodes can be increased. For this purpose the value of the variable  $n$  is increased in program 1.1 line 3. If you now set it to  $n=25$ , the following picture results:



Here the representation is analogous to the interpolation with  $n=15$ . In orange you see the function values of the original function, in blue the interpolated function. Below the two representations of the function is again the error estimation.

It can be seen that the function with more nodal points clearly delivers more accurate results further to the edges. Nevertheless, the extreme swings of the interpolated function still occur. The average error of the calculation with Lagrange is 0.36. With Barycent it is 3.2. The large average error, which comes about here with the Barycent algorithm, can be explained by the fact that the function produces a very large error at the interval boundaries.

It is possible to interpolate  $n$ -many points with a polynomial of degree  $n-1$ . In reality, however, it is often more decisive whether a function of significantly lower degree approximates the given measured values well without necessarily going through the measured values themselves.

Furthermore, it is a possibility not to interpolate the whole polynomial, but to interpolate only the parts between two points with a 3rd degree polynomial. This kind of interpolation is called spline interpolation and gives a good result even at the interval boundaries.<sup>3</sup>

<sup>3</sup> Cf. F.Y. Cheng, F. Zizhi; Computational mechanics in structural engineering: recent developments and future trends. S. 78

## Project 2: Numer. Differentiation

It is the function  $f(x) = (x^3 + 2 * x^2 - 3 * x + 4) * \cos(2 * x)$  In the interval  $-4\pi \leq x \leq 4\pi$  is given. By numerical differentiation now all local extrema and all Determine inflection points with an accuracy of  $10^{-3}$  decimal places.

To numerically form the derivative of a function, the slope between two points is calculated. The closer the points are between which a slope is calculated, the greater the accuracy.<sup>4</sup> To achieve the required accuracy, the two points with which the secant slope is calculated must therefore be no more than 0.001 apart.

The number of nodes then required is calculated using the following formula:

$$n = \left\lceil \frac{|-4\pi - 4\pi|}{0,001} \right\rceil = 25133$$

It is important that the result has an integer number of nodes. It is therefore rounded up to the nearest integer.

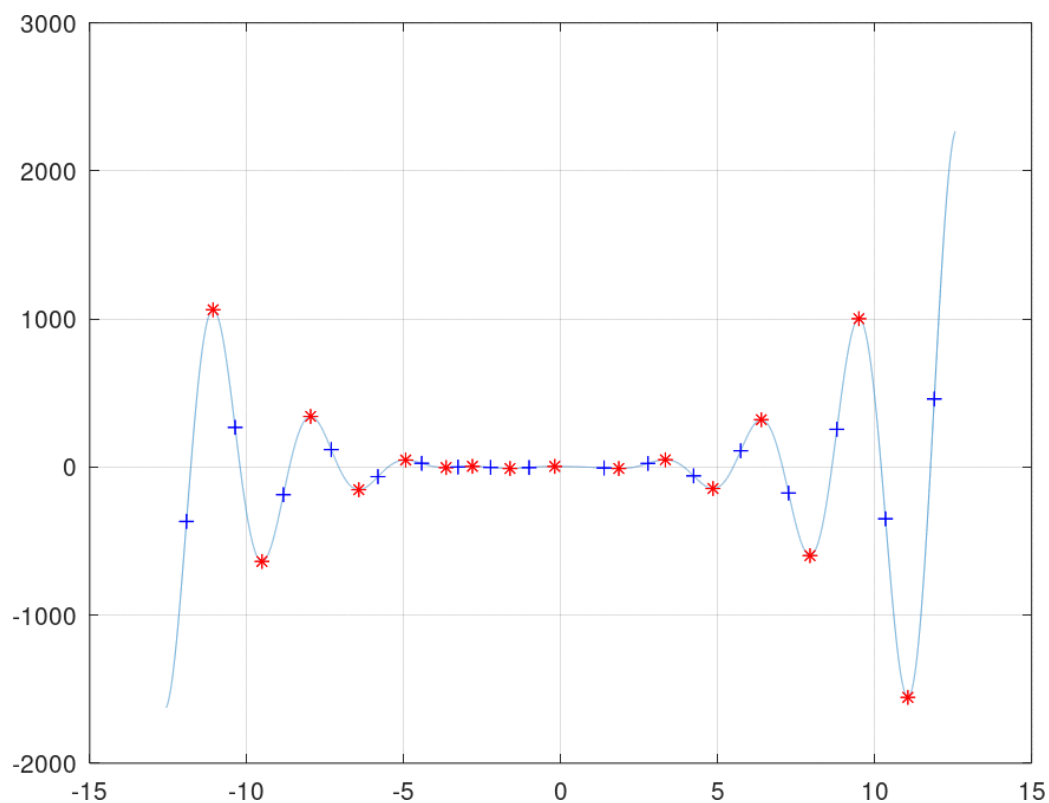
The calculation of the derivative is again done by a function (see "derivative"). Because later also the inflection points of the function are asked for, one can form the second derivative of the function by forming the derivative of the first derivative. By calling the function "extr" a vector is created in which the extrema are marked with two logical ones. The extrema then lies in the interval, which belongs to the respective x-values.

For the points of inflection, the necessary condition is that the second derivative at the x-value is zero. One has the first derivative at the respective x-values already to the computed, in order to determine the extrema. If one now the vector of the y-values of the derivative into the derivative function (see "derivative"), you get the y-values of the second derivative at the respective x-values. If you now pass this vector to the function with which you previously calculated the extreme values (see "extr"), you will again be returned a vector in which the interval in which an inflection point is located is delimited with two logical ones.

One can now represent the function numerically. To represent the extrema and inflection points, the respective vector is searched for ones and they are represented as points on the graph:

---

<sup>4</sup> Cf. Collatz, Numerical Methods of Approximation Theory, p.237.



In the diagram, the extremes are shown as red stars and the turning points as blue crosses. To determine the number of extrema and inflection points, you can now sum the vectors "extrema" and "inflection points" ("Project 2" line 38f). Although they are logical ones, they can still be summed. But because each turning point and each extremum is bounded by two ones (the exact value lies somewhere in between), the sum must be divided by two at the end. The number of extrema in the interval is 16 and the number of inflection points is 17. To determine the local minimum and maximum, the y-values of the individual extrema are compared with the current minimum/maximum. If it is larger or smaller, the value is overwritten. For the local minimum we get  $\text{Min} = (11,06/-1555,365)$ . For the local maximum  $\text{Max} = (-11,069/1062,393)$ .

## Project 3: Numerical Integration

It is necessary to determine numerically the definite integral  $\int_0^{\pi} 4 - 3 \cos(4x) dx$ . For this purpose, on the one hand the Trapezoidal method is used. Two values for  $f(x)$  are calculated which are  $\Delta h$  apart. An interpolation polynomial of the 1st degree is determined between the points and the area is calculated. This is repeated with the step size  $\Delta h$  until the end of the interval is reached. The approximated area then corresponds to the sum of all partial areas. The smaller you choose  $\Delta h$ , the more accurate the result.<sup>5</sup>

On the other hand, the Simpson method can be used. As with the trapezoidal method, the area under the curve is calculated with segments of length  $2h$ , but a 2nd degree polynomial is used here.<sup>6</sup>

Again, both methods were implemented as functions and called in the main program. To have a comparison value of the results, the determined integral is calculated with the function contained in Octave.

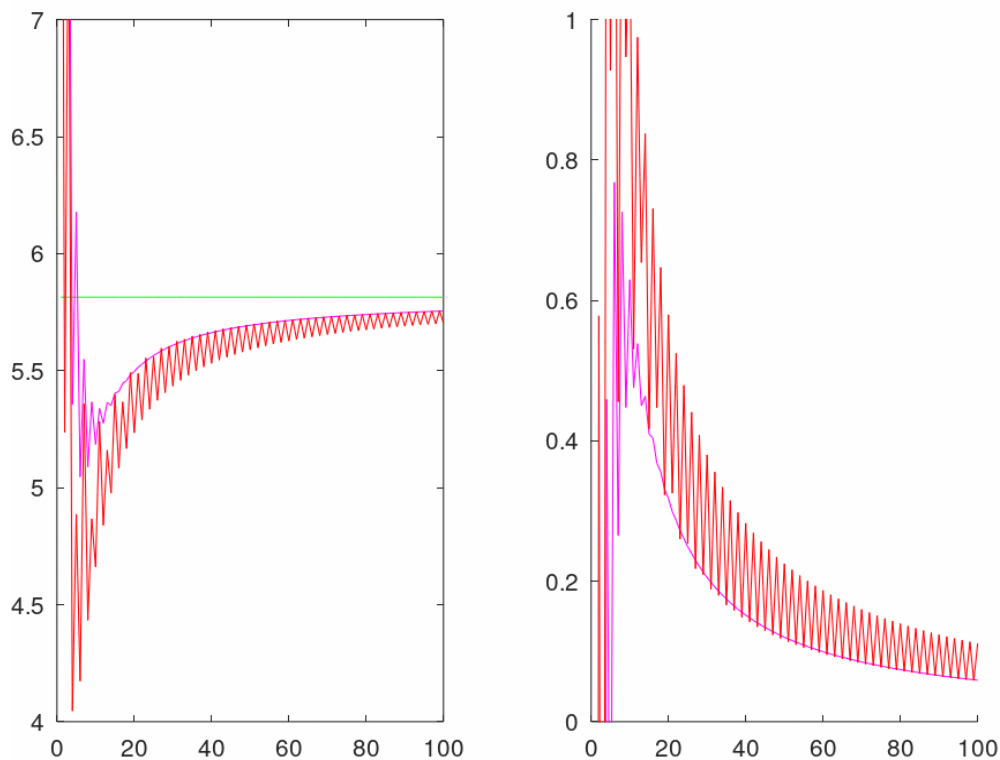
n	Area trapezoid	Error trapezoid	Area Simpson	Error Simpson	Octave surface
10	5,184611	0,629458	4,661012	1,153057	5,814070
15	5,404454	0,409615	5,396930	0,417139	5,814070
20	5,496227	0,317842	5,234427	0,579642	5,814070
50	5,693715	0,120354	5,588996	0,225074	5,814070
100	5,754930	0,059140	5,702570	0,111500	5,814070
1000	5,808246	0,00582	5,803010	0,011059	5,814070

<sup>5</sup> Cf. Bronschein, Taschenbuch der Mathematik, p. 922.

<sup>6</sup> Cf. Bronschein, Taschenbuch der Mathematik, p. 923.



It is also possible to display the convergence behavior and the error graphically:



The left plot shows the calculated area. The magenta colored line represents the trapezoidal method. The red line represents the Simpson method. On the right, you can see the error that results when the numerical values of the respective method and the respective value of the integral are subtracted from the "correct" value of the iteration step.

It can be seen that the error for both methods becomes smaller as the number of iterations increases. However, it is also noticeable that with the Simpson method only every even  $n$ -value is better than the trapezoidal formula. This is due to the fact that Simpson's method only provides meaningful values for even  $n$  values.<sup>7</sup>

When choosing  $n$ , it should be noted that very small  $n$  values give completely wrong results.

With a large  $n$ , the trapezoidal method provides the more accurate result (see table).

## Project 4: Bisection process

Given the function  $f(x) = 3 * \arctan(x) * e^x - 2$ . One is to find the zero that lies in the interval  $0 \leq x \leq 1$ . For this purpose, the bisection method is to be used.

In the bisection method, the midpoint of the interval is determined (in this case 0.5). You now check whether the  $y$ -value is the upper interval limit or the value of the lower interval limit.

<sup>7</sup> Cf. Bronstein, Pocketbook of Mathematics, p.923

is negative. Subsequently, the negative value is replaced with the newly calculated value of the middle of the interval. This function is called recursively and thus includes the zero point more and more closely.<sup>8</sup>

The bisection procedure was implemented via the function "bisection" (function 4.1). The function is passed as a function handle, the interval limits and the number of iteration steps.

The task requires an accuracy of  $10^{-7}$  decimal places. With the bisection method, the iteration number can be estimated with the following function:

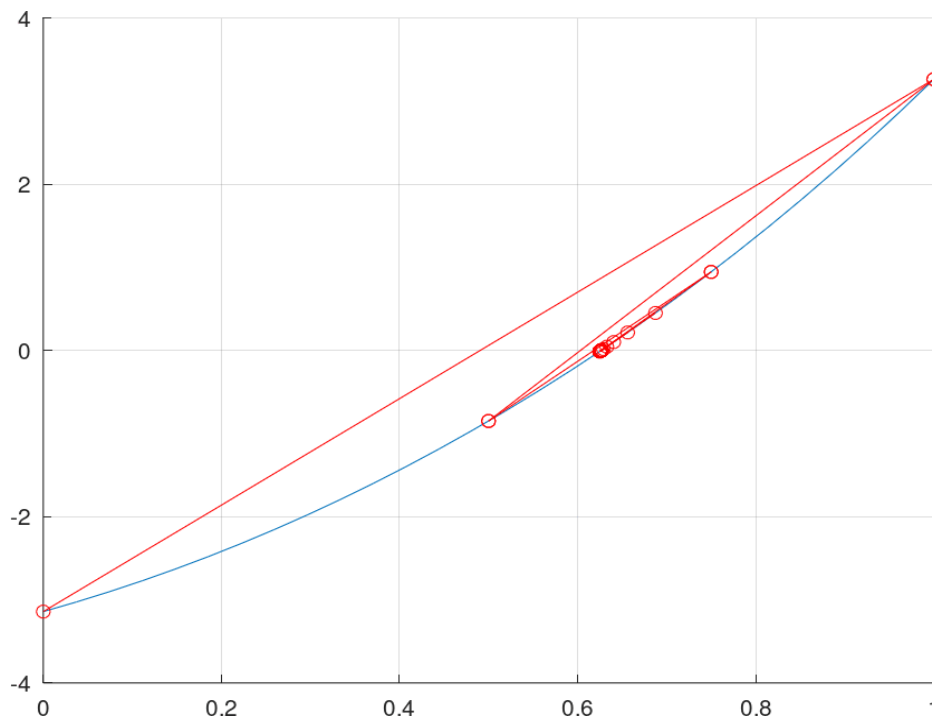
$$n = \frac{t + 1 + \log(|b - a|)}{\log(2)} + 1$$

Here, one sets the required valid digits for t (7), the upper limit of the interval for b, and the lower limit of the interval for a.

$$n = \frac{7 + 1 + \log(|1 - 0|)}{\log(2)} + 1 = 27,575$$

Because you can only make whole iteration steps, the result must be rounded up to the next whole number. So in this case there are  $n=28$  iteration steps.

If you plot the function values and the iteration steps, you get the following graph:



One recognizes well the convergence, which results from this procedure. The result for the zero by the numerical method is  $N = (0.626505527/1.36487 \cdot 10^{-8})$ .

Because this is a numerical procedure, the y-value is not exactly zero. However, if the number of iteration steps were increased, it would approach zero. Because it is numerical

---

<sup>8</sup> M.Jung, U.Langer, Finite Element Method for Engineers: An Introduction to Numerical Principles and Computer Simulation, Cf. p246.

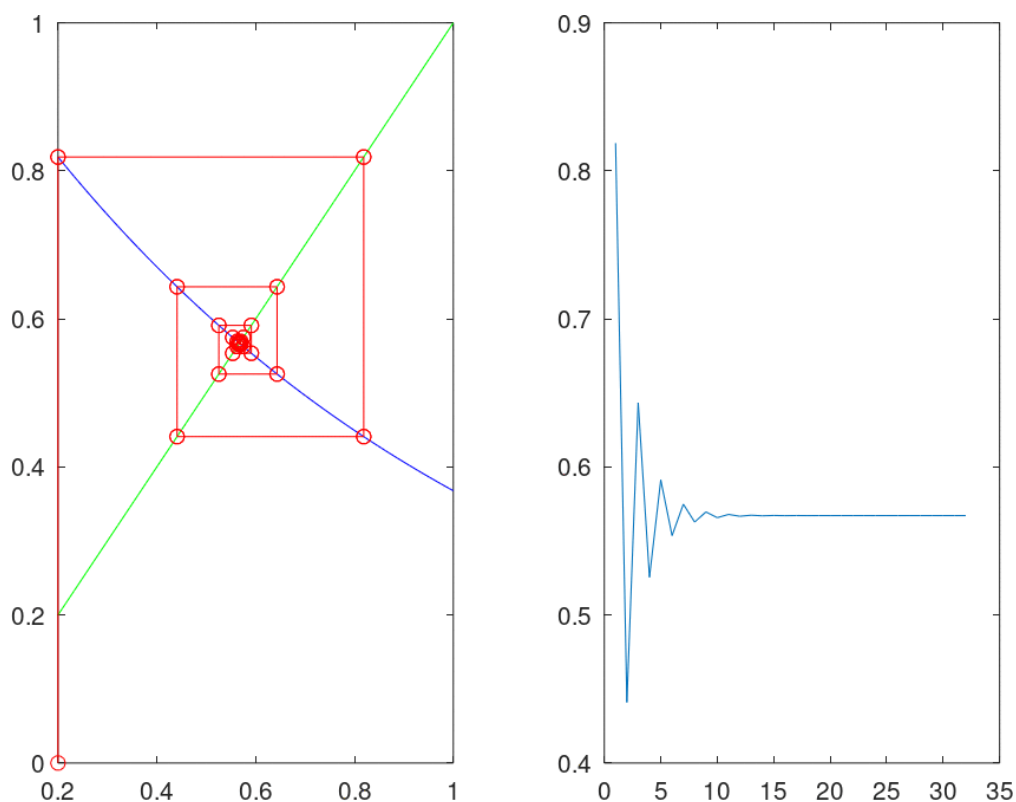
however, there is no exact zero, the lowest value that can be achieved has an accuracy of  $10^{-15}$ .

## Project 5: Fixed point iteration

The fixed point of the function  $f(x) = e^{-x}$  in the interval  $0 \leq x \leq 1$  is to be determined. A fixed point is a point of a function, which, if you put it back into the function, has itself as a result.

The fixed point iteration (function 5.1) and the derivative function (function 5.2) are called in the main program. The contraction number  $q$  is calculated and the fixed point iteration is performed until the desired accuracy of  $10^{-8}$  is reached.<sup>9</sup>

With the starting value  $x_0=0.2$ , iteration must be performed 32 times and  $q$  is 0.82.



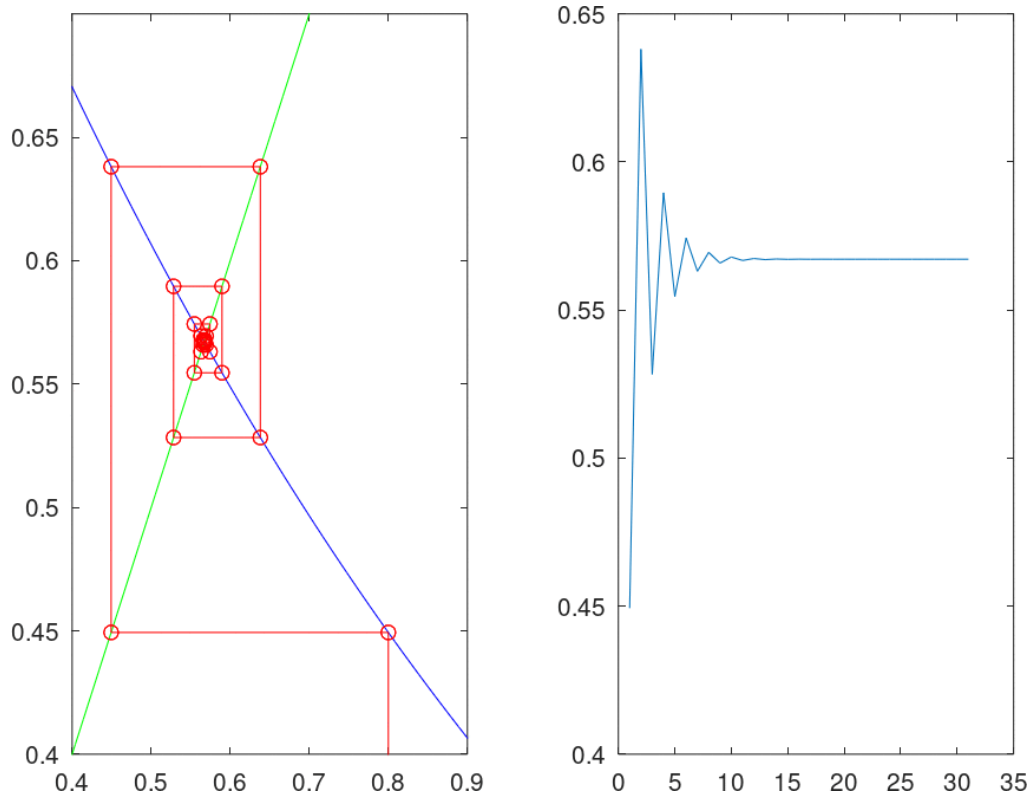
<sup>9</sup> Cf. Appell, J., Vöth, M., Elemente der Funktionalanalysis: Vektorräume, Operatoren und Fixpunktsätze, p.226.

On the left side you can see in blue the function  $f(x)=e^{-x}$ , in green the function  $g(x)=x$  is plotted. It is easy to see that the program starts at  $x_0=0.2$  and converges quickly. The convergence behavior is shown on the right. The fixed point of the function is 0.56714328.

As a rehearsal, you can insert the result of the fixed point into the function:

$$f(0.56714328) = e^{-0.56714328} = 0.56714328$$

If one chooses  $x_0=0.8$  for the start value, 31 iteration steps are necessary.



The result is the same regardless of the starting value. However, it can be seen that the starting value has an influence on the type of convergence.

The divergence is similar for both starting values.

The fixed point iteration does not give an unambiguous result if  $q \geq 1$ . Then two results can occur or even the so-called "deterministic chaos" in which the function values can be calculated deterministically, but otherwise they do not follow any obvious rule.

## Project 6: Newton method

Let us find all real roots, i.e. all real zeros of the polynomial

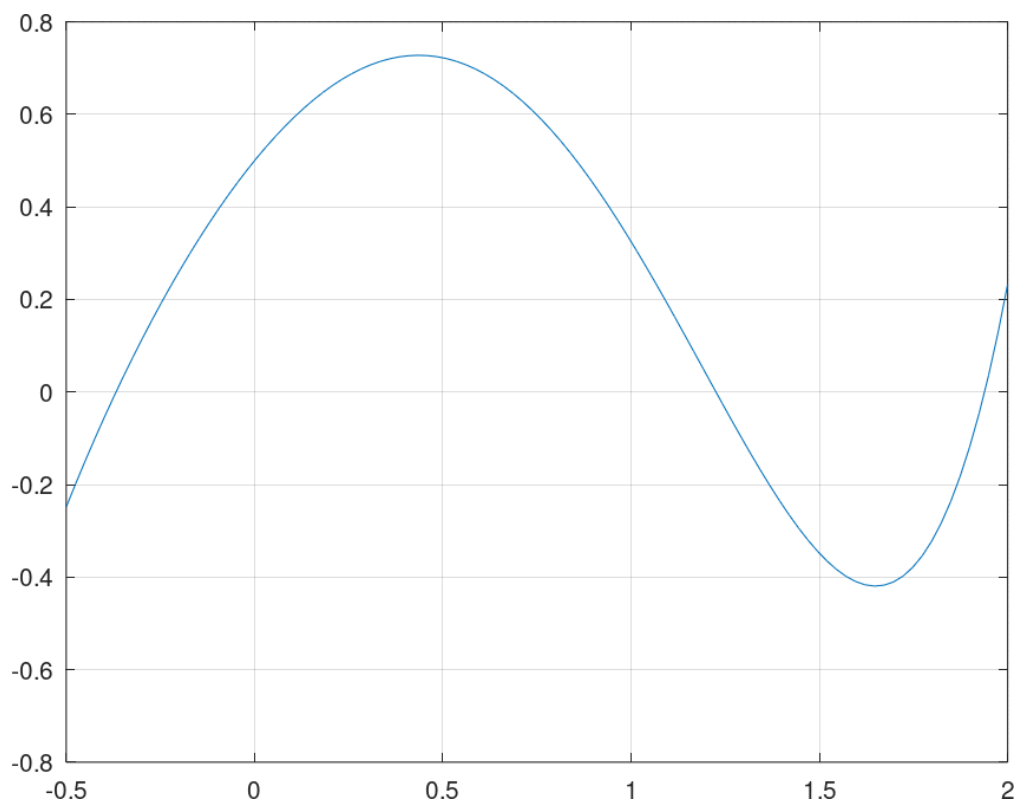
$$P(x) = \frac{1}{5}x^5 - \frac{5}{24}x^4 + \frac{1}{6}x^3 - x^2 + x + \frac{1}{2}$$

amount. For the analysis of the function the Horner scheme (see "hornernewton") can be used. This provides for a polynomial P at the position  $x_0$  both the function value and the values for the first to nth derivative.<sup>10</sup> To use the Newton method, however, only the function values and the values of the first and second derivative are needed. Therefore the function returns only these three values.

The Newton method is an iterative method with the calculation formula

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

However, this means that you need an approximate value of the zero. For this it is sufficient to visualize the function.<sup>11</sup>



<sup>10</sup> Cf. Bronstein, Pocketbook of Mathematics, p. 910.

<sup>11</sup> Cf. Bronstein, Pocketbook of Mathematics, p. 908.

You can see that there are three zeros:

$x_0 \approx -0,2$

$x_1 \approx 1,3$   $x_2 \approx 2$

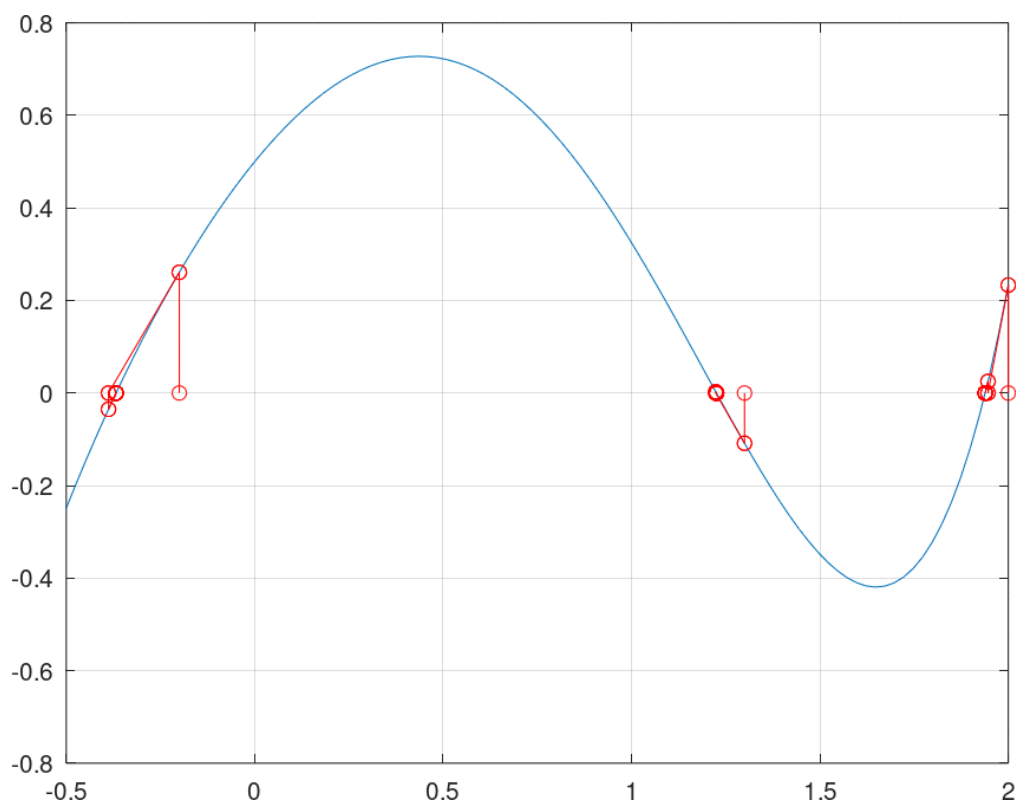
These values are still very inaccurate, but they are suitable as starting values for the Newton method. One begins with the definition of the start values  $x_0$  to  $x_2$ . Then the number of iteration steps is set to  $n=1$ . Now the exact values are compared with the last iteration step of the Newton method. If the difference in one of the zeros determined by Newton's method is greater than the required accuracy of  $10^{-5}$ ,  $n$  is incremented and new zeros are calculated.

After four iteration steps the following values result:

	Newton method	Exact value
$x_0$	-0.367833049084739	-0.367833009259087
$x_1$	1.225382475298674	1.225382475298699
$x_2$	1.938662477820235	1.938662436526359

After only four iteration steps, the values already lie with the required accuracy at the numerically exact values calculated by Octave

This can also be seen if you draw the convergence steps in the diagram:



The convergence behavior is shown here in red.

## Project 7: Differential equation

In the problem, the initial value problem  $y' = -2xy$  with  $y(0) = 1$  in the interval  $0 \leq x \leq 1$  is to be solved numerically and compared with the exact analytical solution.

In order to be able to compare the numerical results with the exact solution, the exact solution must first be found. This is best done by separating the variables:

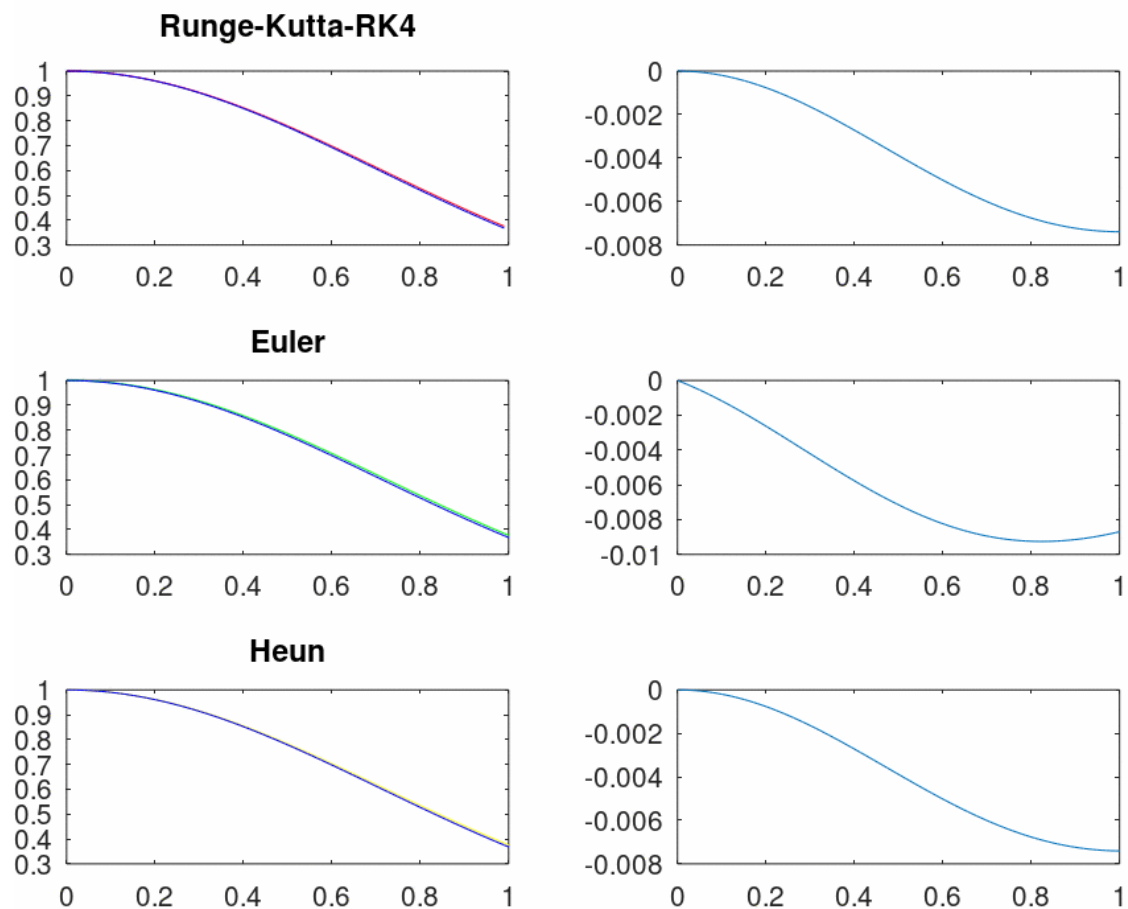
$$\begin{aligned}
 y' &= -2xy \\
 \frac{y'}{y} &= -2x \\
 \frac{1}{y} dy &= -2x * dx \\
 \int \frac{1}{y} dy &= -2 * \left[ \frac{1}{2} * x^2 \right]_0^x \\
 \ln(|y|) &= -2 * \frac{1}{2} * x^2 \\
 \ln(|y|) &= -x^2 \\
 y &= e^{-x^2}
 \end{aligned}$$

I used three numerical methods to solve initial value problems. All of them were implemented in functions to simplify the calling from the main program. The three methods are the "Runge-Kutta method" (function 7.1), "Heun method" (function 7.2) and the "Euler polygonal traction method" (function 7.3).<sup>12</sup>

<sup>12</sup> Cf. Bronstein, Taschenbuch der Mathematik, pp. 928ff.



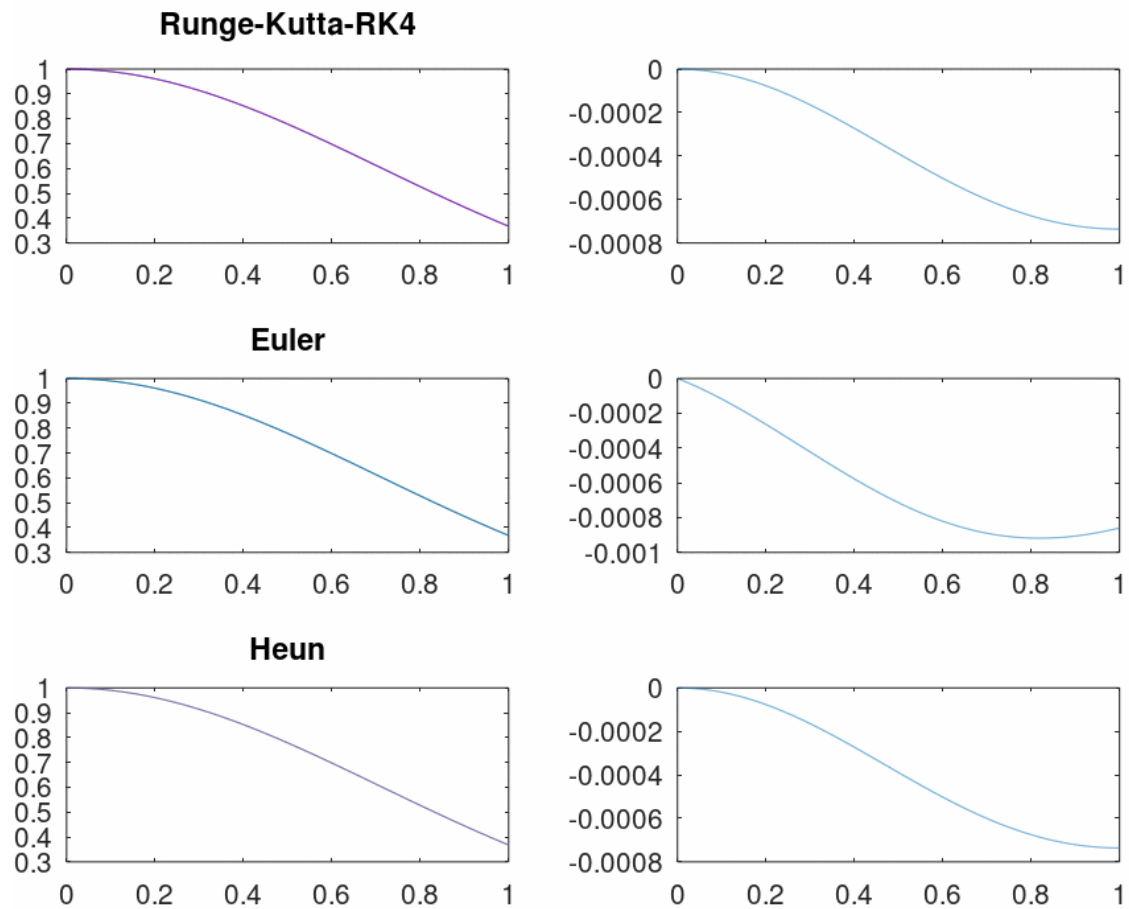
In the following, all three methods are shown with a step size of  $h=0.01$ :



The analytical result is shown in blue, the other color represents the calculation algorithm. On the right side, the value is shown when the respective numerical result is subtracted from the analytical result.

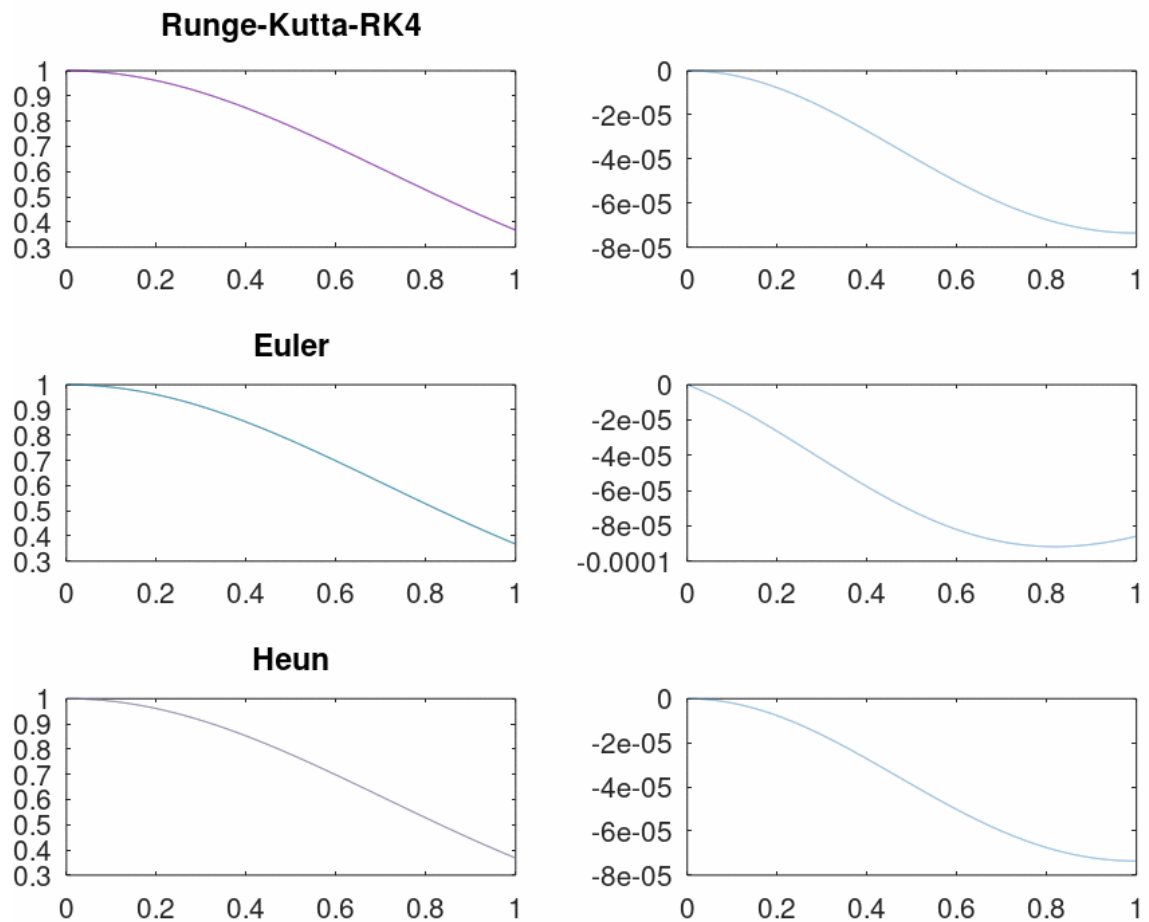
One recognizes only very small deviations if one compares all three numerical procedures with one another. Only if one forms the difference (right side), one recognizes small differences. All three methods become less accurate towards the end. The Euler method is the most inaccurate. The course of the RK4 values and the method according to Heun is almost identical, although the value of the RK4 method is calculated with 4 correction factors.

If we now increase the step size to  $h=0.001$ , leaving all other settings the same, we get the following picture:



All representations and the colors are shown here as above. Again, the curves of the analytical solution and those of the numerical methods are optically almost on top of each other. This is also shown by the representation of the difference of the analytical and numerical values on the right side. The courses of the three curves are analogous to those of the step size  $h=0.01$ , but the errors are smaller by a factor of 0.1.

If we now use  $h=0.0001$ , the following picture appears:



You can see here that the progressions of the two curves are visually on top of each other. The errors have also become smaller again. The error of the RK4 and the Heun method are in the order of  $-8 \times 10^{-5}$ , the error of Euler in the range  $-1 \times 10^{-4}$ .

To estimate which of the three methods gave the most accurate value, I calculated the arithmetic mean of the errors shown on the right side. For the value  $h=0.01$  are the average errors of the three methods:

Euler:	$-6.079567167813153 \times 10^{-3}$
Heun:	$-3.792663257025819 \times 10^{-3}$
RK4:	$-3.789508878062853 \times 10^{-3}$

If  $h=0.0001$  is used, the following values are obtained:

Euler:	$-6.069350010741614 \times 10^{-5}$
Heun:	$-3.789480327794961 \times 10^{-5}$
RK4:	$-3.789446923657294 \times 10^{-5}$

This confirms what could already be seen in the representation of the errors - the error in the Heun method is greatest for the same h-value. The error in the use of the method according to Heun and according to RK4 differs only slightly. Nevertheless, the method according to RK4 provides the more accurate solution (the average error is closest to zero).

## Program- Listing

### Used own functions:

Overview:

Functions	Description
barycentMatr(x,xW,yW,grad)	gives the barycent matrix of the grade of the Point pairs $P_n(xW/yW)$ back
Lagrange(x,xW,yW,grad)	gives the interpolation according to Lagrange of the grade "grad" of the point pairs $P_n(xW/yW)$ back
LagrangeBarycent(x,xW,yW,grad)	gives the LagrangeBarycent interpolation of the grade "grad" of the point pairs $P_n(xW/yW)$ back
extr(f,n,h)	returns a vector, in which intervals are marked with a logical 1, in which the slope of the 1st derivative has a different sign with a number of n values resp. an interval size of h
derivative(y,n,h)	returns the values of the derivative of the y' values passed with n many points and the Step size h
trapezoidal formula(f,gu,go,n)	gives the approximate value of the area of the determined integral f in the limits gu and go with a number of nodes of n with the trapezoidal formula
simpson(f,gu,go,n)	gives the approximate value of the area of the determined integral f in the limits gu and go with a node number of n with the Simpson formula at
bisection(f,a,b,n)	gives the x- and y-vectors of the function f of the zero point with n iteration steps back
fixpoint(f,x0,n)	specifies the respective fixed point as a vector of the function f with the starting point x0 and n iteration steps backwards
hornernewton(a1,z)	gives the values of the 0. to 2. derivative of the polynomial a1 at the position z back
newton(p,x0,n)	gives the value of the derivative of the polynomial p at the reset x0 after n iteration steps

$\text{RK4}(f, x_0, y_0, h, n)$	gives the X and Y values of the Diff. Equation $f$ with $x_0$ and $y_0$ , a step size of $h$ and $n$ many iteration steps back with the RK4 method
$\text{heun}(f, x_0, y_0, h, n)$	gives the X and Y values of the Diff. Equation $f$ with $x_0$ and $y_0$ , a step size of $h$ and $n$ many iteration steps back with the Heun method

euler(f,x0,y0,h,n)	gives the X and Y values of the Diff. Equation f with x0 and y0, a step size of h and n many iteration steps back with Euler's method
--------------------	---

### Used core routines:

abs(a)	returns the amount of a
ceil(a)	rounds to the nearest natural number from a to
disp(s)	prints the variable s in the editor window
integral(f,a,b)	calculates the numerical value of the determined integral of f in the limits a and b
linspace(a,b,n)	returns a vector with n many uniformly distributed values between a and b
max(v)	returns the largest value from the vector v
sum(v)	returns the sum of the vector v

## Bibliography

### Sources

- Scripts of the subject Engineering Mathematics IV; Introduction Numerics
- Script of the subject Engineering Mathematics IV; MatLab

### Resources

- All graphics and tables were created with Octave version 6.2.0
- For verification of the results a Casio fx-991DEX was used

### Bibliography

Pocketbook of mathematics: by I.N. Bronstein and K.A. Semendyaev. (2001). Germany: Harri Deutsch.

Computational Mechanics in Structural Engineering: Recent Developments and Future Trends. (2003). United Kingdom: CRC Press.

Collatz. (2012). Numerical Methods of Approximation Theory, Vol.6 \ Numerische Methoden Der Approximationstheorie, Volume 6: Workshop on Numerical Methods of Approximation Theory Oberwolfach, January 18-24, 1981 \ Tagung Über Numerische Methoden Der Approximationstheorie Oberwolfach, 18-24.Januar 1981. Switzerland: Springer Basel AG.

Langer, U., Jung, M. (2012). Finite element method for engineers: an introduction to numerical principles and computer simulation. Germany: Springer Fachmedien Wiesbaden.

Appell, J., V  th, M. (2005). Elements of functional analysis: vector spaces, operators and fixed point theorems. Germany: Vieweg+Teubner Verlag.

## Declaration of independence

### Selbstständigkeitserklärung

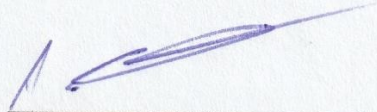
Hiermit erkläre ich, dass ich die vorliegende Arbeit mit dem Titel

Belegarbeit Numerische Mathematik

selbstständig und ohne unerlaubte fremde Hilfe angefertigt, keine anderen als die angegebenen Quellen und Hilfsmittel verwendet und die den verwendeten Quellen und Hilfsmitteln wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Nordhausen, 30. 11. 2021

Ort, Datum



Unterschrift