RiemannZetaFct_and_RiemannHypothesis

May 21, 2021

1 Riemann's Zeta-Function and Riemann's Hypothesis

Powered by: Dr. Hermann Völlinger, DHBW Stuttgart(Germany); May 2021

Prereq.'s: you need to extract the zip-file 'Images.zip' in a directory with name 'Images'

1.1 Item1: Riemann's Zeta-Function

See: https://en.wikipedia.org/wiki/Riemann_zeta_function

For a non-expert introduction about the background and history of the Riemman's Zeta Fct & Riemann's Hypothesis (in German language, Christmas Lecture 2016 of HAW Hamburg) see the following YouTube video: https://www.youtube.com/watch?v=sZhl6PyTflw&vl=en

The Riemann zeta function or Euler–Riemann zeta function, zeta(s), is a function of a complex variable s that analytically continues the sum of the Dirichlet serie which converges when the real part of s is greater than 1.

More general representations of zeta(s) for all s are given below. The Riemann zeta function plays a pivotal role in analytic number theory and has applications in physics, probability theory, and applied statistics. As a function of a real variable, Leonhard Euler first introduced and studied it in the first half of the eighteenth century without using complex analysis, which was not available at the time. Bernhard Riemann's 1859 article "On the Number of Primes Less Than a Given Magnitude" extended the Euler definition to a complex variable, proved its meromorphic continuation and functional equation, and established a relation between its zeros and the distribution of prime numbers.[2]

The values of the Riemann zeta function at even positive integers were computed by Euler. The first of them, zeta(2), provides a solution to the Basel problem. In 1979 Roger Apéry proved the irrationality of zeta(3). The values at negative integer points, also found by Euler, are rational numbers and play an important role in the theory of modular forms. Many generalizations of the Riemann zeta function, such as Dirichlet series, Dirichlet L-functions and L-functions, are known.

See also the following youtube video explaining in mathematical details of the Riemann's Zeta Fct. zetas(s), s = complex number and also the Riemann's Hypothesis: https://youtu.be/sD0NjbwqlYw

1.1.1 Item1.1: Dirichlet Series of Zeta-Function

https://en.wikipedia.org/wiki/Dirichlet_series In mathematics, a Dirichlet series is any series of the form of the following picture (see below).
The Dirichlet series of Riemann's Zeta-Function is a complex sequence. It is a special case of general Dirichlet series. Dirichlet series play a variety of important roles in analytic number theory. The

most usually seen definition of the Riemann zeta function is a Dirichlet series, as are the Dirichlet L-functions.

[1]:
```
print("** DirichletForm of the Riemann Zeta-Fuction (Euler-Function)**")
#print("** LATEX syntax zeta(s) for re(s)>1: $ \zeta(s)=\sum_{n=1}^\infty 1/n^s␣
↪$ **")

from IPython.display import Image

Image('Images/DirichletForm4Riem-ZetaFct.jpg')
```

1.1.2 Item1.2: The Basel Problem

The Basel problem is a problem in mathematical analysis with relevance to number theory, first posed by Pietro Mengoli in 1650 and solved by Leonhard Euler in 1734,[1] and read on 5 December 1735 in The Saint Petersburg Academy of Sciences.[2] Since the problem had withstood the attacks of the leading mathematicians of the day, Euler's solution brought him immediate fame when he was twenty-eight. Euler generalised the problem considerably, and his ideas were taken up years later by Bernhard Riemann in his seminal 1859 paper "On the Number of Primes Less Than a Given Magnitude", in which he defined his zeta function and proved its basic properties. The problem is named after Basel, hometown of Euler as well as of the Bernoulli family who unsuccessfully attacked the problem.

The Basel problem asks for the precise summation of the reciprocals of the squares of the natural numbers, i.e. the precise sum of the infinite series:

The sum of the series is approximately equal to 1.644934.[3] The Basel problem asks for the exact sum of this series (in closed form), as well as a proof that this sum is correct. Euler found the exact sum to be $pi^2/6$ and announced this discovery in 1735. His arguments were based on manipulations that were not justified at the time, although he was later proven correct. He produced a truly rigorous proof in 1741.

[2]:
```
print("Consider the special case s = 2 + i*0, so we get the follw. series:")
print("** This is the famous 'Basel-Problem' solved by L. Euler in 1735 **")

from IPython.display import Image

Image('Images/Basel_Problem.jpg')
```

1.1.3 Item1.3: Riemann's Zeta Fct for Complex Numbers

If you extend the Dirichlet series on the whole complex plane, Riemann found a nice formula, which we call complex Riemann Zeta-Fct. (cRZ). The following picture show the formula.

[3]:
```
print("** This is the famous cRZ formula from Berhard Riemann **")

from IPython.display import Image

Image('Images/complex_RiemannZeta-Formula.jpg')
```

** This is the famous cRZ formula from Berhard Riemann **

[3]:

$$\zeta(s) = \frac{1}{1-2^{1-s}} \sum_{n=0}^{\infty} \frac{1}{2^{n+1}} \sum_{k=0}^{n} (-1)^k \binom{n}{k} (k+1)^{-s}$$

1.1.4 Item1.4: Euler Product Formula

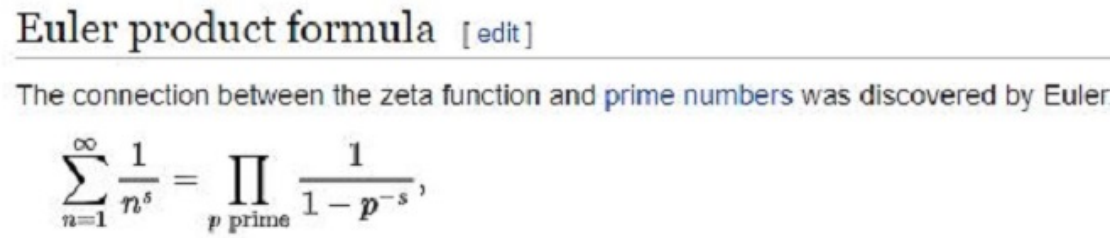
https://en.wikipedia.org/wiki/Proof_of_the_Euler_product_formula_for_the_Riemann_zeta_function

Leonhard Euler proved the Euler product formula for the Riemann zeta function in his thesis Variae observationes circa series infinitas (Various Observations about Infinite Series), published by St Petersburg Academy in 1737.[1][2]

[4]:
```
print ("**********************************************************************")
print ("** The bridge between zeta-fct in 'Complex Analysis' and prim- **")
print ("** numbers in 'Number Theory' is given by EulerProduct formula **")
print ("**********************************************************************")

from IPython.display import Image

Image('Images/EulerProduct.jpg')
```

```
**********************************************************************
** The bridge between zeta-fct in 'Complex Analysis' and prim- **
** numbers in 'Number Theory' is given by EulerProduct formula **
**********************************************************************
```

[4]:



1.2 Item2: Riemann's Hypothesis

See: https://en.wikipedia.org/wiki/Riemann_hypothesis

In mathematics, the Riemann hypothesis is a conjecture that the Riemann zeta function has its zeros only at the negative even integers and complex numbers

with real part = 1/2. Many consider it to be the most important unsolved problem in pure mathematics.[1] It is of great interest in number theory because it implies results about the distribution of prime numbers. It was proposed by Bernhard Riemann (1859), after whom it is named. The Riemann hypothesis and some of its generalizations, along with Goldbach's conjecture and the twin prime conjecture, comprise Hilbert's eighth problem in David Hilbert's list of 23 unsolved problems; it is also one of the Clay Mathematics Institute's Millennium Prize Problems. The name is also used for some closely related analogues, such as the Riemann hypothesis for curves over finite fields.

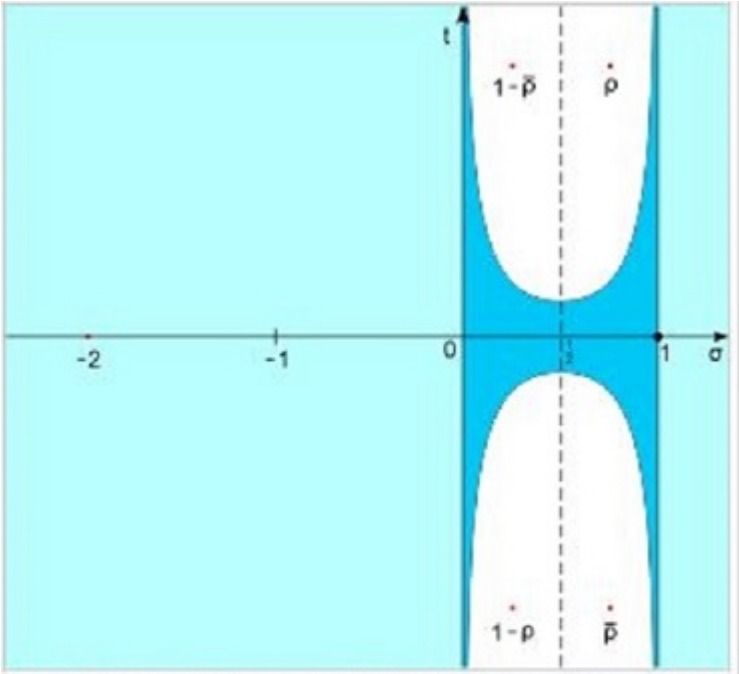
1.2.1 Item2.1: Zero-free region of Zeta-Function

Apart from the trivial zeros, the Riemann zeta function has no zeros to the right of = 1 and to the left of = 0 (neither can the zeros lie too close to those lines). Furthermore, the non-trivial zeros are symmetric about the real axis and the line = 1/2 and, according to the Riemann hypothesis, they all lie on the line = 1/2.

[5]:
```
print ("*** Zero-free_region_for_the_Riemann_zeta-function**** ")
from IPython.display import Image

Image('Images/Zero-free_region_for_the_Riemann_zeta-function.jpg')
```

```
*** Zero-free_region_for_the_Riemann_zeta-function****
```

[5]:



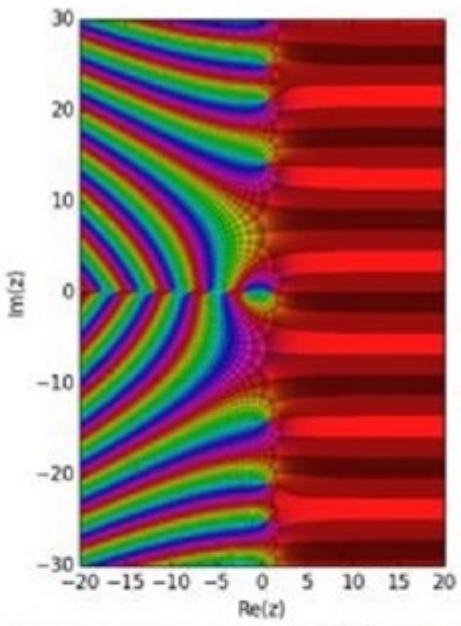
Apart from the trivial zeros, the Riemann zeta function has no zeros to the right of $\sigma = 1$ and to the left of $\sigma = 0$ (neither can the zeros lie too close to those lines). Furthermore, the non-trivial zeros are symmetric about the real axis and the line $\sigma = \frac{1}{2}$ and, according to the Riemann hypothesis, they all lie on the line $\sigma = \frac{1}{2}$.

[6]:
```
print (" ************************************************************ ")
print (" *** Here is an example-plot of the riemann zeta-function *** ")
print (" **** See non-trival zeros at 'critical' line real(z)=0.5 *** ")
print (" **** This is a visualization of the Riemann-Hypothesis ***** ")
print (" ************************************************************ ")

from IPython.display import Image

Image('Images/riemann-zeta1.jpg')
```

```
************************************************************
*** Here is an example-plot of the riemann zeta-function ***
**** See non-trival zeros at 'critical' line real(z)=0.5 ***
**** This is a visualization of the Riemann-Hypothesis *****
************************************************************
```

[6]:

Riemann zeta function



The Riemann zeta function $\zeta(z)$ plotted with domain coloring.[1]

Basic features	
Domain	$\mathbb{C} \setminus \{1\}$
Codomain	$\mathbb{C}$

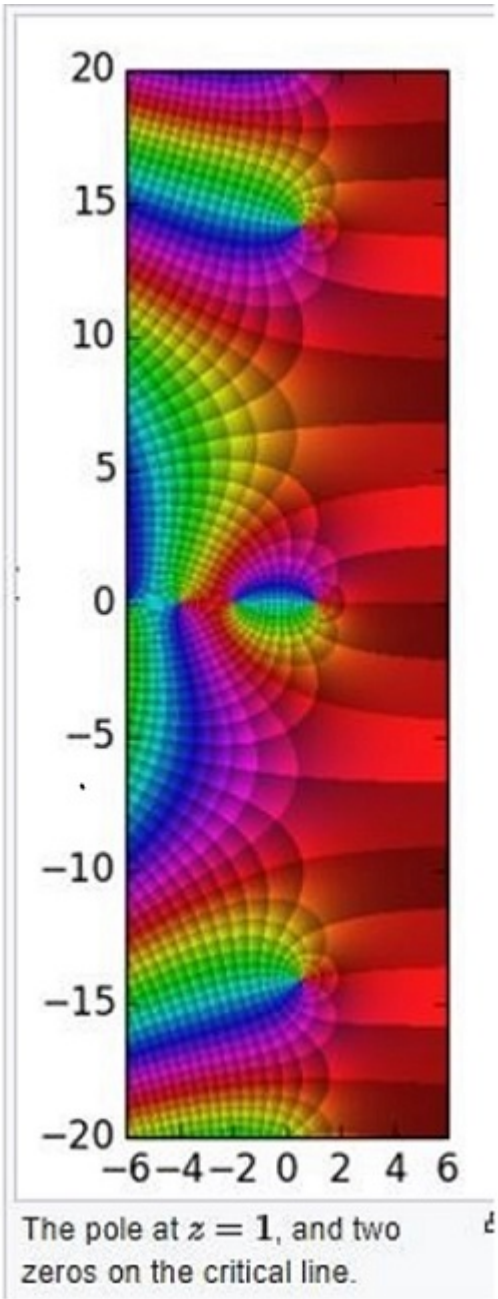
Specific values	
At zero	$-\frac{1}{2}$
Limit to $+\infty$	1
Value at 2	$\frac{\pi^2}{6}$
Value at -1	$-\frac{1}{12}$
Value at -2	0

[7]:
```
print ("************************************************************")
print ("** Here is an example-plot of zeta function in more detail **")
print ("** See two zeros at the points z=0.5 + 14,12 & z=0.5-14,12 **")
```

```
print ("****************************************************************")

from IPython.display import Image

Image('Images/riemann-zeta2.jpg')
```

```
****************************************************************
** Here is an example-plot of zeta function in more detail **
** See two zeros at the points z=0.5 + 14,12 & z=0.5-14,12 **
****************************************************************
```

[7]:



The pole at $z = 1$, and two zeros on the critical line.

1.2.2 Item 2.2: Main Program Code for calculation of Zeta(s), s=complex number

This is the Program/Source Code (in Python) using complex the Riemann's Zeta-Fct (cRZ) for complex numbers (see above). Rounding errors may occur when we are using the cRZ formula, because this is an approximation method for the determination of zeta(s). The parameter t is defining the granularity of the approximation. Choosing smaller t, i.e. t=50 the rounding error will become smaller. The program will be executed later in the JN, when we calculate zeta(s), where s are special real numbers.

[8]:
```
# Import libaries

from itertools import count, islice
from scipy.special import binom

# Program/Source Code (Python)using complex Riemann's Zeta-Fct (cRZ) for␣
 ↪complex Numbers
# It is using the complex RiemannZeta formula (cRZ); see above

print ("** Rounding errors may occur through the calculation of zeta(s,t) ***")
print ("** Choosing a smaller value t,i.e. t=50 the error will get smaller **")

def zeta(s, t = 100):
    if s == 1:
        return float("inf")
    term = (1 / 2 ** (n + 1)
              * sum((-1) ** k * binom(n, k) * (k +1 ) ** -s
                     for k in range (n + 1))
           for n in count(0))
    return sum(islice(term, t)) / (1 - 2 ** (1- s))

print ("** End of Definition of main function zeta(s,t) using formula (cRZ) **")
```

```
** Rounding errors may occur through the calculation of zeta(s,t) ***
** Choosing a smaller value t,i.e. t=50 the error will get smaller **
** End of Definition of main function zeta(s,t) using formula (cRZ) **
```

1.2.3 Item2.3: 3-dim. Plot of Riemann Zeta Fct. for complex plane

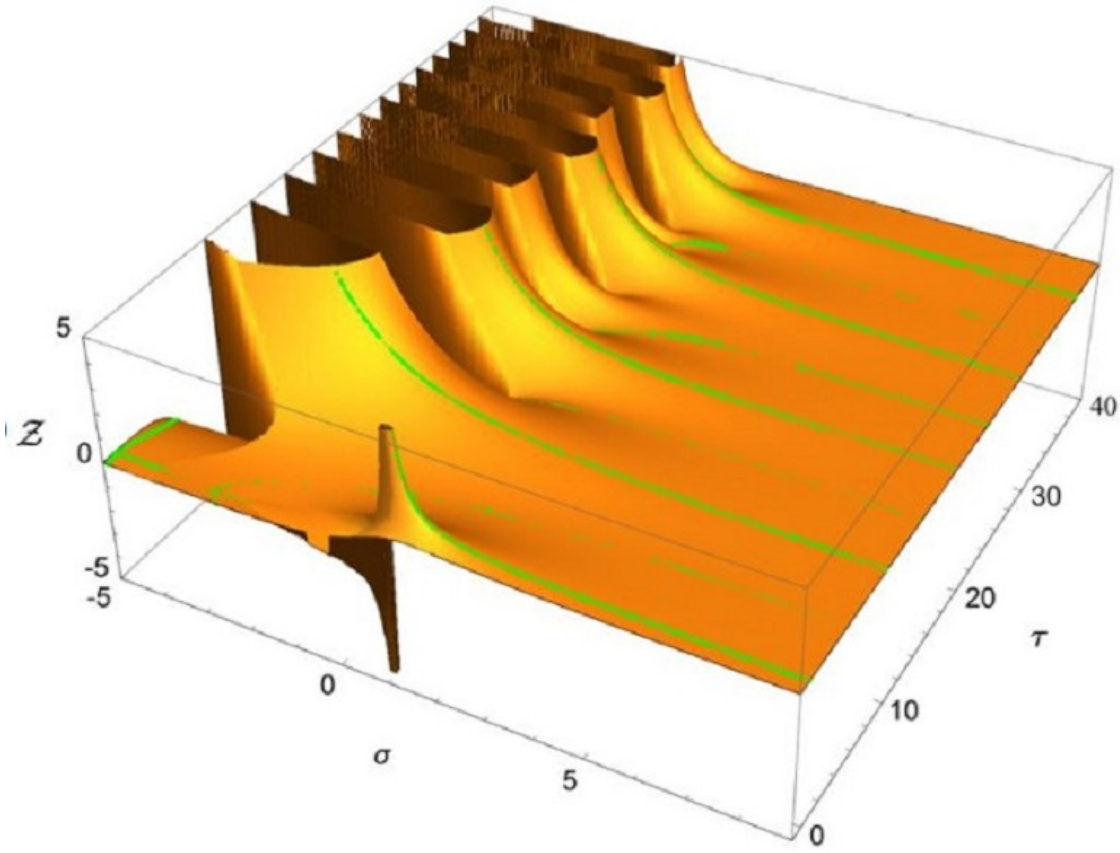
Lines in the complex plane where the Riemann zeta function is real (green) depicted on a relief representing the positive absolute value of zeta for arguments s, sigma and tau, where the real part of zeta is positive, and the negative absolute value of zeta where the real part of zeta is negative. This representation brings out most clearly that the lines of constant phase corresponding to phases of integer multiples of 2pi run down the hills on the left-hand side, turn around on the right and terminate in the non-trivial zeros.

This pattern repeats itself infinitely many times. The points of arrival and departure on the right-hand side of the picture are equally spaced and given by equation (cRZ).

[9]:
```
print ("*** 3-dim. Plot of Riemann Zeta Fct. for complex plane / Explanation␣
 ↪see above ***")

from IPython.display import Image

Image('Images/Plot-complex_RiemannZeta-Fct.jpg')
```

```
*** 3-dim. Plot of Riemann Zeta Fct. for complex plane / Explanation see above
***
```

[9]:



1.2.4 Item2.4: Calulate Zeta(s) for s=integer.

We calcualate here some special values of the Riemann Zeta function Zeta(s), where s is a complex number, with Im(s)=0 and s is an integer. So we list up the values of Zeta(s) with s = {-2, -1, 0, 1, 2, 3, 4, 6, 8}. For s=2 we see the famous Basel-problem (see Item1.2 above)

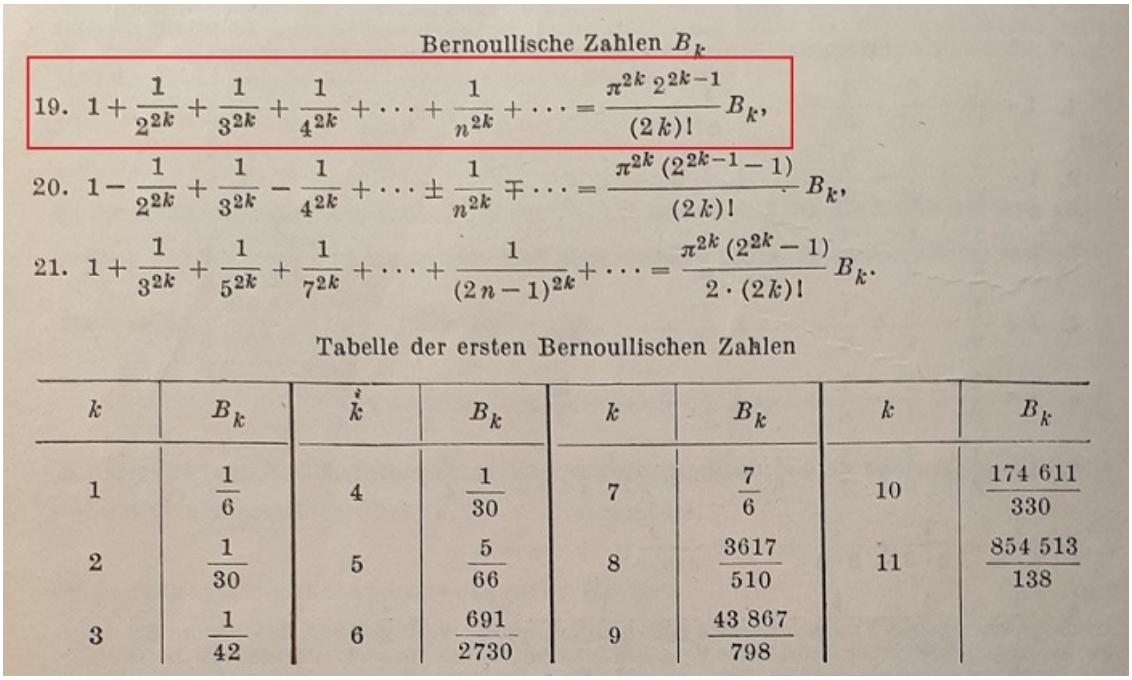
For real(s)>1: when calculate for s=natural numbers greater 2; i.e. s is from the number set S={2,3,4,5,6,7,8….). We see lim(Zeta(s))=1 when s goes in the direction of infinity.

For s=2k (k=1,2,3,…), we see can define the values of Zeta(2k) with Bernoulli numbers Bk… See Bronstein, page 254, Formula `19.' (red box).

[10]:
```
print ("*** Bernoulli Numbers Bk ***")

from IPython.display import Image

Image('Images/bernoulli_numbers.jpg')
```

```
*** Bernoulli Numbers Bk ***
```

[10]:



Bernoullische Zahlen B_k

19. $$1+\frac{1}{2^{2k}}+\frac{1}{3^{2k}}+\frac{1}{4^{2k}}+\cdots+\frac{1}{n^{2k}}+\cdots=\frac{\pi^{2k}\,2^{2k-1}}{(2k)!}B_k,$$

20. $$1-\frac{1}{2^{2k}}+\frac{1}{3^{2k}}-\frac{1}{4^{2k}}+\cdots\pm\frac{1}{n^{2k}}\mp\cdots=\frac{\pi^{2k}\,(2^{2k-1}-1)}{(2k)!}B_k,$$

21. $$1+\frac{1}{3^{2k}}+\frac{1}{5^{2k}}+\frac{1}{7^{2k}}+\cdots+\frac{1}{(2n-1)^{2k}}+\cdots=\frac{\pi^{2k}\,(2^{2k}-1)}{2\cdot(2k)!}B_k.$$

Tabelle der ersten Bernoullischen Zahlen

k	B_k	k	B_k	k	B_k	k	B_k
1	$\frac{1}{6}$	4	$\frac{1}{30}$	7	$\frac{7}{6}$	10	$\frac{174\,611}{330}$
2	$\frac{1}{30}$	5	$\frac{5}{66}$	8	$\frac{3617}{510}$	11	$\frac{854\,513}{138}$
3	$\frac{1}{42}$	6	$\frac{691}{2730}$	9	$\frac{43\,867}{798}$		

[11]:
```
print ("*****************************************")
print ("*** examples: Zeta(s) for s = integers ***")
print ("*****************************************")

# 1. zeta(s)=0 for s=-7,-5,-3
print ("1. check zeta(s) for s=-7, -5, -3:")
print ("zeta(-7) =",zeta(-7))
print ("zeta(-5) =",zeta(-5))
print ("zeta(-3) =",zeta(-3))

# 2. zeta(-2)=0
print ("2. check zeta(-2)= 0:")
```

```
print ("zeta(-2) =",zeta(-2))

# 3. zeta(-1)=-1/12=-0,08333...
print ("****************************************")
print ("3. check zeta(-1)=-1/12=-0,08333...:")
print ("zeta(-1) =",zeta(-1))

# 4. zeta(0)=-1/2
print ("****************************************")
print ("4. check zeta(0)=-1/2:")
print ("zeta(0) =",zeta(0))

# 5. zeta(1)=inifinity
print ("****************************************")
print ("5. check zeta(1)=unendlich(inf):")
print ("zeta(1) =",zeta(1))

# 6. zeta(2)=pi²/6 Bernoulli formula,k=1
print ("****************************************")
print ("zeta(2)=pi²/6 see Bernoulli formula,k=1")
print ("6. check zeta(2)=pi²/6=1,644934...:")
print ("zeta(2) =",zeta(2))

# 7. zeta(3)=1,2020...
print ("****************************************")
print ("7. check zeta(3)= 1,202056...:")
print ("zeta(3) =",zeta(3))

# 8. zeta(4)=(pi²)²/90 Bernoulli formula,k=2
print ("****************************************")
print ("zeta(4)=(pi²)²/90 Bernoulli formula,k=2")
print ("8. zeta(4)=((pi²))²/90 = 1,082323...:")
print ("zeta(4) =",zeta(4))

# 9. zeta(5)=1,0369277...
print ("****************************************")
print ("9. zeta(5)=1,0369277...:")
print ("zeta(5) =",zeta(5))

# 10. zeta(6)=(pi²)³/945 Bernoulli formula,k=3
print ("****************************************")
print ("zeta(6)=(pi²)³/945 Bernoulli formula,k=3")
print ("10. zeta(6)=(pi²)³/945=1,017343...:")
print ("zeta(6) =",zeta(6))

# 11. zeta(7)=1,008349...
print ("****************************************")
```

```
print ("11. zeta(7)=1,008349...:")
print ("zeta(7) =",zeta(7))

# 12. zeta(8)=(pi²)²)²/9450 Bernoulli formula,k=4
print ("*********************************************")
print ("zeta(8)=((pi²)²)²/9450 Bernoulli formula,k=4")
print ("12. zeta(8)=1,0040773..:")
print ("zeta(8) =",zeta(8))

# 13. zeta(s) for s=50,100,201,500, 1201
print ("*********************************************************")
print ("*** 13. calc. zeta(s) for s = 50,100,201,500,1201 to ***")
print ("*** check [lim(s->inf)](zeta(s))=1 for s=number > 1 ****")
print ("*********************************************************")
print ("zeta(50) =",zeta(50))
print ("zeta(100) =",zeta(100))
print ("zeta(201) =",zeta(201))
print ("zeta(500) =",zeta(500))
print ("zeta(1201) =",zeta(1201))
```

```
****************************************
*** examples: Zeta(s) for s = integers ***
****************************************
1. check zeta(s) for s=-7, -5, -3:
zeta(-7) = 0.004167422013553654
zeta(-5) = -0.003968252590985674
zeta(-3) = 0.008333333335927267
2. check zeta(-2)= 0:
zeta(-2) = 1.5603186562147366e-13
****************************************
3. check zeta(-1)=-1/12=-0,08333…:
zeta(-1) = -0.08333333333332381
****************************************
4. check zeta(0)=-1/2:
zeta(0) = -0.49999999999999906
****************************************
5. check zeta(1)=unendlich(inf):
zeta(1) = inf
****************************************
zeta(2)=pi²/6 see Bernoulli formula,k=1
6. check zeta(2)=pi²/6=1,644934…:
zeta(2) = 1.6449340668482266
****************************************
7. check zeta(3)= 1,202056…:
zeta(3) = 1.2020569031595942
****************************************
zeta(4)=(pi²)²/90 Bernoulli formula,k=2
```

```
8. zeta(4)=((pi²))²/90 = 1,082323…:
zeta(4) = 1.0823232337111381
****************************************
9. zeta(5)=1,0369277…:
zeta(5) = 1.03692775514337
****************************************
zeta(6)=(pi²)³/945 Bernoulli formula,k=3
10. zeta(6)=(pi²)³/945=1,017343…:
zeta(6) = 1.0173430619844488
****************************************
11. zeta(7)=1,008349…:
zeta(7) = 1.0083492773819225
****************************************
zeta(8)=((pi²)²)²/9450 Bernoulli formula,k=4
12. zeta(8)=1,0040773..:
zeta(8) = 1.0040773561979444
**************************************************
*** 13. calc. zeta(s) for s = 50,100,201,500,1201 to ***
*** check [lim(s->inf)](zeta(s))=1 for s=number > 1 ****
**************************************************
zeta(50) = 1.0000000000000009
zeta(100) = 1.0
zeta(201) = 1.0
zeta(500) = 1.0
zeta(1201) = 1.0
```

1.2.5 Item2.5: Riem. Fct. Equation (RFE) using Gamma-Fct. & Trivial zeros of Zeta-Fct

We calcualate here some special values with trival zeros of the Riemann Zeta function Zeta(s), where s is a complex number, with Im(s)=0. So we list up the values of Zeta(s) with s = {-8, -6, -4, -2}. In addition we calculate also some Zeta(s) where s is a fracture number.

The zeta function satisfies the `Riemann's Functional Equation (RFE)' - see image below:

This is an equality of meromorphic functions valid on the whole complex plane. The equation relates values of the Riemann zeta function at the points s and 1 - s, in particular relating even positive integers with odd negative integers. Owing to the zeros of the sine function, the functional equation implies that zeta(s) has a simple zero at each even negative integers = -2n, known as the trivial zeros of zeta(s). When s is an even positive integer, the product sine(pi*s/2)* gamma(1 - s) on the right is non-zero because gamma(1 - s) has a simple pole, which cancels the simple zero of the sine factor.

[12]:
```
print␣
↪("**************************************************************************************")
```

```
print ("*** The zeta func. zeta(s) satisfies the 'Riemann's Functional Equation␣
 ↪(RFE)' ***")
print␣
 ↪("*********************************************************************************")

from IPython.display import Image
Image('Images/Riemann_functional_equation.JPG')
```

```
*********************************************************************************
**
*** The zeta func. zeta(s) satisfies the 'Riemann's Functional Equation (RFE)'
***
*********************************************************************************
**
```

[12]:

Riemann's functional equation [edit]

The zeta function satisfies the functional equation

$$\zeta(s) = 2^s \pi^{s-1} \sin\left(\frac{\pi s}{2}\right) \Gamma(1-s)\, \zeta(1-s),$$

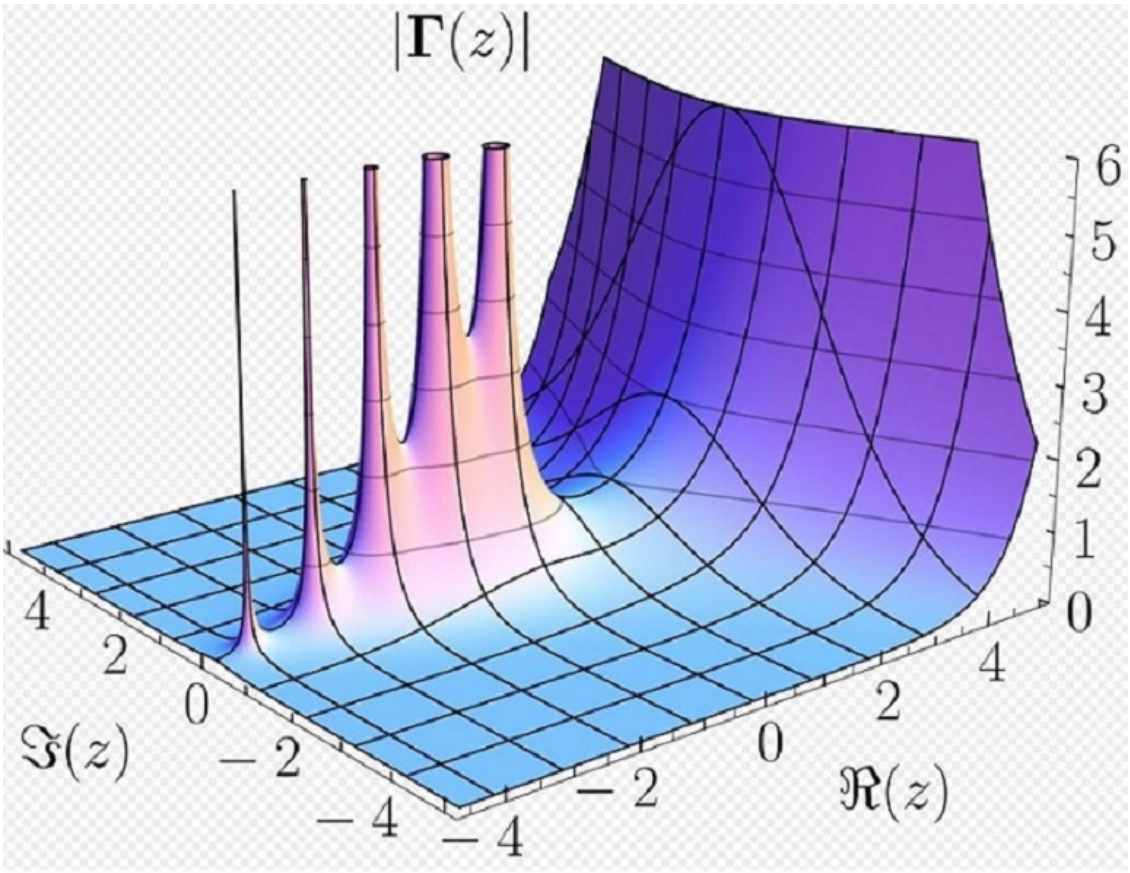
where $\Gamma(s)$ is the gamma function. This is an equality of meromorphic functions

[13]:

```
print␣
 ↪("*********************************************************************************")
print ("***** 3-dimensional plot of the absolute value of the complex gamma␣
 ↪function *****")
print ("*** we also see the poles of gamma(z) where z=-n, n=1,2,3... (natural␣
 ↪numbers) ***")
print␣
 ↪("*********************************************************************************")

from IPython.display import Image

Image('Images/Plot-complex_gamma-fct.JPG')
```

```
*********************************************************************************
**
***** 3-dimensional plot of the absolute value of the complex gamma function
*****
*** we also see the poles of gamma(z) where z=-n, n=1,2,3… (natural numbers)
***
*********************************************************************************
**
```

[13]:



[14]:

```
print("*****************************************************")
print("** 'Calculate zeta(s) for s=-1,-3,-5,-7 by using RFE **")
print("*****************************************************")
# 1. zeta(-1)=-1/12
print ("zeta(-1)=(1/2pi²)*sin(-pi/2)*gamma(2)*zeta(2)")
print ("        =(1/2pi²)*(-1)*1*(pi²/6)=-1/12       ")

# 2. zeta(-3)=1/120
print ("************************************************")
print ("zeta(-3)=(1/8(pi²)²)*sin(-3pi/2)*gamma(4)*zeta(4)")
print ("        =(1/8(pi²)²)*(+1)*3!*((pi²)²/90)=6/(8*90)")
print ("        =6/720=1/120                            ")

# 3. zeta(-5)=-1/252
print ("************************************************")
print ("zeta(-5)=(1/32(pi²)³)*sin(-5pi/2)*gamma(6)*zeta(6)")
print ("        =(1/32(pi²)³)*(-1)*5!*((pi²)³/945)       ")
print ("        =-120/(32*945)=-1/(4*63)=-1/252          ")
```

```
# 4. zeta(-7)=1/240
print ("*****************************************************")
print ("zeta(-7)=(1/128((pi²)²)²)*sin(-7pi/2)*gamma(8)*zeta(8)")
print ("        =(1/128((pi²)²)²)*(+1)*7!*(((pi²)²)²/9450)    ")
print ("        =5040/(128*9450)=504/(128*945)=63/(16*945)    ")
print ("        =1/(16*15)=1/240                              ")
```

```
*****************************************************
** 'Calculate zeta(s) for s=-1,-3,-5,-7 by using RFE **
*****************************************************
zeta(-1)=(1/2pi²)*sin(-pi/2)*gamma(2)*zeta(2)
        =(1/2pi²)*(-1)*1*(pi²/6)=-1/12
***********************************************
zeta(-3)=(1/8(pi²)²)*sin(-3pi/2)*gamma(4)*zeta(4)
        =(1/8(pi²)²)*(+1)*3!*((pi²)²/90)=6/(8*90)
        =6/720=1/120
***************************************************
zeta(-5)=(1/32(pi²)³)*sin(-5pi/2)*gamma(6)*zeta(6)
        =(1/32(pi²)³)*(-1)*5!*((pi²)³/945)
        =-120/(32*945)=-1/(4*63)=-1/252
*****************************************************
zeta(-7)=(1/128((pi²)²)²)*sin(-7pi/2)*gamma(8)*zeta(8)
        =(1/128((pi²)²)²)*(+1)*7!*(((pi²)²)²/9450)
        =5040/(128*9450)=504/(128*945)=63/(16*945)
        =1/(16*15)=1/240
```

[15]:
```
print("***********************************************")
print("** 'Trivial' zeros are for z=-2,-4,-6,-8,etc. **")
print("***********************************************")
# 1. zeta(-2)=0
print ("1. check zeta(-2)=0:")
print ("zeta(-2) =",zeta(-2))

# 2. zeta(-4)=0
print ("*************************************")
print ("2. check zeta(-4)=0:")
print ("zeta(-4) =",zeta(-4))

# 3. zeta(-6)=0
print ("*************************************")
print ("3. check zeta(-6)=0:")
print ("zeta(-6) =",zeta(-6))


# 4. zeta(-8)=0
print ("*************************************")
print ("4. check zeta(-8)=0:")
```

```
print ("zeta(-8) =",zeta(-8))
```

```
*********************************************
** 'Trivial' zeros are for z=-2,-4,-6,-8,etc. **
*********************************************
1. check zeta(-2)=0:
zeta(-2) = 1.5603186562147366e-13
************************************
2. check zeta(-4)=0:
zeta(-4) = 6.429216196053237e-11
************************************
3. check zeta(-6)=0:
zeta(-6) = 2.8347851868673592e-08
************************************
4. check zeta(-8)=0:
zeta(-8) = 1.3859169880942308e-05
```

[16]:
```
# Calculate zeta(s) for fracture numbers s=-15/2,-13/2,...,15/2
print ("*********************************************************")
print ("**** calculate values for s = -15/2, -13/2, ..., 15/2 ****")
print ("*** check the results for s = -3/2, -1/2 and 1/2 using ***")
print ("*** the Riemann's Functional Equation (RFE); see above ***")
print ("*********************************************************")
print ("zeta(-15/2) =",zeta(-15/2))
print ("zeta(-13/2) =",zeta(-13/2))
print ("zeta(-11/2) =",zeta(-11/2))
print ("zeta(-9/2) =",zeta(-9/2))
print ("zeta(-7/2) =",zeta(-7/2))
print ("zeta(-5/2) =",zeta(-5/2))
print ("with RFE follows zeta(-3/2)=(-3/16)*(1/pi²)*zeta(5/2)")
print ("using zeta(5/2),see below,the correct result is found")
print ("zeta(-3/2) =",zeta(-3/2))
print ("with RFE we see that zeta(-1/2)=(-1/4)*(1/pi)*zeta(3/2)")
print ("using zeta(3/2), see below, the correct result is found")
print ("zeta(-1/2) =",zeta(-1/2))
print ("RFE=> zeta(1/2)=(2/2)*root((pi/pi))*zeta(1/2) is correct!")
print ("zeta(1/2) =",zeta(1/2))
print ("zeta(3/2) =",zeta(3/2))
print ("zeta(5/2) =",zeta(5/2))
print ("zeta(7/2) =",zeta(7/2))
print ("zeta(9/2) =",zeta(9/2))
print ("zeta(11/2) =",zeta(11/2))
print ("zeta(13/2) =",zeta(13/2))
print ("zeta(15/2) =",zeta(15/2))
```

```
*********************************************************
**** calculate values for s = -15/2, -13/2, …, 15/2 ****
*** check the results for s = -3/2, -1/2 and 1/2 using ***
```

```
*** the Riemann's Functional Equation (RFE); see above ***
**********************************************************
zeta(-15/2) = 0.003274799574186712
zeta(-13/2) = 0.0027469095530168607
zeta(-11/2) = -0.0026714542649568995
zeta(-9/2) = -0.003091668796611392
zeta(-7/2) = 0.004441011354616652
zeta(-5/2) = 0.008516928778669624
with RFE follows zeta(-3/2)=(-3/16)*(1/pi²)*zeta(5/2)
using zeta(5/2),see below,the correct result is found
zeta(-3/2) = -0.025485201889790032
with RFE we see that zeta(-1/2)=(-1/4)*(1/pi)*zeta(3/2)
using zeta(3/2), see below, the correct result is found
zeta(-1/2) = -0.2078862249773517
RFE=> zeta(1/2)=(2/2)*root((pi/pi))*zeta(1/2) is correct!
zeta(1/2) = -1.460354508809586
zeta(3/2) = 2.612375348685488
zeta(5/2) = 1.341487257250917
zeta(7/2) = 1.1267338673170566
zeta(9/2) = 1.0547075107614543
zeta(11/2) = 1.0252045799546856
zeta(13/2) = 1.0120058998885244
zeta(15/2) = 1.0058267275365227
```

1.2.6 Item2.6: Summary of Results: Values+Graph of Riem. Zeta(s) Fct. with Im(s)=0

As a summary and final result of the above work we show the graph of zeta(s)where Im(s)=0, s.t. s=real number (without s=1). We use also the calculated values of zeta(s) from this Jupyter Notebook and summaries them in a small table (see below). Remarks: We see a pole of zeta(s) at s=1 and an asymtote at f(s)=1 for s>1. Compare also the remarks about the lim(zeta(s)) for s which goes to the positive infinity: lim(s->+inf)=1.

[17]:
```
print ("**** Value-Table of Riem. Zeta(s) Fct. with Im(s)=0 ****")

from IPython.display import Image

Image('Images/Value_Zeta(s)_Im(s)=0.JPG')
```

```
**** Value-Table of Riem. Zeta(s) Fct. with Im(s)=0 ****
```
[17]:

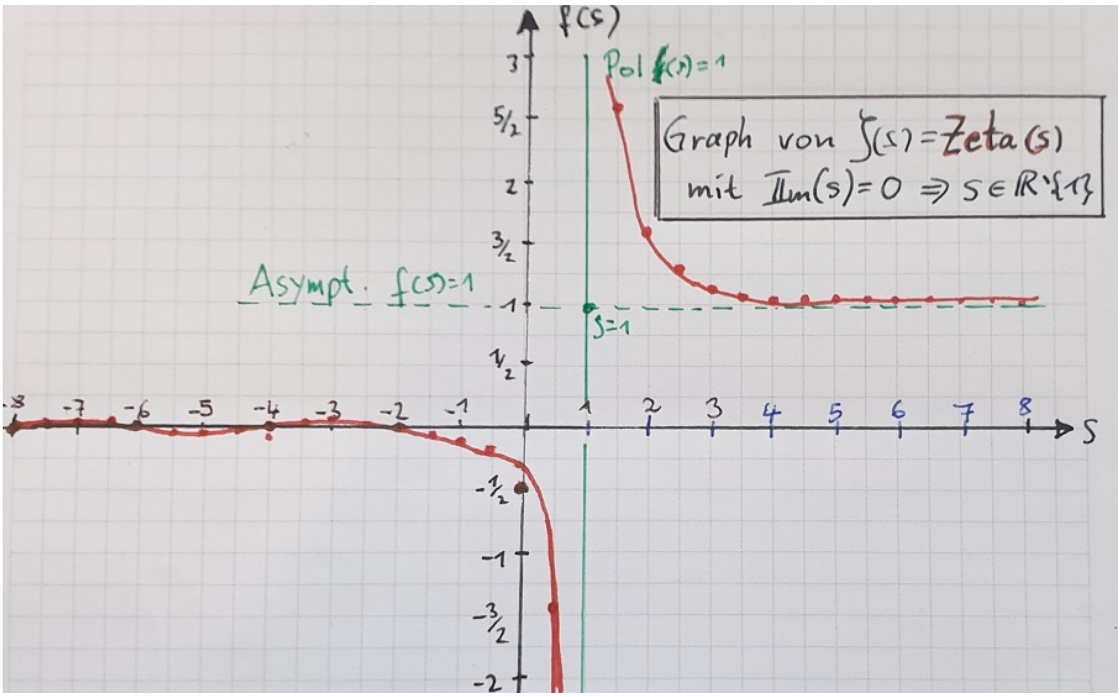
s	Calculation of zetas(s) with (cRZ) Rounding errors may occur in proc. (cRZ)	zeta(s) with (RFE)
-8	zeta(-8) = 1.3859169880942308e-05	0
-7,5	zeta(-15/2) = 0.003274799574186712	
-7	zeta(-7) = 0.004167422013553654	1/240 = 0.0041666... -> Rounding E.
-6,5	zeta(-13/2) = 0.0027469095530168607	
-6	zeta(-6) = 2.8347851868673592e-08	0
-5,5	zeta(-11/2) = -0.0026714542649568995	
-5	zeta(-5) = -0.003968252590985674	-1/252= 0.003968253968 -> Rounding E.
-4,5	zeta(-9/2) = -0.003091668796611392	
-4	zeta(-4) = 6.429216196053237e-11	0
-3,5	zeta(-7/2) = 0.004441011354616652	
-3	zeta(-3) = 0.008333333335927267	1/120 = 0.008333...----> Rounding
-2,5	zeta(-5/2) = 0.008516928778669624	
-2	zeta(-2) = 1.5603186562147366e-13	0
-1,5	zeta(-3/2) = -0.025485201889790032	
-1	zeta(-1) = -0.08333333333332381	-1/12=0.08333...---> Rouding Error
-0,5	zeta(-1/2) = -0.2078862249773517	
0	zeta(0) = -0.49999999999999906	-1/2= 0.5 -----> Rounding Error
0,5	zeta(1/2) = -1.460354508809586	
1	infinity	Pole
1,5	zeta(3/2) = 2.612375348685488	
2	zeta(2) = 1.6449340668482266	$Pi^2/6$
2,5	zeta(5/2) = 1.341487257250917	
3	zeta(3) = 1.2020569031595942	
3,5	zeta(7/2) = 1.1267338673170566	
4	zeta(4) = 1.0823232337111381	$(pi^2)^2/90$
4,5	zeta(9/2) = 1.0547075107614543	
5	zeta(5) = 1.03692775514337	
5,5	zeta(11/2) = 1.0252045799546856	
6	zeta(6) = 1.0173430619844488	$(pi^2)^3/945$
6,5	zeta(13/2) = 1.0120058998885244	
7	zeta(7) = 1.0083492773819225	
7,5	zeta(15/2) = 1.0058267275365227	
8	zeta(8) = 1.0040773561979444	$((pi^2)^2)^2/9450$

[18]:
```
print ("****** Graph of Riem. Zeta(s) Fct. with Im(s)=0 ******")

from IPython.display import Image
```

```
Image('Images/Graph_Zeta(s)_Im(s)=0.JPG')
```

****** Graph of Riem. Zeta(s) Fct. with Im(s)=0 ******

[18]:



[19]:
```
import time
print("****current date and time **************")
print("Date and Time:",time.strftime("%d.%m.%Y %H:%M:%S"))
print("end")
```

```
****current date and time **************
Date and Time: 21.05.2021 22:23:34
end
```