Abgabe 1 Zetafunktion

Wayne Ströbel, Silas Müller

29/4/2021

Riemannsche Zeta Funktion

Die Riemannsche Zeta Funktion spielt eine wichtige Rolle fur viele Anwendungen in der Physik und Mathematik. Sie ist wie folgt definiert:

$$\zeta(s) = \sum_{k=1}^{\infty} \frac{1}{k^s}, \qquad s \in \mathbb{C}\backslash\{0\} \tag{1}$$

Wir wollen nun im folgenden $\zeta(2)$ numerisch auf die 15. Dezimalstelle genau berechnen. Hierfür haben wir zwei verscheidene Methoden benutzt,

Naiver Ansatz

Zuerst hatten wir es auf die Naive Methode versucht. Dabei haben wir die Formel einfach in C programmiert und die Schrittzahl variiert:

```
  //gcc -Wall -pedantic Zeta_slow.c -o Zeta_slow
 //./Zeta_slow
//Silas Mueller, Wayne Stroebel
#include <stdio.h>
#include <math.h>
#include <stdlib.h>


double zetafunktion(double s,int genauigkeit, double* z)
{

    double x = 0;
    for (long int i = genauigkeit;(i > 0); i--)
    {
        x += 1/(pow(i, s));
        if (i < 100)
        {
            z[i] = x;
        }
    }
    return x;
}

int main(int argc, char const *argv[])
```

```
{

    double s = 2;                                                    //Wert fuer Zeta(s)
    int genauigkeit = 9;                                             //Genauigkeit if Form von 10^ge
    int N = pow(10, genauigkeit);
    double z[100];
    int y[100];
    for (int i = 1; i < 100; i++)                                   //Laufwert fuer Plot
    {
        y[i] = i;
    }

    printf("Zeta(2) = %.15f\n",zetafunktion(s,N,z));

    FILE* fp;
    fp = fopen("Zeta_slow_data.txt", "w");                           //oeffnet die Datei Zeta_slow_da
    if(fp == NULL) {
        printf("Datei konnte nicht geoeffnet werden!\n");
        return 1;
    }

    for (unsigned int i = 1; i < 100; i++) {                          //schreibt zwischenergebnisse
        fprintf(fp, "%i %.16f \n",y[i],z[i]);
    }
    fclose(fp);

    return 0;
}
```

Unser Programm berechnet für jeden Schritt den N-ten Wert der Summe aus, und addiert diesen auf einen Gesamtwert. Die Werte nach jedem Schritt speichern wir für die ersten 100 Schritte in einer externen .txt Datei, damit das Konvergenzverhalten später geplotet werden können. Die berechneten Werte sind in der Zeta_slow_data.txt Datei wiederzufinden. Dieser Ansatz liefert uns nach einer Rechenzeit von ca. 1 Minute eine Genauigeikt von 8 Nachkommastellen mit 10^9 Schritten. Diese Methode ist also nicht ausreichend, um auf das gewollte Ergebnis zu kommen.

KONVERGENZ KONVERGENZ KONVERGENZ KONVERGENZ KONVERGENZ

Borwein Methode

Die Borwein Methode nutzt einen anderen Ansatz. Hierbei verwenden wir abgebrochene alternierende Summen. Die Formel dafür sieht wie folgt aus:

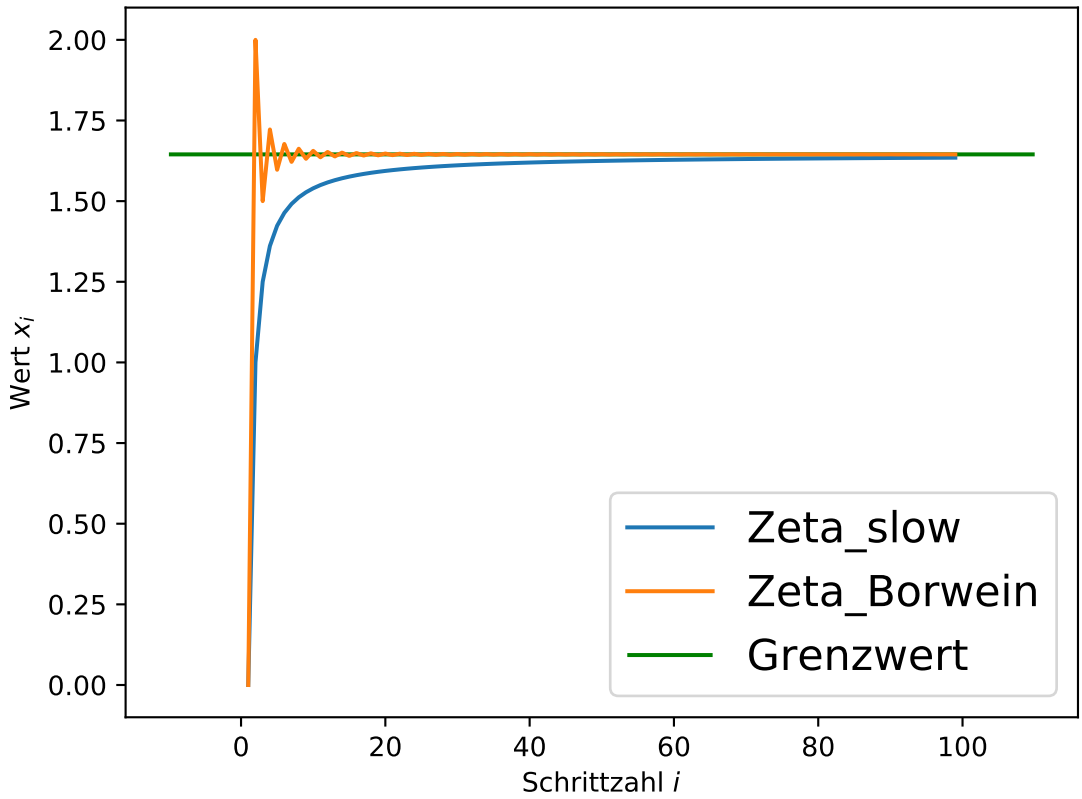
$$\zeta(s) = \frac{1}{1-2^{1-s}} \sum_{k=1}^{infty} \frac{(-1)^{k-1}}{k^s} \tag{2}$$

Die Implementierung in C sieht bei uns wie folgt aus:

```
  //gcc -Wall -pedantic Zeta_Borwein.c -o Zeta_Borwein -lm
 //./Zeta_Borwein
//Silas Mueller, Wayne Stroebel
#include <stdio.h>
#include <math.h>
```

```
double zetafunktion(double s,long int genauigkeit, double* z)
{

    double x;
    double a = 1 / (1-pow(2, 1 - s));
    for (size_t i = genauigkeit;(i > 0); i--)
    {
        x += pow(-1, i-1)/(pow(i, s));
        if (i < 100)
        {
            z[i] = a*x;
        }
    }
    x = a*x;
    return x;
}

int main(int argc, char const *argv[])
{
    double s = 2;                                                    //Wert fuer Zeta(s)
    int genauigkeit = 8;                                             //Genauigkeit if Form von 10^ge
    long int N = pow(10, genauigkeit);
    double z[100];
    int y[100];
    for (int i = 1; i < 100; i++)                                      //Laufwert fuer Plot
    {
        y[i] = i;
    }

    printf("Zeta(2) = %.15f\n",zetafunktion(s,N,z));

    FILE* fp;
    fp = fopen("Zeta_Borwein_data.txt", "w");                           //oeffnet die Datei Zeta_Bor
    if(fp == NULL) {
        printf("Datei konnte nicht geoeffnet werden!\n");
        return 1;
    }

    for (unsigned int i = 1; i < 100; i++) {                           //schreibt zwischenergebnisse
        fprintf(fp, "%i %.16f \n",y[i],z[i]);
    }
    fclose(fp);
    return 0;
}
```

Wir berechnen wieder die einzelnen Summen und multiplizieren das Ergebnis mit dem Vorfaktor $a = \frac{1}{1-2^{1-s}}$, um auf unseren Wert zu kommen. Mit dieser Methode erhalten wir nach nur wenigen Sekunden und wieder einer Schrittzahl von 10^9 das Ergebnis genau auf 15 Nachkommastellen.


2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Wert x_i
0
20
40
60
80
100
Schrittzahl i
Zeta_slow
Zeta_Borwein
Grenzwert