

# Übung 05: Spektrum einer Klaviersaite

Tobias Blesgen und Leonardo Thome

07.07.2021

Schlägt man eine Klaviersaite an, so verhält sie sich nach:

$$\frac{\partial^2 \psi(x, t)}{\partial t^2} = c^2 \frac{\partial^2 \psi(x, t)}{\partial x^2} - \frac{\gamma}{l} \left| \frac{\partial \psi(x, t)}{\partial t} \right| \left( \frac{\partial \psi(x, t)}{\partial t} \right). \quad (1)$$

Wir wollen dieses Verhalten im Folgenden numerisch untersuchen. Hierzu schreiben wir die Gleichung 1 mit  $\Xi = \frac{x}{l}, \phi = \frac{\psi}{l}, \tau = \frac{tc}{l}$  um zu der dimensionslosen Gleichung:

$$\frac{\partial^2 \phi(\Xi, \tau)}{\partial \tau^2} = \frac{\partial^2 \phi(\Xi, \tau)}{\partial \Xi^2} - \gamma \left| \frac{\partial \phi(\Xi, \tau)}{\partial \tau} \right| \left( \frac{\partial \phi(\Xi, \tau)}{\partial \tau} \right). \quad (2)$$

Um dieses Problem analytisch angehen zu können, müssen wir die Differentiale nähern und schreiben hierzu die Gleichung 2 um zu:

$$\frac{\psi_{i+1,j} - 2\psi_{i,j} + \psi_{i-1,j}}{\Delta \tau^2} = \frac{\psi_{i,j+1} - 2\psi_{i,j} + \psi_{i,j-1}}{\Delta \Xi^2} - \gamma \left| \frac{\psi_{i,j} - \psi_{i-1,j}}{\Delta \tau} \right| \frac{\psi_{i,j} - \psi_{i-1,j}}{\Delta \tau} \quad (3)$$

Dies lässt sich weiter umstellen zu:

$$\psi_{i+1,j} = \frac{\Delta \tau^2}{\Delta \Xi^2} (\psi_{i,j+1} - 2\psi_{i,j} + \psi_{i,j-1}) + 2\psi_{i,j} - \psi_{i-1,j} - \gamma |\psi_{i,j} - \psi_{i-1,j}| (\psi_{i,j} - \psi_{i-1,j}) \quad (4)$$

Diese Gleichung wird nun verwendet, um iterativ den nächsten Funktionswert an jedem Ort zu bestimmen. Als Randbedingung werden die Ränder  $\Xi = 0$  und  $\Xi = \Xi_{max}$  auf 0 gesetzt, da sie im Klavier befestigt sind.

## Implementation des numerischen Verfahrens

Nach der Courant-Friedrichs-Lewy Stabilitätsbedingung müssen wir bei der Schrittweitenwahl beachten, dass stets

$$1 \geq c \frac{\Delta t}{\Delta x} \quad (5)$$

gilt. Setzen wir  $\Xi$  und  $\tau$  ein, so erhalten wir nach einmaligem Umstellen die Ungleichung

$$\Delta \Xi \geq \frac{cl}{cl} \Delta \tau = \Delta \tau. \quad (6)$$

Wir müssen also darauf achten, die  $\tau$ -Schritte kleiner als die  $\Xi$ -Schritte zu halten.

```

#include<Rcpp.h>
#include<vector>
#include<algorithm>
#include<math.h>

using namespace Rcpp;

//[[Rcpp::export]]
NumericMatrix klaviersaite2(const double gamma, const int xSchritte,
                           const double dt, const int
                           zeitSchritte){

    // Array der Werte zur späteren Ausgabe
    NumericMatrix matrix(zeitSchritte, xSchritte);
    // Quelltext
    // Startwerte

    for (int i = 0; i<xSchritte; i++){
        matrix(0,i) = 0.0;
        matrix(1,i) = 0.0;
    }
    matrix(0,(int)(0.26*(xSchritte-1))) = 0.01/0.4;
    matrix(1,(int)(0.26*(xSchritte-1))) = 0.01/0.4;

    // Funktionsdurchläufe
    double dx = 1.0/(xSchritte - 1.0);
    double C = dt*dt/(dx*dx);

    for (int i = 2; i<zeitSchritte; i++){
        // Randbedingungen
        matrix(i,0) = 0.0;
        matrix(i,xSchritte-1) = 0.0;

        for (int j = 1; j<xSchritte-1; j++){
            matrix(i,j) = C*(matrix(i-1,j+1) - 2*matrix(i-1,j) +
                matrix(i-1,j-1)) + 2*matrix(i-1,j) - matrix(i-2,j) -
                gamma * fabs(matrix(i-1,j)-matrix(i-2,j)) *
                (matrix(i-1,j)-matrix(i-2,j));
        }
    }

    // Rückgabe für eine grafische Wiedergabe
    return matrix;
}

```

## Stabilitätsüberprüfung

Wir wollen am Beispiel des  $\gamma = 0$  Falls überprüfen, ob unsere Methodik stabil ist. Wir würden erwarten, dass die mittlere Amplitude über die Zeit konstant bleibt, und tragen hierzu die Summe aller Funktionspunkte grafisch gegen die Zeit auf:

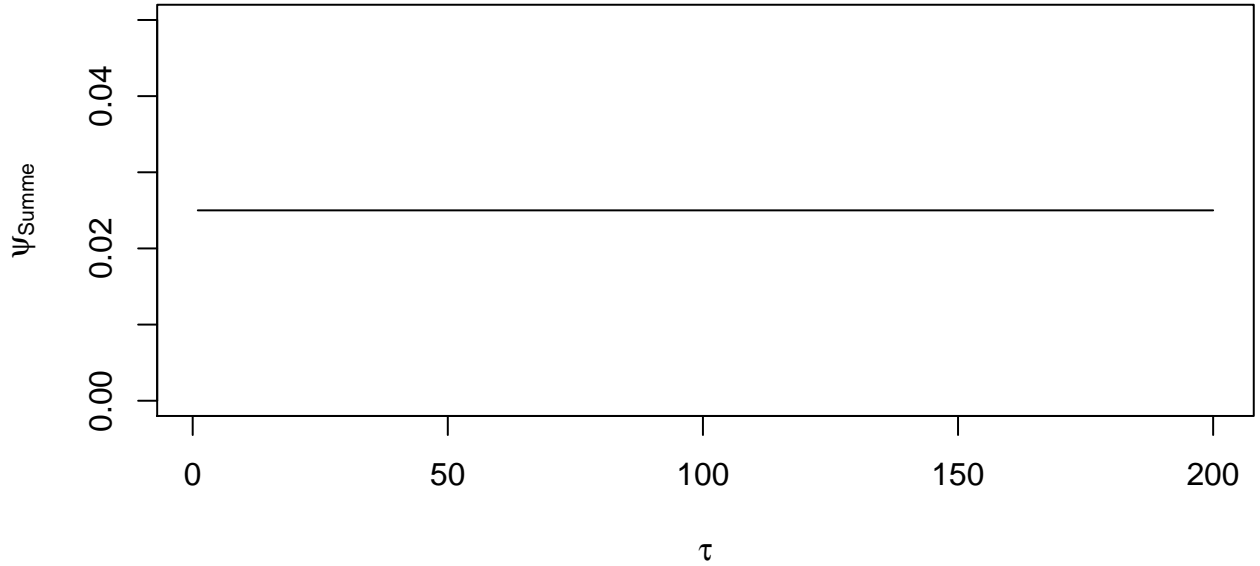


Abbildung 1: Stabilität bei keiner Abschwächung

Wie wir sehen können, bleibt die Amplitudensumme über dem betrachteten Bereich sehr konstant und wir können die Methode im Folgenden auf ein gedämpftes System anwenden.

## Anwendung

Wir wollen eine Klaviersaite schwingen lassen und auf ihre Obertöne und den Grundton untersuchen. Betrachten wir in einer ersten Simulation die räumliche Schwingung zu verschiedenen Zeiten, so können wir in Abb. 2 sehen, wie sich die Schwingung ausbreitet ( $\Delta \Xi = 2 \cdot 10^{-3}$  und  $\Delta \tau = 10^{-3}$ ).

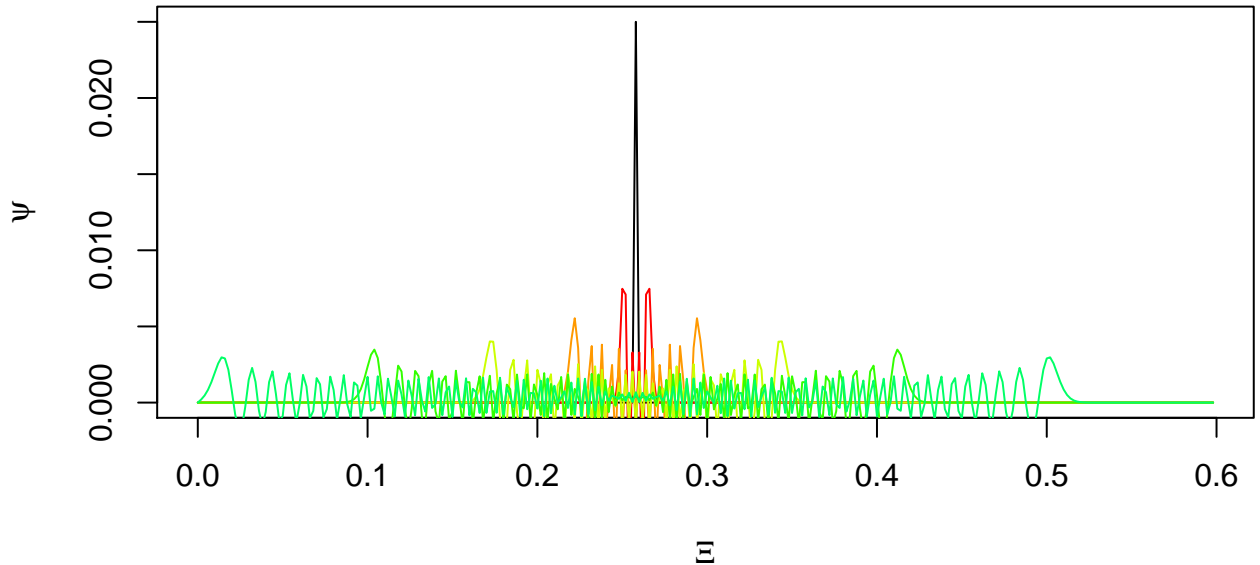


Abbildung 2: Räumlicher Verlauf

Wir wollen im Folgenden mit möglichst kleinen Diskretisierungslängen arbeiten, ohne die Rechenzeiten extrem in die Länge zu ziehen, und haben uns für  $\Delta \tau = 10^{-4}$  und  $\Delta \Xi = \frac{4}{3} \cdot 10^{-3}$  entschieden. Diese Längen halten auch die geforderte Bedingung Gl. 6 ein.

## Zeitlicher Verlauf

Für die genauere Untersuchung benötigen die zeitliche Entwicklung an einem Ort. Wir haben in Abb. 3 den Verlauf an der Stelle  $\Xi = 0,3$  gewählt und für die verschiedenen Zeiten aufgetragen. Die Kurve beginnt in der Ruhelage bis das erste Hochpunkt den Ort erreicht. Es lässt sich nun eine exponentiell abfallende Sinuskurve beobachten, bis die am Rand reflektierte Welle den Punkt erreicht. Die verbleibende Überlagerung stabilisiert sich zu einer annähernd gleichmässigen Schwebung. Diese gilt es im Folgenden zu untersuchen.

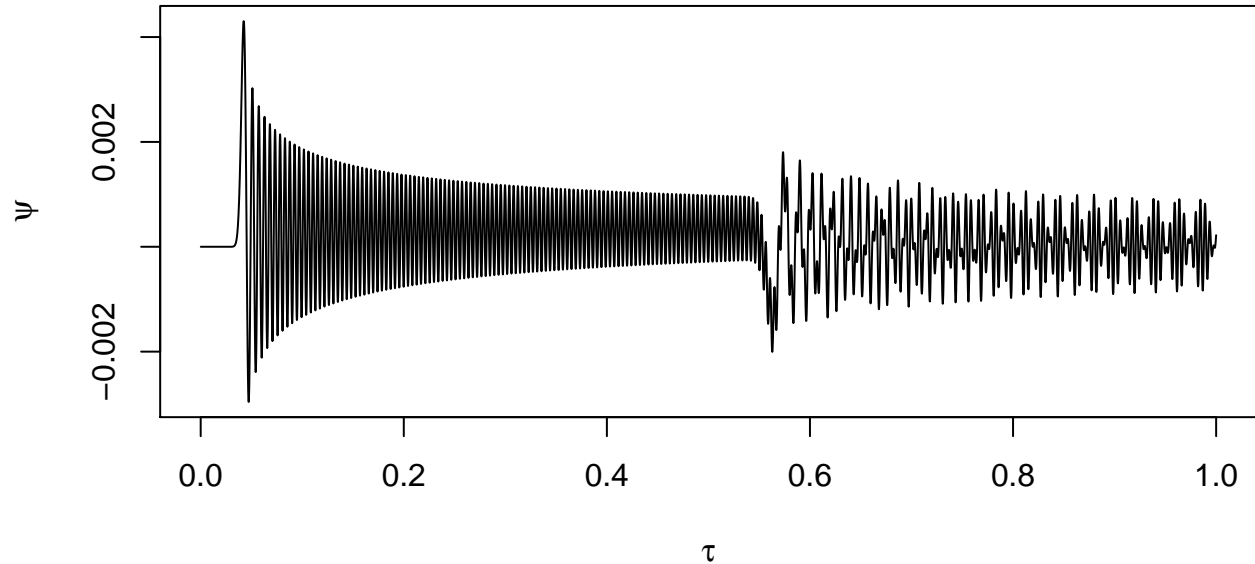


Abbildung 3: Zeitliche Entwicklung

## Fast Fourier Transform

Mit dem Ziel die Schwingung in ihre Frequenzen aufzuteilen, werden wir die Werte der zeitlichen Entwicklung Fourier transformieren. Diese Transformation zerlegt eine reelle Funktion in eine Summe von gewichteten Cosinusfunktionen. Wir werden den Fast Fourier Transform-Algorithmus verwenden, da wir mit diskreten Werten arbeiten. Die  $n$  erhaltenen Werte entsprechen anschließend den Gewichtungen der Cosinus in aufsteigender Wertigkeit, also:

$$T_f(x) = \frac{1}{\sqrt{2\pi}} \sum_{k=1}^n g_k \cos(kx) \quad (7)$$

```
#include <Rcpp.h>
#include <stdlib.h>

using namespace Rcpp;

double pi = 3.14159;
std::complex<double> I = 1i;

// Generiere die w Faktoren
std::vector<double> wInit(int n){
    std::vector<double> W(n);
    for (int a = 0; a < n; a++)
```

```

    {
        W[a]= cos((-2.*pi*(double)a)/((double)n));
    }
    return W;
}

// Funktion zum Umsortieren der verdrehten Terme
std::vector<int> fftResort(const int n){
    std::vector<int> k(n);
    int l = 0, m;
    for (int i = 0; i <= n-2; i++)
    {
        k[i] = 1;
        m = n/2;
        while (m <= 1)
        {
            l -= m;
            m /= 2;
        }
        l += m;
    }
    k[n-1] = n-1;
    return k;
}

// Fast Fourier Transform - Funktion (nimmt die z_i und gibt die g_i unsortiert aus)
//[[Rcpp::export]]
std::vector<double> fft2 (std::vector<double> z, const int r){
    const int n = (int)pow(2,r);
    int m = n/2;
    int K = 1;
    std::vector<double> w = wInit(n);
    int a, b;

    for (int i = 0; i < r; i++)
    {
        for (int k = 0; k < K; k++)
        {
            for (int j = 0; j < m; j++)
            {
                a = 2*k*m + j;
                b = a + m;
                z[a] += z[b];
                z[b] = w[((K*j) % n)]*(z[a] - 2.0*z[b]);
            }
        }
        m /= 2;
        K *= 2;
    }
    std::vector<int> index = fftResort(n);
    std::vector<double> ausgabe(n);
    for (int i = 0; i < n; i++)
    {

```

```

    ausgabe[i]= z[index[i]] / sqrt(n);
}
return ausgabe;
}

```

Zerlegen wir nun verschiedene Orte nach der Fouriertransformation, so erhalten wir Abb. 4.

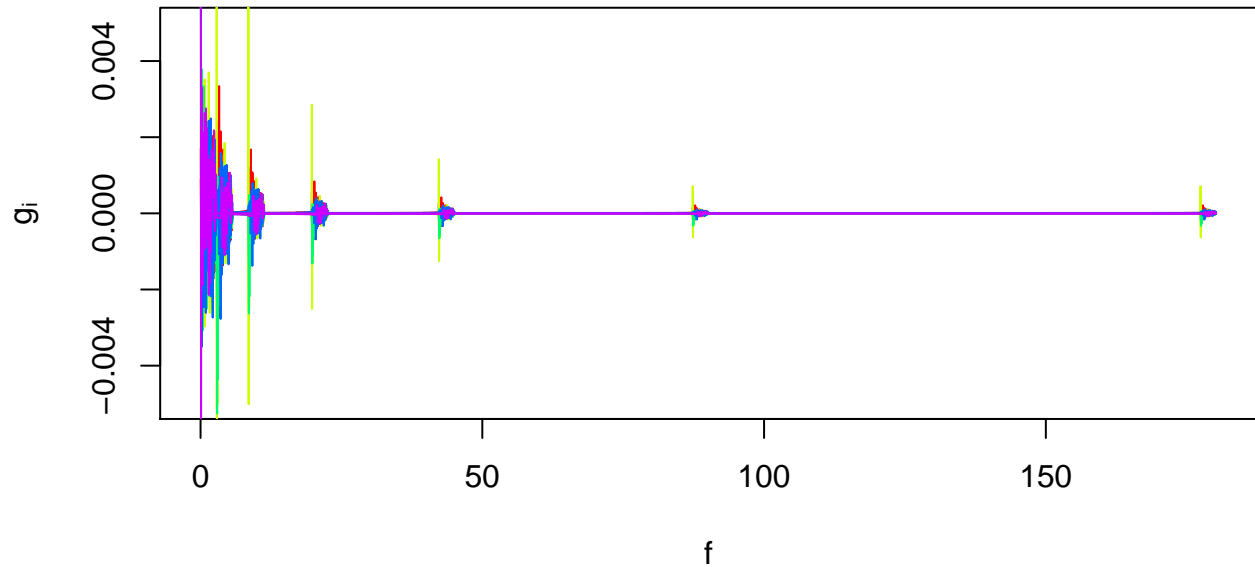


Abbildung 4: Fourier zerlegtes Spektrum

Wir können in Abb. 4 Ansammlungen von Frequenzen, welche dem Grundton bis hoch zum 5ten Oberton entsprechen, auslesen:

Ton	$f$
Grundton	$5 \pm 5$ Hz
1. Oberton	$10 \pm 5$ Hz
2. Oberton	$20 \pm 10$ Hz
3. Oberton	$45 \pm 10$ Hz
4. Oberton	$88 \pm 10$ Hz
5. Oberton	$175 \pm 10$ Hz

Tabelle 1: Frequenzen der Töne

Wir sehen, dass die Frequenzen der Obertöne nicht wie erwartet nach der Gl. 9 linear steigen, sondern in Näherung quadratisch (Abb. 5). Diese Verteilung der Frequenzen für die Töne könnte durch einen möglichen Fehler beim bestimmen der Frequenzen entstanden sein, bei diesem die fehlenden Vielfache des Grundtons verschwunden sind. Wir haben keinen physikalischen Grund gefunden, der diese Verteilung erklärt. Die Dämpfung, die ein Störfaktor sein könnte, beeinflusst zwar die einzelnen Schwingdauern der im Bezug auf den Fall ohne Dämpfung, jedoch sollten die Frequenzen da zueinander dadurch nicht stark beeinflusst werden. Zumindest sollte unseres Verständnis nach kein quadratischer Zusammenhang entstehen.

$$f_n = \frac{1}{2l} \sqrt{\frac{F}{\rho}} n. \quad (8)$$

Die dazu angenommene konstante Dämpfung sollte sich mit dem folgenden Zusammenhang auf die Frequenz

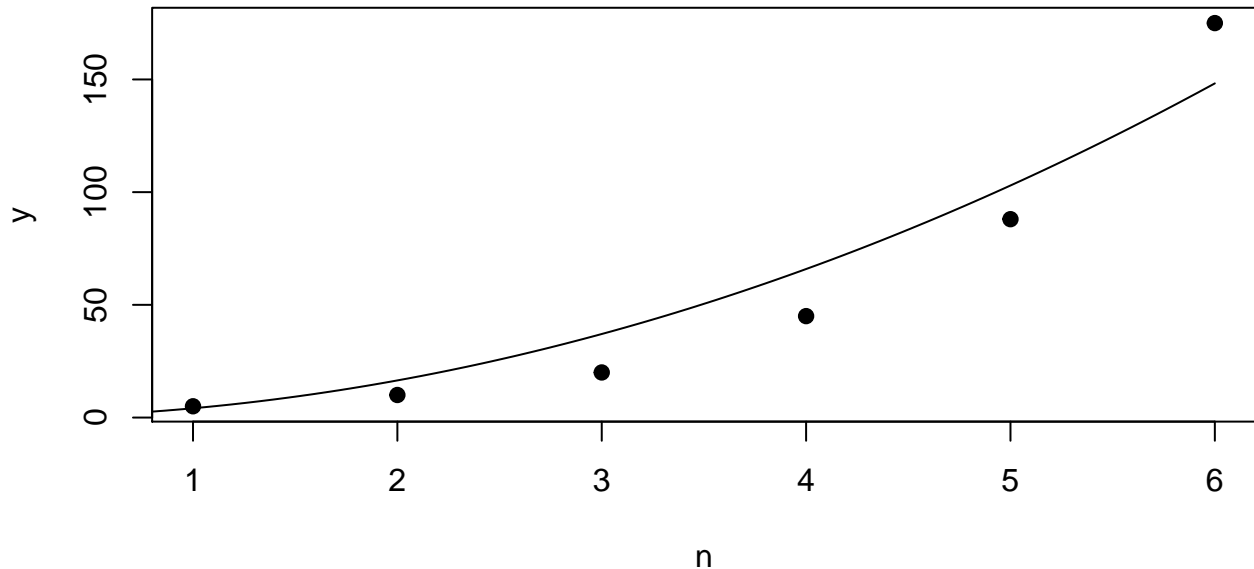


Abbildung 5: Plot der Frequenzen für den Grundton/Obertöne

auswirken:

$$2\pi f = \sqrt{4\pi^2 f_0^2 - \delta^2}. \quad (9)$$

wobei  $f_0$  die Frequenz ohne Dämpfung und  $\delta$  den Dämpfungsfaktor beschreiben, nach der Lösung für eine gedämpften harmonische Schwingung [1]. An der korrigierten Frequenz für die sich bildenden Schwingungen ist zu sehen, dass zumindest mit zunehmender Frequenz auch der Abstand zwischen den Frequenzen steigen muss, da die Dämpfung zunehmend kleineren Einfluss auf die Frequenz hat. Dieser Zusammenhang reicht jedoch nicht aus, den in Abb. 5 gefundenen Zusammenhang der Frequenzen zu erklären.

## Fazit

Durch das Umformen der DGL zur Gleichung 4 ließ sich der zeitliche Schwingungsverlauf örtlich bestimmen. Die dadurch erhaltene numerische Lösung konnten wir daraufhin nach dem Einschwingen über die FFT in die zugehörigen Frequenzen des Grundtons und der ersten 5 Obertöne zerlegen. Die dabei bestimmten Frequenz sind mit 5 Hz für den Grundton und 175 Hz für den 5ten Oberton recht klein, was ein skalierungs Fehler erahnen lässt. Jedoch konnten wir diesen nicht finden. Es ist auch auffallend, dass die Frequenzen vom Grundton in Näherung quadratische zunehmen. Dies kann durch eine konstante Dämpfung nicht erklärt werden. Somit müsste es an der gewählten nicht linearen Dämpfung des DGLs liegen und/oder an einem möglichen Fehler beim Lösungsverfahren des Problems.

## Literatur

- [1] Wolfgang Demtröder, *Experimentalphysik 1, Mechanik und Wärme*, SpringerSpektrum, Auflage 8, 2018, Seite 334.