Hochschule für Technik Zürich

BSC Informatik

Projektarbeit

4. Semester

Die schwingende Saite

Autoren:
Farhan Fayyaz
Pascal Murbach

Betreuer:
Albert Heuberger
Lars Kruse

14. Juni 2011

Diese Dokumentation beschreibt eine Lösung des Problems der schwingenden Saite. Die Aufgabe war, ein Programm zu schreiben, das die Schwingung einer Saite berechnet und grafisch darstellt. Die Umsetzung des Programms wurde in Java geschrieben.

Zur Berechnung der momentanen Funktion, welche die Saite zu einem beliebigen Zeitpunkt beschreibt, wurde die analytische Lösung der Differenzialgleichung zur schwingenden Saite verwendet. Da diese die Integration von beliebigen Funktionen verlangt, mussten numerische Verfahren eingesetzt werden. Die Berechnung der Integrale wurde mit Hilfe des adaptiven Simpson-Algorithmus implementiert.

Dieser rekursive Algorithmus ist sehr ressourcen-intensiv, weshalb eine komplette Periode der Saitenschwingung berechnet werden muss, bevor sie dargestellt wird. Die Funktionen, welche die Saite zu allen berechneten Zeitpunkten beschreibt, werden in eine Matrix abgefüllt und zur Darstellung daraus wieder ausgelesen.

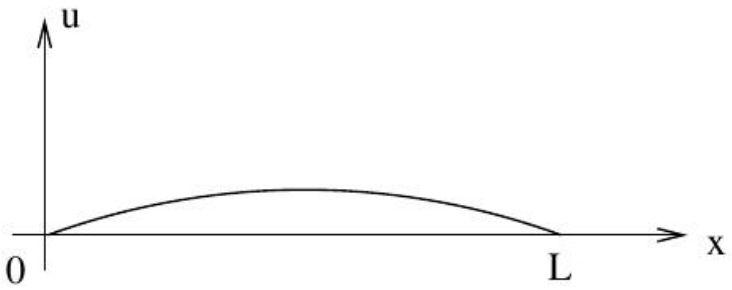
Unser hauptsächliches Problem war die Performance. Auch bei unserer definitiven Lösung braucht die Berechnung bei einer hohen Anzahl Oberschwingungen und einer kleinen Fehlertoleranz im Simpson-Algorithmus viel Zeit.

Inhaltsverzeichnis

1 Einleitung

1.1 Aufgabenstellung

Die Auslenkung $u(x,t)$ einer eingespannten Saite soll in einer objektorientierten Programmiersprache berechnet und dargestellt werden. Die Parameter der Simulation sollen ebenfalls über die grafische Oberfläche eingegeben werden können.



1.2 Vorgehen

Das Vorgehen zur Lösung der gestellten Aufgabe kann mit den aufgelisteten Schritten grob umrissen werden:

1. Erarbeiten der mathematischen Grundlagen
2. Entwerfen eines Software-Designs zur Umsetzung
3. Umsetzung in Java
4. Erweitern und Verbessern des Codes

1.3 Schwierigkeiten

Als echte Knacknuss hat sich das Berechnen der Saite mit angemessenem Zeitaufwand erwiesen. Da zur Lösung teilweise rekursive Algorithmen eingesetzt wurden, lässt die Performance zu wünschen übrig.

2 Mathematische Grundlagen

2.1 Differentialgleichung

Als Grundlage des Problems steht folgende Differentialgleichung:

$$u_{tt} = c^2 u_{xx}, \qquad 0 \leq x \leq L, \;\; t \geq 0, \;\; c \in \mathbb{R}^+ \tag{2.1}$$

Da die Saite eingespannt ist, müssen folgende Randbedingungen erfüllt sein:

$$u(0,t) = 0, \qquad t \geq 0 \tag{2.2}$$
$$u(L,t) = 0, \qquad t \geq 0 \tag{2.3}$$

Zusätzlich müssen Anfangsbedingungen gesetzt werden:

$$u(x,0) = f(x), \qquad 0 \leq x \leq L \tag{2.4}$$
$$u_t(x,0) = g(x), \qquad 0 \leq x \leq L \tag{2.5}$$

Dabei ist c eine fixe Konstante gleich der Ausbreitungs-Geschwindigkeit der Welle. Für eine Schallwelle in der Luft wäre c also etwa 343 m/s (Schallgeschwindigkeit). $f(x)$ und $g(x)$ beschreiben die Urposition der Saite, also bevor sie losgelassen wird und schwingt. $f(x)$ könnte beispielsweise eine Dreiecksfunktion sein.

2.2 Lösung der Differentialgleichung

Die Differentialgleichung 2.1 besitzt folgende Lösung: [1]

$$u(x,t) = \sum_{n=1}^{\infty} (B_n \, cos\frac{cn\pi}{L}t + B_n^* \, sin\frac{cn\pi}{L}t) \, sin\frac{n\pi}{L}x \tag{2.6}$$

mit den Fourierkoeffizienten

$$B_n = \frac{2}{L}\int_0^L f(x) sin\frac{n\pi}{L}x \, \mathrm{d}x \tag{2.7}$$

und

$$B_n^* = \frac{2}{cn\pi}\int_0^L g(x) sin\frac{n\pi}{L}x \, \mathrm{d}x \tag{2.8}$$

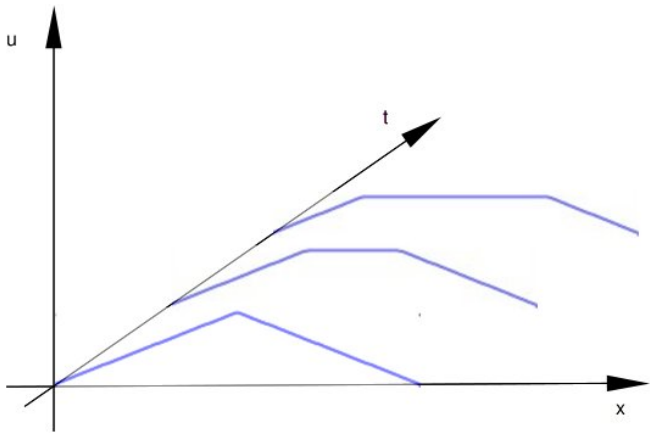


Abbildung 2.1: Die Saite zu den Zeitpunkten t = 1,2,3

Formel 2.6 lässt sich grundsätzlich in zwei Teile zerlegen. Zum einen die Fourierreihe

$$\sum_{n=1}^{\infty}(B_n \, cos\frac{cn\pi}{L}t + B_n^* \, sin\frac{cn\pi}{L}t) \tag{2.9}$$

welche die Funktion der Saite zum Zeitpunkt t beschreibt, und zum anderen die Schwingung

$$sin\frac{n\pi}{L}x \tag{2.10}$$

welche die zeitliche Dimension beschreibt. Das Ganze lässt sich am besten in drei Dimensionen darstellen.

2.2.1 Oberschwingungen

Wird die Saite in Schwingung gebracht, wird nicht nur eine Frequenz angeregt, sondern auch Vielfache der Grundfrequenz mit geringerer Amplitude. Diese Vielfache werden harmonische Oberschwingungen genannt und in der Laufvariable n der Fourierreihe 2.9 beschrieben. Bei Musikinstrumenten bestimmen die Obertöne im Wesentlichen die Klangfarbe. Theoretisch hat eine schwingende Saite undendlich viele Oberschwingungen, was sich natürlich nicht implementieren lässt. Die Anzahl Oberschwingungen lässt sich in unserer Software über die GUI (“Harmonic“) einstellen. Dabei lässt sich die Auswirkung der Oberschwingungen auf die Form der Saite zeigen.

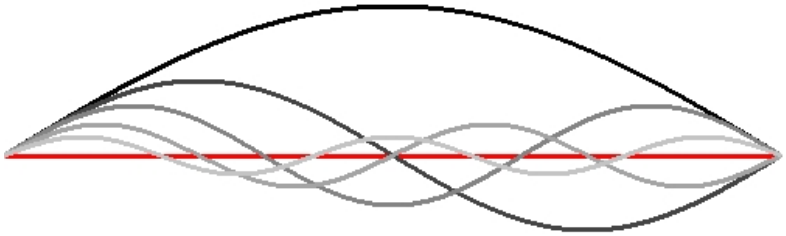


Abbildung 2.2: Die ersten vier Oberschwingungen

2.3 Adaptiver Simpson-Algorithmus

Zum Integrieren der beiden Fourierkoeffizienten B_n und B_n^* muss ein numerisches Verfahren verwendet werden. Nach ersten erfolglosen Versuchen mit dem Integrieren nach der Trapezregel, welche offenbar für die Darstellung zu ungenaue Resultate liefert, ist die Wahl auf den adaptiven Simpson-Algorithums gefallen. Dieser ist rekursiv und lässt sich auf eine gewünschte Fehlertoleranz einstellen. Nachteilig ist sein enormer Rechenaufwand.

Grundsätzlich verwendet der adaptive Simpson-Algorithmus eine Fehlerabschätzung der Simpson-Regel.

$$Sf : \int_a^b f(x)\,\mathrm{d}x \approx \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right) \tag{2.11}$$

Wenn der Fehler eine gewisse Toleranz überschreitet, wird der Intervall in zwei Subintervalle zerlegt, auf welchen dann die Simpson-Methode rekursiv angewendet wird.

Ein Kriterium zum Abbruch des Algorithmus nach J.N.Lyness ist:

$$|Sf(a,c) + Sf(c,b) - Sf(a,b)|/15 \quad < \varepsilon \tag{2.12}$$

wobei $[a,b]$ das Intervall mit dem Mittelpunkt c ist. $S(a,b), S(a,c), S(c,b)$ sind die Annäherungen durch die Simpson-Regel und ε ist die benötigte Toleranz.

Der Algorithmus wurde genau so in der Klasse VibraString.java implementiert. [2]

3 Umsetzung in Java

3.1 Design

Grundsätzlich wird die Saitenschwingung in zwei Threads berechnet. Der eine ist für die Berechnung der momentanen Funktion der Saite zum Zeitpunkt t verantwortlich, der andere für die Darstellung.

3.1.1 VibraString

Die Klasse "VibraString" ist das Herzstück des Programms. Sie stellt das Modell der Saite dar und umfasst alle Methoden und Parameter, die zur Berechnung der Saite notwendig sind.

Damit die Saite berechnet werden kann, muss sie diskretisiert werden. Dies geschieht durch die Abbildung der Saite auf eine Matrix, welche die Länge L der Saite aufnimmt und in die gewünschte Schrittweite h unterteilt. Die Matrix hat also

$$cols = \frac{L}{h} \tag{3.1}$$

Spalten. Somit beschreibt eine $1 \times col$ Matrix den Zustand der Saite zum Zeitpunkt t.

Soll nun die ganze Saitenschwingung schön und ohne Verzögerung dargestellt werden, muss eine ganze Periode im Voraus berechnet werden. Dazu muss die Periodendauer T bekannt sein. Sie berechnet sich aus der Frequenz f der Grundschwingung:

$$f = \frac{c}{2L} \tag{3.2}$$

$$T = \frac{1}{f} = \frac{2L}{c} \tag{3.3}$$

Daraus wiederum, zusammen mit der gewünschten zeitlichen Auflösung $precision$, lässt sich die Anzahl Reihen der Matrix, welche die komplette Saitenschwingung darstellen, berechnen:

$$rows = \frac{T}{precision} = \frac{2L}{c * precision} \tag{3.4}$$

Somit lässt sich die gesamte Periode der Saitenschwingung in einer $\mathbb{R}^{row \times col}$ Matrix zusammenfassen.

3.1.2 Dämpfung der Schwingung

Um die Dämpfung der schwingenden Saite darstellen zu können, muss die Matrix mit einem Skalar s multipliziert werden. Dieser ist vom Zeitpunkt t abhängig und berechnet sich wie folgt:

$$s = e^{\frac{-t}{\tau}} \qquad t \geq 0 \tag{3.5}$$

$$\tau = \frac{2m}{d} \tag{3.6}$$

Wobei m die Masse der Saite und d die Dämpfungskonstante ist. Die Matrix wird nach jeder Periode mit diesem Skalar multipliziert. [3]

3.1.3 IMathFunction

Das Interface IMathFunction eröffnet die Möglichkeit, beliebige Startfunktionen der Saite in die Simulation zu laden. Der Klasse VibraString werden zwei IMathFunctions übergeben, nämlich f(x) und g(x). [1]

3.2 Bedienung

Über die grafische Oberfläche können folgende Parameter verändert werden:

1. Startfunktion f
2. Startfunktion g
3. Länge der Saite L
4. Ausbreitungsgeschwindigkeit der Welle c
5. Schrittweite $XPrec$ für $x \leq 0 \leq L$
6. Zeitliche Auflösung $Tprec$
7. Anzahl Oberschwingungen $Harmonic$
8. Dämpfung $Decay$
9. Dämpfungskonstante $DecayPrec$
10. Fehlerabschätzungstoleranz ε (Eps) (Adaptiver Simpson-Algorithmus)

[1] siehe Kap. 2

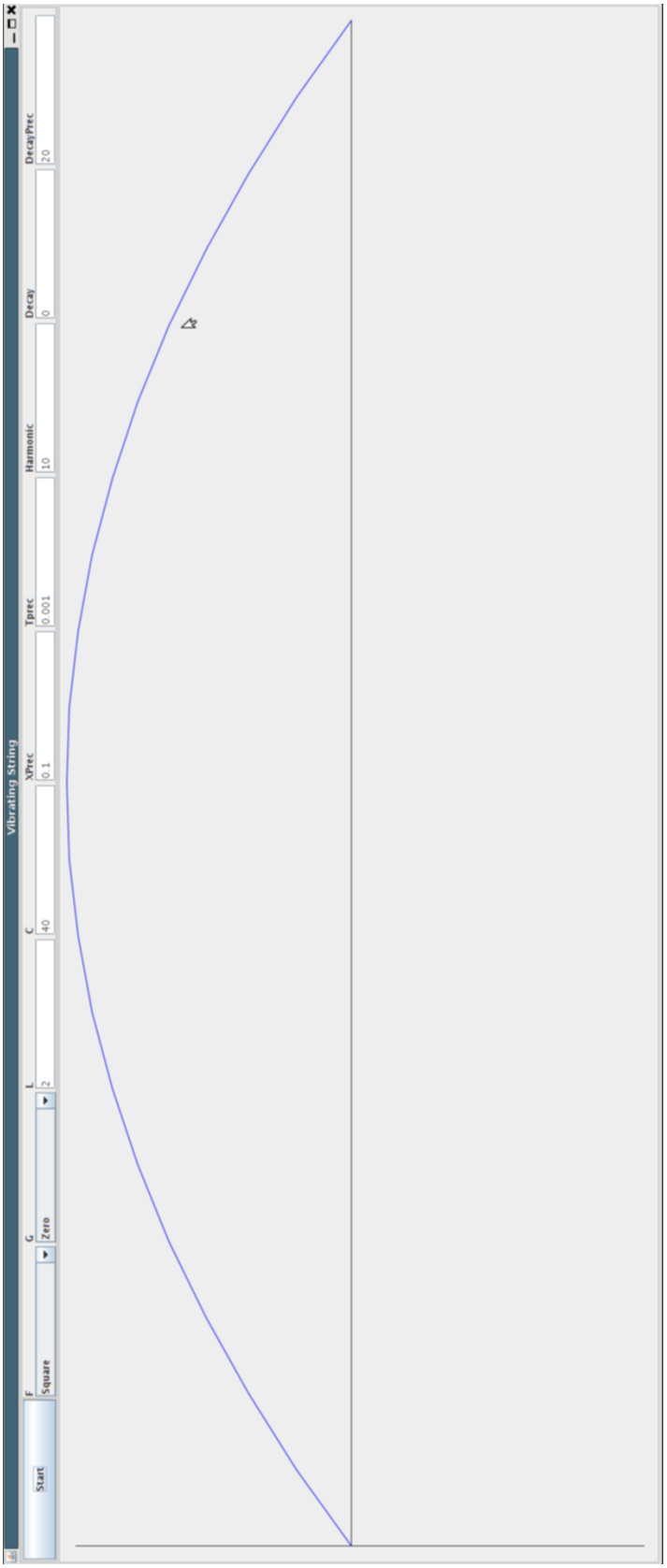


Abbildung 3.1: Screenshot der Anwendung

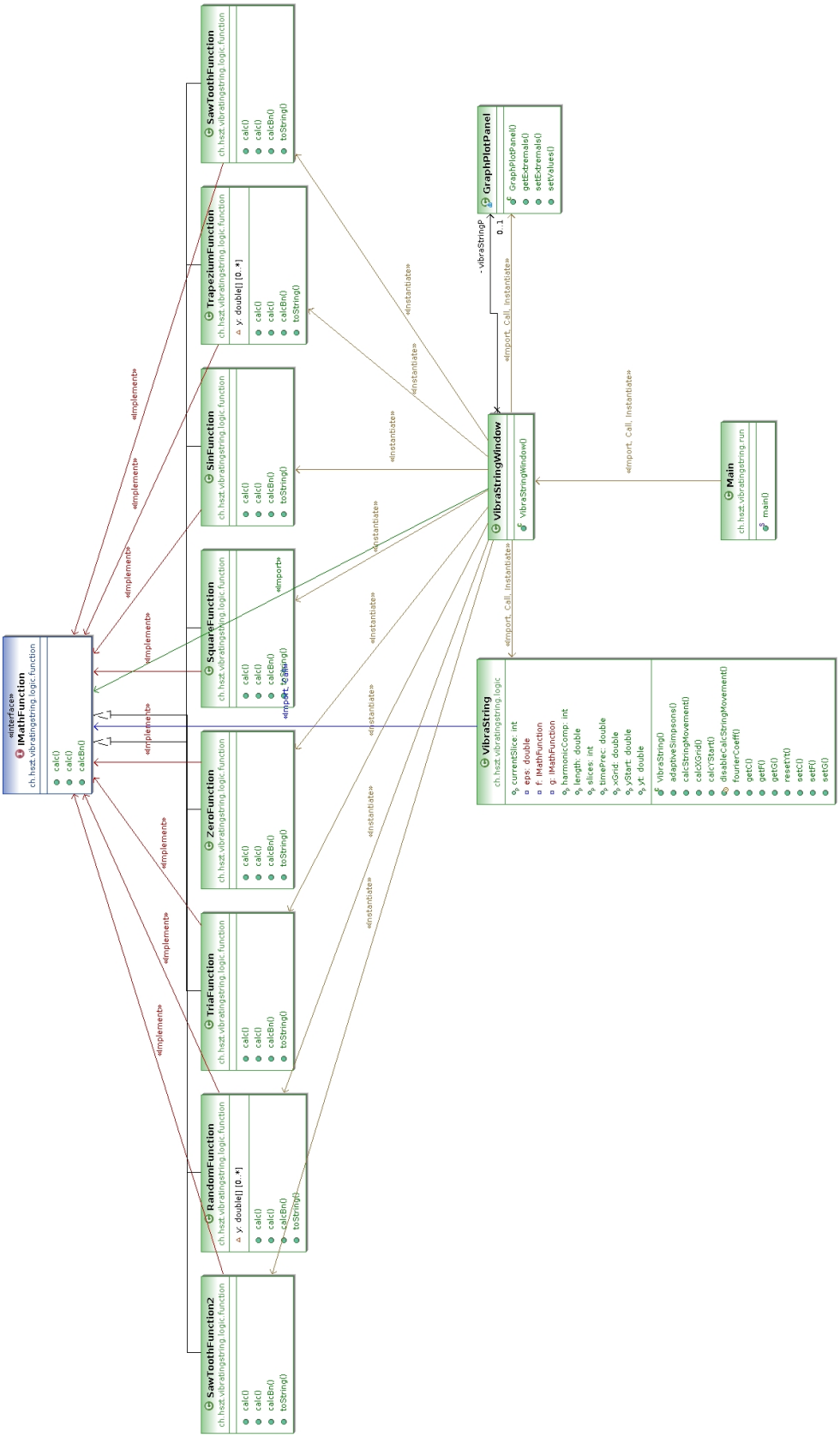


Abbildung 3.2: UML-Diagramm der Anwendung

Literaturverzeichnis

[1] Mike Land and Tara Puzin. The vibrating string — applications of differential equations. http://online.redwoods.cc.ca.us/instruct/darnold/deproj/sp03/taramike/presentation.pdf, 2003.

[2] Wikipedia. Adaptive simpson's method — wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Adaptive_Simpson%27s_method&oldid=409153155, 2011. [Online; accessed 12-June-2011].

[3] Wikipedia. Schwingung — wikipedia, die freie enzyklopädie. http://de.wikipedia.org/w/index.php?title=Schwingung&oldid=89766363, 2011. [Online; Stand 12. Juni 2011].