DIPLOMARBEIT

Bewertung Asiatischer Optionen mittels Fouriertransformation

am Institut für Statistik

unter Anleitung von Ao.Univ.-Prof.Dipl.-Ing.Dr.techn. Wolfgang Müller

durch

Lehner Edith

Matr-Nr. 0330123

im Sommersemester 2009

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst,andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

November 2009,

Lehner Edith

Ich möchte vorallem Ao.Univ.-Prof.Dipl.-Ing.Dr.techn. Wolfang Müller für die wertvolle Betreuung und Unterstützung beim Erstellen dieser Arbeit danken. Ein großer Dank gilt meinen Eltern und meinem Bruder die mich, während des ganzen Studiums unterstützt haben. Weiters möchte ich meinem Freundeskreis danken, der über das Studium hinaus mein Leben sehr bereichert hat.

Inhaltsverzeichnis

Einleitung

Asiatische Optionen sind spezielle exotische Optionen, deren Auszahlungswert nicht nur vom Schlußkurs des Basiswertes abhängig ist, sondern auch von der Kursentwicklung bis zu diesem Zeitpunkt. Diese Eigenschaft erschwert die Bewertung solcher Optionen. Im Allgemeinen kann der faire Preis nur durch Monte Carlo Simulation oder durch numerische Verfahren bestimmt werden.

In dieser Arbeit wird eine Methode vorgestellt, bei der der Preis einer asiatischen Option durch numerische Fouriertransformation berechnet wird. Der Algorithmus zur Berechnung der numerischen Fouriertransformation basiert auf einer Arbeit von Peter den Iseger.

An den zugrunde liegenden Preisprozeß wird nur die Forderung von unabhängigen Returns pro Periode gestellt. Daher ist die Verwendung dieses Verfahrens nicht nur im Black-Scholes-Modell, sondern für beliebige Lèvy-Modelle, möglich.

In Kapitel 1 werden die mathematischen Grundlagen des Verfahrens von Den Iseger vorgestellt. Spezielle Eigenschaften der Legendre Polynome werden in Kapitel 2 beschrieben. Analoge Beschreibungen für Polynome im Bildbereich der Fouriertransformierten werden in Kapitel 3 gegeben. Kapitel 4 führt in die Theorie der Fouriertransformation ein. Insbesondere wird die Poisson'schen Summenformel besprochen. Die detaillierte Beschreibung des Algorithmus zur numerischen Fouriertransformation folgt in Kapitel 5. Den Abschluß bildet Kapitel 6, in dem zunächst eine sehr kurze Einführung in das Thema exotische Optionen gegeben wird, gefolgt von einer Bepreisungsmethode für Asiatische Optionen. Weiters werden numerische Testresultate des Algorithmus zur Optionspreisberechnung im Black-Scholes-Modell angegeben.

Kapitel 1

Orthogonale Polynome

In diesem Abschnitt werden die wichtigsten Eigenschaften von orthogonalen bzw. orthonormalen Polynomen behandelt, die für spätere Berechnungen relevant sind. Als Referenz, siehe [11] ,[23] sowie [18].

1.1 Allgemeine Definition

Sei μ ein endliches Maß auf $\mathbb{R}$ das nicht auf endlich vielen Punkten konzentriert ist. $L^2(\mu)$ bezeichne den Raum der bezüglich μ quadratisch integrierbaren komplexwertigen Funktionen. Für zwei Funktionen $f, g \in L^2(\mu)$ sei

$$\langle f, g\rangle := \int_{\mathbb{R}} f(t)\overline{g(t)}\, d\mu(t). \tag{1.1}$$

Weil aus $\|f\|^2 = \langle f, f\rangle = 0$ nicht $f = 0$ folgt, ist $\langle \cdot, \cdot\rangle$ im allgemeinen kein echtes Skalarprodukt. Um ein echtes Skalarprodukt zu erhalten, muss zum Faktorraum der Klassen $[f]$ übergegangen werden, wobei $[f] = [g]$, wenn f und g bis auf eine $\mu-$Nullmenge übereinstimmen.

Man beachte allerdings, dass für positive Polynome p, aus $\langle p, 1\rangle = 0$ stets $p = 0$ folgt. (aus $p = 0$ $(\mu - f.s)$ folgt $p = 0$, weil μ andernfalls auf den endlich vielen Nullstellen von p konzentriert ist). Insbesonders gilt $\|p\| > 0$ für $p \neq 0$.

Zwei Funktionen $f, g \in L^2(\mu)$ heißen orthogonal, wenn $\langle f, g\rangle = 0$ gilt. Eine Folge von Funktionen $\{f_n, n \geq 0\} \in L^2(\mu)$ heißt orthonormal, wenn

$$\langle f_n, f_m\rangle = \int_{\mathbb{R}} f_n(t)\overline{f_m(t)}\, d\mu(t) = \delta_{nm}, \quad \text{mit} \quad \delta_{nm} = \begin{cases} 1 & n = m \\ 0 & n \neq m \end{cases},$$

gilt. Insbesonders gilt

$$\|f_n\| = 1 \quad (n \geq 0).$$

In den kommenden Abschnitten, werden Folgen $(p_n)_{n\geq 0}$ **reeller** orthogonaler Polynome mit $deg(p_n) = n$ betrachtet. Dazu setzen wir stets voraus, dass $L^2(\mu)$ alle Polynome $(x^n)_{n\geq 0}$ enthält. Mit Hilfe des Gram-Schmidt'schen Orthogonalisierungsverfahrens (siehe [18]) lässt sich stets eine solche Folge orthonormaler Polynome aus der Polynomfolge $(x^n)_{n\geq 0}$ konstruieren. Die so erzeugten p_n sind bis auf einen nichtverschwindenden multiplikativen Faktor eindeutig bestimmt.

1.2 Die Nullstellen orthogonaler Polynome

Sei $(p_n)_{n\leq 1}$ eine Folge reeller orthogonaler Polynome mit $deg(P_n) = n$.
Satz 1.2.1. *Die Nullstellen von $p_n(x)$ sind reell und einfach. Ist $supp(\mu) \subset [a, b]$, dann liegen alle Nullstellen im Intervall $[a, b]$.*

Beweis. Der Beweis basiert auf der Orthogonalität der Polynome bezüglich des in (1.1) definierten Skalarproduktes. Da

$$\langle p_n, 1\rangle = \int_{\mathbb{R}} p_n(x)\, d\mu(x) = 0 \quad \text{für } n \geq 1$$

gilt, muss $p_n(x)$ mindestens an einem Punkt das Vorzeichen wechseln. Seien $x_1, x_2, \ldots, x_l$ mit $l \leq n$, diese Punkte. Das Produkt $p_n(x) \prod_{i=1}^{l}(x - x_i)$ hat konstantes Vorzeichen auf dem gesamten Invervall. Für l < n gilt auf Grund der Orthogonalität

$$\int_{\mathbb{R}} p_n(x) \prod_{i=1}^{l}(x - x_i)\, d\mu(x) = 0.$$

Das ist nicht möglich, da der Integrand konstantes Vorzeichen hat. Somit gilt l=n. □

Satz 1.2.2. *Die orthogonalen Polynome $p_n(x) = c_n x^n + c'_n x^{n-1} + \ldots$ mit $c_n \neq 0$, erfüllen die Rekursion*

$$\begin{aligned} p_{n+1}(x) &= (A_n x + B_n)p_n(x) - C_n p_{n-1}(x), \qquad (n \geq 0) \\ p_{-1}(x) &= 0, \qquad p_0(x) = c_0. \end{aligned} \tag{1.2}$$

Dabei sind A_n, B_n und C_n reelle Konstante. Es gilt

$$A_n = \frac{c_{n+1}}{c_n}, \quad B_n = A_n\left(\frac{c'_{n+1}}{c_{n+1}} - \frac{c'_n}{c_n}\right), \quad C_n = \frac{c_{n+1}c_{n-1}}{c_n^2}\frac{\langle p_n, p_n\rangle}{\langle p_{n-1}, p_{n-1}\rangle}. \tag{1.3}$$

Beweis. Zuerst wird der Koeffizient $A_n = \frac{c_{n+1}}{c_n}$ gewählt, um durch $p_{n+1} - A_n x p_n$ ein Polynom vom Grad n zu erzeugen.

$$p_{n+1} - A_n x p_n = \sum_{k=1}^{n} a_k p_k.$$

Auf Grund der Orthogonalität folgt für die Koeffizienten $a_l = 0$ für $l < n-1$. Um den Koeffizienten a_{n-1} zu berechnen, wird in der obigen Gleichung das Skalarprodukt mit p_{n-1} gebildet. Es gilt

$$\begin{aligned}
-\frac{c_{n+1}}{c_n}\langle x p_n, p_{n-1}\rangle &= a_{n-1}\langle p_{n-1}, p_{n-1}\rangle, \\
-\frac{c_{n+1}}{c_n}\langle p_n, x p_{n-1}\rangle &= a_{n-1}\langle p_{n-1}, p_{n-1}\rangle, \\
-\frac{c_{n+1}c_{n-1}}{c_n^2}\langle p_n, p_n\rangle &= a_{n-1}\langle p_{n-1}, p_{n-1}\rangle.
\end{aligned}$$

Für den Koeffizienten a_n gilt unter Verwendung der Orthogonalität

$$\begin{aligned}
c_{n+1}x^{n+1} + c'_{n+1}x^n + \cdots - \frac{c_{n+1}}{c_n}(c_n x^{n+1} + c'_n x^n + \dots) &= a_n p_n - a_{n-1}p_{n-1} \\
c_{n+1}\left(\frac{c'_{n+1}}{c_{n+1}} - \frac{c'_n}{c_n}\right)\langle x^n, x^n\rangle &= a_n c_n \langle x^n, x^n\rangle \\
A_n\left(\frac{c'_{n+1}}{c_{n+1}} - \frac{c'_n}{c_n}\right) &= a_n.
\end{aligned}$$

Zusammen führen diese Ergebnisse auf

$$p_{n+1}(x) - A_n x p_n(x) = A_n\left(\frac{c'_{n+1}}{c_{n+1}} - \frac{c'_n}{c_n}\right)p_n(x) - \frac{c_{n+1}c_{n-1}}{c_n^2}\frac{\langle p_n, p_n\rangle}{\langle p_{n-1}, p_{n-1}\rangle}p_{n-1}(x)$$

und somit auf (1.2). □

Satz 1.2.3 (Formel von Christoffel - Darboux)**.** *Sei* $\{p_k(x), k \geq 0\}$ *eine Folge von orthogonalen Polynomen, dann gelten folgende Gleichungen*

$$\sum_{k=0}^{n} p_k(x)p_k(t) = \frac{c_n}{c_{n+1}}\frac{p_{n+1}(x)p_n(t) - p_n(x)p_{n+1}(t)}{x-t} \quad (x \neq t), \tag{1.4}$$

und

$$\sum_{k=0}^{n} |p_k(x)|^2 = \frac{c_n}{c_{n+1}}(p'_{n+1}(x)p_n(x) - p'_n(x)p_{n+1}(x)). \tag{1.5}$$

Beweis. Die Formel von Christoffel - Darboux kann aus der Rekursionsformel (1.2) hergeleitet werden.

$$\begin{aligned} p_{k+1}(x)p_k(t) - p_k(x)p_{k+1}(t) &= ((A_k x + B_k)p_k(x) - C_k p_{k-1}(x))p_k(t) \\ &\quad -p_k(x)((A_k t + B_k)p_k(t) - C_k p_{k-1}(t)) \\ &= A_k(x-t)p_k(x)p_k(t) + C_k(p_k(x)p_{k-1}(t) \\ &\quad -p_{k-1}(x)p_k(t)) \end{aligned}$$

Mit den Koeffizienten aus (1.3) folgt

$$p_k(x)p_k(t) = \frac{c_k}{c_{k+1}} \frac{p_{k+1}(x)p_k(t) - p_k(x)p_{k+1}(t)}{x-t} - \frac{c_{k-1}}{c_k} \frac{p_k(x)p_{k-1}(t) - p_{k-1}(t)p_k(x)}{x-t}.$$

Summation über $0 \leq k \leq n$ liefert (1.4). Diese Gleichung gilt auch für n=0, wenn c_{-1} beliebig definiert wird. (1.5) folgt aus (1.4) indem man den Grenzübergang $t \to x$ durchführt. □

Die Nullstellen des Polynoms p_n können (auf numerisch stabile Weise) als die Eigenwerte einer reellen, symmetrischen nxn Matrix bestimmt werden. Dazu wird zunächst ein Multiplikationsoperator definiert

$$\mathcal{M}f(t) := t \cdot f(t). \tag{1.6}$$

Dieser Operator bildet den von den Polynomen $(p_n)_{n\geq 0}$ aufgespannten linearen Raum $L^2(\mu) = \overline{span(\{p_n | n \geq 0\})}$ (siehe Abschnitt 1.4) auf sich selbst ab. Bezüglich der Basis $(p_n)_{n\geq 0}$ wird $\mathcal{M}$ durch eine unendlichdimensionale Matrix mit den Einträgen

$$M := (\langle \mathcal{M}p_k, p_j \rangle)_{k,j\geq 0}$$

dargestellt. Wegen

$$\langle \mathcal{M}p_k, p_j \rangle = \int_{\mathbb{R}} x p_k(x) p_j(x)\, d\mu(x) = \langle \mathcal{M}p_j, p_k \rangle$$

ist M symmetrisch.

Bezeichne P_n die orthogonale Projektion von $L^2(\mu)$ auf π_{n-1}, den Raum der Polynome mit komplexen Koeffizienten vom Grad $\leq n-1$, dann bildet der lineare Operator $\mathcal{M}_n := P_n\mathcal{M}$, π_{n-1} auf π_{n-1} ab . Bezüglich der Basis $\{p_0, \ldots, p_{n-1}\}$ wird $\mathcal{M}_n$ durch die Matrix

$$M_n := (\langle \mathcal{M}_n p_k, p_j \rangle)_{0\leq k,j\leq n-1} \tag{1.7}$$

beschrieben. M_n ist die Tridiagonalmatrix

$$M_n = \begin{bmatrix} \alpha_0 & \beta_0 & 0 & \cdots & 0 \\ \beta_0 & \alpha_1 & \beta_1 & \ddots & \vdots \\ 0 & \beta_1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \alpha_{n-2} & \beta_{n-2} \\ 0 & \cdots & 0 & \beta_{n-2} & \alpha_{n-1} \end{bmatrix}. \tag{1.8}$$

mit

$$\alpha_k = -\left(\frac{c'_{k+1}}{c_{k+1}} - \frac{c'_k}{c_k}\right)\langle p_k, p_k\rangle = -\frac{B_k}{A_k}\langle p_k, p_k\rangle, \tag{1.9}$$

$$\beta_k = \frac{c_k}{c_{k+1}}\langle p_{k+1}, p_{k+1}\rangle = \frac{1}{A_k}\langle p_{k+1}, p_{k+1}\rangle. \tag{1.10}$$

Das sieht man wie folgt. Zunächst folgt aus der Orthogonalität $\langle \mathcal{M}p_k, p_j\rangle = 0$ für $k < j-1$. Für die Elemente der Nebendiagonale gilt

$$\langle \mathcal{M}_n p_k, p_{k+1}\rangle = \left\langle \frac{c_k}{c_{k+1}} p_{k+1}, p_{k+1}\right\rangle = \frac{c_k}{c_{k+1}}\langle p_{k+1}, p_{k+1}\rangle = \beta_k.$$

Die Elemente der Hauptdiagonale erhält man durch

$$\langle \mathcal{M}_n p_k, p_k\rangle = \left\langle c_k\left(\frac{p_{k+1}}{c_{k+1}} - \frac{c'_{k+1}}{c_k c_{k+1}} p_k\right) + \frac{c'_k}{c_k} p_k, p_k\right\rangle = -\left(\frac{c'_{k+1}}{c_{k+1}} - \frac{c'_k}{c_k}\right)\langle p_k, p_k\rangle = \alpha_k.$$

Satz 1.2.4. *Sei $(p_n)_{n\geq 0}$ eine Folge orthonormaler Polynome. Die Nullstellen $\{x_k : k = 0, 1, \ldots, n-1\}$ des Polynoms p_n sind die Eigenwerte der Operatormatrix M_n. Der zu x_k gehörende, bis auf einen multiplikativen Faktor eindeutige, Eigenvektor ist $\vec{p}(x_k) := [p_0(x_k), p_1(x_k), \ldots, p_{n-1}(x_k)]^t$.*

Beweis. Für orthonormale Polynome gilt zunächst

$$\alpha_k = -\frac{B_k}{A_k}, \qquad \beta_k = \frac{1}{A_k}.$$

Aufgrund des Satzes 1.2.2 gilt für $0 \leq k \leq n-1$

$$\begin{aligned} \beta_k p_{k+1}(x) + \alpha_k p_k(x) + \beta_{k-1} p_{k-1}(x) &= (\beta_{k-1} - \beta_k C_k) p_{k-1}(x) + \beta_k A_k x p_k(x) \\ &\quad + (\alpha_k + \beta_k B_k) p_k(x) \\ &= x p_k(x). \end{aligned}$$

Daraus folgt

$$M_n\vec{p}(x) = x\vec{p}(x) - \beta_{n-1}\begin{bmatrix} 0 \\ \vdots \\ 0 \\ p_n(x) \end{bmatrix}.$$

Speziell folgt $M_n\vec{p}(x_k) = x_k\vec{p}(x_k)$ für jede Nullstelle x_k von p_n. □

1.3 Gauß Quadratur

Mit Hilfe von Quadraturformeln ist es möglich Integrale numerisch zu berechnen bzw. zu approximieren. Ein bekanntes Verfahren ist die Gauß Quadratur. Die Gauß Quadratur vom Grad n ermöglicht es Polynome bis zum Grad 2n - 1 exakt zu integrieren. Mit Hilfe der im Abschnitt zuvor eingeführten Operatormatrix ist es weiters möglich eine Darstellung der Gauß'schen Quadraturformel durch eben diese Matrix zu finden. Diese Darstellung ermöglicht eine schnelle Berechung der Gauß Quadratur. Näheres zum Thema Gauß Quadratur kann in [23], [25] und [5] nachgelesen werden.

Sei $(p_n)_{n\geq 0}$ eine Folge reeller orthonormaler Polynome mit $deg(p_n) = n$.

Satz 1.3.1. *Sind $x_0 < x_1 < \cdots < x_{n-1}$ die Nullstellen von p_n, dann gibt es eindeutig bestimmte reelle Zahlen $w_0, \ldots, w_{n-1}$ (die Christoffel Zahlen), sodass für zwei Polynome f und g mit, $deg(f\overline{g}) \leq 2n-1$, stets*

$$\langle f, g\rangle = \langle f, g\rangle_n \qquad (f \cdot \overline{g} \in \pi_{2n-1}), \tag{1.11}$$

gilt, wobei

$$\langle f, g\rangle_n := \sum_{k=0}^{n-1} w_k f(x_k)\overline{g}(x_k).$$

Weiters gibt es keine reellen Zahlen x_i und w_i für $i = 0, \ldots, n-1$, sodass (1.11) auch für alle Polynome f und g mit $f \cdot \overline{g} \in \pi_{2n}$ gilt.

Beweis. Um (1.11) zu beweisen wird die Interpolationstechnik von Lagrange verwendet. Die Basis dieser Interpolationstechnik bilden die Lagrange Polynome. Diese sind gegeben durch

$$l_i(x) = \prod_{\substack{j=0 \\ i\neq j}}^{n-1} \frac{x - x_j}{x_i - x_j} = \frac{p_n(x)}{p_n'(x_i)(x - x_i)} \qquad (0 \leq j \leq n-1). \tag{1.12}$$

Das Interpolationspolynom $L(x)$ vom Grad $n-1$, dass mit $f(x)\overline{g}(x)$ in den Punkten $\{x_i : 0 \leq i < n\}$ übereinstimmt, kann mit Hilfe der Lagrange Polynome angegeben werden,

$$L(x) = \sum_{i=0}^{n-1} f(x_i)\overline{g}(x_i) l_i(x). \tag{1.13}$$

Da das Produkt $f(x)\overline{g}(x)$ höchstens Grad $2n-1$ hat, gilt $f(x)\overline{g}(x) = p_n(x)r(x) + L^*(x)$ mit $r, L^* \in \pi_{n-1}$. Da $f\overline{g}$ und L^* in den Punkten $\{x_i : 0 \leq i < n\}$ übereinstimmen gilt $L^* = L$, und es folgt

$$f(x)\overline{g}(x) = p_n(x)r(x) + L(x) \qquad r(x), L(x) \in \pi_{n-1}.$$

Somit gilt

$$\begin{aligned} \langle f, g\rangle &= \int_{\mathbb{R}} p_n(x)r(x)\, d\mu(x) + \int_{\mathbb{R}} L(x)\, d\mu(x) \\ &= \int_{\mathbb{R}} L(x)\, d\mu(x) = \sum_{i=0}^{n-1} f(x_i)\overline{g}(x_i) \int_{\mathbb{R}} l_i(x)\, d\mu(x). \end{aligned}$$

Das erste Integral auf der rechten Seite verschwindet auf Grund der Orthogonalität.

Daraus folgt (1.11) mit

$$w_i = \int_{\mathbb{R}} l_i(x)\, d\mu(x). \tag{1.14}$$

Um die letze Aussage des Satzes zu beweisen, nehmen wir an es existieren Zahlen w_i und x_i sodass (1.11) auch für alle $f\overline{g} \in \pi_{2n}$ richtig ist. Für das Polynom

$$f^*(x)\overline{g}^*(x) := \prod_{j=0}^{n-1} (x - x_j)^2 \in \pi_{2n}$$

führt dies sofort auf den Widerspruch

$$0 < \langle f^*, g^*\rangle = \sum_{k=0}^{n-1} w_k f^*(x_k)\overline{g}^*(x_k) = 0.$$

□

Satz 1.3.2. *Ist $\vec{p}(x_i)$ der Eigenvektor von M_n zum Eigenwert x_i, aus Satz 1.2.4, dann gilt für die Christoffel-Zahlen*

$$w_i = \left(p_0(x_i)^2 + p_1(x_i)^2 + \cdots + p_{n-1}(x_i)^2\right)^{-1} = \frac{1}{|\vec{p}(x_i)|^2} \quad (0 \leq i < n). \tag{1.15}$$

Beweis. Setzt man in (1.4) $t = x_i$ folgt mit (1.12)

$$\sum_{k=0}^{n} p_k(x)p_k(x_i) = -\frac{c_n}{c_{n+1}}\frac{p_n(x)p_{n+1}(x_i)}{x - x_i} = -\frac{c_n}{c_{n+1}}p_n'(x_i)p_{n+1}(x_i)l_i(x).$$

Integriert man beide Seiten nach $d\mu(x)$, erhält man mit (1.14)

$$1 = -\frac{c_n}{c_{n+1}}p_{n+1}(x_i)p_n'(x_i)w_i. \tag{1.16}$$

Aus (1.2) folgt

$$p_{n+1}(x_i) = -\frac{c_{n+1}c_{n-1}}{c_n^2}p_{n-1}(x_i).$$

Zusammen mit (1.16) und (1.5) liefert das

$$\frac{1}{w_i} = \frac{c_{n-1}}{c_n}p_{n-1}(x_i)p_n'(x_i) = \sum_{k=0}^{n-1}\left((p_k(x_i)\right)^2.$$

□

Nun folgt die Darstellung der Gauß Quadratur durch die zuvor eingeführte Operatormatrix.

Satz 1.3.3. *Sei f eine beliebige Funktion in $L^2(\mu)$. Sei $\vec{f} = [f(x_0), \ldots, f(x_{n-1})]^t$ der Vektor der Funktionswerte von f, ausgewertet an den Nullstellen des Polynoms p_n und $\overrightarrow{\langle f, p\rangle_n} := [\langle f, p_0\rangle_n, \ldots, \langle f, p_{n-1}\rangle_n]^t$ der Vektor der Gauß'schen Skalarprodukte, dann gilt*

$$\overrightarrow{\langle f, p\rangle_n} = U_0\vec{f}, \tag{1.17}$$

mit $U_0 = U diag(\sqrt{w_0}, \ldots, \sqrt{w_{n-1}})$. Dabei bezeichnet U die orthonormale Matrix, deren Spalten die normierten Eigenvektoren von M_n enthalten.

Beweis. In Satz 1.2.4 wurde gezeigt, dass die Eigenwerte der Matrix M_n die Nullstellen des Polynoms p_n sind, und die normierten Eigenvektoren die Form

$$\vec{u}(x_k) = \sqrt{w_k}\vec{p}(x_k) \quad \text{mit } w_k = \left(\sum_{j=0}^{n-1}|p_j(x_k)|^2\right)^{-1},$$

haben. Daraus folgt für die zuvor beschriebenen Gauß Quadratur

$$\begin{aligned}\langle f, p_j\rangle_n &= \sum_{k=0}^{n-1} w_k f(x_k)p_j(x_k) \\ &= \sum_{k=0}^{n-1}\sqrt{w_k}f(x_k)u_j(x_k) = e_j U_0\vec{f}.\end{aligned}$$

mit $e_j = [0, \ldots, 1, \ldots, 0]$. □

Satz 1.3.4. *Für $f \in C^{2n}(supp(\mu))$ gilt für den Fehler der Gauß-Quadratur*

$$|\langle f, p_n\rangle - \langle f, p_n\rangle_n| \leq \frac{1}{(2n)!} \sup_{\zeta \in supp(\mu)} |(fp_n)^{(2n)}(\zeta)|.$$

Beweis. Sei $h \in \pi_{2n-1}$ die Lösung des Hermitischen Interpolationsproblems (siehe Abschnitt 2.1.5 in [25])

$$h(x_i) = (f \cdot p_n)(x_i), \quad h'(x_i) = (f \cdot p_n)'(x_i), \quad i = 0, \ldots, n-1.$$

Da h vom Grad < 2n ist, folgt

$$\int_{\mathbb{R}} h(x)\, d\mu(x) = \sum_{i=0}^{n-1} w_i h(x_i) = \sum_{i=0}^{n-1} w_i f(x_i) p_n(x_i).$$

Somit folgt für den Integrationsfehler

$$\langle f, p_n\rangle - \langle f, p_n\rangle_n = \int_{\mathbb{R}} (f(x) - h(x))\, d\mu(x).$$

Aus dem Satz über Interpolationsfehler der Hermite-Interpolation (siehe Satz 2.1.5.9 in [25]), folgt weiter

$$f(x) - h(x) = \frac{(fp_n)^{(2n)}(\zeta)}{(2n)!}(x - x_0)^2 \ldots (x - x_{n-1})^2 = \frac{(fp_n)^{(2n)}(\zeta)}{(2n)!} p_n^2(x),$$

wobei $\zeta = \zeta(x)$ im kleinsten Intervall $I(x, x_0, \ldots, x_{n-1})$ liegt, dass x und alle Nullstellen enthält. Somit folgt für den Fehler

$$\begin{aligned}
|\langle f, p_n\rangle - \langle f, p_n\rangle_n| &= \left| \int_{\mathbb{R}} (f(x) - h(x))\, d\mu(x) \right| \\
&= \left| \int_{\mathbb{R}} \frac{(fp_n)^{(2n)}(\zeta)}{(2n)!} p_n^2(x)\, d\mu(x) \right| \\
&\leq \frac{1}{(2n)!} \sup_{\zeta \in supp(\mu)} |(fp_n)^{(2n)}(\zeta)| \int_{\mathbb{R}} p_n^2(x)\, d\mu(x) \\
&= \frac{1}{(2n)!} \sup_{\zeta \in supp(\mu)} |(fp_n)^{(2n)}(\zeta)|.
\end{aligned}$$

□

1.4 Fourierentwicklung

Satz 1.4.1. *Der Faktorraum $L^2(\mu)$ ist stets ein Hilbertraum.*

Beweis. siehe z.B. [20] oder [24]. □

Satz 1.4.2. *Sei $\boldsymbol{H}$ ein Hilbertraum und $S \subset \boldsymbol{H}$. Folgende Aussagen sind äquivalent.*

1. *S ist ein vollständiges Orthonormalsystem.*
2. *$x = \sum_{e \in S} \langle x, e \rangle e \quad \forall x \in \boldsymbol{H}$.*
3. *Es gilt $\boldsymbol{H} = \overline{span(S)}$.*
4. *$\langle x, y \rangle = \sum_{e \in S} \langle x, e \rangle \langle e, y \rangle \quad \forall x, y \in \boldsymbol{H}$.*
5. *(Parseval'sche Gleichung)*

$$\|x\|^2 = \sum_{e \in S} |\langle x, e \rangle|^2 \quad \forall x \in \boldsymbol{H}.$$

Beweis. siehe p. 234 in [24] □

Eine Folge $(p_n)_{n \geq 0}$ orthonormaler Polynome mit $deg(p_n) = n$ bildet stets ein vollständiges Orthonormalsystem. Aus Punkt (2) folgt somit, dass sich jede Funktion $f \in L^2(\mu)$ in eine Fourierreihe bezüglich einer Basis aus orthonormalen Polynomen $(p_n)_{n \geq 0}$ entwickeln lässt. Es gilt also für eine Funktion $f \in L^2(\mu)$

$$f = \sum_{n=0}^{\infty} \langle f, p_n \rangle p_n,$$

im Sinne der L^2-Norm. Die Koeffizienten $\langle f, p_n \rangle$ werden Fourierkoeffizienten genannt und haben die Form

$$\langle f, p_n \rangle = \int_{\mathbb{R}} f(t) \overline{p_n(t)} \, d\mu(t).$$

Kapitel 2

Legendre Polynome

Eine wichtige Klasse orthogonaler Polynome bilden die Legendre Polynome. Da sie in der Fouriertransformationstechnik nach Den Iseger Verwendung finden, werden nun ihre Eigenschaften näher beschrieben.

In Kapitel 1 wurden einige allgemein gültige Eigenschaften orthogonaler Polynome besprochen, die nun auf den Spezialfall der Legendre Polynome umgelegt werden. Dazu sei $L^2[0,1]$, der Raum der Lebesgue messbaren und quadratisch integrierbaren Funktionen auf $[0,1]$ und μ das Lebesgue Maß auf $[0,1]$. Das Skalarprodukt, für zwei Funktionen f und g aus $L^2[0,1]$ hat dann die Form

$$\langle f, g\rangle_{L^2[0,1]} = \int_0^1 f(x)\overline{g(x)}\,dx. \tag{2.1}$$

Definition 2.0.1. *Die Legendre Polynome* $\{\phi_n, n \geq 0\}$ *sind definiert durch*

$$\phi_n(t) = \frac{\sqrt{2n+1}}{n!} D^n((t(t-1))^n), \tag{2.2}$$

wobei $D^n = \frac{d^n}{dt^n}$.

Satz 2.0.3. *Die Legendre Polynome* $\{\phi_n, n \geq 0\}$ *bilden ein vollständiges Orthonormalsystem in* $L^2[0,1]$.

Beweis. Ohne Beschränkung der Allgemeinheit wird $m \geq n$ vorausgesetzt.

$$\langle \phi_n, \phi_m\rangle_{L^2[0,1]} = \frac{\sqrt{2n+1}}{n!}\frac{\sqrt{2m+1}}{m!}\int_0^1 \frac{d^n}{dt^n}((t(t-1))^n)\frac{d^m}{dt^m}((t(t-1))^m)\,dt$$

Für das Integral $I_{n,m} := \int_0^1 \frac{d^n}{dt^n}((t(t-1))^n)\frac{d^m}{dt^m}((t(t-1))^m)\,dt$ folgt nach partieller Integration

$$\begin{aligned} I_{n,m} &= \frac{d^n}{dt^n}((t(t-1))^n)\frac{d^{m-1}}{dt^{m-1}}((t(t-1))^m)|_0^1 \\ &\quad - \int_0^1 \frac{d^{n+1}}{dt^{n+1}}((t(t-1))^n)\frac{d^{m-1}}{dt^{m-1}}((t(t-1))^m)\,dt. \end{aligned}$$

Das Polynom $(t(t-1))^m$ hat in 0 und 1, je eine $m-$fache Nullstelle. Demnach gilt

$$\frac{d^{m-k}}{dt^{m-k}}((t(t-1))^m) = 0 \quad \text{für } t = \{0,1\}, \text{ und } k = 1,\ldots,m.$$

Nach $(m-1)$ analogen, partiellen Integrationsschritten erhält man

$$I_{n,m} = (-1)^m \int_0^1 \frac{d^{n+m}}{dt^{n+m}}((t(t-1))^n)((t(t-1))^m)\,dt.$$

Unter Voraussetzung $m > n$ folgt $n+m > 2n$ und daraus $\frac{d^{n+m}}{dt^{n+m}}((t(t-1))^n) = 0$. Somit ist $I_{n,m} = 0$ für $m \neq n$. Für $m = n$ folgt aus $\frac{d^{2n}}{dt^{2n}}((t(t-1))^n) = (2n)!$

$$I_{n,n} = (-1)^n(2n)! \int_0^1 (t(t-1))^n\,dt.$$

Mehrmalige partielle Integration liefert

$$\begin{aligned} I_{n,n} &= (-1)^n(2n)!\left(\frac{t^n(t-1)^{n+1}}{n+1}|_0^1 - \frac{n}{n+1}\int_0^1 t^{n-1}(t-1)^{n+1}\,dt\right) \\ &= \cdots \\ &= (-1)^{2n}(2n)!\frac{n(n-1)\ldots 1}{(n+1)\ldots 2n}\int_0^1 (t-1)^{2n}\,dt \\ &= \frac{(n!)^2}{2n+1}. \end{aligned}$$

Somit folgt für das Skalarprodukt

$$\langle \phi_n, \phi_m \rangle_{L^2[0,1]} = \begin{cases} 1 & n = m \\ 0 & n \neq m \end{cases}.$$

Zur Vollständigkeit des Systems sei auf Abschnitt 1.4 verwiesen. □

Bemerkung 2.0.1. Die klassischen Legendre Polynome sind definiert durch

$$P_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n} \left[\left(x^2 - 1 \right)^n \right]. \tag{2.3}$$

Sie bilden ein vollständiges Orthonormalsystem in $L^2[-1, 1]$. Die in (2.2) definierten Legendre Polynome können als verschobene Version der klassischen Legendre Polynome aufgefasst werden. Es gilt

$$\phi_n(t) = \sqrt{2n+1} P_n(2t-1).$$

Dieser Zusmmenhang kann durch einfaches Nachrechnen gezeigt werden

$$\begin{aligned}
\phi_n(t) &= \frac{\sqrt{2n+1}}{n!} \frac{d^n}{dt^n}((t(t-1))^n) \\
&= \frac{\sqrt{2n+1}}{4^n n!} \frac{d^n}{dt^n}(((2t-1)^2 - 1)^n) \\
&= \frac{\sqrt{2n+1}}{2^n n!} \frac{d^n}{dx^n}((x^2-1)^n)|_{x=2t-1} \\
&= \sqrt{2n+1} \quad P_n(2t-1).
\end{aligned}$$

Wegen

$$\phi_n(t) = \frac{\sqrt{2n+1}}{n!} \frac{d^n}{dt^n} \left(t^n (t-1)^n \right) = \frac{\sqrt{2n+1}}{n!} \sum_{k=0}^{n} (-1)^{n-k} \binom{n}{k} \frac{(n+k)!}{k!} t^k$$

folgt in der Notation von Satz 1.2.2

$$\begin{aligned}
c_n &= \frac{(2n)!}{(n!)^2} \sqrt{2n+1}, \\
c_n' &= -\frac{(2n-1)!}{((n-1)!)^2} \sqrt{2n+1}.
\end{aligned}$$

Mit

$$\theta_n = \frac{n+1}{4n+2} \sqrt{\frac{2n+1}{2n+3}}. \tag{2.4}$$

erhält man

$$A_n = \frac{c_{n+1}}{c_n} = \frac{1}{\theta_n}, \quad B_n = A_n \left(\frac{c_{n+1}'}{c_{n+1}} - \frac{c_n'}{c_n} \right) = -\frac{1}{2\theta_n}, \quad C_n = \frac{c_{n+1} c_{n-1}}{c_n^2} = \frac{\theta_{n-1}}{\theta_n}.$$

Satz 1.2.2 liefert die Rekursion

$$\phi_0(t) = 1, \qquad \phi_1(t) = 2\sqrt{3} \left(t - \frac{1}{2} \right),$$

$$(t - \frac{1}{2})\phi_n(t) \quad = \quad \theta_{n-1}\phi_{n-1}(t) + \theta_n \phi_{n+1}(t), \qquad (n \geq 1) \tag{2.5}$$

In Kapitel 1 wurde eine spezielle Operatormatrix gefunden, mit deren Hilfe es möglich ist, die Nullstellen eines orthogonalen Polynoms zu berechnen. Für die Legendre Polynome soll nun diese Matrix dargestellt werden.

Laut (1.9) und (1.10) gilt für die Einträge jener Matrix

$$\alpha_k = -\frac{B_k}{A_k} = \frac{1}{2}, \qquad \beta_k = \frac{1}{A_k} = \theta_k, \qquad (k \geq 0).$$

Die Matrix N_n, mittels derer die Nullstellen $\{\lambda_0, \ldots, \lambda_{n-1}\}$ des n-ten Legendre Polynoms berechnet werden können, hat daher die Darstellung

$$N_n = \begin{bmatrix} \frac{1}{2} & \theta_0 & 0 & \cdots & 0 \\ \theta_0 & \frac{1}{2} & \theta_1 & \ddots & \vdots \\ 0 & \theta_1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \frac{1}{2} & \theta_{n-2} \\ 0 & \cdots & 0 & \theta_{n-2} & \frac{1}{2} \end{bmatrix}. \tag{2.6}$$

mit θ_k wie in (2.4).

Die Eigenwerte der Matrix bzw. die Nullstellen des Polynoms ϕ_n liegen, laut Satz 1.2.1, alle im Inneren des Intervalls $[0, 1]$. Der zu einer Nullstelle λ_k gehördende Eigenvektor, ist gegeben durch $\vec{\phi}(\lambda_k) = [\phi_0(\lambda_k), \ldots, \phi_{n-1}(\lambda_k)]^t$.

Laut Satz 1.3.1 hat die Gauß Quadratur vom Grad n, für zwei Funktionen $f, g \in L^2[0, 1]$ die Form

$$\langle f, g \rangle_n := \sum_{k=0}^{n-1} w_k f(\lambda_k) \overline{g}(\lambda_k), \tag{2.7}$$

wobei $\{\lambda_k : 0 \leq k < n\} \in [0, 1]$ die Nullstellen des Polynoms ϕ_n bezeichnen und die Gewichte w_k die Christoffel-Zahlen $w_k = \left(\sum_{j=0}^{n-1} |\phi_j(\lambda_k)|^2\right)^{-1}$ sind.

Es gilt $\langle f, g \rangle_n = \langle f, g \rangle_{L^2[0,1]}$ für $f \cdot g \in \pi_{2n-1}$, wiederum aufgrund des Satzes 1.3.1.

Um diese Quadraturformel möglichs effizient zu berechnen, wird eine Darstellung durch Matrizen gewählt. Im Fall der Legendre Polynome hat Satz 1.3.3 folgende Form

Satz 2.0.4. *Sei f eine Funktion in $L^2[0, 1]$. Sei $\vec{f} = [f(\lambda_0), \ldots, f(\lambda_{n-1})]^t$ der Vektor der Funktionswerte ausgewertet an den Nullstellen des n-ten Legendre Polynoms. Der Vektor der Gauß'schen Skalarprodukte $\overrightarrow{\langle f, \phi \rangle_n} = [\langle f, \phi_0 \rangle_n, \ldots, \langle f, \phi_{n-1} \rangle_n]^t$, kann berechnet werden durch*

$$\overrightarrow{\langle f, \phi \rangle_n} = U_0 \vec{f} \qquad \vec{f} = U_0^{-1} \overrightarrow{\langle f, \phi \rangle_n} \tag{2.8}$$

mit $U_0 = Udiag(\sqrt{w_0}, \ldots, \sqrt{w_{n-1}})$ *sowie* $w_k = \left(\sum_{j=0}^{n-1} |\phi_j(\lambda_k)|^2\right)^{-1}$ *für* $0 \leq k \leq n-1$.

Die Ergebnisse aus Satz 2.0.4 können auch auf Funktionen $f \in L^2(\mathbb{R})$ angewendet werden. Faßt man $f_j(x) := f(x+j)$ mit $0 \leq x \leq 1$ als Funktion in $L^2(0,1)$ auf folgt

$$\overrightarrow{\langle f_j, \phi\rangle_n} = \overrightarrow{\langle f(j+\cdot), \phi\rangle_n} = U_0 \vec{f}_j \qquad \vec{f}_j = U_0^{-1} \overrightarrow{\langle f(j+\cdot), \phi\rangle_n} \quad (j \in \mathbb{Z}). \tag{2.9}$$

Kapitel 3

Orthonormale Polynome im Bildbereich der Fouriertransformation

Wir beginnen mit der expliziten Berechnung der Fouriertransformierten des n-ten auf $[0,1]$ eingeschränkten Legendre Polynoms

$$\begin{aligned}\widehat{\phi_n}(t) &= \int_0^1 e^{-itx}\phi_n(x)\,dx \\ &= \frac{\sqrt{2n+1}}{n!}\int_0^1 e^{-itx}D^n(x^n(x-1)^n)\,dx.\end{aligned}$$

Für $t=0$ gilt $\widehat{\phi_n}(0)=\delta_{0,n}$. Für $t\neq 0$ folgt, durch mehrmalige partielle Integration,

$$\begin{aligned}&\int_0^1 e^{-itx}D^n(x^n(x-1)^n)\,dx = \\ =\;& \frac{1}{it}D^n(x^n(x-1)^n)|_{x=0} - \frac{e^{-it}}{it}D^n(x^n(x-1)^n)|_{x=1} + \int_0^1 \frac{e^{-itx}}{it}D^{n+1}(x^n(x-1)^n\,dx \\ =\;& \frac{1}{it}\left(\sum_{k=0}^{n} D^{(n+k)}(x^n(x-1)^n)|_{x=0}\left(\frac{1}{it}\right)^k - e^{-it}\sum_{k=0}^{n} D^{(n+k)}(x^n(x-1)^n)|_{x=1}\left(\frac{1}{it}\right)^k\right)\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{it}\Bigg(\sum_{k=0}^{n}\sum_{j=0}^{n+k}\binom{n+k}{j}D^{(n+k-j)}(x^n)D^{(j)}((x-1)^n)|_{x=0}\left(\frac{1}{it}\right)^k \\
&\quad -e^{-it}\sum_{k=0}^{n}\sum_{j=0}^{n+k}\binom{n+k}{j}D^{(j)}(x^n)D^{(n+k-j)}((x-1)^n)|_{x=1}\left(\frac{1}{it}\right)^k\Bigg) \\
&= \frac{1}{it}\left(n!(-1)^n\sum_{k=0}^{n}\frac{(n+k)!(-1)^{-k}}{(n-k)!k!}\left(\frac{1}{it}\right)^k - n!e^{-it}\sum_{k=0}^{n}\frac{(n+k)!}{(n-k)!k!}\left(\frac{1}{it}\right)^k\right).
\end{aligned}
$$

Somit gilt für die Fouriertransformierte des n-ten Legendre Polynoms

$$\widehat{\phi_n}(t) = \frac{1}{it}\left((-1)^n p_n\left(\frac{1}{it}\right) - e^{-it}p_n\left(-\frac{1}{it}\right)\right) \quad (t \neq 0),$$

wobei $p_n(t)$ wie folgt definiert ist

$$p_n(t) := \sqrt{2n+1}\sum_{k=0}^{n}\frac{(k+n)!}{(n-k)!}\frac{(-t)^k}{k!}. \tag{3.1}$$

Mit Hilfe des Operators

$$\psi f(t) := \frac{1}{t}f(\frac{1}{t}) \tag{3.2}$$

folgt

$$\widehat{\psi\phi_n}(t) = \frac{1}{t}\widehat{\phi_n}\left(\frac{1}{t}\right) = -i((-1)^n p_n(-it) - e^{-i/t}p_n(it)). \tag{3.3}$$

Für Argumente t der Form $t = (2\pi(k+\nu))^{-1}$ mit $k \in \mathbb{Z}$ sowie $\nu \in (0,1)$, ist $e^{-i/t} = e^{-2\pi i(k+\nu)} = e^{-2\pi i\nu}$ unabhängig von k. Für diese t gilt also

$$\widehat{\psi\phi_n}(t) = -i\left((-1)^n p_n\left(-it\right) - e^{-2\pi i\nu}p_n\left(it\right)\right) = e^{-i\pi\nu}i^n q_n^\nu(t). \tag{3.4}$$

wobei

$$q_n^\nu(t) = (-i)^{n+1}\left((-1)^n e^{\pi i\nu}p_n\left(-it\right) - e^{-\pi i\nu}p_n\left(it\right)\right). \tag{3.5}$$

Da das Polynom p_n von der Form $p_n(t) = \sum_{k=0}^{n} a_{n,k}t^k$ mit $a_{n,k} \in \mathbb{R}$ ist, gilt

$$
\begin{aligned}
q_n^\nu(t) &= (-i)^{n+1}\left(\sum_{k=0}^{n}a_{n,k}\left((-1)^n e^{\pi i\nu}(-i)^k - e^{-\pi i\nu}(i)^k\right)t^k\right) \\
&= \sum_{k=0}^{n}a_{n,k}\frac{1}{i}\left(i^{n-k}e^{\pi i\nu} - i^{-n+k}e^{-\pi i\nu}\right)t^k \\
&= 2\sum_{k=0}^{n}a_{n,k}\sin\left(\pi\left(\nu + \frac{n-k}{2}\right)\right)t^k.
\end{aligned}
$$

Damit ist gezeigt, dass q_n^ν ein reelles Polynom vom Grad n ist.

Definition 3.0.2. *Sei $\nu \in (0,1)$. Auf dem Folgenraum*

$$\mathbf{l}_\nu^2 = \left\{ (a_k)_{k\in\mathbb{Z}} | \sum_{k\in\mathbb{Z}} \frac{|a_k|^2}{|2\pi(k+\nu)|^2} < \infty \right\} \tag{3.6}$$

wird durch

$$\langle a, b\rangle_\nu := \sum_{k\in\mathbb{Z}} \frac{1}{|2\pi(k+\nu)|^2} a_k \overline{b_k}, \tag{3.7}$$

ein Skalarprodukt definiert. $\mathbf{l}_\nu^2$ wird dadurch zu einem Hilbertraum.

Für fest vorgegebenes ν aus $(0,1)$ betrachten wir die durch

$$\mathcal{F}^\nu f := \left(2\pi(k+\nu)e^{\pi i\nu}\hat{f}(2\pi(k+\nu))\right)_{k\in\mathbb{Z}},$$

mit $\hat{f}(t) = \int_0^1 f(x)e^{-itx}$ definierte lineare Abbildung von $L^2[0,1]$ auf $\mathbf{l}_\nu^2$.

Satz 3.0.5. *$\mathcal{F}^\nu$ ist eine Isometrie (Hilbertraum - Isomorphismus). Es gilt $\mathcal{F}^\nu\phi_n = \left(i^n q_n^\nu\left(\frac{1}{2\pi(k+\nu)}\right)\right)_{k\in\mathbb{Z}}$, d.h. die Orthonormalbasis $\{\phi_n : n \geq 0\}$ in $L^2[0,1]$ wird durch $\mathcal{F}^\nu$ auf die Orthonormalbasis $\left(i^n q_n^\nu\left(\frac{1}{2\pi(k+\nu)}\right)\right)_{k\in\mathbb{Z}}$ abgebildet.*

Beweis. $\mathcal{F}^\nu$ ist offensichtlich eine lineare Abbildung. Weiters gilt

$$\begin{aligned}
\mathcal{F}^\nu\phi_n &= \left(2\pi(k+\nu)e^{\pi i\nu}\hat{\phi_n}(2\pi(k+\nu))\right)_{k\in\mathbb{Z}} \\
&= e^{\pi i\nu}\left(-i\left((-1)^n p_n\left(\frac{-i}{2\pi(k+\nu)}\right) - e^{-2\pi i\nu} p_n\left(\frac{i}{2\pi(k+\nu)}\right)\right)\right)_{k\in\mathbb{Z}} \\
&= i^n\left(q_n^\nu\left(\frac{1}{2\pi(k+\nu)}\right)\right)_{k\in\mathbb{Z}}.
\end{aligned}$$

Zum Nachweis der Isometrie betrachte man folgende Fourierreihe

$$e^{-2\pi i\nu x} f(x) = \sum_{k\in\mathbb{Z}} \hat{f}(2\pi(k+\nu))e^{2\pi ikx}.$$

Aus der Parseval'schen Gleichung folgt

$$\|f\|_{L^2[0,1]} = \sum_{k\in\mathbb{Z}} |\hat{f}(2\pi(k+\nu))|^2 = \|\mathcal{F}^\nu f\|_\nu.$$

Das zeigt insbesondere, dass $\mathcal{F}^\nu$ in $\mathbf{l}_\nu^2$ liegt. Wegen der für Skalarprodukte stets richtigen Beziehung

$$\langle x, y\rangle = \frac{1}{4}\left(\|x+y\|^2 - \|x-y\|^2\right),$$

gilt weiters

$$\langle f, g\rangle_{L^2[0,1]} = \langle \mathcal{F}^\nu f, \mathcal{F}^\nu g\rangle_\nu.$$

Weiters gilt auch die Injektivität und Surjektivität der Abbildung und somit folgt die Aussage des Satzes. □

Wir betrachten nun in weiterer Folge das Skalarprodukt

$$\langle f, g\rangle_Q := \sum_{k\in\mathbb{Z}} \frac{1}{|2\pi(k+\nu)|^2} f\left(\frac{1}{2\pi(k+\nu)}\right)\overline{g}\left(\frac{1}{2\pi(k+\nu)}\right) = \int_{\mathbb{R}} f(x)\overline{g(x)}\, d\mu(x), \quad (3.8)$$

wobei μ auf die Punkte $(2\pi(k+\nu))^{-1}$ konzentriert ist mit

$$\mu\left((2\pi(k+\nu))^{-1}\right) = |2\pi(k+\nu)|^{-2}.$$

Sei

$$Q := L^2(\mu) = \{f : \mathbb{R} \to \mathbb{C} | \langle f, f\rangle_Q < \infty\}.$$

Für ein fixes $\nu \in (0,1)$, ist $(q_n^\nu)_{n\geq 0}$ eine Folge orthonomaler reeller Polynome in Q mit $deg(q_n^\nu) = n$, denn

$$\begin{aligned}
\langle q_n^\nu, q_m^\nu\rangle_Q &= \sum_{k\in\mathbb{Z}} \frac{1}{|2\pi(k+\nu)|^2} q_n^\nu\left(\frac{1}{2\pi(k+\nu)}\right)\overline{q_m^\nu\left(\frac{1}{2\pi(k+\nu)}\right)} \\
&= \sum_{k\in\mathbb{Z}} \frac{1}{|2\pi(k+\nu)|^2} i^{-n}\mathcal{F}^\nu\phi_n\overline{i^{-m}\mathcal{F}^\nu\phi_m} \\
&= i^{m-n}\langle \mathcal{F}^\nu\phi_n, \mathcal{F}^\nu\phi_m\rangle_\nu \\
&= i^{m-n}\langle \phi_n, \phi_m\rangle_{L^2[0,1]} = \delta_{n,m}.
\end{aligned}$$

Um die Nullstellen des Polynoms q_n^ν zu berechnen wird wieder die in Kapitel 1 eingeführte Operatormatrix verwendet. Laut (1.9) und (1.10) gilt für die Einträge dieser Matrix

$$\alpha_k = -\left(\frac{c'_{k+1}}{c_{k+1}} - \frac{c'_k}{c_k}\right), \qquad \beta_k = \frac{c_k}{c_{k+1}}, \qquad (k \geq 0),$$

Wobei c_k und c'_k die beiden führenden Koeffizienten von q_k^ν bezeichnen, d. h.

$$q_k^\nu(t) = c_k t^k + c'_k t^{k-1} + c''_k t^{k-2} + \cdots.$$

Aus (3.1) und (3.5) folgt

$$\begin{aligned}
c_k &= 2\sqrt{2k+1}\frac{(2k)!}{k!}(-1)^k \sin(\pi\nu) \quad (k \geq 0) \\
c'_k &= 2\sqrt{2k+1}\frac{(2k-1)!}{(k-1)!}(-1)^{k-1}\cos(\pi\nu) \quad (k \geq 1).
\end{aligned}$$

Daraus folgt für die Einträge der Operatormatrix

$$\beta_k = \frac{c_k}{c_{k+1}} = -\frac{1}{2}\frac{1}{\sqrt{2k+3}\sqrt{2k+1}} \quad (k \geq 0).$$

sowie

$$\alpha_0 = \frac{1}{2}\cot(\pi\nu), \qquad \alpha_k = 0 \quad (k \geq 1).$$

Somit hat die reelle symmetrische Matrix $M_n(\nu)$ die Form

$$M_n(\nu) = \begin{bmatrix} \frac{1}{2}\cot(\pi\nu) & \beta_0 & 0 & \cdots & 0 \\ \beta_0 & 0 & \beta_1 & \ddots & \vdots \\ 0 & \beta_1 & \ddots & \ddots & \beta_{n-2} \\ 0 & \cdots & 0 & \beta_{n-2} & 0 \end{bmatrix} \tag{3.9}$$

mit

$$\beta_k = -\frac{1}{2}\frac{1}{\sqrt{2k+3}\sqrt{2k+1}}. \tag{3.10}$$

Die Eigenwerte der Matrix bzw. die Nullstellen $\{\eta_0^\nu, \eta_1^\nu, \ldots, \eta_{n-1}^\nu\}$ des Polynoms q_n^ν können durch diese Matrix $M_n(\nu)$ berechnet werden. Der zur Nullstelle η_k^ν gehörenden Eigenvektor, hat die Form $\vec{q}^{\,\nu}(\eta_k^\nu) = [q_0^\nu(\eta_k), \ldots, q_{n-1}^\nu(\eta_k)]^t$.

Laut Satz 1.3.1 hat die Gauß Quadratur vom Grad n, für zwei Funktionen $\hat{f}, \hat{g} \in Q$ die Form

$$\langle \hat{f}, \hat{g} \rangle_{Q,n} = \sum_{k=0}^{n-1} w_k^\nu \hat{f}(\eta_k^\nu)\overline{\hat{g}}(\eta_k^\nu). \tag{3.11}$$

Dabei sind die $\{\eta_k^\nu : 0 \leq k < n\}$ die Nullstellen des Polynoms q_n^ν und die Gewichte w_k^ν, die zugehörigen Christoffel - Zahlen, mit $w_k^\nu = \left(\sum_{j=0}^{n-1} |q_j^\nu(\eta_k^\nu)|^2\right)^{-1}$.

Die Spezialisierung von Satz 1.3.3 auf die Polynome $(q_n^\nu)_{n\leq 0}$ lautet:

Satz 3.0.6. *Sei $\vec{\hat{f}} = [\hat{f}(\eta_0^\nu), \hat{f}(\eta_1^\nu), \cdots, \hat{f}(\eta_{n-1}^\nu)]^t$ der Vektor der Funktionswerte ausgewertet an den Nullstellen des Polynoms q_n^ν. Der Vektor der Gauß'schen Skalarprodukte $\overrightarrow{\langle \hat{f}, q^\nu \rangle_{Q,n}} = \left[\langle \hat{f}, q_0^\nu \rangle_{Q,n}, \ldots, \langle \hat{f}, q_{n-1}^\nu \rangle_{Q,n}\right]^t$, kann berechnet werden durch*

$$\overrightarrow{\langle \hat{f}, q^\nu \rangle_{Q,n}} = V_0 \vec{\hat{f}}, \quad \vec{\hat{f}} = V_0^{-1} \overrightarrow{\langle \hat{f}, \psi\hat{\phi} \rangle_{Q,n}}$$

mit $V_0 = V diag(\sqrt{w_0^\nu}, \ldots, \sqrt{w_{n-1}^\nu})$ sowie $w_k^\nu = \left(\sum_{j=0}^{n-1} |q_j^\nu(\eta_k^\nu)|^2\right)^{-1}$ für $0 \leq k \leq n-1$. Dabei bezeichnet V die orthonormale Matrix deren Spalten $v(\eta_k^\nu) = \sqrt{w_k^\nu} q^\nu(\eta_k^\nu)$ sind.

Kapitel 4

Poisson'sche Summenformel

In diesem Abschnitt werden einige Eigenschaften der Fouriertransformierte diskutiert. Weiters wird die Poisson'sche Summenformel angegeben. Als Referenz siehe [27], [3] sowie [12].

4.1 Die Fouriertransformierte

Definition 4.1.1. *Für ein $f \in L^1(\mathbb{R})$ mit $t \in \mathbb{R}$, lautet die Fouriertransformierte*

$$\hat{f}(t) := \int_{-\infty}^{\infty} f(u)e^{-itu}\,du. \tag{4.1}$$

Die so definierte Fouriertransformierte $\hat{f}$ ist eine beschränkte, stetige Funktion.

$$|\hat{f}(t)| \leq \int_{-\infty}^{\infty} |f(u)|\,du < \infty.$$

Satz 4.1.1 (vgl. [17], p.23)**.** *Ist für ein $f \in L^1(\mathbb{R})$ auch $\hat{f} \in L^1(\mathbb{R})$ dann gilt*

$$f(u) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \hat{f}(t)e^{itu}\,dt \quad f.s. \tag{4.2}$$

Satz 4.1.2 (Faltungssatz, vgl. [15], p.373)**.** *Seien $f, g \in L^1(\mathbb{R})$. Bezeichne*

$$f * g(x) = \int_{\mathbf{R}} f(x-u)g(u)\,du$$

die Faltung von f mit g. Dann gilt

$$\widehat{f * g}(t) = \hat{f}(t)\hat{g}(t),$$

wobei $\hat{f}$ und $\hat{g}$ die Fouriertransformierten von f und g beschreiben.

4.2 Die Poisson'sche Summenformel

Definition 4.2.1. *Eine Funktion $f : [a, b] \to \mathbb{R}$ heißt von beschränkter Variation, wenn*

$$V(f)_a^b = \sup \sum_{i=1}^{n} |f(t_i) - f(t_{i-1})| < \infty,$$

wobei das Supremum über alle Zerlegungen $a = t_0 < t_1 < \cdots < t_n = b$ von $[a, b]$ zu erstrecken ist.

Satz 4.2.1. *Sei $f : \mathbb{R} \to \mathbb{R}$ eine 1-periodische Funktion von beschränkter Variation. Dann gilt für jeden Punkt $t \in \mathbb{R}$*

$$\frac{f(t^+) + f(t^-)}{2} = \lim_{N\to\infty} \sum_{n=-N}^{N} c_n e^{2\pi i n t}.$$

Mit dem Fourierkoeffizienten c_n

$$c_n = \int_{\mathbb{R}} f(t) e^{-2\pi i n t}\, dt.$$

Beweis. siehe p. 57-58 in [27] □

Lemma 4.2.1. *Sei $f \in L^1(-\infty, \infty)$ und von beschränkter Variation. Dann konvergieren die Partialsummen*

$$F_N(t) = \sum_{k=-N}^{N} f(t + k)$$

absolut und gleichmäßig. Die Grenzfunktion ist 1-periodisch und von beschränkter Variaton.

Beweis. Sei $V(f)_n$ die totale Variaton der Funktion f in $[n, n+1]$, dann gilt $\sum_n V(f)_n < \infty$. Weil $f \in L^1(-\infty, \infty)$ gibt es ein $t_0 \in [0, 1]$ mit $\lim_{N\to\infty} \sum_{k=-N}^{N} f(t_0 + k) < \infty$.

Für beliebige $t \in [0, 1]$ gilt dann

$$\begin{aligned} \sum_{k=-N}^{N} |f(t + k)| &\leq \sum_{k=-N}^{N} (|f(t + k) - f(t_0 + k)| + |f(t_0 + k)|) \\ &\leq \sum_{k=-N}^{N} V(f)_k + \sum_{k=-N}^{N} |f(t_0 + k)| < \infty. \end{aligned}$$

Also konvergiert F_N absolut und gleichmäßig.

Es ist noch zu zeigen, dass auch die Grenzfunktion von beschränkter Variation ist. Sei $0 = t_0 < t_1 < \cdots < t_m = 1$ eine Zerlegung von $[0, 1]$. Dann gilt

$$\begin{aligned} & \sum_{j=0}^{m} |\sum_{k=-\infty}^{\infty} f(t_j + k) - \sum_{k=-\infty}^{\infty} f(t_{j-1} + k)| \\ \leq & \sum_{k=-\infty}^{\infty} \sum_{j=0}^{m} |f(t_j + k) - f(t_{j-1} + k)| \\ \leq & \sum_{k=-\infty}^{\infty} V(f)_k < \infty. \end{aligned}$$

□

Satz 4.2.2 (Poisson'sche Summenformel)**.** *Sei $f \in L^1(-\infty, \infty)$ von beschränkter Variation und $f(t) = \frac{f(t^+)+f(t^-)}{2}$ für alle $t \in \mathbb{R}$. Dann gilt:*

$$\sum_{k=-\infty}^{\infty} \hat{f}(2\pi k) = \sum_{k=-\infty}^{\infty} f(k). \tag{4.3}$$

Beweis. Sei $g(t)$ die 1-periodische Funktion.

$$g(t) := \sum_{k=-\infty}^{\infty} f(t + k).$$

Nach Lemma 4.2.1 ist g von beschränkter Variation. Weiters ist $g(t) = \frac{g(t^+)+g(t^-)}{2}$. Die Fourierreihe der Funktion $g(t)$ lautet

$$g(t) = \sum_{k=-\infty}^{\infty} c_k e^{2\pi ikt}.$$

Die Fourierkoeffizienten lauten

$$\begin{aligned} c_k &= \int_0^1 g(t)e^{-2\pi ikt}\, dt = \int_0^1 \sum_{l=-\infty}^{\infty} f(t + l)e^{-2\pi ik(t+l)}\, dt \\ &= \sum_{l=-\infty}^{\infty} \int_l^{l+1} f(t)e^{-2\pi ik(t)}\, dt = \int_{-\infty}^{\infty} f(s)e^{-2\pi iks}\, ds \\ &= \hat{f}(2\pi k). \end{aligned}$$

Nach Satz 4.2.1 gilt für $t \in \mathbb{R}$

$$g(t) = \sum_{k=-\infty}^{\infty} f(t + k) = \sum_{k=-\infty}^{\infty} \hat{f}(2\pi k)e^{2\pi ikt}$$

und mit $t = 0$ folgt

$$\sum_{k=-\infty}^{\infty} f(k) = \sum_{k=-\infty}^{\infty} \hat{f}(2\pi k).$$

□

Wendet man dies auf $f_1(x) := e^{-(a+2\pi i\nu)x} f(x) \in L^1(-\infty, \infty)$ an, folgt wegen $\hat{f}_1(t) = \hat{f}(-ai + 2\pi\nu + t)$ die Gleichung

$$\sum_{k=-\infty}^{\infty} \hat{f}(-ai + 2\pi(k+\nu)) = \sum_{k=-\infty}^{\infty} e^{-ak} e^{-2\pi ik\nu} f(k). \tag{4.4}$$

Die Poisson'sche Summenformel wird im Verfahren von Den Iseger dazu verwendet die Fourierkoeffizienten der Originalfunktion mit jenen der Fouriertransformierten zu verknüpfen. Diese Koeffizienten werden durch die in (2.1) sowie (3.9) definierten Skalarprodukte gebildet.

Sei $f \in L^1(\mathbb{R})$ und $\hat{f}(t) := \int_{\mathbb{R}} e^{-itx} f(x)\, dx$. Dann gilt unter der Verwendung von

$$\overline{\hat{\phi}_n(t)} = \int_0^1 \phi_n(x) e^{itx}\, dx = \hat{\phi}_n(-t) = \hat{\phi_n^-}(t)$$

$$\begin{aligned}
\langle \psi\hat{f}, q_n^\nu \rangle_Q &= \langle \psi\hat{f}, i^{-n} e^{i\pi\nu} \psi\hat{\phi}_n \rangle_Q \\
&= \sum_{k=-\infty}^{\infty} \frac{1}{|2\pi(k+\nu)|^2} \psi\hat{f}\left(\frac{1}{2\pi(k+\nu)}\right) \overline{\psi\hat{\phi}_n\left(\frac{1}{2\pi(k+\nu)}\right)} i^n e^{-i\pi\nu} \\
&= i^n e^{-i\pi\nu} \sum_{k=-\infty}^{\infty} \hat{f}(2\pi(k+\nu)) \overline{\hat{\phi}_n(2\pi(k+\nu))} \\
&= i^n e^{-i\pi\nu} \sum_{k=-\infty}^{\infty} \widehat{f * \phi_n^-}(2\pi(k+\nu)) \\
&\overset{\text{PSF}}{=} i^n e^{-i\pi\nu} \sum_{k=-\infty}^{\infty} e^{-2\pi ik\nu} (f * \phi_n^-)(k) \\
&= i^n e^{-i\pi\nu} \sum_{k=-\infty}^{\infty} e^{-2\pi ik\nu} \langle f(k+\cdot), \phi_n \rangle_{L^2[0,1]}
\end{aligned}$$

Also gilt

$$\langle \psi\hat{f}, q_n^\nu \rangle_Q = e^{-i\pi\nu} i^n \sum_{k=-\infty}^{\infty} e^{-2\pi ik\nu} \langle f(k+\cdot), \phi_n \rangle_{L^2[0,1]}. \tag{4.5}$$

Kapitel 5

Die numerische Fouriertransformation

5.1 Kurzbeschreibung des Verfahrens

Im folgenden Kapitel wird ein numerisches Verfahren zur Berechnung der Fouriertransformierten bzw. deren Inversen vorgestellt, dass von Peter Den Iseger entwickelt wurde. Aus einer Menge von Funktionswerten der Originalfunktion können durch Verwendung dieses Verfahrens Funktionswerte der Fouriertransformierte berechnet werden und umgekehrt.

Die Basis des Verfahrens von Den Iseger ist die Poisson'schen Summenformel. Die Poisson'sche Summenformel stellt das Bindeglied zwischen Werten der Fouriertransformierten und Werten der Originalfunktion dar. Weitere wichtige Bestandteile des Algorithmus sind die Gauß Quadratur und die Fast Fourier Transform (FFT) sowie die Inverse Fast Fourier Transform (IFFT) nach Cooley und Tukey.

Bei diesem Verfahren wird keine direkte Transformation von Funktionswert zu Funktionswert durchgeführt. Vielmehr wird die Funktion in eine Fourierreihe nach verschobenen Legendre Polynomen entwickelt und analog dazu wird ihre Fouriertransformierte in eine Fourierreihe nach orthonormalen Polynomen, die durch die Fouriertransformation der Legendre Polynome entstehen, entwickelt. Die Entwicklungskoeffizienten können durch die Gauß Quadratur approximativ berechnet werden. Die Poisson'sche Summenformel liefert den Zusammenhang zwischen den Entwicklungskoeffizienten der Originalfunktion und ihrer Fouriertransformierten. Mit Hilfe der FFT bzw. der IFFT können diese Entwicklungskoeffizienten schnell ineinander übergerechnet werden.

Durch die Anwendung der FFT Routinen ergibt sich für die Berechnung von M Werten der Fouriertransformierten (bzw. der Inversen) eine Laufzeit des Algorithmus von $O(M\log(M))$ Schritten.

Das Verfahren ist besonders gut geeignet zur Transformation von stetige Funktionen, liefert aber auch gute Ergebnisse für Funktionen mit Singularitäten oder Unstetigkeitstellen bei steigender Laufzeit.

Zuerst wird die Funktionsweise des Verfahrens und die Beschreibung des dazugehörigen Algorithmus für stetige Funktionen behandelt. In den folgenden Abschnitten werden die nötigen Modifikationen zur Behandlung von Singularitäten und Unstetigkeitsstellen beschrieben. Als Referenz siehe [6],[7] sowie [10].

5.2 Umsetzung der Fourierentwicklung

Die Menge der Legendre Polynome $\{\phi_n, n \geq 0\}$ sowie die Polynome $\{q_n^\nu, n \geq 0\}$, definiert im Bildbereich der Fouriertransformierten, sind vollständige Orthonormalsysteme im $L^2[0,1]$ sowie im Raum $Q = L^2(\mu)$. Es ist also möglich Funktion aus $L^2[0,1]$ und Q in ihre Fourierreihen zu entwickeln.

Funktionen $f \in L^2(\mathbb{R})$ werden in Funktionen aus $L^2[0,1]$ zerlegt, indem wir $\mathbb{R}$ in Teilintervalle der Form $[j, j+1)$ zerlegen. Die Funktion $f_j = f(j+\cdot) \in L^2[0,1]$ mit $j \in \mathbb{Z}$, besitzt die Fourierreihendarstellung

$$f(j+x) = \sum_{k=0}^{\infty} \langle f(j+\cdot), \phi_k \rangle_{L^2[0,1]} \, \phi_k(x), \qquad x \in [0,1). \tag{5.1}$$

Liegt andererseits für $f \in L^2(\mathbb{R})$ die Funktion

$$\psi\hat{f}(t) = \frac{1}{t}\hat{f}\left(\frac{1}{t}\right) \quad \text{mit } \hat{f}(t) = \int_{\mathbb{R}} f(x)e^{-itx}\,dx$$

in Q, d.h.

$$\langle \psi\hat{f}, \psi\hat{f} \rangle_Q = \sum_{k\in\mathbb{Z}} |\hat{f}(2\pi(k+\nu))|^2 < \infty,$$

dann besitzt $\psi\hat{f}$ die Fourierentwicklung

$$\psi\hat{f}(t) = \sum_{k=0}^{\infty} \langle \psi\hat{f}, q_k^\nu \rangle q_k^\nu, \quad (t \in \mathbb{R}). \tag{5.2}$$

Um die Genauigkeit der Berechnungen zu verbessern, betrachtet man an Stelle von $f(x)$ die Funktion $f(\Delta x)$. Dies entspricht einer Zerlegung von $\mathbb{R}$ in Intervalle der Länge Δ. Aus (5.1) folgt

$$f(\Delta(j+x)) = \sum_{k=0}^{\infty} \langle f\left(\Delta(j+\cdot)\right), \phi_k \rangle_{L^2[0,1]} \phi_k(x), \qquad x \in [0,1). \tag{5.3}$$

Eine endliche Versionen der Reihenentwicklung in (5.1) und (5.3) wird durch die Projektion der Fourierreihen auf π_{n-1}, dem Raum der Polynome vom Grad kleiner gleich $n-1$ erzeugt

$$P_n^{\Delta}(f)(\Delta(j+x)) := \sum_{k=0}^{n-1} \langle f(\Delta(j+\cdot)), \phi_k \rangle_{L^2[0,1]} \phi_k(x), \quad x \in [0,1). \tag{5.4}$$

Mit Hilfe dieser Projektion können die Werte der Funktion $f \in L^2(\mathbb{R})$ approximiert werden. Es folgt für $x \in [0,1)$

$$f(\Delta(j+x)) \sim P_n^{\Delta}(f)(\Delta(j+x)) = \sum_{k=0}^{n-1} \langle f\left(\Delta(j+\cdot)\right), \phi_k \rangle_{L^2[0,1]} \phi_k\left(x\right). \tag{5.5}$$

In jedem Fall gilt

$$\| f(\Delta(j+x) - P_n^{\Delta}(f)(\Delta(j+x)) \|_{L^2[0,1]} \to 0 \qquad n \to \infty.$$

5.2.1 Funktionsweise des Algorithmus

Ausgangspunkt der Betrachtungen ist die aus der Poisson'sche Summenformel folgende Beziehung

$$\langle \psi \hat{f}, q_k^{\nu} \rangle_Q = e^{-i\pi\nu} i^k \sum_{j=-\infty}^{\infty} e^{-2\pi i j \nu} \langle f(j+\cdot), \phi_k \rangle_{L^2[0,1]} \quad (k \geq 0).$$

Mit Hilfe dieser Relation kann der Algorithmus zur Berechnung der Inversen wie folgt skizziert werden.

Algorithmus : Inverse der Fouriertransformierten

Sei $M = 2^p$ mit $p \geq 0$ sowie $n \geq 1$. Weiters wird $\nu = \frac{m}{M}$ gewählt, für $0 \leq m < M$. Wir nehmen an, dass $f \in L^2[0,M]$, d.h. der (numerische) Träger der Funktion f ist in $[0,M]$. Dann ist

$$\langle \psi \hat{f}, q_k^{m/M} \rangle_Q = e^{-i\pi m/M} i^k \sum_{j=0}^{M-1} e^{-2\pi i j m/M} \langle f(j+\cdot), \phi_k \rangle_{L^2[0,1]}.$$

1. Aus Funktionswerten der Fouriertransformierten werden durch die Gauß Quadratur die Skalarprodukte $\langle \psi \hat{f}, q_k^\nu \rangle_{Q,n}$, für $0 \leq k < n$, berechnet. Diese approximieren die Skalarprodukte $\langle \psi \hat{f}, q_k^\nu \rangle_Q$, für $0 \leq k < n$.

2. Die rechte Seite der Poisson'schen Summenformel ist für $\nu = m/M$, mit $0 \leq m < M$, gegeben durch

$$e^{-i\pi m/M} i^k \sum_{j=0}^{M-1} e^{-2\pi i j m/M} \langle f(j+\cdot), \phi_k \rangle_{L^2[0,1]}.$$

Für jedes $0 \leq k < n$ und $0 \leq m < M$ gilt die approximative Gleichheit

$$\langle \psi \hat{f}, q_k^{m/M} \rangle_{Q,n} \sim e^{-i\pi m/M} i^k \sum_{j=0}^{M-1} e^{-2\pi i j m/M} \langle f(j+\cdot), \phi_k \rangle_{L^2[0,1]}.$$

Die Koeffizienten $\langle f(j+\cdot), \phi_k \rangle_{L^2[0,1]}$, für $0 \leq j < M$, können durch Anwendung der IFFT berechnet werden.

3. Die abgebrochene Fourierreihe

$$f(j+x) \sim \sum_{k=0}^{n-1} \langle f(j+\cdot), \phi_k \rangle_{L^2[0,1]} \phi_k(x), \quad x \in [0,1), \quad j \in [0,M).$$

erlaubt die approximative Berechnung von $f(j+x)$.

Analog dazu erfolgt die Berechnung der Fouriertransformierten, und der zugehörige Algorithmus lässt sich durch folgendes Schema erklären.

Algorithmus : Fouriertransformierte

Auch hier sei $M = 2^p$ mit $p \geq 0$ sowie $n \geq 1$ und $\nu = \frac{m}{M}$, für $0 \leq m < M$. Die betrachtete Originalfunktion f sei aus $L^2[0,M]$, mit dem numerischen Träger $[0,M]$.

A. Aus Funktionswerten der Originalfunktion werden durch die Gauß Quadratur die Skalarprodukte $\langle f(j+\cdot), \phi_k \rangle_n$, für $0 \leq k < n$ und $0 \leq j < M$ berechnet. Diese sind Approximationen für die Skalarprodukte $\langle f(j+\cdot), \phi_k \rangle_{L^2[0,1]}$, für $0 \leq k < n$ und $0 \leq j < M$.

B. Durch Anwenden der FFT kann die rechte Seite der Poisson'schen Summenformel approximativ berechnet werden. Es gilt

$$e^{-i\pi m/M} i^k \sum_{j=0}^{M-1} e^{-2\pi i j m/M} \langle f(j+\cdot), \phi_k \rangle_n,$$

für $0 \leq k < n$ und $0 \leq m < M$. Durch die Poisson'sche Summenformel gilt für jedes $0 \leq k < n$ und $0 \leq m < M$ die Approximation

$$\langle \psi \hat{f}, q_k^{m/M} \rangle_Q \sim e^{-i\pi m/M} i^k \sum_{j=0}^{M-1} e^{-2\pi i j m/M} \langle f(j+\cdot), \phi_k \rangle_n$$

mit $\nu = m/M$.

C. Aufgrund der Gauß Quadratur gilt $\langle \psi \hat{f}, q_k^{m/M} \rangle_Q \sim \langle \psi \hat{f}, q_k^{m/M} \rangle_{Q,n}$, für $0 \leq k < n$ und $0 \leq m < M$. Die Funktionswerte der Fouriertransformierten können an bestimmten Stützstellen durch die Relationen in Satz 3.0.6 berechnet werden.

In Kapitel 1 und 3 wurden spezielle Methoden zur Berechnung der Gauß Quadratur mittels Matrizenrechnung angeführt. Um nun die Schritte 1 und 2 sowie die Schritte B und C dieser Berechnungsvorschriften effizient zu implementieren werden diese Techniken angewandet und führen auf folgende Relationen.

Sei $\vec{\hat{f}} = [\hat{f}(\eta_0^{m/M}), \hat{f}(\eta_1^{m/M}), \cdots, \hat{f}(\eta_{n-1}^{m/M})]^t$ der Vektor der Funktionswerte der Fouriertransformierten, ausgewertet an den Nullstellen des Polynoms $q_n^{m/M}$, für $0 \leq m < M$ und $\overrightarrow{\langle \hat{f}, q^\nu \rangle_{Q,n}} = \left[\langle \hat{f}, q_0^{m/M} \rangle_{Q,n}, \ldots, \langle \hat{f}, q_{n-1}^{m/M} \rangle_{Q,n} \right]^t$ jener der Gauß'schen Skalarprodukte. Es gilt

$$\begin{aligned} \psi \vec{\hat{f}} &\overset{Satz 3.0.6}{=} V_0^{-1} \overrightarrow{\langle \psi \hat{f}, q^{m/M} \rangle_{Q,n}} \\ &\overset{(4.5)}{\sim} V_0^{-1} \vec{F}\left(\frac{m}{M}\right), \end{aligned} \tag{5.6}$$

mit dem Vektor $\vec{F}\left(\frac{m}{M}\right)$, dessen k-ter Eintrag die Form

$$F_k\left(\frac{m}{M}\right) = e^{-\pi i m/M} i^k \sum_{j=0}^{M-1} e^{-2\pi i j m/M} \langle f(j+\cdot), \phi_k \rangle_{L^2[0,1]} \tag{5.7}$$

hat.

Bemerkung 5.2.1. Aus der in (5.7) beschriebenen Fourierreihe können die Koeffizienten, $\langle f(j+\cdot), \phi_k \rangle_{L^2[0,1]}$, durch Verwendung der IFFT berechnet werden.

Weiters gilt auch die Umkehrung

$$V_0 \psi \vec{\hat{f}} \sim \vec{F}\left(\frac{m}{M}\right), \tag{5.8}$$

wobei der Vektor $\vec{F}\left(\frac{m}{M}\right)$ wie zuvor definiert ist.

Bemerkung 5.2.2. Um die benötigten Koeffizienten $\langle f(j+\cdot), \phi_k \rangle_n$ für Schritt A zu berechnen, werden die in Satz 2.0.4 gefundenen Relationen verwendet. Durch Verwendung dieser Skalarprodukte kann der Vektor $\vec{F}\left(\frac{m}{M}\right)$, für $0 \leq m < M$ approximiert werden.

Bemerkung 5.2.3. Der numerischer Träger der Originalfunktion f liegt in der Beschreibung des Algorithmus im Intervall $[0, M]$. Dieser Träger kann jedoch beliebig gewählt werden, d.h. die linke Grenze muss nicht 0 sein. Der numerische Träger hat die allgemeine Form $[L, U]$, mit $L, U \in \mathbb{R}$. Dieser Träger wird in M Intervalle der Länge Δ geteilt, mit $\Delta = \frac{U-L}{M}$.

5.2.2 Die Fast Fourier Transform

Die diskrete Fouriertransformierte der Ordnung N bzw. deren Inverse sind gegeben durch

$$\begin{aligned} F(k) &= \sum_{n=0}^{N-1} f(n) e^{-2\pi i k n/N}, \qquad (0 \le k < N) \\ f(n) &= \frac{1}{N} \sum_{k=0}^{N-1} F(k) e^{2\pi i k n/N} \qquad (0 \le n < N). \end{aligned} \tag{5.9}$$

Die direkte Berechnung der diskreten Fouriertransformierten für alle $0 \le k < N$ benötigt N^2 komplexe Multiplikationen und Additionen. Mit der FFT ist es möglich die Laufzeit auf $O(N \log_2 N)$ Schritte zu reduzieren, wenn $N = 2^p$ eine Zweierpotenz ist.

Zuerst werden jeweils die geraden Indizes der diskreten Fouriersumme, sowie die ungeraden zusammengefasst. Es gilt

$$\begin{aligned} F(k) &= \sum_{n=0}^{N-1} f(n) e^{-2\pi i k n/N} \\ &= \sum_{n=0}^{N/2-1} f(2n) e^{-2\pi i k 2n/(N/2)} + \sum_{n=0}^{N/2-1} f(2n+1) e^{-2\pi i k (2n+1)/(N/2)} \\ &= \sum_{n=0}^{N/2-1} f(2n) e^{-2\pi i k 2n/(N/2)} + e^{-2\pi i k/(N/2)} \sum_{n=0}^{N/2-1} f(2n+1) e^{-2\pi i k (2n)/(N/2)}. \end{aligned}$$

Die diskrete Fouriertransformierte der Ordnung N, kann somit durch 2 diskrete Fouriertransformationen der Ordnung $N/2$ und $O(N)$ Additionen berechnet werden. Führt man die Berechnungen der neuen Fouriertransformatierten rekursiv aus ($N/2$ ist wieder

eine Zweierpotenz) so folgt für die Laufzeit $T(N)$ dieser Algorithmus

$$\begin{aligned} T(N) &= 2T\left(\frac{N}{2}\right) + O(N), \\ T(1) &= 0. \end{aligned}$$

Durch Auflösen der Rekursion erhält man die Laufzeit $T(N) = O(N \log_2 N)$.

5.3 Verbesserung der Genauigkeit sowie Vereinfachung der Implementierung

Obwohl der im Abschnitt zuvor beschriebene Algorithmus, bereits relativ gute Rechenergebnisse liefert, ist es möglich einige Änderungen vorzunehmen um die Implementierung zu vereinfachen bzw. die numerische Stabilität zu verstärken.

In Schritt 1 des Algorithmus zur Berechnung der Inversen, wird mittels der Gauß Quadratur $\langle \psi \hat{f}, q_k^{m/M} \rangle_{Q,n}$ berechnet, für $0 \leq m < M$. Somit muss für jeden Wert $\nu = m/M$ eine eigene Version der Gauß Quadratur durchgeführt werden.

Da die Gauß Quadratur für Polynome vom Grad $\leq 2n-1$, exakte Ergebnisse liefert, gilt

$$\sum_{k=0}^{n-1} w_k^\nu = \langle 1, 1 \rangle_{Q,n} = \langle 1, 1 \rangle_Q = \frac{\langle q_0^\nu, q_0^\nu \rangle_Q}{2(1-\cos(2\pi\nu))} = \frac{1}{2(1-\cos(2\pi\nu))}.$$

Dieses Ergebnis ist minimal für $\nu = 0.5$. Es ist wünschenswert mit möglichst kleinen Gewichten zu arbeiten, um die Fehlerfortpflanzung zu kontrollieren. Die Verwendung von zu großen Gewichten würde aufgetretene Approximationsfehler noch zusätzlich verstärken.

Für einen numerisch stabilere Version des Algorithmus wird $\nu = 0.5$ fixiert. Somit wird Schritt 1 des Algorithmus vereinfacht, und die Gauß Quadratur wird nur einmal mit diesem festen Wert von ν berechnet.

Durch ein Fixieren von ν wird die Funktionsweise des Algorithmus natürlich sehr stark eingeschränkt. Um die Flexibilität wieder herzustellen, wird die Fouriertransformierte an neuen, verschobenen Stützstellen ausgewertet.

In diesem Fall wird für die Verschiebung, $\alpha = 2\pi(\omega - \nu)$ gewählt, mit $\omega \in (0, 1)$.

Für die in (4.5) beschriebene Poisson'sche Summenformel folgt die Darstellung

$$\langle \psi \hat{f}_\alpha, q_k^\nu \rangle_Q = e^{-\pi i \nu} i^k \sum_{j \in \mathbb{Z}} e^{-2\pi i j \nu} \langle f_\alpha(j + \cdot), \phi_k \rangle_{L^2[0,1]} \quad (k \geq 0). \tag{5.10}$$

Mit der Festsetzung von $\nu = 0.5$ und $\alpha = 2\pi(\omega - 0.5)$ wird (5.10) in folgende Form überführt

$$\begin{aligned}\langle \psi \hat{f}_\alpha, q_k^{0.5} \rangle_Q &= e^{-\pi i 0.5} i^k \sum_{j\in\mathbb{Z}} e^{-\pi i j} \langle f_\alpha(j+\cdot), \phi_k \rangle_{L^2[0,1]} \qquad (5.11)\\ &= e^{-\pi i 0.5} i^k \sum_{j\in\mathbb{Z}} e^{-\pi i j - (2\pi i(\omega-0.5))j} \langle e^{-(2\pi(\omega-0.5))ix} f(j+x), \phi_k(x) \rangle_{L^2[0,1]} \\ &= e^{-\pi i 0.5} i^k \sum_{j\in\mathbb{Z}} e^{-2\pi i \omega j} \langle e^{-(2\pi i(\omega-0.5)x)} f(j+x), \phi_k(x) \rangle_{L^2[0,1]}\end{aligned}$$

mit $x \in [0,1)$.

Mit dieser Wahl von α ist es möglich, trotz der fixen Wahl von ν, eine Fourierreihendarstellung der Skalarprodukte der Funktion $e^{-\alpha i \cdot} f(j+\cdot)$ für beliebige Werte von $\omega \in (0,1)$ zu erhalten.

In einem nächsten Schritt wird die rechte Seite der Gleichung (5.11) näher betrachtet. Hier werden die Skalarprodukte von $\langle e^{-\alpha i \cdot} f(j+\cdot), \phi_k \rangle_{L^2[0,1]}$, für $k \geq 0$, verwendet. Das Ziel ist jedoch die Skalarprodukte der Originalfunktion $\langle f(j+\cdot), \phi_k \rangle_{L^2[0,1]}$, für $k \geq 0$, zu berechnen. Weiters soll auch die Umkehrung möglich sein, d.h. aus den Skalarprodukten der Originalfunktion soll diese Fourierreihe gebildet werden, um die neue Version des Algorithmus anwenden zu können.

Zunächst wird das Skalarprodukt $\langle e^{-\alpha i \cdot} f(j+\cdot), \phi_k \rangle_{L^2[0,1]}$, wie folgt zerlegt

$$\begin{aligned}\langle e^{-\alpha i \cdot} f(j+\cdot), \phi_k \rangle_{L^2[0,1]} &= \sum_{l\in\mathbb{N}_0} \langle f(j+\cdot), \phi_l \rangle_{L^2[0,1]} \langle e^{-\alpha i \cdot} \phi_l, \phi_k \rangle_{L^2[0,1]} \\ &\sim \sum_{l<n} \langle f(j+\cdot), \phi_l \rangle_{L^2[0,1]} \langle e^{-\alpha i \cdot} \phi_l, \phi_k \rangle_{L^2[0,1]}.\end{aligned}$$

Nun wird das zweite Skalarprodukt auf der rechten Seite approximativ berechnet. Dazu wird der in (1.6) definierte Multiplikationsoperator $\mathcal{M}$ verwendet. Für das gesuchte Skalarprodukt gilt

$$\langle e^{-\alpha i \cdot} \phi_l, \phi_k \rangle_{L^2[0,1]} = \langle e^{-\alpha i \mathcal{M}} \phi_l, \phi_k \rangle_{L^2[0,1]} \qquad (5.12)$$

mit

$$e^{-\alpha i \mathcal{M}} \phi_l(x) := e^{-\alpha i x} \phi_l(x), \qquad x \in [0,1).$$

Um (5.12) approximativ zu berechnen wird der lineare Operator $\mathcal{M}_n$ verwendet (siehe Abschnitt 1.2). Zuerst wird die Matrixdarstellung für $\mathcal{M}_n^s$, $s \in \mathbb{N}$, bezüglich der Legendre Polynome gesucht. Es gilt

$$N_n^s = \left(\langle \mathcal{M}_n^s \phi_j, \phi_k \rangle_{L^2[0,1]} \right)_{0 \leq j,k \leq n-1},$$

sowie

$$\langle \mathcal{M}_n^s \phi_j, \phi_k \rangle_{L^2[0,1]} = \langle \mathcal{M}^s \phi_j, \phi_k \rangle_{L^2[0,1]} \qquad \text{für } s+j<n.$$

Nun wird die Exponentialfunktion des Operators betrachtet. Es gilt

$$e^{-\alpha i \mathcal{M}_n} := \sum_{s\geq 0} \frac{(-\alpha i)^s}{s!} \mathcal{M}_n^s.$$

Die zugehörige Matrix ist

$$\begin{aligned} e^{-\alpha i N_n} &= \sum_{s\geq 0} \frac{(-\alpha i)^s}{s!} N_n^s \\ &= \left(\langle e^{-\alpha i \mathcal{M}_n} \phi_j, \phi_k \rangle_{L^2[0,1]} \right)_{0\leq j,k\leq n-1}. \end{aligned}$$

Da die Matrix $e^{-\alpha i N_n}$ eine tridiagonale Matrix ist, lässt sich diese diagonalisieren. Es folgt

$$e^{-\alpha i N_n} = U diag(e^{-\alpha i \lambda_0}, \ldots, e^{-\alpha i \lambda_{n-1}}) U^t. \tag{5.13}$$

Wobei λ_k, für $0 \leq k < n$, sowohl die Eigenwerte der Matrix N_n, als auch die Nullstellen des $n-$ten Legendre Polynoms ϕ_n sind. Die Spalten von U werden durch die zugehörigen normierten Eigenvektoren gebildet (siehe Satz 1.3.3 sowie Satz 2.0.4).

Als nächstes soll gezeigt werden, dass sich das gesuchte Skalarprodukt $\left\langle e^{\alpha i \mathcal{M}} \phi_j, \phi_k \right\rangle$ durch den leicht zu berechnenden Ausdruck $\left\langle e^{\alpha i \mathcal{M}_n} \phi_j, \phi_k \right\rangle$ approximieren lässt. Es gilt zunächst

$$\begin{aligned} \langle e^{-\alpha i \mathcal{M}} \phi_j, \phi_k \rangle_{L^2[0,1]} - \langle e^{-\alpha i \mathcal{M}_n} \phi_j, \phi_k \rangle_{L^2[0,1]} &= \sum_{s\geq 0} \frac{(-\alpha i)^s}{s!} \langle \mathcal{M}^s \phi_j - \mathcal{M}_n^s \phi_j, \phi_k \rangle_{L^2[0,1]} \\ &= \sum_{s\geq n-j} \frac{(-\alpha i)^s}{s!} \langle \mathcal{M}^s \phi_j - \mathcal{M}_n^s \phi_j, \phi_k \rangle_{L^2[0,1]}. \end{aligned}$$

Die letze Gleichung folgt aus $\langle \mathcal{M}^s \phi_j - \mathcal{M}_n^s \phi_j, \phi_k \rangle_{L^2[0,1]} = 0$ für $j+s<n$. Für die weitere Fehlerabschätzung beachte man

$$\begin{aligned} |\langle \mathcal{M}^s \phi_j, \phi_k \rangle_{L^2[0,1]}| &\leq \int_0^1 x^s |\phi_j(x)\phi_k(x)|\, dx \leq \int_0^1 |\phi_j(x)\phi_k(x)|\, dx \leq 1. \\ |\langle \mathcal{M}_n^s \phi_j, \phi_k \rangle_{L^2[0,1]}| &= \left| (N_n^s)_{j,k} \right| = \left| \left(U diag\left(\lambda^s\right) U^t \right)_{j,k} \right| \\ &= \left| \sum_{0\leq m<n} u_{i,m} \lambda_m^s u_{k,m} \right| \leq \sum_{0\leq m<n} |u_{i,m} u_{k,m}| \leq 1. \end{aligned}$$

Die letzen Abschätzungen folgen jeweils aus der Ungleichung von Cauchy - Schwarz.

Zusammen ergibt sich folgende Fehlerschranke

$$\left|\langle e^{-\alpha i\mathcal{M}}\phi_j,\phi_k\rangle_{L^2[0,1]} - \langle e^{-\alpha i\mathcal{M}_n}\phi_j,\phi_k\rangle_{L^2[0,1]}\right| \leq 2\sum_{s\geq n-j}\frac{|\alpha|^s}{s!}.$$

Dieser Fehler ist klein, für ein kleines $|\alpha|$ sowie für einen großen Wert von $n-j$.

Durch die in (5.13) gefundene Matrizendarstellungen ist es möglich aus den Skalarprodukten der Funktion $e^{-\alpha i\cdot}f(j+\cdot)$, jene der Originalfunktion zu berechnen und umgekehrt. Es folgt mit $u_{l,k} = \left(UD\left(e^{-\alpha i\lambda}\right)U^t\right)_{l,k}$

$$\langle e^{-\alpha i\cdot}f(j+\cdot),\phi_k\rangle_{L^2[0,1]} \sim \sum_{l<n}\langle f(j+\cdot),\phi_l\rangle_{L^2[0,1]}u_{l,k} \tag{5.14}$$

für $0 \leq k \leq n-1$.

5.3.1 Dämpfungsfaktoren

Die Hauptaufgabe der Dämpfungsfaktoren ist es die Berechnung der Fouriertransformation einer nur schwach fallenden Funktion f auf die numerische einfachere Berechnung der Fouriertransformation von $f_a(x) = e^{-ax}f(x)$ zurückzuführen.

Im folgende Satz wird die Berechnung der Inversen der diskreten Fouriertransformierten näher betrachtet, sowie den dabei enstehenden Diskretisierungsfehlers.

Satz 5.3.1. *Sei*

$$F(\nu) = \sum_{k=-\infty}^{\infty} f(k)e^{-ak-2\pi ik\nu}, \qquad a\in\mathbb{R}. \tag{5.15}$$

Dann gilt

$$\frac{1}{M_2}\sum_{j=0}^{M_2-1} F\left(\frac{j}{M_2}\right)e^{\frac{2\pi imj}{M_2}} = e^{-am}(f(m)+e(m)), \tag{5.16}$$

mit

$$M = 2^p, \qquad M_2 = 2^qM, \quad \textit{für p,q beliebig,} \quad 0<m<M,$$

und dem Fehler

$$|e(m)| \leq \sum_{k\geq M_2} e^{a(m-k)}|f(k)|.$$

Beweis.

$$\begin{aligned}\frac{1}{M_2}\sum_{j=0}^{M_2-1} F\left(\frac{j}{M_2}\right) e^{\frac{2\pi i m j}{M_2}} &= \sum_{k=-\infty}^{\infty} e^{-ak} f(k) \frac{1}{M_2}\sum_{j=0}^{M_2-1} e^{\frac{2\pi i (m-k) j}{M_2}} \\ &= \sum_{\substack{k=0 \\ k\equiv m \bmod M_2}}^{\infty} e^{-ak} f(k) \\ &= e^{-am}(f(m)+e(m))\end{aligned}$$

Wobei der Fehler durch die Ungleichung erfüllt ist

$$|e(m)| \leq \sum_{k\geq M_2} e^{a(m-k)}|f(k)| \leq e^{-a(M_2-M)} \sum_{k\geq M_2} |f(k)|.$$

Somit ist der Fehler nur klein, wenn $M_2 - M$ groß ist. □

Die Rechengenauigkeit der IFFT steigt mit der Größe des Parameters M_2. Da dies jedoch die Laufzeit beeinflusst ist es ein Ziel, eine gute Balance zwischen dem Wert von M_2 und dem Dämpfungsfaktor a sowie M zu finden.

5.3.2 Verbesserte Version des Algorithmus

Mit den zu Beginn dieses Abschnitts besprochenen Änderungen und der Verwendung von reellen Dämpfungsfaktoren kann die verbesserten Version des Algorithmus beschrieben werden.

Sei $\nu = 0.5$. Für ein $a \in \mathbb{R}$ sowie $\omega \in (0,1)$ sei $\alpha = -ai + 2\pi(\omega - 0.5)$.

Algorithmus : Inverse der Fouriertransformierten

Sei $M = 2^p$ mit $p \geq 0$ sowie $n \geq 1$ und gerade. In Abhängigkeit des Parameters M wird $M_2 = 2^q M$ für $q \geq 0$ bestimmt. Wir nehmen an, dass $f \in L^2[0,M]$, d.h. der (numerische) Träger von f ist in $[0,M]$ enthalten.

1. Aus Funktionswerten der Fouriertransformierten werden durch die Gauß Quadratur die Skalarprodukte $\langle \psi \hat{f}_\alpha, q_k^{0.5}\rangle_{Q,n}$, für $0 \leq k < n$, berechnet. Diese approximieren die Skalarprodukte $\langle \psi \hat{f}_\alpha, q_k^{0.5}\rangle_Q$, für $0 \leq k < n$.

2. Sei $\omega = l/M_2$ für $0 \leq l < M_2$. Die rechte Seite der Poisson'schen Summenformel hat damit die Form

$$e^{-i\pi 0.5} i^k \sum_{j=0}^{M-1} e^{-aj} e^{-2\pi i l j/M_2} \langle e^{-\alpha i \cdot} f(j+\cdot), \phi_k\rangle_{L^2[0,1]}.$$

Man setze $\langle e^{-\alpha i \cdot} f(j+\cdot), \phi_k \rangle_{L^2[0,1]} = 0$ für $j = M, \ldots, M_2 - 1$.
Für jedes $0 \leq k < n$ und $0 \leq r < n$ gilt die approximative Gleichheit

$$\langle \psi \hat{f}_\alpha, q_k^{0.5} \rangle_{Q,n} \sim \left(UD\left(e^{-\alpha i \lambda}\right) U^t \right)_{r,k} e^{-i\pi 0.5} i^k \sum_{j=0}^{M_2-1} e^{-aj} e^{-2\pi i l j / M_2} \langle f(j+\cdot), \phi_r \rangle_{L^2[0,1]}.$$

Die Koeffizienten $e^{-aj}\langle f(j+\cdot), \phi_k \rangle_{L^2[0,1]}$, $0 \leq j < M$, können durch Anwendung der IFFT berechnet werden. Durch multiplizieren des Ergebnisses mit e^{aj} für $0 \leq j < M$, erhält man die Koeffizienten $\langle f(j+\cdot), \phi_k \rangle_{L^2[0,1]}$.

3. Die abgebrochene Fourierreihe

$$f(j+x) \sim \sum_{k=0}^{n-1} \langle f(j+\cdot), \phi_k \rangle_{L^2[0,1]} \phi_k(x), \quad x \in [0,1), \quad j \in [0, M).$$

erlaubt die approximative Berechnung von $f(j+x)$.

Analog dazu wird der Algorithmus zur Bestimmung der Fouriertransformierten beschrieben.

Algorithmus : Fouriertransformierte

Sei $M = 2^p$ mit $p \geq 0$ sowie $n \geq 1$ und gerade. In Abhängigkeit des Parameters M wird $M_2 = 2^q M$ für $q \geq 0$ bestimmt. Die Funktion f sei aus $L^2[0, M]$,d.h. der (numerische) Träger ist in $[0, M]$ enthalten.

A. Aus Funktionswerten der Originalfunktion werden durch die Gauß Quadratur die Skalarprodukte $\langle f(j+\cdot), \phi_k \rangle_n$, für $0 \leq k < n$ und $0 \leq j < M$ berechnet. Diese sind Approximationen für die Skalarprodukte $\langle f(j+\cdot), \phi_k \rangle_{L^2[0,1]}$, für $0 \leq k < n$ und $0 \leq j < M$.

B. Berechnung von $e^{-aj}\langle f(j+\cdot), \phi_k \rangle_n$ für $0 \leq k < n$ und $0 \leq j < M$.

C. Durch Anwenden der FFT mit der Ordnung M_2, kann die rechte Seite der Poisson'schen Summenformel approximativ berechnet werden. Dazu wurden weiters $e^{-aj}\langle f(j+\cdot), \phi_k \rangle_n = 0$ gesetzt für $j = M, \ldots, M_2 - 1$.
Somit gilt für $0 \leq k < n$ und $0 \leq j < M_2$

$$e^{-i\pi 0.5} i^k \sum_{l=0}^{M_2-1} e^{-aj} e^{-2\pi i l j / M_2} \langle f(j+\cdot), \phi_k \rangle_n.$$

Durch die Poisson'sche Summenformel gilt für jedes $0 \leq k < n$ und $0 \leq m < M$ die Approximation

$$\langle \psi \hat{f}_\alpha, q_k^{0.5} \rangle_Q \sim \left(UD\left(e^{-\alpha i \lambda}\right) U^t \right)_{r,k} e^{-i\pi 0.5} i^k \sum_{j=0}^{M_2-1} e^{-aj} e^{-2\pi i l j / M_2} \langle f(j+\cdot), \phi_r \rangle_n$$

mit $\omega = l/M_2$ für $0 \leq l < M_2$.

C. Aufgrund der Gauß Quadratur gilt $\langle \psi \hat{f}, q_k^{0.5} \rangle_Q \sim \langle \psi \hat{f}, q_k^{0.5} \rangle_{Q,n}$, für $0 \leq k < n$. Die Funktionswerte der Fouriertransformierten können an bestimmten Stützstellen durch die Relationen in Satz 3.0.6 berechnet werden.

Um die einzelnen Schritte effizient zu implementieren werden die nachfolgenden Relationen verwendet. $\vec{\hat{f}_\alpha} = [\hat{f}(\eta_0^{0.5} + \alpha), \hat{f}(\eta_1^{0.5} + \alpha), \cdots, \hat{f}(\eta_{n-1}^{0.5} + \alpha)]^t$ ist der Vektor der Fouriertransformierten, ausgewertet an den verschobenen Nullstellen des orthonormalen Polynoms $q_n^{0.5}$.

Es gilt

$$\begin{aligned} \psi \vec{\hat{f}_\alpha} &\overset{Satz 3.0.6}{=} V_0^{-1} \overrightarrow{\langle \psi \hat{f}_\alpha, q^\nu \rangle_{Q,n}} \\ &\overset{(5.10)+(5.13)}{\sim} V_0^{-1} U D\left(e^{-\alpha i \lambda}\right) U^t \vec{L}(0.5), \end{aligned} \tag{5.17}$$

mit dem Vektor $\vec{L}(0.5)$, dessen $k-ter$ Eintrag folgende Form hat

$$L_k(0.5) := e^{-\pi i \nu} i^k \sum_{j=-\infty}^{\infty} e^{-2\pi i j \nu} e^{-\alpha i j} \langle f(j+\cdot), \phi_k \rangle_{L^2[0,1]}. \tag{5.18}$$

Bemerkung 5.3.1. Die Koeffizienten $\langle f(j+\cdot), \phi_k \rangle_{L^2[0,1]}$ können durch $\langle f(j+\cdot), \phi_k \rangle_n$ approximiert werden. Diese approximative Berechnung kann effizient mit den Relationen aus Satz 2.0.4 durchgeführt werden.

Auch die Umkehrung gilt

$$U D\left(e^{\alpha i \lambda}\right) U^t V_0 \psi \vec{\hat{f}_\alpha} \sim \vec{L}(0.5), \tag{5.19}$$

mit $\vec{L}(0.5)$ wie zuvor beschrieben.

Bemerkung 5.3.2. Die Entwicklungskoeffizienten $\langle f(j+\cdot), \phi_k \rangle_{L^2[0,1]}$ können mit Hilfe der IFFT aus $L_k(0.5)$ berechnet werden.

Bemerkung 5.3.3. Der numerischer Träger der Originalfunktion f liegt in der Beschreibung des Algorithmus im Intervall $[0, M]$. Dieser Träger kann jedoch beliebig gewählt werden, d.h. die linke Grenze muss nicht 0 sein. Der numerische Träger hat die allgemeine Form $[L, U]$, mit $L, U \in \mathbb{R}$. Dieser Träger wird in M Intervalle der Länge Δ geteilt, mit $\Delta = \frac{U-L}{M}$.

Bemerkung 5.3.4. Der Wert n sollte durch eine gerade Zahl gegeben sein. Denn für ungerade n ist eine der Nullstellen $\{\eta_k^{0.5}, k = 0, 1, \ldots, n-1\}$ des Polynoms $q_n^{0,5}$ gleich 0 und das würde bei der Auswertung von $\psi \vec{\hat{f}}$ zu Problemen führen.

5.4 Detaillierte Beschreibung der Algorithmen

In diesem Abschnitt wird eine detaillierte Beschreibung der zuvor skizzierten Algortihmen gegeben. Zuerst wird auf die Berechnung der Fouriertransformierten eingegangen, gefolgt von der Bestimmung der Inversen. Es wird immer zuerst der Pseudocode des Originalalgorithmus aus Abschnitt 5.2 dargestellt und im Anschluss jener des verbesserten Algorithmus mit den in Abschnitt 5.3 vorgestellten Änderungen.

5.4.1 Algorithmus: Fouriertransformation

Zu Beginn der Berechnungen wird der numerische Träger der Funktion festgelegt. Die Originalfunktion wird auf dem Träger $[L, U]$, mit $L, U \in \mathbb{R}$, betrachtet. Dieser Träger wird in M gleichgroßen Intervallen der Länge Δ geteilt, mit $\Delta = \frac{U-L}{M}$. Die weiteren Input Parameter des Algorithmus werden alle in Abhängigkeit des Parameters M berechnet.

Als Input werden Werte der Originalfunktion verwendet, ausgewertet an den Nullstellen $\lambda = [\lambda_0, \lambda_1, \ldots, \lambda_{n-1}]$ der in Kapitel 2 definierten Legendre Polynome. Somit wird der Vektor der Funktionswerte

$$\vec{f_j} = \vec{f}(L + \Delta(\lambda + j)), \qquad 0 \leq j < M, \quad L \in \mathbb{R} \quad \Delta \in \mathbb{R}^+.$$

verwendet. Der Algorthmus liefert die Funktionswerte der Fouriertransformierten, ausgewertet an den Nullstellen $\eta^\nu = [\eta_0^\nu, \eta_1^\nu, \ldots, \eta_{n-1}^\nu]$ des orthonormalen Polynoms q_n^ν.

Pseudocode des Originalalgorithmus

Input Parameter: n, den Träger der Funktion $[U, L]$, die Anzahl der Intervalle M und somit $\Delta = \frac{U-L}{M}$.

Input: Vektor der Funktionswerte $\vec{f_j}$ für $0 \leq j < M$.

Output: Vektor der Funktionswerte der Fouriertransformierten für $0 \leq m < M$ $\psi\vec{\hat{f}}_\Delta = \left[\frac{1}{\Delta\eta_0^{m/M}}\hat{f}\left(\frac{1}{\Delta\eta_0^{m/M}}\right), \ldots, \frac{1}{\Delta\eta_{n-1}^{m/M}}\hat{f}\left(\frac{1}{\Delta\eta_{n-1}^{m/M}}\right)\right]^t \in \mathbb{C}$.

Schritt 1: Berechne $\overrightarrow{\langle f(L + \Delta(j + \cdot)), \phi\rangle}_n = U_0\vec{f_j}$ mit Hilfe der in Satz 2.0.4 gefundenen Relationen.

Schritt 2: Berechne $F_k^{\Delta}\left(\frac{m}{M}\right)$ mit Hilfe der FFT für $0 \leq m < M$ und $0 \leq k < n$

$$F_k^{\Delta}\left(\frac{m}{M}\right) = e^{-\pi i \frac{m}{M}} i^k \sum_{j=0}^{M-1} e^{\frac{-2\pi i m(j+L/\Delta)}{M}} \langle f(L+\Delta(j+\cdot)), \phi_k \rangle_n.$$

Schritt 3: Nun gilt aufgrund der Poisson'schen Summenformel
$\langle \psi \hat{f}_{\Delta}, q_k^{m/M} \rangle_Q \sim e^{\frac{2\pi i m L}{\Delta M}} F_k^{\Delta}\left(\frac{m}{M}\right)$.

Schritt 4: Berechnung der Funkionswerte durch die (5.6) gefundene Relation
$\psi \vec{\hat{f}}_{\Delta} \sim V_0^{-1} \vec{F}_{\Delta}$, mit $\vec{F}_{\Delta} = e^{\frac{2\pi i m L}{\Delta M}} [F_0^{\Delta}(m/M), \ldots, F_{n-1}^{\Delta}(m/M)]^t$, für $0 \leq m < M$.

Pseudocode des verbesserten Algorithmus

Input Parameter: n, den Träger der Funktion $[L, U]$, die Anzahl der Intervalle M, $\Delta = \frac{U-L}{M}$, $M_2 = 8M, a = \frac{44}{M_2}$
(ohne reelle Dämpfung gilt $a = 0$ und $M_2 = M$)

Input: Vektor der Funktionswerte $\vec{f_j}$ für $0 \leq j < M$.

Output: Vektor der Funktionswerte der Fouriertransformierten
$\psi \vec{\hat{f}}_{\alpha}^{\Delta} = \left[\frac{\exp(Li(\alpha+\pi)/\Delta)}{\Delta \eta_0^{0.5}} \hat{f}\left(\frac{1}{\Delta \eta_0^{0.5}} + \frac{\alpha}{\Delta}\right), \ldots, \frac{\exp(Li(\alpha+\pi)/\Delta)}{\Delta \eta_{n-1}^{0.5}} \hat{f}\left(\frac{1}{\Delta \eta_{n-1}^{0.5}} + \frac{\alpha}{\Delta}\right)\right]^t \in \mathbb{C}$,
für $\nu = 0.5, \quad \alpha = -ai - \pi + \frac{2\pi j}{M_2}$ und $0 \leq j < M_2$.

Schritt 1: Berechne $\overrightarrow{\langle f(L+\Delta(j+\cdot)), \phi \rangle_n} = U_0 \vec{f_j}$ mit Hilfe der in Satz 2.0.4 gefundenen Relationen.

Schritt 2: Setze $f_{kj} = e^{-aj} \langle f(L+\Delta(j+\cdot)), \phi_k \rangle_n$ für $0 \leq j < M_2$ und $0 \leq k < n$, um in der Folge (5.18) berechnen zu können.

Schritt 3: Setze $f_{kj} = 0$ für $M \leq j < M_2$ und $0 \leq k < n$.

Schritt 4: Berechne $L_k(0.5)$ wie in (5.18) mit Hilfe der FFT für $0 \leq j < M_2 - 1$.

$$L_k(0.5) = e^{-\pi i 0.5} i^k \sum_{l=0}^{M_2-1} f_{kl} e^{-\frac{2\pi i l j}{M_2}}.$$

Schritt 5: Berechnung der Funkionswerte durch die in (5.17) gefundenen Methoden
$\psi \vec{\hat{f}}_{\alpha}^{\Delta} \sim V_0^{-1} U D\left(e^{-\alpha i \lambda}\right) U^t \vec{L}(0.5)$.

5.4.2 Algorithmus: Inverse der Fouriertransformierten

Um Funktionswerte der Originalfunktion zu berechnen werden wieder die selben Input Parameter wie in den Algorithmen zuvor verwendet. Als Input werden Funktionswerte

der Fouriertransformierten verwendet, ausgwertet an den Nullstellen des Polynoms q_n^ν.

Daraus können die Entwicklungskoeffizienten der Originalfunktion berechnet werden, und die Funktionswerte können durch Bildung der Fourierreihe berechnet werden.

Pseudocode des Originalalgorithmus

Input Parameter: n, den Träger der Funktion $[L, U]$, die Anzahl der Intervalle M und somit $\Delta = \frac{U-L}{M}$.

Input: Vektor der Funktionswerte der Fouriertransformierten für $0 \leq m < M$
$\psi\vec{\hat{f}}_\Delta = \left[\frac{1}{\Delta\eta_0^{m/M}}\hat{f}\left(\frac{1}{\Delta\eta_0^{m/M}}\right), \ldots, \frac{1}{\Delta\eta_{n-1}^{m/M}}\hat{f}\left(\frac{1}{\Delta\eta_{n-1}^{m/M}}\right)\right]^t \in \mathbb{C}$.

Output: $f(L + \Delta(j + x))$ für $x \in [0, 1)$, $\Delta \in \mathbb{R}^+$ und $0 \leq j < M$.

Schritt 1: Berechnung von $\vec{F}_\Delta\left(\frac{m}{M}\right) \sim V_0\psi\vec{\hat{f}}_\Delta$ mit Hilfe von (5.8).

Schritt 2: Berechne die Werte von f_{kj} mit Hilfe der IFFT für $0 \leq k < n$ und $0 \leq j < M$

$$f_{kj} = \langle f(L + \Delta(j + \cdot)), \phi_k\rangle_{L^2[0,1]} = \frac{1}{M}\sum_{m=0}^{M-1} F_k^\Delta\left(\frac{m}{M}\right) e^{\frac{2\pi i m(j+L/\Delta)}{M}}.$$

Schritt 3: Berechnung der Funktionswerte mittels der Fourierreihe für $0 \leq j < M$

$$f(L + \Delta(j + x)) \sim \sum_{k=0}^{n-1}\langle f(L + \Delta(j + \cdot)), \phi_k\rangle_{L^2[0,1]}\phi_k(x), \qquad x \in [0, 1).$$

Pseudocode des verbesserten Algorithmus

Input Parameter: n, den Träger der Funktion $[L, U]$, die Anzahl der Intervalle M, $\Delta = \frac{U-L}{M}$, $M_2 = 8M, a = \frac{44}{M_2}$
(ohne reelle Dämpfung gilt $a = 0$ und $M_2 = M$)

Input: Vektor der Funktionswerte der Fouriertransformierten
$\psi\vec{\hat{f}}_\alpha^\Delta = \left[\frac{\exp(Li(\alpha+\pi)/\Delta)}{\Delta\eta_0^{0.5}}\hat{f}\left(\frac{1}{\Delta\eta_0^{0.5}} + \frac{\alpha}{\Delta}\right), \ldots, \frac{\exp(Li(\alpha+\pi)/\Delta)}{\Delta\eta_{n-1}^{0.5}}\hat{f}\left(\frac{1}{\Delta\eta_{n-1}^{0.5}} + \frac{\alpha}{\Delta}\right)\right]^t \in \mathbb{C}$,
für $\nu = 0.5, \quad \alpha = -ai - \pi + \frac{2\pi j}{M_2}$ und $0 \leq j < M_2$.

Output: $f(L + \Delta(j + x))$ für $x \in [0, 1)$, $\Delta \in \mathbb{R}^+$ und $0 \leq j < M$.

Schritt 1: Berechnung von $\vec{L}(0.5) = UD\left(e^{\alpha i\lambda}\right)U^tV_0\psi\vec{\hat{f}}_\alpha^\Delta$ mit Hilfe der in (5.19) gefundenen Darstellung. Danach wird weiters $L_k(0.5) = i^{-k}e^{\pi i 0.5}(\vec{L}(0.5))_k$ berechnet, für $0 \leq k < n$.

Schritt 2: Berechne die Werte von f_{kj} mit Hilfe der IFFT für $0 \leq k < n$ und $0 \leq j < M_2$

$$f_{kj} = \frac{1}{M_2} \sum_{l=0}^{M_2-1} L_k(0.5) e^{\frac{2\pi i l j}{M_2}}.$$

Schritt 3: $\langle f(L + \Delta(j + \cdot)), \phi_k \rangle_n = f_{kj} e^{aj}$ für $0 \leq k < n$ und $0 \leq j < M$.

Schritt 4: Berechnung der Funktionswerte mittels der abgebrochenen Fourierreihe $0 \leq j < M$

$$f(L + \Delta(j + x)) \sim \sum_{k=0}^{n-1} \langle f(L + \Delta(j + \cdot)), \phi_k \rangle_n \phi_k(x), \qquad x \in [0, 1).$$

5.5 Testresultate für stetige Funktionen

Die beschriebenen Algorithmen wurden für folgende stetige Funktionen getestet.

Tabelle 1. stetige Testfunktionen

Nr.	Funktion	Fouriertransformierte
1.	$e^{-x(3/2)} I(0, \infty)$	$(it + 1.5)^{-1}$
2.	$e^{-1.2x} \sin(x) I(0, \infty)$	$((it + 1.2)^2 + 1)^{-1}$
3.	$\frac{1}{\sqrt{2\pi}\sigma} e^{-1/2((x-\mu)/\sigma)^2}$	$e^{-it\mu - \sigma^2 t^2/2}$
4.	$\frac{b^p}{\Gamma(p)} x^{p-1} e^{-bx} I(0, \infty)$	$\frac{b^p}{(b+it)^p}$

Bei Nummer 3 wurde für $\mu = 0.1$ sowie $\sigma = 0.2$ gewählt und Nummer 4 wurde für $b = 2$ und $p = 2$ ausgewertet.

In den nachfolgenden Tabellen werden jeweils die mittleren absoluten Fehler der angewandten Algorithmen angeführt. Dieser Fehler wird berechnet durch

$$MAE := \frac{1}{M} \sum_{k=0}^{M-1} |f(k\Delta) - f^*(k\Delta)|,$$

wobei f^* der approximierten Funktion entspricht.

Zusätzlich wurde auch der maximale Fehler betrachtet. Dieser ist definiert durch

$$E_{\max} := \max_{0 \leq k < M} |f(k\Delta) - f^*(k\Delta)|.$$

Zunächst wurden die Algorithmen zur Berechnung der Inversen getestet. In den Tabellen 2-5 werden die Ergebnisse für den Originalalgorithmus aufgelistet sowie die Laufzeiten der Algorithmen.

Tabelle 2. Resultate für Funktion Nr.1
(Originalalgorithmus)

$[L,U]$	M	Δ	$E_{\max}$	MAE	$CPU(s)$
$[0,6]$	8	$\frac{3}{4}$	$4.00e^{-04}$	$8.47e^{-06}$	0.1716
$[0,6]$	16	$\frac{3}{8}$	$7.03e^{-05}$	$1.08e^{-06}$	0.2964
$[0,6]$	32	$\frac{3}{16}$	$9.31e^{-06}$	$1.22e^{-06}$	0.4524
$[0,6]$	64	$\frac{3}{32}$	$1.07e^{-07}$	$1.29e^{-07}$	0.4056

Tabelle 3. Resultate für Funktion Nr.2
(Originalalgorithmus)

$[L,U]$	M	Δ	$E_{\max}$	MAE	$CPU(s)$
$[0,6]$	8	$\frac{3}{4}$	$5.38e^{-05}$	$2.26e^{-05}$	0.1235
$[0,6]$	16	$\frac{3}{8}$	$1.47e^{-04}$	$4.12e^{-05}$	0.1872
$[0,6]$	32	$\frac{3}{16}$	$1.44e^{-05}$	$4.07e^{-05}$	0.3900
$[0,6]$	64	$\frac{3}{32}$	$1.47e^{-07}$	$4.14e^{-07}$	0.4056

Tabelle 4. Resultate für Funktion Nr.3
(Originalalgorithmus)

$[L,U]$	M	Δ	$E_{\max}$	MAE	$CPU(s)$
$[-3,3]$	8	$\frac{3}{4}$	$6.51e^{-05}$	$1.66e^{-05}$	0.0936
$[-3,3]$	16	$\frac{3}{8}$	$3.06e^{-06}$	$4.10e^{-06}$	0.3588
$[-3,3]$	32	$\frac{3}{16}$	$1.06e^{-08}$	$1.02e^{-08}$	0.4524
$[-3,3]$	64	$\frac{3}{32}$	$9.61e^{-08}$	$1.74e^{-08}$	0.8268

Tabelle 5. Resultate für Funktion Nr.4
(Originalalgorithmus)

$[L,U]$	M	Δ	$E_{\max}$	MAE	$CPU(s)$
$[0,6]$	8	$\frac{3}{4}$	$1.36e^{-05}$	$4.48e^{-05}$	0.2808
$[0,6]$	16	$\frac{3}{8}$	$1.44e^{-05}$	$4.07e^{-05}$	0.2496
$[0,6]$	32	$\frac{3}{16}$	$1.47e^{-05}$	$4.12e^{-06}$	0.3900
$[0,6]$	64	$\frac{3}{32}$	$1.47e^{-05}$	$4.14e^{-06}$	0.4056

Für den verbesserten Algorithmus wurden, unter Verwendung der reellen Dämpfung, folgende Input Parameter verwendet $M_2 = 8M$, $n = 16$ sowie $a = 44/M_2$. (Für die Version ohne reelle Dämpfung gilt $M = M_2$ sowie $a = 0$).

Tabelle 6. Resultate für Funktion Nr.1
(Algorithmus mit reeller Dämpfung)

$[L,U]$	M	Δ	$E_{\max}$	MAE	$CPU(s)$
$[0,6]$	8	$\frac{3}{4}$	$5.38e^{-14}$	$2.26e^{-15}$	0.0936
$[0,6]$	16	$\frac{3}{8}$	$1.10e^{-14}$	$2.52e^{-15}$	0.1872
$[0,6]$	32	$\frac{3}{16}$	$1.24e^{-15}$	$3.82e^{-15}$	0.2652
$[0,6]$	64	$\frac{3}{32}$	$3.68e^{-15}$	$1.11e^{-15}$	0.4056

Tabelle 7. Resultate für Funktion Nr.2
(Algorithmus mit reeller Dämpfung)

$[L,U]$	M	Δ	$E_{\max}$	MAE	$CPU(s)$
$[0,6]$	8	$\frac{3}{4}$	$8.30e^{-14}$	$2.28e^{-14}$	0.1248
$[0,6]$	16	$\frac{3}{8}$	$7.28e^{-15}$	$1.58e^{-15}$	0.1872
$[0,6]$	32	$\frac{3}{16}$	$5.47e^{-15}$	$2.60e^{-15}$	0.2496
$[0,6]$	64	$\frac{3}{32}$	$2.16e^{-15}$	$6.42e^{-15}$	0.3432

Tabelle 8. Resultate für Funktion Nr.3
(Algorithmus mit reeller Dämpfung)

$[L,U]$	M	Δ	$E_{\max}$	MAE	$CPU(s)$
$[-3,3]$	8	$\frac{3}{4}$	$4.68e^{-07}$	$1.42e^{-07}$	0.1248
$[-3,3]$	16	$\frac{3}{8}$	$1.88e^{-10}$	$5.20e^{-11}$	0.1716
$[-3,3]$	32	$\frac{3}{16}$	$8.68e^{-15}$	$2.41e^{-15}$	0.2964
$[-3,3]$	64	$\frac{3}{32}$	$1.28e^{-15}$	$3.36e^{-15}$	0.4056

Tabelle 9. Resultate für Funktion Nr.4
(Algorithmus mit reeller Dämpfung)

$[L,U]$	M	Δ	$E_{\max}$	MAE	$CPU(s)$
$[0,6]$	8	$\frac{3}{4}$	$2.12e^{-12}$	$5.40e^{-13}$	0.1248
$[0,6]$	16	$\frac{3}{8}$	$6.25e^{-14}$	$1.87e^{-14}$	0.1404
$[0,6]$	32	$\frac{3}{16}$	$1.08e^{-14}$	$4.81e^{-14}$	0.2340
$[0,6]$	64	$\frac{3}{32}$	$4.66e^{-14}$	$1.44e^{-14}$	0.4368

Die beiden Algorithmen liefern bei ziemlich ähnlicher Laufzeit sehr unterschiedliche Fehlergrößen. Der verbesserte Algorithmus liefert in den selben Vergleichsintervallen bei ähnlicher Laufzeit deutlich bessere Approximationen. Die Qualität der Approximation liegt vorallem an der Größenordungn der Fast Fourier Transform. Bei der verbesserten

Version des Algorithmus wird, bei selber Laufzeit, eine FFT der Größe $8M$ durchgeführt hingegen im Originalalgorithmus liegt die Größenordnung der FFT nur bei M.

Die beiden Algorithmen zur Berechnung der Fouriertransformierten wurde an den selben Funktionen aus Tabelle 1 getestet. Im Anschluss wird wieder der absolute Fehler sowie der maximale Fehler in tabellarischer Form angeführt.

In den Tabellen 10-13 findet man die Ergebnisse des Originalalgorithmus, und in den Tabellen 14-17 jene des verbesserten Algorithmus mit reeller Dämpfung. Die angegebene Laufzeit wird wieder in Sekunden angegeben.

Tabelle 10. Resultate für Funktion Nr.1 (Originalalgorithmus)

$[L,U]$	M	Δ	$E_{\max}$	MAE	$CPU(s)$
$[0,6]$	8	$\frac{3}{4}$	$1.20e^{-04}$	$9.47e^{-05}$	0.1295
$[0,6]$	16	$\frac{3}{8}$	$6.25e^{-05}$	$1.08e^{-05}$	0.1404
$[0,6]$	32	$\frac{3}{16}$	$1.23e^{-05}$	$1.15e^{-05}$	0.2340
$[0,6]$	64	$\frac{3}{32}$	$4.66e^{-05}$	$1.44e^{-06}$	0.4368

Tabelle 11. Resultate für Funktion Nr.2 (Originalalgorithmus)

$[L,U]$	M	Δ	$E_{\max}$	MAE	$CPU(s)$
$[0,6]$	8	$\frac{3}{4}$	$2.48e^{-04}$	$1.95e^{-04}$	0.0936
$[0,6]$	16	$\frac{3}{8}$	$6.25e^{-05}$	$2.03e^{-05}$	0.1092
$[0,6]$	32	$\frac{3}{16}$	$2.48e^{-05}$	$2.06e^{-05}$	0.2964
$[0,6]$	64	$\frac{3}{32}$	$2.48e^{-05}$	$2.07e^{-05}$	1.0764

Tabelle 12. Resultate für Funktion Nr.3 (Originalalgorithmus)

$[L,U]$	M	Δ	$E_{\max}$	MAE	$CPU(s)$
$[-3,3]$	8	$\frac{3}{4}$	$3.12e^{-06}$	$3.03e^{-06}$	0.0468
$[-3,3]$	16	$\frac{3}{8}$	$9.69e^{-06}$	$3.70e^{-06}$	0.0624
$[-3,3]$	32	$\frac{3}{16}$	$4.64e^{-07}$	$1.33e^{-07}$	0.2496
$[-3,3]$	64	$\frac{3}{32}$	$1.20e^{-07}$	$3.29e^{-07}$	0.9828

Tabelle 13. Resultate für Funktion Nr.4
(Originalalgorithmus)

$[L,U]$	M	Δ	$E_{\max}$	MAE	$CPU(s)$
$[0,6]$	8	$\frac{3}{4}$	$1.43e^{-05}$	$1.08e^{-05}$	0.0748
$[0,6]$	16	$\frac{3}{8}$	$6.25e^{-05}$	$1.27e^{-06}$	0.1248
$[0,6]$	32	$\frac{3}{16}$	$1.47e^{-05}$	$1.37e^{-06}$	0.2340
$[0,6]$	64	$\frac{3}{32}$	$1.00e^{-06}$	$1.42e^{-06}$	0.9048

Tabelle 14. Resultate für Funktion Nr.1
(Algorithmus mit reeller Dämpfung)

$[L,U]$	M	Δ	$E_{\max}$	MAE	$CPU(s)$
$[0,6]$	8	$\frac{3}{4}$	$1.88e^{-07}$	$2.60e^{-08}$	0.0938
$[0,6]$	16	$\frac{3}{8}$	$2.08e^{-07}$	$2.52e^{-07}$	0.2028
$[0,6]$	32	$\frac{3}{16}$	$2.08e^{-07}$	$4.96e^{-08}$	0.3276
$[0,6]$	64	$\frac{3}{32}$	$2.08e^{-07}$	$3.00e^{-08}$	0.5616

Tabelle 15. Resultate für Funktion Nr.2
(Algorithmus mit reeller Dämpfung)

$[L,U]$	M	Δ	$E_{\max}$	MAE	$CPU(s)$
$[0,6]$	8	$\frac{3}{4}$	$2.39e^{-07}$	$1.72e^{-07}$	0.1560
$[0,6]$	16	$\frac{3}{8}$	$2.39e^{-07}$	$1.20e^{-07}$	0.1872
$[0,6]$	32	$\frac{3}{16}$	$2.39e^{-08}$	$7.77e^{-08}$	0.2964
$[0,6]$	64	$\frac{3}{32}$	$2.39e^{-08}$	$4.76e^{-08}$	0.5928

Tabelle 16. Resultate für Funktion Nr.3
(Algorithmus mit reeller Dämpfung)

$[L,U]$	M	Δ	$E_{\max}$	MAE	$CPU(s)$
$[-3,3]$	8	$\frac{3}{4}$	$2.39e^{-07}$	$1.72e^{-07}$	0.1560
$[-3,3]$	16	$\frac{3}{8}$	$2.39e^{-07}$	$1.20e^{-07}$	0.1872
$[-3,3]$	32	$\frac{3}{16}$	$2.39e^{-08}$	$7.77e^{-08}$	0.2964
$[-3,3]$	64	$\frac{3}{32}$	$2.39e^{-08}$	$4.76e^{-08}$	0.5928

Tabelle 17. Resultate für Funktion Nr.4
(Algorithmus mit reeller Dämpfung)

$[L,U]$	M	Δ	$E_{\max}$	MAE	$CPU(s)$
$[0,6]$	8	$\frac{3}{4}$	$2.18e^{-07}$	$1.30e^{-07}$	0.1248
$[0,6]$	16	$\frac{3}{8}$	$2.18e^{-07}$	$8.94e^{-08}$	0.2028
$[0,6]$	32	$\frac{3}{16}$	$2.18e^{-08}$	$5.70e^{-08}$	0.3432
$[0,6]$	64	$\frac{3}{32}$	$2.18e^{-08}$	$3.47e^{-08}$	0.7332

Auch hier zeigt sich wieder die besser Funktionsweise des zweiten Algorithmus. Die Fehler haben bei beiden Algorithmen jeweils eine ähnliche Größenordnung. Die Funktionswerte der Fouriertransformierten wurden mittels der Relationen in (5.6) sowie (5.17) berechnet. Da die Fouriertransformierten durch diese Relationen an sehr speziellen Stellen ausgewertet werden, sind diese sehr empfindlich bezüglich der gewählten Größen von Δ. Die Berechnung der Funktionswerte der Fouriertransformierten mittels der Fourierreihenentwicklung wurde hier nicht durchgeführt.

Abschließend wird noch die kombinierte Anwendung der beiden Algorithmen demonstriert. Zuerst werden die Koeffizienten der Fouriertransformierten aus der Originalfunktion berechnet (mit dem verbesserten Algorithmus mit reeller Dämpfung) und daraus wieder die Koeffizienten der Originalfunktion. Die Fehler wurden bezüglich der Originalfunktion berechnet.

Tabelle 18. Kombinierte Anwendung

Nr.	$[L,U]$	M	Δ	$E_{\max}$	MAE	$CPU(s)$
3.	$[-3,3]$	32	$\frac{3}{16}$	$7.94e^{-12}$	$2.54e^{-13}$	0.6970
4.	$[0,6]$	32	$\frac{3}{16}$	$3.18e^{-11}$	$8.18e^{-12}$	0.6084

5.6 Modifikationen für nicht stetige Funktionen

5.6.1 Stückweise stetige Funktionen

In diesem Abschnitt werden die nötigen Änderungen beschrieben um den zuvor dargestellten Algorithmus auch für stückweise stetige Funktionen anwenden zu können. Als Ausgangspunkt wird der verbesserte Algorithmus mit reeller Dämpfung aus Abschnitt 5.3 gewählt.

Sei die Funktion f eine stückweise stetige Funktion mit Diskontinuitäten in k, für

$k \in \mathbb{Z}$. Mit

$$f_j(x) = \begin{cases} f(j+x) & 0 \le x < 1 \\ 0 & sonst \end{cases}$$

gilt

$$f(u) = \sum_{j\in\mathbb{Z}} f_j(u-j).$$

Für die Fouriertransformierte von f folgt daraus

$$\begin{aligned} \hat{f}(t) &= \int_{-\infty}^{\infty} e^{-itu} f(u)\,du = \int_{\infty}^{\infty} e^{-itu} \sum_{j\in\mathbb{Z}} f_j(u-j)\,du \\ &= \sum_{j\in\mathbb{Z}} \int_{-\infty}^{\infty} e^{-itu} f_j(u-j)\,du = \sum_{j\in\mathbb{Z}} \int_{-\infty}^{\infty} e^{-it(x+j)} f_j(x)\,dx \\ &= \sum_{j\in\mathbb{Z}} e^{-itj} \int_{-\infty}^{\infty} e^{-itx} f_j(x)\,dx \\ &= \sum_{j\in\mathbb{Z}} e^{-itj} \hat{f}_j(t) =: V(t, e^{-t}). \end{aligned}$$

Das heißt, die Fouriertransformierte einer stückweise stetigen Funktion kann als Funktion V beschrieben werden, abhängig von der Fouriertransformierten einer in $(0,1)$ stetigen Funktion und einer Exponentialfunktion.

Da es sich bei f_j um eine in $(0,1)$ stetige Funktion handelt, können die in den Abschnitten zuvor gefundenen Methoden für stetige Funktionen auch hier angewandt werden.

Sei $\nu = 0.5$. Mit den Christoffel-Zahlen $w_k^{0.5}$ wie in Satz 3.0.6, sowie $a \in \mathbb{R}$ und $\alpha = -ai - \pi + 2\pi\omega$ mit $\omega \in (0,1)$ gilt für $t \in \mathbb{Z}$

$$\begin{aligned} e^{\pi i 0.5} i^{-n} \sum_{k=0}^{n-1} w_k^{0.5} e^{-(-ai+2\pi\omega)ij} \psi \hat{f}_j(t+\alpha) q_n^{0.5}(t) &= \langle e^{-(-ai+2\pi\omega)ij} \psi \hat{f}_{j\alpha}(t), e^{-\pi i 0.5} i^n q_n^{0.5}(t) \rangle_{Q,n} \\ &\overset{\text{Gauß}}{\sim} \langle e^{-(-ai+2\pi\omega)ij} \psi \hat{f}_{j\alpha}(t), e^{-\pi i 0.5} i^n q_n^{0.5}(t) \rangle_{Q} \\ &\overset{\text{PSF}}{=} \sum_{k=-\infty}^{\infty} e^{-a(k+j)} e^{-2\pi i\omega(k+j)} \langle f(k), \phi_n \rangle_{L^2[0,1]} \\ &= \sum_{k=j}^{\infty} e^{-ak} e^{-2\pi ik\omega} \langle f, \phi_n \rangle_{L^2[0,1]}. \end{aligned}$$

Weiters gilt

$$\begin{aligned} \sum_{j\in\mathbb{Z}} \sum_{k=0}^{n-1} w_k^{0.5} e^{-(-ai+2\pi\omega)ij} \psi \hat{f}_j(t+\alpha) q_n^{0.5}(t) &= \sum_{k=0}^{n-1} \sum_{j\in\mathbb{Z}} w_k^{0.5} e^{-(-ai+2\pi\omega)ij} \psi \hat{f}_j(t+\alpha) q_n^{0.5}(t) \\ &= \sum_{k=0}^{n-1} w_k^{0.5} \psi V(t+\alpha, e^{-ai+2\pi\omega}) q_n^{0.5}(t). \end{aligned}$$

Wobei folgender Zusammenhang verwendet wurde

$$\psi V(t+\alpha, e^{-ai+2\pi\omega}) = \frac{1}{t} V(\frac{1}{t}+\alpha, e^{-ai+2\pi\omega}).$$

Zusammen führen diese Ergebnisse auf

$$\sum_{k=-\infty}^{\infty} e^{-ak} e^{-2\pi ik\nu} \langle f, \phi_n \rangle_{L^2[0,1]} \sim e^{\pi i\nu} i^{-n} \sum_{k=0}^{n-1} w_k^{0.5} \psi V(t+\alpha, e^{-ai+2\pi\omega}) q_n^{0.5}(t). \tag{5.20}$$

Bemerkung 5.6.1. Sei f eine stückweise stetige Funktion mit Diskontinuitäten in Δk, für $k \in \mathbb{Z}$. Dann kann die Fouriertransformation durch Verwendung von $\hat{f} = V(t, e^{-t\Delta})$ berechnet werden.

Der Pseudocode für den Algorithmus zur Berechnung der Inversen für stückweise stetige Funktionen

Pseudocode

Input Parameter: n, den Träger der Funktion $[L, U]$, die Anzahl der Intervalle M, $\Delta = \frac{U-L}{M}$, $M_2 = 8M, a = \frac{44}{M_2}$
(ohne reelle Dämpfung gilt $a = 0$ und $M_2 = M$)

Input: Vektor der Funktionswerte der Fouriertransformierten
$\psi \vec{\hat{f}}_\alpha^{\Delta} = \Bigg[\frac{\exp(Li(\alpha+\pi)/\Delta)}{\eta_0^{0.5}\Delta} V\left(\frac{1}{\Delta}\left(\frac{1}{\eta_0^{0.5}}+\alpha\right), e^{-(a+\frac{2\pi k}{M_2})}\right), \ldots,$
$\frac{\exp(Li(\alpha+\pi)/\Delta)}{\eta_{n-1}^{0.5}\Delta} V\left(\frac{1}{\Delta}\left(\frac{1}{\eta_{n-1}^{0.5}}+\alpha\right), e^{-(a+\frac{2\pi k}{M_2})}\right)\Bigg] \in \mathbb{C},$
für $\nu = 0.5, \quad \alpha = -ai - \pi + \frac{2\pi j}{M_2} \quad$ und $0 \le j < M_2$

Output: $f(L + \Delta(j+x))$ für $x \in [0,1)$, $\Delta \in \mathbb{R}^+$ und $0 \le j < M$

Schritt 1: Berechnung von $\vec{L}(0.5) = UD\left(e^{\alpha i\lambda}\right) U^t V_0 \psi \vec{\hat{f}}_\alpha^{\Delta}$ mit Hilfe der in (5.19) gefundenen Darstellung. Danach wird weiters $L_k(0.5) = i^{-k} e^{\pi i 0.5} (\vec{L}(0.5))_k$ berechnet, für $0 \le k < n$.

Schritt 2: Berechne die Werte von f_{kj} mit Hilfe der IFFT für $0 \le k < n$ und $0 \le j < M_2$

$$f_{kj} = \frac{1}{M_2} \sum_{l=0}^{M_2-1} L_k(0.5) e^{\frac{2\pi i l j}{M_2}}.$$

Schritt 3: $\langle f(L+\Delta(j+\cdot)), \phi_k \rangle_n = f_{kj} e^{aj}$ für $0 \le k < n$ und $0 \le j < M$.

Schritt 4: Berechnung der Funktionswerte mittels der abgebrochenen Fourierreihe $0 \leq j < M$

$$f(L + \Delta(j + x)) \sim \sum_{k=0}^{n-1} \langle f(L + \Delta(j + \cdot)), \phi_k \rangle_n \phi_k(x), \qquad x \in [0, 1).$$

Bemerkung 5.6.2. Die Fouriertransformierte der Funktion f kann mit Hilfe des in Abschnitt 5.4.1 beschriebenen Algorithmus für stetige Funktionen, angewandt auf die Funktion f_j berechnet werden. Dieses Ergebniss muss im Anschluss noch mit der Exponentialfunktion passend adaptiert werden.

Testresultate für stückweise stetige Funktionen

Die Funktionsweise des soeben modifizierten Algorithmus wird an zwei stückweise stetigen Funktion demonstriert. Wie zuvor werden in der Tabelle die absoluten Fehler, sowie die maximalen Fehler dargestellt. Die Definition der Fehler kann in Abschnitt 5.5 nachgelesen werden, auch die Input Paramter werden analog zu Abschnitt 5.5 gewählt.

Tabelle 19. stückweise stetige Testfunktionen

Nr.	Funktion	Fouriertransformierte
5.	$H(x-1)e^{-x}$	$(it+1)^{-1}e^{-(it+1)}$
6.	Square(x)	$(it+1)^{-1}(1+e^{-(it+1)})^{-1}$

Wobei die Funktion $H(x-1)$ die Sprungfunktion ist, für die gilt

$$H(x-1) = \begin{cases} 0 & x \leq 1 \\ 1 & x > 1 \end{cases}.$$

Die Funktion Square steht für eine Square-Wave, die hier folgende Form hat

$$Square(x) = \begin{cases} 1 & 0 \leq x < 1 \\ 0 & 1 \leq x < 2 \end{cases}$$

mit $Square(x+2) = Square(x)$.

Tabelle 20. Resultate für stückweise stetige Funktionen

Nr.	$[L, U]$	M	Δ	$E_{\max}$	MAE	$CPU(s)$
5.	$[0, 2]$	32	$\frac{1}{16}$	$2.89e^{-14}$	$6.89e^{-15}$	0.23
6.	$[0, 2]$	32	$\frac{1}{16}$	$4.77e^{-13}$	$4.82e^{-14}$	0.20

Auch hier zeigt sich wieder die hohe Genauigkeit und die Schnelligkeit des Algorithmus.

5.6.2 Funktionen mit Singularitäten

Es ist weiters möglich den Algorithmus auf Funktionen mit Singularitäten anzuwenden. Der Algorithmus liefert genaue Ergebnisse für Funktionen mit beliebigen aber a priori bekannten Singularitäten.

Die nachfolgenden Änderungen betreffen wieder den verbesserten Algorithmus zur Berechnung der Inversen aus Abschnitt 5.4.2.

Sei eine Singularität der Funktion f gegeben in $t = s$ mit $s \in \mathbb{R}$, $s \neq 0$ und sei die folgende Funktion f_w definiert durch

$$f_w(x) := w(x)^q f(x), \tag{5.21}$$

mit $w(x)$ der sogenannten Window - Funktion. Die Window-Funktion ist ein trigonometrisches Polynom mit Periode 1 und $w(0) = 1, w(s) = 0$. Wir setzen voraus, dass $q \in \mathbb{Z}^+$ so gewählt werden kann, dass f_w stetig in der Singularität s ist. Es gilt

$$f_w(k) = f(k) \qquad (k \in \mathbb{Z}).$$

Bemerkung 5.6.3. Für mehrere Singularitäten s_j mit $0 \leq j \leq m$ besitzt die Window-Funktion folgende Darstellung

$$w^q(x) = \prod_{j=1}^{m} w_j^q(x).$$

Wobei w_j wieder ein trigonometrisches Polynom mit Periode 1 ist und wieder gilt $w_j(0) = 1$ und $w_j(s_j) = 0$.

Window-Funktionen

Mit den nachfolgenden Klassen von Window-Funktionen können die besten Rechenergebnisse im Bezug auf diesen Algorithmus erzielt werden. Sei $t = s$ die Singularität der Funktion f, wobei zunächst $s \neq 0$ gilt. Die Window-Funktion ist gegeben durch

$$\begin{aligned} w(x) &= \left(\cos(p\pi x) - \frac{\cos(p\pi s)}{\sin(p\pi s)}\sin(p\pi x)\right)^2 \\ &= (Ae^{ip\pi x} + Be^{-ip\pi x})^2, \qquad (p \in \mathbb{N}) \end{aligned}$$

mit den Koeffizienten

$$A = \left(\frac{1}{2} - \frac{1}{2i}\frac{\cos(p\pi s)}{\sin(p\pi s)}\right) \text{ und } B = \left(\frac{1}{2} + \frac{1}{2i}\frac{\cos(p\pi s)}{\sin(p\pi s)}\right).$$

Der Wert p wird so gewählt, dass $(ps \ mod \ 1)$ in der Nähe von $1/2$ liegt.

Die Fouriertransformierte der Funktion f_w hat unter Verwendung der soeben definierten Window-Funktion die Form,

$$\begin{aligned}\hat{f}_w(t) &= \int_{-\infty}^{\infty} e^{-itu}(Ae^{ip\pi u} + Be^{-ip\pi u})^{2q} f(u)\,du \\ &= \int_{-\infty}^{\infty} \sum_{k=0}^{2q} \binom{2}{q} kA^{2q-k}B^k e^{-(it-(2q-2k)ip\pi)u} f(u)\,du \\ &= \sum_{k=0}^{2q} \binom{2}{q} kA^{2q-k}B^k \hat{f}(t - 2\pi p(q-k)).\end{aligned}$$

Für Funktionen mit Singularitäten in $t = 0$ müssen andere Window-Funktionen gewählt werden, denn die Koeffizienten A und B sind im Punkt 0 nicht definiert. Deshalb wählt man die folgende Window-Funktion im Falle einer Singularität im Punkt 0,

$$w(x) = \sin^2\left(\frac{\pi x}{2}\right).$$

Die Funktion w hat Periode 2 und es gilt $w(1) = 1$ sowie $w(s) = 0$. Wir setzen voraus, dass ein $q \in \mathbb{Z}^+$ existiert, sodass die Funktion f_w stetig in $t = 0$ ist. Es gilt

$$f_w(2k+1) = f(2k+1) \qquad (k \in \mathbb{Z}).$$

Für die Fouriertransformierte von f_w gilt

$$\begin{aligned}\hat{f}_w(t) &= \int_{-\infty}^{\infty} e^{-itu}\left(\sin\left(\frac{\pi u}{2}\right)\right)^{2q} f(u)\,du \\ &= \int_{-\infty}^{\infty} e^{-itu}\left(\frac{e^{i(\pi/2)u} - e^{-i(\pi/2)u}}{2i}\right)^{2q} f(u)\,du \\ &= \int_{-\infty}^{\infty} e^{-itu}\left(\frac{1}{2}\right)^{2q} \sum_{k=0}^{2q} \binom{2q}{k} e^{(2q-k)i(\pi/2)u} e^{-i(\pi/2)uk} i^{-2(q-k)} f(u)\,du \\ &= \left(\frac{1}{2}\right)^{2q} \sum_{k=0}^{2q} \binom{2q}{k} (-1)^{q-k} \int_{-\infty}^{\infty} e^{-(itu+(q-k)i\pi u)} f(u)\,du \\ &= \left(\frac{1}{2}\right)^{2q} \sum_{k=0}^{2q} \binom{2q}{k} (-1)^{q-k} \hat{f}(t + \pi(q-k)).\end{aligned}$$

Bemerkung 5.6.4. Die Funktionswerte von f können aus $\hat{f}_w$ berechnet werden, unter Verwendung des Algorithmus für stetige Funktionen. Die Fouriertransformierte von f kann unter Verwendung von f_w mit dem in Abschnitt 5.4.2 beschriebenen Algorithmus berechnet werden.

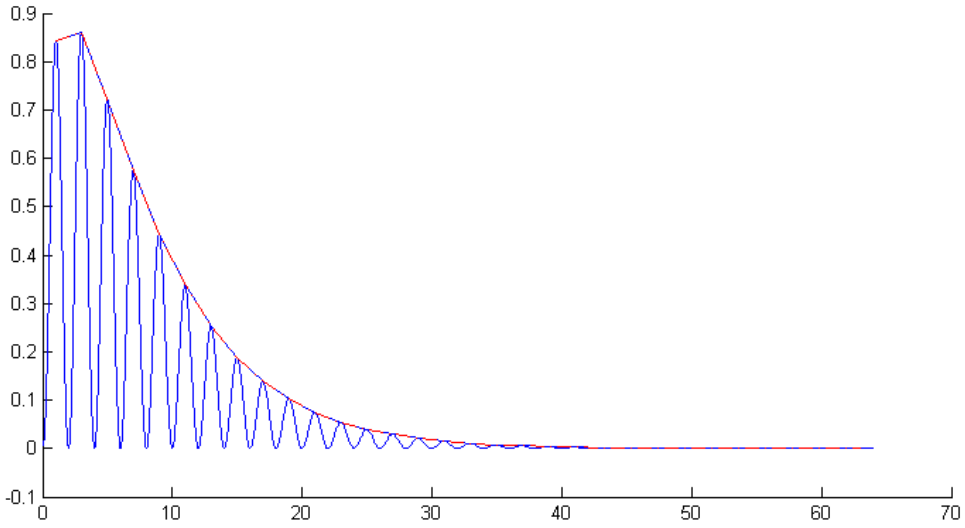

Abbildung 5.1: *Funktionsweise der Window-Funktion für Funktion Nr.10*

Numerische Testresultate für Funktionen mit Singularitäten

Der Algorithmus wurde für eine Reihe von Funktionen mit Singularitäten im Punkt 0 getestet. Als Input Parameter wurde für alle Funktionen $n = 32$ fixiert. Die weiteren Parameter werden in Abhängigkeit des Parameters M berechnet. In den Tabellen werden der maximale Fehler, der MAE sowie die Trägerintervalle und die benötigte Laufzeit in Sekunden angegeben.

Tabelle 21. Funktionen mit Singularitäten in x=0

Nr.	Funktion	Fouriertransformierte
7.	$(\pi x)^{-0.5}e^{-x/4}I(0,\infty)$	$(it+1/4)^{-0.5}$
8.	$(e^{-x/4}-e^{-x/2})(4\pi x^3)^{-0.5}I(0,\infty)$	$(it+0.5)^{0.5}-(it+0.25)^{0.5}$
9.	$x^{-1/3}e^{-44x/256}I(0,\infty)$	$\Gamma(2/3)(it+44/256)^{-2/3}$
10.	$x^{-1/4}e^{-x/8}I(0,\infty)$	$\Gamma(3/4)(it+1/8)^{-3/4}$

Tabelle 22. Resultate für Funktion Nr. 7

$[L,U]$	M	Δ	$E_{\max}$	MAE	$CPU(s)$	q
$[0,20]$	64	$\frac{20}{64}$	$4.36e^{-10}$	$1.15e^{-10}$	31.09	2
$[0,20]$	128	$\frac{20}{128}$	$1.41e^{-09}$	3.05^{-11}	61.51	2

Tabelle 23. Resultate für Funktion Nr. 8

$[L,U]$	M	Δ	$E_{\max}$	MAE	$CPU(s)$	q
$[0,20]$	64	$\frac{20}{64}$	$1.32e^{-10}$	$2.56e^{-10}$	31.37	2
$[0,20]$	128	$\frac{20}{128}$	$4.93e^{-10}$	1.54^{-11}	61.74	2

Tabelle 24. Resultate für Funktion Nr. 9

$[L,U]$	M	Δ	$E_{\max}$	MAE	$CPU(s)$	q
$[0,40]$	64	$\frac{40}{64}$	$1.32e^{-11}$	$7.36e^{-11}$	32.91	2
$[0,40]$	128	$\frac{40}{128}$	$1.42e^{-10}$	3.05^{-11}	67.01	2

Tabelle 25. Resultate für Funktion Nr. 10

$[L,U]$	M	Δ	$E_{\max}$	MAE	$CPU(s)$	q
$[0,60]$	64	$\frac{60}{64}$	$2.37e^{-10}$	$6.56e^{-10}$	31.00	2
$[0,60]$	128	$\frac{60}{128}$	$1.39e^{-11}$	5.06^{-12}	67.9	2

Die Ergebnisse in der Tabelle zeigen wieder die gute Funktionsweise des Algorithmus, jedoch auch die längeren Laufzeiten. Die Ergebnisse sind für alle Werte von Δ von annähernd gleicher Qualität.

Kapitel 6

Preiskalkulation für Asiatische Optionen

In diesem Kapitel wird zuerst ein kurzer Überblick über exotische Optionen gegeben, um dann speziell auf die Bewertung diskreter Asiatischer Optionen einzugehen. Eine Bepreisungsmethode für diesen Optionstyp wird detailiert dargestellt und im Anschluss wird der in Kapitel 5 beschriebene Algorithmus verwendet um die Optionspreise im Black-Scholes-Modell zu berechnen.

6.1 Exotische Optionen

Allgemein bezeichnet man als exotische Optionen jene Optionstypen die keine europäischen Standardoptionen sind. Somit bilden exotische Optionen einen sehr große Gruppe von Verträgen. Insbesondere zählen Verträge dazu, deren Auszahlung nicht nur vom Schlusskurs des Wertpapieres zum Ausübungszeitpunkt, sondern auch von der Kursentwicklung bis zu diesem Zeitpunkt beeinflusst wird. Die Berücksichtigung des gesamten Kursverlaufes zur Berechnung des Auszahlungswertes bezeichnet man als Pfadabhängigkeit.

Wichtige Vertreter von pfandabhängigen, exotischen Optionen sind Asiatische Optionen deren Auszahlung von einem Durchschnittskurs bestimmt wird, Barrier Optionen deren Wert vom Erreichen einer Kursschranke abhängig gemacht wird oder sogenannte Lookback-Optionen deren Wert durch die Extremwerte des Kursverlaufes beeinflusst wird. Für zahlreiche spezielle Beispiele exotischer Optionen vergleiche [21] sowie [26].

6.2 Bepreisungsmethode für diskrete Asiatische Optionen

Wie in der Einleitung dieses Kapitels erwähnt, gehören Asiatische Optionen zur Familie der pfadabhängigen, exotischen Optionen. Die Auszahlung dieser Optionen beruht auf dem Durchschnittskurs des zugrundeliegenden Wertpapiers in einem vordefinierten Zeitintervall.

Die wichtigsten Basisinstrumente für Asiatische Optionen sind Wechselkurse, Rohstoffpreise, Zinssätze und Renditen.

Für Asiatische Optionen lassen sich eine Vielzahl von Typisierungen unterscheiden. Vorallem bezüglich der Durchschnittsbildung lassen sich mehrere Untergruppen bilden. Man spricht von einer diskreten Asiatischen Option, wenn der Durchschnittskurs nur aus endlich vielen Wertpapierkursen, aus einem bestimmten Zeitintervall, berechnet wird. Es kann sich zum Beispiel um Tages-, Wochen- oder Jahresabschlusskurse handeln. Bei stetigen Asiatischen Optionen wird der Durchschnitt zeitstetig gebildet, d.h. es werden alle Kurse während der Laufzeit zur Mittelwertbildung herangezogen.

Weiters kann zwischen geometrischer bzw. arithmetischer Durchschnittsbildung unterschieden werden. In der Praxis werden fast ausschließlich diskrete Asiatische Optionen verwendet, und meist wird der Durchschnitt mit Hilfe des arithmetischen Mittels gebildet.

Ein weitere Unterscheidung kann zwischen festem oder variablen Strikepreis gemacht werden. Handelt es sich beim Strikepreis um eine, bei Vertragsbeginn, festgelegte konstante Größe so spricht man von einer *fixed strike* Option. Die zweite Möglichkeit ist, den Schlusskurs des Wertpapiers mit dem Durchschnittskurs zu vergleichen. Diese Verträge bezeichnet man als *floating strike* Optionen.

Zur Bewertung von Asiatischen Optione gibt es unterschiedliche Möglichkeiten. Für den Fall der zeitstetigen Asiatischen Optionen sei auf die Arbeit von Geman und Yor [14] hingewiesen, die die Laplacetransformierte verwendet.

In der Praxis ist es jedoch nötig diskrete Optionen bewerten zu können. Hier wird auf die Arbeit von Carverhill und Clewlow [1] hingewiesen. Diese verwendet die Fouriertransformation zur Berechnung der Optionspreise. Eine Weiterentwicklung dieser Arbeit stammt von Benhamou [4]. Fusai und Meucci [13] verwenden ebenfalls die Fouriertransformation zur Bepreisung von diskreten und zeitstetigen Asiatischen Optionen.

Zunächst wird nun eine Bepreisungsmethode für diskrete, arithmetische Asiatische fixed strike Optionen näher beschrieben.

Seien $0 = t_0 < t_1 < \cdots < t_n = T$ vorgegebene Zeitpunkte und S_{t_i} der Preis des zugrundeliegenden Wertpapieres zum Zeitpunkt t_i. Der Return im Intervall $[t_{i-1}, t_i]$ ist definiert durch $X_{t_i} = S_{t_i}/S_{t_{i-1}}$. Es wird davon ausgegangen, dass die Returns $X_{t_1}, \ldots, X_{t_n}$ unabhängig sind. Da im weiteren Verlauf der Logarithmus des Returns benötigt wird, sei dieser definiert durch $R_{t_i} = \log(S_{t_i}/S_{t_{i-1}})$. Die Dichte des jeweiligen Returns R_{t_i} sei bekannt und werde mit f_{R_i} bezeichnet. Der Preis des Wertpapieres berechnet sich aus einem Startpreis S_{t_0}, der pro Intervall um den Return erhöht wird. Es gilt

$$S_{t_i} = S_{t_0} X_{t_1} X_{t_2} \ldots X_{t_i} = S_{t_0} e^{R_{t_1} + R_{t_2} + \cdots + R_{t_i}}.$$

Weiters bezeichne $T = t_n$ die Endfälligkeit, K den Strikepreis und r die risikolose Zinsrate. Da K zu Vertragsbeginn fixiert wird, handelt es sich um fixed strike Optionen.

Der Mittelwert der zur Preisbestimmung nötig ist, wird durch das arithmetische Mittel gebildet,

$$A := \frac{1}{n} \sum_{i=1}^{n} S_{t_i}. \tag{6.1}$$

Der Wert der asiatischen Put-Option zum Ausübungszeitpunkt T ist

$$\left(K - \frac{1}{n} \sum_{i=1}^{n} S_{t_i} \right)^+. \tag{6.2}$$

Somit lautet der Preis der Asiatischen Put-Option zum Zeitpunkt t

$$S_t(P) := e^{-r(T-t)} \mathbb{E}_Q \left(K - \frac{1}{n} \sum_{i=1}^{n} S_{t_i} \right)^+,$$

bzw. der einer Asiatischen Call-Option

$$S_t(C) := e^{-r(T-t)} \mathbb{E}_Q \left(\frac{1}{n} \sum_{i=1}^{n} S_{t_i} - K \right)^+.$$

Wobei $\mathbb{E}_Q$ den Erwartungswert bezüglich dem risikoneutralem Maß Q beschreibt und $X^+ = \max(0, X)$. Näheres zur Preistheorie bzw. die Definition eines risikoneutralen Maßes kann in Abschnitt 6.4.1 nachgelesen werden.

Da es nicht möglich ist für den Mittelwert in (6.1) eine Dichtefunktion in geschlossener Form anzugeben, ist es auch nicht möglich eine geschlossene Formel zur Preisberechnung anzugeben. Um den Preis zu berechnen, wird die Verteilung bzw. die Dichtefunktion des arithmetischen Mittels benötigt.

Es wird in der Folge ein Verfahren zur Bestimmung dieser Dichte und somit des Optionspreises, mittels der Fouriertransformation, vorgestellt. Da sich die Summe in (6.1) aus korrelierten Termen zusammensetzt, muss auf ein rekursive Methode zurückgegriffen werden.

Zuerst werden die zu berechnenden Mittelwerte als Funktionen der Returns dargestellt.

Seien zunächst

$$\begin{aligned} S_{1,n} &:= \sum_{i=1}^{n} \frac{S_{t_i}}{S_{t_0}} \\ S_{k,n} &:= \sum_{i=k}^{n} \prod_{j=k}^{i} X_{t_j} = \sum_{i=k}^{n} \frac{S_{t_i}}{S_{t_{k-1}}}. \end{aligned}$$

Für den in (6.1) definierten Mittelwert gilt

$$A = \frac{S_{t_0}}{n} S_{1,n}.$$

Weiters gilt

$$S_{n,n} = \frac{S_{t_n}}{S_{t_{n-1}}} = X_{t_n}, \tag{6.3}$$

und

$$\begin{aligned} S_{k-1,n} &= \sum_{i=k-1}^{n} \frac{S_{t_i}}{S_{t_{k-2}}} \\ &= \left(\sum_{i=k}^{n} \frac{S_{t_i}}{S_{t_{k-1}}} \frac{S_{t_{k-1}}}{S_{t_{k-2}}} + \frac{S_{t_{k-1}}}{S_{t_{k-2}}} \right) \\ &= X_{t_{k-1}}(1 + S_{k,n}). \end{aligned} \tag{6.4}$$

Man beachte, dass $S_{k,n}$ und der Return $X_{t_{k-1}}$ unabhängig sind.

Sei nun für ein festes n

$$B_k := \ln(S_{k,n}) \qquad (1 \leq k \leq n).$$

Dann folgt $B_n = \ln(X_{t_n})$ und

$$B_{k-1} = \ln(X_{t_{k-1}}) + \ln\left(1 + e^{B_k}\right), \qquad (2 \leq k \leq n). \tag{6.5}$$

Mit der zuvor eingeführten Bezeichung für den Logarithmus des Returns kann (6.5) auf folgende Form überführt werden

$$B_{k-1} = R_{t_{k-1}} + \ln\left(1 + e^{B_k}\right), \qquad B_n = R_{t_n}. \tag{6.6}$$

Bezeichne f_k die Dichtefunktion von B_k und g_k jene von $\ln\left(1+e^{B_k}\right)$, dann gilt

$$\begin{aligned} g_k(z) = \frac{d}{dz}\mathbb{P}\left(\ln\left(1+e^{B_k}\right) \leq z\right) &= \frac{d}{dz}\mathbb{P}\left(B_k \leq \ln\left(e^z-1\right)\right) \\ &= \frac{d}{dz}\int_{-\infty}^{\ln(e^z-1)} f_k(u)\,du \\ &= f_k\left(\ln\left(e^z-1\right)\right)\frac{e^z}{e^z-1}. \end{aligned} \tag{6.7}$$

Somit kann die Dichtefunktion von B_{k-1} durch

$$f_{k-1}(z) = (f_{R_{k-1}} * g_k)(z), \tag{6.8}$$

berechnet werden. Wegen

$$\hat{f}_{k-1}(t) = \hat{f}_{R_{k-1}}\hat{g}_k(t) \tag{6.9}$$

kann f_{k-1} durch die inverse Fouriertransformation bestimmt werden. Rekursiv kann dann die gesuchte Dichtefunktion f_1 von B_1 berechnet werden.

Der Pseudocode des Rekursionsalgorithmus zur Bestimmung der Dichte f_1 hat folgende Form.

Pseudocode

Schritt 1: k=n, $f_n(x) := f_{R_{t_n}}(x)$.

Schritt 2: Berechne die Dichte g_k

$$g_k(z) = f_k\left(\ln\left(e^z-1\right)\right)\frac{e^z}{e^z-1}.$$

Schritt 3: Bestimmung der Fouriertransformierten: $\hat{g}_k = \mathcal{F}(g_k)$.

Schritt 4: $\hat{f}_{k-1}(s) = \hat{f}_{R_{t_{k-1}}}(s)\hat{g}_k(s)$.

Schritt 5: Bestimmung der Inversen Fouriertransformierten: $f_{k-1} = \mathcal{F}^{-1}(\hat{f}_{k-1})$.

Schritt 6: k=k-1, wenn $k > 1$ dann gehe zu Schritt 2.

Mit der so gefundenen Dichte kann nun der Erwartungswert berechnet werden, der zur Preisbestimmung nötig ist. Da die errechnete Dichte jene des Logarithmus des Returns ist, kann der Preis einer Asiatische Put-Option, zum Zeitpunkt $t = t_0 = 0$, wie folgt berechnet werden.

$$
\begin{aligned}
S_0(P) &= e^{-rT}\mathbb{E}_Q\left(K - \frac{1}{n}\sum_{i=1}^{n} S_{t_i}\right)^+ && (6.10)\\
&= e^{-rT}\mathbb{E}_Q\left(K - \frac{S_{t_0}}{n} S_{1,n}\right)^+ && (6.11)\\
&= e^{-rT}\int_{-\infty}^{t}\left(K - \frac{S_{t_0}}{n}e^{x}\right)^+ f_1(x)\,dx && (6.12)\\
&= e^{-rT}\left(\int_{-\infty}^{-d_1} K f_1(x)\,dx - \int_{-\infty}^{-d_1}\frac{S_{t_0}}{n}e^{x} f_1(x)\,dx\right) && (6.13)
\end{aligned}
$$

mit $-d_1 = \log\left(Kn/S_{t_0}\right)$.

Aus den berechneten Put-Preisen ist es möglich, mittels der Call - Put - Parität die Call-Preise einer Asiatischen Option zu berrechnen. Diese lautet zum Zeitpunkt $t = t_0 = 0$.

$$S_0(P) + e^{-rT}\frac{S_{t_0}}{n}\sum_{i=1}^{n} e^{rt_i} = S_0(C) + Ke^{-rT},$$

mit den Beobachtungszeitpunkten t_i für $i = 1, \ldots, n$.

6.3 Algorithmus zur Berechnung des Optionspreises

Der zuvor skizzierte Algorithmus zur Berechnung der Dichtefunktion wird anschließend im Detail beschrieben. Siehe dazu [10].

6.3.1 Das Trägerintervall

Zu Beginn wird auf die numerischen Träger der verwendeten Funktionen eingegangen. Das Intervall $[L_f, U_f]$, wird als ein numerischer ϵ-Träger der Funktion f bezeichnet, wenn

$$\int_{L_f}^{U_f} f(t)\,dt \geq (1-\epsilon)||f||_1.$$

Ist $[L_f, U_f]$ der Träger von f und $[L_g, U_g]$ der Träger von g, dann ist der Träger von $f * g$ in $[L_f + L_g, U_f + U_g]$ enthalten.

Da in dem verwendeten Algorithmus in jedem Rekursionsschritt gefalten wird, werden die Trägerintervalle der zu berechnenden Funktionen unnötig groß. Deshalb wird ein

Intervall $[L^*, U^*]$ gesucht, für das gilt

$$\int_{L^*}^{U^*} f(t)\,dt \geq (1-\epsilon)\int_{L_{fg}}^{U_{fg}} f(t)\,dt.$$

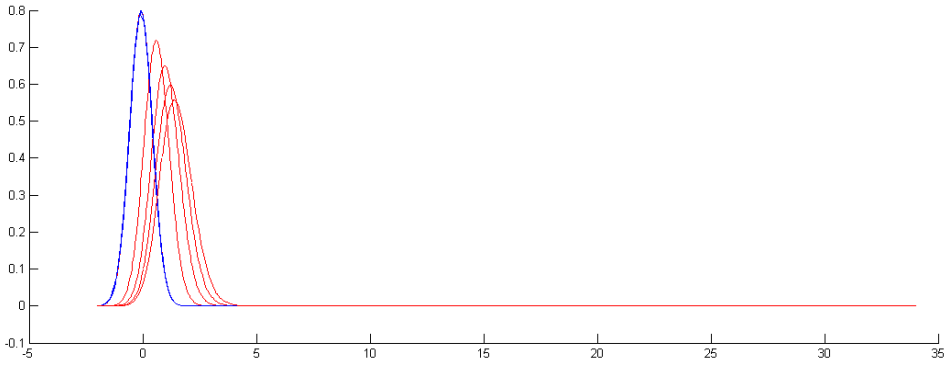

Abbildung 6.1: *Evolution der Dichtefunktionen mit zugehörigem Trägerintervall*

Mit Hilfe des folgenden Algorithmus ist es möglich $[L^*, U^*]$ zu finden.

Pseudocode Algorithmus A

Input: Die Entwicklungskoeffizienten der Funktion f, sowie ihren Träger $[L_f, U_f]$, die Anzahl M der Teilintervalle in $[L_f, U_f]$ sowie den Kontrollparameter ϵ.

Output: Ein numerischer ϵ-Träger $[L^*, U^*]$.

Schritt 1: Setze $L := L_f$, $U := L_f$, $L^* := L_f$, $U^* := U_f$, $n = 0$ und $\Delta = \frac{U_f - L_f}{M}$ sowie $Int = \int_{L_f}^{U_f} f(x)\,dx = \Delta \sum_{j=0}^{M-1} \langle f(L_f + \Delta(j+\cdot)), \phi_0 \rangle_{L^2[0,1]}$.

Schritt 2: Während U $\leq U_f - 1$ ist, wiederhole die Schritte 3-5. In den nachfolgenden Schritten wird durch kontinuierliches hinzufügen/entfernen von Entwicklungskoeffizienten festgestellt, welche Schranken benötigt werden, um das Integral der Funktion genügend gut zu berechnen.

Schritt 3: wiederhole

$n = n + \Delta\langle f(U + \Delta\cdot), \phi_0 \rangle_{L^2[0,1]}$

$U = U + \Delta$

bis $(n \geq (1-\epsilon)Int)$ oder $(U \geq U_f)$.

Schritt 4: wiederhole

$n = n - \Delta\langle f(L + \Delta\cdot), \phi_0 \rangle_{L^2[0,1]}$

$L = L + \Delta$
bis $(n \leq (1 - \epsilon)Int)$.

Schritt 5: falls $((U - L + 1) < U^* - L^*)$ und $(n \geq (1 - \epsilon)Int)$
$L^* = L - \Delta$
$U^* = U$.

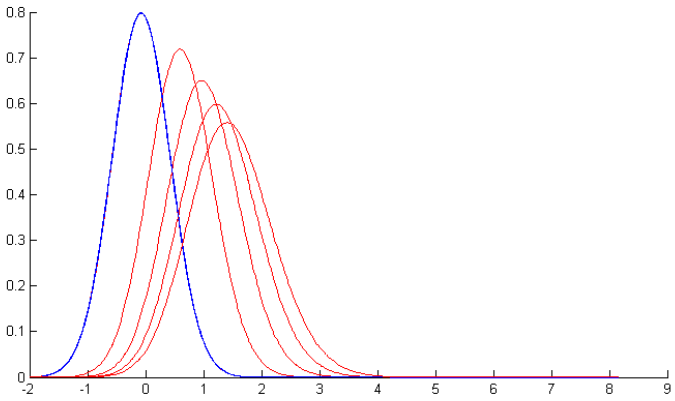

Abbildung 6.2: *Evolution der Dichtefunktionen im angepassten Trägerintervall*

6.3.2 Die numerische Faltung

In diesem Abschnitt wird die Qualität der numerische Faltung genauer betrachtet. Zunächst wird die Funktionsweise dargestellt und dann wird das Verfahren an zwei Funktionen getestet.

Als Input werden zwei Funktionen f und g verwendet. Auf beide Funktionen wird der verbesserte Algorithmus zur Bestimmung der Fouriertransformierten aus Abschnitt 5.4.1 angewandt. Die berechneten Funktionswerte der Fouriertransformierten $\hat{f}$ und $\hat{g}$ werden multipliziert und dieses Produkt wird als Input für den in Abschnitt 5.4.2 beschrieben Algorithmus zur Berechnung der Inversen verwendet.

Die Methode wurde an den nachfolgenden zwei Dichtefunktionen getestet. Es wurden jeweils zwei Funktionen mit derselben Verteilung gefalten. Diese Verteilungen wurden speziell ausgewählt, da ihre Faltungen in analytischer Form dargestellt werden können.

Nr.	Funktion 1	Funktion 2	Faltung
1.	$\frac{1}{0.2\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{x-0.1}{0.2}\right)^2}$	$\frac{1}{0.5\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{x-0.75}{0.5}\right)^2}$	$\frac{1}{\sqrt{(0.2^2+0.5^2)}\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{x-(0.1+0.75)}{\sqrt{(0.2^2+0.5^2)}}\right)^2}$
2.	$\frac{2^2}{\Gamma(2)}xe^{-2x}I(0,\infty)$	$\frac{2^2}{\Gamma(2)}xe^{-2x}I(0,\infty)$	$\frac{2^4}{\Gamma(4)}x^3e^{-2x}I(0,\infty)$

In der nachfolgenden Tabelle werden jeweils die Trägerintervalle der Einzelfunktionen sowie der Faltung dargestellt, die Anzahl der gewünschten Intervalle M und der mittlere absolute Fehler.

Nr.	$[L_1, U_1]$	$[L_2, U_2]$	$[L, U]$	M	MAE
1.	$[-1, 1]$	$[-2, 3]$	$[-3, 4]$	64	$9.96e^{-11}$
2.	$[0, 4]$	$[0, 4]$	$[0, 8]$	64	$1.16e^{-15}$

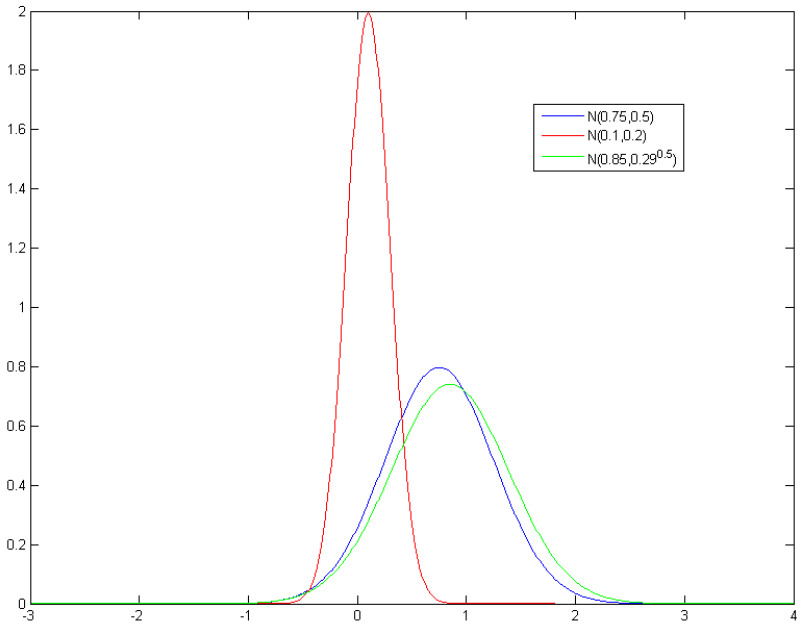

Abbildung 6.3: *Faltung der Normalverteilung (Bsp.1)*

Da nun alle nötigen Bestandteile dargestellt wurden, kann eine vollständige Beschreibung des Algortihmus zur Berechnung der Dichte gegeben werden.

6.3.3 Algorithmus zur Berechnung der Dichte

Inizialisierung

Input: Die Dichtefunktion des Returns R_{t_n} sowie dessen Träger $[L_n, U_n]$, Anzahl M der Teilintervalle von $[L_n, U_n]$.

Output: Entwicklungskoeffizienten Z_n der gesuchten Dichte f_n.

Schritt 1: $\Delta = \frac{(U_n - L_n)}{M}$.

Schritt 2: $f_{kj} = f_{R_{t_n}}(L_n + (j + \lambda_k)\Delta)$, für $0 \leq k < n$ sowie $0 \leq j < M$.
λ_k ist eine Nullstellen des Polynoms ϕ_n.

Schritt 3: Berechne den Vektor der Koeffizienten mit Hilfe der Relationen aus Satz 2.0.4
$Z_n = U_0 \vec{f_j}$ für $0 \leq j < M$. Wobei $\vec{f_j} = [f_{0j}, \ldots, f_{(n-1)j}]^t$.

Body

Input: Der Träger $[L_n, U_n]$, Anzahl der Teilintervalle M, die Entwicklungskoeffizienten Z_n, die Dichte des Returns $R_{t_{n-1}}$ sowie dessen Träger $[L_{n-1}, U_{n-1}]$.

Output: Entwicklungskoeffizienten Z_{n-1} der Dichte f_{n-1}, sowie den numerischen Träger $[L^*_{f_{n-1}}, U^*_{f_{n-1}}]$.

Schritt 1: Berechne den Träger der neuen Funktion g,
$L_g = \log(e^{L_n} + 1)$ und $U_g = \log(e^{U_n} + 1)$ sowie $\Delta_g = \frac{(U_g - L_g)}{M}$.

Schritt 2: Berechnung der Funktion g durch auswerten der Funktion f_n an verschobenen Stützstellen.
Berechnung der Funktionswerte $y_{kj} = f_n\left(\log\left(e^{L_g + (j+\lambda_k)\Delta_g} - 1\right)\right)$ aus den Koeffizienten Z_n durch Bildung der Fourierreihe für $0 \leq k < n$ sowie $0 \leq j < M$.
Berechnung von $x_{kj} = \frac{e^{L_g + (j+\lambda_k)\Delta_g}}{(e^{L_g + (j+\lambda_k)\Delta_g} - 1)}$.
Berechnung von $g_{kj} = x_{kj} y_{kj}$ für $0 \leq k < n$ und $0 \leq j < M$.

Schritt 3: Berechne $ZG = U_0 \vec{g_j}$ mit den Relationen aus Satz 2.0.4 für $0 \leq j < M$.

Schritt 4: Bestimme $M_1 = \min\left\{2^p : 2^p > M + \frac{(U_{n-1} - L_{n-1})}{\Delta_g}\right\}$
$L_{f_{n-1}} = L_g + L_{n-1}$.
$U_{f_{n-1}} = L_{f_{n-1}} + M_1 \Delta_g$.

Schritt 5: Setze $ZG_{kj} = 0$ für $0 \leq k < n$ und $M + 1 \leq j < M_1$.

Schritt 6: Bestimme die Fouriertransformierte $\hat{g}$ der Funktion g durch Verwendung des verbesserten Algorithmus aus Abschnitt 5.4.1.

Schritt 7: Bestimme die Fouriertransformierte $\hat{f}_{R_{t_{n-1}}}$ der Funktion $f_{R_{t_{n-1}}}$ durch Verwendung des verbesserten Algorithmus aus Abschnitt 5.4.1.

Schritt 8: Berechne das Produkt $\hat{f}_{n-1}(s_{kj}) = \hat{f}_{R_{t_{n-1}}}(s_{kj}) \cdot \hat{g}(s_{kj})$, für $0 \leq k < n$ sowie $0 \leq j < M_1$.

Schritt 9: Berechne die Entwicklungskoeffizienten von f_{n-1} mittels des verbesserten Algorithmus aus Abschnitt 5.4.2.

Schritt 10: Berechne den numerischen Träger von f_{n-1} mit Algorithmus A.

Der Body des Algorithmus wird nun $n-1$ mal aufgerufen bis die gewünschte Dichte f_1 berechnet ist.

6.3.4 Integralberechnung

Für die Preisberechnung ist es erforderlich Integrale berechnen zu können. Die Berechnung erfolgt durch die Fourierkoeffizienten der Funktion.

In der weitern Folge wird dargestellt, wie aus den Entwicklungskoeffizienten der Dichte, die Entwicklungskoeffizienten des zugehörigen Integrals berechnet werden können.

Sei der Integraloperator I definiert durch

$$If(x) := \int_{-\infty}^{x} f(t)\, dt.$$

Für ein $f \in \pi_{n-1}$ kann dieser Operator durch $I_n = P_n(If)$, der Projektion auf π_{n-1} approximiert werden.

Für $0 \leq x < 1$ gilt

$$I_n f(x+j) = \sum_{k=0}^{n-1} \langle If(j+\cdot), \phi_k \rangle_{L^2[0,1]} \phi_k(x).$$

Für die Koeffizienten $\langle If(j+\cdot), \phi_k \rangle_{L^2[0,1]}$ dieser Reihendarstellung gilt

$$\begin{aligned}
\left\langle \int_{-\infty}^{j+x} f(u)\,du, \phi_k(x) \right\rangle_{L^2[0,1]} &= \int_0^1 \int_{-\infty}^{j+x} f(u)\,du\phi_k(x)\,dx \\
&\overset{x\geq u-j}{=} \int_{-\infty}^{j+1} f(u)\,du \int_{\max(0,u-j)}^{1} \phi_k(x)\,dx \\
&= \delta_{k,0} \int_{\infty}^{j} f(u)\,du + \int_j^{j+1} f(u) \int_{u-j}^1 \phi_k(x)\,dx\,du
\end{aligned} \tag{6.14}$$

Für $k \neq 0$ ist der erste Term auf der rechten Seite gleich 0. Deshalb wird zuerst dieser Fall betrachtet und damit nur mehr das folgende Integral

$$\begin{aligned}
\int_j^{j+1} f(u) \int_{u-j}^1 \phi_k(x)\,dx\,du &= \int_0^1 f(j+u) \int_u^1 \phi_k(x)\,dx\,du \\
&= -\int_0^1 f(j+u) \int_0^u \phi_k(x)\,dx\,du \\
&= -\int_0^1 \int_0^u \phi_k(x)\,dx f(j+u)\,du \\
&= -\left\langle \int_0^u \phi_k(x)\,dx, f_j(u) \right\rangle_{L^2[0,1]} \\
&\overset{f\in\pi_{n-1}}{=} -\sum_{0\leq l<n} \langle f_j, \phi_l \rangle_{L^2[0,1]} \left\langle \int_0^u \phi_k(x)\,dx, \phi_l(u) \right\rangle_{L^2[0,1]} \\
&= -\begin{bmatrix} \langle f_j, \phi_0 \rangle_{L^2[0,1]} \\ \vdots \\ \langle f_j, \phi_{n-1} \rangle_{L^2[0,1]} \end{bmatrix} M_n e_k
\end{aligned}$$

Die Einträge der Matrix M_n werden durch $\left(\left\langle \int_0^u \phi_k(x)\,dx, \phi_l(u) \right\rangle_{L^2[0,1]}\right)_{0\leq l,k<n}$ gebildet. Für $0 \leq u < 1$ folgt aus

$$\begin{aligned}
\left\langle \int_0^u \phi_k(x)\,dx, \phi_l(u) \right\rangle_{L^2[0,1]} &= \int_0^1 \int_0^u \phi_k(x)\,dx\phi_l(u)\,du \\
&\overset{\text{part.Int.}}{=} \int_0^u \phi_k(x)\,dxu\phi_l(u)|_0^1 - \int_0^1 \phi_k(u) \int_0^u \phi_l(x)\,dxdu \\
&= \int_0^1 \phi_k(x)\,dx\phi_l(1) - \int_0^1 \phi_k(u) \int_0^u \phi_l(x)\,dxdu
\end{aligned}$$

schiefsymmetrische Form von M_n, da der erste Term nur ungleich Null ist für $k = 0$.

Weiters gilt für $0 \leq u < 1$

$$\begin{aligned} u\phi_k(u) - (k+1)\int_0^u \phi_k(x)\,dx &= p \in \pi_k \\ \langle u\phi_k, \phi_{k+1}\rangle_{L^2[0,1]} - (k+1)\left\langle \int_0^u \phi_k(x)\,dx, \phi_{k+1}\right\rangle_{L^2[0,1]} &= 0 \\ \theta_k - (k+1)\left\langle \int_0^u \phi_k(x)\,dx, \phi_{k+1}\right\rangle_{L^2[0,1]} &= 0 \\ \left\langle \int_0^u \phi_k(x)\,dx, \phi_{k+1}\right\rangle_{L^2[0,1]} &= \frac{\theta_k}{(k+1)} \end{aligned}$$

mit θ_k wie in (2.4). Weiters gilt folgende Relation $\beta_k = -\theta_k/(k+1)$ mit β_k wie in (3.10). Somit entsprechen die Einträge in den Nebendiagonalen dieser Matrix in leicht modifizierter Form, jenen der Matrix $M_n(\nu)$ wie in (3.9), für $\nu = 0.5$.

In der Hauptdiagonale gilt

$$\left\langle \int_0^u \phi_k(x)\,dx, \phi_k\right\rangle_{L^2[0,1]} = 0 \quad \text{für} \qquad 0 < k < n.$$

Für $k = 0$ folgt

$$\left\langle \int_0^u \phi_0(x)\,dx, \phi_0\right\rangle_{L^2[0,1]} = \frac{1}{2}.$$

Somit gilt für die in (6.14) gefundene Darstellung

$$\begin{aligned} \overrightarrow{\langle If(j+\cdot), \phi\rangle}_{L^2[0,1]} &= \sum_{s=-\infty}^{j-1} \langle f(s+\cdot), \phi_0\rangle_{L^2[0,1]} e_0 + M_n(0.5)\overrightarrow{\langle f(j+\cdot), \phi\rangle}_{L^2[0,1]} \\ &\quad +0.5\langle f(j+\cdot), \phi_0\rangle_{L^2[0,1]} e_0, \end{aligned}$$

mit $e_0 = [1, 0, \ldots, 0]^t$ und $\phi = [\phi_0, \phi_1, \ldots, \phi_{n-1}]$.

Die Entwicklungskoeffizienten des Integraloperators können also ebenfalls durch Matrizendarstellungen beschrieben werden. Es gilt

$$\overrightarrow{\langle If(j+\cdot), \phi\rangle}_{L^2[0,1]} = M_n(\nu)\overrightarrow{\langle f(j+\cdot), \phi\rangle}_{L^2[0,1]} + A_0 e_0. \tag{6.15}$$

Wobei die Matrix $M_n(\nu)$, beschrieben in (3.9), für $\nu = 0.5$ berechnet wurde . Der zweite Teil dieser Summe hat die Form

$$A_0 = \sum_{s=-\infty}^{j-1} \langle f(s+\cdot), \phi_0\rangle_{L^2[0,1]} + 0.5\langle f(j+\cdot), \phi_0\rangle_{L^2[0,1]}. \tag{6.16}$$

Wird ein Trägerintervall $[L, U]$ verwendet, dass in M Teilintervalle der Größe $\Delta = \frac{U-L}{M} \neq 1$ geteilt wurde, gilt

$$\overrightarrow{\langle If(L+\Delta(j+\cdot)), \phi\rangle}_{L^2[0,1]} = \Delta M_n(\nu)\overrightarrow{\langle f(L+\Delta(j+\cdot)), \phi\rangle}_{L^2[0,1]} + \Delta A_0 e_0.$$

mit

$$A_0 = \sum_{s=-L}^{j-1} \langle f(s+\Delta\cdot), \phi_0 \rangle_{L^2[0,1]} + 0.5\langle f(L+\Delta(j+\cdot)), \phi_0 \rangle_{L^2[0,1]}.$$

Im Anschluß wird der Pseudocode zur Berechnung des Integrals dargestellt.

Algorithmus B

Input: Koeffizienten Z_{kj} der Dichtefunktion f, für $0 \leq k < n$ und $0 \leq j < M$.

Output: Entwicklungskoeffizienten des Integrals IZ_{kj} für $0 \leq k < n$ und $0 \leq j < M$.

Schritt 1: Berechne $IZ = \Delta M_n Z$.

Schritt 2: Setze $A_0 = 0.5Z_{1,1}$.

Schritt 3: Berechne für $0 \leq j < M-1$
$IZ_{0,j} = IZ_{0,j} + \Delta A_0$.
$A_0 = A_0 + 0.5Z_{0,j} + 0.5Z_{0,j+1}$.

Schritt 4: Setze $IZ_{0,M-1} = IZ_{0,M-1} + A_0$.

Mit diesen Entwicklungskoeffizienten können die Werte des Integrals an unterschiedlichen Stellen berechnet werden, durch Anwendung der Fourierreihenentwicklung.

Die Genauigkeit der Methode wird an zwei Beispielen demonstriert. Für unterschiedliche Dichtefunktion wurden zuerst die Entwicklungskoeffizienten berechnet und aus diesen Werten das zugehörige Integral gebildet. Die Vergleichsdaten wurden in Matlab mit der Routine 'cdf' für die entsprechenden Verteilungen berechnet.

In der Tabelle werden die jeweiligen Verteilungen angeführt, mit dem verwendeten Trägerintervall und der Faktor M, sowie der entstehende mittlere absolute Fehler.

[L,U]	M	Verteilung	MAE
$[-2,2]$	32	$\frac{1}{\sqrt{2\pi}0.2} e^{-\frac{1}{2}\left(\frac{x-0.1}{0.2}\right)^2}$	$2.89e^{-16}$
$[0,4]$	32	$\frac{2^2}{\Gamma(2)} x e^{-2x} I(0,\infty)$	$3.51e^{-16}$

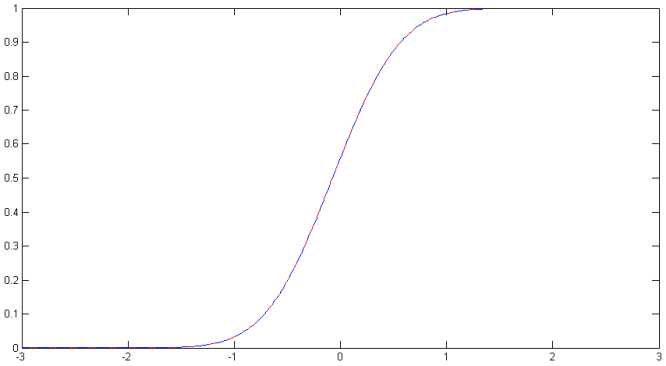

Abbildung 6.4: *In rot sind die mittels des beschriebenen Algorithmus berechneten Werte der Normalverteilung dargestellt und in blau jene Ergebnisse der Matlab Funktion cdf.*

6.3.5 Detaillierte Beschreibung des Algorithmus zur Optionspreisberechnung

Da nun Integrale mit der im Abschnitt zuvor beschriebenen Methode berechnet werden können, kann der gesamte Algorithmus zur Berechnung des Optionspreise beschrieben werden. Die Berechnungsvorschrift des Put-Optionspreises kann in (6.10) nachgelesen werden.

Zunächst wird noch eine zentralen Subroutinen beschrieben, welche zur Berechnung des Produktes der Funktionen f und g nötig ist.

Algorithmus C

Input: Koeffizienten Z_{kj} der Dichtefunktion f, für $0 \leq k < n$ und $0 \leq j < M$,das Trägerintervall von f, $[L_f, U_f]$, und eine weitere Funtkion g.

Output: Entwicklungskoeffizienten PZ_{kj} des Produktes der beiden Funtkionen f und g, für $0 \leq k < n$ und $0 \leq j < M$.

Schritt 1: Berechne $\vec{y}_j = U_0^{-1}\vec{Z}_j$ für $0 \leq j < M$. (siehe Satz 2.0.4).

Schritt 2: Berechne $g_{kj} = y_{kj}g(L_f + \Delta(j + \lambda_k))$ für $0 \leq k < n$ und $0 \leq j < M$.

Schritt 3: Berechne $PZ_j = U_0\vec{g}_j$, für $0 \leq j < M$. (siehe Satz 2.0.4).

Optionspreis

Schritt 1: Berechne die gesuchte Dichtefunktion f_1 mit dem in Abschnitt 6.3.1 beschriebenen Algorithmus.

Schritt 2: Berechne die Koeffizienten des Integrals von $f_1(t)$ mittels Algorithmus B und bilde die Fourierreihe um die gesuchten Integralwerte an den Stellen $\log(Kn/S_0)$ zu bekommen. Anschließend wird das Ergebnis mit Ke^{-rT} multipliziert.

Schritt 3: Berechne das Produkt von $g(t) = f(t)e^t$ mit Hilfe von Algorithmus C.

Schritt 4: Berechne wieder die Koeffizienten des Integrals von $g(t)$ mit Algorithmus B und bilde die Fourierreihe um die gesuchten Integralwerte an den Stellen $\log(Kn/S_0)$ zu berechnen. Das Ergebnis wird mit $(S_0/n)e^{-rT}$ multipliziert.

Schritt 5: Abschließend wird die Differenz zwischen dem Ergebnis aus Schritt 2 und 4 gebildet.

6.4 Bewertung von Asiatischen Optionen im Black-Scholes-Modell

6.4.1 Grundlagen der Preistheorie

In diesem Abschnitt werden die wichtigsten Werkzeuge und Sätze der Optionspreistheorie diskutiert. Als Referenz, siehe [2],[8] sowie [9].

Ein zeitstetiges Marktmodell mit endlichem Zeithorizont T, wird beschrieben durch

- einen Wahrscheinlichkeitsraum $(\Omega, \mathfrak{A}, \mathbb{P})$.
- $T \in [0, \infty)$, dem letzten im Modell berücksichtigten Handelszeitpunkt.
- $\mathcal{F} = (\mathcal{F}_t)_{t\in[0,T]}$, die den Informationsverlauf beschreibende Filtration.
- $S = (S_t^i)_{t\in[0,T]}$, dem stochastische Preiprozeß des Finanzgutes i. Es wird angenommen, dass S_t $\mathcal{F}_t - meßbar$ ist $\forall t$ (d.h. S ist an $\mathcal{F}$ adaptiert).
- $\tilde{S} = (\tilde{S}_t^i)_{t\in[0,T]}$ dem diskontierte Preisprozeß.

Definition 6.4.1. *Seien $\mathbb{P}$ und $\mathbb{Q}$ zwei Wahrscheinlichkeitsmaße aus $(\Omega, \mathfrak{A})$. Diese sind zueinander äquivalent, wenn $\mathbb{P}$ absolut stetig bezüglich $\mathbb{Q}$ ist und $\mathbb{Q}$ absolut stetig bezüglich $\mathbb{P}$ ist. Dass heißt die beiden Maße besitzen die gleichen Nullmengen.*

$$\mathbb{P}(A) = 0 \Leftrightarrow \mathbb{Q}(A) = 0, \qquad A \in \Omega.$$

Satz 6.4.1. *$\mathbb{P}$ ist genau dann absolut stetig bezüglich $\mathbb{Q}$, wenn es eine positive Zufallsvariable Z auf $(\Omega, \mathfrak{A})$ gibt, sodass*

$$P(A) = \int_A Z(\omega) dQ(\omega) \qquad \forall A \in \mathfrak{A}$$

gilt. Z wird als Dichte von $\mathbb{P}$ bezüglich $\mathbb{Q}$ bezeichnet und es gilt $Z = \frac{d\mathbb{Q}}{d\mathbb{P}}$.

Mit Hilfe der eben definierten äquivalenten Maße ist es möglich die Fundamentalsätze der Preistheorie zu beschreiben.

Satz 6.4.2. *Ein Marktmodell ist genau dann arbitragefrei, wenn ein zu $\mathbb{P}$ äquivalentes Wahrscheinlichkeitsmaß $\mathbb{Q}$ existiert, bzgl. dem der diskontierte Peisprozeß $\tilde{S}$ ein Martingal ist. $\mathbb{Q}$ heißt ein zu $\mathbb{P}$ äquivalentes Martingalmaß für $\tilde{S}$, bzw. risikoneutrales Maß.*

Satz 6.4.3. *Ein arbitragefreies Marktmodell ist genau dann vollständig, wenn das zu $\mathbb{P}$ äquivalente Matringalmaß für $\tilde{S}$ auf $\mathcal{F}$ eindeutig bestimmt ist.*

Die Beweise der Sätze 6.4.1, 6.4.2 sowie 6.4.3 können in [8] nachgelesen werden.

Die Existenz eines äquivalenten Martingalmaßes erlaubt die Bewertung von Optionen durch den Erwartungswert des diskontierten Payoffs der Option.

6.4.2 Das Black-Scholes-Modell

Eine detaillierte Beschreibung des Black-Scholes-Modell kann zum Beispiel in Irle [2] nachgelesen werden.

Definition 6.4.2. *Das Black-Scholes Modell ist ein zeitstetiges, arbitragefreies Marktmodell mit Zeihorizont T, wobei T das Endfälligkeitsdatum ist. In diesem Modell werden 2 Finanzgüter betrachtet.*
Das Erste Finazgut B_t, ist ein risikolos (Anleihe, Bond,etc.), mit stetiger konstanter Verzinsung und unterliegt dem Prozess

$$dB_t = rB_t dt.$$

Dabei ist $r > 0$ die risikolose Zinsrate. Das zweite Finanzgut ist risikobehaftet. Der Preisprozess dieses Finanzgutes zum Zeitpunkt t, ist durch die stochastische Differentialgleichung

$$dS_t = S_t(\mu dt + \sigma dW_t) \tag{6.17}$$

gegeben. Wobei $(W_t)_{0\leq t\leq T}$ eine standard Brown'sche Bewegung ist und μ die erwartete Returnrate und σ die Volatilität.

Die Lösung der Gleichung (6.17) *ist gegeben durch eine geometrische Brown'sche Bewegung und lautet*

$$S_t = S_0 e^{\left(\mu - \frac{\sigma^2}{2}\right)t + \sigma W_t}$$

mit dem Startpreis S_0.

Der Informationsverlauf durch eine Filtration $(\mathcal{F}_t)_{0\leq t\leq T}$ beschrieben, diese hat die Form $\mathcal{F}_t = \sigma\{(W_s|s \leq t) \cup \mathcal{N}\}$. $\mathcal{N}$ ist die Menge aller $\mathbb{P}$-Nullmengen.

Im weiteren gelten folgende Annahmen:

1. *ν und σ sind konstant.*
2. *Es gibt keine Transaktionskosten und Steuern.*
3. *Es gibt keine Dividenden während der Laufzeit.*
4. *Der risikolose Zinssatz ist konstant.*

Im Black-Scholes-Modell kann ein eindeutiges äquivalentes Martingalmaß Q konstruiert werden, somit ist das Modell vollständig.

In diesem speziellen Fall folgt der Preisprozess einer Log-Normal-Verteilung und somit ist der Logarithmus des Returns R_{t_i} im Intervall $[t_{i-1}, t_i]$ normalverteilt mit

$$\mu = \left(r - \frac{\sigma^2}{2}\right)(t_i - t_{i-1}), \qquad var = \sigma^2(t_i - t_{i-1}).$$

6.4.3 Testresultate im Black-Scholes-Modell

Zunächst wird die Optionspreisberechnung im Black-Scholes-Modell für sehr einfache Asiatische Optionen, mit nur einer Periode betrachtet. Diese Optionspreise entsprechen den Call- oder Put-Preisen einer europäischen Option.

Für folgende Parametersets wurden die Put-Preise bestimmt.

Nr.	r	σ	K	T	S_0
1.	0.05	0.5	2	1	2
2.	0.1	0.5	95	1	100
3.	0.03	0.2	105	2	100

In der nachfolgenden Tabelle sind zunächst die errechneten Put-Optionspreise der europäischen Option (Asiatische Option mit nur einer Periode) dargestellt. Als Vergleichswerte dienen die Optionspreise des europäischen Puts. Diese Werte wurden in Matlab mittels der Funktion $'[OptCall, OptPut] = blsprice(S_0, K, r, T, \sigma, q)'$ berechnet. Weiters wird der Fehler, also die Differenz, zwischen diesen beiden Größen angegeben.

Nr.	europ.Option (Algorithmus)	europ.Option	Fehler
1.	0.338310933258765	0.338310933258765	$4.44e^{-16}$
2.	12.149935115699728	12.149935115699716	$1.24e^{-14}$
3.	10.634970761170445	10.634970761170422	$2.30e^{-14}$

Der Algorithmus zur Berechnung Asiatischer Optionen mit mehreren Perioden wurde an folgendem Parameterset getestet (siehe [16] sowie [10]).

r	σ	K	T	S_0
0.05	0.5	2	1	2

Für dieses Parameterset wurden die Put-Preise berechnet und mittels der Call-Put-Parität die Preise der Call-Option. Die Periodenanzahl wurde in jedem Schritt verdoppelt. In der Tabelle wird weiters die Laufzeit des Algorithmus in Sekunden angegeben.

Perioden	$S_0(C)$	CPU(s)	
1	0.435852084257357	0.892	$1.78e^{-14}$
2	0.341915189968563	1.334	$1.35e^{-13}$
4	0.294343324407761	2.1528	$-1.98e^{-14}$
8	0.270431987681639	2.1528	$8.27e^{-14}$
16	0.258439135453751	5.2416	$4.87e^{-13}$
32	0.252431606650541	11.1850	$4.78e^{-13}$
64	0.249424750322772	22.6201	$4.78e^{-13}$
128	0.247920502995972	92.9297	$2.81e^{-12}$
256	0.247168168314599	189.1044	$5.81e^{-12}$
512	0.246791947405830	389.5812	$1.26e^{-11}$

Die letze Spalte gibt den Unterschied zwischen den hier berechneten Werten $S_0(C)$ und den in [10] angegebenen Preisen.

In der folgenden Graphik wurden die Call-Optionspreise für verschiedene Werte von K berechnet. Es wurde wieder das oben definierte Parameterset gewählt und die Anzahl der Perioden wurde auf 12 gesetzt, d.h. eine monatlich betrachtete Asiatische Option. Solche Berechnungen für unterschiedliche Werte von K, sind mit dem beschriebenen Algorithmus sehr schnell und unkompliziert durchzuführen. Da man als Output des Algorithmus die Entwicklungskoeffizienten der Preisfunktion bekommt, muss nur noch die Fourierreihe gebildet werden für die gewünschten Werte von K.

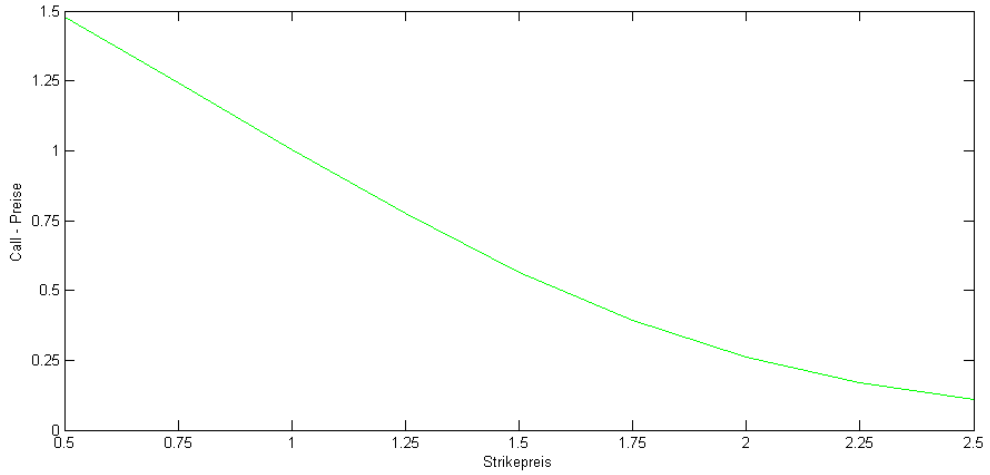

Abbildung 6.5: *Entwicklung des Call-Preises für unterschiedliche Werte des StrikePreises K.*

6.4.4 Kritik am Black-Scholes-Modell und weiterführende Überlegungen

Das Black-Scholes-Modell ist das wohl populärste Marktmodell, es beruht jedoch auf einigen urealistischen Annahmen.

- Es gibt keine Steuern und Transaktionskosten.
- Der Logarithmus des Returns basiert auf der Normalverteilung.
- Die Volatilität ist konstant.

Eine ausführlichere Diskussion dieser Punkte kann z.B. in [22] nachgelesen werden.

Eine allgemeinere Form um ein Marktmodell zu beschreiben bieten Lèvy-Prozesse. In den zugehörigen Lèvy-Modellen basiert der Logarithmus des Returns nicht mehr auf

der Normalverteilung, statt dessen werden Verteilungen mit „fetteren„ Tails verwendet. Die Returns sind aber immer noch unnabhängig. Wichtig ist zu bedenken das diese Modelltypen nicht mehr vollständig sind, also muss stets definiert werden welches äquivalente Martingalmaß verwendet wurde. Näheres zu Lèvy-Modellen bzw. der Berechnungsmöglichkeiten des äquivalenten Martingalmaßes kann in [19] nachgelesen werden.

Die Algorithmus bietet sich auch zur Berechnung von Optionspreisen in Lèvy-Modellen an, da als Voraussetzung an die Dichtefunktion nur die Unabhängigkeit der Returns gefordert wird, und dies erfüllt ist. Bei vielen der zugrundeliegenden Prozeße sind die Dichten nicht analytisch darstellbar, es ist jedoch immer möglich eine Darstellung der charakteristischen Funktion zu geben. Da diese Funktionen mit der Fouriertransformierten übereinstimmen können die Funktionswerte bzw. die Entwicklungskoeffizienten der Inversen durch Anwendung des Verfahrens von Peter den Iseger berechnet werden. Diese Werte können dann als Input für den hier beschriebenen Algorithmus verwendet werden.

Anhang: Matlab Code

In diesem Abschnitt kann ein Großteil des verwendeten Matlab Codes nachgelesen werden.

****

Die Funkion 'basicFun1' erzeugt die benötigten Matrizen, sowie Eigenwerte und Eigenvektoren.

****

```
function [lambda , eta , U0 ,U0inv , V01,V0inv1 ,U,V2,M1,N] = basicFun1(n,nu)

% ------------------------------------------------------------------------
% Nullvektor mit 1 an der ersten Stelle
% ------------------------------------------------------------------------
e0=zeros(1,n);
e0(1)=1;
% ------------------------------------------------------------------------
% Einsvektor
% ------------------------------------------------------------------------
one=ones(1,n);
% ------------------------------------------------------------------------
% Berechnung der n Nullstellen im Fourier Bereich (eta) der spez.
% definierten Polynome vom Grad n.  Diese sind durch die Eigenwerte
% der Tridiagonal-Matrix M gegeben. Gleichzeitig werden auch die
% dazugehörigen Eigenvektoren berechnet.
% ------------------------------------------------------------------------
clear y k;
y=zeros(1,n);
for k = 1:n
   y(k)=-1/(2*sqrt(4*k^2-1)); % Einträge in den Nebendiagonalen von M
end
clear M1 s;
M1=zeros(n);
M1(1,1)=0.5*cos(pi*nu)/sin(pi*nu); %  spez. Eintrag in M(1,1)
for s=1:n-1
```

```
        M1(s+1,s)=y(s);
        M1(s,s+1)=y(s);
end
%-------------------------------------------------------------------------
% Berechnung der Eigenwerte bzw. Eigenvektoren mit bereits impl. Matlab
% Funktion.
%-------------------------------------------------------------------------
[V2,D2]=eig(M1);
%-------------------------------------------------------------------------
% spez. Darstellung der Matrix V0
%-------------------------------------------------------------------------
clear V21 V0inv1 V01 ;
V21=V2(1,:)/(-i*(exp(i*pi*nu)-exp(-i*pi*nu))).';
V01=V2*diag(V21);
V0inv1=inv(V01);
%-------------------------------------------------------------------------
clear eta;
eta=D2*one.'; % Nullstellen bzw. Eigenwerte
%-------------------------------------------------------------------------
% Berechnung der n Nullstellen der Legendrepolynome (lambda)
% mit Hilfe der Matrix U. Die Nullstellen entsprechen wieder den
% Eigenwerten der Tridiagonalmatrix - Matrix. Gleichzeitig werden wieder
% die dazugehörigen Eigenvektoren berechnet.
%-------------------------------------------------------------------------
clear k N teta U B lambda s;
teta=zeros(1,n);
for k = 0:n-1 % Einträge in den Nebendiagonalen der Matrix N
  teta(k+1)= ((k+1)/(4*k+2))*sqrt((2*k+1)/(2*k+3));
end
%-------------------------------------------------------------------------
% Konstruktion der Matrix N
%-------------------------------------------------------------------------
N=0.5*eye(n);
for s=1:n-1
        N(s+1,s)=teta(s);
        N(s,s+1)=teta(s);
end
%-------------------------------------------------------------------------
% Eigenwerte und Eigenvektoren der Matrix U
%-------------------------------------------------------------------------
[U,B]=eig(N);
%-------------------------------------------------------------------------
% spez. Darstellung der Matrix U0
%-------------------------------------------------------------------------
clear U1 U0 U0inv ;
```

```
U1=U'*e0.';
U0=U*diag(U1);
U0inv=inv(diag(U1))*U'; % Inverse
%-------------------------------------------------------------------------
lambda = B*one.'; % Eigenwerte bzw. Nullstellen

*************************************************************************
```

Der Quellcode des verbesserten Algortihmus zur Berechnung der Inversen.

```
*************************************************************************
function [Z]= transformInversion11(m1,t,funname1,funname2,n,delta,t0,fourier)
begintime=cputime;

%-------------------------------------------------------------------------
% Verbesserter Algorithmus zur Berechnung der Inversen
%
%
% Input sind die Originalfunktion sowie die Fouriertransformierte.
% Der Shift-Operator t0, der Verkleinerungsfaktor delta und
% die Genauigkeit  der Gauß-Quadratur n, sowie die Anzahl der
% Intervalle m1.
%
% optional ist die Eingabe von fourier, einer Matrix von bereits berechneten
% Werten der Fouriertransformierten.
%
% 2009 Lehner Edith
%-------------------------------------------------------------------------
% input parameter
%-------------------------------------------------------------------------

nu = 0.5;
%m=m1;  (optional : ohne reelle Dämpfung)
m =8*m1;
c = 44/m;
%c=0; (optional: ohne reelle Dämpfung)

%-------------------------------------------------------------------------
[lambda, eta, U0 ,U0inv, V0 ,V0inv,U,V,M,N]=basicFun1(n,0.5);
%-------------------------------------------------------------------------
% Dämpfungsparameter
%-------------------------------------------------------------------------
alpha=zeros(1,m+1);
```

```
clear r;
for r=0:m
alpha(r+1)=-c*i-pi+2*pi*(r)/m;
end
%-------------------------------------------------------------------------
% Nullvektor mit 1 an der ersten Stelle
%-------------------------------------------------------------------------
e0=zeros(1,n);
e0(1)=1;
%-------------------------------------------------------------------------
% Einsvektor
%-------------------------------------------------------------------------
one=ones(1,n);
%-------------------------------------------------------------------------
clear Lf l r eta1 eta3;
Lf=zeros(n,m);
eta1=zeros(n,m+1);
%-------------------------------------------------------------------------
eta3=1./eta;  % Reziprokwerte der orig. Nullstellen
%-------------------------------------------------------------------------
for r=0:m
    eta1(:,r+1)=eta3 + alpha(r+1);
end
%-------------------------------------------------------------------------
% Auswerten der Fouriertransformierten
%-------------------------------------------------------------------------
if nargin < 8
for l=1:n
    for r=0:m-1
   Lf(l,r+1)=exp(t0*alpha(r+1)*i/delta)*exp(i*pi*t0/delta)*(eta3(l)/delta)
   *feval(funname1,eta1(l,r+1)/delta);
    end
end
%-------------------------------------------------------------------------
% Auswerten der Funktionswerte der Fouriertransformierten, aus der
% vorgegebenen Matix fourier
% (optional)
%-------------------------------------------------------------------------
else
Lf=fourier;
[n,m]=size(Lf);
m1=m/8;
%c=0;
%m1=m;
c=44/m;
```

```
alpha=zeros(1,m+1);
clear r;
for r=0:m
alpha(r+1)=c-i*pi+2*pi*i*(r)/m;
end
end
%-------------------------------------------------------------------------
% Anwendung der Gauß-Quadratur im Bildbereich der Fouriertransformierten
%-------------------------------------------------------------------------
clear G r G1;
G=zeros(n,m);

for r = 0:m-1
  G(:,r+1)=exp(pi*i*0.5)*V0*Lf(:,r+1);
end
for s=0:n-1
    G(s+1,:)=i^(-s)*G(s+1,:);
end
for r=0:m-1
    G(:,r+1)=U*diag(exp(alpha(r+1)*i*(lambda)))*U'*G(:,r+1);
end
%-------------------------------------------------------------------------
% IFFT
%-------------------------------------------------------------------------
clear s Y;
Y1=zeros(n,m);
Y1=ifft(G(:,1:m).');
Y1=Y1.';
%-------------------------------------------------------------------------
% IFFT - Inverse Fast(Finite)Fourier Transformation ohne Verwendung
% der bereits in Matlab implementierten ifft Funktion.
%-------------------------------------------------------------------------
% clear l s r Y1 G2;
%
% G2=zeros(n,m);
% Y1=zeros(n,m);

% %
% for l=0:m-1
%     for r=0:m-1
%         G2(:,r+1)=(G(:,r+1)/m)*exp(2*pi*r*i*l/m);
%         %G2(:,r+1)=(G(:,r+1)/m)*exp(2*pi*r*i*l/m);
%     end
%      Y1(:,l+1)=sum(G2,2);
% end
```

```
%-------------------------------------------------------------------------
% Berechnung der einfachen Expansionskoeffizienten, ohne relle Dämpfung.
%-------------------------------------------------------------------------
clear r s Z;
Z=zeros(n,m1);
for r=0:m1-1
    for s=0:n-1
    Z(s+1,r+1)=Y1(s+1,r+1)*exp(c*r);
    end
end
%-------------------------------------------------------------------------
% Fehlerabschätzung
%-------------------------------------------------------------------------
 clear r funwerte g;

[ywerte1]=expansion(m1,Z,t,n,t0,funname2,delta);

funwerte=feval(funname2,t);

error=sum(abs(funwerte-ywerte1),2)/m1

maxerror=max(abs(funwerte-ywerte1))

% Zeitnehmung
e=cputime-begintime

*************************************************************************
```

Anschließend der Code des verbesserten Algorithmus zur Berechnung der Fouriertransformierten

```
*************************************************************************

function [lap]=transform11neu(m1,funname1,funname2,n,delta,t0,L)
begintime=cputime;
%-------------------------------------------------------------------------
% Verbesserter Algorithmus zur Berechnung der Fouriertransformierten
%
%
% Input sind die Originalfunktion sowie die Fouriertransformierte.
% Weiters der Shift-Operator t0, der Verkleinerungsfaktor delta und
% die Genauigkeit  der Gauß-Quadratur n, sowie die Anzahl der
% Intervalle m1.
```

```
%
% optional ist die Eingabe von L, einer Matrix von bereits berechneten
% Koeffizienten der Originalfunktion.
%
% 2009 Lehner Edith
%
%-------------------------------------------------------------------------
% input parameter
%-------------------------------------------------------------------------
nu = 0.5;
m=8*m1;
%c=0;
c= 44/m;
%m=m1;
%-------------------------------------------------------------------------
[lambda, eta, U0 ,U0inv, V0 ,V0inv,U,V,M,N]=basicFun1(n,0.5);
%-------------------------------------------------------------------------
% Dämpfungsparameter
%-------------------------------------------------------------------------
alpha=zeros(1,m+1);
clear r;
for r=0:m
alpha(r+1)=-c*i-pi+2*pi*r/m;
end
%-------------------------------------------------------------------------
% Nullvektor mit 1 an der ersten Stelle
%-------------------------------------------------------------------------
e0=zeros(1,n);
e0(1)=1;
%-------------------------------------------------------------------------
% Einsvektor
%-------------------------------------------------------------------------
one=ones(1,n);
%-------------------------------------------------------------------------
% Berechung der Funktionswerte an den verschobenen Nullstellen.
% optional
%-------------------------------------------------------------------------
clear r s fun;
fun=zeros(n,m1);
if nargin < 7
for r=0:m1-1
        fun(:,r+1)=feval(funname2,((lambda+r)*delta+t0));
end
%-------------------------------------------------------------------------
% Berechung der Entwicklungskoeffizienten mit Hilfe der Matrix U0.
```

```
%-------------------------------------------------------------------------
clear r s koeff Z;
koeff=zeros(n,m1);
for r=0:m1-1
    koeff(:,r+1)=U0*fun(:,r+1);
end
%-------------------------------------------------------------------------
% Berechnung der Entwicklungskoeffizienten aus bereits vorhandenen
% Koeffizienten (optional)
%-------------------------------------------------------------------------
else
    koeff=L;
    [n,m1]=size(koeff);
m=8*m1;
%m=m1;
%c=0;
c=44/m;
clear r;
%-------------------------------------------------------------------------
for r=0:m
alpha(r+1)=-c*i-pi+2*pi*r/m;
end
end
%-------------------------------------------------------------------------
clear l g;
clear r s Z;
Z=zeros(n,m);
for r=0:m1-1
    Z(:,r+1)=koeff(:,r+1)*exp(-c*r); % reelle Dämpfung
end
%-------------------------------------------------------------------------
% Berechnung der IFFT mit der in Matlab implementierten Funktion
%-------------------------------------------------------------------------
 clear s Y
%Y=zeros(m,n);
Y=fft(Z.');
Y1=Y.';
%-------------------------------------------------------------------------
% Berechnung der FFT ohne Verwendung der bereits in Matlab implementierten
% Funktion.
%-------------------------------------------------------------------------
% for s=0:n-1
%     Y1(s+1,:)=Y1(s+1,:)*exp(-pi*i*0.5)*i^s;
% end
% %clear l s r Y1 G2;
```

```
%
%  G2=zeros(n,m);
%  Y1=zeros(n,m);
%
%  for l=0:m-1
%      for r=0:m-1
%          G2(:,r+1)=Z(:,r+1)*exp(-2*pi*r*i*l/m);
%      end
%      Y1(:,l+1)=sum(G2,2);
%  end
%-------------------------------------------------------------------------
% Berechnung der Werte der Fourier Transformierten aus den Koeffizienten
% im Fourier-Bereich. ( 1/s*fourier(1/s))
%-------------------------------------------------------------------------
clear r lap;
 for s=0:n-1
     V0inv(:,s+1)=V0inv(:,s+1)*i^(s);
end
lap=zeros(n,m);
for r=0:m-1
    lap(:,r+1)=(-i)*V0inv*U*diag(exp(-alpha(r+1)*i*(lambda)))*U'*Y1(:,r+1);
end
%-------------------------------------------------------------------------
% Berechnung der Werte der Fourier Transformierten aus den Koeffizienten
% im Fourier- Bereich. (fourier(1/s))
%-------------------------------------------------------------------------
for r=0:m-1
for s=1:n
    lap2(s,r+1)=lap(s,r+1)*eta(s)*delta;
end
end
%-------------------------------------------------------------------------
clear Lf l r eta1 eta2 eta3;
Lf=zeros(n,m);
eta1=zeros(n,m);
%-------------------------------------------------------------------------
eta3=1./eta;  % Reziprokwerte der orig. Nullstellen
%-------------------------------------------------------------------------
for r=0:m-1
    eta1(:,r+1)=eta3 + alpha(r+1); % Eta's um den Dämpfungsfaktor verschoben
end
%-------------------------------------------------------------------------
% Auswerten der Fouriertransformierten
%-------------------------------------------------------------------------
for l=1:n
```

```
    for r=0:m-1
        Lf(l,r+1)=exp(t0*alpha(r+1)*i/delta)*exp(t0*i*pi/delta)
        *feval(funname1,eta1(l,r+1)/delta);
    end
end
%-------------------------------------------------------------------------
% Fehlerberechnung
%-------------------------------------------------------------------------
clear res;
%%
 res=abs(-Lf(1,:)+lap2(1,:));
%
 error=sum(res,2)/m
 maxerror=max(res)
  % Zeitnehmung
e=cputime-begintime

*************************************************************************
```

Hier folgen nun die Algorithmen zur Berechnung des Integrals sowie des Produktes.

```
*************************************************************************
function [Int]=Integrate1(coefficients, delta)

%-------------------------------------------------------------------------
% Berechnung des Integrals
%
% Input sind die Entwicklungskoeffizienten des Integranden sowie der
% Verkleinerungsfaktor delta.
%
% Output sind die Entwicklungskoeffizienten des Integrals
%
% 2009 - Lehner Edith
%-------------------------------------------------------------------------

[n,m1]=size(coefficients);

[lambda, eta, U0 ,U0inv, V0 ,V0inv,U,V,M,N] = basicFun1(n,0.5);

Int=delta*M*coefficients;
clear r

 A0=0.5*coefficients(1,1);
```

```
for r=0:m1-2
    Int(1,r+1)=Int(1,r+1)+delta*A0;
    A0=A0+0.5*coefficients(1,r+1)+0.5*coefficients(1,r+2);
end

Int(1,m1)=Int(1,m1)+delta*A0;

*************************************************************************
*************************************************************************

function [Koeffneu]= ProductFunction(koeff,func,Lx,delta)

%-------------------------------------------------------------------------
% Berechnung des Produktes
%
% Input sind die Entwicklungskoeffizienten der ersten Funtkion sowie der
% Verkleinerungsfaktor delta, die untere Schranke des Trägers und die
% zweite Funtkion.
%
% Output sind die Entwicklungskoeffizienten des Produktes
%
% 2009 - Lehner Edith
%-------------------------------------------------------------------------

[n,m]=size(koeff);
%-------------------------------------------------------------------------
[lambda, eta, U0 ,U0inv, V0 ,V0inv,U,V,M] = basicFun1(n);
%-------------------------------------------------------------------------
% Berechnung der Entwicklungskoeffizienten
yvalues=U0inv*koeff;

%-------------------------------------------------------------------------

% Auswerten der Funktion
for r=0:m-1
funcv(:,r+1)=feval(func,(Lx+(lambda+r)*delta));
end
%-------------------------------------------------------------------------
clear r s;

% Berechnung des Produktes
for r=0:m-1
    for s=1:n
zvalues(s,r+1)=yvalues(s,r+1)*funcv(s,r+1);
    end
```

```
end
% Berechnung der neuen Entwicklungskoeffizienten
Koeffneu=U0*zvalues;

*************************************************************************
```

Anschließend wird der Algorithmus zur Berechnung des Optionspreises dargestellt, gefolgt von seinen Subroutinen.

```
*************************************************************************
function [PriceP,PriceC,timeend]= PriceAlgorithm(m,n,Lowerf,Upperf,p,funname1,
Lowerx,Upperx,n1,r,K,S0,T)
%-------------------------------------------------------------------------
% Berechnung des Optionspreises einer Asiatischen Option.
%
% Input:
%  m ... Anzahl der Intervalle
%  n ... Tiefe der Gauß-Quadratur
%  Lowerf,Upperf,Lowerx, Upperx ... Träger der Dichte sowie des Returns
%  p ... 1
%  n1 ... Perioden
%  r ... Zinsrate
%  K ... Strikepreis
%  T ... Fälligkeit
%  S0 ... Startpreis
%
% Output sind die Preise der Asiatischen Call- sowie Put-Option.
%
% 2009 - Lehner Edith
%-------------------------------------------------------------------------
timestart=cputime;
[coeffs,t,delta,L1,U1]= coefficient(m,n,Lowerf,Upperf,p,funname1);

[Z,deltag,Lfneu,Ufneu,Z12] =getDensity(m,L1,U1,p,funname1,Lowerx,
Upperx,coeffs,delta);

[Z1,Lfneu1,Ufneu1]=recDestiny(2,n1,funname1,p,Z,Lfneu,Ufneu,Lowerx,
Upperx,m,deltag);

[PriceP,PriceC]=getPrice(Lfneu1,Ufneu1,Z1,'func',n1,r,K,S0,T);

timeend=cputime-timestart;

*************************************************************************
```

```
*************************************************************************

function [coeffs,t,delta,L1,U1]= coefficient(m,n,Lowerx,Upperx,p,funname)
%-------------------------------------------------------------------------
% Berechnung der Entwicklungskoeffizienten einer Funtkion
%
% Input:
%  m ... Anzahl der Intervalle
%  n ... Tiefe der Gauß-Quadratur
%  Lowerx, Upperx ... Träger der Funktion
%  p ... 1
%  funname ... analytische Darstellung der Funktion
%
% Output sind die Entwicklungskoeffizienten der Funktion
%
% 2009 - Lehner Edith
%-------------------------------------------------------------------------
[lambda , eta, U0 ,U0inv, V0 ,V0inv,U,V,M,N] = basicFun1(n,0.5);

L1=Lowerx-log(p);
U1=Upperx-log(p);

delta=(U1 - L1)/m;

fun=zeros(n,m);

clear r t;
t=zeros(n,m);
for r=0:m-1
    t(:,r+1)=(lambda+r)*delta+L1+log(p);
end
clear r;
for r=0:m-1
    fun(:,r+1)=feval(funname,t(:,r+1));
end
coeffs=zeros(n,m);
clear r;
for r=0:m-1
    coeffs(:,r+1)=U0*fun(:,r+1);
end

*************************************************************************
*************************************************************************

function [Z,deltag,Lfneu,Ufneu,m3] =getDensity(m1,Lowerf,Upperf,p,funname,
Lowerx,Upperx,coeffs,deltaf)
```

```
%-------------------------------------------------------------------------
% Berechnung der Dichtefunktion
%
% Input:
%  m1 ... Anzahl der Intervalle
%  Lowerf,Upperf,Lowerx, Upperx ... Träger der Dichte sowie des Returns
%  p ... 1
%  funname ... analytische Darstellung des Returns
%  coeffs ... Entwicklungskoeffizienten der Dichte 1
%  deltaf ... Verkleinerungsfaktor
%
% Output sind die Entwicklungskoeffizienten der  Dichte 2, sowie der neuen
% Träger.
%
% 2009 - Lehner Edith
%-------------------------------------------------------------------------
[n,m2]=size(coeffs);
%-------------------------------------------------------------------------
[lambda , eta, U0 ,U0inv, V0 ,V0inv,U,V,M] = basicFun1(n,0.5);
%-------------------------------------------------------------------------
Lg=log(exp(Lowerf)+p);
Ug=log(exp(Upperf)+p);
%-------------------------------------------------------------------------
[tg,deltag,xvalues] =getDensity(m1,n,Lowerf,Upperf,p);
%-------------------------------------------------------------------------
[ywerte]=expansion(m1,coeffs,tg,n,Lowerf,funname,deltaf);
%-------------------------------------------------------------------------
clear r s;
zvalues=zeros(n,m1);
for r=0:m1-1
    for   s=1:n
        zvalues(s,r+1)=ywerte(s,r+1)*xvalues(s,r+1);
    end
end
%-------------------------------------------------------------------------
clear r;
for r=0:m1-1
     Pg(:,r+1)=U0*zvalues(:,r+1);
end
%-------------------------------------------------------------------------
mneu=(Upperx-Lowerx)/deltag;
m2=mneu+m1;
p1=log(m2)/log(2);
p2=min(ceil(p1));
m3=2^p2;
```

```
% ------------------------------------------------------------------------
Lfneu=Lowerx+Lg;
Ufneu=Lfneu+m3*deltag;
% ------------------------------------------------------------------------
Pg1=zeros(n,m3);
for r=0:m1-1
    Pg1(:,r+1)=Pg(:,r+1);
end
% ------------------------------------------------------------------------
[lap]=transform11neu(m1,'fun2','fun21',n,deltag,Lg,Pg1);
[lap1]=transform11neu(m3,'fun2',funname,n,deltag,Lowerx);
[n,m4]=size(lap);
% ------------------------------------------------------------------------
clear s r;
for r=0:m4-1
     for s=1:n
          trans(s,r+1)=lap(s,r+1)*lap1(s,r+1);
     end
end
for s=1:n
trans1(s,:)=trans(s,:)/(deltag*eta(s))      ;
end
% ------------------------------------------------------------------------
[Z]= transformInversion11(m1,1,'fun2',funname,n,deltag,Lfneu,trans1);

**************************************************************************
**************************************************************************

function [Z,Lfneu,Ufneu]=recDestiny(s1,s,funname,p,Z,Lf,Uf,Lx,Ux,m)
% ------------------------------------------------------------------------
% Rekursion zur Berechnung der Dichte .
%
% Input:
%  m ... Anzahl der Intervalle
%  s1 ... Startwert der Rekursion
%  s ... Ende der Rekursion
%  Lowerf,Upperf,Lowerx, Upperx ... Träger der Dichte sowie des Returns
%  p ... 1
%  Z ... Koeffizienten der Dichtefunktion
%
% Output sind Entwicklungskoeffizienten der Dichtefuntkion am Ende
% der gewünschten Rekursionstiefe.
%
% 2009 - Lehner Edith
% ------------------------------------------------------------------------
```

```
[n,m1]=size(Z);
[L1,U1,r,k,delta2]=getborders(m1,Lf,Uf,Z);
[Z,deltag,Lfneu,Ufneu,m3] =getDensity5(m,L1,U1,p,funname,Lx,Ux,Z(:,r+1:k+1),delta2);
s1=s1+1;
if s1 < s
    [Z,Lfneu,Ufneu]=recDestiny(s1,s,funname,p,Z,Lfneu,Ufneu,Lx,Ux,m,deltag);
end

************************************************************************
************************************************************************

function [PriceP,PriceC]=getPrice(Lf,Uf,Z,func,n1,r,K,S0,T)
% ---------------------------------------------------------------------
% Berechnung des Optionspreises einer Asiatischen Option.
%
% Input:
%  Z ... Koeffizienten der Dichtefunktion
%  Lf,Uf,Lowerx, Upperx ... Träger der Dichte sowie des Returns
%  func ... die Funktin exp(x)
%  n1 ... Perioden
%  r ... Zinsrate
%  K ... Strikepreis
%  T ... Fälligkeit
%  S0 ... Startpreis
%
% Output sind die Preise der Asiatischen Call- sowie Put-Option.
%
% 2009 - Lehner Edith
% ---------------------------------------------------------------------
[n,m]=size(Z);
delta=(Uf-Lf)/m;
Int=Integrate(Z,delta);
[Koeffneu]= ProductFunction(Z,func,Lf,delta);
Int1=Integrate(Koeffneu,delta);

[ywerte1]=expansion(32,Int,log(K*n1/S0),n,Lf,'funx1',delta);
[ywerte]=expansion(32,Int1,log(K*n1/S0),n,Lf,'funx1',delta);

x=1:n1;
for s=1:n1
    x1(s)=exp(r*x(s)/n1);
end
y=exp(-r)*S0*sum(x1,2)/n1;
PriceP=exp(-r*T)*(ywerte1*K-ywerte*S0/n1);
PriceC=exp(-r*T)*(ywerte1*K-ywerte*S0/n1)+ y-K*exp(-r*T);
```

Literaturverzeichnis

[1] L.Clewlow A.Carverhill. Flexible convolution. *Risk 3*, pages 25–29, 1990.

[2] A.Irle. *Finanzmathematik:Die Bewertung von Derivaten*. Teubner Verlag, 2003.

[3] Zimmermann Georg Benedetto John. Sampling multipliers and the poisson summation formula. *The Journal of Fourier Analysis and Applications*, 3(5):505–523, 1997.

[4] Eric Benhamou. Fast fourier transform for discrete asian options. *Journal of Computational Finance*, 6(1):349–375, 2002.

[5] John B. Conway. *A course in Functional Analysis*. Springer-Verlag New York Inc, 2000.

[6] Peter den Iseger. Numerical transform inversion using gaussian quadrature. *Probability in the Engineering and Informational Sciences*, 20:1–44, 2006.

[7] Peter den Iseger. Laplace transform inversion on the entire line. *SSRN*, pages 1–24, 2008.

[8] B.Lapeyre D.Lamberton. *Introduction to Stochastic Calculus Applied to Finance*. Chapman & Hall /CRC, 1996.

[9] M.Rockinger E.Jondeau, S.Poon. *Financial Modeling Under Non-Gaussian Distributions*. Springer Finance, 2007.

[10] Peter den Iseger Emöke Oldenkamp. Pricing guaranteed return rate products and discretely sampled asian options. *SSRN*, 9(3):1–39, 2006.

[11] Walter Gautschi. *Orthogonal Polynomials, Computation and Approximation*. Oxford University Press Inc. New York, 2004.

[12] Ranjan Roy George E. Andrews, Richard Askey. *Special Functions*. Cambridge University Press, Cambridge, 2000.

[13] A. Meucci G.Fusai. Pricing discretley monitored asian options under lévy processes. *Journal of Banking & Finance 32*, pages 2076–2088, 2008.

[14] M. Yor H. Geman. Bessel process, asian options and perpetuities. *Mathematical Finance*, pages 349–375, 1993.

[15] Kenneth B. Howell. *Priciples of Fourier Analysis*. Chapman & Hall /CRC, 2001.

[16] V. Linetsky. Spectral expansion for asian (average price) options. *Operations Research 52*, pages 856–857, 2004.

[17] Stéphane Mallat. *A wavelet tour of signal processing*. Academic Press Limited, 1998.

[18] T.Erdélyi P. Borwein. *Polynomials and Polynomial Inequalities*. Springer Verlag, 1995.

[19] Antonis Papapantoleon. An introduction to lèvy processes with applications in finance. *http://www.fam.tuwien.ac.at/ papapan/papers/*, pages 1–35, 2005.

[20] Walter Rudin. *Reelle und Komplexe Analysis*. Oldenbourg, 1999.

[21] Klaus Sandmann. *Einführung in die Stochastik der Finanzmärkte*. Springer-Verlag Berlin Heidelberg, 2001.

[22] Wim Schoutens. *Lévy Processes in Finance*. Wiley series in probability and statistics, 2003.

[23] Gabor Szegö. *Orthogonal Polynomials*. American Mathematical Society, Providence, Rhode Island, third edition, 1967.

[24] D. Werner. *Funktionalanalysis*. Springer Verlag, 2007.

[25] R. Hoppe W.Freund. *Stoer/Bulirsch: Numerische Mathematik 1*. Springer-Verlag Berlin Heidelberg, 2005.

[26] Peter G. Zhang. *Exotic Options 2nd Edition*. World Scientific, 2001.

[27] A. Zygmund. *Trigonometric Series Vol. I & II*. Cambridge University Press, Cambridge, 2002.