Dokumentation zur Erstellung einer Wavelet-Datenbank und deren Website

Institut für Mathematik
der Johannes-Gutenberg Universität Mainz
Stand: 29. Oktober 2017

In den Jahren 2016-2017 arbeiteten Andreas Andrzejczak und Simon Thomas unter der Leitung von Prof. Thorsten Raasch an dem Beginn einer Datenbank und einer damit verbundenen Website, die sämtliche Informationen über Wavelets enthalten soll. Nachdem die beiden Studenten das Projekt abgeben müssen und Hr. Raasch bedauerlicherweise die Universität wechselt, steht die Zukunft der Idee in den Sternen.
Diese Dokumentation soll zumindest theoretisch eine Chance zur Weiterarbeit an dem Projekt mit den bisher erzielten Ergebnissen geben.
Wir möchten die Gelegenheit außerdem nutzen, um uns bei Herrn Raasch für die das spannende Projekt und die angenehme Zusammenarbeit bedanken.

Andreas Andrzejczak und Simon Thomas, Oktober 2017.

Inhaltsverzeichnis

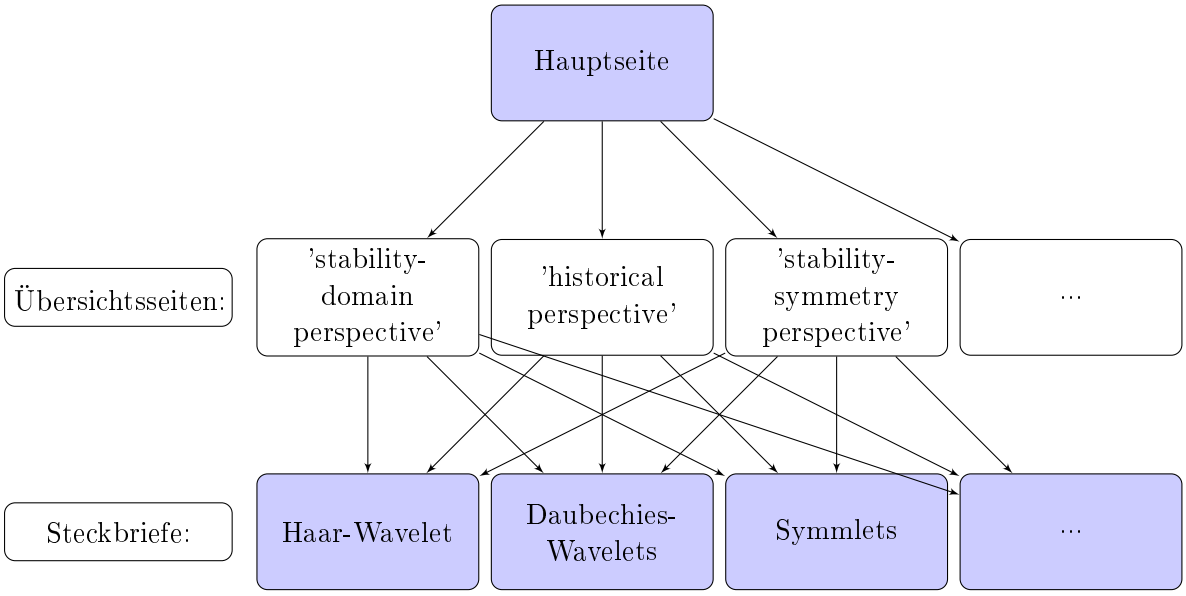
1 Aufbau des Dateisystems

Die Website selbst soll mit Wordpress gestaltet werden. Alle Dateien die in Wordpress zusätzlich benötigt werden sind in dem Ordner 'for_wordpress_oews' gespeichert. Diese werden an den entsprechenden Stellen über einen absoluten Pfad in Wordpress geladen. Alle Dateien befinden sich auf unseren lokalen Computern, auf github.org, sowie auf der mitgelieferten CD.

```
/
├── for_wordpress_oews
│   ├── html_testfiles_wp ..... Test-HTML-Seiten um die Fuktionen zu testen
│   │   └── wavelet-systems.sqlite ..... Datenbank mit allen Infos
│   │                                    über die Wavelet Systeme
│   ├── js ..... js-Dateien
│   │   ├── ext_libs ..... externe js-Bibliotheken
│   │   ├── libs ..... Selbst geschriebene js-Dateien
│   │   │   └── functionTypes ..... Plot-Routinen (Je eine Datei pro Wavelet-Typ)
│   │   └── sites ..... js-Dateien, die zum Aufbau der grafischen Benutzeroberfläche
│   │                   einer jeweiligen Website benötigt werden
│   └── pictures ..... Bilder der Website
└── documentation ..... Diese Dokumentation sowie zugehörige latex Bestandteile
```

2 Aufbau der Website

Die Website selbst ist folgendermaßen aufgebaut:



Öffnet man die Website, so gelangt man zunächst auf eine Hauptseite mit grundsätzlichen Informationen. Von dieser gelangt man auf verschieden Übersichts-Seiten, wo die Wavelet-Familien nach ihrer Geschichte oder ihren Eigenschaften sortiert sind. Von dort aus gelangt man zu den jeweiligen Steckbriefen eines Wavelets (bzw. einer Wavelet-Familie).

Zusätzlich ist eine Navigation zwischen den einzelnen Seiten über eine Menüleiste möglich.

2.1 Integration von Javascript in Wordpress

Um Javascript in Wordpress zu integrieren wurde möglichst viel Code in externe Dateien geschrieben. Diese liegen neben den Wordpress-Dateien auf dem Server.
Nichtsdestotrotz benötigt man Java-Script in HTML-Dateien von Wordpress um die js-Dateien zu integrieren und die benötigten Routinen aufzurufen. Dazu wurde in Wordpress das Tool 'Script n Styles' installiert.
Damit erscheint unter der HTML-Eingabe einer jeden Seite in Wordpress ein weiterer Abschnitt 'Script n Styles' in dem js-Code geschrieben werden kann. (Ggf. muss das Anzeigen des Abschnitts unter 'Screen Options' aktiviert werden.)
Hier können wir den Headteil der HTML-Seite nutzen, um externe Dateien für diese Seite zu laden.
In den Bodyteil schreiben wir den js-Code, der notwendig ist um beispielsweise select-Menüs aufzubauen, plot-Routinen oder andere Funktionen aufzurufen. Der Anteil des js-Codes im Bodyteil soll der Übersicht halber möglichst gering gehalten werden. Im Moment entspricht der js-Code in diesem Teil genau dem, der in den HTML-Testfiles (for_wordpress_oews → html_testfiles_wp) zu finden ist. So können Änderungen von js-Code in den Bibliotheken oder HTML-Dateien getestet werden, ohne jedes Mal neue Dateien auf den Server laden zu müssen.

3 Aufbau der Datenbank

Die Datenbank dient als Informationsspeicher der Website und beinhaltet die wichtigsten Informationen aller Wavelets bzw. Skalierungsfunktionen. Sie ist in sqlite implementiert. Ein Programm um sqlite-Dateien zu öffnen und zu bearbeiten ist beispielsweise der 'SQLite Manager' von Firefox.

Die Datenbank besitzt eine Haupttabelle 'TYPES', sowie für jeden Typ, sprich jede Wavelet-Familie, eine Untertabelle.
Die Haupttabelle dient dazu die 'Übersichtsseiten' aufzubauen.
Die Untertabellen werden genutzt, um die Steckbriefe aufzubauen.

Bei der Namensgebung verwenden wir folgende Konventionen:

- Tabellennamen werden ausschließlich in Großbuchstaben geschrieben.
- Spaltennamen werden ausschließlich in Kleinbuchstaben geschrieben.
- Zahlen, Listen von Zahlen oder anderen Informationen werden im JSON-Format gespeichert. (So ist das Parsen im js-Code einfach und schnell.)

Um die Datenbank in js zu laden wird das package sql.js (https://github.com/kripken/sql.js/) genutzt.
Die Haupttabelle 'TYPES' enthält folgende Spalten:

- id - (Unsere) Identifikationsnummer des Wavelet-Typs.
- name - Name des Wavelettyps. (Meist der Name des Entdeckers.)
- year - Jahr des ersten Erwähnens in der Literatur.
- reference - Titel der Hauptreferenz.
- doi - DOI der Hauptreferenz.
- symmetry - Symmetrie des Wavelet-Typs. Erlaubte Werte bis jetzt: yes, nearly, none.
- stability - Art der erzeugten Basis. Erlaubte Werte bis jetzt: ONB, Riez, Frame.
- table_with_functions - Name der Tabelle mit den Details zu den einzelnen Wavelets dieses Typs.
- logo - URL zum Logo, damit es bei dem Aufbau der Übersichtsseiten geladen werden kann.
- description- kurze Beschreibung zum Wavelet-Typ. (Erscheint auf den Übersichtsseiten als Kurzbeschreibung.)
- url - Link zum Steckbrief des jeweiligen Wavelet-Typs.
- abbr - Abkürzung des Wavelet-Typs.
- domain - 'Domain' des Wavelet-Typs. (Erlaubte Werte bis jetzt: 'ONB', 'Riez', 'Frame'.)

Die anderen Tabellen sind je nach Wavelet-Typ verschieden aufgebaut. Die Spaltennamen sind meist selbsterklärend. Erwähnenswerte Ausnahmen sind:

- critical_sobolev_exponent - hier wird der kritische Sobolev-Exponent abgespeichert. Dieser wird von dem jeweiligen Plot-Objekt abgefragt, um die Stetigkeit zu garantieren. Bis jetzt sind folgende Werte erlaubt: numerische Werte größer Null (exakter Sobolev-Exponent), der String '>0.5' (für den Fall, dass der exakte Wert unbekannt ist, aber der Wert größer 0.5 und die Funktion damit zumindest stetig), ein leeres Feld oder der String 'null' (für den Fall, dass der Wert völlig unbekannt ist) und der String 'x' (für den Fall, dass der Sobolev-Exponent unbekannt ist, aber die Funktion sicher nicht stetig.)

  Im Falle der Primbs-Wavelets werden zwei Sobolev-Exponenten in einem Array gespeichert. Der erste steht für die primale, der zweite für die duale Skalierungsfunktion.
- parameters (nur in der Primbs Tabelle) - Enthält zusätzliche, für das Plotten notwendige Parameter. Im Falle von Primbs z.B. ["d",2,"d_tilde",2].
- id_of_dual_function - enthält ein Integer oder ein Array mit ganzen Zahlen, die die Id(s) der dualen Funktion(en) speichern.

Weitere Funktionen können jederzeit in die Tabellen eingefügt werden. Sie erscheinen direkt auf der Website.
Falls ein neuer Typ zu in die Tabelle 'TYPES' eingefügt wird, so erscheint dieser auch direkt mit in den Übersichtsseiten.
Um auch einen Steckbrief zu einem neuen Typ zu erhalten, muss dieser in Wordpress neu angelegt werden und eine neue Tabelle zu diesem Typ in die Datenbank mit aufgenommen werden.

4 Bisherige Wavelettypen

Zu den folgenden Wavelettypen existieren zum jetzigen Zeitpunkt Steckbriefseiten, auf denen der Plot der Wavelets und der Skalierungsfunktionen, sowie teilweise noch weitere Informationen zu sehen sind.

4.1 Daubechies-Wavelets

Die berühmtesten Wavelets sind vermutlich die Daubechies-Wavelets. Sie besitzen die interessante Eigenschaft, dass sie theoretisch mit beliebig hoher Regularität konstruiert werden können. Die ursprüngliche Konstruktion findet sich in [Dau88]. Die Sobolev-Exponenten, sowie die Masken findet man in [Dau92]. Wie ein Plot mithilfe der Punktauswertung durchgeführt wir steht in Abschnitt 5.1.

4.2 Coiflets

R. Coifman schlug erstmals vor, Wavelets mit möglichst vielen verschwindenden Momenten zu konstruieren. Diese wurden erstmals von Daubechies ([Dau93], [Dau92]) konstruiert und Coiflets genannt. Genauer heißt ein orthonormales Wavelet ψ mit kompaktem Träger Coiflet der Ordnung N, falls gilt:

$$i) \int_{-\infty}^{\infty} x^n \psi(x) dx = 0 \text{ für alle } n = 1...N-1, \tag{4.1}$$

$$ii) \int_{-\infty}^{\infty} x^n \phi(x) dx = \delta_n \text{ für alle } n = 1...N-1, \tag{4.2}$$

Die Verfeinerungskoeffizienten in der Datenbank, sowie die Sobolevexponenten stammen aus [Cer08].
Geplottet werden können die Coiflets genauso wie die Daubechies-Wavelets.

4.3 Symmlets

In [Dau93] wird zunächst gezeigt, dass es außer das Haar-Wavelet kein weiteres orthonormales Wavelet gibt, dass eine Symmetrie besitzt. Es gibt jedoch Möglichkeiten 'symmetrischere' Wavelets zu konstruieren. Möglichst symmetrische Wavelets werden beispielsweise in [Dau92] konstruiert und Symmlets genannt. Die Koeffizienten der Symmlets stammen aus [Dau92]

Auch die Symmlets können analog zu den Daubechies-Wavelets geplottet werden.

4.4 Cohen-Daubechies-Feauveau-Wavelets

Die Cohen-Daubechies-Feauveau Wavelets sind die ältesten sogenannten biorthogonalen Wavelets. Deren erste Konstruktion stammt aus [AC92].
Wir fassen die Konstruktion an dieser Stelle kurz zusammen, da sie erstens die Basis für das Plotten schafft und zweitens als Grundlage für weitere Wavelets (z.B. Primbs) dient.

Definition 4.1 (Dualität)**.** Die Skalierungsfunktionen ϕ und $\tilde{\phi}$ heißen zueinander dual, falls gilt:

$$\langle \phi, \tilde{\phi}(\cdot - k) \rangle = \delta_{0,k} \tag{4.3}$$

Die zugehörigen Multiskalenanalysen (MRA) $(V_j)_j, (\tilde{V}_j)_j$ heißen zueinander dual, wenn dies für alle Funktionen $\phi \in (V_j)_j, \tilde{\phi} \in (\tilde{V}_j)_j$ erfüllt ist.

Definition 4.2 (Biorthogonalität)**.** Zwei indizierte Systeme von Vektoren $(u_i)_{i\in\mathbb{Z}}$, $(v_i)_{i\in\mathbb{Z}}$ heißen zueinander biorthogonal, falls gilt $\langle u_i, v_j \rangle = \delta_{i,j}$.

Der folgende Satz aus [AC92], übernommen aus [Pri06], garantiert nun biorthogonale Waveletbasen von $L^2(\mathbb{R})$ unter gewissen Voraussetzungen an die Skalierungsfunktionen:

Satz 4.3. *Seien $\phi, \tilde{\phi}$ zueinander duale Skalierungsfunktionen mit kompaktem Träger, die eine MRA erzeugen und deren Fouriertransformationen in der folgenden Art für $C, \epsilon > 0$ beschränkt sind:*

$$\hat{\phi}(\xi) \leq C(1 + |\xi|)^{-0.5-\epsilon}, \hat{\tilde{\phi}}(\xi) \leq C(1 + |\xi|)^{-0.5-\epsilon}, \tag{4.4}$$

dann bilden ψ und $\tilde{\psi}$ mit:

$$\psi = \sum_{k\in\mathbb{Z}} b_k \phi(2 \cdot -k) \text{ mit } b_k = (-1)^k \tilde{a}_{1-k}, \tag{4.5}$$

$$\tilde{\psi} = \sum_{k\in\mathbb{Z}} \tilde{b}_k \phi(2 \cdot -k) \text{ mit } \tilde{b}_k = (-1)^k a_{1-k} \tag{4.6}$$

zueinander duale (d.h. $\langle \psi_{j,k}, \tilde{\psi}_{\tilde{j},\tilde{k}} \rangle = \delta_{j,\tilde{j}} \delta_{k,\tilde{k}}$) Rieszbasen von $L^2(\mathbb{R})$.

Die biorthogonalen Skalierungsfunktionen aus [AC92] werden nun wie folgt konstruiert:

Biorthogonale Wavelets aus kardinalen B-Splines
Als erste Skalierungsfunktion ϕ_d wird ein kardinaler B-Spline der Ordnung $d \in \mathbb{N}$ verwendet. Dieser besitzt viele nützliche Eigenschaften:

- ϕ_d besitzt kompakten Träger: $supp(\phi_d) = [l_1, l_2] = [-\lfloor \frac{d}{2} \rfloor, \lceil \frac{d}{2} \rceil]$.
- Sie erzeugen (wie gewünscht) eine MRA. Die Verfeinerungsrelation lautet:

$$\phi_d = \sum_{k=l_1}^{l_2} a_k \phi_d(2 \cdot -k) \text{ mit } a_k = 2^{1-d} \binom{d}{k + \lfloor d/2 \rfloor}. \tag{4.7}$$

Weiterhin sind alle ϕ_d normiert und symmetrisch. Alle Eigenschaften sowie weitere Informationen kann man in [Sch07] nachlesen.
In [AC92] werden wie folgt zu einem kardinalen B-Spline der Ordnung d duale Skalierungsfunktionen erzeugt:

Satz 4.4. *Sei $\tilde{d} \in \mathbb{N}$ so gewählt, dass $\frac{\tilde{d}}{d}$ groß genug ist und die Summe $\tilde{d}+d$ gerade, dann ist*

$$\tilde{\phi}_{d,\tilde{d}} = \sum_{k\in\mathbb{Z}} \tilde{a}_k \tilde{\phi}_{d,\tilde{d}}(2\cdot -k) \tag{4.8}$$

mit

$$\tilde{a}_k = \sum_{n=0}^{\frac{d+\tilde{d}}{2}-1} \sum_{l=0}^{2n} 2^{-\tilde{d}-2n}(-1)^{n+l} \binom{\tilde{d}}{k+\lfloor\frac{\tilde{d}}{2}\rfloor - l + n} \binom{\frac{d+\tilde{d}}{2}-1+n}{n} \binom{2n}{l} \tag{4.9}$$

eine duale Skalierungsfunktion zu ϕ_d.
Sie besitzt kompakten Träger mit: $supp(\tilde{\phi}) = [\tilde{l}_1, \tilde{l}_2] = [-\tilde{d} - \lfloor\frac{d}{2}\rfloor + 1, \tilde{d} + \lceil\frac{d}{2}\rceil - 1]$

Diese Formeln für die Masken der CDF-Skalierungsfunktionen sind in js implementiert, zusätzlich sind die Koeffizienten auch in der Datenbank gespeichert. Die Skalierungsfunktionen können somit auch mit der Routine für verfeinerbare Funktionen geplottet werden.
Für die Wavelets werden die Masken anhand der Gleichung 4.5 zur Laufzeit berechnet. Damit können auch die Wavelets mit der Routine für verfeinerbare Funktionen berechnet werden.

4.5 Primbs-Wavelets

Die Primbs-Wavelets sind bisher die einzigen in die Datenbank aufgenommenen Wavelets, die keine Basis auf $L^2(\mathbb{R})$, sondern nur auf $L^2([0,1])$ bilden.
Zwar könnte man die zuvor konstruierten Wavelets nutzen und am Rand einfach 'abschneiden', dies ist jedoch im Allgemeinen nicht wünschenswert, da beispielsweise nicht einmal die konstanten Funktionen durch die abgeschnittenen B-Splines reproduziert werden können.
In der Dissertation von Frau Primbs ([Pri06]) findet man alles nötige, was für die Konstruktion und das Plotten der Primbs-Wavelets notwendig ist.
Die Verfeinerungsmatrizen $M_{j,1}$ wurden für das kleinstmögliche j (in der Datenbank unter j_0 gespeichert) mithilfe einer vorhandenen C++-Routine [TR] berechnet und in der Datenbank gespeichert.

Erweitern der Matrix $M_{j,1}$
In der Datenbank ist die Matrix $M_{j,1}$ für das kleinstmögliche j gespeichert. Um die Matrix für größeres j zu erhalten muss die Matrix erweitert werden. Wie das funktioniert wird auch aus [Pri06] ersichtlich. Die entsprechende Methode ist in 'primbsWavelets.js' unter dem Namen 'extendMj1(...)' implementiert.

5 Plotten der relevanten Funktionen

5.1 Punktauswertung verfeinerbarer Funktionen

Sei ϕ eine stetige Skalierungsfunktion mit kompakten Träger, $supp(\phi) \subseteq [l_1, l_2]$ mit $l_1, l_2 \in \mathbb{N}$ (d.h. insbesondere $\phi(l_1) = 0 = \phi(l_2)$), dann gilt an allen ganzzahligen Punkten $x = l \in \mathbb{Z}$:

$$\phi(l) = \sum_{k=l_1}^{l_2} a_k \phi(2l-k) = \sum_{m=l_1+1}^{l_2-1} a_{2l-m} \phi(m) \text{ für alle } l_1 + 1 \leq l \leq l_2 - 1. \tag{5.1}$$

Fasst man diese Bedingungen in einem Gleichungssystem zusammen, so erhält man ein Eigenwertproblem zum Eigenwert 1 (siehe auch [Raa16]):

$$\begin{pmatrix} \phi(l_1+1) \\ \phi(l_1+2) \\ \vdots \\ \phi(l_2-2) \\ \phi(l_2-1) \end{pmatrix} = \begin{pmatrix} a_{l_1+1} & a_{l_1} & & & & & \\ a_{l_1+3} & a_{l_1+2} & a_{l_1+1} & a_{l_1} & & & \\ a_{l_1+5} & a_{l_1+4} & a_{l_1+3} & a_{l_1+2} & a_{l_1+1} & a_{l_1} & \\ & & & \ddots & \ddots & \ddots & \\ & & & a_{l_2} & a_{l_2-1} & a_{l_2-2} & a_{l_2-3} \\ & & & & & a_{l_2} & a_{l_2-2} \end{pmatrix} \begin{pmatrix} \phi(l_1+1) \\ \phi(l_1+2) \\ \vdots \\ \phi(l_2-2) \\ \phi(l_2-1) \end{pmatrix}. \tag{5.2}$$

Für die Skalierungsfunktionen gilt außerdem $\sum_{k\in\mathbb{Z}} \phi(x-k) = 1$ für alle $x \in \mathbb{R}$. Nehmen wir diese Bedingung auf, so erhalten wir das Gleichungssystem:

$$\begin{pmatrix} & A - I & \\ 1 & \cdots & 1 \end{pmatrix} \begin{pmatrix} \phi(l_1+1) \\ \vdots \\ \phi(l_2-1) \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}. \tag{5.3}$$

Dahmen und Micceli haben gezeigt, dass dieses Gleichungssystem unter schwachen Voraussetzungen eindeutig lösbar ist.
Bis jetzt funktioniert es in allen praktischen Fällen außer bei der CDF(2,2)-Skalierungsfunktion.

Die Funktionswerte von ϕ an allgemeinen dyadischen Punkten $x = 2^{-j}l, j \geq 0, l \in \mathbb{Z}$ kann man rekursiv mit Hilfer der Verfeinerungsgleichung brechnen, denn

$$\phi(2^{-j}l) = \sum_{k=l_1}^{l_2} a_k \phi(2^{1-j}l - k) = \sum_{k=l_1}^{l_2} a_k \phi(2^{1-j}(l - 2^{j-1}k)). \tag{5.4}$$

Eine naive rekursive Berechnung führt zu hohem Rechenaufwand, da viele Werte mehrfach berechnet werden. Stattdessen empfiehlt es sich zunächst für $j = 1$ alle Werte zu berechnen. Diese Werte können dann für $j = 2$ genutzt werden. Dann berechnet man alle Werte für $j = 3$ usw. So muss jeder Wert nur einmal berechnet werden.

Die entsprechende Implementierung findet sich in der Datei 'math2.js' und heißt 'iterativePointEvaluation2'. Mit dieser Routine können außerdem auch Punktwerte von Ableitungen verfeinerbarer Funktionen berechnet werden.

5.2 Punktauswertung per Subdivison-Schema

Die Punktauswertung von verfeinerbaren Funktionen kann auch mittels eines sogenannten Subdivision-Schemas vorgenommen werden. Diese Methode ist bis jetzt noch nicht in Javascript implementiert. Sie soll aber noch implementiert werden, da sie es ermöglicht die CDF(2,2)-Skalierungsfunktion zu plotten. Außerdem lieferte sie 'schärfere' Bilder für nicht stetige Skalierungsfunktionen.
Eine Abfrage des Sobolev-Exponenten könnte entscheiden, welche Plotroutine verwendet wird. Im Allgemeinen sollte jedoch die bisherige Methode vorgezogen werden, da sie wesentlich effizienter ist.
Ein Referenz-Matlab-Code für das Subdivison-Schema ist auf Anfrage verfügbar.

5.3 Punktauswertung der B-Splines

Zwar lassen sich die kardinalen B-Splines auch über die Verfeinerbarkeitsrelation auswerten, die folgende Methode ist jedoch effizienter (zumindest bei der Auswertung eines festen Punktes). Außerdem wird sie für die B-Splines bei einem Nicht-Standard-Gitter benötigt.

Definition 5.1 (Normierter B-Spline)**.** Sei $y_i, ..., y_{i+m}$ eine Knotenfolge. Dann ist der normierte B-Spline erster Ordnung durch

$$N_i^1(x) = \begin{cases} 1 & , y_i \leq x < y_{i+1} \\ 0 & , \textit{sonst} \end{cases} \tag{5.5}$$

definiert. Der normierte B-Spline m-ter Ordnung, $m \geq 1$ ist durch

$$N_i^m(x) = \begin{cases} 0 & , y_i = y_{i+m} \\ \frac{x-y_i}{y_{i+m-1}-y_i} N_i^{m-1}(x) & , y_i < y_{i+m-1} \text{ und } y_{i+1} = y_{i+m} \\ \frac{y_{i+m}-x}{y_{i+m}-y_{i+1}} N_{i+1}^{m-1}(x) & , y_i = y_{i+m-1} \text{ und } y_{i+1} < y_{i+m} \\ \frac{x-y_i}{y_{i+m-1}-y_i} N_i^{m-1}(x) + \frac{y_{i+m}-x}{y_{i+m}-y_{i+1}} N_{i+1}^{m-1}(x) & , y_i < y_{i+m-1} \text{ und } y_{i+1} < y_{i+m} \end{cases} \tag{5.6}$$

gegeben.
Die μ-te Ableitung ${}^{\mu}N_i^m$ lässt sich rekursiv folgendermaßen berechnen:

$${}^{\mu}N_i^m = (m-1) \cdot \begin{cases} 0 & , y_i = y_{i+m+1} \\ \frac{{}^{(\mu-1)}N_i^{m-1}}{y_{i+m-1}-y_i} & , y_i < y_{i+m-1} \text{ und } y_{i+1} = y_{i+m} \\ -\frac{{}^{(\mu-1)}N_{i+1}^{m-1}}{y_{i+m}-y_{i+1}} & , y_i = y_{i+m-1} \text{ und } y_{i+1} < y_{i+m} \\ \frac{{}^{(\mu-1)}N_i^{m-1}}{y_{i+m-1}-y_i} - \frac{{}^{(\mu-1)}N_{i+1}^{m-1}}{y_{i+m}-y_{i+1}} & , y_i < y_{i+m-1} \text{ und } y_{i+1} < y_{i+m} \end{cases} \tag{5.7}$$

Eine Herleitung der Formel findet man in etwa in [TL08].

6 Erklärung zu bestimmten Dateien

6.1 PlotFeatures.js

Zum Plotten der Skalierungsfunktionen und Wavelets wurde die Bibliothek function-plot.js (https://mauriciopoppe.github.io/function-plot/) genutzt.
Wir benötigen dabei allerdings ein paar 'Extras', die mittels der Datei PlotFeatures hinzugefügt werden:

- Die vielen Punkte die (beispielsweise bei verfeinerbaren Funktionen) berechnet werden, sollen nicht alle geplottet werden, sondern nur so viele, wie für ein 'scharfes' Bild in dem ausgewählten Bereich der Funktion notwendig sind.
- Ein jeder Plot soll ein kleinen 'Message-div' besitzen, der anzeigt, falls die Funktion nicht stetig sein könnte.
- Es soll Plots geben, die zusätzlich über einen Schieberegler verfügen, mit dem ausgewählt werden kann, welche der Funktionen gerade angezeigt wird.

Um alle diese Wünsche zu erfüllen wurde in der PlotFeatures.js Datei eine Funktion *buildPlot*() geschrieben. Diese geht in etwa folgendermaßen vor:

- Es wird ein function-plot-Objekt (Datentyp der externen Bibliothek 'function-plot') erstellt.
- Es wird ein bigPlot Objekt (eigener Datentyp) erstellt. Dieser enthält den Plot selbst, alle verfügbaren Werte, sowie eine Funktion bestimmte Werte zu plotten.
- Nun wird der plot mit der Methode 'zoomFilter()' verknüpft. Diese sorgt dafür, dass beim rein- und rauszoomen, stets eine gewisse Anzahl von Werten aus dem aktuellen Ausschnitt geplottet werden. (Alle Werte würde zu lange dauern und ergibt kein schönes Bild.) Wird soweit reingezoomt, dass keine detaillierten Punkte mehr verfügbar sind, so wird ein Warnfenster geöffnet.
- Es wird der mitgegebene Sobolev-Exponent überprüft und ggf. ein div mit einer Warnnachricht an den Plot angehängt.

Die Funktion $'buildPlot2(target, calcValFunc, params, slider1Range, slider2RangeFunct, params2, sobolevExp)'$ besitzt alle Funktionalitäten der vorherigen Funktion. Sie fügt zusätzlich 3 Schieberegler zum Plot dazu. Sie ist insbesondere für die Primbs-Funktionen notwendig, wo die Skalierungsfunktionen verschieden aussehen. Weiter besitzt sie die Funktion 'calcValFunc', die in Abhängigkeit von der Einstellung zweier Schieberegler dynamisch neue Werte für den Plot berechnet.

Die weiteren Eingabeparameter der Funktion sind:

- params - weitere Parameter für die Funktion calcValFunc.

- slider1Range - zulässiges Intervall für den ersten Schieberegler
- slider2RangeFunc - Funktion, die aus dem Wert des ersten Schiebereglers das zulässige Intervall des zweitens berechnet.
- params2 - zusätzliche Parameter für 'slider2RangeFunc'

7 Aktueller Stand und mögliche Weiterarbeit

Bis jetzt wurde die Struktur der Datenbank und der Website festgelegt und mit den Wavelettypen (Haar, Daubechies, Coiflets, Symmlets, CDF und Primbs) gefüllt. Für diese Wavelettypen sind mittlerweile viele wichtigen Informationen in der Datenbank gespeichert. Außerdem stehen für alle Wavelettypen Plotroutinen zur Verfügung und die Plots können auf der Website angeschaut werden. Es gibt 3 Übersichtsseiten, auf denen die Waveletfamilien in verschiedenen Arten sortiert sind und von denen man aus auf die Steckbriefe gelangt.

Folgende Punkte fehlen noch und könnten als nächstes umgesetzt werden:

- Formulieren und umsetzen einer klaren Gliederung für die Steckbriefe selbst
- Füllen der Steckbriefe mit weiteren Informationen
- Implementieren des Subdivison-Schemas als weitere Hilfsroutine fürs Plotten der Wavelettypen
- Füllen der Lücken in der Datenbank (z.B.: Berechnung der offenen Hölder- und Sobolevexponenten)
- Hinzufügen von neuen Wavelettypen zur Datenbank
- ...

Für den Fall, dass an diesem Projekt weitergearbeitet wird, wünschen wir viel Spaß und Erfolg!!!
Für Rückfragen stehen wir dann natürlich gerne zur Verfügung ;-).

Andreas Andrzejczak (aandrzej@students.uni-mainz.de) und
Simon Thomas (sithomas@students.uni-mainz.de)

Literatur

[AC92] A. Cohen, J.-C. F. I. Daubechies D. I. Daubechies: Biorthogonal bases of compactly supported wavelets. In: *Communications on Pure and Applied Mathematics* (1992)

[Cer08] Cerna, Dana: On the exact values of coefficients of coiflets. In: *Central European Journal of Mathematics* (2008), Nr. 6, S. 159–169

[Dau88] Daubechies, Ingrid: Orthonormal bases of ccompactly supported wavelets. In: *Comm. Pure Appl. Math.* (1988), Nr. 41, S. 909–996

[Dau92] Daubechies, Ingrid: *Ten Lectures on Wavelets* -. SIAM, 1992

[Dau93] Daubechies, Ingrid: Orthonormal bases of ccompactly supported wavelets II - Variations on a theme. In: *SIAM J. Math. Anal.* (1993), Nr. 24, S. 499–519

[Pri06] Primbs, M.: *Stabile biorthogonale Spline-Waveletbasen auf dem Intervall*, Universität Duisburg-Essen, Diss., 2006. `http://duepublico.uni-duisburg-essen.de/servlets/DerivateServlet/Derivate-5693/Disse`

[Raa16] Raasch, Thorsten: *Lecture notes in 'Numerische Fourier- und Waveletmethoden'.* 2016

[Sch07] Schumaker, Larry: *Spline Functions: Basic Theory* -. Cambridge : Cambridge University Press, 2007

[TL08] T. Lyche, K.Morken: *Spline Methods Draft.* `http://folk.uio.no/in329/nchap3.pdf`. Version: 2008

[TR] Thorsten Raasch, Manuel W.: *WaveletTL - the Wavelet Template Library.* Copyright (c) 2002-2009,