Non-equidistant approximate DFT based on Z-splines

Bachelor Thesis

by
Claude J. Gittelson

supervised by
Prof. R. Hiptmair

Department of Mathematics
Eidgenössische Technische Hochschule
CH-8092 Zürich, Switzerland

June 2006

Abstract

In this paper, we consider an algorithm that efficiently evaluates a trigonometric polynomial at arbitrarily spaced nodes. It is based on the approximation of the polynomial by a function we can evaluate easily. We are particularly interested in the case where we interpolate the trigonometric polynomial using high-order cardinal interpolation kernels known as Z-splines, which we construct and study in a general setting.

Introduction

We are interested in the fast evaluation of a trigonometric polynomial

$$f(x_j) = \sum_k \hat{f}_k \, \mathrm{e}^{-2\pi \mathrm{i} k x_j} \ .$$

For equidistant x_j, this can be done with the well-known FFT algorithm. In [5], G. Steidl describes an algorithm that efficiently evaluates f at arbitrary x_j. The idea is to approximate $f(x_j)$ by

$$s(x_j) = \sum_l g_l \psi(n x_j - l) \ ,$$

which can be evaluated efficiently if ψ has compact support. G. Steidl suggests two possibilities for ψ: truncated Gaussian bells and B-splines. We will study this algorithm with compactly supported high-order cardinal splines called Z-splines.

Throughout this paper, we will restrict ourselves to one dimension. For a multidimensional treatment of this algorithm, see [2, Section 2].

In Section 1, we review the various Fourier transforms that appear later on, i.e. the continuous Fourier transform, Fourier series, and the discrete Fourier transform. We also prove the Poisson summation formula, which we use repeatedly in our analysis.

Section 2 develops cardinal interpolation theory in a general setting. In particular, we relate properties of the interpolation kernel and the interpolation operator. In the following section, we construct Z-splines as an example for high-order cardinal interpolation kernels.

In Section 4, we formulate and analyze our evaluation algorithm for trigonometric polynomials in a general form. Section 4.6 specializes our results to the case of cardinal interpolation. In Section 4.7, we look at a 'dual' algorithm for evaluating

$$f_j = \sum_k \hat{f}(\xi_k)\, \mathrm{e}^{-2\pi \mathrm{i} j \xi_k} ,$$

where the nodes are spaced equally in the time domain and arbitrarily in the frequency domain. For a more general version of this algorithm, in which the nodes are arbitrary in both time and frequency domains, see [2, Section 3].

Section 5 is an overview of our MATLAB implementation of the above algorithms using Z-splines. Finally, in Section 6, we present a series of numerical experiments. These show that the approximate evaluation of trigonometric polynomials with Z-splines does not converge as fast as with truncated Gaussian bells or B-splines.

Notation

We will use the convenient MATLAB notation for intervals in $\mathbb{Z}$, that is,

$$n = a : b \;:\Leftrightarrow\; n \in \{a, a+1, \ldots, b\} .$$

It will also be useful to denote by

$$\mathbb{Z}_n := \left\{ -\left\lfloor \frac{n}{2} \right\rfloor, \ldots, -\left\lfloor \frac{n}{2} \right\rfloor + n - 1 \right\} = \left\{ k \in \mathbb{Z}\,;\, -\frac{n}{2} \le k < \frac{n}{2} \right\}$$

the n whole numbers of smallest magnitude. Furthermore, $P_n(X)$ will denote the vector space of polynomials of degree $\le n$ on X and $P_n := P_n(\mathbb{R})$.

1 Fourier transforms

Throughout this paper, we will use various types of Fourier transforms. In this section, we define these and state a few important properties, mainly to set our notation and conventions[i]. For simplicity, we restrict ourselves to the one-dimensional case. All of the functions[ii] in this section are assumed to be complex valued.

1.1 The continuous Fourier transform

Our convention for the continuous *Fourier transform* is

$$\hat{\varphi}(k) := \int_{-\infty}^{\infty} \varphi(x)\, \mathrm{e}^{2\pi \mathrm{i} k x}\, \mathrm{d}x . \tag{1.1}$$

[i]We use the factors $\mathrm{e}^{2\pi \mathrm{i} kx}$ and $\mathrm{e}^{-2\pi \mathrm{i} kx}$ instead of $\mathrm{e}^{-\mathrm{i}kx}$ and $\mathrm{e}^{\mathrm{i}kx}$, which are more commonly seen in mathematical literature.

[ii]unvaryingly named φ

The inverse transform is

$$\varphi(x) = \int_{-\infty}^{\infty} \hat{\varphi}(k)\, \mathrm{e}^{-2\pi \mathrm{i} kx}\, \mathrm{d}k \;, \tag{1.2}$$

since the functions $x \mapsto \mathrm{e}^{2\pi \mathrm{i} kx}$ are orthogonal with respect to the standard L^2-inner product, that is,

$$\int_{-\infty}^{\infty} \mathrm{e}^{2\pi \mathrm{i} kx}\, \mathrm{e}^{-2\pi \mathrm{i} lx}\, \mathrm{d}x = \delta(k-l) \;, \tag{1.3}$$

where δ refers to the Dirac distribution.

The Plancherel theorem for our choice of Fourier transform is

$$\|\varphi\|_{L^2(\mathbb{R})} = \|\hat{\varphi}\|_{L^2(\mathbb{R})} \;. \tag{1.4}$$

1.2 Fourier series

If φ is T-periodic, we define its *Fourier series* as

$$\varphi(x) = \sum_{k\in\mathbb{Z}} \hat{\varphi}_k\, \mathrm{e}^{-2\pi \mathrm{i} kx/T} \;, \tag{1.5}$$

where the Fourier coefficients are

$$\hat{\varphi}_k := \frac{1}{T} \int_{-T/2}^{T/2} \varphi(x)\, \mathrm{e}^{2\pi \mathrm{i} kx/T}\, \mathrm{d}x \;. \tag{1.6}$$

Equation (1.5) follows from the orthonormality of the $\mathrm{e}^{2\pi \mathrm{i} kx/T}$,

$$\frac{1}{T} \int_{-T/2}^{T/2} \mathrm{e}^{2\pi \mathrm{i} kx/T}\, \mathrm{e}^{-2\pi \mathrm{i} lx/T}\, \mathrm{d}x = \delta_{kl} \;, \tag{1.7}$$

where δ_{kl} is the Kronecker delta. Pointwise convergence holds if φ is piecewise C^1 and $\varphi(x) = \frac{1}{2}\left(\varphi(x^-) + \varphi(x^+)\right)$ at discontinuities.

Parseval's theorem states that

$$\frac{1}{T} \int_{-T/2}^{T/2} |\varphi(x)|^2\, \mathrm{d}x = \sum_{k\in\mathbb{Z}} |\hat{\varphi}_k|^2 \;. \tag{1.8}$$

1.3 The discrete Fourier transform

Similarly, we define the *discrete Fourier transform* of length n as

$$\hat{\varphi}_k = \frac{1}{n} \sum_{j\in\mathbb{Z}_n} \varphi_j\, \mathrm{e}^{2\pi \mathrm{i} kj/n} \tag{1.9}$$

for $k \in \mathbb{Z}_n$ and $(\varphi_j)_{j\in\mathbb{Z}_n} \in \mathbb{C}^n$. Since

$$\frac{1}{n} \sum_{j\in\mathbb{Z}_n} \mathrm{e}^{2\pi \mathrm{i} kj/n}\, \mathrm{e}^{-2\pi \mathrm{i} lj/n} = \delta_{kl} \;, \tag{1.10}$$

the inverse transform is

$$\varphi_j = \sum_{k\in\mathbb{Z}_n} \hat{\varphi}_k \, \mathrm{e}^{-2\pi \mathrm{i} kj/n} \ . \tag{1.11}$$

Both the discrete Fourier transform and its inverse can be computed in $O(n \log n)$ time using a fast Fourier transform (FFT) algorithm; see for example [1].

Parseval's theorem for the discrete Fourier transform is

$$\frac{1}{n} \sum_{j\in\mathbb{Z}_n} |\varphi_j|^2 = \sum_{k\in\mathbb{Z}_n} |\hat{\varphi}_k|^2 \ . \tag{1.12}$$

1.4 The Poisson summation formula

The Poisson summation formula relates the continuous Fourier transform to its discrete counterparts by expressing the periodization of a function as a Fourier series. It is used frequently throughout this paper.

Proposition 1.1. *For $\varphi : \mathbb{R} \to \mathbb{C}$ integrable and piecewise C^1 and $\varphi(x) = \frac{1}{2}\left(\varphi(x^-) + \varphi(x^+)\right)$ at discontinuities,*

$$\sum_{j\in\mathbb{Z}} \varphi(x - jT) = \frac{1}{T} \sum_{k\in\mathbb{Z}} \hat{\varphi}\left(\frac{k}{T}\right) \mathrm{e}^{-2\pi \mathrm{i} kx/T} \ ,$$

where $\hat{\varphi}$ is the continuous Fourier transform (1.1) *of φ and $T > 0$.*

Proof. Since the left-hand side is T-periodic, the assumptions imply pointwise convergence of the Fourier series $\sum_{k\in\mathbb{Z}} c_k \, \mathrm{e}^{-2\pi \mathrm{i} kx/T}$ with

$$c_k = \frac{1}{T} \sum_{j\in\mathbb{Z}} \int_{-T/2}^{T/2} \varphi(x - jT) \, \mathrm{e}^{2\pi \mathrm{i} kx/T} \, \mathrm{d}x = \frac{1}{T} \int_{-\infty}^{\infty} \varphi(y) \, \mathrm{e}^{2\pi \mathrm{i} ky/T} \, \mathrm{d}y = \frac{1}{T} \hat{\varphi}\left(\frac{k}{T}\right) \ .$$

We can switch integration and summation in the first step because φ is integrable. □

2 Cardinal interpolation theory

2.1 Cardinal interpolation operators

In this section, we study some properties of cardinal interpolation operators and, in particular, link these properties to those of the interpolation kernel. In Section 3.3, we will specialize our results to Z-spline interpolation kernels.

Let $\psi \in L^1(\mathbb{R})$ be piecewise C^1 with $\psi(x) = \frac{1}{2}\left(\psi(x^-) + \psi(x^+)\right)$ at discontinuities and $h > 0$. Also, let $C_{\mathrm{pw}}(\mathbb{R}, \mathbb{R}^d)$ denote the space of piecewise continuous functions from $\mathbb{R}$ to $\mathbb{R}^d$. We will call the operator $I_\psi^h : C_{\mathrm{pw}}(\mathbb{R}, \mathbb{R}^d) \to C_{\mathrm{pw}}(\mathbb{R}, \mathbb{R}^d)$ defined by

$$\left(I_\psi^h u\right)(x) := \sum_{j\in\mathbb{Z}} u(jh) \psi\left(\frac{x}{h} - j\right) \tag{2.1}$$

a *cardinal interpolation operator* if $(I_\psi^h u)(jh) = u(jh)$ for all $j \in \mathbb{Z}$. In this situation, the function ψ is the *cardinal interpolation kernel* of I_ψ^h.

Proposition 2.1. *For any $h > 0$, the following conditions are equivalent:*

(i) I_ψ^h *is a cardinal interpolation operator,*

(ii) I_ψ^1 *is a cardinal interpolation operator,*

(iii) $\psi(j) = \delta_{j0}$ *for* $j \in \mathbb{Z}$.

Proof. (i)⇔(ii) follows from $(I_\psi^h u)(x) = \sum_{j\in\mathbb{Z}} u(jh)\psi\big(\frac{x}{h} - j\big) = (I_\psi^1 u_h)(\frac{x}{h})$, where $u_h(x) := u(xh)$.
(ii)⇒(iii) follows from (2.1) if $u(j) = \delta_{j0}$, since $u(j) = (I_\psi^1 u)(j) = \psi(j)$.
(iii)⇒(ii) is implied by (2.1), $(I_\psi^1 u)(j) = \sum_{l\in\mathbb{Z}} u(l)\delta_{jl} = u(j)$. □

Proposition 2.1 states that cardinal interpolation kernels are characterized by (iii). If ψ and φ are cardinal interpolation kernels, then $(I_\psi^1\varphi)(x) = \psi(x)$ by (2.1) since $\varphi(j) = \delta_{j0}$. Also, $I_\psi^h = I_\psi^h \circ I_\varphi^h$ for all $h > 0$ because

$$\big(I_\psi^h I_\varphi^h u\big)(x) = \sum_{j\in\mathbb{Z}} \big(I_\varphi^h u\big)(jh)\psi\Big(\frac{x}{h} - j\Big) = \sum_{j\in\mathbb{Z}} u(jh)\psi\Big(\frac{x}{h} - j\Big) = \big(I_\psi^h u\big)(x) \ .$$

In particular, I_ψ^h is a projection.

We will consider the case $d = 1$; the general case follows from componentwise application of the one-dimensional results.

Often, we will require ψ to have compact support. This allows an efficient evaluation of $I_\psi^h u$ and, more importantly for theoretical purposes, it ensures a certain regularity of the Fourier transform $\hat{\psi}$. By the Paley-Wiener theorem,[iii] $\hat{\psi}$ will be an entire analytic function of exponential type in this situation.

It turns out that the interpolation property $(I_\psi^h u)(jh) = u(jh)$ is not important for our interpolation results; these are in fact approximation results that hold for any I_ψ^h defined as in (2.1) with certain polynomial-preservation properties. We define the approximation order accordingly.

Definition 2.2. The operator I_ψ^h is of *order* ν iff it acts as the identity on $P_{\nu-1}$.

Clearly, this is equivalent to the discrete moment conservation property

$$\sum_{j\in\mathbb{Z}} (jh)^n \psi\Big(\frac{x}{h} - j\Big) = x^n \quad \text{for} \quad n = 0 : \nu - 1 \ . \tag{2.2}$$

Proposition 2.3. *If I_ψ^1 is of order ν, then so are I_ψ^h for all $h > 0$.*

Proof.

$$\sum_{j\in\mathbb{Z}} (jh)^n \psi\Big(\frac{x}{h} - j\Big) = h^n \sum_{j\in\mathbb{Z}} j^n \psi\Big(\frac{x}{h} - j\Big) = h^n \left(\frac{x}{h}\right)^n = x^n \ .$$

□

We will say that ψ is of order ν if I_ψ^h is for any, and therefore all, $h > 0$. The following theorem gives a rather elementary L^∞ estimate based on a Taylor series expansion.[iv]

[iii] see for example [3, Theorem 7.23]

[iv] In fact, all of the interpolation results in this section are based on Taylor series expansions.

Theorem 2.4. *If ψ has compact support* $\operatorname{supp}\psi \subseteq [-m,m]$ *and I_ψ^h is of order ν, then there is a constant $c = c(\nu,\psi)$ such that for all $u \in C^\nu(\mathbb{R})$,*

$$\left\|u - I_\psi^h u\right\|_\infty \leq ch^\nu \left\|u^{(\nu)}\right\|_\infty .$$

Proof. A simple calculation gives us

$$\begin{aligned}\sum_{j\in\mathbb{Z}} (x-jh)^n \,\psi\Big(\frac{x}{h}-j\Big) &= \sum_{k=0}^{n} \binom{n}{k} x^{n-k}(-1)^k \sum_{j\in\mathbb{Z}} (jh)^k \psi\Big(\frac{x}{h}-j\Big) \\ &= \sum_{k=0}^{n} \binom{n}{k} x^{n-k}(-x)^k = (x-x)^n = \delta_{n0}\end{aligned} \tag{2.3}$$

for $n \leq \nu - 1$. A Taylor series expansion about $x \in \mathbb{R}$ shows

$$\begin{aligned} u(x) - \left(I_\psi^h u\right)(x) &= f(x) - \sum_{j\in\mathbb{Z}} u(jh)\psi\Big(\frac{x}{h}-j\Big) \\ &\overset{(2.2),n=0}{=} \sum_{j\in\mathbb{Z}} \left(\sum_{p=1}^{\nu-1} \frac{u^{(p)}(x)}{p!}(jh-x)^p + r_j(h) \right) \psi\Big(\frac{x}{h}-j\Big) \\ &\overset{(2.3)}{=} \sum_{j=\lceil x/h\rceil - m}^{\lfloor x/h\rfloor + m} r_j(h)\psi\Big(\frac{x}{h}-j\Big) .\end{aligned}$$

The estimate now follows from

$$|r_j(h)| = \left| \frac{u^{(\nu)}(\xi)}{\nu!}(jh-x)^\nu \right| \leq \left\|u^{(\nu)}\right\|_\infty \frac{m^\nu}{\nu!} h^\nu .$$

□

2.2 Characterization of high-order kernels

For the following theorems, we need to know the Fourier transform of an interpolated function.

Lemma 2.5. *For $u \in C_{\mathrm{pw}}(\mathbb{R})$ such that $\hat{u}$ satisfies the conditions of Proposition 1.1 and $h > 0$,*

$$\widehat{\left(I_\psi^h u\right)}(k) = \left(\sum_{r\in\mathbb{Z}} \hat{u}\Big(k + \frac{r}{h}\Big) \right) \hat{\psi}(hk) \quad \forall k \in \mathbb{R} .$$

Proof. The calculation

$$\int_{-\infty}^{\infty} \psi\Big(\frac{x}{h}-j\Big)\, \mathrm{e}^{2\pi \mathrm{i} kx}\,\mathrm{d}x = h \int_{-\infty}^{\infty} \psi(y)\,\mathrm{e}^{2\pi\mathrm{i}kh(y+j)}\,\mathrm{d}y = h\hat{\psi}(kh)\,\mathrm{e}^{2\pi\mathrm{i}kjh}$$

and Proposition 1.1 imply

$$\widehat{\left(I_\psi^h u\right)}(k) = h \sum_{j\in\mathbb{Z}} u(jh)\,\mathrm{e}^{2\pi\mathrm{i}kjh}\,\hat{\psi}(hk) = \sum_{r\in\mathbb{Z}} \hat{u}\Big(k+\frac{r}{h}\Big)\,\hat{\psi}(hk) .$$

□

We will switch to the frequency domain to show an estimate in the Sobolev norm, similar to G. Strang and G. Fix in [6]. For any $s \in \mathbb{R}$, the Sobolev norm is given by

$$\|v\|_{H^s}^2 := \int_{-\infty}^{\infty} |\hat{v}(k)|^2 \, (1+k^2)^s \mathrm{d}k \tag{2.4}$$

where the Fourier transform $\hat{v}$ is defined as in (1.1). The space $H^s(\mathbb{R})$ is the set of all distributions v for which $\|v\|_{H^s}$ is finite, endowed with the obvious Hilbert space structure.

The following theorem is similar to [6, Theorem 1]. Apart from our restriction to the one-dimensional case, the difference is that our first two conditions are slightly stronger. This gives us an interpolation result as the third condition instead of a 'best possible approximation'-type result.

Theorem 2.6. *For $\psi \in H^\eta(\mathbb{R})$ with compact support and $0 \leq \eta \leq \nu - 1$, the following conditions are equivalent:*

(i) *I_ψ^1 is of order ν,*

(ii) *$\hat{\psi}^{(n)}(k) = \delta_{k0}\delta_{n0}$ for $k \in \mathbb{Z}$ and $0 \leq n \leq \nu - 1$,*

(iii) *for all $s \leq \eta$, there is a constant c such that for all $h > 0$ and all $u \in L^2(\mathbb{R})$ with $\operatorname{supp}\hat{u} \subseteq \left[-\frac{1}{2h}, \frac{1}{2h}\right]$,*

$$\left\|u - I_\psi^h u\right\|_{H^s} \leq c h^{\nu - s} \|u\|_{H^\nu} \ .$$

Note that in (iii), the requirement $\operatorname{supp}\hat{u} \subseteq \left[-\frac{1}{2h}, \frac{1}{2h}\right]$ implies that u is an entire function by the Paley-Wiener theorem.[v] Furthermore, $u \in L^2$ implies $\hat{u} \in L^2$ by Plancherel's theorem, so in particular $u \in H^\nu(\mathbb{R}) \cap C_{\mathrm{pw}}(\mathbb{R})$. Therefore, $\|u\|_{H^\nu}$ and $I_\psi^h u$ are well-defined and $\|u\|_{H^\nu} < \infty$.

Proof. Using the Poisson summation formula, Proposition 1.1, for $x = 0$, we get

$$\begin{aligned}\sum_{j\in\mathbb{Z}} j^n \psi(x-j) &= \sum_{k\in\mathbb{Z}} \int_{-\infty}^{\infty} (-y)^n \psi(x+y)\, \mathrm{e}^{2\pi \mathrm{i} k y}\, \mathrm{d}y \\ &= \sum_{k\in\mathbb{Z}} \frac{1}{(-2\pi\mathrm{i})^n} \left(\frac{\mathrm{d}}{\mathrm{d}k}\right)^n \left(\hat{\psi}(k)\, \mathrm{e}^{-2\pi\mathrm{i}kx}\right) \\ &= \sum_{k\in\mathbb{Z}} \frac{1}{(-2\pi\mathrm{i})^n} \sum_{l=0}^{n} \binom{n}{l} \hat{\psi}^{(l)}(k)(-2\pi\mathrm{i}x)^{n-l}\, \mathrm{e}^{-2\pi\mathrm{i}kx}\end{aligned} \tag{2.5}$$

(ii)⇒(i). By (ii), the last term in (2.5) is x^n for $0 \leq n \leq \nu - 1$.
(i)⇒(ii). For $n = 0$, (2.5) and (i) imply

$$1 = \sum_{j\in\mathbb{Z}} \psi(x-j) = \sum_{k\in\mathbb{Z}} \hat{\psi}(k)\, \mathrm{e}^{-2\pi\mathrm{i}kx} \ ,$$

[v]see for example [3, Theorem 7.23]

so $\hat\psi(k) = \delta_{k0}$. Assuming $\hat\psi^{(l)}(k) = \delta_{k0}\delta_{l0}$ for $l \le n-1$ and $n \le \nu - 1$, (2.5) and (i) imply

$$\begin{aligned} x^n &= \sum_{j\in\mathbb{Z}} j^n \psi(x-j) = \sum_{k\in\mathbb{Z}} \frac{1}{(-2\pi\mathrm{i})^n}\left(\delta_{k0}(-2\pi\mathrm{i}x)^n + \hat\psi^{(n)}(k)\right)\mathrm{e}^{-2\pi\mathrm{i}kx} \\ &= x^n + \sum_{k\in\mathbb{Z}} \frac{1}{(-2\pi\mathrm{i})^n}\hat\psi^{(n)}(k)\,\mathrm{e}^{-2\pi\mathrm{i}kx}\ , \end{aligned}$$

so $\hat\psi^{(n)}(k) = 0$ for $n \ge 1$, which shows (ii).
(iii)$\Rightarrow$(ii). We shall assume $h \le 1$ and use (iii) for $\hat u = \chi_{[-1/2,1/2]}$. By Lemma 2.5, (iii) gives us

$$h^{-2(\nu-1)}\left\|u - I_\psi^h u\right\|_{H^0}^2 = h^{-2(\nu-1)}\sum_{r\in\mathbb{Z}}\int_{-\frac{1}{2h}}^{\frac{1}{2h}}\left|\delta_{r0} - \hat\psi(r+hk)\right|^2 \mathrm{d}k \to 0\ , \quad h\to 0\ .$$

Expanding $\hat\psi$ in a Taylor series about r and using $h \le 1$ leads to

$$\int_{-\frac12}^{\frac12}\left|\sum_{n=0}^{\nu-1}\frac{\delta_{r0}\delta_{n0} - \hat\psi^{(n)}(r)}{n!}h^{n-\nu+1}k^n + O(h)\right|^2 \mathrm{d}k \to 0\ , \quad h \to 0$$

for all $r \in \mathbb{Z}$. This implies $\hat\psi^{(n)}(r) = \delta_{r0}\delta_{n0}$ for $n = 0:\nu-1$.
(ii)$\Rightarrow$(iii). By Lemma 2.5, since $\operatorname{supp}\hat u \subseteq \left[-\frac{1}{2h}, \frac{1}{2h}\right]$,

$$\begin{aligned} \left\|u - I_\psi^h u\right\|_{H^s}^2 &= \int_{-\infty}^{\infty}\left|\hat u(k) - \sum_{r\in\mathbb{Z}}\hat u\Big(k + \frac{r}{h}\Big)\,\hat\psi(hk)\right|^2 (1+k^2)^s \mathrm{d}k \\ &= \sum_{r\in\mathbb{Z}}\int_{-\frac{1}{2h}}^{\frac{1}{2h}} |\hat u(k)|^2\left|\delta_{r0} - \hat\psi(r+hk)\right|^2\left(1 + h^{-2}\left(r+hk\right)^2\right)^s \mathrm{d}k\ . \end{aligned} \tag{2.6}$$

We will estimate the last term separately for $r = 0$ and $r \ne 0$. In the former case, using $\hat\psi(\xi) \overset{\text{(ii)}}{=} 1 + O(\xi^\nu)$, we have

$$\begin{aligned} \int_{-\frac{1}{2h}}^{\frac{1}{2h}} |\hat u(k)|^2\left|1 - \hat\psi(hk)\right|^2 (1+k^2)^s\mathrm{d}k &\lesssim \int_{-\frac{1}{2h}}^{\frac{1}{2h}} |\hat u(k)|^2 |hk|^{2\nu}(1+k^2)^s \mathrm{d}k \\ &\le \int_{-\frac{1}{2h}}^{\frac{1}{2h}} |\hat u(k)|^2 |hk|^{2(\nu-s)}(1+k^2)^s \mathrm{d}k \\ &\le h^{2(\nu-s)}\int_{-\frac{1}{2h}}^{\frac{1}{2h}} |\hat u(k)|^2 (1+k^2)^\nu \mathrm{d}k \end{aligned}$$

with a constant depending only on ψ and ν.

For $r \ne 0$, we will expand $\hat\psi$ in a Taylor series about r; by (ii), this is just the remainder term,

$$\hat\psi(hk+r) = \frac{\hat\psi^{(\nu)}(\vartheta_r(k))}{\nu!}(hk)^\nu\ ,$$

where for each k, $\vartheta_r(k)$ is between r and $r+hk$. Since $|hk| \le \frac12$ and $r \ne 0$, $|r+hk| \le \frac32|\vartheta_r(k)|$. Therefore, we can estimate the sum in (2.6) over $\mathbb{Z}\backslash\{0\}$ up to a constant factor by

$$\int_{-\frac{1}{2h}}^{\frac{1}{2h}} |\hat{u}(k)|^2 h^{-2s} |hk|^{2\nu} \sum_{r\in\mathbb{Z}\setminus\{0\}} \left|\hat{\psi}^{(\nu)}(\vartheta_r(k))\right|^2 |\vartheta_r(k)|^{2s} \, \mathrm{d}k \, .$$

As in the proof of [6, Theorem 1], we can bound the sum in this term by a constant for $s \leq \eta$. □

Note that the restrictions on u in (iii) are very limiting, much more limiting than in [6, Theorem 1]. This significantly weakens the implications (i),(ii)⇒(iii); however, it does no harm to (iii)⇒(ii). So Theorem 2.6 is not as much an error estimate as it is a characterization of high-order cardinal interpolation kernels.

2.3 'p-convergence'

The above estimates address the convergence of cardinal interpolation with a single ψ as h goes to zero. We would also like to study convergence for a sequence of kernels ψ of increasing order, with fixed h, similar to p-convergence in finite element theory.

Theorem 2.6 tells us that the Fourier transforms of high-order cardinal interpolation kernels are one at zero and vanish at every nonzero whole number. In this sense, they approximate the rectangular function rect $:= \chi_{(-1/2,1/2)}$ at least near $\mathbb{Z}$.

Theorem 2.7. *Let $\psi \in L^2(\mathbb{R})$ be bounded and compactly supported and $\gamma \leq 1$; also, let $R_0 := \left(-\frac{\gamma}{2}, \frac{\gamma}{2}\right)$ and $R := R_0 + \mathbb{Z}$. Then for all $h > 0$ and $u \in L^2$ with* $\operatorname{supp} \hat{u} \subseteq \frac{1}{h}R_0$*, $\hat{u}$ bounded and piecewise continuously differentiable,*[vi]

$$\left\|u - I_\psi^h u\right\|_{L^2(\mathbb{R})} \leq \|u\|_{L^2(\mathbb{R})} \max_{|k|\leq\frac{\gamma}{2}} \left(\sum_{j\in\mathbb{Z}} \left|\delta_{j0} - \hat{\psi}(j+k)\right|^2\right)^{\frac{1}{2}}$$

and

$$\left\|u - I_\psi^h u\right\|_{L^2(\mathbb{R})} \leq \frac{1}{\sqrt{h}} \|\hat{u}\|_{L^\infty(\mathbb{R})} \left\|\operatorname{rect} - \hat{\psi}\right\|_{L^2(R)} \, .$$

Proof. Let $k \in \left(-\frac{1}{2h}, \frac{1}{2h}\right)$ and $j \in \mathbb{Z}$. By Lemma 2.5, since $\operatorname{supp} \hat{u} \subseteq \frac{1}{h}R_0$,

$$\left(\hat{u} - \widehat{I_\psi^h u}\right)\left(k + \tfrac{j}{h}\right) = \hat{u}(k)\delta_{j0} - \hat{u}(k)\hat{\psi}(j+hk) = \hat{u}(k)(\operatorname{rect}(j+hk) - \hat{\psi}(j+hk)) \, .$$

In particular, $\left(\hat{u} - \widehat{I_\psi^h u}\right)\left(k + \frac{j}{h}\right) = 0$ for $k \notin \frac{1}{h}R_0$. Therefore,

$$\begin{aligned}\left\|\hat{u} - \widehat{I_\psi^h u}\right\|_{L^2(\mathbb{R})}^2 = \left\|\hat{u} - \widehat{I_\psi^h u}\right\|_{L^2(R)}^2 &= \sum_{j\in\mathbb{Z}} \int_{\frac{1}{h}R_0} |\hat{u}(k)|^2 \left|(\operatorname{rect} - \hat{\psi})(j+hk)\right|^2 \mathrm{d}k \\ &\leq \|\hat{u}\|_{L^\infty(\frac{1}{h}R_0)}^2 \int_R \left|(\operatorname{rect} - \hat{\psi})(\xi)\right|^2 \frac{\mathrm{d}\xi}{h}\end{aligned}$$

[vi]These assumptions can be weakened if we use a weaker form of the Poisson summation formula than Proposition 1.1.

where we can switch summation and integration by the monotone convergence theorem. Since the last term is finite, $\hat{u} - \widehat{I_\psi^h u} \in L^2(\mathbb{R})$ and the second estimate follows from Plancherel's identity (1.4).

The first inequality follows from the opposite estimate in the last step, i.e.

$$\left\| \hat{u} - \widehat{I_\psi^h u} \right\|_{L^2(\mathbb{R})}^2 \le \|\hat{u}\|_{L^2(\frac{1}{h}R_0)}^2 \max_{|k| \le \frac{\gamma}{2}} \sum_{j \in \mathbb{Z}} \left| \delta_{j0} - \hat{\psi}(j+k) \right|^2$$

and Plancherel's identity. □

If $(\psi_n)_{n\in\mathbb{N}}$ is a sequence of kernels with $\hat{\psi}_n \to \operatorname{rect}$ in $L^2(R)$, for example if $\psi_n \to \operatorname{sinc}$ in $L^2(\mathbb{R})$ where $\operatorname{sinc}(x) := \frac{\sin(\pi x)}{\pi x}$ is the (inverse) Fourier transform of rect, then by the second estimate, $I_{\psi_n}^h u \to u$ in L^2 when $n \to \infty$ for u satisfying the assumptions of Theorem 2.7. We will see that, under some further assumptions on $(\psi_n)_{n\in\mathbb{N}}$, we get exponential convergence. We will restrict ourselves to the second estimate; a similar analysis holds for the first.

Consider a set $\Psi \subseteq L^2(\mathbb{R})$ of compactly supported cardinal interpolation kernels. If there exists a c_0 such that for all $\nu \in \mathbb{N}$ and all $\psi \in \Psi$ with ψ of order ν,

$$\left\| \frac{\hat{\psi}^{(\nu)}}{\nu!} \right\|_{L^\infty(R_0)} \le c_0 \ , \tag{2.7}$$

then by Theorem 2.6 and a Taylor series expansion around 0,

$$\left\| 1 - \hat{\psi} \right\|_{L^\infty(R_0)} \le c_0 \left(\frac{\gamma}{2} \right)^\nu \ ,$$

and therefore

$$\left\| 1 - \hat{\psi} \right\|_{L^2(R_0)} \le c_0 \sqrt{\gamma} \left(\frac{\gamma}{2} \right)^\nu \ .$$

If we can bound the tails of $\hat{\psi}$ uniformly by

$$\left\| \hat{\psi} \right\|_{L^2(R \setminus R_0)} \le c_1 \left(\frac{\gamma}{2} \right)^{p(\nu)} \ , \tag{2.8}$$

then Theorem 2.7 implies

$$\left\| u - I_\psi^h u \right\|_{L^2(\mathbb{R})} \le \frac{c}{\sqrt{h}} \|\hat{u}\|_{L^\infty(\mathbb{R})} \left(\frac{\gamma}{2} \right)^{\min(\nu + \frac{1}{2}, p(\nu))} \tag{2.9}$$

for all u that satisfy the conditions in Theorem 2.7. If p is linear, this gives us exponential convergence for $\nu \to \infty$.

In particular, if (2.7) and (2.8) hold for $\gamma = 1$, we can set $h = 1$ and $\hat{u} = \operatorname{rect}$. In this situation, $u = \operatorname{sinc}$ and, since $I_\psi^1 \operatorname{sinc} = \psi$,[vii] (2.9) translates to

$$\|\operatorname{sinc} - \psi\|_{L^2(\mathbb{R})} \le c \left(\frac{1}{2} \right)^{\min(\nu + \frac{1}{2}, p(\nu))} \ . \tag{2.10}$$

Therefore, high-order cardinal interpolation kernels approximate sinc.

[vii] see the remark after Proposition 2.1

3 Z-splines

3.1 Definition

In this section, we will construct cardinal splines[viii] $Z_{m,q} \in C^{q-1}(\mathbb{R})$ with compact support $\operatorname{supp} Z_{m,q} \subseteq [-m, m]$. We will denote the corresponding interpolation operator by $I^h_{m,q} := I^h_{Z_{m,q}}$, so for $f \in C_{\text{pw}}(\mathbb{R})$,

$$\big(I^h_{m,q} f\big)(x) = \sum_{j\in\mathbb{Z}} f(jh) Z_{m,q}\Big(\frac{x}{h} - j\Big) \ , \tag{3.1}$$

and $I_m := I_{m,m}$. For the moment, we will set $h = 1$ and assume $f \in C^s(\mathbb{R})$ for large enough s.

Consider the Taylor series expansion of f about 0,

$$f(j) \approx \sum_{p=0}^{2m-2} \frac{f^{(p)}(0)}{p!} j^p \ , \quad j = -(m-1) : m-1 \ . \tag{3.2}$$

Up to order $2m-1$, the $2m-1$ values $\{f(j)\}_{j=-(m-1)}^{(m-1)}$ depend linearly on the $2m-1$ values $\{f^{(p)}(0)\}_{p=0}^{2m-2}$. By solving for the latter, we get

$$f^{(p)}(0) \approx \sum_{j=-(m-1)}^{m-1} a^{(m)}_{p,j} f(j) \ , \quad p = 0 : 2m-2 \tag{3.3}$$

for some $a^{(m)}_{p,j}$. Also, using $\operatorname{supp} Z_{m,q} \subseteq [-m, m]$ and the continuity of $Z_{m,q}$, (3.1) implies

$$\big(I^1_{m,q} f\big)^{(p)}(0) = \sum_{j=-(m-1)}^{m-1} Z^{(p)}_{m,q}(-j) f(j) \ , \quad p = 0 : q-1 \ . \tag{3.4}$$

We therefore require

$$Z^{(p)}_{m,q}(j) = \begin{cases} a^{(m)}_{p,-j}, & |j| \le m-1 \\ 0, & |j| \ge m \end{cases} \ , \quad p = 0 : q-1 \tag{3.5}$$

for all $j \in \mathbb{Z}$. Note that this defines q conditions for $Z_{m,q}$ at every $j \in \mathbb{Z}$, so for any interval $[j, j+1]$ there is a unique polynomial of degree $2q-1$ that satisfies (3.5) at j and $j+1$.

Definition 3.1 (Z-spline)**.** The *Z-spline* $Z_{m,q}$ of order $(m, q) \in \mathbb{N}^2$, $q \le 2m-1$, is the piecewise polynomial of degree $2q-1$ that satisfies (3.5) for all $j \in \mathbb{Z}$. The *standard Z-spline* is $Z_m := Z_{m,m}$.

The above construction can be generalized to nonequispaced nodes and bounded domains; see [4, Section 3].

[viii] i.e. piecewise polynomial cardinal interpolation kernels

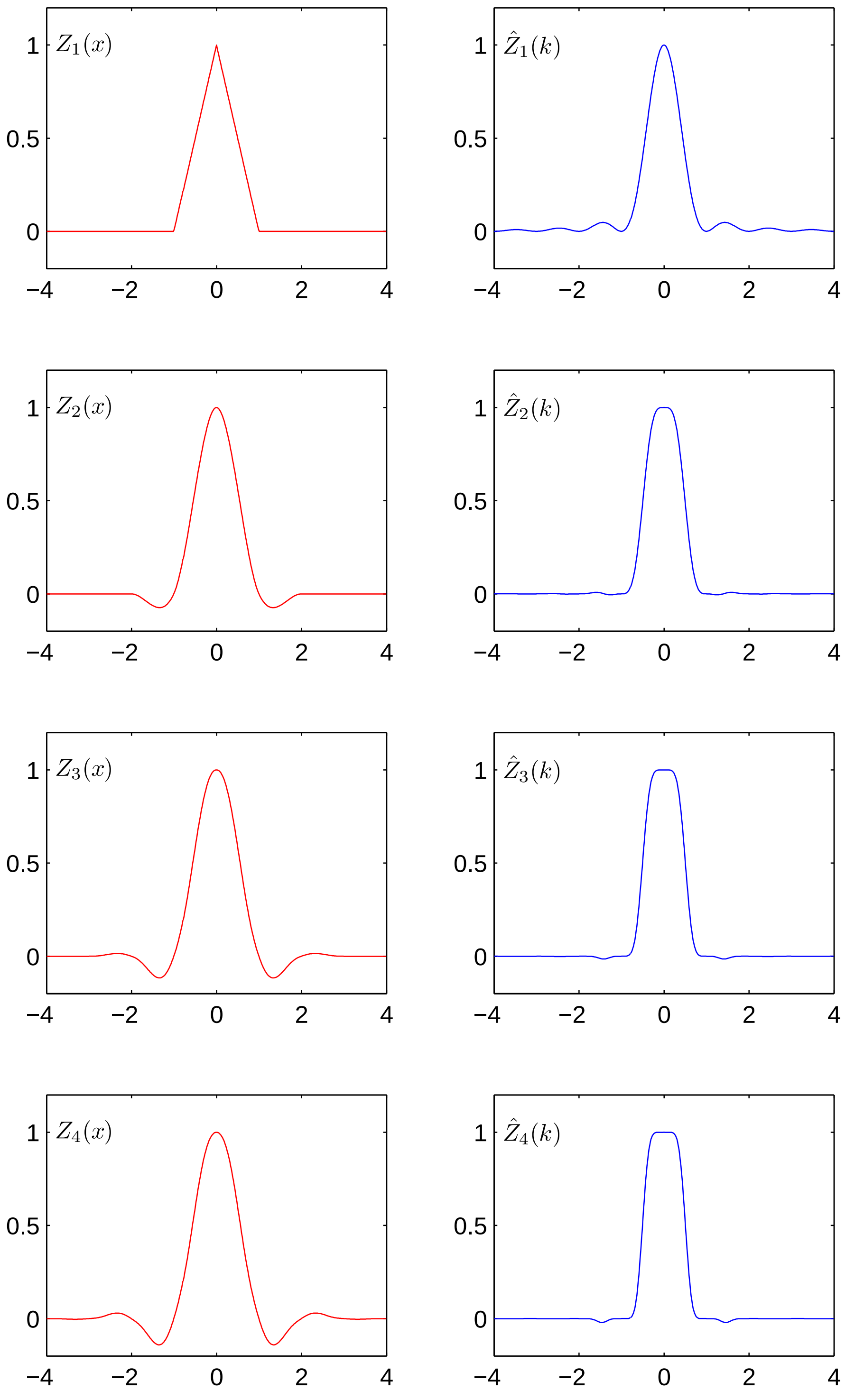

Figure 3.1: The first four standard Z-splines (left) and their Fourier transforms (right).

3.2 Construction in matrix form

For an explicit construction of the Z-spline $Z_{m,q}$, we need to translate the above arguments to matrix form. Let $F_m := \{f(j)\}_{j=-(m-1)}^{(m-1)}$ and $F'_m := \{f^{(p)}(0)\}_{p=0}^{2m-2}$. The Taylor series expansion (3.2) can be written as

$$F_m \approx V_m D_m F'_m \ , \tag{3.6}$$

where D_m is a diagonal matrix with $[D_m]_{l,l} = \frac{1}{(l-1)!}$ and V_m is a Vandermonde matrix with $[V_m]_{l,p} = (l-m)^{p-1}$. Therefore,

$$F'_m \approx A_m F_m \quad \text{for} \quad A_m := D_m^{-1} V_m^{-1} \ . \tag{3.7}$$

A_m is the *finite difference matrix*; it is related to $a_{p,j}^{(m)}$ by $[A_m]_{p+1,m+j} = a_{p,j}^{(m)}$. Using (3.5), we can evaluate $Z_{m,q}$ on $[j, j+1]$, $j \in \mathbb{Z}$, by Hermite-Birkhoff interpolation with

$$Z_{m,q}^{(p)}(i) = \begin{cases} [A_m]_{p+1,m-i}, & |i| \le m-1 \\ 0, & |i| \ge m \end{cases} \ , \quad p = 0 : q-1 \tag{3.8}$$

for $i \in \{j, j+1\}$.

There exist explicit formulas for $A_m = D_m^{-1} V_m^{-1}$. Clearly, D_m^{-1} is just the diagonal matrix with entries $[D_m^{-1}]_{l,l} = (l-1)!$. The inverse of the Vandermonde matrix V_m is

$$[V_m^{-1}]_{p,l} = \frac{(-1)^{l+1}}{(2m-1-l)!(l-1)!} v_{p,l} \ , \tag{3.9}$$

where $v_{p,l}$ is the coefficient of x^{l-1} in the polynomial

$$\frac{(x+m-1)(x+m-2)\cdots(x-(m-1))}{x+m-p} \ .$$

Example 3.2. The first few finite difference matrices are

$$A_1 = (1) \qquad A_2 = \begin{pmatrix} 0 & 1 & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} \\ 1 & -2 & 1 \end{pmatrix} \qquad A_3 = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ \frac{1}{12} & -\frac{2}{3} & 0 & \frac{2}{3} & -\frac{1}{12} \\ -\frac{1}{12} & \frac{4}{3} & -\frac{5}{2} & \frac{4}{3} & -\frac{1}{12} \\ -\frac{1}{2} & 1 & 0 & -1 & \frac{1}{2} \\ 1 & -4 & 6 & -4 & 1 \end{pmatrix}$$

See Section 5.1 for a MATLAB implementation of Z-splines.

3.3 Properties

Let us first summarize a few elementary properties of Z-splines.

Proposition 3.3. *Z-splines satisfy*

1. $Z_{m,q}$ *is unique*
2. $Z_{m,q} \in C^{q-1}(\mathbb{R})$ *and* $Z_{m,q}|_{(j,j+1)} \in C^{\infty}(j, j+1)$ *for all* $j \in \mathbb{Z}$
3. $\operatorname{supp} Z_{m,q} = [-m, m]$

4. $Z_{m,q} \in H^q(\mathbb{R})$

5. $Z_{m,q}(j) = \delta_{j0}$ *for* $j \in \mathbb{Z}$

6. $Z_{m,q}(-x) = Z_{m,q}(x)$

Proof. The first 3 properties are clear. Point 4 follows from properties 2 and 3 by the compatibility condition for Sobolev spaces. Point 5 follows from (3.2) for $j = 0$: $f(0) = f^{(0)}(0)$ implies $a_{0,j}^{(m)} = \delta_{j0}$ in (3.3).

To prove the symmetry of $Z_{m,q}$, i.e. property 6, we will show $Z_{m,q}^{(p)}(-j) = (-1)^p Z_{m,q}^{(p)}(j)$ for $p = 0 : q-1$. Note that, for $\mu(x) := -x$ and $f \in C^{q-1}$, $(f \circ \mu)^{(p)} = (-1)^p f^{(p)} \circ \mu$. If $f \in P_{2m-2}$, equation (3.2) is exact and therefore so is (3.3).

$$\sum_{j=-(m-1)}^{m-1} a_{p,j}^{(m)} f(j) = f^{(p)}(0) = (-1)^p (f \circ \mu)^{(p)}(0) = (-1)^p \sum_{j=-(m-1)}^{m-1} a_{p,j}^{(m)} f(-j)$$

for all $f \in P_{2m-2}$ implies $a_{p,-j}^{(m)} = (-1)^p a_{p,j}^{(m)}$ for $p = 0 : 2m-2$ and the claim follows from (3.5). $\square$

By Proposition 2.1, property 5 implies that $I_{m,q}^h$ is a cardinal interpolation operator, i.e. $(I_{m,q}^h f)(jh) = f(jh)$ for all $j \in \mathbb{Z}$.

Looking at the construction of Z-splines in Section 3.1, in particular the Taylor expansion in (3.2) and the interpolation requirement (3.5), we might predict a high order interpolation property. The following theorem states that the order of $Z_{m,q}$ is indeed $\min(2m-1, 2q)$, so Z-splines give us interpolation kernels of arbitrary order.

Theorem 3.4. *The interpolation operator* $I_{m,q}^h$ *is of order* $\min(2m-1, 2q)$.

Proof. First let $h = 1$. For $f \in P_{2m-2}$, equation (3.2) is exact and therefore so is (3.3). By (3.5), it follows that

$$\left(I_{m,q}^1 f\right)^{(p)}(j) = f^{(p)}(j)\ , \quad \forall j \in \mathbb{Z}\ ,\ p = 0 : q-1\ .$$

For $f \in P_{2q-1}$, since $f|_{[j,j+1]}, \left(I_{m,q}^1 f\right)|_{[j,j+1]} \in P_{2q-1}[j, j+1]$, this implies $f|_{[j,j+1]} = \left(I_{m,q}^1 f\right)|_{[j,j+1]}$. For general $h > 0$, the claim follows from Proposition 2.3. $\square$

Theorem 3.4 lets us apply results from Section 2 to get more concrete interpolation estimates.

Theorem 3.5. *Let* $\nu := \min(2m-1, 2q)$.

1. $\hat{Z}_{m,q}^{(n)}(k) = \delta_{k0}\delta_{n0}$ *for* $k \in \mathbb{Z}$ *and* $0 \le n \le \nu - 1$ *or* n *odd.*

2. *There is a constant* $c = c(\nu)$ *such that for all* $u \in C^\nu(\mathbb{R})$,

$$\left\| u - I_{m,q}^h u \right\|_\infty \le c h^\nu \left\| u^{(\nu)} \right\|_\infty\ .$$

3. *For all $s \le q$, there is a constant c such that for all $h > 0$ and all $u \in L^2(\mathbb{R})$ with* $\operatorname{supp} \hat{u} \subseteq \left[-\frac{1}{2h}, \frac{1}{2h}\right]$,

$$\left\| u - I^h_{m,q} u \right\|_{H^s(\mathbb{R})} \le c h^{\nu - s} \left\| u \right\|_{H^\nu(\mathbb{R})} \ .$$

Proof. 1. For $0 \le n \le \nu - 1$, this follows from Theorem 3.4. For n odd, it follows from the symmetry of Z-splines, Proposition 3.3.6.

2. This follows from Theorem 3.4 and Theorem 2.4.

3. This is a consequence of Theorem 3.4 and Theorem 2.6.

□

Note that, for a given m, choosing $q = m$ maximizes ν; there seems to be no gain in setting $q > m$. So for a given support, the interpolation order is highest for the standard Z-spline.

There is, however, at least a heuristic reason for setting $m > q$. In (3.3), we get a better approximation of $f^{(p)}$ for larger m. So, even though interpolation is only exact for polynomials of degree at most $2q - 1$, the derivatives of $I^h_{m,q} f$ at the nodes $h\mathbb{Z}$ are better approximations of the derivatives of f for arbitrary f if m is large. The numerical experiments in Section 6.1 show that it can make sense to use $I^h_{m,q}$ with $m > q$.

4 DFT Algorithm

4.1 Derivation and general formulation

We are interested in the fast evaluation of a 1-periodic trigonometric polynomial

$$f(x) = \sum_{k \in \mathbb{Z}_N} \hat{f}_k \, \mathrm{e}^{-2\pi \mathrm{i} k x} \tag{4.1}$$

at arbitrary $x_j \in \mathbb{R}$, $j = 1 : N$. For some $n \ge N$ and $\psi : \mathbb{R} \to \mathbb{R}$, we will approximate f by

$$s(x) := \sum_{l \in \mathbb{Z}} g_l \psi(nx - l) \ . \tag{4.2}$$

The evaluation of s is efficient if ψ has (small enough) compact support in $\mathbb{R}$; more precisely, we require $\operatorname{supp} \psi \subseteq [-m, m]$ for some $m \in \mathbb{N}$.

We first need to transform s to the frequency domain. We require s to be 1-periodic. This is the case iff $g_{l+rn} = g_l$ for all $r \in \mathbb{Z}$, $l \in \mathbb{Z}_n$. The Poisson summation formula, Proposition 1.1, implies

$$s(x) = \sum_{l \in \mathbb{Z}_n} g_l \sum_{r \in \mathbb{Z}} \psi(nx - l - nr) = \sum_{l \in \mathbb{Z}_n} g_l \frac{1}{n} \sum_{k \in \mathbb{Z}} \hat{\psi}\left(\frac{k}{n}\right) \mathrm{e}^{-2\pi \mathrm{i} k (x - l/n)} \ ,$$

where $\hat{\psi}$ is the continuous Fourier transform of ψ defined in (1.1). Note that, if we switch the order of summation and write $\mathrm{e}^{-2\pi \mathrm{i} k (x - l/n)} = \mathrm{e}^{2\pi \mathrm{i} k l/n} \, \mathrm{e}^{-2\pi \mathrm{i} k x}$, we get (1.9), the formula for the discrete Fourier transform. Let $(\hat{g}_k)_{k \in \mathbb{Z}_n}$ be the discrete Fourier transform of $(g_l)_{l \in \mathbb{Z}_n}$. If $\hat{g}_{k+rn}$, $r, n \in \mathbb{Z}$, is computed analogously, it is clear that $\hat{g}_{k+rn} = \hat{g}_k$. So if we set $\hat{\psi}_k := \hat{\psi}\left(\frac{k}{n}\right)$ for $k \in \mathbb{Z}$, somewhat abusing our notation, we get

$$s(x) = \sum_{k\in\mathbb{Z}} \hat{\psi}_k \hat{g}_k \, \mathrm{e}^{-2\pi \mathrm{i} kx} \ . \tag{4.3}$$

We can now let s approximate f by choosing appropriate $\hat{g}_k$. For more generality, we'll first consider an intermediate approximation

$$s_1(x) := \sum_{k\in\mathbb{Z}} \hat{\varphi}_k \hat{g}_k \, \mathrm{e}^{-2\pi \mathrm{i} kx} \ , \tag{4.4}$$

where $\hat{\varphi}_k \approx \hat{\psi}_k$ and $\hat{\varphi}_k \neq 0$ for $k \in \mathbb{Z}_N$. For s_1 to approximate f, we set

$$\hat{g}_k := \begin{cases} \frac{\hat{f}_k}{\hat{\varphi}_k}, & k \in \mathbb{Z}_N \\ 0, & k \in \mathbb{Z}_n \backslash \mathbb{Z}_N \end{cases} \ . \tag{4.5}$$

This gives us

$$f(x) \approx s_1(x) \approx s(x) = \sum_{l=\lceil nx\rceil - m}^{\lfloor nx \rfloor + m} g_l \psi(nx - l) \ . \tag{4.6}$$

These considerations lead to the following algorithm for the approximate evaluation of f at arbitrary x_j. Note that we have not yet chosen $\hat{\varphi}_k$.

Algorithm 4.1. Parameters: $a \in [1, \infty)$, $m \in \mathbb{N}$.
Input: $N \in \mathbb{N}$, $n := aN$, $x_j \in \mathbb{R}$ $(j = 1 : \tilde{N})$, $\hat{f}_k \in \mathbb{C}$ $(k \in \mathbb{Z}_N)$.

0. (Precomputation) calculate $\hat{\varphi}_k$ for $k \in \mathbb{Z}_N$ and evaluate $\psi(nx_j - l)$ for $l = \lceil nx_j \rceil - m : \lfloor nx_j \rfloor + m$, $j = 1 : \tilde{N}$.

1. For $k \in \mathbb{Z}_N$ compute
$$\hat{g}_k := \frac{\hat{f}_k}{\hat{\varphi}_k}$$
and set $\hat{g}_k := 0$ for $k \in \mathbb{Z}_n \backslash \mathbb{Z}_N$.

2. Calculate
$$g_l := \sum_{k\in\mathbb{Z}_N} \hat{g}_k \, \mathrm{e}^{-2\pi \mathrm{i} kl/n}$$
by reduced FFT for $l \in \mathbb{Z}_n$.

3. For $j = 1 : \tilde{N}$ compute[ix]
$$s(x_j) = \sum_{l=\lceil nx\rceil - m}^{\lfloor nx \rfloor + m} g_l \psi(nx_j - l) \ .$$

[ix]The index in g_l is taken modulo n.

Note that Algorithm 4.1 evaluates f at an arbitrary number $\tilde{N}$ of points. For simplicity, we will assume $\tilde{N} = N$, i.e. that the number of evaluation points coincides with the length of the trigonometric polynomial f.

Clearly, step 1 requires just N multiplications and step 3 uses at most $(2m+1)N$ multiplications and additions. The FFT in step 2 has complexity $O(n \log n)$, so the overall complexity of the algorithm is $O(n \log n + mN)$. Introducing the oversampling factor

$$a := \frac{n}{N} \tag{4.7}$$

to eliminate n, we can write the complexity of Algorithm 4.1 as

$$O(aN \log N + (a \log a + m)N) \ . \tag{4.8}$$

In step 0, which we consider to be precomputation, we need up to $(2m+1)N$ evaluations of ψ; we also need to compute $\hat{\varphi}_k$ for N values of k. Both calculations are clearly independent of f; however, the former depends on x_j, so one cannot always consider this to be precomputation. If the evaluation of ψ requires m^β time, this adds a factor $O(m^{\beta+1}N)$ to the total complexity.

If the evaluation of ψ is considered precomputation, Algorithm 4.1 requires $O(aN)$ memory for the Fourier transform (see [1, Section 7.2]) and (at least) $O(mN)$ to evaluate ψ, since all of the evaluations are done beforehand and stored. However, if ψ is evaluated in step 3, the memory usage can be reduced to $O(N+m)$ for the evaluation; see Code 5.5 for details.

4.2 Matrix form

It will be useful to formulate the above algorithm in matrix form. If $\hat{\mathrm{f}} := (\hat{f}_k)_{k\in\mathbb{Z}_N} \in \mathbb{C}^N$ and $\mathrm{f} := (f(x_j))_{j=1:N} \in \mathbb{C}^N$, then (4.1) simply states

$$\mathrm{f} = A\hat{\mathrm{f}} \quad \text{for} \quad A \in \mathbb{C}^{N\times N} \ , \quad A_{jk} = \mathrm{e}^{-2\pi \mathrm{i} k x_j} \ . \tag{4.9}$$

Algorithm 4.1 approximates this matrix-vector multiplication by

$$\mathrm{f} = A\hat{\mathrm{f}} \approx B_\psi F_n D_{\hat{\varphi}} \hat{\mathrm{f}} \ , \tag{4.10}$$

where

$$\begin{aligned}
D_{\hat{\varphi}} &\in \mathbb{C}^{n\times N} \ , & [D_{\hat{\varphi}}]_{ik} &:= \begin{cases} \frac{1}{\hat{\varphi}_k}, & i = k + \lfloor \frac{n}{2} \rfloor - \lfloor \frac{N}{2} \rfloor \\ 0, & \text{otherwise} \end{cases} \ , \\
F_n &\in \mathbb{C}^{n\times n} \ , & [F_n]_{lk} &:= \mathrm{e}^{-2\pi \mathrm{i} kl/n} \ , \\
B_\psi &\in \mathbb{C}^{N\times n} \ , & [B_\psi]_{jl} &:= \psi(n x_j - l) \ .
\end{aligned}$$

$D_{\hat{\varphi}}$ is essentially a diagonal matrix containing the factors $\frac{1}{\hat{\varphi}_k}$, F_n is the Fourier matrix for the inverse discrete Fourier transform (1.11) of length n, and B_ψ contains all the necessary evaluations of ψ.

4.3 An error estimate

Theorem 4.2 gives bounds for the error in the max-norm and the L^2-norm. We will use the following norms for f:

$$\|\hat{\mathrm{f}}\|_{l^1} := \sum_{k\in\mathbb{Z}_N} \left|\hat{f}_k\right| \quad \text{and} \quad \|f\|_{L^2} := \int_{-1/2}^{1/2} |f(x)|^2 \,\mathrm{d}x \ . \tag{4.11}$$

Theorem 4.2. *For the norms defined above,*

$$\sup_{x\in\mathbb{R}} |f(x)-s(x)| \le \|\hat{\mathrm{f}}\|_{l^1} \max_{k\in\mathbb{Z}_N} \left(\left| \frac{\hat{\varphi}_k - \hat{\psi}_k}{\hat{\varphi}_k} \right| + \sum_{r\in\mathbb{Z}\setminus\{0\}} \left| \frac{\hat{\psi}_{k+rn}}{\hat{\varphi}_k} \right| \right)$$

and

$$\|f(x)-s(x)\|_{L^2}^2 \le \|f\|_{L^2}^2 \max_{k\in\mathbb{Z}_N} \left(\left| \frac{\hat{\varphi}_k - \hat{\psi}_k}{\hat{\varphi}_k} \right|^2 + \sum_{r\in\mathbb{Z}\setminus\{0\}} \left| \frac{\hat{\psi}_{k+rn}}{\hat{\varphi}_k} \right|^2 \right) .$$

Proof.

$$\begin{aligned} f(x)-s(x) &\overset{(4.3)}{=} \sum_{k\in\mathbb{Z}_N} \hat{f}_k \,\mathrm{e}^{-2\pi \mathrm{i}kx} - \sum_{k\in\mathbb{Z}} \hat{\psi}_k \hat{g}_k \,\mathrm{e}^{-2\pi \mathrm{i}kx} \\ &\overset{(4.5)}{=} \sum_{k\in\mathbb{Z}_N} \hat{f}_k \,\mathrm{e}^{-2\pi \mathrm{i}kx} - \sum_{k\in\mathbb{Z}_N} \frac{\hat{f}_k}{\hat{\varphi}_k} \sum_{r\in\mathbb{Z}} \hat{\psi}_{k+rn} \,\mathrm{e}^{-2\pi \mathrm{i}(k+rn)x} \\ &= \sum_{k\in\mathbb{Z}_N} \hat{f}_k \left(\frac{\hat{\varphi}_k - \hat{\psi}_k}{\hat{\psi}_k} \,\mathrm{e}^{-2\pi \mathrm{i}kx} + \sum_{r\in\mathbb{Z}\setminus\{0\}} \frac{\hat{\psi}_{k+rn}}{\hat{\varphi}_k} \,\mathrm{e}^{-2\pi \mathrm{i}(k+rn)x} \right) . \end{aligned}$$

The first inequality now follows from the triangle inequality.

By Parseval's theorem (1.8), taking Fourier coefficients from the above calculation,

$$\|f-s\|_{L^2} = \sum_{k\in\mathbb{Z}_N} |\hat{f}_k|^2 \left(\left| \frac{\hat{\varphi}_k - \hat{\psi}_k}{\hat{\varphi}_k} \right|^2 + \sum_{r\in\mathbb{Z}\setminus\{0\}} \left| \frac{\hat{\psi}_{k+rn}}{\hat{\varphi}_k} \right|^2 \right) .$$

The second inequality now follows like the first, since $\|f\|_{L^2}^2 = \sum_k |\hat{f}_k|^2$ by (1.8). □

In [5], G. Steidl uses a different approach. She splits the error into two components,

$$|f(x)-s(x)| \le |f(x)-s_1(x)| + |s_1(x)-s(x)| ,$$

and estimates these separately. This has the advantage that the estimate is independent of $\hat{\psi}$, which is important, for example, when ψ is the truncation of some function φ and $\hat{\varphi}_k = \hat{\varphi}\left(\frac{k}{n}\right)$. In this situation, $\hat{\varphi}$ may be much simpler than $\hat{\psi}$. In general, however, it seems unnatural to introduce $\hat{\varphi}_k$ into the estimate for $k \notin \mathbb{Z}_N$, since these values do not appear in Algorithm 4.1.

4.4 A natural choice for $\hat{\varphi}_k$

It is natural to choose $\hat{\varphi}_k := \hat{\psi}_k$ for $k \in \mathbb{Z}_N$. In this situation, we can simplify the estimates in Theorem 4.2 to the following:

Corollary 4.3. *If $\hat{\varphi}_k = \hat{\psi}_k$ for $k \in \mathbb{Z}_N$,*

$$\sup_{x\in\mathbb{R}} |f(x)-s(x)| \le \|\hat{\mathrm{f}}\|_{l^1} \max_{|k|\le\frac{1}{2a}} \sum_{r\in\mathbb{Z}\setminus\{0\}} \left| \frac{\hat{\psi}(k+r)}{\hat{\psi}(k)} \right|$$

and

$$\|f(x)-s(x)\|_{L^2}^2 \le \|f\|_{L^2}^2 \max_{|k|\le\frac{1}{2a}} \sum_{r\in\mathbb{Z}\setminus\{0\}} \left|\frac{\hat\psi(k+rn)}{\hat\psi(k)}\right|^2 .$$

Proof. Set $\hat\varphi_k = \hat\psi_k = \hat\psi\left(\frac{k}{n}\right)$ in Theorem 4.2 and use $\frac{1}{n}\mathbb{Z}_N \subseteq \left[-\frac{1}{2a}, \frac{1}{2a}\right]$. □

Consider Algorithm 4.1 for a sequence $(\psi_m)_{m\in\mathbb{N}}$ with $\operatorname{supp}\psi_m \subseteq [-m,m]$. Let's assume that we can estimate $\hat\psi_m$ by

$$\left|\hat\psi_m(k+r)\right| \le (c(a)r)^{-p(m)} \quad \text{for} \quad |k| \le \frac{1}{2a}\ ,\ r\in\mathbb{Z}\setminus\{0\}\ , \tag{4.12}$$

where p and c are some functions with $p(m)>1$. If $\min_{|k|\le\frac{1}{2a}}\left|\hat\psi_m(k)\right| \ge c_0$, it follows that

$$\sum_{r\in\mathbb{Z}\setminus\{0\}} \left|\frac{\hat\psi_m(k+r)}{\hat\psi_m(k)}\right| \le \frac{2}{c_0} c(a)^{-p(m)} \sum_{r=1}^{\infty} r^{-p(m)} \le \frac{2}{c_0} c(a)^{-p(m)} \frac{p(m)}{p(m)-1}\ ,$$

so if $p(m) \ge p_0 > 1$ and $\min_{|k|\le\frac{1}{2}}\left|\hat\psi_m(k)\right| \ge c_0$,

$$|f(x)-s(x)| \le C\|\hat{\mathrm{f}}\|_{l^1} c(a)^{-p(m)} \tag{4.13}$$

with $C = C(c_0,p_0) = \frac{2p_0}{c_0(p_0-1)}$. In the linear case, (4.12) and (4.13) simplify to

$$\left|\hat\psi_m(k+r)\right| \le (\eta a r)^{-\gamma m} \quad \Longrightarrow \quad |f(x)-s(x)| \le C\|\hat{\mathrm{f}}\|_{l^1}(\eta a)^{-\gamma m} \tag{4.14}$$

for k and r as in (4.12). We can therefore expect exponential convergence in m for any large enough a, with a convergence rate roughly similar to $\log a$.

Of course, a similar estimate holds for the term in the second inequality of Corollary 4.3, and the analogous argument gives us the same convergence rate in the L^2-norm if $p_0 > \frac{1}{2}$.

Note that the constants in Corollary 4.3 and estimate (4.13) only depend on properties of $(\hat\psi_m)_{m\in\mathbb{N}}$ and the parameters m and a. In particular, they are independent of N, apart from a hidden N in $\|\hat{\mathrm{f}}\|_{l^1}$.

4.5 Another possibility: interpolation

If the x_j are almost equidistant, that is if $\operatorname{dist}(x_j, \frac{1}{n}\mathbb{Z}_n)$ is small, it makes sense to choose $\hat\varphi_k$ such that $s(\frac{i}{n}) = f(\frac{i}{n})$ for $i\in\mathbb{Z}$. Note that this dictates a value for n.

If $\hat\varphi_k$ are the coefficients of a trigonometric polynomial

$$\varphi(x) := \sum_{k\in\mathbb{Z}_n} \frac{\hat\varphi_k}{n} \,\mathrm{e}^{-2\pi\mathrm{i}kx/n}\ , \tag{4.15}$$

then $s_1 = f$, since

$$s_1(x) \overset{(4.4)}{=} \sum_{k\in\mathbb{Z}_n} \hat\varphi_k \hat g_k \,\mathrm{e}^{-2\pi\mathrm{i}kx} \overset{(4.5)}{=} \sum_{k\in\mathbb{Z}_N} \hat f_k \,\mathrm{e}^{-2\pi\mathrm{i}kx} = f(x)\ . \tag{4.16}$$

We can transform s_1 to the time domain by

$$\begin{aligned} s_1(x) &\overset{(4.4)}{=} \sum_{k\in\mathbb{Z}_n} \hat{\varphi}_k \hat{g}_k \,\mathrm{e}^{-2\pi \mathrm{i} kx} \overset{(1.9)}{=} \sum_{k,l\in\mathbb{Z}_n} \frac{\hat{\varphi}_k}{n} g_l \,\mathrm{e}^{-2\pi \mathrm{i} k(x-l/n)} \\ &\overset{(4.15)}{=} \sum_{l\in\mathbb{Z}_n} g_l \varphi(nx-l) \;. \end{aligned} \tag{4.17}$$

If we set

$$\hat{\varphi}_k := \sum_{l\in\mathbb{Z}_n} \left(\sum_{r\in\mathbb{Z}} \psi(l-nr) \right) \mathrm{e}^{2\pi \mathrm{i} kl/n} \tag{4.18}$$

for $k \in \mathbb{Z}_n$, then $\varphi(x)$ interpolates $\sum_{r\in\mathbb{Z}} \psi(x-nr)$ since

$$\varphi(i) \overset{(4.15)}{=} \sum_{k,l\in\mathbb{Z}_n} \frac{1}{n} \left(\sum_{r\in\mathbb{Z}} \psi(l-nr) \right) \mathrm{e}^{2\pi \mathrm{i} k(l-i)/n} \overset{(1.10)}{=} \sum_{r\in\mathbb{Z}} \psi(i-nr) \tag{4.19}$$

for $i \in \mathbb{Z}$. Therefore,

$$s\left(\frac{i}{n}\right) = \sum_{l\in\mathbb{Z}_n} g_l \sum_{r\in\mathbb{Z}} \psi(i-l-nr) = \sum_{l\in\mathbb{Z}_n} g_l \varphi(i-l) = s_1\left(\frac{i}{n}\right) = f\left(\frac{i}{n}\right) \;. \tag{4.20}$$

4.6 Algorithm 4.1 with cardinal interpolation

Let ψ be a cardinal interpolation kernel as in Section 2, for example a Z-spline defined in Section 3.1, and let s interpolate f, i.e. $s = I_\psi^{1/n} f$.

Comparing (4.2) to (2.1), we see that $g_l = f\left(\frac{l}{n}\right)$. Since $(\hat{g}_k)_{k\in\mathbb{Z}_n}$ is the discrete Fourier transform of $(g_l)_{l\in\mathbb{Z}_n}$, $\hat{g}_k = \hat{f}_k$ for $k \in \mathbb{Z}_N$ and zero otherwise. Equivalently, since $\psi(j) = \delta_{j0}$ for $j \in \mathbb{Z}$ by Proposition 2.1, equation (4.18) implies that setting $\hat{\varphi}_k = 1$ for $k \in \mathbb{Z}_n$ ensures that s interpolates f.

By these considerations, we can simplify Algorithm 4.1 using cardinal interpolation.

Algorithm 4.4. Parameters: $a \in [1,\infty)$, $m \in \mathbb{N}$.
Input: $N \in \mathbb{N}$, $n := aN$, $x_j \in \mathbb{R}$ $(j = 1:N)$, $\hat{f}_k \in \mathbb{C}$ $(k \in \mathbb{Z}_N)$.

0. (Precomputation) evaluate $\psi(nx_j - l)$ for $l = \lceil nx_j \rceil - m : \lfloor nx_j \rfloor + m$, $j = 1:N$.

1. Calculate
$$g_l := \sum_{k\in\mathbb{Z}_N} \hat{f}_k \,\mathrm{e}^{-2\pi \mathrm{i} kl/n}$$
by reduced FFT for $l \in \mathbb{Z}_n$.

2. For $j = 1:N$ compute[x]
$$s(x_j) = \sum_{l=\lceil nx \rceil - m}^{\lfloor nx \rfloor + m} g_l \psi(nx_j - l) \;.$$

[x]The index in g_l is taken modulo n.

The following Corollary specializes the error estimates in Theorem 4.2 to the case of cardinal interpolation. Note the similarities to the estimates in Theorem 2.7.

Corollary 4.5. *If $\hat{\varphi}_k = 1$ for all $k \in \mathbb{Z}_N$,*

$$\sup_{x\in\mathbb{R}} |f(x) - s(x)| \le \|\hat{\mathrm{f}}\|_{l^1} \max_{|k|\le\frac{1}{2a}} \sum_{r\in\mathbb{Z}} \left|\delta_{r0} - \hat{\psi}(k+r)\right|$$

and

$$\|f(x) - s(x)\|_{L^2}^2 \le \|f\|_{L^2}^2 \max_{|k|\le\frac{1}{2a}} \sum_{r\in\mathbb{Z}} \left|\delta_{r0} - \hat{\psi}(k+r)\right|^2 .$$

Proof. Set $\hat{\varphi}_k = 1$ in Theorem 4.2 and use $\frac{1}{n}\mathbb{Z}_N \subseteq \left[-\frac{1}{2a}, \frac{1}{2a}\right]$. □

By Theorem 2.6, each of the terms in the first sum is asymptotically bounded by $a^{-\nu}$ if ψ is of order ν. However, this is not enough to bound the entire sum. Note nevertheless that the estimates in Theorem 4.5 are independent of N. See Section 6.2 for numerical calculations of these error estimates for Z-splines.

4.7 Dual algorithm

In some situations, we may need to evaluate (4.1) with equidistant nodes in time but nonequispaced nodes in the frequency domain. In this section, we show how to use the above considerations on the evaluation of (4.1) to evaluate

$$f_j = \sum_{k=1}^{N} \hat{f}(\xi_k)\, \mathrm{e}^{-2\pi \mathrm{i} j\xi_k} \tag{4.21}$$

at $j \in \mathbb{Z}_N$. For $\hat{\mathrm{f}} := (\hat{f}(\xi_k))_{k=1:N} \in \mathbb{C}^N$ and $\mathrm{f} := (f_j)_{j\in\mathbb{Z}_N} \in \mathbb{C}^N$, using the matrices introduced in Section 4.2, (4.21) relates f and $\hat{\mathrm{f}}$ by

$$\mathrm{f} = A^{\mathrm{T}}\hat{\mathrm{f}} . \tag{4.22}$$

If we approximate A as in Section 4.2,

$$\mathrm{f} = A^{\mathrm{T}}\hat{\mathrm{f}} \approx D_{\hat{\varphi}}^{\mathrm{T}} F_n B_{\psi}^{\mathrm{T}} \hat{\mathrm{f}} , \tag{4.23}$$

we get the following algorithm for the efficient evaluation of (4.21):

Algorithm 4.6. Parameters: $a \in [1,\infty)$, $m \in \mathbb{N}$.
Input: $N \in \mathbb{N}$, $n := aN$, $\xi_k \in \mathbb{R}$, $\hat{f}(\xi_k) \in \mathbb{C}$ $(k \in \mathbb{Z}_N)$.

0. (Precomputation) calculate $\hat{\varphi}_j$ for $j \in \mathbb{Z}_N$ and evaluate $\psi(n\xi_k - l)$ for $l = \lceil n\xi_k\rceil - m : \lfloor n\xi_k\rfloor + m$, $k = 1 : N$.

1. For $l \in \mathbb{Z}_n$ compute

$$h_l := \sum_{k=1}^{N} \hat{f}(\xi_k) \sum_{r\in\mathbb{Z}} \psi(n\xi_k - l - nr) .$$

2. Calculate by FFT of length n for $j \in \mathbb{Z}_n$

$$\hat{h}_j := \sum_{l \in \mathbb{Z}_n} h_l \, \mathrm{e}^{-2\pi \mathrm{i} jl/n} \ .$$

3. For $j \in \mathbb{Z}_N$ compute

$$f_j \approx s_j = \frac{\hat{h}_j}{\hat{\varphi}_j} \ .$$

Apparently, the steps in Algorithm 4.6 are similar to those in Algorithm 4.1 in reverse order. The two are, in a sense, dual. They clearly have the same complexity, $O(aN \log N + (a \log a + m)N)$. We will see that they share several other properties.

First of all, Algorithm 4.6 is also particularly simple in the case of cardinal interpolation. If $I_\psi^{1/n}$ is the cardinal interpolation operator (2.1) with kernel ψ and grid length $\frac{1}{n}$, and $\hat{\varphi}_j = 1$ for all $j \in \mathbb{Z}_N$ as in Section 4.6, then

$$s_j = \sum_{k=1}^{N} \hat{f}(\xi_k) \sum_{l \in \mathbb{Z}} \psi(n\xi_k - l) \, \mathrm{e}^{-2\pi \mathrm{i} jl/n} = \sum_{k=1}^{N} \hat{f}(\xi_k) \Big(I_\psi^{1/n} \, \mathrm{e}^{-2\pi \mathrm{i} j \cdot} \Big)(\xi_k) \ . \quad (4.24)$$

So, in this situation, we are evaluating (4.21) simply by interpolating the exponential function using equispaced nodes.

Furthermore, the first error estimate in Theorem 4.2 also applies to Algorithm 4.6.

Theorem 4.7. *For all $j \in \mathbb{Z}_N$,*

$$|f_j - s_j| \le \|\hat{\mathrm{f}}\|_{l^1} \left(\left| \frac{\hat{\varphi}_j - \hat{\psi}_j}{\hat{\varphi}_j} \right| + \sum_{r \in \mathbb{Z} \setminus \{0\}} \left| \frac{\hat{\psi}_{j+rn}}{\hat{\varphi}_j} \right| \right) \ .$$

Proof. Inserting the definitions of $\hat{h}_j$ and h_l into the last step of Algorithm 4.6, we get

$$\begin{aligned} s_j &= \frac{1}{\hat{\varphi}_j} \sum_{l \in \mathbb{Z}} \sum_{k=1}^{N} \hat{f}(\xi_k) \psi(n\xi_k - l) \, \mathrm{e}^{-2\pi \mathrm{i} jl/n} \\ &= \sum_{k=1}^{N} \hat{f}(\xi_k) \, \mathrm{e}^{-2\pi \mathrm{i} j \xi_k} \left(\frac{1}{\hat{\varphi}_j} \sum_{l \in \mathbb{Z}} \psi(n\xi_k - l) \, \mathrm{e}^{2\pi \mathrm{i} j(\xi_k - l/n)} \right) \ . \end{aligned}$$

By the Poisson summation formula, Proposition 1.1,

$$\sum_{l \in \mathbb{Z}} \psi(n\xi_k - l) \, \mathrm{e}^{2\pi \mathrm{i} j(\xi_k - l/n)} = \sum_{r \in \mathbb{Z}} \hat{\psi}\left(\frac{j}{n} + r \right) \mathrm{e}^{-2\pi \mathrm{i} rn\xi_k} \ .$$

The claim follows from abbreviating $\hat{\psi}_{j+rn} = \hat{\psi}\big(\frac{j}{n} + r\big)$. $\square$

Therefore, arguments analogous to those in Section 4.4 suggest exponential convergence of Algorithm 4.6 at the same rate as Algorithm 4.1 if $\hat{\varphi}_j = \hat{\psi}_j$.

5 MATLAB implementation

In this section, we look more closely at a MATLAB implementation of our DFT algorithm with Z-splines. We first discuss algorithms for constructing and evaluating Z-splines, then go over to our implementations of Algorithms 4.1 and 4.6 with Z-splines.

5.1 Construction and evaluation of Z-splines

Since we defined Z-splines (Definition 3.1) through their derivatives at the interpolation nodes, it is natural to use Hermite-Birkhoff (fully Hermite) interpolation to evaluate them. Indeed, we use divided differences to carry out the Hermite Birkhoff interpolation and calculate coefficients necessary for the actual evaluation, which is done later using Horner's backwards recursion algorithm.

The following code calculates the coefficients for a Z-spline and stores them in a suitable structure.

Code 5.1. `function zs = zsmake(mq)`
```
%ZSMAKE calculate coefficients for Z-spline
%   zs = ZSMAKE([m q]) or zs = ZSMAKE(m) (default q=m) returns a structure
%   zs, which can be evaluated at x by zs.val(x) or zs.vali(x,intx).  See
%   ZSEVAL and ZSEVALINTX for details.

% use standard Z-spline (q=m) if no q is given; q denotes the number of
% function values/derivatives that are used to evaluate Z-splines.  Note
% that a Z-spline is a piecewise polynomial of degree 2*q-1 and is q-1
% times differentiable.
m = mq(1);
q = mq(length(mq));

% initialize Z-spline structure.  zs.cc(j+m+1,:) are the polynomial
% coefficients for the Z-spline in [j,j+1) in the Newton divided
% differences diagram, j=-m:m-1.
zs.m = m;
zs.q = q;
zs.cc = zeros(2*m,2*q);

% calculate finite difference matrix
A = zeros(q,2*m+1);
A(:,2*m:-1:2) = fdmat(m,q);

% on every interval [j,j+1), j=-m:m-1, use hermite interpolation to
% calculate coefficients of Z-spline.
for j=-m:m-1

% N is a Newton diagram containing divided differences
    N = zeros(2*q);
    N(1:q,1) = A(1,m+j+1);
    N(q+1:2*q,1) = A(1,m+j+2);

% calculate entries in Newton diagram
    for k=2:q
        N(1:q-k+1,k) = A(k,m+j+1)/factorial(k-1);
        N(q+1:2*q-k+1,k) = A(k,m+j+2)/factorial(k-1);
        for i=q-k+2:q
           N(i,k) = (N(i+1,k-1)-N(i,k-1));
        end
    end
    for k=q+1:2*q
        for i=1:2*q-k+1
            N(i,k) = (N(i+1,k-1)-N(i,k-1));
        end
    end

% the coefficients are in the first row
    zs.cc(j+m+1,:) = N(1,:);

end

% define evaluation algorithms
zs.val = @(x) zseval(zs,x);
zs.vali = @(x,intx) zsevalintx(zs,x,intx);
```

In Code 5.1, a function `fdmat` is called, which calculates the finite difference matrix as described in Section 3.2. We include this straightforward code mainly for completeness.

Code 5.2. `function A = fdmat(m,q)`

```
%FDMAT calculate finite difference matrix
%   A = FDMAT(m,q) returns the first q rows of the mth finite difference
%   matrix.
%
% The bulk of the effort goes into calculating coefficients of given
% polynomials; these are stored in the variable U. The inverse of the
% Vandermonde matrix is W.*U, where W stores certain factors.  The finite
% difference matrix is then given by A=D*(W.*U) for a diagonal matrix D.

% use default (maximal) value for q if none is given
if nargin==1 || q<=0 || q>=2*m
    q = 2*m-1;
end

% the pth column of the Vandermonde matrix is s.^(p-1)
s = -m+1:m-1;

% initialize polynomials as a cell array
P = cell(2*m-1,1);
for k = 1:2*m-1
    P{k} = 1;
end

% calculate coefficients of polynomials
for p = 1:2*m-1
    for k = [1:p-1,p+1:2*m-1]
        P{k} = conv(P{k},[1,-s(p)]);
    end
end
```

```
% assemble matrix U (U contains coefficients of the polynomials P)
U = zeros(2*m-1);
for k = 1:2*m-1
    U(2*m-1:-1:1,k) = P{k}';
end

% construct W
w = (-1).^(2:2*m)./(factorial(2*m-2:-1:0).*factorial(0:2*m-2));
W = w(ones(q,1),:);

% calculate finite difference matrix
D = diag(factorial(0:q-1));
A = D*(W.*U(1:q,:));
```

As mentioned above, the actual evaluation of a Z-spline is done with Horner's algorithm on every interval `[j,j+1)`. Before we can apply this, however, we need to calculate in which of these intervals the argument `x` lies, or, if `x` is a matrix, in which interval each of its elements lies.

Code 5.3.
```
function y = zseval(zs, x)
%ZSEVAL evaluate Z-spline zs at x
%   y = ZSEVAL(zs,x) where zs is a structure created by ZSMAKE evaluates
%   the Z-spline zs at the elements of x.
%
%   Note that both ZSEVAL and ZSEVALINTX evaluate Z-splines.  The
%   difference is that ZSEVAL calculates intx while ZSEVALINTX takes it as
%   an argument.  The former is more flexible; the latter is a bit faster,
%   but meant for use only in ADFTZ.

% intx denotes which x need to be evaluated on which interval.
%       intx{j+m+1} = {i; floor(x(i)) = j} .
intx = cell(1,2*zs.m);
for i=1:numel(x)
    j = floor(x(i));
    if (-zs.m<=j) && (j<zs.m)
        intx{j+zs.m+1} = [intx{j+zs.m+1},i];
    end
end

% initialize solution vector y
y = zeros(size(x));

% on each interval [j,j+1),j=-m:m-1, evaluate Z-spline at x(intx{j+m+1}).
for j=-zs.m:zs.m-1
    P = [j*ones(1,zs.q), (j+1)*ones(1,zs.q)];
    y(intx{j+zs.m+1}) = zs.cc(j+zs.m+1,2*zs.q);
    for p=2*zs.q-1:-1:1
        y(intx{j+zs.m+1}) = y(intx{j+zs.m+1}).*(x(intx{j+zs.m+1})-P(p))...
            + zs.cc(j+zs.m+1,p);
    end
end
```

5.2 Implementation of Algorithm 4.1

Before looking at our MATLAB implementation of Algorithm 4.1, let's reformulate it in a manner more accessible to implementation.

Algorithm 5.4.

```
  **function** ADFT$(f, x, a, m, \hat\varphi)$
      $N \leftarrow \text{length}(f)$
      $n \leftarrow aN$
      $g \leftarrow \text{fft}\left(\frac{f}{\hat\varphi}\right)$        $\triangleright\ g_l := \sum_{k\in\mathbb{Z}_N} \frac{\hat f_k}{\hat\varphi_k}\, \mathrm{e}^{-2\pi \mathrm{i} kl/n}$ for $l \in \mathbb{Z}_n$
      **for** $j \leftarrow 1$ **to** $N$ **do**
          $X \leftarrow nx(j) - \lfloor nx(j)\rfloor - (-m:m)$        $\triangleright$ evaluation points of $\psi$
          $G \leftarrow g(\lfloor nx(j)\rfloor + (-m:m) \mod n)$
          $s(j) \leftarrow \langle G, \psi(X)\rangle$        $\triangleright\ s(x_j) := \sum_{l=\lceil nx\rceil - m}^{\lfloor nx\rfloor + m} g_l \psi(nx_j - l)$
      **end for**
      **return** $s$
  **end function**
```

Code 5.5 is an implementation of Algorithm 4.1 similar to Algorithm 5.4, with ψ equal to a Z-spline $Z_{m,q}$. It uses cardinal interpolation as described in Section 4.6, i.e. $\hat\varphi_k = 1$ for all $k \in \mathbb{Z}_N$, so it is actually an implementation of Algorithm 4.4, which is a special case of Algorithm 4.1.

For efficiency, the for loop in Algorithm 5.4 is evaluated in blocks instead of separately for every j. The code is faster for large block sizes, but it uses less memory for small blocks.

There is a difficulty in the evaluation of Z-splines that comes from their piecewise definition. We could use Code 5.3, but have elected to take advantage of the extra knowledge we have on where we need to evaluate a Z-spline. We call a code similar to the one above, that does not, however, construct the cell array `intx` but instead takes it as an argument in matrix form.

Code 5.5. `function s = adftz(fc, x, a, zs, mbs)`

```
%ADFTZ approximate dft with Z-splines
%   s = ADFTZ(fc,x,a,zs,mbs) or s = ADFTZ(fc,x,a,zs) (default mbs =
%   numel(x)) or s = ADFTZ(fc,x,a,[m q],...) or s = ADFTZ(fc,x,a,m,...)
%   (default q = m) uses Z-splines to calculate an approximate discrete
%   Fourier transfrom of fc, a trigonometric polynomial given in symmetric
%   form, at arbitrarily spaced x in R.  The parameter a denotes the
%   oversampling factor, that is, n=a*length(fc) Z-splines are used to
%   interpolate fc at equidistant nodes.  The Z-splines are either given by
%   the structure zs, which should be generated with ZSMAKE, or is
%   constructed from the parameters m and q; they have (translated) support
%   [-m,m] and are piecewise polynomials of degree 2*q-1 in C^(q-1),
%   where q and m satisfy 1 <= q <= 2*m-1.
%
%   Since the evaluation of Z-splines at all of the necessary values can
%   use a lot of memory, this code evaluates the Z-splines in blocks of
%   size at most mbs.  This limits the memory usage to about 60*m*mbs
%   bytes for evaluation and 8*a*length(fc) bytes for FFT.

% if parameters [m q] of Z-spline are given, construct the Z-spline.
% Coefficients necessary for evaluation are stored in the struct zs.
if isa(zs,'numeric')
```

```
    zs = zsmake(zs);
end

% initialize constants and solution
N = length(fc);             % length of trigonometric polynomial
n = floor(N*a);             % number of nodes
n0 = floor(n/2);            % first node is at -n0/n
Nx = numel(x);              % length of x, often Nx=N
s = zeros(size(x));         % solution
supp = -zs.m+1:zs.m;        % nodes in support of Z-spline

% f is approximated by f(x) = sum_j g(j)*z(n*x-j), where z is the Z-spline.
% The coefficients g are calculated using fft and the fourier coefficients
% fc.
g = symfft(fc,n);

% calculate blocks for evaluation.  block stores the first index (in x) of
% each block, plus an extra value to end the last block.
if nargin < 5 || ~isa(mbs,'numeric')
    mbs = Nx;
end
block = [1:mbs:Nx,Nx+1];
blocksize = diff(block);

% start evaluating blocks
for k=1:(length(block)-1)
% evaluate x(blockid) in this loop
blockid = block(k):(block(k+1)-1);

% find the required evaluations of the (standard) Z-spline around 0; the
% actual evaluation is not done until later.  X denotes the arguments for
% the Z-spline; G stores the coefficients in the formula
%       f(x) = sum_j g(j)*z(n*x-j) .
nx = n*x(blockid)';
fnx = floor(nx);
dnx = nx - fnx;
X = dnx(:,ones(2*zs.m,1)) - supp(ones(blocksize(k),1),:);
fnx = fnx + n0;
idg = fnx(:,ones(2*zs.m,1)) + supp(ones(blocksize(k),1),:);
G = g(mod(idg,n)+1);

% for technical reasons, we need to know in which interval
%       [-m,-m+1),...,[0,1),...,[m-1,m)
% that X(i) is.  More precisely,
%       intX(j+m+1,:) = {i; floor(X(i)) = j} .
% Note that X is actually a matrix, so the index of X(j,h) is
% i=j+blocksize(k)*(h-1), ie X(i) = X(j,h) for this i.
I = blocksize(k)*(2*zs.m-1:-1:0)';
J = 1:blocksize(k);
intX = J(ones(1,2*zs.m),:) + I(:,ones(1,blocksize(k)));

% finally, we can evaluate
%       s = f(x) = sum_j g(j)*z(n*x-j)
% for this block.
```

```
s(blockid) = sum(G.*zs.vali(X,intX),2);
end
```

5.3 Implementation of (dual) Algorithm 4.6

Our implementation of Algorithm 4.6 is similar to that of Algorithm 4.1, which we covered Section 5.2 above. We include it mainly for completeness.

Algorithm 5.6.

```
function DUALADFT($f, \xi, a, m, \hat{\varphi}$)
    $N \leftarrow \mathrm{length}(f)$
    $n \leftarrow aN$
    $h \leftarrow 0$
    for $k \leftarrow 1$ to $N$ do
        $\Psi \leftarrow \psi(n\xi(k) - \lfloor n\xi(k) \rfloor - (-m : m))$        ▷ values of $\psi$
        $I \leftarrow \lfloor n\xi(k) \rfloor + (-m : m) \mod n$        ▷ indices of $h$ affected by $\xi_k$
        $h(I) \leftarrow h(I) + f(k)\Psi$        ▷ $h_l \leftarrow h_l + \hat{f}(\xi_k) \sum_{r\in\mathbb{Z}} \psi(n\xi_k - l - nr)$
    end for
    $\hat{h} \leftarrow \mathrm{fft}(h)$        ▷ $\hat{h}_j := \sum_{l\in\mathbb{Z}_n} h_l \,\mathrm{e}^{-2\pi \mathrm{i} jl/n}$
    $s = \frac{\hat{h}}{\hat{\varphi}}$        ▷ where $\hat{h} \leftarrow \hat{h}(\mathbb{Z}_N)$
    return $s$
end function
```

Code 5.7 is an implementation of Algorithm 4.6 similar to Algorithm 5.6. It uses Z-splines for ψ and $\hat{\varphi}_j = 1$ for all $j \in \mathbb{Z}$ and therefore has similar properties to Code 5.5, which we discussed above.

Code 5.7. `function s = dualadftz(fc, x, a, zs, mbs)`

```
%DUALADFTZ dual approximate dft with Z-splines
%   s = DUALADFTZ(fc,x,a,zs,mbs) (or variations as in ADFTZ) uses Z-splines
%   to approximate an inverse discrete Fourier transform for arbitrarily
%   spaced frequencies x of the vecter fc (with length(fc) = length(x)).
%   The other arguments are the same as in ADFTZ.

% if parameters [m q] of Z-spline are given, construct the Z-spline.
% Coefficients necessary for evaluation are stored in the struct zs.
if isa(zs,'numeric')
    zs = zsmake(zs);
end

% initialize constants and solution
N = length(fc);             % length of x and fc
N0 = floor(N/2);            % half of that
n = floor(N*a);             % number of nodes
n0 = floor(n/2);            % first node is at -n0/n
h = zeros(1,n);             % will store evaluations of psi
supp = -zs.m+1:zs.m;        % nodes in support of Z-spline

% calculate blocks for evaluation.  block stores the first index (in x) of
% each block, plus an extra value to end the last block.
if nargin < 5 || ~isa(mbs,'numeric')
    mbs = N;
end
```

```
block = [1:mbs:N,N+1];
blocksize = diff(block);

% start evaluating blocks
for k=1:(length(block)-1)
% evaluate x(blockid) in this loop
blockid = block(k):(block(k+1)-1);

% find the required evaluations of the (standard) Z-spline around 0; the
% actual evaluation is not done until later.  X denotes the arguments for
% the Z-spline;
nx = n*x(blockid)';
fnx = floor(nx);
dnx = nx - fnx;
X = dnx(:,ones(2*zs.m,1)) - supp(ones(blocksize(k),1),:);
fnx = fnx + n0;

% for technical reasons, we need to know in which interval
%       [-m,-m+1),...,[0,1),...,[m-1,m)
% that X(i) is.  More precisely,
%       intX(j+m+1,:) = {i; floor(X(i)) = j} .
% Note that X is actually a matrix, so the index of X(j,h) is
% i=j+blocksize(k)*(h-1), ie X(i) = X(j,h) for this i.
I = blocksize(k)*(2*zs.m-1:-1:0)';
J = 1:blocksize(k);
intX = J(ones(1,2*zs.m),:) + I(:,ones(1,blocksize(k)));

% evaluate z(n*x-j) for this block
Z = zs.vali(X,intX);

% sum the values of z(n*x-j) over x
for j=1:blocksize(k)
    idh = mod(fnx(j)+supp,n)+1;
    h(idh) = h(idh) + fc(blockid(j))*Z(j,:);
end
end

% the solution is just the (inverse) discrete fourier transform of h
S = symfft(h);
s = S(1-N0+n0:n0-N0+N);
```

6 Numerical experiments

6.1 Z-spline interpolation

h-convergence

In this section, we study the estimates from Section 2 for Z-splines. By Theorem 3.4, the Z-spline $Z_{m,q}$ is of order $\min(2m-1,2q)$. Therefore, for smooth enough arguments, Theorem 2.4 bounds the L^∞-error by $ch^{\min(2m-1,2q)}$.

We will say that a cardinal interpolation kernel ψ has *algebraic h-convergence rate* γ in the norm $\|\cdot\|$, iff for all smooth enough u, there is a c independent of

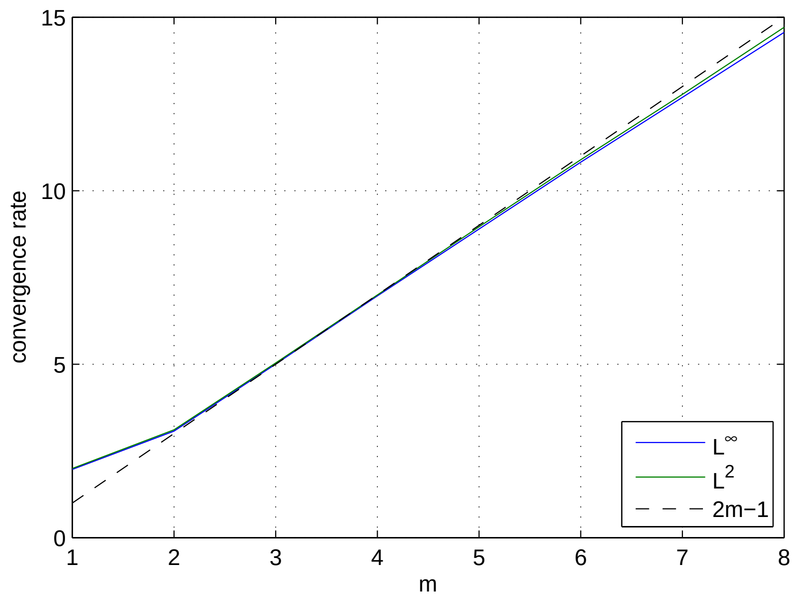

Figure 6.1: Algebraic h-convergence rate for interpolation of $\cos(x)$ with standard Z-splines Z_m.

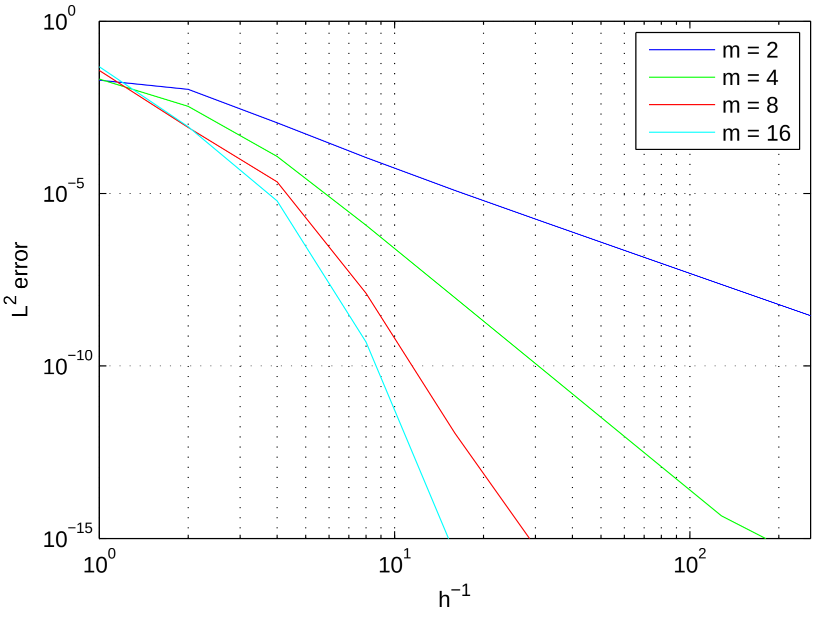

Figure 6.2: L^2 interpolation error for $\frac{1}{1+x^2}$ on the interval $[-3, 3]$ using standard Z-splines Z_m.

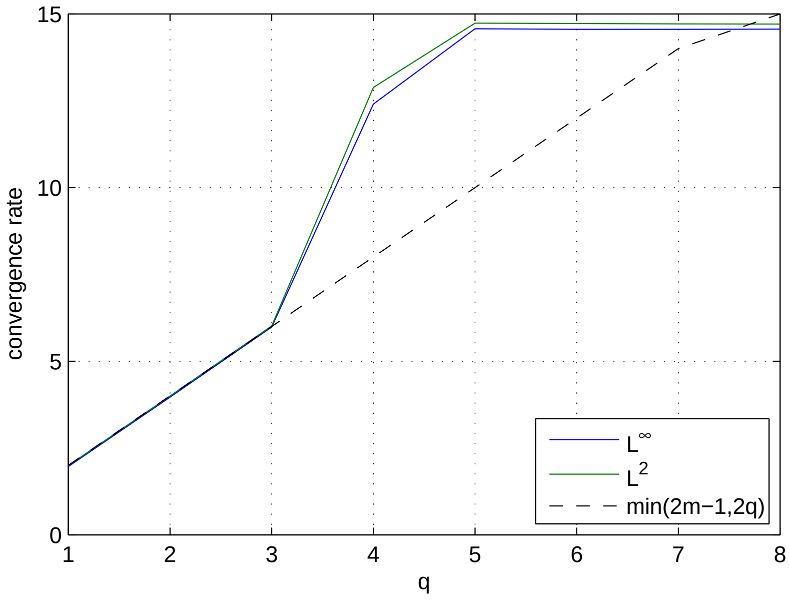

Figure 6.3: Algebraic h-convergence rate for interpolation of $\cos(x)$ with Z-splines $Z_{m,q}$ for $m = 8$.

h for which

$$\left\| u - I_\psi^h u \right\| \leq c h^\gamma \ . \tag{6.1}$$

By the remarks above, theory predicts an algebraic h-convergence rate of $\min(2m-1, 2q)$ for the Z-spline $Z_{m,q}$ in the L^∞-norm.

Figure 6.1 shows that this estimate for the convergence rate is very accurate for standard Z-splines Z_m. In fact, as could be expected, the same convergence rate holds for the L^2-norm. The errors in Figure 6.1 refer to the interpolation of $\cos(x)$ on a bounded interval. We estimated the convergence rates by the slopes of least-square lines through numerical estimates of the errors for various grid sizes h.

In Figure 6.2, we interpolated

$$f(x) := \frac{1}{1+x^2} \tag{6.2}$$

over the interval $[-3, 3]$. f is a meromorphic function with poles at $\{-\mathrm{i}, \mathrm{i}\}$. Apparently, these singularities limit the convergence around zero for large grid sizes h. However, after this pre-asymptotic effect, we seem to regain the expected convergence with increasing convergence rates for higher order Z-splines.

Figure 6.3 shows the h-convergence rates for Z-splines $Z_{m,q}$ with fixed $m = 8$. We used the same function, $\cos(x)$, and the same estimates as in Figure 6.1 to calculate these convergence rates. Surprisingly, we get much better results than expected; the maximal convergence rate $2m - 1$ is reached for $q < m$, in this case for $q = 5$ when $m = 8$!

This effect is also visible in Figure 6.4. Here, we consider the interpolation of f as in Figure 6.2, but with fixed $m = 12$ and various q instead of always

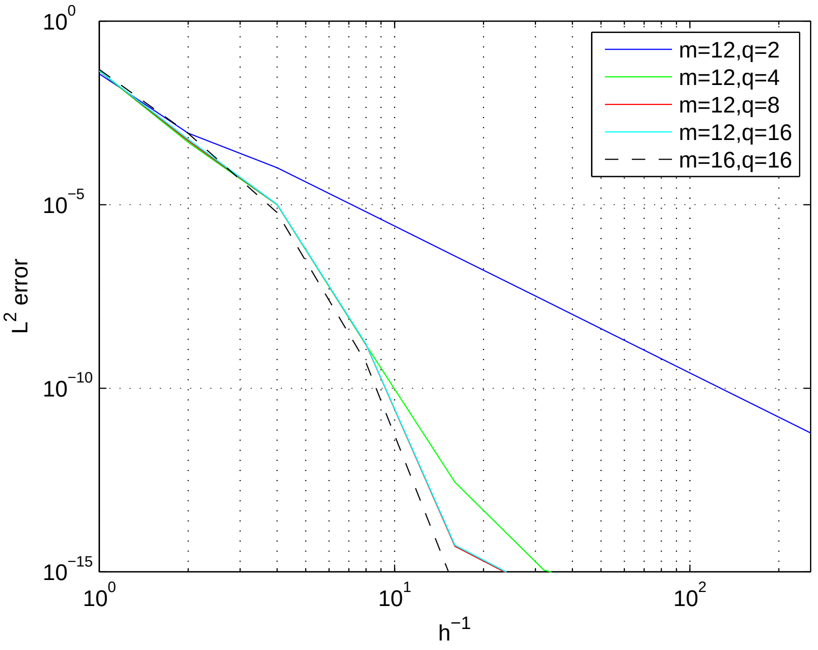

Figure 6.4: L^2 interpolation error for $\frac{1}{1+x^2}$ on the interval $[-3, 3]$ using Z-splines $Z_{12,q}$ for various q.

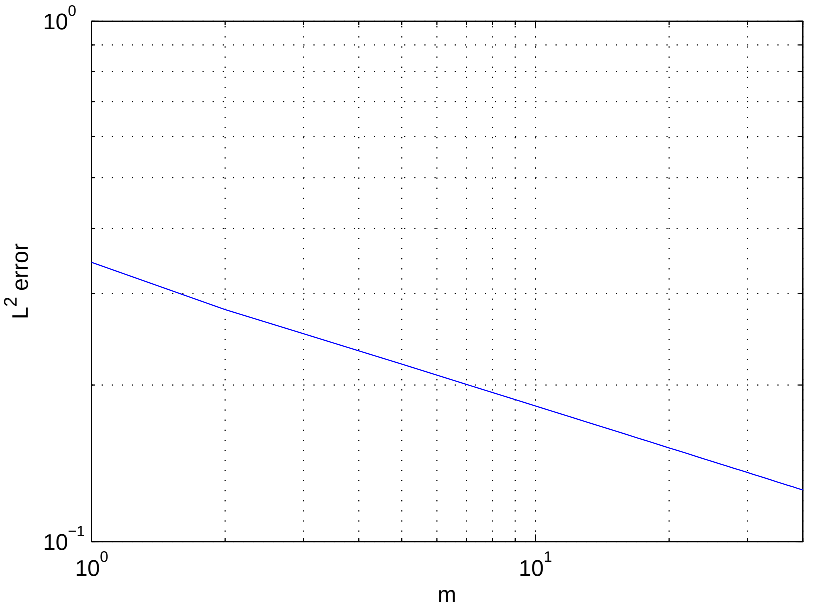

Figure 6.5: L^2-convergence of Z-splines Z_m to sinc on the interval $[-50, 50]$.

setting $q = m$. Apparently, the error for $q = 2$ is only slightly better than in Figure 6.2, but for $q = 4$, the error is almost as small as for the optimal q. The solutions for $q = 8 < m$ and $q = 16 > m$ are indistinguishable. This indicates that there is some $q_0 < m$ such that for all $q_0 \leq q \leq 2m - 1$, the interpolation error for $Z_{m,q}$ is as small as that for Z_m. So not only does the error not get better for $q > m$, it reaches this optimal level for some $q < m$.

Note that the flattening of the curves at an error of about 10^{-15} in Figure 6.4 and others is due to numerical effects; we are only calculating to an accuracy of this order of magnitude.

p-convergence

In Section 2.3, we saw that under certain conditions, cardinal interpolation kernels approximate sinc. More precisely, we predicted exponential convergence of a sequence of kernels $(\psi_n)_{n\in\mathbb{N}}$ to sinc in the L^2-norm. However, this significantly restricts the supports of ψ_n; since sinc only falls as $\frac{1}{n}$, the support of ψ_n must increase exponentially in order to get exponential convergence. Clearly, this is not the case for $(Z_m)_{m\in\mathbb{N}}$. Accordingly, as Figure 6.5 shows, Z-splines do not converge to sinc exponentially in the L^2-norm; the convergence is only algebraic, and at an extremely slow rate.

Theorem 2.7 predicts that if we decrease the grid spacing h, the convergence rate improves, since we can decrease γ and still satisfy the assumptions of the theorem.[xi] Indeed, Figure 6.6 shows that I_m^h sinc converges to sinc exponentially in the L^2-norm for $m \to \infty$ when h is small enough, i.e. for $h < h_0$ there is a

[xi]In the case of sinc, we can simply set $\gamma = h$.

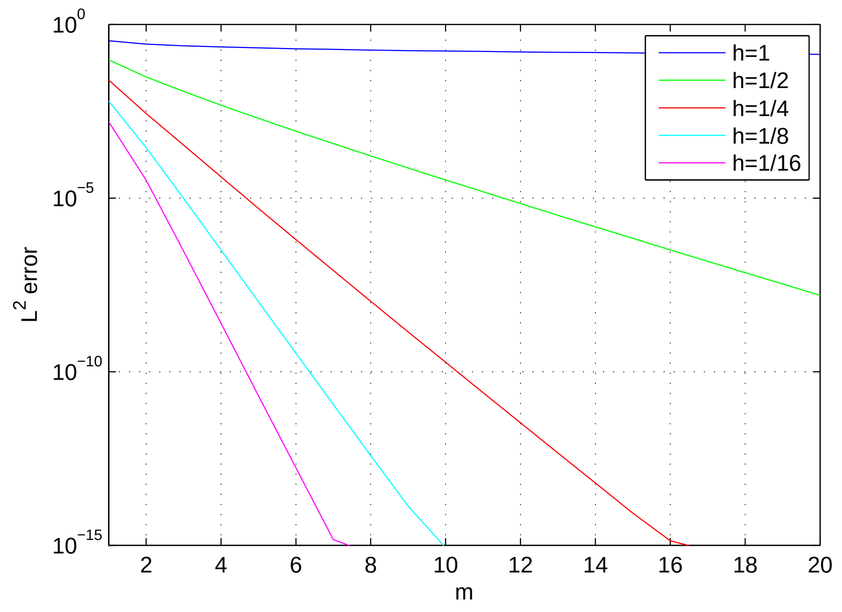

Figure 6.6: L^2 interpolation error for sinc on the interval $[-20, 20]$ using standard Z-splines Z_m and various grid sizes.

$c \in \mathbb{R}$ and $\gamma > 0$ such that

$$\left\|\operatorname{sinc} - I_m^h \operatorname{sinc}\right\|_{L^2} \le c\, e^{-\gamma m} \;.$$

We saw above that interpolation with $Z_{m,q}$ is as accurate as interpolation with Z_m for some $q < m$. In Figure 6.7, we consider the L^2-convergence of $I_{m,q}^{1/4} \operatorname{sinc}$ to sinc as $m \to \infty$ and $q = \min(m, q_{\max})$. Apparently, if we limit q this way, we no longer have convergence. Of course, for $m \le q_{\max}$, the situation is identical to that in Figure 6.6. At some point $\bar{m}$, though, the error flattens and stays constant for $m \ge \bar{m}$. Interestingly, $\bar{m}$ can be significantly larger than $q_{\max}$. For example, for $h = \frac{1}{4}$, when $q_{\max} = 5$, $\bar{m} = 12$ and the error reaches about 10^{-11}. If $q_{\max} = 7$, the convergence of $I_{m,q}^{1/4}$ is indistinguishable from that of $I_m^{1/4}$ up to the precision of our calculations. These results depend on the choice of h and, to a lesser extent, on the regularity of the function being interpolated.

We have seen that we get exponential convergence for the entire function sinc. To what extent does this carry over to less regular functions? Figure 6.8 shows that it does not. We consider here the same situation as in Figure 6.6, but with $f(x) := \frac{1}{1+x^2}$ in place of sinc. This function is analytic on all of $\mathbb{R}$, but we still do not have exponential convergence. In fact, for large h, $I_m^h f$ does not seem to converge to f in L^2 at all.

6.2 Error estimates for Z-splines

In Corollaries 4.3 and 4.5, we estimated the error of Algorithm 4.1 in various norms for various choices of $\hat{\varphi}$. We would like to study the terms that appear in

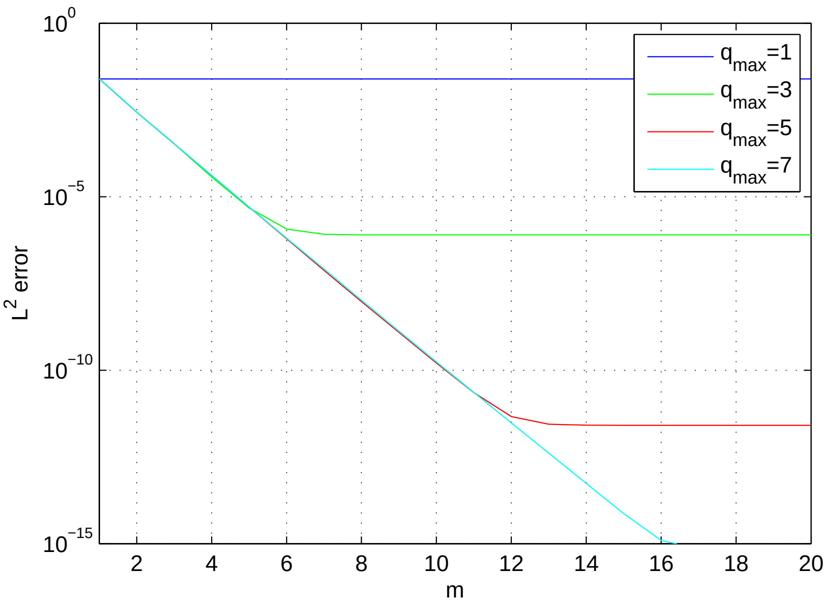

Figure 6.7: L^2 interpolation error for sinc on the interval $[-20, 20]$ using Z-splines $Z_{m,q}$ with $q := \min(m, q_{\max})$ and $h = \frac{1}{4}$.

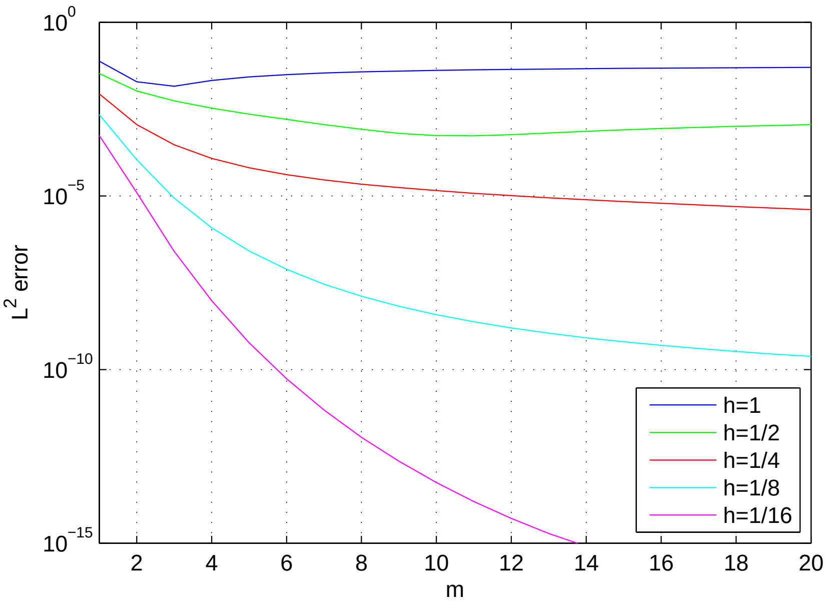

Figure 6.8: L^2 interpolation error for $\frac{1}{1+x^2}$ on the interval $[-3, 3]$ using standard Z-splines Z_m and various grid sizes.

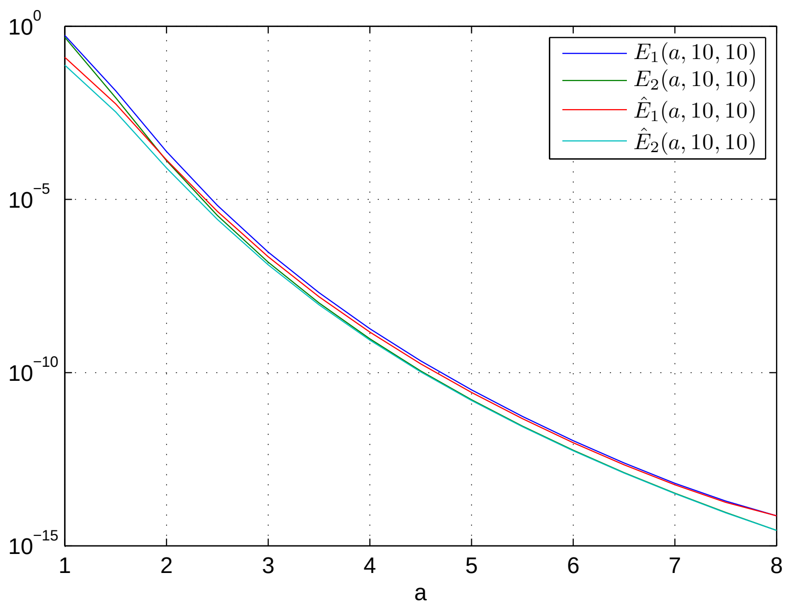

Figure 6.9: Terms in (6.3) for Z_{10} and various a.

these estimates for Z-splines. Note that similar terms appear in Theorem 2.7 and Theorem 4.7, so these considerations have a wider scope than just Algorithm 4.1.

The terms we are interested in are:

$$\begin{aligned}
E_1(a,m,q) &:= \max_{|k|\le\frac{1}{2a}} \sum_{r\in\mathbb{Z}} \left|\delta_{r0} - \hat{Z}_{m,q}(k+r)\right| \\
E_2(a,m,q) &:= \max_{|k|\le\frac{1}{2a}} \left(\sum_{r\in\mathbb{Z}} \left|\delta_{r0} - \hat{Z}_{m,q}(k+r)\right|^2\right)^{\frac{1}{2}} \\
\hat{E}_1(a,m,q) &:= \max_{|k|\le\frac{1}{2a}} \sum_{r\in\mathbb{Z}\setminus\{0\}} \left|\frac{\hat{Z}_{m,q}(k+r)}{\hat{Z}_{m,q}(k)}\right| \\
\hat{E}_2(a,m,q) &:= \max_{|k|\le\frac{1}{2a}} \left(\sum_{r\in\mathbb{Z}\setminus\{0\}} \left|\frac{\hat{Z}_{m,q}(k+rn)}{\hat{Z}_{m,q}(k)}\right|^2\right)^{\frac{1}{2}}
\end{aligned} \tag{6.3}$$

Our goal is to compare the error estimates given by Corollary 4.5 for Algorithm 4.4 to the estimates in Corollary 4.3 for Algorithm 4.1 with $\hat{\varphi}_k = \hat{Z}_{m,q}\left(\frac{k}{n}\right)$. The former contain the terms E_1 and E_2 and the latter are identical, but with E_1 and E_2 replaced by $\hat{E}_1$ and $\hat{E}_2$.

Figure 6.9 plots the terms in equation (6.3) for $m = q = 10$ as functions of the oversampling factor a. We are approximating $\max_{|k|\le\frac{1}{2a}} F(k)$ simply by $F\left(\frac{1}{2a}\right)$. This assumes some monotonicity, which is implied by the monotonicity of the graphs in Figure 6.9. In Figure 6.10, we plot the terms in equation (6.3) for $a = 3$ constant as a function of m, where we always set $q = m$. In accordance

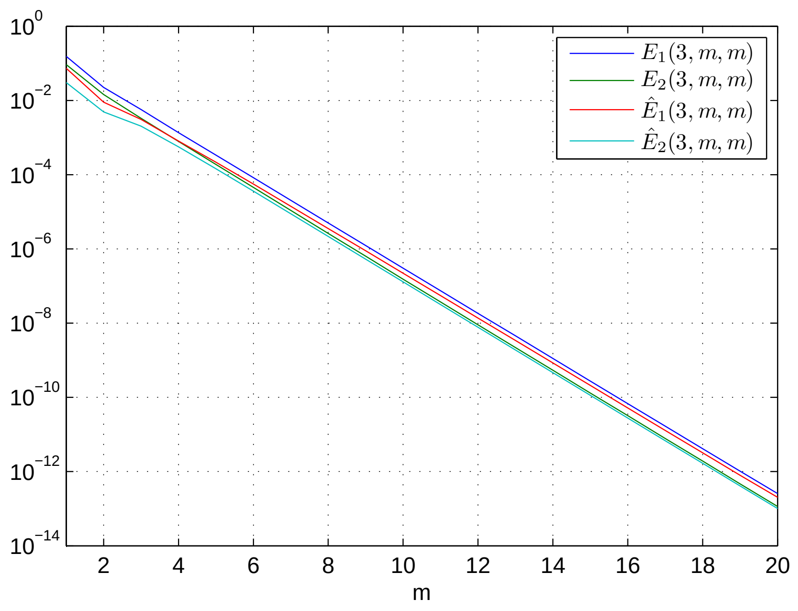

Figure 6.10: Terms in (6.3) for $a = 3$ and various standard Z-splines.

with our results from Section 6.1 above, we have exponential convergence as $m \to \infty$ and some slower convergence for $a \to \infty$, i.e. $h \to 0$.

Both plots show that all of the terms in equation (6.3) are very close together. In particular, E_i and $\hat{E}_i$ seem to converge towards each other as $a \to \infty$ or $m \to \infty$. This indicates that our two choices for $\hat{\varphi}$ in Algorithm 4.1 have the same convergence properties. Therefore, we can get around calculating $\hat{Z}_{m,q}$ with no loss of accuracy by using Algorithm 4.4, which simply interpolates the trigonometric polynomial we want to evaluate.

6.3 Convergence of Algorithm 4.1

In this section, we compare the convergence of Algorithm 4.1 with Z-splines to the other possibilities mentioned in the introduction: truncated Gaussian bells and B-splines. Our disappointing conclusion is that the algorithm works well with Z-splines, but not as well as with the standard choices of ψ.

As Section 6.2 indicates, Algorithm 4.1 has identical convergence properties for both choices of $\hat{\varphi}$ we considered above. We will therefore restrict ourselves to the choice which seems most natural for a given ψ, that is, cardinal interpolation as described in Section 4.6 for Z-splines and $\hat{\varphi}_k = \hat{\psi}\left(\frac{k}{n}\right)$ for truncated Gaussian bells and B-splines.

For simplicity, we will restrict ourselves to standard Z-splines. The performance of Algorithm 4.1 with $Z_{m,q}$ for $q \neq m$ is analogous to that of cardinal interpolation discussed in Section 6.1. Of course, in the case of Algorithm 4.4, the two are identical since this version of the DFT algorithm simply interpolates the trigonometric polynomial.

We consider Algorithm 4.1 not with a single ψ but with a sequence $(\psi_m)_{m\in\mathbb{N}}$,

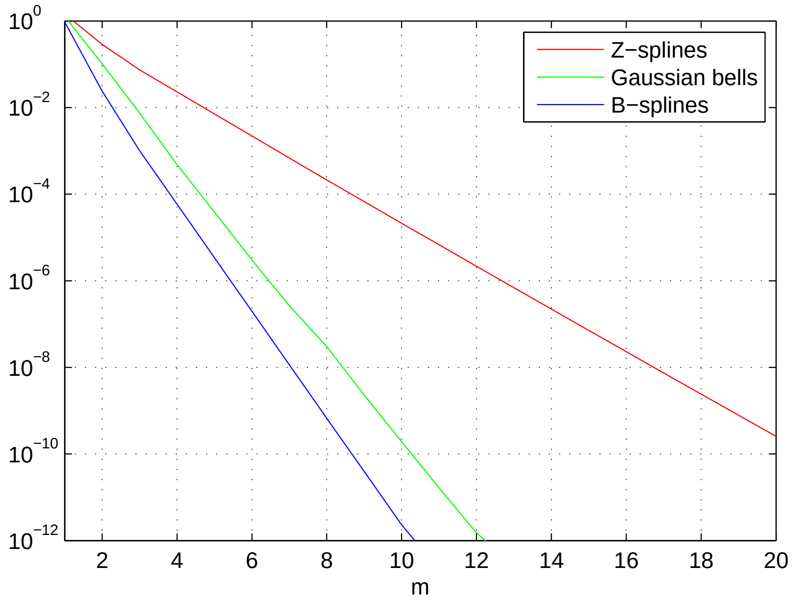

Figure 6.11: L^∞-convergence of Algorithm 4.1 with oversampling factor $a = 2.5$

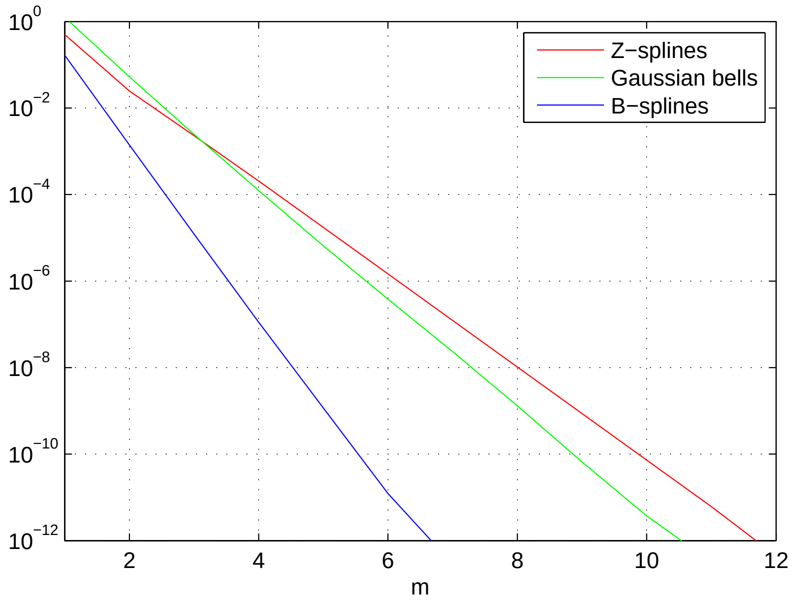

Figure 6.12: L^∞-convergence of Algorithm 4.1 with oversampling factor $a = 5$

where $\operatorname{supp}\psi_m = [-m, m]$. For Z-splines, of course, $\psi_m = Z_m$. For truncated Gaussian bells,

$$\psi_m(x) = \frac{1}{\sqrt{b}}\,\mathrm{e}^{-\pi x^2/b}\,\chi_{[-m,m]}(x) \quad \text{with} \quad b = \frac{2am}{2a-1}\ .$$

This is the choice of b recommended by G. Steidl in [5]. Finally, for B-splines, ψ_m is the B-spline B_{2m} of order $2m$.

In Figure 6.11, we plot the maximal error of Algorithm 4.1 with a reasonable oversampling factor of $a = 2.5$. More precisely, the error is the maximal deviation of the approximate evaluation of a random complex trigonometric polynomial of length $N = 128$ at N random points.

Apparently, the convergence of Algorithm 4.1 with Z-splines is far slower than with Gaussian bells or B-splines. As Figure 6.12 shows, Z-splines manage to catch up with Gaussian bells for large oversampling factors a, but the convergence rate of B-splines seems out of reach.

As expected, we have perfect exponential convergence in both cases. In our further experiments, we will gauge the accuracy of Algorithm 4.1 for various parameters by this convergence rate. More precisely, for given $(\psi_m)_{m\in\mathbb{N}}$, a and N, Algorithm 4.1 has *exponential convergence rate* γ iff for all trigonometric polynomials f of length N, there is a $c \in \mathbb{R}$ such that

$$E_\infty(f, a, \psi_m) \le c\,\mathrm{e}^{-\gamma m} \tag{6.4}$$

for all $m \in \mathbb{N}$, where $E_\infty(f, a, \psi_m)$ is the L^∞-error for the approximate evaluation of f using Algorithm 4.1 with $\psi = \psi_m$ and oversampling factor a.

In the following experiments, we use $E_\infty(f, a, \psi_m)$ for random trigonometric polynomials f to approximate γ. Our estimate for γ is the slope of the least squares line approximating the maxima over ten polynomials f of $\log E_\infty(f, a, \psi_m)$ for various m.

In Section 4.4, we saw that, under certain conditions, the convergence rate γ is independent of N. In Figure 6.13, we plot the (approximate) convergence rates of Algorithm 4.1 for N equal to the powers of two between $2^6 = 64$ and $2^{12} = 4096$ and least-square lines for these data points. Clearly, for all choices of ψ, the convergence rates are independent of N. We can therefore restrict ourselves to $N = 128$ without loss of generality.

Figure 6.14 shows the dependence of the convergence rate γ on the oversampling factor a. The circles are the numerical approximations of γ described above and the lines are estimates of the convergence rates given by the first estimate in Corollary 4.3. This estimate actually does not apply to our choice of $\hat{\varphi}$ for Z-splines, but, as discussed in Section 6.2, it does hold at least asymptotically.

This plot confirms the disappointing performance of Algorithm 4.1 with Z-splines suggested by Figures 6.11 and 6.12. For reasonable[xii] oversampling factors a, the convergence with Z-splines is much slower than with truncated Gaussian bells or with B-splines. For larger a, the convergence rate with Z-splines is higher than with Gaussian bells, but it never reaches the convergence rate of Algorithm 4.1 with B-splines.

[xii] small

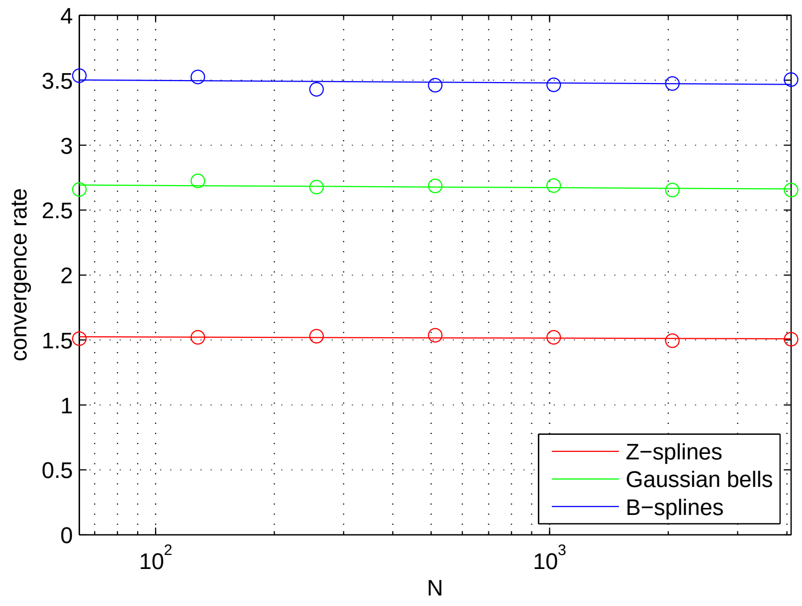

Figure 6.13: Exponential convergence rate γ of Algorithm 4.1 with $a = 3$ for various N. The circles are numerical experiments and the lines are least-square approximations.

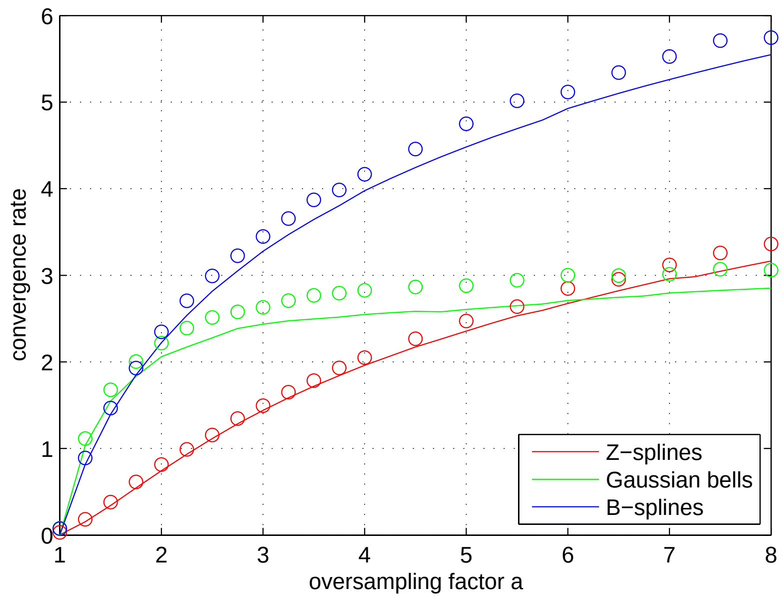

Figure 6.14: Exponential convergence rate γ of Algorithm 4.1 for various oversampling factors a. The circles are numerical experiments and the lines are error estimates using $\hat{E}_1$ from (6.3) as in Corollary 4.3.

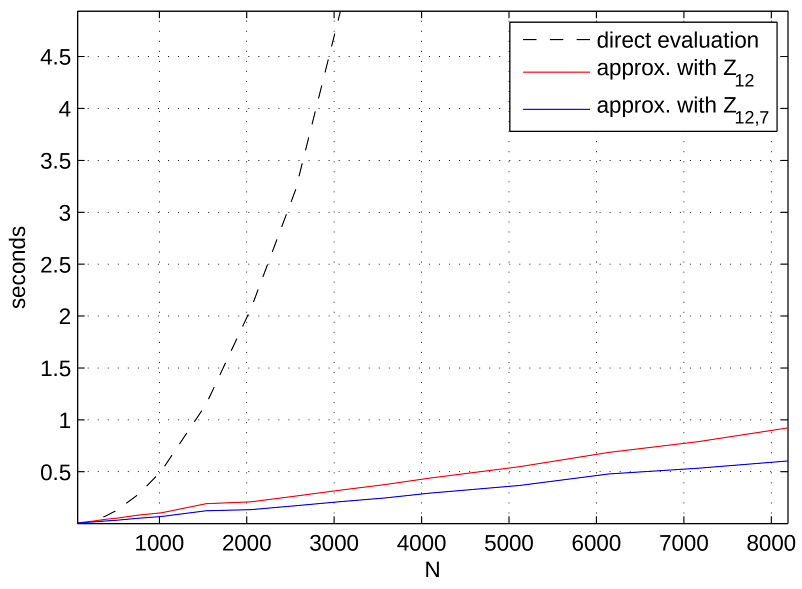

Figure 6.15: Performance of Algorithm 4.1 with Z-splines, to an accuracy of 10^{-10}.

6.4 Complexity of Algorithm 4.1

By (4.8), Algorithm 4.1 evaluates a trigonometric polynomial of length N in $O(N \log N)$ time instead of $O(N^2)$ required for a direct evaluation. Figure 6.15 shows what a difference this makes.

We plot the time required to evaluate a random trigonometric polynomial of length N with $\max_k |\hat{f}_k| \leq 1$ at N random points using a direct evaluation and Algorithm 4.1. We used an oversampling factor $a = 4$ and Z-splines Z_{12} and $Z_{12,7}$, both of which lead to an accuracy of 10^{-10} in the max-norm. The times include the evaluation, but not the construction, of the Z-splines.[xiii] This experiment was carried out on an IBM ThinkPad T40p laptop computer with the MATLAB codes from Section 5.

Clearly, even for small N, Algorithm 4.1 is much faster than direct evaluation. Also, there is a significant advantage to using $Z_{12,7}$ instead of Z_{12}, as discussed in Section 6.1.

[xiii]In fact, the evaluation of Z-splines accounts for most of the time plotted in Figure 6.15.

References

[1] P. Duhamel and M. Vetterli. Fast Fourier transforms: a tutorial review and state of the art. *Signal Processing*, 19:259–299, 1990.

[2] D. Potts, G. Steidl, and M. Tasche. Fast Fourier transforms for nonequispaced date: A tutorial. In J. J. Benedetto and P. Ferreira, editors, *Modern Sampling Theory: Mathematics and Applications*, chapter 12, pages 253–274. Birkhäuser Basel, 2001.

[3] W. Rudin. *Functional Analysis.* McGraw-Hill, second edition, 1991.

[4] J. T. B. Sagredo. Z-splines: Moment conserving cardinal spline interpolation of compact support for arbitrarily spaced data. Research Report 2003-10, Seminar für Angewandte Mathematik, ETHZ, August 2003.

[5] G. Steidl. A note on fast Fourier transforms for nonequidistant grids. *Advances in Computational Mathematics*, 9:337–352, 1998.

[6] G. Strang and G. Fix. A Fourier analysis of the finite element variational method. In *Constructive aspects of functional analysis*, volume 2, pages 793–840. Centro Internationale Matematico Estivo, 1971.