On computing high-dimensional Riemann theta functions

Chimmalgi, Shrinivas; Wahls, Sander

DOI
10.1016/j.cnsns.2023.107266

Publication date
2023

Document Version
Final published version

Published in
Communications in Nonlinear Science and Numerical Simulation

Communications in Nonlinear Science and Numerical Simulation 123 (2023) 107266

Contents lists available at ScienceDirect

Communications in Nonlinear Science and Numerical Simulation

journal homepage: www.elsevier.com/locate/cnsns

Research paper

On computing high-dimensional Riemann theta functions

Shrinivas Chimmalgi [a], Sander Wahls [b,*]

[a] Karlsruhe Institute of Technology, Communications Engineering Lab (CEL), Karlsruhe, Germany
[b] Delft University of Technology, Delft Center for Systems and Control, Delft, The Netherlands

ARTICLE INFO

Article history:
Received 30 January 2023
Received in revised form 4 April 2023
Accepted 11 April 2023
Available online 18 April 2023

Keywords:
Riemann theta function
Tensor-train decomposition
Finite-genus solutions
Korteweg–de-Vries equation
Nonlinear Schrödinger equation
Nonlinear Fourier transform

ABSTRACT

Riemann theta functions play a crucial role in the field of nonlinear Fourier analysis, where they are used to realize inverse nonlinear Fourier transforms for periodic signals. The practical applicability of this approach has however been limited since Riemann theta functions are multi-dimensional Fourier series whose computation suffers from the curse of dimensionality. In this paper, we investigate several new approaches to compute Riemann theta functions with the goal of unlocking their practical potential. Our first contributions are novel theoretical lower and upper bounds on the series truncation error. These bounds allow us to rule out several of the existing approaches for the high-dimension regime. We then propose to consider low-rank tensor and hyperbolic cross based techniques. We first examine a tensor-train based algorithm which utilizes the popular scaling and squaring approach. We show theoretically that this approach cannot break the curse of dimensionality. Finally, we investigate two other tensor-train based methods numerically and compare them to hyperbolic cross based methods. Using finite-genus solutions of the Korteweg–de Vries (KdV) and nonlinear Schrödinger equation (NLS) equations, we demonstrate the accuracy of the proposed algorithms. The tensor-train based algorithms are shown to work well for low genus solutions with real arguments but are limited by memory for higher genera. The hyperbolic cross based algorithm also achieves high accuracy for low genus solutions. Its novelty is the ability to feasibly compute moderately accurate solutions (a relative error of magnitude 0.01) for high dimensions (up to 60). It therefore enables the computation of complex inverse nonlinear Fourier transforms that were so far out of reach.

1. Introduction

The Riemann theta function

$$\theta(\mathbf{z} \mid \mathbf{\Omega}) = \sum_{\mathbf{n}\in\mathbb{Z}^g} \mathrm{e}^{2\pi \mathrm{i}\left(\frac{1}{2}\mathbf{n}\cdot\mathbf{\Omega}\mathbf{n}+\mathbf{n}\cdot\mathbf{z}\right)}, \quad \mathbf{z}\in\mathbb{C}^g, \tag{1}$$

where $\mathbf{\Omega} = \mathbf{\Omega}^T \in \mathbb{C}^{g\times g}$ has a strictly positive definite imaginary part and $\cdot$ indicates the dot product, is a particular multi-dimensional Fourier series that plays a key role in the area of nonlinear Fourier analysis [1,2]. There, it is used to synthesize periodic signals as part of inverse nonlinear Fourier transforms. Periodic nonlinear Fourier analysis has recently received a lot of attention in nonlinear signal processing problems arising in fiber-optic communications [3–7] and coastal and ocean engineering [8–14]. The Riemann theta function also sees application in quantum coding [15],

* Corresponding author.
E-mail addresses: s.chimmalgi@kit.edu (S. Chimmalgi), s.wahls@tudelft.nl (S. Wahls).

https://doi.org/10.1016/j.cnsns.2023.107266

algebraic geometry [16–18], number theory [19], discrete mathematics [20], machine learning [21,22], cryptography [23] and statistics [24]. Despite its applicability in many fields, the practical utility of the Riemann theta function is limited to low number of dimensions due to its high computational cost. While there has been much work on designing efficient algorithms for the computation of the Riemann theta function [8,25–28], the complexity of these methods nevertheless increases exponentially with the number of dimensions. Hence, they are limited to low-dimensional problems. In this paper we propose novel approaches that overcome this limitation and allow us to synthesize high-dimensional non-trivial signals for fiber-optic communications and coastal engineering problems.

Algorithms for computing the Riemann theta function can be primarily classified into two categories. The first category of algorithms concentrate on computing the Riemann theta function value up to a certain number of bits [26]. The second category of algorithms aim at computing the theta function value up to a small threshold. In this paper we focus on the second category of algorithms [8,25,27,28]. These methods approximate the theta function value by summing a truncated series. The number of terms in the summation grows exponentially with the number of dimensions g. Hence, the computational cost grows exponentially as well. It becomes infeasible to compute the sum even for moderate values of g. This is famously known as the *curse of dimensionality*. The curse is seen in many high-dimensional problems where the number of operations required for a particular action grow exponentially with the underlying dimensionality [29]. In recent years tensor based methods have been increasingly employed to mitigate the curse of dimensionality. They have been applied with great success in signal processing, statistics, data mining, and machine learning [30–38]. In particular, they have been used to develop efficient algorithms for computing multi-dimensional Fourier series [39,40]. Another approach used to reduce the computation cost of the multi-dimensional Fourier series is the utilization of special index sets [41–45].

In this paper we study the applicability of tensor based algorithms and special index sets to efficiently compute approximations of the high-dimensional Riemann theta function. Our main contributions are: (1) Lower and upper bounds on the error introduced from the series truncation for certain index sets, (2) theoretical proof that a standard scaling-and-squaring approach applied to tensor-train approximations cannot break the curse of dimensionality, and (3) numerical investigations of other tensor-train and hyperbolic-cross based algorithms. To the best of our knowledge, this is the first time that tensor-train and hyperbolic-cross methods have been proposed for the computation of Riemann theta functions. In our numerical experiments, we are able to compute Riemann theta function values for very high number of dimensions ($g = 60$) that dramatically exceed the current state of the art in this area, which significantly extends the range of practical problems to which Riemann theta functions can be applied.

The remainder of this paper has the following structure. In Section 2 we introduce some preliminaries about the Riemann theta function and the tensor-train decomposition. In Section 3 we derive some lower bounds and an upper bound on the series truncation error for certain index sets. In Section 4 we provide a theoretical analysis to prove that a tensor-train based approach using scaling and squaring cannot break the curse of dimensionality. In Section 5 we present two alternatives for the tensor-train based algorithm and use the hyper-elliptic solutions of the KdV and NLS equations as numerical examples to study the accuracies of the algorithms. We conclude our findings in Section 6.

Notations. $\mathbb{C}$ — complex numbers; $\mathbb{N}$ — Natural numbers; $\mathbb{Z}$ — Integers; $\#A$ denotes the number of elements in the set A. We use $\mathrm{e}^{(\cdot)}$ and $\exp(\cdot)$ interchangeably to indicate the exponential function (applied element-wise for tensors).

2. Preliminaries

In this section, we recapitulate several results related to the computation of the Riemann theta function from the literature. First, the most common ways to truncate the infinite series (1) are introduced. Then, a technique to make truncation more efficient by transforming the Riemann matrix $\boldsymbol{\Omega}$ that is known as Siegel transform is discussed. Finally, tensor trains are introduced as a potential tool to evaluate the truncated series.

2.1. Truncated Riemann theta functions

For numerical purposes, the Riemann theta function (1) is typically approximated by a truncated series of the form

$$\hat{\theta}(\mathbf{z} \mid \boldsymbol{\Omega}) = \sum_{\mathbf{n} \in \mathcal{N}^g(N)} \underline{\mathbf{C}}(\mathbf{n}) \mathrm{e}^{2\pi \mathrm{i}(\mathbf{n}\cdot\mathbf{z})}, \tag{2}$$

where $\underline{\mathbf{C}}$ is a g-dimensional tensor with

$$\underline{\mathbf{C}}(\mathbf{n}) = \mathrm{e}^{\mathrm{i}\pi \mathbf{n}\cdot\boldsymbol{\Omega}\mathbf{n}}. \tag{3}$$

The index set $\mathcal{N}^g = \mathcal{N}^g(N)$ depends on a truncation parameter $N \in \mathbb{N}$. Before we can introduce several popular index sets, the definition of the matrix p-norms has to be recalled.

Definition 1. Let $p \in \mathbb{N} \cup \{\infty\}$. The p-norm of $\mathbf{z} \in \mathbb{C}^g$ is $\|\mathbf{z}\|_p := (|z_1|^p + \cdots + |z|_g^p)^{1/p}$ for $p < \infty$ and $\|\mathbf{z}\|_p := \max\{|z_1|, \ldots, |z_g|\}$ for $p = \infty$. The corresponding induced p-norm of a matrix $\mathbf{A} \in \mathbb{C}^{g\times g}$ is $\|\mathbf{A}\|_p := \max_{\substack{\mathbf{z}\in\mathbb{C}^g \\ \|\mathbf{z}\|_p \le 1}} \|\mathbf{A}\mathbf{z}\|_p$.

The first popular family index set, $\mathcal{I}^g$, is defined as follows.

Definition 2. For any $N \in \mathbb{N}$ and $p \in \mathbb{N} \cup \{\infty\}$, we set

$$\mathcal{I}^g = \mathcal{I}^g(p, N) := \left\{ \mathbf{n} \in \mathbb{Z}^g \ : \ \|\mathbf{n}\|_p \leq N \right\}.$$

The most common choices of p are $p = 1$ (summation over a cross-polytope), $p = 2$ (summation over a hypersphere), and $p = \infty$ (summation over a hypercube). The symmetric hyperbolic cross $\mathcal{H}^g$ is another interesting index set [46, Fig. 2.1(b)]. It is defined as follows.

Definition 3. For any $N \in \mathbb{N}$, we set

$$\mathcal{H}^g = \mathcal{H}^g(N) := \left\{ \mathbf{n} \in \mathbb{Z}^g \ : \ \prod_{j=1}^{g} \max\left(1, 2|n_j|\right) \leq N \right\}.$$

The hyperbolic cross $\mathcal{H}^g$ can be built recursively using the algorithm given in [47, Sec. 2.6]. In contrast to the index sets $\mathcal{I}^g(p, N)$, the number of elements in the hyperbolic cross $\mathcal{H}^g$ does not grow exponentially in the genus g if N is fixed.

Lemma 1. *The number of elements in $\mathcal{H}^g(N)$ satisfies*

$$\#\mathcal{H}^g(N) \leq \mathrm{e}^2 N^2 g^{\log_2 N}.$$

Proof. Since $1 + |n_j| \leq \max(1, 2|n_j|)$, we have that $\max(1, 2|n_j|) \leq N \Rightarrow 1 + |n_j| \leq N$. Therefore, $\mathcal{H}^g(N) \subseteq \{\mathbf{n} \in \mathbb{Z}^g \ : \ \prod_{j=1}^{g}(1+|n_j|) \leq N\}$. The bound [48, Eq. 10.2.3] implies $\#\mathcal{H}^g(N) \leq \mathrm{e}^2 N^{2+\log_2 g} = \mathrm{e}^2 N^2 N^{\log_2 g}$. The lemma now follows since $N^{\log_2 g} = (2^{\log_2 N})^{\log_2 g} = (2^{\log_2 g})^{\log_2 N} = g^{\log_2 N}$. □

The Riemann theta function is well-known to converge absolutely and uniformly in $\mathbf{z}$ [49, Ch. II.1]. Therefore, we have the following result.

Theorem 1 (*Convergence*)**.** *Let $Z \subset \mathbb{C}$ be compact. Then*

$$\lim_{N\to\infty} \max_{\mathbf{z}\in Z} \left| \theta(\mathbf{z} \mid \boldsymbol{\Omega}) - \hat{\theta}(\mathbf{z} \mid \boldsymbol{\Omega}) \right| = 0$$

for any of the index sets $\mathcal{I}^g$ and $\mathcal{H}^g$.

2.2. Siegel transform

For certain Riemann matrices $\boldsymbol{\Omega}$, it may happen that an index set contains many coefficients that could be neglected during the summation. If we sum over a hypersphere, this issue would for example arise if the indices of the non-negligible coefficients form an hyper-ellipsoid with high eccentricity. An algorithm for finding an hyper-ellipsoid of indices that includes all terms above a threshold is given in [25, Section 4]. The algorithm for identifying the hyper-ellipsoid of indices however unfortunately has a significant computational cost itself that grows sharply with increasing g. Furthermore, even if the hyper-ellipsoid is known, the number of terms inside it can still be very large [25, p. 1734]. In [25, Sec. 7], it was therefore proposed to use a modular transform of the form

$$\boldsymbol{\Omega} \mapsto \hat{\boldsymbol{\Omega}} = (\mathbf{A}\boldsymbol{\Omega} + \mathbf{B})(\mathbf{C}\boldsymbol{\Omega} + \mathbf{D})^{-1}, \quad \mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D} \in \mathbb{Z}^g, \tag{4}$$

where the integer matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{D}$ must satisfy

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{0}_g & \mathbf{I}_g \\ -\mathbf{I}_g & \mathbf{0}_g \end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^T = \begin{bmatrix} \mathbf{0}_g & \mathbf{I}_g \\ -\mathbf{I}_g & \mathbf{0}_g \end{bmatrix}, \tag{5}$$

known as Siegel transformation to reduce the eccentricity of the set of non-negligible indices. The relation between the two Riemann theta functions that correspond to the two Riemann matrices $\boldsymbol{\Omega}$ and $\hat{\boldsymbol{\Omega}}$ is then given by [49, Eq. 5.1]

$$\theta(\mathbf{z} \mid \boldsymbol{\Omega}) = \frac{\theta((\mathbf{C}\boldsymbol{\Omega} + \mathbf{D})^{-1}\mathbf{z} \mid \hat{\boldsymbol{\Omega}})}{\zeta \sqrt{\det(\mathbf{C}\boldsymbol{\Omega} + \mathbf{D})} \mathrm{e}^{\pi \mathrm{i} \mathbf{z}\cdot(\mathbf{C}\boldsymbol{\Omega}+\mathbf{D})^{-1}\mathbf{C}\mathbf{z}}}, \tag{6}$$

where ζ is an eighth root of one, i.e., $\zeta^8 = 1$. The main computational step in the construction of the modular transform is the approximation of the shortest vector in a lattice. In [25], the authors employed the Lenstra–Lenstra–Lovász (LLL) lattice basis reduction algorithm [50] for that purpose. The complexity of the LLL algorithm is only polynomial in the g, but the error in the approximation increases exponentially in g. Several authors therefore investigated replacements for the LLL algorithm [24,27].

The authors of [51] mention that once a Siegel transform has been applied, the summation can be carried over a hyper-cube instead of the hyper-ellipsoid as it was done in [25], at an additional cost.

2.3. Tensor-train decomposition

The truncated Riemann theta function (2) is a multi-dimensional Fourier series. In [39], it was observed that if the coefficient tensor has a low-rank representation in the tensor-train format [52], a multi-dimensional Fourier series can be evaluated with low computational complexity. The idea was to exploit that a multi-dimensional Fourier series constitutes the inner product between the coefficient tensor and the rank one tensor formed by the terms $\mathrm{e}^{2\pi \mathrm{i}\mathbf{n}\cdot\mathbf{z}}$, which can be then computed efficiently using the method in [52, Sec. 4.2]. Later in this paper, we will exploit this idea for the computation of the truncated Riemann theta function. Since this requires us to investigate approximations of the coefficient tensor (3), we now quickly recall some facts about the tensor-train format.

Any given tensor $\underline{\mathbf{Y}}$ can be approximated arbitrarily well by a tensor $\underline{\mathbf{X}} \approx \underline{\mathbf{Y}}$ of the form $\underline{\mathbf{X}} = \underline{\mathbf{G}}^{(1)} \times^1 \underline{\mathbf{G}}^{(2)} \times^1 \cdots \times^1 \underline{\mathbf{G}}^{(g)}$, [52], where $\times^1$ is the contracted product [53, Sec. 2.2] and the $\underline{\mathbf{G}}^{(k)}$ are 3rd-order tensors with sizes $R_{k-1} \times N_k \times R_k$, $k = 1, \ldots, g$ and $R_0 = R_g = 1$. Any tensor of this form is said to be a tensor train. The tensors $\underline{\mathbf{G}}^{(k)}$ are called the tensor-train cores, while the integers $R_1, \ldots, R_{g-1}$ are called the tensor-train ranks. A tensor-train can alternatively be written entry-wise as a product of slice matrices,

$$x_{\mathbf{n}} = x_{n_1,n_2,\ldots,n_g} = \mathbf{G}^{(1)}_{n_1} \mathbf{G}^{(2)}_{n_2} \cdots \mathbf{G}^{(g)}_{n_g}, \tag{7}$$

where $\mathbf{G}^{(k)}_{n_k} = \underline{\mathbf{G}}^{(k)}(:, n_k, :) \in \mathbb{C}^{R_{k-1} \times R_k}$ is the lateral slice of the nth tensor-train core, $k = 1, \ldots, g$, and $\mathbf{G}^{(1)}_{n_1} \in \mathbb{C}^{1 \times R_1}$ and $\mathbf{G}^{(g)}_{n_g} \in \mathbb{C}^{R_{g-1} \times 1}$. There exist efficient algorithms with which a tensor-train approximation of a given tensor can be found under accuracy and rank constraints, see e.g. [52,54]. In the following we will refer to the highest rank in a tensor-train as the rank of the tensor-train.

3. Analysis of the truncation error

For the numerical evaluation of the truncated Riemann theta function $\hat{\theta}(\mathbf{z} \mid \mathbf{\Omega})$, the truncation parameter N has to be chosen large enough such that the truncation error

$$|\theta(\mathbf{z} \mid \mathbf{\Omega}) - \hat{\theta}(\mathbf{z} \mid \mathbf{\Omega})| = \left| \sum_{\mathbf{n} \notin \mathcal{N}^g(N)} c_{\mathbf{n}} \mathrm{e}^{2\pi \mathrm{i}\mathbf{n}\cdot\mathbf{z}} \right| \tag{8}$$

is sufficiently small. The truncation error depends on the choice of the index set $\mathcal{N}^g(N)$. The parameters g and N furthermore determine the complexity of evaluation methods for the truncated Riemann theta function. In this section, we show that the two parameters are connected. We derive lower and upper bounds on the truncation parameter N such that a certain truncation error can be guaranteed. Since we are interested in high dimensional cases, their behavior is studied for large g. The consequences of these studies are discussed.

3.1. Lower bounds on the truncation error

Some authors have proposed to choose the truncation parameter N such that the truncated tensor coefficients satisfy $|c_{\mathbf{n}}| < \varepsilon$, where $\varepsilon > 0$ denotes some small parameter such as machine precision [51, Sec. 3.3], [27, p. 150], [8]. However, even if the errors in the individual coefficients $c_{\mathbf{n}}$ are very small, the truncation error (8) can be large since the number of neglected coefficients grows exponentially with the genus g. (Note that it is possible to sum terms accurately even when the numbers have significantly different orders of magnitude and are smaller than the machine precision [55]. Hence, it would be possible to include coefficients below machine precision also in finite precision arithmetic.) The following proposition, which is our first contribution, formalizes this observation for most of the index sets $\mathcal{I}^g$. It demonstrates that the strategy of truncating coefficients below machine precision can achieve small truncation errors only for small values of g with these index sets since the truncation error in general grows exponentially in the number of dimensions g if N is fixed.

Proposition 1. *Let the index set be $\mathcal{N}^g = \mathcal{I}^g(p, N)$ for any $p \in \mathbb{N}$, $p \geq 2$. The truncation error is then lower bounded as*

$$\left| \theta(\mathbf{z} \mid \mathbf{\Omega}) - \hat{\theta}(\mathbf{z} \mid \mathbf{\Omega}) \right| \geq \mathrm{e}^{g\left(\log(2) - \pi\lambda_{\max}\left\lceil \frac{N+1}{\sqrt[p]{g}} \right\rceil^2\right)} \tag{9}$$

at $\mathbf{z} = \mathbf{0}_{g\times 1}$ whenever $\Re\{\mathbf{\Omega}\} = \mathbf{0}_g$, where $\lambda_{\max}$ denotes the largest eigenvalue of $\Im\{\mathbf{\Omega}\}$.

Proof. Let $\mathcal{J}^g(N+1)$ denote the index set that contains all $\mathbf{n} \in \mathbb{Z}^g$ of the form $\mathbf{n} = \left\lceil \frac{N+1}{\sqrt[p]{g}} \right\rceil \begin{bmatrix} s_1 & s_2 & \cdots & s_g \end{bmatrix}^T$ where $s_k = \pm 1$ for $k = 1, 2, \ldots, g$. Then

$$\mathbf{n} \in \mathcal{J}^g(N+1) \Rightarrow \|\mathbf{n}\|_p \geq N+1 \Rightarrow \mathbf{n} \notin \mathcal{I}^g(p, N). \tag{10}$$

The truncation error thus satisfies

$$\begin{aligned}
\left|\theta(\mathbf{z} \mid \boldsymbol{\Omega}) - \hat{\theta}(\mathbf{z} \mid \boldsymbol{\Omega})\right| &= \left| \sum_{\mathbf{n} \notin \mathcal{I}^g(p,N)} c_{\mathbf{n}} \underbrace{\exp(2\pi \mathrm{i} \mathbf{n} \cdot \mathbf{z})}_{=1} \right| \\
(c_{\mathbf{n}} \in \mathbb{R}^+) \quad &= \sum_{\mathbf{n} \notin \mathcal{I}^g(p,N)} c_{\mathbf{n}} \\
(10) \quad &\geq \sum_{\mathbf{n} \in \mathcal{J}^g(N+1)} \mathrm{e}^{-\pi \mathbf{n} \cdot \Im\{\boldsymbol{\Omega}\} \mathbf{n}} \\
(\mathbf{n} \cdot \Im\{\boldsymbol{\Omega}\}\mathbf{n} \leq \lambda_{\max} \|\mathbf{n}\|_2^2) \quad &\geq \sum_{\mathbf{n} \in \mathcal{J}^g(N+1)} \mathrm{e}^{-\pi \lambda_{\max} g \left\lceil \frac{N+1}{\sqrt[p]{g}} \right\rceil^2} \\
(\text{since } \#\mathcal{J}^g(N+1) = 2^g) \quad &= 2^g \mathrm{e}^{-\pi \lambda_{\max} g \left\lceil \frac{N+1}{\sqrt[p]{g}} \right\rceil^2} \\
&= \mathrm{e}^{g\left(\log(2) - \pi \lambda_{\max} \left\lceil \frac{N+1}{\sqrt[p]{g}} \right\rceil^2\right)} \quad \square
\end{aligned}$$

To the best of our knowledge, this is the first lower bound on the approximation error of Riemann theta functions. Note that in order to keep the lower bound (9) on the truncation error from blowing up as g increases, the truncation parameter N should grow at least proportionally to $\sqrt[p]{g}$.

The previous result did not cover the cases $p \in \{1, \infty\}$. The next proposition provides a weaker but more lower general bound on the truncation error, which shows that it must grow at least linearly in the genus g.

Proposition 2. *Let the index set be $\mathcal{N}^g = \mathcal{I}^g(p, N)$ for any $p \in \mathbb{N} \cup \{\infty\}$. The truncation error is then lower bounded as*

$$\left|\theta(\mathbf{z} \mid \boldsymbol{\Omega}) - \hat{\theta}(\mathbf{z} \mid \boldsymbol{\Omega})\right| \geq g \mathrm{e}^{-\pi \lambda_{\max} (N+1)^2}$$

for $\mathbf{z} = \mathrm{i}\mathbb{R}_{g\times 1}$ whenever $\Re\{\boldsymbol{\Omega}\} = \mathbf{0}_g$, where $\lambda_{\max}$ denotes the largest eigenvalue of $\Im\{\boldsymbol{\Omega}\}$.

Proof. Let $\mathcal{J}^g(N+1)$ denote the index set that contains all $\mathbf{n} \in \mathbb{Z}^g$ for which exactly one element is non-zero, and this element is given by

$$n_k = -\mathrm{sign}\{\Im\{z_k\}\}(N+1).$$

Then

$$\mathbf{n} \in \mathcal{J}^g(N+1) \Rightarrow \|\mathbf{n}\|_p = N+1 \Rightarrow \mathbf{n} \notin \mathcal{I}^g(p,N). \tag{11}$$

The truncation error thus satisfies

$$\begin{aligned}
\left|\theta(\mathbf{z} \mid \boldsymbol{\Omega}) - \hat{\theta}(\mathbf{z} \mid \boldsymbol{\Omega})\right| &= \left| \sum_{\mathbf{n} \notin \mathcal{I}^g(p,N)} c_{\mathbf{n}} \exp(2\pi \mathrm{i} \mathbf{n} \cdot \mathbf{z}) \right| \\
&= \left| \sum_{\mathbf{n} \notin \mathcal{I}^g(p,N)} c_{\mathbf{n}} \exp(2\pi \mathrm{i} \mathbf{n} \cdot (\mathrm{i}\Im\{\mathbf{z}\})) \right| \\
&= \left| \sum_{\mathbf{n} \notin \mathcal{I}^g(p,N)} \underbrace{c_{\mathbf{n}}}_{\geq 0} \underbrace{\exp(-2\pi \mathbf{n} \cdot \Im\{\mathbf{z}\})}_{\geq 0} \right| \\
&= \sum_{\mathbf{n} \notin \mathcal{I}^g(p,N)} c_{\mathbf{n}} \exp(-2\pi \mathbf{n} \cdot \Im\{\mathbf{z}\}) \\
(11) \quad &\geq \sum_{\mathbf{n} \in \mathcal{J}^g(N+1)} c_{\mathbf{n}} \mathrm{e}^{2\pi \min_k |\Im\{z_k\}|(N+1)} \\
(\mathbf{n} \cdot \Im\{\boldsymbol{\Omega}\}\mathbf{n} \leq \lambda_{\max} \|\mathbf{n}\|_2^2) \quad &\geq \sum_{\mathbf{n} \in \mathcal{J}^g(N+1)} \mathrm{e}^{2\pi \min_k |\Im\{z_k\}|(N+1)} \mathrm{e}^{-\pi \lambda_{\max} (N+1)^2} \\
(\text{since } \#\mathcal{J}^g(N+1) = g) \quad &= g \mathrm{e}^{2\pi \min_k |\Im\{z_k\}|(N+1)} \mathrm{e}^{-\pi \lambda_{\max} (N+1)^2} \\
&\geq g \mathrm{e}^{-\pi \lambda_{\max} (N+1)^2} \quad \square
\end{aligned}$$

The next proposition finally provides a similar bound for the hyperbolic cross.

Proposition 3. *Let the index set be $\mathcal{N}^g = \mathcal{H}^g(N)$. The truncation error is then lower bounded as*

$$\left|\theta(\mathbf{z} \mid \mathbf{\Omega}) - \hat{\theta}(\mathbf{z} \mid \mathbf{\Omega})\right| \geq g\mathrm{e}^{-\pi\lambda_{\max}\left\lceil\frac{N+1}{2}\right\rceil^2}$$

at $\mathbf{z} = \mathrm{i}\mathbb{R}_{g\times 1}$ whenever $\Re\{\mathbf{\Omega}\} = \mathbf{0}_g$, where $\lambda_{\max}$ denotes the largest eigenvalue of $\Im\{\mathbf{\Omega}\}$.

Proof. Let $\mathcal{J}^g(N+1)$ denote the index set that contains all $\mathbf{n} \in \mathbb{Z}^g$ for which exactly one element is non-zero, and this element is given by

$$n_k = -\mathrm{sign}\{\Im\{z_k\}\}\left\lceil\frac{N+1}{2}\right\rceil.$$

Then

$$\begin{aligned}\mathbf{n} \in \mathcal{J}^g(N+1) &\Rightarrow \prod_{j=1}^{g}\max\left(1, 2|n_j|\right) \geq N+1\\ &\Rightarrow \mathbf{n} \notin \mathcal{H}^g(N).\end{aligned} \tag{12}$$

Following the same steps as in the proof of Prop. 2, we arrive at the lower bound on the truncation error for $\mathcal{H}^g$. □

3.2. An upper bound for $\mathcal{I}^g(\infty, N)$

The lower bounds in the previous subsection have shown that the truncation parameter N in general has to grow with the genus g if small truncation errors are desired. We are now investigating upper bounds. Upper bounds for the truncation errors of transformed Riemann theta functions that are summed over ellipsoids are provided in [25, Thm. 3], [28, Thm. 3.1]. Upper bounds with respect to the hyperbolic cross $\mathcal{H}^g(N)$ and $\mathcal{I}^g(1, N)$ are provided in [45, Ch. 8.1] for the case $\mathbf{z} \in \mathbb{R}^g$, but the influence of the genus on these bounds is unfortunately not investigated.

The following proposition, which is our next contribution, shows that the truncation error can be bounded independently of the genus for the hypercube if the truncation parameter grows slightly faster than $\Omega(\sqrt{g})$ for real $\mathbf{z}$, or $\Omega(g)$ for non-real $\mathbf{z}$ where Ω notation provides the asymptotic lower bound. This case is later of special interest since it is possible to evaluate the truncated Riemann theta function fast over the hypercube if the coefficient tensor is in some sense low rank.

Proposition 4. *Let the index set be $\mathcal{I}^g(\infty, N)$ and fix any $\delta \in (0, 1)$ and $a > 0$. Then there exists a constant $c > 0$ (independent of g and N) such that the truncation error is upper bounded as*

$$\left|\theta(\mathbf{z} \mid \mathbf{\Omega}) - \hat{\theta}(\mathbf{z} \mid \mathbf{\Omega})\right| \leq \sum_{k=N+1}^{\infty}\mathrm{e}^{-(\pi/2)\|\Im\{\mathbf{\Omega}\}^{-1}\|_p^{-1}k^2}$$

whenever $cN^{2-\delta} \geq g$ and $\Im\{\mathbf{z}\} = \mathbf{0}_{g\times 1}$. If $\|\Im\{\mathbf{z}\}\|_\infty \leq a$, the same bound holds whenever $cN \geq g$ (for a different $c > 0$).

Proof. Recall that from the proof of Theorem 1 that

$$\begin{aligned}&\left|\theta(\mathbf{z} \mid \mathbf{\Omega}) - \hat{\theta}(\mathbf{z} \mid \mathbf{\Omega})\right|\\ &\leq \sum_{k=N+1}^{\infty}\mathrm{e}^{g\log(2k+1)+2\pi\|\Im\{\mathbf{z}\}\|_\infty gk-\pi\|\Im\{\mathbf{\Omega}\}^{-1}\|_p^{-1}k^2}.\end{aligned}$$

We first consider the case $g \leq cN^{2-\delta}$ and $\Im\{\mathbf{z}\} = \mathbf{0}_{g\times 1}$. The constants are then chosen as $M := \max_{x>0}\frac{\log(1+2x)}{x^\delta} > 0$, which is finite because $x^{-\delta}\log(1+2x)$ is continuous and converges to zero for both $x \to 0$ and $x \to \infty$, and $c := \frac{\pi\|\Im\{\mathbf{\Omega}\}^{-1}\|_p^{-1}}{2M} > 0$. The bound on the truncation error becomes

$$\begin{aligned}&\left|\theta(\mathbf{z} \mid \mathbf{\Omega}) - \hat{\theta}(\mathbf{z} \mid \mathbf{\Omega})\right|\\ &\leq \sum_{k=N+1}^{\infty}\mathrm{e}^{cN^{2-\delta}\log(2k+1)-\pi\|\Im\{\mathbf{\Omega}\}^{-1}\|_p^{-1}k^2}\\ &\leq \sum_{k=N+1}^{\infty}\mathrm{e}^{ck^{2-\delta}\log(2k+1)-\pi\|\Im\{\mathbf{\Omega}\}^{-1}\|_p^{-1}k^2}\\ &= \sum_{k=N+1}^{\infty}\mathrm{e}^{\left(ck^{-\delta}\log(2k+1)-\pi\|\Im\{\mathbf{\Omega}\}^{-1}\|_p^{-1}\right)k^2}\end{aligned}$$

$$\leq \sum_{k=N+1}^{\infty} \mathrm{e}^{\left(cM-\pi\|\Im\{\boldsymbol{\Omega}\}^{-1}\|_p^{-1}\right)k^2}$$
$$= \sum_{k=N+1}^{\infty} \mathrm{e}^{-(\pi/2)\|\Im\{\boldsymbol{\Omega}\}^{-1}\|_p^{-1}k^2}.$$

In the case $g \leq cN$ and $\Im\{\mathbf{z}\} \neq \mathbf{0}_{g\times 1}$, we instead choose $M := \max_{x>0} \frac{\log(1+2x)}{x} > 0$, $c := \frac{\pi\|\Im\{\boldsymbol{\Omega}\}^{-1}\|_p^{-1}}{2(M+2\pi a)} > 0$. We arrive at the same bound as before:

$$\left|\theta(\mathbf{z} \mid \boldsymbol{\Omega}) - \hat{\theta}(\mathbf{z} \mid \boldsymbol{\Omega})\right|$$
$$\leq \sum_{k=N+1}^{\infty} \mathrm{e}^{cN\log(2k+1)+2\pi\|\Im\{\mathbf{z}\}\|_\infty cNk-\pi\|\Im\{\boldsymbol{\Omega}\}^{-1}\|_p^{-1}k^2}$$
$$\leq \sum_{k=N+1}^{\infty} \mathrm{e}^{\left(cM+2\pi ac-\pi\|\Im\{\boldsymbol{\Omega}\}^{-1}\|_p^{-1}\right)k^2}$$
$$\leq \sum_{k=N+1}^{\infty} \mathrm{e}^{-(\pi/2)\|\Im\{\boldsymbol{\Omega}\}^{-1}\|_p^{-1}k^2}. \quad \square$$

3.3. Discussion

In this section, we have found that the truncation parameter N needed to achieve a certain truncation error (8) increases with the genus g. *Therefore, the complexity of numerical methods for the large genus regime has to scale not only well directly in the genus g, but also in the truncation parameter N!* The exact behavior of the error depends on the choice of the index set $\mathcal{N}^g(N)$. We have shown that for the hypersphere $\mathcal{I}^g(2, N)$, N has in general to grow *at least* as $\sqrt{g}$ for real $\mathbf{z}$. Since we have to sum approximately $V_g(N) \sim \frac{1}{\sqrt{g\pi}}\left(\frac{2\pi\mathrm{e}}{g}\right)^{g/2} N^g$ terms for the naive evaluation of the truncated Riemann function, we see that this approach quickly becomes infeasible as the genus g increases. For the hypercube $\mathcal{I}^g(\infty, N)$, we have shown that the truncation parameter N has to grow *at most* slightly faster than $\sqrt{g}$ for real $\mathbf{z}$. Naive evaluation of the truncated Riemann theta function requires us to sum $(2N+1)^g$ terms, which also becomes quickly infeasible. *In [39], it was shown that multi-dimensional Fourier series such as the Riemann theta function can be summed efficiently over the hypercube if the coefficient tensor is approximated well by a low-rank tensor-train, even if the dimension g and the truncation parameter N are both large. Therefore, we will investigate this case further in the next section.*

The behavior of the hyperbolic cross is less clear at the moment. We will later investigate its performance numerically.

4. Summing over hypercubes using tensor trains and scaling and squaring

The truncated Riemann theta function $\hat{\theta}(\mathbf{z} \mid \boldsymbol{\Omega})$ in (2) over a hypercube can be represented as the inner product between the tensor $\underline{\mathbf{C}}$ in (3) and the tensor $\mathrm{e}^{2\pi\mathrm{i}\mathbf{n}\cdot\mathbf{z}}$. If the tensor $\underline{\mathbf{C}}$ has a low rank approximation in the tensor-train format, the complexity of computing the Riemann theta function value will be low since the tensor-train rank of the second tensor is one [52, Sec. 4.2]. The tensor $\underline{\mathbf{C}}$ is the point-wise exponential of the tensor formed from $\mathrm{i}\pi\,\mathbf{n}\cdot\boldsymbol{\Omega}\mathbf{n}$, which can be represented exactly in the tensor-train format (see Appendix B). The most common approach to approximate the pointwise exponential of a tensor is the scaling and squaring method. In our case, this means that the truncated Riemann theta function is approximated with

$$\tilde{\theta}(\mathbf{z} \mid \boldsymbol{\Omega}) = \sum_{\mathbf{n}\in\mathcal{I}^g(\infty,N)} \tilde{c}_{\mathbf{n}}\mathrm{e}^{2\pi\mathrm{i}\mathbf{n}\cdot\mathbf{z}}, \tag{13}$$

where the coefficient tensor $\underline{\tilde{\mathbf{C}}}$ with terms

$$\tilde{c}_{\mathbf{n}} := \left[\sum_{k=0}^{K-1} \frac{1}{k!}\left(\frac{q_{\mathbf{n}}}{s}\right)^k\right]^s, \quad q_{\mathbf{n}} := \pi\,\mathrm{i}\mathbf{n}\cdot\boldsymbol{\Omega}\mathbf{n}, \quad K, s \in \mathbb{N},$$

is an approximation of the true coefficient tensor $\underline{\mathbf{C}}$ with terms

$$c_{\mathbf{n}} = \mathrm{e}^{q_{\mathbf{n}}} = [\mathrm{e}^{q_{\mathbf{n}}/s}]^s = \left[\sum_{k=0}^{\infty} \frac{1}{k!}\left(\frac{q_{\mathbf{n}}}{s}\right)\right]^s,$$

in tensor-train format. The scaling and squaring approach is easy to implement and has been proven to work well for matrices [56]. It has been used for computing the elementwise exponential of tensor-trains and other tensor formats [57,58]. It is also the method implemented in the `tt_exp` function in the TT-toolbox [59].

In this section, we analyze the applicability of the scaling and squaring method for computing the tensor-train approximation of the tensor $\underline{\mathbf{C}}$ in (3). For all its merits we nevertheless show in this section that it is not suitable for our problem.

4.1. Complexity analysis for fixed K and s

We now provide a lower bound for the computational cost of evaluating $\tilde{\theta}(\mathbf{z} \mid \boldsymbol{\Omega})$ using the efficient inner product approach discussed above. That is, $\tilde{\theta}(\mathbf{z} \mid \boldsymbol{\Omega})$ is computed as the inner product between the tensor $\underline{\tilde{\mathbf{C}}}$ and the tensor with terms $\mathrm{e}^{2\pi \mathrm{i}\mathbf{n}\cdot\mathbf{z}}$ using the inner product algorithm in [52, Sec. 4.2] for tensor trains. (A full description is provided in Appendix B.) We start by noticing that the tensor $\underline{\tilde{\mathbf{C}}}$ has the following special diagonal representation in the tensor train format.

Lemma 2. *Let $\hat{g} := \frac{g^2+g}{2}$, $R := \frac{\hat{g}^K-1}{\hat{g}-1}$, and $\hat{R} := R^s$. The terms $\tilde{c}_{\mathbf{n}} = p_{\mathbf{n}}^s$ of the tensor $\underline{\tilde{\mathbf{C}}}$ have the tensor train representation*

$$\tilde{c}_{\mathbf{n}} = \left(\mathbf{P}_{n_1}^{(1)}\right)^{\otimes s} \left(\mathbf{P}_{n_2}^{(2)}\right)^{\otimes s} \cdots \left(\mathbf{P}_{n_g}^{(g)}\right)^{\otimes s},$$

where the inner cores are diagonal $\hat{R} \times \hat{R}$ matrices, and $\mathbf{A}^{\otimes s} := \mathbf{A} \otimes \cdots \otimes \mathbf{A}$ (s times) denotes the s-fold Kronecker product of a matrix with itself.

Proof. See Appendix B. □

In the proof of the lemma, it is shown that the rank of the tensor $\underline{\mathbf{Q}}$ is not larger than R. Since $\mathrm{rank}(\mathbf{A}^{\otimes s}) = \mathrm{rank}(\mathbf{A})^s$, the rank of $\underline{\tilde{\mathbf{C}}}$ therefore cannot be larger than $\hat{R} = R^s$. In general, the rank of $\underline{\mathbf{Q}}$ is equal to R. In that case, the rank of $\underline{\tilde{\mathbf{C}}}$ will be equal to $\hat{R}$. Otherwise, $\hat{R}$ provides an upper bound on the rank of $\underline{\tilde{\mathbf{C}}}$.

Proposition 5. *The computational cost of evaluating $\tilde{\theta}(\mathbf{z} \mid \boldsymbol{\Omega})$ using the standard tensor-train inner product algorithm [52, Sec. 4.2] applied to the diagonal representation in Lemma 1 is lower bounded by $\Omega\left((g-1)(2N+1)sR^s + (2N+1)R^s\right)$.*

Proof. See Appendix B. □

Note that this lower bound grows exponentially with s even for small parameter values such as $R = 2$ and $N = 1$.

4.2. Complexity analysis for a given error bound

Scaling and squaring can be implemented in different ways, where the freedom mostly lies in the choice of K and s. These parameters should be chosen such that the approximation error is below a given bound. Here we consider a simple strategy for choosing K and s that is similar to the one used in older versions of MATLAB (with Padé approximations instead of Taylor expansions) [60]. The Taylor polynomial

$$P_K(x) = \sum_{k=0}^{K-1} \frac{x^k}{k!} \tag{14}$$

is a good approximation of $\exp(x)$ at $x = 0$. The worst case approximation error for $|x| < 1$ can be made arbitrarily small by increasing K. Once such a K is found, the argument z has to be rescaled such that $|z/s| < 1$. Since $\mathrm{e}^z = [\mathrm{e}^{z/s}]^s$, the terms of the coefficient tensor $\tilde{c}_{\mathbf{n}} = [P_K(q_{\mathbf{n}}/s)]^s$ approximate $c_{\mathbf{n}}$. The strategy for choosing the parameters therefore is as follows.

- Choose K large enough so that $|\exp(x) - P_K(x)| \leq \epsilon$ for all $|x| < 1$, where $\epsilon > 0$ is a error parameter [61].
- Choose s large enough so that $|z/s| < 1 \; \forall \, z \in Z \subset \mathbb{C}$.

The following proposition provides a lower bound on the computational complexity of the scaling and squaring method if K and s are chosen in this way.

Proposition 6. *If K and s are chosen as above and we have $\Re\{\boldsymbol{\Omega}\} = \mathbf{0}_g$, then the numerical complexity of evaluating $\tilde{\theta}(\mathbf{z} \mid \boldsymbol{\Omega})$ using the standard tensor-train inner product algorithm applied to the diagonal representation in Lemma 1 is lower bounded by $\Omega\left((g-1)(2N+1)\pi\lambda_{\min} g N^2 R^{\pi\lambda_{\min} g N^2}\right)$ with $\lambda_{\min}$ being the smallest eigenvalue of $\Im\{\boldsymbol{\Omega}\}$.*

Proof. For the Riemann theta function we have $z = q_{\mathbf{n}} = \pi \mathrm{i}\mathbf{n} \cdot \boldsymbol{\Omega}\mathbf{n}$. For the index vector $\mathbf{n} = [N \; N \; \ldots \; N]^T$, we have $z \leq -\pi\lambda_{\min} g N^2$. Thus, to have $|z/s| < 1$ we need $s \geq \pi\lambda_{\min} g N^2$. Hence, even for a fixed N, s would grow linearly in g. (Recall from Proposition 2 that N actually has to grow with g to keep the error of the truncated Riemann theta sum bounded.) Application of Proposition 3 with $s \geq \pi\lambda_{\min} g N^2$ now provides the following lower bound on the complexity, $\Omega\left((g-1)(2N+1)\pi\lambda_{\min} g N^2 R^{\pi\lambda_{\min} g N^2}\right)$. □

The lower bound on the computational complexity thus grows exponentially in g even if there are lower and upper bounds on the eigenvalues of $\Im\{\mathbf{\Omega}\}$ that are independent of g. Since we sum over $(2N+1)^g$ coefficients in the truncated Riemann theta sum, ϵ should actually decrease with g to keep the error introduced by the approximation of the coefficient tensor fixed. The constant K is inversely proportional to ϵ and hence K increases with g increasing the computational complexity even further. Thus, scaling and squaring is not suited for the numerical computation of high dimensional Riemann theta functions.

We finally remark that in practice, a rounding procedure is applied to the tensor-trains to reduce their ranks (like in the implementation `tt_exp` in the TT-toolbox [59]). We nevertheless observed in numerical experiments that even with rounding, scaling and squaring was ill-suited for our purposes.

5. Numerical experiments

In this section, several methods for computing truncated Riemann functions (2) are investigated numerically and compared with respect to accuracy, for both low and high genus cases. First, the choice of algorithms is motivated. Then, the background of the numerical experiments is discussed. Finally, the results are presented.

5.1. Choice of benchmark algorithms

We so far considered two different approaches to approximate the Riemann theta function (1). The first approach is the naive computation of the truncated Riemann theta function (2) over the index sets $\mathcal{N}^g(N) = \mathcal{I}^g(p, N)$ and $\mathcal{H}^g(N)$, that were introduced in Section 2. Even for $N = 2$, the number of elements in $\mathcal{I}^g(p, N)$ grows exponentially in the genus g for any p. Hence, naive evaluation of (1) can be applied when the genus g is small for these index sets, but it is not well-suited for computing high-genus solutions. For the index set $\mathcal{I}^g(\infty, N)$, we will refer to this approach as the `Hcube` algorithm. When the Riemann matrix is diagonal, the multi-dimensional Riemann theta function can be computed efficiently using multiple one-dimensional Riemann theta functions. We will refer to the algorithm that neglects the off-diagonal elements in order to exploit this fact to facilitate fast computation as `Diag_approx` in the following. This algorithm will allow us to verify that the Riemann matrices used in the examples cannot be approximated well with just the diagonal part of the matrix. We furthermore know from Lemma 1 that the complexity of naive computation over the hyperbolic cross $\mathcal{N}^g(N) = \mathcal{H}^g(N)$ grows at a slower rate. Therefore, it might be better suited for high genus cases. We will refer to this algorithm as `HC` in the following.

The second approach to approximate the Riemann theta function is to replace the true coefficient tensor in the truncated Riemann function (2) with an approximation in the tensor-train format. The resulting approximation (13) of the Riemann theta function can then be evaluated with low numerical complexity since it is an inner product between low-rank tensor-trains. There are different ways to obtain the tensor-train approximation of the coefficient tensor. In the previous section, we showed that the popular scaling and squaring approach is not a good choice when then the genus g is high. However, there are other methods for this task. The first method we consider is from the paper [54] and will be referred to as `TT_cross`. Specifically, we employ the `dmrg_cross` routine from the TT-toolbox [59]. We also investigate the performance of the `funcrs` routine from the toolbox, for which unfortunately no reference is provided. This algorithm will be referred to as `TT_funcrs` in the following.

5.2. Generation of test data

Riemann theta functions play, as was already mentioned in the introduction, a fundamental role in the area of nonlinear Fourier analysis [1]. The (quasi-)periodic solutions of many integrable systems can be approximated arbitrarily well using so-called hyperelliptic (or also finite-gap, finite-genus or finite-band) solutions, which have especially simple closed-form representations that involve the Riemann theta function. For the numerical demonstrations we make use of the hyperelliptic solutions of the normalized KdV equation

$$u_t + u_{xxx} + 6uu_x = 0 \tag{15}$$

and the normalized NLS equation

$$\mathrm{i}q_x + q_{tt} + 2|q|^2 q = 0. \tag{16}$$

The hyperelliptic solutions of the KdV equation are given in terms of the Riemann theta function as [51, Eq. (19)]

$$u(x,t) = 2\frac{\partial^2}{\partial x^2}\log\theta(\mathbf{z} \mid \mathbf{\Omega}), \quad \mathbf{z} = \mathbf{k}x - \boldsymbol{\omega}t + \boldsymbol{\phi}, \tag{17}$$

where $\mathbf{\Omega}$ is typically called the period matrix in the literature, and $\mathbf{k}$, $\boldsymbol{\omega}$ and $\boldsymbol{\phi}$ are constant real vectors. The matrix $\mathbf{\Omega}$ is of size $g \times g$, where g is as before called the genus. The parameters $\mathbf{\Omega}$, $\mathbf{k}$, $\boldsymbol{\omega}$ and $\boldsymbol{\phi}$ cannot be chosen freely. The theory of integrable systems shows that the parameters provide a valid solution if and only they can be derived from a so-called hyperelliptic Riemann surface, which is a special kind of one-dimensional complex manifold. In our examples,

Table 1
Error in computing $\theta(\mathbf{z} \mid \mathbf{\Omega})$ for a genus-2 solution of the KdV equation.

Algorithm	N	#terms	Run time	$E_1 = \max_{\mathbf{z}}\{E(\mathbf{z})\}$	$E_2 = \text{median}_{\mathbf{z}}\{E(\mathbf{z})\}$
`Hcube`	4	81	0.1181 s	0 by definition	0 by definition
`TT_funcrs`	4	125* $1 \times 9 \times 1$	3.4557* s	$\max\{E_1\}$ over 20 runs: 7.3286×10^{-13} $\text{median}\{E_1\}$ over 20 runs: 7.3286×10^{-13} $\min\{E_1\}$ over 20 runs: 7.3286×10^{-13}	$\max\{E_2\}$ over 20 runs: 5.1681×10^{-13} $\text{median}\{E_2\}$ over 20 runs: 5.1681×10^{-13} $\min\{E_2\}$ over 20 runs: 5.1681×10^{-13}
`TT_cross`	4	101* $1 \times 7 \times 1$	3.2160* s	$\max\{E_1\}$ over 20 runs: 7.3319×10^{-13} $\text{median}\{E_1\}$ over 20 runs: 7.3313×10^{-13} $\min\{E_1\}$ over 20 runs: 7.3308×10^{-13}	$\max\{E_2\}$ over 20 runs: 5.1692×10^{-13} $\text{median}\{E_2\}$ over 20 runs: 5.1686×10^{-13} $\min\{E_2\}$ over 20 runs: 5.1675×10^{-13}
`Diag approx`	4	9	0.0951 s	8.3307×10^{-2}	5.3869×10^{-2}
`HC`	28	121	0.1712 s	8.1535×10^{-13}	5.7232×10^{-13}
`HC`	20	81	0.1261 s	2.4035×10^{-9}	1.5476×10^{-9}
`HC`	14	49	0.0777 s	6.2153×10^{-6}	4.3970×10^{-6}
`HC`	3	5	0.0276 s	8.5271×10^{-2}	5.9622×10^{-2}
`HC`	1	2	0.0076 s	3.3724×10^{-1}	1.1661×10^{-1}

we computed the parameters numerically using the methods given in [51, Section 3.2], [8, Section 14.4]. The hyperelliptic Riemann surfaces that we used are specified by their so-called branch points, which we provide for each example.

The finite-genus solutions of the NLS equation are given by

$$q(x,t) = K_0 \frac{\theta\left(\frac{\mathbf{z}_-}{2\pi} \mid \mathbf{\Omega}\right)}{\theta\left(\frac{\mathbf{z}_+}{2\pi} \mid \mathbf{\Omega}\right)} e^{i\omega_0 t + ik_0 z}, \quad \mathbf{z}_\pm = \mathbf{k}x + \boldsymbol{\omega}t + \boldsymbol{\delta}_\pm, \tag{18}$$

where $\mathbf{\Omega}, \mathbf{k}, \boldsymbol{\omega}, \boldsymbol{\delta}_\pm, k_0, \omega_0$ and K_0 are again constant parameters [6]. The vectors $\boldsymbol{\delta}_\pm$ can be complex valued and hence arguments to the Riemann theta function $\mathbf{z}_\pm$ can be complex valued. As before, these parameters have to be obtained from a hyperelliptic Riemann surface, which is specified by branch points. In our test we used the period matrix and parameter vectors provided in [6, Table II].

In the following we will look at three scenarios. In the first case we will assess the accuracy of the algorithms for a genus-2 and a genus-6 solution of the KdV equation. In the second case we will test the accuracy in computing a genus-3 solution of the NLS equation for which the Riemann theta function has complex arguments. In the third case, we compute solutions of the KdV equation up to the very high genus of 60. To the best of our knowledge, a successful computation of non-trivial Riemann theta functions for genera this large has never been reported in the literature before.

5.3. Accuracy for genus-2 and genus-6 KdV solutions

In this scenario, we evaluate the accuracy of computing the Riemann theta function for $\mathbf{z} \in \mathbb{R}^g$ with $\mathbf{z}$ as defined in (17). The period matrices $\mathbf{\Omega}$ were derived from the finite-genus KdV solutions in [51, Section 4]. First we have the genus-2 solution of the KdV equation with branch points [0 0.5 1 1.5 5]. We compute the Riemann theta function for 16384 arguments $\mathbf{z}$ corresponding to the grid formed by 128 equispaced values of x in $[0, 4]$ and 128 equispaced values of t in $[-0.5, 0.5]$. Secondly we have the genus-6 solution with branch points [0 0.5 2 2.5 4 4.5 6 6.5 8 8.5 10 10.5 12]. We again compute the Riemann theta function for 16384 arguments $\mathbf{z}$ corresponding to the grid formed by 128 equispaced values of x in $[0, 3]$ and 128 equispaced values of t in $[-0.3, 0.3]$. The values of x and t chosen are sufficiently representative for the respective solutions. The phases $\boldsymbol{\phi}$ are set to 0. We then define the point-wise error as

$$E(\mathbf{z}) = \left| \hat{\theta}\left(\mathbf{z} \mid \mathbf{\Omega}\right) - \tilde{\theta}\left(\mathbf{z} \mid \mathbf{\Omega}\right) \right| \tag{19}$$

where $\hat{\theta}\left(\mathbf{z} \mid \mathbf{\Omega}\right)$ is the reference value and $\tilde{\theta}\left(\mathbf{z} \mid \mathbf{\Omega}\right)$ is the value computed by the other methods. The reference value $\hat{\theta}\left(\mathbf{z} \mid \mathbf{\Omega}\right)$ is computed using the `Hcube` algorithm. The truncation parameter for each example is fixed by starting with $N = 1$ and increasing it until

$$\max_{\mathbf{z}} \left| \hat{\theta}\left(\mathbf{z} \mid \mathbf{\Omega}\right)|_{N+1} - \hat{\theta}\left(\mathbf{z} \mid \mathbf{\Omega}\right)|_N \right| < 1 \times 10^{-14}, \tag{20}$$

where $\hat{\theta}\left(\mathbf{z} \mid \mathbf{\Omega}\right)|_N$ is the Riemann theta function calculated by the `Hcube` algorithm with the truncation parameter N. If Eq. (20) is satisfied for $N = \hat{N}$, then the values $\hat{\theta}\left(\mathbf{z} \mid \mathbf{\Omega}\right)|_{\hat{N}+1}$ are set as the reference values in (19). Note that we can compute the solution classically only because the genera are low in this example. For the algorithms `TT_cross` and `TT_funcrs`, we set the accuracy parameter of the coefficient tensor $\underline{\mathbf{C}}$ approximation to 10^{-12}.

In Table 1 we see the median and maximum of the error $E(\mathbf{z})$, the truncation parameter N and the number of terms in the summation used for the genus-2 solution. As the coefficient tensor approximation step in the `TT_cross` and

Table 2
Error in computing $\theta(\mathbf{z} \mid \boldsymbol{\Omega})$ for a genus-2 solution of the KdV equation (with Siegel transform).

Algorithm	N	#terms	Run time	$E_1 = \max_{\mathbf{z}}\{E(\mathbf{z})\}$	$E_2 = \text{median}_{\mathbf{z}}\{E(\mathbf{z})\}$
Hcube	4	81	0.2294 s	1.7763×10^{-15}	2.2216×10^{-16}
TT_funcrs	4	124* $1 \times 9 \times 1$	3.4185* s	$\max\{E_1\}$ over 20 runs: 1.4664×10^{-5} $\text{median}\{E_1\}$ over 20 runs: 1.4664×10^{-5} $\min\{E_1\}$ over 20 runs: 1.4664×10^{-5}	$\max\{E_2\}$ over 20 runs: 6.6613×10^{-15} $\text{median}\{E_2\}$ over 20 runs: 6.6613×10^{-15} $\min\{E_2\}$ over 20 runs: 6.6613×10^{-15}
TT_cross	4	104* $1 \times 7 \times 1$	3.4777* s	$\max\{E_1\}$ over 20 runs: 2.9409×10^{-5} $\text{median}\{E_1\}$ over 20 runs: 2.9409×10^{-5} $\min\{E_1\}$ over 20 runs: 2.9409×10^{-5}	$\max\{E_2\}$ over 20 runs: 3.5527×10^{-15} $\text{median}\{E_2\}$ over 20 runs: 3.5527×10^{-15} $\min\{E_2\}$ over 20 runs: 3.5527×10^{-15}
Diag approx	4	9	0.3276 s	2.2379	4.3191×10^{-2}
HC	28	121	0.2498 s	3.6859×10^{-14}	2.2204×10^{-16}
HC	20	81	0.2219 s	3.6859×10^{-14}	2.2204×10^{-16}
HC	14	49	0.2160 s	1.6653×10^{-5}	2.9109×10^{-8}
HC	3	5	0.2036 s	3.5881×10^{-1}	2.5139×10^{-2}
HC	1	2	0.1996 s	7.7519×10^{-1}	2.7817×10^{-1}

Table 3
Error in computing $\theta(\mathbf{z} \mid \boldsymbol{\Omega})$ for a genus-6 solution of the KdV equation.

Algorithm	N	#terms	Run time	$E_1 = \max_{\mathbf{z}}\{E(\mathbf{z})\}$	$E_2 = \text{median}_{\mathbf{z}}\{E(\mathbf{z})\}$
Hcube	5	1771561	2423.8738 s	0 by definition	0 by definition
TT_funcrs	5	69005* $1 \times 12 \times 40 \times 73 \times 34 \times 11 \times 1$	26.4751 s	$\max\{E_1\}$ over 20 runs: 1.3615×10^{-11} $\text{median}\{E_1\}$ over 20 runs: 7.8933×10^{-12} $\min\{E_1\}$ over 20 runs: 3.6349×10^{-12}	$\max\{E_2\}$ over 20 runs: 1.8069×10^{-12} $\text{median}\{E_2\}$ over 20 runs: 1.2623×10^{-12} $\min\{E_2\}$ over 20 runs: 6.4602×10^{-13}
TT_cross	5	37904* $1 \times 9 \times 37 \times 42 \times 31 \times 9 \times 1$	22.4916 s	$\max\{E_1\}$ over 20 runs: 7.8628×10^{-9} $\text{median}\{E_1\}$ over 20 runs: 1.4163×10^{-9} $\min\{E_1\}$ over 20 runs: 1.2142×10^{-9}	$\max\{E_2\}$ over 20 runs: 8.7031×10^{-10} $\text{median}\{E_2\}$ over 20 runs: 3.4114×10^{-10} $\min\{E_2\}$ over 20 runs: 3.1115×10^{-10}
Diag_approx	5	11	0.2792 s	1.027 10	1.1549×10^{-1}
HC	1430	1753893	2406.2909 s	8.2058×10^{-11}	3.9893×10^{-11}
HC	715	615521	871.8843 s	5.9266×10^{-9}	2.3027×10^{-9}
HC	143	47353	69.5281 s	8.3491×10^{-6}	1.2015×10^{-6}
HC	15	545	1.2000 s	1.3741×10^{-1}	1.0142×10^{-2}

TT_funcrs algorithms is non-deterministic, we report the errors for both the algorithms over 20 runs. For the TT_cross and TT_funcrs algorithms, the number of terms corresponds to the number of non-zero terms in the tensor-train approximation of the coefficient tensor $\underline{\tilde{\mathbf{C}}}$. The * symbol over the number indicates that the value listed is the median value over 20 runs. Correspondingly, we report the maximum, median and minimum over the 20 runs of the median and maximum of the pointwise error $E(\mathbf{z})$. For the Diag_approx algorithm we use the same truncation parameter as the reference Hcube algorithm. For the HC algorithm we initially choose the truncation parameter N_{HC} such that the number of terms in the sum is close to that of the Hcube algorithm. We also run the HC algorithm with approximately 50 %, 10 % and 1 % of N_{HC} as the truncation parameter. The errors for the Diag_approx algorithm are high indicating that the period matrix has significant non-diagonal components. Both the TT_cross and TT_funcrs algorithms have very low errors and the variation over the multiple runs is very small. The error for the HC algorithm with N_{HC} as the truncation parameter is also low. However, it is higher than that of the tensor-train based methods. The tensor-train methods on the other hand use more terms. The errors of the HC algorithm increase slowly with decreasing truncation parameter.

In Table 2 we again show the errors for the genus-2 case. The only difference being the application of the Siegel transform Section 2.2 in all the algorithms. The process to calculate the truncation parameter is repeated as described previously. For this example the Siegel transform does not lead to a reduction in the truncation parameter. The main differences in between Tables 1 and 2 are the maximum errors for the tensor-train based methods. The maximum errors are significantly higher when the Siegel transform is applied. The median error for the HC algorithm decreases for higher values of N while the max error increases significantly for the Diag_approx algorithm.

In Tables 3 and Table 4 we have the results for the genus-6 example. Without the Siegel transform the truncation parameter was fixed to 5. The Diag_approx algorithm has high errors indicating the presence of significant non-diagonal terms. Both TT_cross and TT_funcrs algorithms have low errors while using significantly less number of terms compared to the classical Hcube algorithm. The HC algorithm also performs well but is less accurate when compared to the tensor-train based methods for the same number of terms. Application of the Siegel transform helps reduce the truncation parameter to 4 in Table 4. Both the TT_cross and TT_funcrs algorithms fail for this genus-6 example when the Siegel

Table 4
Error in computing $\theta(\mathbf{z} \mid \boldsymbol{\Omega})$ for a genus-6 solution of the KdV equation (with Siegel transform).

Algorithm	N	#terms	Run time	$E_1 = \max_{\mathbf{z}}\{E(\mathbf{z})\}$	$E_2 = \text{median}_{\mathbf{z}}\{E(\mathbf{z})\}$
`Hcube`	4	531441	313.9509 s	3.7750×10^{-14}	5.8950×10^{-15}
`TT_funcrs`	4	18138* $1 \times 10 \times 24 \times 34 \times 22 \times 9 \times 1$	11.7017 s	$\max\{E_1\}$ over 20 runs: 9.7672×10^{12} $\text{median}\{E_1\}$ over 20 runs: 1.7581×10^{12} $\min\{E_1\}$ over 20 runs: 6.0793×10^{11}	$\max\{E_2\}$ over 20 runs: 1.4351 $\text{median}\{E_2\}$ over 20 runs: 7.3642×10^{-1} $\min\{E_2\}$ over 20 runs: 2.3702×10^{-1}
`TT_cross`	4	13494* $1 \times 7 \times 21 \times 31 \times 19 \times 7 \times 1$	11.5948 s	$\max\{E_1\}$ over 20 runs: 1.3213×10^{13} $\text{median}\{E_1\}$ over 20 runs: 5.6749×10^{11} $\min\{E_1\}$ over 20 runs: 1.0188×10^{8}	$\max\{E_2\}$ over 20 runs: 5.6102×10^{-1} $\text{median}\{E_2\}$ over 20 runs: 1.0112×10^{-1} $\min\{E_2\}$ over 20 runs: 8.6124×10^{-5}
`Diag_approx`	4	9	0.7420 s	1.2994	5.0057×10^{-1}
`HC`	640	549113	301.8520 s	2.3065×10^{-7}	4.0635×10^{-14}
`HC`	320	188993	101.0471 s	1.6138×10^{-5}	3.5885×10^{-11}
`HC`	64	15241	7.7543 s	$6.153\,86 \times 10^{-3}$	2.4742×10^{-6}
`HC`	7	97	0.3319 s	1.0231	1.6241×10^{-1}

transform is applied. We attribute this to numerical ill-conditioning arising from the faster decay of the coefficients $c_{\mathbf{n}}$. It seems to become harder to approximate the tensor $\tilde{\underline{\mathbf{C}}}$ with a tensor-train as the range of magnitudes of the coefficients $c_{\mathbf{n}}$ increases. For the `HC` algorithm, the maximum errors are slightly higher however the median errors are lower when the Siegel transform is applied.

The number of terms listed in Tables 1–4 indicate the memory requirements of the algorithms and is also related to the computational complexity. The total computation cost for the `HC` algorithm is the sum of the cost for generating the index set $\mathcal{H}^g$ and the sum (2) over $\mathcal{H}^g$. Even with the recursive algorithm from [47, Sec. 2.6], the cost of generating the index set grows quickly with N and g. The cost of the summation depends on the specific implementation and can be made quite efficient using parallelized implementations. For the `TT_cross` and `TT_funcrs` algorithms the computation cost is divided into the cost of computing the tensor-train approximations and the cost of computing the inner-product. Given the rank of the tensor-train approximation of $\tilde{\underline{\mathbf{C}}}$, the computation cost of the inner-product can be estimated (see Appendix B or [52, Sec. 4.2]). The cost of computing the tensor-train approximation $\tilde{\underline{\mathbf{C}}}$ is however non-trivial due to the iterative nature of both the `dmrg_cross` and `funcrs` routines. The actual complexity of all three algorithms depends significantly on the specific implementations. Hence, we have chosen to avoid a detailed comparison of the computational complexity and instead provided only the number of terms as an indicator. Furthermore, they are also an indicator for the memory requirements of the algorithms, which in our experience has been the major limiting factor for larger genera.

5.4. Accuracy for genus-3 NLS solution

The solution of the NLS equation is given as the ratio of Riemann theta function values (18). We use the genus-3 example from [6, Fig. 3]. We did not have to compute the period matrix and parameter vectors from the branch points in this case since they are provided in [6, Table II]. The theta function values are computed for 16384 arguments $\mathbf{z}_{\pm}$ as defined in (18) for 128 equispaced points $x \in [-0.002, 0.002]$ and for 128 equispaced points $t \in [-0.08, 0.08]$. For both the Riemann theta functions in the numerator and denominator of (18), the truncation parameter was fixed to be 6 using the procedure described for the KdV solutions in the previous subsection. The tensor-train methods were run 20 times and the truncation parameter for the `HC` algorithm was set to have similar number of terms as the `Hcube` algorithm.

In Tables 5 and 6, we list the errors in computing the Riemann theta function with arguments $\mathbf{z}_-$ and the Riemann theta function with arguments $\mathbf{z}_+$. In this example the arguments $\mathbf{z}_-$ are complex valued while the arguments $\mathbf{z}_+$ are real valued. Both the tensor-train based algorithms `TT_cross` and `TT_funcrs` fail to compute the Riemann theta function values correctly. We suspect the reason to be the same ill-conditioning that we observed for the genus-6 KdV solution in the previous subsection. The `HC` algorithm appears to be moderately accurate even for lower number of terms. Application of the Siegel transform did not lead to a reduction in the truncation parameter. Hence we have not mentioned specific error values for the same. From the results for the NLS example and the KdV examples in the previous subsection, the `HC` algorithm emerges as a practical algorithm for computing the Riemann theta function.

5.5. Computing high genus KdV solutions

In this subsection we will test the ability of the `HC` algorithm to compute the Riemann theta function value for high number of dimensions. Unfortunately, both the `TT_funcrs` and `TT_cross` algorithms were limited by memory even for moderate number of dimensions. Hence we only test the `HC` algorithm in the following. As there are no non-trivial Riemann matrices for which the Riemann theta function value is known analytically and using the `Hcube` algorithm is not

Table 5
Error in computing $\theta\left(\frac{\mathbf{z}}{2\pi} \mid \boldsymbol{\Omega}\right)$ for a genus-3 solution of the NLS equation.

Algorithm	N	#terms	Run time	$E_1 = \max_{\mathbf{z}}\{E(\mathbf{z})\}$	$E_2 = \text{median}_{\mathbf{z}}\{E(\mathbf{z})\}$
`Hcube`	6	2197	2.7474 s	0 by definition	0 by definition
`TT_funcrs`	6	1918* $1 \times 10 \times 13 \times 1$	8.0290* s	$\max\{E_1\}$ over 20 runs: 1.4971 $\text{median}\{E_1\}$ over 20 runs: 3.9583×10^{-1} $\min\{E_1\}$ over 20 runs: 2.0053×10^{-3}	$\max\{E_2\}$ over 20 runs: 1.3031 $\text{median}\{E_2\}$ over 20 runs: 3.5048×10^{-1} $\min\{E_2\}$ over 20 runs: $1.355\,39 \times 10^{-3}$
`TT_cross`	6	1224* $1 \times 7 \times 11 \times 1$	4.4811* s	$\max\{E_1\}$ over 20 runs: 4.4819×10^{12} $\text{median}\{E_1\}$ over 20 runs: 2.1824×10^{7} $\min\{E_1\}$ over 20 runs: 2.0406×10^{-1}	$\max\{E_2\}$ over 20 runs: 9.9790×10^{11} $\text{median}\{E_2\}$ over 20 runs: 2.0426×10^{7} $\min\{E_2\}$ over 20 runs: 1.7638×10^{-1}
`Diag_approx`	6	13	0.1096 s	1.5084	1.1001
`HC`	112	2237	2.8121 s	3.7768×10^{-11}	2.9799×10^{-11}
`HC`	56	885	1.1673 s	9.3304×10^{-6}	9.3077×10^{-6}
`HC`	12	105	0.1419 s	3.4411×10^{-2}	3.0213×10^{-2}
`HC`	2	7	0.03720 s	1.5242	1.0956

Table 6
Error in computing $\theta\left(\frac{\mathbf{z}}{2\pi} \mid \boldsymbol{\Omega}\right)$ for a genus-3 solution of the NLS equation.

Algorithm	N	#terms	Run time	$E_1 = \max_{\mathbf{z}}\{E(\mathbf{z})\}$	$E_2 = \text{median}_{\mathbf{z}}\{E(\mathbf{z})\}$
`Hcube`	6	2197	2.6247 s	0 by definition	0 by definition
`TT_funcrs`	6	1918* $1 \times 10 \times 13 \times 1$	8.0483* s	$\max\{E_1\}$ over 20 runs: 1.4971 $\text{median}\{E_1\}$ over 20 runs: 3.9583×10^{-1} $\min\{E_1\}$ over 20 runs: 2.0053×10^{-3}	$\max\{E_2\}$ over 20 runs: 1.3031 $\text{median}\{E_2\}$ over 20 runs: 3.5048×10^{-1} $\min\{E_2\}$ over 20 runs: $1.355\,39 \times 10^{-3}$
`TT_cross`	6	1224* $1 \times 7 \times 11 \times 1$	4.4682* s	$\max\{E_1\}$ over 20 runs: 4.4819×10^{12} $\text{median}\{E_1\}$ over 20 runs: 2.1824×10^{7} $\min\{E_1\}$ over 20 runs: 2.0406×10^{-1}	$\max\{E_2\}$ over 20 runs: 9.9790×10^{11} $\text{median}\{E_2\}$ over 20 runs: 2.0426×10^{7} $\min\{E_2\}$ over 20 runs: 1.7638×10^{-1}
`Diag_approx`	6	13	0.1368 s	9.1129×10^{-2}	3.2034×10^{-2}
`HC`	112	2237	2.5279 s	4.8849×10^{-15}	1.3323×10^{-15}
`HC`	56	885	1.0758 s	8.9268×10^{-10}	3.9137×10^{-10}
`HC`	12	105	0.1381 s	3.6694×10^{-4}	2.5105×10^{-4}
`HC`	2	7	0.03814 s	9.1123×10^{-2}	3.2032×10^{-2}

feasible, we resort to an alternative approach to verify the correctness of the approximations. We use the hyperelliptic solutions of the KdV equations with the branch points $\lambda_j = 0.5j,\ j = 0, 1, \ldots, g$ for different values of the genus g. We would like to remark that to compute the period matrices $\boldsymbol{\Omega}$ reliably we had to use 1000 bits of precision. We accomplished this using the Julia programming language. For a given g and N the Riemann theta function value is computed for $x_j = -2.0518 + 0.0120j,\ j = 0, 1, \ldots, 340$ and $t = [-0.2460\ \ -0.2457\ \ -0.2455]$. The approximate solution $\tilde{u}(x, t)$ is computed using central-difference to calculate the derivatives in (17).

To quantify the error in the calculated solutions, we first compute approximations of the time derivative u_t and space derivatives u_x and u_{xxx} using central differences. We then calculate the relative error

$$E_r = \frac{\sqrt{\sum_i |\,\text{LHS}_i - \text{RHS}_i\,|^2}}{\sqrt{\sum_i 0.25(|\,\text{LHS}_i\,| + |\,\text{RHS}_i\,|)^2}}, \tag{21}$$

where $\text{LHS}_i = -u_t(x_i, t)$ and $\text{RHS}_i = u_{xxx}(x_i, t) + 6u(x_i, t)u_x(x_i, t)$. Due to the absence of the true values of LHS_i or RHS_i, we have used the mean value $0.5(|\,\text{LHS}_i\,| + |\,\text{RHS}_i\,|)$ as the reference value in the relative error. We discard the values at the boundaries for which the numerical derivative cannot be calculated correctly using central-difference. As an example, for the solution computed using the `HC` algorithm for $g = 30$ and $N = 4$, we plot $-u_t$ and $u_{xxx} + 6uu_x$ in Fig. 1. We can see that the lines almost overlap which indicates that the error is small and that the computed values do correspond to a solution of the KdV equation. In Fig. 2 we show the error E_r for the `HC` algorithm for varying values of g and N. The choices of g and N were limited by the available system memory. We can observe that the relative error is small even for high genus solutions. From our understanding, this is the first instance in literature where such high genus solutions have been computed. Additional details on the impact of N for the case $g = 60$ are provided in Table 7.

We can thus surmise that the `HC` algorithm is suited for computing remarkably high genus Riemann theta function with moderate accuracy. The tensor-train based algorithms `TT_funcrs` and `TT_cross` work well for low genus but do not scale well with the number of dimensions. We would like to remark that for both the tensor-train based algorithms and the `HC` algorithm, the truncation parameter N can be chosen independently for each dimension to further reduce the computation cost. The impact of such a choice would be a topic for future research.

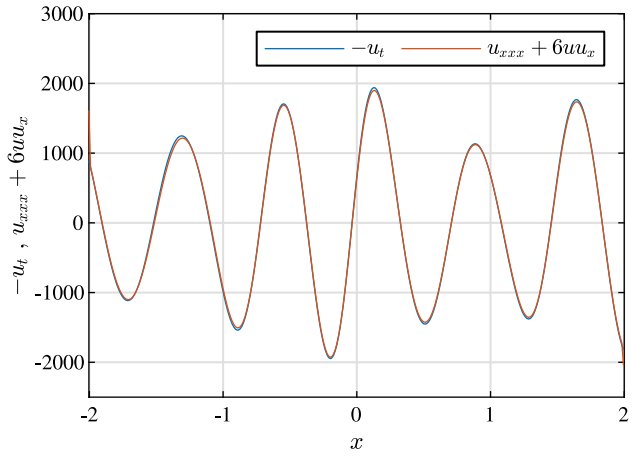

Fig. 1. Checking the genus-30 solution computed using the HC algorithm with $N = 4$. We can see that $-u_t \approx u_{xxx} + 6uu_x$ verifying that $u(x, t)$ is a good approximation of the finite-gap KdV solution.

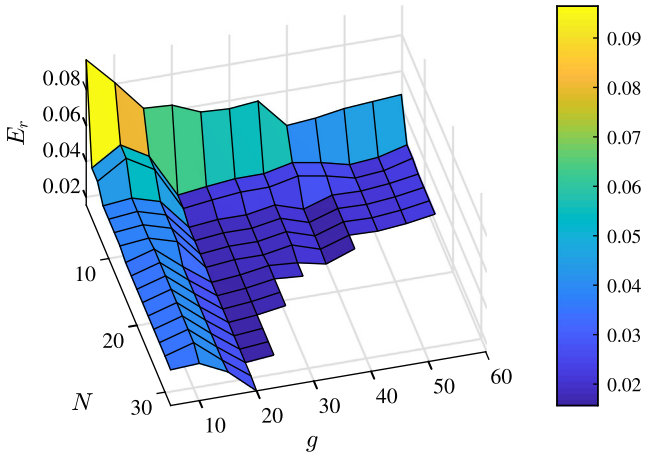

Fig. 2. The error E_r for KdV solutions computed using the HC algorithm for varying values of g and N.

Table 7
Error in computing $\theta(\mathbf{z} \mid \mathbf{\Omega})$ for a genus-60 solution of the KdV equation.

Algorithm	N	#terms	Run time	E_1	E_2
HC	2	121	0.0447 s	6.5482×10^{-3}	4.7278×10^{-2}
HC	4	7321	1.6434 s	2.9494×10^{-3}	2.6739×10^{-2}
HC	6	7441	1.7087 s	2.9494×10^{-3}	2.6739×10^{-2}
HC	8	295481	82.7293 s	2.9494×10^{-3}	2.5345×10^{-2}
HC	10	295601	82.8343 s	2.9494×10^{-3}	2.5345×10^{-2}
HC	12	309881	86.4405 s	2.9494×10^{-3}	2.5345×10^{-2}
Diag_approx	2	300	0.0728 s	1.3595×10^{-2}	1.0730×10^{-1}
Diag_approx	4	540	0.0931 s	1.3595×10^{-2}	1.0730×10^{-1}
Diag_approx	6	780	0.1107 s	1.3595×10^{-2}	1.0730×10^{-1}
Diag_approx	8	1020	0.1510 s	1.3595×10^{-2}	1.0730×10^{-1}
Diag_approx	10	1260	0.1486 s	1.3595×10^{-2}	1.0730×10^{-1}
Diag_approx	12	1500	0.1824 s	1.3595×10^{-2}	1.0730×10^{-1}

6. Conclusion

The Riemann theta function plays a crucial role in the nonlinear Fourier analysis of signals in fields such as fiber-optic communications and coastal engineering. It is used to synthesize periodic signals through the inverse nonlinear Fourier transforms. Numerical computation of the Riemann theta function as a multi-dimensional Fourier series is challenging due to the *curse of dimensionality*. This significantly limits the practical applicability despite much interest. To better

understand the limitations, we derived some lower bounds and an upper bound on the series truncation error for certain index sets in Section 3. We investigated a tensor-train method to compute the function which utilizes the scaling and squaring approach for computing the exponential. We theoretically proved in Section 4 that such a tensor-train based approach cannot break the curse of dimensionality. Following that we proposed to exploit two other tensor-train based algorithms and another algorithm based on the hyperbolic cross index set. Using hyperelliptic solutions of the KdV and NLS equations as numerical examples, in Section 5 we showed that while the two tensor-train based algorithms work for low genus examples with real arguments, they are prone to numerical ill-conditioning. Their memory requirement is a limiting factor for high genera. While the algorithm based on the hyperbolic-cross index set can also achieve high accuracy for the low genus solutions, its novelty is the ability to compute moderately accurate solutions of high genera (up to 60) with relatively low computational cost. It therefore enables the computation of high dimensional inverse nonlinear Fourier transforms that were so far impractical. Similar algorithms based on related yet more general index sets such as the weighted Zaremba cross [47] may provide further reduction in the computation cost of high genus solutions.

CRediT authorship contribution statement

Shrinivas Chimmalgi: Methodology, Software, Validation, Formal analysis, Writing – original draft. **Sander Wahls:** Conceptualization, Formal analysis, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 716669).

Appendix A

Lemma 3. *Let $p \in \mathbb{N} \cup \{\infty\}$ and $\bar{N} \in \mathbb{N}$. Then*

$$\mathbf{n} \notin \mathcal{I}^g(p, \bar{N}) \;\Rightarrow\; |c_{\mathbf{n}}| \leq \exp\left(-\pi \frac{(\bar{N}+1)^2}{\|\Im\{\boldsymbol{\Omega}\}^{-1}\|_p}\right). \tag{A.1}$$

Proof. The matrix lower bound $\ell(\mathbf{A})$ of a matrix $\mathbf{A}$ is the smallest number m such that $m\|\mathbf{x}\|_p \leq \|\mathbf{y}\|_p$ whenever $\mathbf{y} = \mathbf{A}\mathbf{x}$ [62]. Since $\Im\{\boldsymbol{\Omega}\}$ is symmetric and positive definite,

$$\begin{aligned}
\min_{\mathbf{n}\notin\mathcal{I}^g(p,\bar{N})} \mathbf{n}\cdot\Im\{\boldsymbol{\Omega}\}\mathbf{n} &= \min_{\mathbf{n}\notin\mathcal{I}^g(p,\bar{N})} \|\mathbf{n}\cdot\Im\{\boldsymbol{\Omega}\}\mathbf{n}\|_p \\
&\geq \min_{\mathbf{n}\notin\mathcal{I}^g(p,\bar{N})} \ell(\mathbf{n}^T\Im\{\boldsymbol{\Omega}\})\|\mathbf{n}\|_p \\
&\geq \min_{\mathbf{n}\notin\mathcal{I}^g(p,\bar{N})} \ell(\mathbf{n}^T)\ell(\Im\{\boldsymbol{\Omega}\})\|\mathbf{n}\|_p \\
&\geq \min_{\mathbf{n}\notin\mathcal{I}^g(p,\bar{N})} \left\|\frac{\mathbf{n}}{\mathbf{n}\cdot\mathbf{n}}\right\|_p^{-1} \|\Im\{\boldsymbol{\Omega}\}^{-1}\|_p^{-1}\|\mathbf{n}\|_p \\
&= \min_{\mathbf{n}\notin\mathcal{I}^g(p,\bar{N})} \|\Im\{\boldsymbol{\Omega}\}^{-1}\|_p^{-1}\mathbf{n}\cdot\mathbf{n} \\
&\geq \|\Im\{\boldsymbol{\Omega}\}^{-1}\|_p^{-1}(\bar{N}+1)^2,
\end{aligned}$$

where we used [62, Lem. 4.4] in the third step, and [62, Lem. 2.2] in the fourth step (twice). Hence,

$$\begin{aligned}
|c_{\mathbf{n}}| &= |\exp(\pi\mathrm{i}\mathbf{n}\cdot\boldsymbol{\Omega}\mathbf{n})| \\
&= |\exp(\pi\mathrm{i}\mathbf{n}\cdot[\Re\{\boldsymbol{\Omega}\}+\mathrm{i}\Im\{\boldsymbol{\Omega}\}]\mathbf{n})| \\
&= \underbrace{|\exp(\pi\mathrm{i}\mathbf{n}\cdot\Re\{\boldsymbol{\Omega}\}\mathbf{n})|}_{=1}\exp(-\pi\underbrace{\mathbf{n}\cdot\Im\{\boldsymbol{\Omega}\}\mathbf{n}}_{>0}) \\
&\leq \exp(-\pi\|\Im\{\boldsymbol{\Omega}\}^{-1}\|_p^{-1}(\bar{N}+1)^2). \quad \square
\end{aligned} \tag{A.2}$$

The following results is well-known, but we present an explicit proof for the convenience of the reader. The explicit bounds given in the proof will furthermore be useful later.

Proof of Theorem 1 for $p = \infty$. Let us look at the sets $\mathcal{I}^g(\infty, N+k)\backslash\mathcal{I}^g(\infty, N+k-1)$, where $k \in \mathbb{N}$. The number of elements satisfy

$$\begin{aligned}&\#\mathcal{I}^g(\infty, N+k)\backslash\mathcal{I}^g(\infty, N+k-1)\\&= (2(N+k)+1)^g - (2(N+k-1)+1)^g\\&< (2(N+k)+1)^g,\end{aligned} \tag{A.3}$$

Using Lemma 3 with $\bar{N} = N+k-1$ shows that

$$\begin{aligned}&\mathbf{n} \in \mathcal{I}^g(\infty, N+k)\backslash\mathcal{I}^g(\infty, N+k-1)\\&\Rightarrow \mathbf{n} \notin \mathcal{I}^g(\infty, N+k-1)\\&\Rightarrow |c_{\mathbf{n}}| \overset{(A.1)}{\leq} \exp(-\pi\|\Im\{\boldsymbol{\Omega}\}^{-1}\|_p^{-1}(N+k)^2).\end{aligned} \tag{A.4}$$

Note that for any column vector $\mathbf{x}$, the matrix norm $\|\mathbf{x}^T\|_\infty$ is equal to the vector norm $\|\mathbf{x}\|_1$. We thus have

$$\begin{aligned}|\mathrm{e}^{2\pi\mathrm{i}\mathbf{n}\cdot\mathbf{z}}| = \mathrm{e}^{-2\pi\mathbf{n}^T\Im\{\mathbf{z}\}} &\leq \mathrm{e}^{2\pi\|\mathbf{n}^T\|_\infty\|\Im\{\mathbf{z}\}\|_\infty} = \mathrm{e}^{2\pi\|\mathbf{n}\|_1\|\Im\{\mathbf{z}\}\|_\infty}\\&\leq \mathrm{e}^{2\pi g(N+k)\|\Im\{\mathbf{z}\}\|_\infty},\ \forall\ \mathbf{n} \text{ as in (A.4)}.\end{aligned}$$

With these results, we find that

$$\begin{aligned}&\max_{\mathbf{z}\in Z}|\theta(\mathbf{z}\mid\boldsymbol{\Omega}) - \hat{\theta}(\mathbf{z}\mid\boldsymbol{\Omega})| \leq \sum_{\mathbf{n}\notin\mathcal{I}^g(\infty,N)}|\mathrm{e}^{2\pi\mathrm{i}\mathbf{n}\cdot\mathbf{z}}||c_{\mathbf{n}}|\\&= \sum_{k=1}^{\infty}\sum_{\mathbf{n}\in\mathcal{I}^g(\infty,N+k)\backslash\mathcal{I}^g(\infty,N+k-1)}|\mathrm{e}^{2\pi\mathrm{i}\mathbf{n}\cdot\mathbf{z}}||c_{\mathbf{n}}|\\&\leq \sum_{k=1}^{\infty}(2(N+k)+1)^g\mathrm{e}^{2\pi\|\Im\{\mathbf{z}\}\|_\infty g(N+k)-\pi\|\Im\{\boldsymbol{\Omega}\}^{-1}\|_p^{-1}(N+k)^2}\\&= \sum_{k=N+1}^{\infty}(2k+1)^g\mathrm{e}^{2\pi\|\Im\{\mathbf{z}\}\|_\infty gk-\pi\|\Im\{\boldsymbol{\Omega}\}^{-1}\|_p^{-1}k^2}\\&= \sum_{k=N+1}^{\infty}\mathrm{e}^{g\log(2k+1)+2\pi\|\Im\{\mathbf{z}\}\|_\infty gk-\pi\|\Im\{\boldsymbol{\Omega}\}^{-1}\|_p^{-1}k^2}\\&\to 0, \quad \text{for } N\to\infty. \quad \square\end{aligned}$$

Appendix B

Lemma 4. *The quadratic form tensor*

$$\underline{\mathbf{Q}}(n_1, n_2, \ldots, n_g) := \pi\mathrm{i}\mathbf{n}\cdot\boldsymbol{\Omega}\mathbf{n}, \quad \mathbf{n}\in\mathcal{I}^g$$

has a tensor-train representation with rank not larger than $\frac{g^2+g}{2}$.

Proof. Recall that the Riemann matrix $\boldsymbol{\Omega}$ is symmetric, thus we have $\pi\mathrm{i}\mathbf{n}\cdot\boldsymbol{\Omega}\mathbf{n} = \sum_{k=1}^g \pi\mathrm{i}\Omega_{kk}n_k^2 + \sum_{k=1}^g\sum_{l=k+1}^g 2\pi\mathrm{i}\Omega_{kl}n_kn_l$. The tensors $\underline{\mathbf{S}^{\mathbf{kl}}} = s_{\mathbf{n}}^{kl} := \pi\mathrm{i}\Omega_{kl}n_kn_l$, $\mathbf{n}\in\mathcal{I}^g$, can be written in the tensor-train form $s_{\mathbf{n}}^{kl} = \mathbf{S}_{n_1}^{\mathbf{kl}(1)}\mathbf{S}_{n_2}^{\mathbf{kl}(2)}\cdots\mathbf{S}_{n_g}^{\mathbf{kl}(g)}$ with

$$\mathbf{S}^{\mathbf{kl}(m)} = \begin{cases}\mathbf{1}, & m\notin\{k,l\}\\ \pi\mathrm{i}\Omega_{kl}\mathbf{u}, & m=k\\ 2\mathbf{u}, & m=l\end{cases} \in\mathbb{C}^{(2N+1)\times 1}$$

for $k\neq l$, $\mathbf{u} = [-N\ \ -N+1\ \cdots\ 0\ \cdots\ N]^T$ and

$$\mathbf{S}^{\mathbf{kk}(m)} = \begin{cases}1, & m\notin\{k,l\}\\ \pi\mathrm{i}\Omega_{kl}\mathbf{u}^2, & m=k\end{cases} \in\mathbb{C}^{(2N+1)\times 1}$$

with $\mathbf{u}^2 = \left[(-N)^2\ (-N+1)^2\ \cdots\ 0\ \cdots\ N^2\right]^T$. Thus, all $\underline{\mathbf{S}^{\mathbf{kl}}}$ are tensor-trains of rank one. It implies that

$$\underline{\mathbf{Q}} = \sum_{k=1}^{g} \underline{\mathbf{S}^{\mathbf{kk}}} + \sum_{k=1}^{g} \sum_{l=k+1}^{g} \underline{\mathbf{S}^{\mathbf{kl}}} \tag{B.1}$$

is a tensor-train with rank at most $\hat{g} = \frac{g^2+g}{2}$. □

Trivial implementation of the tensor-train $\underline{\mathbf{Q}}$ contains $((g-2)\hat{g}^2 + 2\hat{g})(2N+1)$ elements. The first and last tensor cores $\underline{\mathbf{Q}}^{(1)}$ and $\underline{\mathbf{Q}}^{(g)}$ consist of rank one matrices (vectors) while all other cores consist of rank $\hat{g}$ matrices. However, the tensor-train $\underline{\mathbf{Q}}$ is a sum of rank one tensors. Therefore each of the rank $\hat{g}$ matrix is a diagonal matrix. Hence the tensor-train $\underline{\mathbf{Q}}$ can be represented using only $\hat{g}g(2N+1)$ non-zero elements.

Proof of Lemma 2. The scaling and squaring based approximation of $c_{\mathbf{n}}$ is given by $\tilde{c}_{\mathbf{n}} = p_{\mathbf{n}}^s$, where $p_{\mathbf{n}} \coloneqq 1 + \frac{1}{1!}\hat{q}_{\mathbf{n}} + \frac{1}{2!}\hat{q}_{\mathbf{n}}^2 + \cdots + \frac{1}{(K-1)!}\hat{q}_{\mathbf{n}}^{K-1}$ and $\hat{q}_{\mathbf{n}} = q_{\mathbf{n}}/s$. Let $\hat{q}_{\mathbf{n}} = \hat{\mathbf{Q}}^{(1)}_{n_1}\hat{\mathbf{Q}}^{(2)}_{n_2}\cdots\hat{\mathbf{Q}}^{(g)}_{n_g}$. The tensor cores $\hat{\mathbf{Q}}^{(k)}_{n_k}$ can be obtained from the tensor cores $\mathbf{Q}^{(k)}_{n_k}$ as $\hat{\mathbf{Q}}^{(k)}_{n_k} = \mathbf{Q}^{(k)}_{n_k}$ for $k = 2, 3, \ldots, g$ and $\hat{\mathbf{Q}}^{(1)}_{n_1} = \mathbf{Q}^{(1)}_{n_1}/s$. The terms $p_{\mathbf{n}}$ can be written as $p_{\mathbf{n}} = \mathbf{P}^{(1)}_{n_1}\mathbf{P}^{(2)}_{n_2}\cdots\mathbf{P}^{(g)}_{n_g}$. Then by the properties of the tensor-train format [52]

$$\mathbf{P}^{(k)}_{n_k} = \begin{bmatrix} 1 & & & \\ & \left(\hat{\mathbf{Q}}^{(k)}_{n_k}\right)^{\otimes 1} & & \\ & & \ddots & \\ & & & \left(\hat{\mathbf{Q}}^{(k)}_{n_k}\right)^{\otimes (K-1)} \end{bmatrix}$$

for $k \notin \{1, g\}$, and

$$\mathbf{P}^{(1)}_{n_1} = \begin{bmatrix} 1 & \frac{1}{\sqrt{1!}}\left(\hat{\mathbf{Q}}^{(1)}_{n_1}\right)^{\otimes 1} & \cdots & \frac{1}{\sqrt{(K-1)!}}\left(\hat{\mathbf{Q}}^{(1)}_{n_1}\right)^{\otimes (K-1)} \end{bmatrix},$$

$$\mathbf{P}^{(g)}_{n_g} = \begin{bmatrix} 1 & \frac{1}{\sqrt{1!}}\left(\hat{\mathbf{Q}}^{(g)}_{n_g}\right)^{\otimes 1} & \cdots & \frac{1}{\sqrt{(K-1)!}}\left(\hat{\mathbf{Q}}^{(g)}_{n_g}\right)^{\otimes (K-1)} \end{bmatrix}^T.$$

The inner cores $\mathbf{P}^{(k)}_{n_k}$, $k \notin \{1, g\}$ thus are diagonal $R \times R$ matrices with $R = \left(\frac{\hat{g}^K - 1}{\hat{g} - 1}\right)$. Therefore $\tilde{c}_{\mathbf{n}} = p_{\mathbf{n}}^s$ has the tensor-train representation $\tilde{c}_{\mathbf{n}} = \left(\mathbf{P}^{(1)}_{n_1}\right)^{\otimes s}\left(\mathbf{P}^{(2)}_{n_2}\right)^{\otimes s}\cdots\left(\mathbf{P}^{(g)}_{n_g}\right)^{\otimes s}$. It follows that the inner cores are diagonal $\hat{R} \times \hat{R}$ matrices with $\hat{R} = R^s$. □

Remark 1. As the cores $\underline{\mathbf{P}}^{(k)}$ consist of only diagonal matrices, the number of non-zero elements in the tensor-train $\underline{\mathbf{P}}$ is only $\frac{\hat{g}^K - 1}{\hat{g} - 1}g(2N+1)$.

Proof of Proposition 5. Computing the approximation of the Riemann theta function is equivalent to the inner product of two tensors in the tensor-train format. We can work it out as the following.

$$\begin{aligned} \tilde{\theta}(\mathbf{z} \mid \boldsymbol{\Omega}) &= \sum_{\mathbf{n} \in \{-N, \ldots, N\}^g} \tilde{c}_{\mathbf{n}} \exp(2\pi \mathrm{i}\mathbf{n} \cdot \mathbf{z}) \\ &= \sum_{\mathbf{n} \in \{-N, \ldots, N\}^g} \mathrm{e}^{2\pi \mathrm{i} n_1 z_1} \left(\mathbf{P}^{(1)}_{n_1}\right)^{\otimes s} \cdots e^{2\pi \mathrm{i} n_g z_g} \left(\mathbf{P}^{(g)}_{n_g}\right)^{\otimes s} \\ &= \boldsymbol{\Gamma}_1\boldsymbol{\Gamma}_2\cdots\boldsymbol{\Gamma}_g, \quad \boldsymbol{\Gamma}_k \coloneqq \sum_{j=-N}^{N} \mathrm{e}^{2\pi \mathrm{i} j z_k} \left(\mathbf{P}^{(k)}_{j}\right)^{\otimes s}. \end{aligned}$$

Note that $\boldsymbol{\gamma}_g \coloneqq \boldsymbol{\Gamma}_g$ is a column vector. With

$$\boldsymbol{\gamma}_{k-1} \coloneqq \boldsymbol{\Gamma}_{k-1}\boldsymbol{\gamma}_k = \sum_{j=-N}^{N} \mathrm{e}^{2\pi \mathrm{i} j z_{k-1}} \left(\left(\mathbf{P}^{(k-1)}_{j}\right)^{\otimes s} \gamma_k\right),$$

we have $\tilde{\theta}(\mathbf{z} \mid \boldsymbol{\Omega}) = \boldsymbol{\gamma}_1$.

We count the number of multiplications required to compute $\boldsymbol{\gamma}_1$ as a measure of the computation cost. For computing $\tilde{\theta}(\mathbf{z} \mid \boldsymbol{\Omega})$, we start with $\boldsymbol{\gamma}_g$ which requires $\Omega((2N+1)R^s)$ multiplications for $R = \frac{\hat{g}^K - 1}{\hat{g} - 1}$. In the next stage, as the matrices $\mathbf{P}^{(g-1)}_{n_k}$ are diagonal, the matrix–vector products $\left(\mathbf{P}^{(g-1)}_{n_k}\right)^{\otimes s}\boldsymbol{\gamma}_g$ can be evaluated as the Hadamard product

$diag\left\{\left(\mathbf{P}_{n_k}^{(g-1)}\right)^{\otimes s}\right\} \odot \boldsymbol{\gamma}_g$. Here $diag\{\cdot\}$ means the vector of the diagonal elements of a matrix. Using the ideas from [63–65] the computation of the term $\left(\mathbf{P}_{n_k}^{(g-1)}\right)^{\otimes s} \boldsymbol{\gamma}_g$ given $\mathbf{P}_{n_k}^{(g-1)} \in \mathbb{C}^{R\times R}$ and $\boldsymbol{\gamma}_g \in \mathbb{C}^{R^s\times 1}$ requires at least $\Omega\left(sR^s\right)$ multiplications. Computing $\boldsymbol{\gamma}_{g-1}$ given $\boldsymbol{\gamma}_g$ thus requires $\Omega\left((2N+1)sR^s\right)$ multiplications. Continuing the same way, we can see that computing $\boldsymbol{\gamma}_1$ requires at least $\Omega\left((g-1)(2N+1)sR^s + (2N+1)R^s\right)$ multiplications. □

References

[1] Dubrovin BA. Theta functions and non-linear equations. Russian Math Surveys 1981;36(2):11–92. http://dx.doi.org/10.1070/rm1981v036n02abeh002596.
[2] Belokolos ED. Algebro-geometric approach to nonlinear integrable equations. Springer-Verlag; 1994.
[3] Wahls S, Poor HV. Fast numerical nonlinear Fourier transforms. IEEE Trans Inf Theory 2015;61(12):6957–74. http://dx.doi.org/10.1109/tit.2015.2485944.
[4] Kamalian M, Prilepsky JE, Le ST, Turitsyn SK. Periodic nonlinear Fourier transform for fiber-optic communications, part I: theory and numerical methods. Opt Express 2016;24(16):18353–69.
[5] Kamalian M, Vasylchenkova A, Prilepsky J, Shepelsky D, Turitsyn S. Communication system based on periodic nonlinear Fourier transform with exact inverse transformation. In: Proc. ECOC. 2018.
[6] Goossens J-W, Hafermann H, Jaouen Y. Data transmission based on exact inverse periodic nonlinear Fourier transform, part I: Theory. 2020;38(23):6499–519. http://dx.doi.org/10.1109/jlt.2020.3013148.
[7] Goossens J-W, Hafermann H, Jaouen Y. Data transmission based on exact inverse periodic nonlinear Fourier transform, part II: Waveform design and experiment. 2020;38(23):6520–8. http://dx.doi.org/10.1109/jlt.2020.3013163.
[8] Osborne AR. Nonlinear ocean waves and the inverse scattering transform. Elsevier, AP; 2010.
[9] Randoux S, Suret P, El G. Inverse scattering transform analysis of rogue waves using local periodization procedure. Sci Rep 2016;6:29238.
[10] Brühl M, Becker M. Analysis of subaerial landslide data using nonlinear Fourier transform based on korteweg-de vries equation (KdV-NLFT). J Earthq Tsunami 2018;12(2):1840002.
[11] Jeans G, Xiao W, Osborne A, Jackson C, Mitchell D. The application of nonlinear Fourier analysis to soliton quantification for offshore engineering. In: Proc. ASME OMAE. 2017.
[12] Osborne A, Ponce de León S. Properties of rogue waves and the shape of the ocean wave power spectrum. In: Proc. ASME OMAE. 2017.
[13] Randoux S, Suret P, Chabchoub A, Kibler B, El G. Nonlinear spectral analysis of peregrine solitons observed in optics and in hydrodynamic experiments. Phys Rev E 2018;98:022219.
[14] Osborne A, Resio D, Ponce de León S, Chirivì E. Highly nonlinear wind waves in currituck sound: dense breather turbulence in random ocean waves. Ocean Dyn 2019;69:187–219.
[15] Baragiola B, Pantaleoni G, Alexander R, Karanjai A, Menicucci N. All-Gaussian universality and fault tolerance with the gottesman-kitaev-preskill code. Phys Rev Lett 2019;123.
[16] Bobenko AI. All constant mean curvature tori in R^3, S^3, H^3 in terms of theta-functions. Math Ann 1991;290(1):209–45. http://dx.doi.org/10.1007/bf01459243.
[17] Deconinck B, van Hoeij M. Computing Riemann matrices of algebraic curves. Physica D 2001;152–153:28–46. http://dx.doi.org/10.1016/s0167-2789(01)00156-7.
[18] Birkenhake C. Complex abelian varieties. Springer; 2004.
[19] Eichler M, Zagier D. The theory of Jacobi forms. Birkhäuser; 1985, http://dx.doi.org/10.1007/978-1-4684-9162-3.
[20] Regev O, Stephens-Davidowitz N. An inequality for Gaussians on lattices. SIAM J Discrete Math 2017;31(2):749–57. http://dx.doi.org/10.1137/15m1052226.
[21] Krefl D, Carrazza S, Haghighat B, Kahlen J. Riemann-theta Boltzmann machine. Neurocomputing 2020;388:334–45. http://dx.doi.org/10.1016/j.neucom.2020.01.011.
[22] Nielsen F. On the Kullback-Leibler divergence between discrete normal distributions. 2021, http://dx.doi.org/10.1007/s41745-021-00279-5, arXiv:2109.14920,
[23] Gaudry P. Fast genus 2 arithmetic based on theta functions. J Math Cryptol 2007;1(3). http://dx.doi.org/10.1515/jmc.2007.012.
[24] Agostini D, Améndola C. Discrete Gaussian distributions via theta functions. SIAM J Appl Algebra Geom 2019;3(1). http://dx.doi.org/10.1137/18m1164937.
[25] Deconinck B, Heil M, Bobenko A, van Hoeij M, Schmies M. Computing Riemann theta functions. Math Comp 2004;73. http://dx.doi.org/10.1090/s0025-5718-03-01609-0.
[26] Labrande H, Thomé E. Computing theta functions in quasi-linear time in genus two and above. LMS J Comput Math 2016;19.
[27] Frauendiener J, Jaber C, Klein C. Efficient computation of multidimensional theta functions. J Geom Phys 2019;141. http://dx.doi.org/10.1016/j.geomphys.2019.03.011.
[28] Agostini D, Chua L. Computing theta functions with julia. J Soft Algebra Geom 2021;11:41–51. http://dx.doi.org/10.2140/jsag.2021.11.41.
[29] Chen L. Curse of dimensionality. In: Encyclopedia of database systems. Springer US; 2009, p. 545–6. http://dx.doi.org/10.1007/978-0-387-39940-9_133.
[30] Sidiropoulos ND, De Lathauwer L, Fu X, Huang K, Papalexakis EE, Faloutsos C. Tensor decomposition for signal processing and machine learning. IEEE Trans Signal Process 2017;65. http://dx.doi.org/10.1109/TSP.2017.2690524.
[31] Ghadermarzy N, Plan Y, Yilmaz O. Learning tensors from partial binary measurements. IEEE Trans Signal Process 2019;67. http://dx.doi.org/10.1109/tsp.2018.2879031.
[32] Kanatsoulis CI, Fu X, Sidiropoulos ND, Akcakaya M. Tensor completion from regular sub-nyquist samples. IEEE Trans Signal Process 2020;68. http://dx.doi.org/10.1109/tsp.2019.2952044.
[33] Chen H, Vorobyov SA, So HC, Ahmad F, Porikli F. Introduction to the special issue on tensor decomposition for signal processing and machine learning. IEEE J STSP 2021;15(3):433–7. http://dx.doi.org/10.1109/jstsp.2021.3065184.
[34] Chen H, Ahmad F, Vorobyov S, Porikli F. Tensor decompositions in wireless communications and MIMO radar. IEEE J STSP 2021;15. http://dx.doi.org/10.1109/jstsp.2021.3061937.
[35] Muti D, Bourennane S. Multidimensional signal processing using lower-rank tensor approximation. In: 2003 IEEE ICASSP. IEEE; 2003, http://dx.doi.org/10.1109/icassp.2003.1199510.
[36] de Goulart JHM, Comon P. A novel non-iterative algorithm for low-multilinear-rank tensor approximation. In: Proc. EUSIPCO. 2017, http://dx.doi.org/10.23919/eusipco.2017.8081288.
[37] Yuan L, Zhao Q, Cao J. High-order tensor completion for data recovery via sparse tensor-train optimization. In: Proc. IEEE ICASSP. 2018, http://dx.doi.org/10.1109/icassp.2018.8462592.
[38] Gelß P. The tensor-train format and its applications [Ph.D. thesis], Freie Universität Berlin; 2017, http://dx.doi.org/10.17169/REFUBIUM-7566.

[39] Wahls S, Koivunen V, Poor HV, Verhaegen M. Learning multidimensional Fourier series with tensor trains. In: Proc. IEEE GlobalSIP. 2014.
[40] Kargas N, Sidiropoulos N. Supervised learning and canonical decomposition of multivariate functions. IEEE Trans. Signal Process 2021;69.
[41] Hallatschek K. Fouriertransformation auf dünnen gittern mit hierarchischen basen. Numer Mat 1992;63(1):83–97.
[42] Döhler M, Kunis S, Potts D. Nonequispaced hyperbolic cross fast Fourier transform. SIAM J Numer Anal 2010;47. http://dx.doi.org/10.1137/090754947.
[43] Kämmerer L. Reconstructing hyperbolic cross trigonometric polynomials by sampling along rank-1 lattices. SIAM J Numer Anal 2013;51. http://dx.doi.org/10.1137/120871183.
[44] Kämmerer L, Potts D, Volkmer T. Approximation of multivariate periodic functions by trigonometric polynomials based on sampling along rank-1 lattice with generating vector of korobov form. J Complexity 2015;31. http://dx.doi.org/10.1016/j.jco.2014.09.001.
[45] Plonka G, Potts D, Steidl G, Tasche M. Numerical fourier analysis. Springer; 2018, http://dx.doi.org/10.1007/978-3-030-04306-3.
[46] Kämmerer L, Kunis S, Potts D. Interpolation lattices for hyperbolic cross trigonometric polynomials. J Complexity 2012;28. http://dx.doi.org/10.1016/j.jco.2011.05.002.
[47] Cools R, Kuo FY, Nuyens D. Constructing lattice rules based on weighted degree of exactness and worst case error. Computing 2010;87. http://dx.doi.org/10.1007/s00607-009-0076-1.
[48] Dũng D, Temlyakov V, Ullrich T. In: Tikhonov S, editor. Hyperbolic Cross Approximation. Springer; 2018, http://dx.doi.org/10.1007/978-3-319-92240-9.
[49] Mumford D. Tata lectures on theta I. Springer; 2007.
[50] Lenstra AK, Lenstra Jr HW, Lovász L. Factoring polynomials with rational coefficients. Math Ann 1982;261.
[51] Frauendiener J, Klein C. Hyperelliptic theta-functions and spectral methods: KdV and KP solutions. Let Math Phys 2006;76. http://dx.doi.org/10.1007/s11005-006-0068-4.
[52] Oseledets I. Tensor-train decomposition. SIAM J Comput 2011;33. http://dx.doi.org/10.1137/090752286.
[53] Lee N, Cichocki A. Fundamental tensor operations for large-scale data analysis using tensor network formats. Multidimens Syst Signal Process 2017;29(3):921–60. http://dx.doi.org/10.1007/s11045-017-0481-0.
[54] Savostyanov D, Oseledets I. Fast adaptive interpolation of multi-dimensional arrays in tensor train format. In: Proc. int. wksp. multidim. (ND) syst.. 2011, http://dx.doi.org/10.1109/nds.2011.6076873.
[55] Rump SM, Ogita T, Oishi S. Accurate floating-point summation part I: Faithful rounding. SIAM J Sci Comput 2008;31. http://dx.doi.org/10.1137/050645671.
[56] Higham NJ. The scaling and squaring method for the matrix exponential revisited. SIAM J Matrix Anal Appl 2005;26. http://dx.doi.org/10.1137/04061101x.
[57] Greene SM, Batista VS. Tensor-train split-operator Fourier transform (TT-SOFT) method: Multidimensional nonadiabatic quantum dynamics. J Chem Theory Comput 2017;13. http://dx.doi.org/10.1021/acs.jctc.7b00608.
[58] Dolgov S, Khoromskij B. Two-level QTT-tucker format for optimized tensor calculus. SIAM J Matrix Anal Appl 2013;34. http://dx.doi.org/10.1137/120882597.
[59] Oseledets I. TT-toolbox Version 2.2.2. 2021, Online; https://github.com/oseledets/TT-Toolbox. [Accessed 3 March 2021].
[60] A balancing act for the matrix exponential. 2021, https://blogs.mathworks.com/cleve/2012/07/23/a-balancing-act-for-the-matrix-exponential/. [Accessed: 6 August 2021].
[61] Phillips GM, Taylor PJ. Taylor's polynomial and series. In: Theory and Applications of Numerical Analysis. Elsevier; 1996, p. 39–51. http://dx.doi.org/10.1016/b978-012553560-1/50004-5.
[62] Grcar JF. A matrix lower bound. Linear Algebra Appl 2010;433. http://dx.doi.org/10.1016/j.laa.2010.02.014.
[63] Schweiger R, Erlich Y, Carmi S. FactorialHMM: fast and exact inference in factorial hidden Markov models. In: Schwartz R, editor. Bioinformatics 2018;35. http://dx.doi.org/10.1093/bioinformatics/bty944.
[64] Williams A. Efficient computation of a kronecker - vector product (with multiple matrices). 2021, URL https://gist.github.com/ahwillia/f65bc70cb30206d4eadec857b98c4065. [Online; Accessed 29 April 2021].
[65] Vector multiplication with multiple kronecker products. 2021, URL https://math.stackexchange.com/questions/1879933/vector-multiplication-with-multiple-kronecker-products%7D. [Online; Accessed 29 April 2021].