Adv Comput Math (2014) 40:1159–1168
DOI 10.1007/s10444-014-9345-4

An algorithm for fast Hilbert transform of real functions

Roberto Bilato · Omar Maj · Marco Brambilla

Received: 13 November 2012 / Accepted: 12 March 2014 /
Published online: 11 April 2014
© Springer Science+Business Media New York 2014

Abstract A simple and accurate algorithm to evaluate the Hilbert transform of a real function is proposed using interpolations with piecewise–linear functions. An appropriate matrix representation reduces the complexity of this algorithm to the complexity of matrix-vector multiplication. Since the *core* matrix is an antisymmetric Toeplitz matrix, the discrete trigonometric transform can be exploited to calculate the matrix–vector multiplication with a reduction of the complexity to $O(N \log N)$, with N being the dimension of the core matrix. This algorithm has been originally envisaged for self-consistent simulations of radio-frequency wave propagation and absorption in fusion plasmas.

Keywords Hilbert transform · B–spline · Fast Fourier transform · Discrete trigonometric transform

Mathematics Subject Classifications (2010) 15B05 · 65T50 · 44A15 · 65Y20

1 Introduction

The Hilbert transform on the real line,

$$\mathcal{H}_{\mathbb{R}} f(x) = \frac{1}{\pi}\text{p.v.}\int_{-\infty}^{+\infty} \frac{f(y)}{x-y}dy = \frac{1}{\pi}\lim_{\epsilon\to 0^+}\int_{|y-x|>\epsilon} \frac{f(y)}{x-y}dy, \tag{1}$$

enters the coefficients of the wave equation describing electromagnetic waves in fusion plasmas [1]. Specifically, in self-consistent simulations of plasma waves,

Communicated by: Zydrunas Gimbutas

R. Bilato (✉) · O. Maj · M. Brambilla
Max Planck Institute for Plasma Physics, EURATOM Association, Boltzmannstr. 2, 85748 Garching, Germany
e-mail: roberto.bilato@ipp.mpg.de

a loop is set up between a solver for the electromagnetic fields, which perturb the plasma, and a solver for the velocity distribution of plasma particles, whose Hilbert transform, in turn, determines the coefficients of the wave equation for the electromagnetic fields.

In the development of the interface between the wave solver `TORIC` and the Fokker–Planck solver `SSFPQL` [2], we have devised an algorithm for the Hilbert transform, (we believe) novel at the time of [3]. Because of the ubiquitousness of the Hilbert transform, particularly for the analysis of time-series [4–7] (and references therein), we have later realized that this algorithm, properly reformulated, can be of general interest.

The standard method of calculating the Hilbert transform implemented in many commercial and open-source software is based on its relation to the Fourier transform [8]. This method was originally proposed by Henrici, and thus hereafter it will be referred to simply as the Henrici's algorithm. It inherits the complexity of the Fast Fourier Transform (FFT), i.e., $O(N \log N)$, with N being the length of the signal. However, as observed e.g. in [9], Henrici's algorithm is affected by an error which becomes larger when the argument approaches the extrema of integration (see for example Fig. 1a below).

A new method based on the Haar multiresolution approximation, whose precision is superior, has been proposed by Zhou et al. [9]. A disadvantage of Zhou's approach, however, is the higher complexity, namely $O(N^2)$, which plays against it when processing long signals. More recently, Micchelli and co-workers have proposed a general approach based on the B–spline series approximation, which combines the accuracy of Zhou's method with the low complexity of the FFT [10].

In Section 2, we present the algorithm we have developed few years ago [3], originally inspired by Valeo [11]. The result is particularly simple and easily implemented. In addition, the use of discrete trigonometric transforms (DTT) reduces the complexity of the algorithm from $O(N^2)$ to $O(N \log N)$. Theoretical error estimates for the method are given in Section 3. In Section 4, we discuss the numerical implementation based on the `FFTW` package [12], and presents numerical experiments with

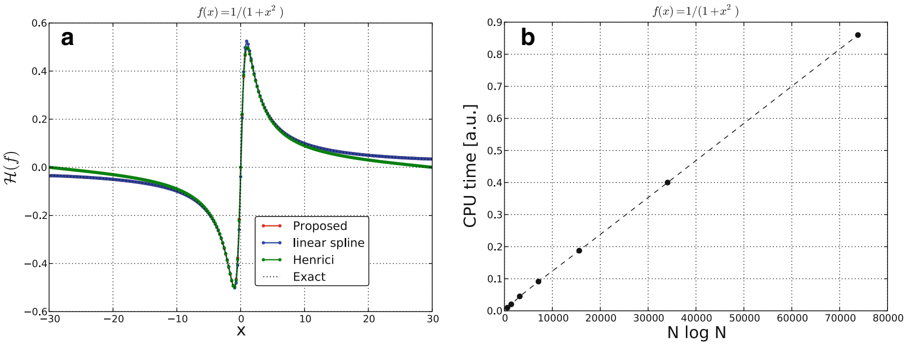

Fig. 1 **a** Hilbert transform of the function $f(x) = 1/(1 + x^2)$ calculated by using Eq. 5; the proposed algorithm, linear splines, and the exact transform are overlapped within the accuracy of the plot. **b** CPU time as function of the number of points N

test cases proposed in [9]. Numerical results for these test functions are compared to those obtained by an ad-hoc implementation of the Henrici's algorithm as well as of the linear-spline version of the the method of Micchelli et al. [10].

2 Algorithm

In this section we describe the algorithm envisaged in [3] for the Hilbert transform of a function $f(x)$. Here, we proceed formally, mathematically precise results being given in Section 3.

For $h > 0$, let $\tau = \{x_n = x_0 + hn\}_{n=0}^{N}$ be a grid of equally spaced $N+1$ points, determining the closed interval $I_\tau = [x_0, x_N]$. The grid τ should be such that the function $f(x)$ can be approximated by zero outside I_τ. Then, for every interior point $x_k \in \tau$ with $k = 1, \dots, N-1$, we have

$$\begin{aligned}\big(\mathcal{H}_{\mathbb{R}} f\big)(x_k) &\approx \frac{1}{\pi} \lim_{\epsilon \to 0^+} \sum_{n=0}^{N-1} \int_{\substack{|y-x_k|>\epsilon \\ x_n \le y \le x_{n+1}}} \frac{f(y)}{x_k - y} dy \\ &= \frac{1}{\pi} \lim_{\epsilon \to 0^+} \left[\int_{x_{k-1}}^{x_k-\epsilon} + \int_{x_k+\epsilon}^{x_{k+1}} \right] \frac{f(y)}{x_k - y} dy \\ &\qquad + \frac{1}{\pi} \left[\sum_{n=0}^{k-2} + \sum_{n=k+1}^{N-1} \right] \int_{x_n}^{x_{n+1}} \frac{f(y)}{x_k - y} dy.\end{aligned} \tag{2}$$

If the nodal values $f(x_k)$ of the considered function are known, on each interval $[x_n, x_{n+1}]$ the function $f(y)$ is approximated by linear interpolation,

$$f(y) \approx f(x_n) + \frac{f(x_{n+1}) - f(x_n)}{h} (y - x_n). \tag{3}$$

Each contribution can now be evaluated analytically, and the approximated Hilbert transform (2) becomes

$$\begin{aligned}\big(\mathcal{H}_{\mathbb{R}} f\big)(x_k) \approx -\frac{1}{\pi} \Bigg\{ & f(x_{k+1}) - f(x_{k-1}) \\ &+ \sum_{n=1}^{N-1-k} \Big[-\big(1 - (n+1) b_n\big) f(x_{k+n}) + \big(1 - n b_n\big) f(x_{k+n+1}) \Big] \\ &+ \sum_{n=1}^{k-1} \Big[\big(1 - (n+1) b_n\big) f(x_{k-n}) - \big(1 - n b_n\big) f(x_{k-n-1}) \Big] \Bigg\},\end{aligned} \tag{4}$$

where $b_n = \log\big((n+1)/n\big)$. The right-hand side of Eq. 4 defines a linear operator, which maps the vector of sample values $F = \big(f(x_0), f(x_i), \dots, f(x_N)\big)$ into the approximate values of the Hilbert transform on interior grid points. Such an operator is the proposed discrete Hilbert transform H_τ, with the subscript denoting explicitly the dependence on the grid τ. In matrix form,

$$H_\tau F = \mathbf{A} F_{\text{int}} + \mathbf{C} F_{\text{bnd}}, \tag{5}$$

where the vector F has been split into its projection on internal nodes $F_{\text{int}} = \big(f(x_1), \ldots, f(x_{N-1})\big)$ and boundary points $F_{\text{bnd}} = \big(f(x_0), f(x_N)\big)$, whereas $\mathbf{A}$ is a $(N-1) \times (N-1)$ antisymmetric Toeplitz matrix,

$$\mathbf{A} = \begin{pmatrix} a_0 & a_1 & \cdots & a_{N-3} & a_{N-2} \\ -a_1 & a_0 & \cdots & a_{N-4} & a_{N-3} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -a_{N-2} & -a_{N-3} & \cdots & -a_1 & a_0 \end{pmatrix}, \tag{6}$$

with

$$a_k = -\frac{1}{\pi} \cdot \begin{cases} 0, & \text{for } k = 0, \\ 2b_1, & \text{for } k = 1, \\ (k+1)b_k - (k-1)b_{k-1}, & \text{for } k > 1, \end{cases} \tag{7}$$

$$b_k = \log\big((k+1)/k\big), \tag{8}$$

and $\mathbf{C}$ is a rectangular $(N-1) \times 2$ matrix

$$\mathbf{C} = \begin{pmatrix} 0 & c_{N-1} \\ -c_1 & c_{N-2} \\ \vdots & \vdots \\ -c_{N-1} & 0 \end{pmatrix}, \qquad c_k = -\frac{1}{\pi}(1 - kb_k). \tag{9}$$

An antisymmetric Toeplitz matrix is completely defined by its first row. Thus the evaluation of the $N-1$ elements $(a_0, \ldots, a_{N-2})$ is enough to complete determine $\mathbf{A}$, and that requires only $N-2$ calculations of the logarithm.

In general the complexity of matrix-vector product involving matrices of rank N is $O(N^2)$ [13]. This could be the bottleneck of Eq. 5. However, for some special matrices, such as the Toeplitz ones, it is possible to use Discrete Trigonometric Transforms (DTT) to perform the matrix-vector product, whose complexity is thereby reduced to that of the fast DFT, namely, $O(N \log N)$. A list of possible representations are given by Potts and Steidl in [14]. In particular, we consider

$$\mathbf{A} = \mathbf{C}^{IV}_{N-1}\mathbf{D}\mathbf{S}^{IV}_{N-1} - \mathbf{S}^{IV}_{N-1}\mathbf{D}\mathbf{C}^{IV}_{N-1}, \tag{10}$$

where $\mathbf{D}$ is a diagonal matrix

$$\mathbf{D}_{N-1} = \text{diag}(d_0, \ldots, d_{N-2}),$$
$$(d_0, \ldots, d_{N-2})^T = \tilde{\mathbf{S}}^{III}_{N-1}(a_1, \ldots, a_{N-2}, 0)^T,$$

where the superscript T denotes the transpose. For every integer $n > 0$, the $n \times n$ matrices of discrete trigonometric transforms are

$$\begin{aligned} \big(\mathbf{C}^{IV}_n\big)_{jk} &= \sqrt{\frac{2}{n}} \cos\left(\pi \frac{(2j+1)(2k+1)}{4n}\right), \\ \big(\mathbf{S}^{IV}_n\big)_{jk} &= \sqrt{\frac{2}{n}} \sin\left(\pi \frac{(2j+1)(2k+1)}{4n}\right), \\ \big(\tilde{\mathbf{S}}^{III}_n\big)_{jk} &= (\epsilon^n_{k+1})^2 \; \sin\left(\pi \frac{(2j+1)(k+1)}{2n}\right), \end{aligned} \tag{11}$$

with $j, k \in \{0, \dots, n-1\}$, and

$$\epsilon_k^n = \begin{cases} 1/\sqrt{2}, & \text{for } k = 0, n, \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

The matrix-vector products involving matrices $\mathbf{C}_{N-1}^{IV}$, $\mathbf{S}_{N-1}^{IV}$, and $\mathbf{S}_{N-1}^{III}$ have the same complexity as the FFT, namely $O(N \log N)$. As a consequence, the complexity of all matrix-vector products are not worse than $O(N \log N)$.

3 Error estimates

The numerical Hilbert transform (5) has been obtained via piecewise linear interpolation of the sampled function, and, in this sense, it is a special case of the method proposed by Micchelli et al. [10], in which the approximation of the Hilbert transform is sought in a finite-dimensional subspace of $L^2(\mathbb{R})$ spanned by the Hilbert transform of B–splines.

In our case, however, the Hilbert transform is collocated on the same grid τ where the considered function is sampled, allowing us to remove explicitly the logarithmic singularities in the Hilbert transform of the interpolant. In the case of Micchelli et al. [10] the numerical Hilbert transform can also be collocated on a grid, which, however, cannot be the sampling grid τ, as the Hilbert transform of the B–splines is singular at the sampling nodes.

For such collocated numerical transform (5), the theoretical error is naturally estimated at the discrete level directly, as opposite to error estimates of interpolants in L^2. This will require the control of the error in a stronger norm than the usual L^2 error estimates.

First, we consider the error due to the restriction of f to the interval I_τ.

Lemma 1 *If $I = [a, b] \subset \mathbb{R}$ is any closed bounded interval and $f \in L^2(\mathbb{R})$,*

$$\left| \mathcal{H}_{\mathbb{R}} f(x) - \frac{1}{\pi} \text{p.v.} \int_I \frac{f(y)}{x-y} dy \right| \leq \frac{1}{\pi} \left(\frac{2}{\text{dist}(x, I^c)} \right)^{1/2} \|f\|_{L^2(I^c)}, \tag{13}$$

almost everywhere in (a, b). Here, $\text{dist}(x, I^c) = \min\{x-a, b-x\}$ is the distance of the point $x \in I$ from the complement $I^c = \mathbb{R} \setminus I$.

Proof The result follows from Schwartz inequality in $L^2(I^c)$, namely,

$$\left| \mathcal{H}_{\mathbb{R}} f(x) - \frac{1}{\pi} \text{p.v.} \int_I \frac{f(y)}{x-y} dy \right| \leq \frac{1}{\pi} \left(\int_{I^c} (x-y)^{-2} dy \right)^{\frac{1}{2}} \left(\int_{I^c} |f(y)|^2 dy \right)^{\frac{1}{2}},$$

almost everywhere in I, and the first factor on the right-hand side is bounded by $\sqrt{2/\text{dist}(x, I^c)}$. □

Let us notice that the foregoing estimate does not control the truncation error near boundary points. Nonetheless, with $I = I_\tau$ and $J = [x_1, x_{N-1}] \subset I_\tau$, Lemma 1 amounts to a uniform bound of the truncation error for $x \in J$, but with constant scaling like $1/\sqrt{h}$.

Next we consider the discretization error. Since standard results about linear interpolation are used, we shall consider those functions $f \in L^2(\mathbb{R})$ having a C^2 restriction $f|_{I_\tau}$ to the closed interval I_τ.

Let χ_τ be the characteristic function of the interval I_τ, i.e., $\chi_\tau(x) = 1$ for $x \in I_\tau$ and $\chi_\tau(x) = 0$ elsewhere, and let f_τ be the linear interpolant

$$f_\tau(x) = f(x_k) + (x - x_k) f[x_k, x_{k+1}],$$

for $x \in [x_k, x_{k+1})$, $k \in \{0, \ldots, N-1\}$, and $f_\tau(x) = 0$ when $x \notin I_\tau$. Here, $f[x_k, \ldots, x_{k+j}]$ denotes the standard divided differences. We compare the Hilbert transform of $\chi_\tau f$ and f_τ. In L^2, $\mathcal{H}_\mathbb{R}$ is an isometry, hence we just have $\|\mathcal{H}_\mathbb{R}(\chi_\tau f - f_\tau)\|_{L^2(J)} \leq \sqrt{|J|} \cdot \|\chi_\tau f - f_\tau\|_{L^\infty}$ and, by interpolation theory, $\|\mathcal{H}_\mathbb{R}(\chi_\tau f - f_\tau)\|_{L^2(J)} = O(h^2)$, as shown by Micchelli et al. [10]. For the control of the point-wise error, however, we find $O(h)$.

Lemma 2 *Let $\tau = \{x_0, \ldots, x_N\}$ and I_τ be as in Section 2, with step size $h > 0$, $f \in L^2(\mathbb{R})$ with $f|_{I_\tau} \in C^2(I_\tau)$, and χ_τ and f_τ given above. Then, $\mathcal{H}_\mathbb{R}(\chi_\tau f - f_\tau)$ is continuous and*

$$\sup_{x\in\mathbb{R}} \left|\mathcal{H}_\mathbb{R}(\chi_\tau f - f_\tau)(x)\right| \leq Ch, \tag{14}$$

for a constant $C > 0$ independent of the step size h.

Proof Let us start observing that $g = \chi_\tau f - f_\tau$ is continuous, vanishes on nodes $x_k \in \tau$ and it has a piecewise continuous derivative. Hence, $g \in H^1(\mathbb{R})$ and $\mathcal{H}_\mathbb{R} g \in H^1(\mathbb{R})$ in view of the property that $(\mathcal{H}_\mathbb{R} g)' = \mathcal{H}_\mathbb{R} g'$. Here, $H^s = H^s(\mathbb{R})$, $s \in \mathbb{R}$, are the standard Sobolev spaces. It follows that $\mathcal{H}_\mathbb{R} g$ is continuous (actually, it is Hölder-continuous of index $\alpha = 1/2$, [15, Chapter 4, Proposition 1.5]). In addition,

$$\mathcal{H}_\mathbb{R} g(x) = \frac{1}{2\pi} \int e^{ix\xi} [-i \operatorname{sgn}(\xi)] \widehat{g}(\xi) dx,$$

and

$$\begin{aligned} \left|\mathcal{H}_\mathbb{R} g(x)\right| &= \frac{1}{2\pi} \left| \int e^{ix\xi} \operatorname{sgn}(\xi) \widehat{g}(\xi) dx \right| \\ &\leq \frac{1}{2\pi} \Big(\int \frac{d\xi}{1+\xi^2} \Big)^{\frac{1}{2}} \Big(\int (1+\xi^2) |\widehat{g}(\xi)|^2 d\xi \Big)^{\frac{1}{2}} = C' \|g\|_{H^1}, \end{aligned}$$

where $\widehat{g}$ is the Fourier transform of g. The H^1-error of the piecewise linear interpolation of $f \in C^2$ is $\|g\|_{H^1} = O(h)$, hence, $|\mathcal{H}_\mathbb{R} g| \leq Ch$, as claimed. □

The combination of Lemma 1 and Lemma 2 together with the observation that $(H_\tau F)_i = \mathcal{H}_\mathbb{R} f_\tau(x_i)$ for $i = 1, \ldots, N-1$ gives a general error estimate for the numerical Hilbert transform $H_\tau F$. Specifically, since $f|_{I_\tau}$ is of class C^2, $\mathcal{H}_\mathbb{R} f$ is continuous on the interior of I_τ and for every interior node x_i, $i = 1, \ldots, N-1$, one has

$$\left|\mathcal{H}_\mathbb{R} f(x_i) - (H_\tau F)_i\right| \leq \left|\mathcal{H}_\mathbb{R}(\chi_\tau f - f_\tau)(x_i)\right| + \left|\mathcal{H}_\mathbb{R} f(x_i) - \mathcal{H}_\mathbb{R}(\chi_\tau f)(x_i)\right|.$$

On the right-hand side, we find the sum of discretization and truncation errors.

Theorem 1 *Let $\tau = \{x_0, \ldots, x_N\}$ and I_τ be as in Section 2, with step size $h > 0$, and let $f \in L^2(\mathbb{R})$ with $f|_{I_\tau} \in C^2(I_\tau)$. Then*

$$\max_{x_i \in \tau} \left| \mathcal{H}_{\mathbb{R}} f(x_i) - (H_\tau F)_i \right| \leq Ch + \frac{1}{\pi}\sqrt{\frac{2}{h}} \|f\|_{L^2(I_\tau^c)}, \tag{15}$$

for a constant $C > 0$ independent of the step size h.

In Section 4, a number of numerical experiments are presented for the same test cases considered by Zhou et al. [9] and Micchelli et al. [10]. For such cases we find that the error estimate (15) is not sharp.

4 Numerical implementation and experiments

The main steps of the algorithm we have implemented are summarized in Table 1 with the corresponding complexity. In estimating the complexity, only the multiplications are counted, whereas additions and logarithms necessary to build $(a_0, a_1, \ldots, a_{N-2})$ are neglected for they are an overhead of complexity $O(N)$. If the Hilbert transform is used to analyze equal-length sub-intervals of a long data series, the evaluation of the matrix **D** needs to be done only once in the initialization phase.

The best performances of the FFT are obtained when the dimensionality of the problem is an integer power of 2 [13]. Thus, in our examples $(N - 1)$ is chosen to be integer of power 2, and at each increase of N the sub-intervals are halved. For the following analysis and testing, we consider the set of functions proposed in [9, 10], and reported in Table 2. For the FFT and DTT we use the `FFTW3` package [12].

Figure 1a shows the Hilbert transform of the first function calculated by the present method, by linear splines as proposed in Micchelli et al. [10], and by Henrici's algorithm, together with the exact expression. The proposed method, the linear spline method and the exact solution are overlapped within the accuracy of the plot. The computational time is plotted in Fig. 1b as function of $N \log N$, and it scales linearly

Table 1 Main steps of the algorithm implemented to evaluate Eq. 5 with **A** decomposed according to Eq. 10

Operation	Complexity
Diagonal matrix **D**	$(N-1)\ \log(N-1)$
$V_s = \mathbf{S}^{\mathbf{IV}}_{\mathbf{N-1}} F_{\text{int}}$ and $V_c = \mathbf{C}^{\mathbf{IV}}_{\mathbf{N-1}} F_{\text{int}}$	$2 \times \left[(N-1)\ \log(N-1)\right]$
$W_s = \mathbf{D} \cdot V_s$ and $W_c = \mathbf{D} \cdot V_c$	$2 \times (N-1)$
$\mathbf{C}^{\mathbf{IV}}_{\mathbf{N-1}} \cdot V_s$ and $\mathbf{S}^{\mathbf{IV}}_{\mathbf{N-1}} \cdot V_c$	$2 \times \left[(N-1)\ \log(N-1)\right]$
$\mathbf{C} F_{\text{bnd}}$	$2 \log(N-1)$

The complexity is calculated with respect to the number of multiplications

Table 2 Functions with their Hilbert transform considered in this work. For the error function we have used the routine `WOFZ` of Poppe and Wijers [16]

	Function $f(x)$	Hilbert transform $(\mathcal{H}_{\mathbb{R}} f)(x)$
1.	$\frac{1}{1+x^2}$	$\frac{x}{1+x^2}$
2.	$\frac{1}{1+x^4}$	$\frac{x(1+x^2)}{\sqrt{2}(1+x^4)}$
3.	$\frac{\sin x}{1+x^2}$	$\frac{1/e-\cos x}{1+x^2}$
4.	$\frac{\sin x}{1+x^4}$	$\frac{1}{1+x^4}\left[e^{-1/\sqrt{2}}\cos\frac{1}{\sqrt{2}}+e^{-1/\sqrt{2}}\sin\frac{1}{\sqrt{2}}x^2-\cos x\right]$
5.	e^{-x^2}	$-\frac{2}{\sqrt{\pi}}e^{-x^2}\int_0^x e^{t^2}\,dt=-e^{-x^2}\mathrm{Im}\{\mathrm{erfc}(-i\,x)\}$

with $N\log N$ as expected from Table 1. In this respect, the three algorithms have the same complexity scaling. According to Theorem 1, a measure of the error is the maximum norm (15), namely,

$$E_\tau = \max_{x_i\in\tau}\left|\mathcal{H}_{\mathbb{R}} f(x_i) - (H_\tau F)_i\right|. \tag{16}$$

Figure 2a shows E_τ of the first function of Table 2 versus the step size h and for various lengths of the interval I_τ: when h decreases below a threshold the truncation error of Lemma 1 takes over, and E_τ slightly increases when decreasing further h. This threshold decreases with the inverse of the length of the interval I_τ, for the residual $\|f\|_{L^2(\mathbb{R}\setminus I_\tau)}$ decreases when increasing the extension of I_τ. Figure 2b shows E_τ for the same I_τ intervals in the case of the fifth function of Table 2: since this function decreases rapidly, the truncation error is negligible for the range of parameters here considered.

As already observable in Fig. 2, when the truncation error (13) is not dominant, the convergence rate of the error is $O(h^2)$ for the present algorithm. For sake of comparison, Fig. 3 shows the errors of the the present algorithm together with those of

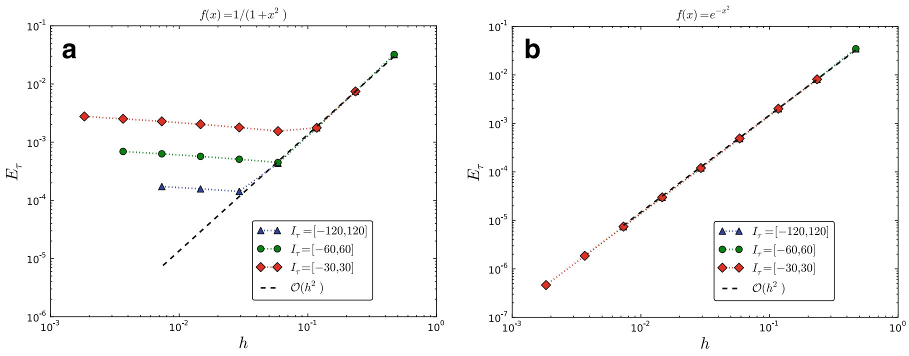

Fig. 2 Error (16) for the first (**a**) and the last (**b**) function of Table 2 as versus the step size h, and for three lengths of the interval. The *dashed line* is the 2nd-order reference scaling

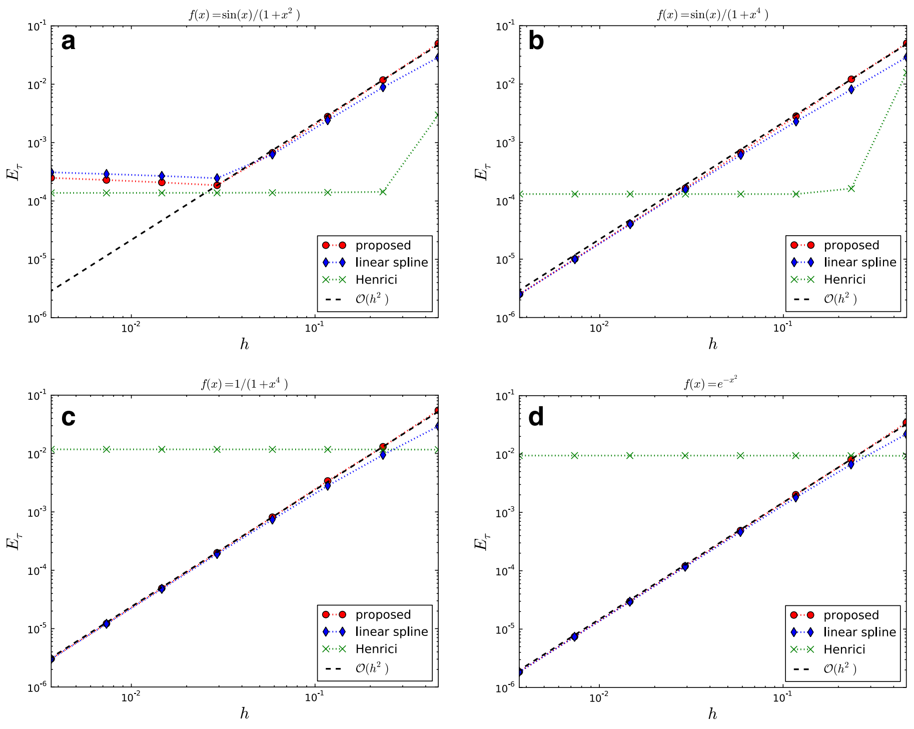

Fig. 3 Error (16) for the functions of Table 2 with $I_\tau = [-60, 60]$

Henrici and Micchelli for the functions of Table 2. The convergence rate $O(h^2)$ of the present algorithm is confirmed also for the other functions of Table 2, and it is equal to the convergence rate of the linear-spline version of the the method of Micchelli et al. [10]. In the case of Henrici's method, the convergence rate is in general slow, as already discussed in [9], because of the large errors at the points close to the extrema.

5 Conclusions

We have presented a new algorithm (5) for the Hilbert transform on the real line, characterized by simplicity combined with high numerical accuracy. Its numerical bottleneck is the matrix-vector multiplication $\mathbf{A}F_{\text{int}}$. However, since $\mathbf{A}$ is an anti-symmetric Toeplitz matrix, the matrix-vector multiplication can be performed with complexity $O(N \log N)$ by exploiting the decomposition of $\mathbf{A}$ in discrete trigonometric matrices (2). The present algorithm offers the same advantages of the linear-spline version of the the method of Micchelli et al. [10], namely high precision and low complexity, although it is not as general as that in [10]. The main difference with Micchelli's algorithm is that the present evaluates the Hilbert transform on the same grid τ where the function is sampled. We believe that the simplicity of its implementation might be proved useful for many practical applications of the Hilbert transform.

Acknowledgments We are grateful to the referees for their valuable advice on how to improve the formulation and to complete the error analysis. We wish to thank Marco Restelli and Juan Vicente Gutierrez Santacreu for the discussions on the error analysis.

References

1. Brambilla, M.: Kinetic Theory of Plasma Waves. Oxford University Press (1998)
2. Bilato, R., Brambilla, M., Maj, O., Horton, L., Maggi, C., Stober, J.: Simulations of combined neutral beam injection and ion cyclotron heating with the toric-ssfpql package. Nucl. Fusion. **51**(10), 103034 (2011). http://stacks.iop.org/0029-5515/51/i=10/a=103034
3. Brambilla, M., Bilato, R.: Advances in numerical simulations of ion cyclotron heating of non-maxwellian plasmas. Nucl. Fusion. **49**(8), 085004 (2009). http://stacks.iop.org/0029-5515/49/085004
4. Huang, N.E., Shen, Z., Long, S.R., Wu, M.C., Shih, H.H., Zheng, Q., Yen, N.-C., Tung, C.C., Liu, H.H.: The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis. Proc. Math. Phys. Eng. Sci. **454**(1971), 903–995 (1998). http://www.jstor.org/stable/53161
5. Huang, N.E., Wu, Z.: A review on hilbert-huang transform: Method and its applications to geophysical studies. Rev. Geophys. **46**, 8755–1209 (2008)
6. Goswami, J.C., Hoefel, A.E.: Algorithms for estimating instantaneous frequency. Signal Process. **84**(8) (1423). doi:10.1016/j.sigpro.2004.05.016, http://www.sciencedirect.com/science/article/pii/S0165168404001033
7. Knockaert, L., Dhaene, T.: Causality determination and time delay extraction by means of the eigenfunctions of the hilbert transform. In: 12th IEEE Workshop on Signal Propagation Interconnects, 2008. SPI 2008, pp. 14 (2008). doi:10.1109/SPI.2008.4558337
8. Weideman, J.A.C.: Computing the hilbert transform on the real line. Math. Comput. **64**, 745 (1995)
9. Zhou, C., Yang, L., Liu, Y., Yang, Z.: A novel method for computing the hilbert transform with haar multiresolution approximation. J. Comput. Appl. Math. **223**(2), 585–597 (2009). doi:10.1016/j.cam.2008.02.006, http://www.sciencedirect.com/science/article/pii/S0377042708000526
10. Micchelli, C., Xu, Y., Yu, B.: On computing with the hilbert spline transform. Adv. Comput. Math. 1–24 (2012). doi:10.1007/s10444-011-9252-x
11. Wright, J.C., Valeo, E.J., Phillips, C.K., Bonoli, P.T., Brambilla, M.: Full wave simulations of lower hybrid waves in toroidal geometry with non-maxwellian electrons. Communincations Comput. Phys. **4**, 545 (2008)
12. Frigo, M., Johnson, S.: The design and implementation of fftw3. Proc. IEEE **93**(2), 216 –231 (2005). doi:10.1109/JPROC.2004.840301
13. Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T.: Numerical Recipes in Fortran. Cambridge University Press (1992)
14. Potts, D., Steidl, G.: Optimal trigonometric preconditioners for nonsymmetric toeplitz systems. Linear Algebra Appl. **281**(13), 265–292 (265). doi:10.1016/S0024-3795(98)10042-3, http://www.sciencedirect.com/science/article/pii/S0024379598100423
15. Taylor, M.E.: Partial Differential Equations I: Basic Theory. Springer, New York (1996)
16. Poppe, G.P.M., Wijers, C.M.J.: More efficient computation of the complex error function. ACM Trans. Math. Softw. **16**(1), 38–46 (1990)