Fast Gauss transforms with complex parameters using NFFTs

Stefan Kunis[*] Daniel Potts[†] Gabriele Steidl[‡]

June 13, 2005

We construct a fast algorithm for the computation of discrete Gauss transforms with complex parameters, capable of dealing with non equispaced points. Our algorithm is based on the fast Fourier transform at non equispaced knots and requires only $\mathcal{O}(N)$ arithmetic operations.

Key words and phrases. Gauss transform; Unequally spaced Fourier transforms; Fast algorithms; Chirped Gaussian; NFFT

1 Introduction

Given complex coefficients $\alpha_k \in \mathbb{C}$ and source knots $x_k \in [-\frac{1}{4}, \frac{1}{4}]$, our goal consists in the fast evaluation of the sum

$$f(y) = \sum_{k=1}^{N} \alpha_k \, \mathrm{e}^{-\sigma |y - x_k|^2} \tag{1.1}$$

at the target knots $y_j \in [-\frac{1}{4}, \frac{1}{4}]$, $j = 1, \ldots, M$, where $\sigma = a + \mathrm{i}b$, $a > 0, b \in \mathbb{R}$ denotes a complex parameter. Fast Gauss transforms for real parameters σ were developed, e.g., in [15, 8, 9]. In [12], we have specified a more general fast summation algorithm for the Gaussian kernel.

Recently, a fast Gauss transform for complex parameters σ with arithmetic complexity $\mathcal{O}(N \log N + M)$ was introduced by Andersson and Beylkin [1]. In this paper, we show how our general fast summation algorithm developed in [11, 12, 6] can be specified for the Gaussian kernel with complex parameter σ to obtain a fast Gauss transform with arithmetic complexity $\mathcal{O}(N+M)$. This results in a simpler algorithm than those in [1] with competitive performance in practice. We prove error estimates concerning the dependence of the computational speed on the desired accuracy and the parameters a and $|\sigma|$.

The heart of our algorithm is the discrete Fourier transform for non equispaced knots (NDFT), i.e., the evaluation of

$$g_j = \sum_{l=-\frac{n}{2}}^{\frac{n}{2}-1} \hat{g}_l \, \mathrm{e}^{2\pi \mathrm{i} l y_j}, \qquad j = 1, \ldots, M, \tag{1.2}$$

[*] kunis@math.uni-luebeck.de, University of Lübeck, Institute of Mathematics, D–23560 Lübeck
[†] potts@math.uni-luebeck.de, University of Lübeck, Institute of Mathematics, D–23560 Lübeck
[‡] steidl@math.uni-mannheim.de, University of Mannheim, Institute of Mathematics, D–68131 Mannheim

and its adjoint

$$\hat{g}_l = \sum_{j=1}^{M} g_j \,\mathrm{e}^{-2\pi \mathrm{i} l y_j}\,, \qquad l = -\frac{n}{2}, \dots, \frac{n}{2} - 1, \tag{1.3}$$

for arbitrary knots $y_j \in [-\frac{1}{2}, \frac{1}{2})$ and a degree $n \in 2\mathbb{N}$. Starting with [5, 2], there exists meanwhile a broad literature on fast Fourier transforms for non equispaced knots (NFFT), taking $\mathcal{O}((\rho n)\log(\rho n) + mM)$ arithmetic operations for the approximate computation of (1.2) and (1.3). Here ρ is an *oversampling factor* and m a *cut–off parameter*. Both parameters have to be chosen in accordance with the desired accuracy of the NFFT computations. In general, the approximation error introduced by the NFFT decreases exponentially in m with basis depending on ρ. A unified approach to NFFTs was given in [14, 13] and a corresponding software package can be found in [10].

The remainder of this paper is organised as follows: In Section 2, we modify our fast summation algorithm for the Gaussian kernel with complex parameter σ. In Section 3, we prove error bounds for the fast Gauss transform to justify its arithmetic complexity $\mathcal{O}(N+M)$. Finally, Section 4 presents various numerical experiments.

2 Fast Gauss transform

An algorithm for the fast computation of sums of the form

$$f(y_j) := \sum_{k=1}^{N} \alpha_k K(y_j - x_k) \qquad j = 1, \dots, M,$$

where K are special real–valued kernels was proposed for one dimension in [11] and generalised to the multivariate setting in [12]. We want to apply this method to the kernel $K(x) = \mathrm{e}^{-\sigma x^2}$, where

$$\sigma = a + b\mathrm{i} = |\sigma|\mathrm{e}^{\mathrm{i}\varphi}\,, \qquad a > 0,\ b \in \mathbb{R},\ \varphi := \arctan\frac{b}{a} \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$$

is a complex parameter. Following the lines of [11], we start by approximating K by a periodic function. For some period $p \geq 1$, let

$$K_{\mathrm{P}}(x) := \sum_{r \in \mathbb{Z}} K(x - p\,r) \tag{2.1}$$

denote the periodisation of K. The uniformly convergent Fourier series of K_{P} with Fourier coefficients

$$b_l = \frac{1}{p} \int_{-\frac{p}{2}}^{\frac{p}{2}} K_{\mathrm{P}}(x)\,\mathrm{e}^{-2\pi \mathrm{i} l x/p}\,\mathrm{d}x = \frac{1}{p} \int_{-\infty}^{\infty} \mathrm{e}^{-\sigma x^2}\,\mathrm{e}^{-2\pi \mathrm{i} l x/p}\,\mathrm{d}x = \frac{\sqrt{\pi}}{p\sqrt{\sigma}}\,\mathrm{e}^{-l^2\pi^2/(\sigma p^2)}, \tag{2.2}$$

where $\sqrt{\sigma} = |\sigma|^{\frac{1}{2}}\,\mathrm{e}^{\mathrm{i}\frac{\varphi}{2}}$ is truncated with a degree $n \in 2\mathbb{N}$ by

$$K_n(x) := \sum_{l=-\frac{n}{2}}^{\frac{n}{2}-1} b_l\,\mathrm{e}^{2\pi \mathrm{i} l x/p}\,. \tag{2.3}$$

For $y \in [-\frac{1}{4}, \frac{1}{4}]$, we approximate f by

$$\tilde{f}(y) := \sum_{k=1}^{N} \alpha_k K_n(y - x_k)\,, \tag{2.4}$$

where $x_k \in [-\frac{1}{4}, \frac{1}{4}]$. Substituting (2.3) into (2.4) we obtain

$$\tilde{f}(y_j) = \sum_{k=1}^{N} \alpha_k \sum_{l=-\frac{n}{2}}^{\frac{n}{2}-1} b_l \, \mathrm{e}^{2\pi \mathrm{i} l (y_j - x_k)/p} = \sum_{l=-\frac{n}{2}}^{\frac{n}{2}-1} b_l \left(\sum_{k=1}^{N} \alpha_k \, \mathrm{e}^{-2\pi \mathrm{i} l x_k/p} \right) \mathrm{e}^{2\pi \mathrm{i} l y_j/p}\,.$$

The expression in the inner brackets can be computed by an adjoint NFFT in $\mathcal{O}(N + n \log n)$ arithmetic operations. This is followed by n multiplications with b_l and completed by a NFFT to compute the outer sum in $\mathcal{O}(M + n \log n)$ arithmetic operations. In Section 3, we will prove that n depends only on the desired accuracy of our algorithm and on the complex parameter σ, but not on the numbers M and N. Thus, the overall arithmetic complexity of our algorithm is $\mathcal{O}(N + M)$, in particular this performance does not depend on the distribution of the points x_k and y_j. In summary, we propose the following algorithm.

Algorithm 2.1

1. *For* $l = -\frac{n}{2}, \ldots, \frac{n}{2} - 1$, *compute* $a_l = \sum\limits_{k=1}^{N} \alpha_k \, \mathrm{e}^{-2\pi \mathrm{i} l x_k/p}$ *by the adjoint NFFT.*

2. *For* $l = -\frac{n}{2}, \ldots, \frac{n}{2} - 1$, *compute the products* $d_l = a_l b_l$.

3. *For* $j = 1, \ldots, M$, *compute* $\tilde{f}(y_j) = \sum\limits_{l=\frac{n}{2}}^{\frac{n}{2}-1} d_l \, \mathrm{e}^{2\pi \mathrm{i} l y_j/p}$ *by the NFFT.*

Remark 2.2

1. Step 1 and 3 of our algorithm can be also realized by NDFTs in $\mathcal{O}(nN)$ and $\mathcal{O}(nM)$ operations, respectively, yielding an $\mathcal{O}(M + N)$ algorithm, too.

2. From another point of view, the function K_n was obtained by applying the trapezoidal quadrature rule within the interval $[-n/(2p), n/(2p)]$ as follows:

$$\mathrm{e}^{-\sigma x^2} = \frac{\sqrt{\pi}}{\sqrt{\sigma}} \int_{-\infty}^{\infty} \mathrm{e}^{-y^2 \pi^2/\sigma} \, \mathrm{e}^{2\pi \mathrm{i} x y} \, \mathrm{d}y \approx \frac{\sqrt{\pi}}{\sqrt{\sigma}} \frac{1}{p} \sum_{l=-\frac{n}{2}}^{\frac{n}{2}-1} \mathrm{e}^{-\left(\frac{l}{p}\right)^2 \pi^2/\sigma} \, \mathrm{e}^{2\pi \mathrm{i} x \frac{l}{p}}\,.$$

   It may be interesting to further reduce the number of summands in K_n by applying a more advanced quadrature rule as the generalized Gaussian quadrature proposed in [3] or the method in [4]. However, this involves a lot of work to determine the corresponding knots and weights which is beyond the scope of this paper.

$\square$

3 Error estimates

Beyond the well–known errors appearing in the NFFT computations, see [11], our algorithm produces for $j = 1, \dots, M$ the errors

$$\begin{aligned} |f(y_j) - \tilde{f}(y_j)| &= \left| \sum_{k=1}^{N} \alpha_k K_{\mathrm{ERR}}(y_j - x_k) \right| \\ &\le \|\boldsymbol{\alpha}\|_1 \, \|K_{\mathrm{ERR}}\|_\infty \,, \end{aligned}$$

where $K_{\mathrm{ERR}} := K - K_n$, $||K_{\mathrm{ERR}}||_\infty := \max\limits_{|x|\le\frac{1}{2}} |K_{\mathrm{ERR}}(x)|$ and $\|\boldsymbol{\alpha}\|_1 := \sum\limits_{k=1}^{N} |\alpha_k|$.

Theorem 3.1 *For K_n defined by* (2.3), *the following error estimate holds true*

$$\|K_{\mathrm{ERR}}\|_\infty \le 2\,\mathrm{e}^{-\frac{a(2p-1)^2}{4}} \left(1 + \tfrac{1}{ap(2p-1)}\right) + \tfrac{\sqrt{\pi}}{p\sqrt{|\sigma|}}\; \mathrm{e}^{-\frac{n^2\pi^2 a}{4p^2|\sigma|^2}} \left(1 + \tfrac{2|\sigma|^2 p^2}{n\pi^2 a}\right) . \tag{3.1}$$

Proof: By definition of K_{ERR}, we obtain for $x \in [-\frac{p}{2}, \frac{p}{2})$ that

$$|K_{\mathrm{ERR}}(x)| \le |K(x) - K_{\mathrm{P}}(x)| + |K_{\mathrm{P}}(x) - K_n(x)|,$$

and further by (2.1), (2.3), and (2.2) that

$$|K_{\mathrm{ERR}}(x)| \le \sum_{r\in\mathbb{Z}\setminus\{0\}} \left| \mathrm{e}^{-\sigma(x+p\,r)^2} \right| + \frac{\sqrt{\pi}}{p\sqrt{|\sigma|}} \left(\left| \mathrm{e}^{-\frac{n^2\pi^2}{4p^2\sigma}} \right| + 2 \sum_{l=\frac{n}{2}+1}^{\infty} \left| \mathrm{e}^{-\frac{l^2\pi^2}{p^2\sigma}} \right| \right).$$

Since $|\,\mathrm{e}^{a+\mathrm{i}\,b}| = \mathrm{e}^a$, this can be estimated for $x \in [-\frac{1}{2}, \frac{1}{2})$ by

$$\begin{aligned} |K_{\mathrm{ERR}}(x)| &\le 2\sum_{r=1}^{\infty} \mathrm{e}^{-\frac{a(2p\,r-1)^2}{4}} + \frac{\sqrt{\pi}}{p\sqrt{|\sigma|}} \left(\mathrm{e}^{-\frac{n^2\pi^2 a}{4p^2|\sigma|^2}} + 2 \sum_{l=\frac{n}{2}+1}^{\infty} \mathrm{e}^{-\frac{l^2\pi^2 a}{p^2|\sigma|^2}} \right) \\ &\le 2 \left(\mathrm{e}^{-\frac{a(2p-1)^2}{4}} + \frac{1}{2p} \int_{2p-1}^{\infty} \mathrm{e}^{-\frac{ay^2}{4}}\,\mathrm{d}y \right) + \frac{\sqrt{\pi}}{p\sqrt{|\sigma|}} \left(\mathrm{e}^{-\frac{n^2\pi^2 a}{4p^2|\sigma|^2}} + 2 \int_{\frac{n}{2}}^{\infty} \mathrm{e}^{-\frac{y^2\pi^2 a}{p^2|\sigma|^2}}\,\mathrm{d}y \right). \end{aligned}$$

Finally, we obtain the assertion by estimating the integrals with the help of

$$\int_v^\infty \mathrm{e}^{-u\,x^2}\,\mathrm{d}x = \int_0^\infty \mathrm{e}^{-u(x+v)^2}\,\mathrm{d}x \le \mathrm{e}^{-uv^2} \int_0^\infty \mathrm{e}^{-2uvx}\,\mathrm{d}x = \frac{\mathrm{e}^{-uv^2}}{2uv}\,, \qquad u, v > 0. \qquad \blacksquare$$

The error estimate (3.1) consists of two parts. The *first part* depends only on the real part of σ. In particular, we have that

$$E_1(a,p) := \mathrm{e}^{-\frac{a(2p-1)^2}{4}} < \varepsilon < 1 \quad \Leftrightarrow \quad p > \frac{1}{2} + \sqrt{\frac{\ln(1/\varepsilon)}{a}}.$$

Since $p \ge 1$, this part has no influence if a is large. For smaller a, the parameter p acts as a scaling factor as follows: for $p = 1$ and real part a we obtain the same error as for $p > 1$ and smaller real part $a/(2p-1)^2$. The *second part* of the error is determined by the real and the imaginary part of σ. Here we have that

$$E_2(\sigma,p,n) = \mathrm{e}^{-\frac{n^2\pi^2 a}{4p^2|\sigma|^2}} < \varepsilon < 1 \quad \Leftrightarrow \quad n > \frac{2p|\sigma|\sqrt{\ln(1/\varepsilon)}}{\pi\sqrt{a}} = \frac{2p\sqrt{a\,\ln(1/\varepsilon)}}{\pi\cos\varphi}. \tag{3.2}$$

With respect to our scaling factor p, we see that for a well localised Gaussian with parameter σ and $p=1$ we obtain the same error as for a less localised Gaussian with parameter σ/p^2.

Figure 3 illustrates the behaviour of the error. The right-hand sides of the figure show the level lines for E_2, while the left-hand sides present those of (3.1). The top and the middle images moreover demonstrate the role of the parameter p. In particular, the left hand sides show that the influence of the first summand E_1 in (3.1) becomes less dominant for small values of a if we increase the period p.

Remark 3.2

1. Instead of K_{P} we can use the truncated function

$$K_{\mathrm{T}}(x) := \sum_{r\in\mathbb{Z}} \chi_{[-\frac{1}{2},\frac{1}{2})}(x-r)K(x-r),$$

   with the characteristic function $\chi_{[-\frac{1}{2},\frac{1}{2})}$ of the interval $[-\frac{1}{2},\frac{1}{2})$ together with an appropriate boundary regularisation as described in [11, 6]. By the boundary regularisation, K_{T} becomes a smooth one periodic function with uniformly convergent Fourier series. For its truncated version K_n we use the approximation of the Fourier coefficients by the trapezoidal quadrature rule

$$b_l := \frac{1}{n}\sum_{j=-\frac{n}{2}}^{\frac{n}{2}-1} K_{\mathrm{T}}\left(\frac{j}{n}\right)\mathrm{e}^{-2\pi\mathrm{i}jl/n}.$$

   The corresponding error estimates show that we do not need an additional parameter p to cope with smaller values of a if we accept the additional precomputation effort due to the boundary regularisation.

2. In the Gauss transform one should also consider the case where the parameter a and thus the shape of the Gaussian changes in some 'fair' fashion with the number of samples $N=M$. If we spread points uniformly and scale a so that the Gaussian has fixed effective width when measured in units of the average spacing between points $1/N$, then we need $a=\mathcal{O}(N^2)$. Using (3.2) we see that $n=\mathcal{O}(N)$. Substituting this into the overall complexity estimate after (1.3) we get the complexity $\mathcal{O}(N\log N)$ using NFFTs and $\mathcal{O}(N^2)$ using NDFTs. However, if N becomes very large and $a=\mathcal{O}(N^2)$, the Gaussian becomes very small and it would be better to switch to real space techniques using a simple truncation technique.

□

4 Numerical experiments

We present numerical experiments in order to demonstrate the performance of our algorithm. All algorithms were implemented in C and tested on an AMD Athlon$^{\mathrm{TM}}$XP 2700+ with

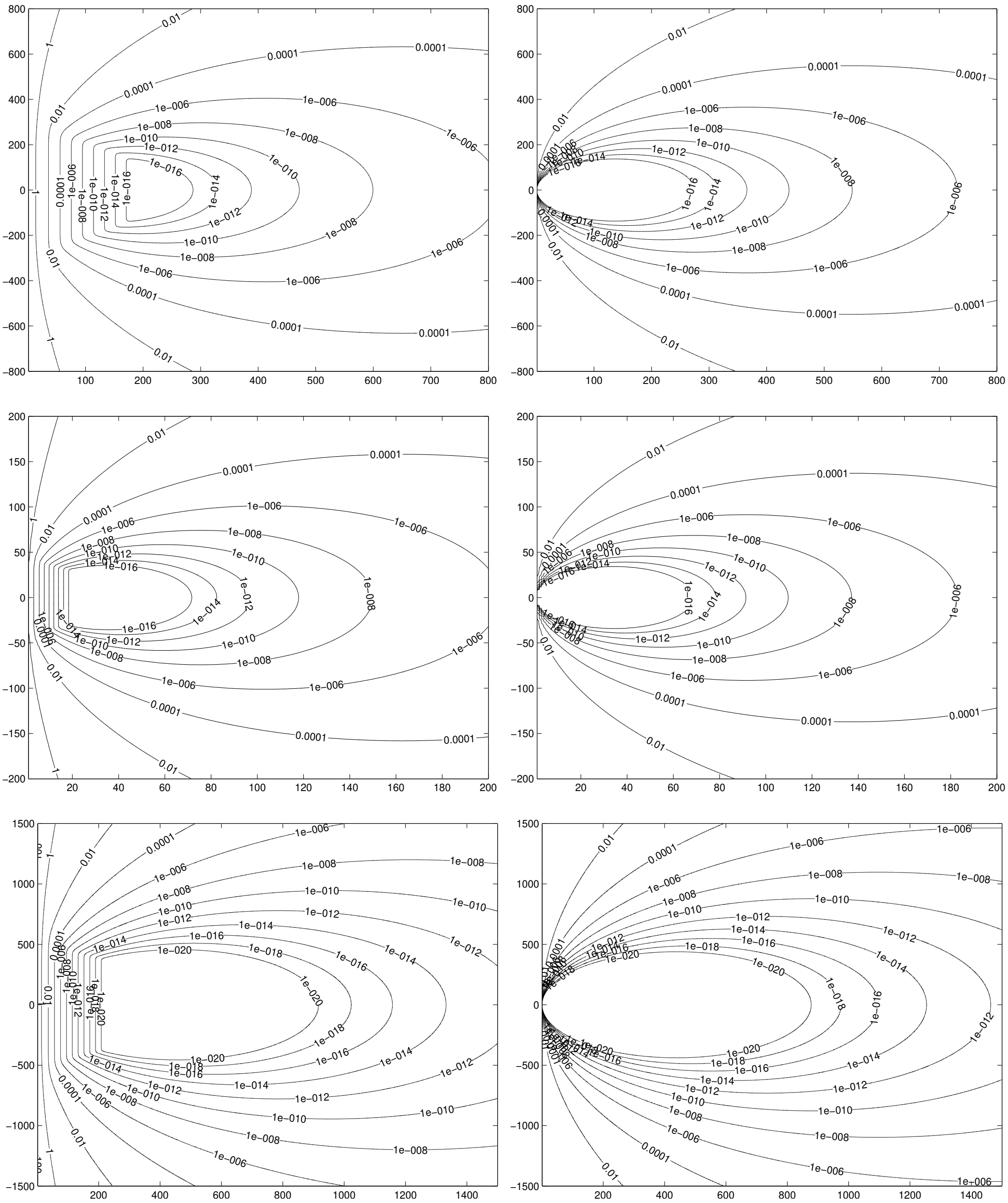

Figure 3.1: Level sets in the complex (a, b) plane for (3.1) (left) and for $E_2(\sigma, p, n)$ (right). Top: $p = 1$, $n = 64$, Middle: $p = 2$, $n = 64$, Bottom: $p = 1$, $n = 128$.

2GB main memory, SuSe-Linux (kernel 2.4.20-4GB-athlon, gcc 3.3) using double precision arithmetic. Moreover, we have used the libraries FFTW 3.0.1 [7] and NFFT 2.0.1 [10]. Throughout our experiments we have applied the NFFT package [10] with precomputed Kaiser–Bessel functions and an oversampling factor $\rho = 2$.

In our tests we have always chosen random source and target knots in $[-\frac{1}{4}, \frac{1}{4})$ and coefficients α_k uniformly distributed in the complex box $[-\frac{1}{2}, \frac{1}{2}] \times [-\frac{1}{2}, \frac{1}{2}]$i. We have considered the Gaussian kernels with

i) $\sigma = 4(138 + 100\mathrm{i})$,

ii) $\sigma = 20 + 40\mathrm{i}$.

The first parameter σ was taken from [1] in order to make the results comparable. The second choice of a considerably smaller σ serves to demonstrate the influence of the parameter p.

First we examine the errors that are generated by our fast Gauss transform. Figure 4.1 presents the error

$$E_\infty := \frac{\max\limits_{j=1,\ldots,M} |f(y_j) - \tilde{f}(y_j)|}{\sum\limits_{k=0}^{N} |\alpha_k|} \quad \approx \quad \|K_{\mathrm{ERR}}\|_\infty$$

introduced by our algorithms as function of the parameter n. These results confirm the error estimates in Theorem 3.1.

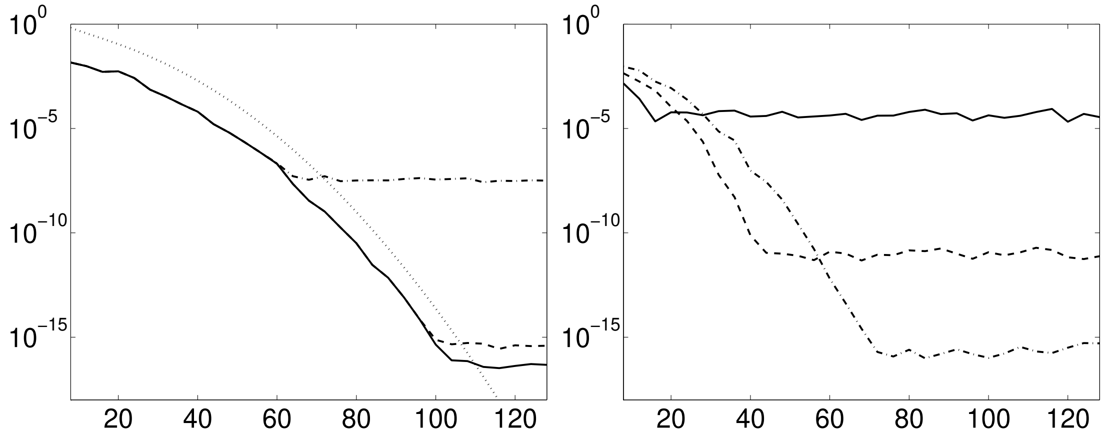

Figure 4.1: Error E_∞ for $n = 8, 12, 16, \ldots, 128$ and $N = M = 1000$.
Left: for $\sigma = 4(138 + 100\mathrm{i})$, fast Gauss transform with NDFT (solid), fast Gauss transform with NFFT, cut-off parameter $m = 3$ (dash-dot), fast Gauss transform with NFFT, cut-off parameter $m = 7$ (dashed), error estimate for $\|K_{\mathrm{ERR}}\|_\infty$ (dotted);
Right: for $\sigma = 20 + 40\mathrm{i}$, $p = 1$ (solid), $p = 1.5$ (dashed), $p = 2$ (dash-dot), and the fast Gauss transform with NFFT, cut-off parameter $m = 7$.

Finally, we compare the computation time of the straightforward summation, the straightforward summation with the precomputed matrix, the fast Gauss transform with NDFT, and the fast Gauss transform with NFFT for increasing $N = M$. The CPU time required by the four algorithms is shown in Table 4.1. As expected the fast Gauss transforms outperform the straightforward algorithms, yielding an $\mathcal{O}(N)$ complexity in both variants, whereas the NFFT–version is considerably faster.

5 Conclusions

We have presented a fast algorithm for the computation of sums of type (1.1) in $\mathcal{O}(N + M)$, respectively $\mathcal{O}(N \log N + M)$ arithmetic operations, depending on a 'fair' scaling of the Gaussian. We have proved error estimates concerning the dependence of the computational speed

$N = M$	direct alg.	with precomp.	fast GT, NDFT	fast GT, NFFT	error E_∞
64	$6.0e-04$	$3.0e-05$	$1.9e-03$	$1.2e-04$	$1.4e-15$
128	$2.4e-03$	$1.4e-04$	$3.9e-03$	$2.3e-04$	$1.7e-15$
256	$9.6e-03$	$1.3e-03$	$7.6e-03$	$4.3e-04$	$8.9e-16$
512	$3.8e-02$	$5.0e-03$	$1.5e-02$	$8.5e-04$	$5.8e-16$
1024	$1.5e-01$	$2.0e-02$	$3.0e-02$	$1.8e-03$	$6.0e-16$
2048	$6.2e-01$	$8.1e-02$	$6.1e-02$	$3.5e-03$	$2.3e-16$
4096	$2.5e+00$	$3.7e-01$	$1.2e-01$	$6.9e-03$	$2.4e-16$
8192	$9.9e+00$	$1.4e+00$	$2.4e-01$	$1.4e-02$	$1.4e-16$
16384	$4.0e+01$	*	$4.9e-01$	$2.7e-02$	$1.1e-16$
32768	$1.6e+02$	*	$9.7e-01$	$5.4e-02$	$1.1e-16$
65536	$6.4e+02$	*	$2.0e+00$	$1.1e-01$	$8.7e-17$
131072	$2.6e+03$	*	$3.9e+00$	$2.1e-01$	$7.9e-17$
262144	$1.0e+04$	*	$7.8e+00$	$4.3e-01$	$6.2e-17$
524288	*	*	$1.6e+01$	$8.8e-01$	*
1048576	*	*	$3.1e+01$	$1.7e+00$	*
2097152	*	*	$6.5e+01$	$3.6e+00$	*

Table 4.1: CPU-Time and error E_∞ for the fast Gauss transform for $\sigma = 4(138 + 100\mathrm{i})$, $n = 128$, and NFFT-cut-off parameter $m = 7$. Note that we used accumulated measurements in case of small times and the times/error (*) are not displayed due to the large response time or the limited size of memory.

on the desired accuracy and the parameters a and $|\sigma|$. The software for this algorithm including all described tests is available within the NFFT package [10, `./example/fastgauss`]. For the only available common example our algorithm is faster than those in [1].

References

[1] F. Andersson and G. Beylkin. The fast Gauss transform with complex parameters. *J. Comput. Physics*, 203:274 – 286, 2005.

[2] G. Beylkin. On the fast Fourier transform of functions with singularities. *Appl. Comput. Harmon. Anal.*, 2:363 – 381, 1995.

[3] G. Beylkin and L. Monzon. On generalized Gaussian quadratures for exponentials and their applications. *Appl. Comput. Harmon. Anal.*, 12:332 – 373, 2002.

[4] G. Beylkin and L. Monzon. On approximations of functions by exponential sums. *Appl. Comput. Harmon. Anal.*, to appear.

[5] A. Dutt and V. Rokhlin. Fast Fourier transforms for nonequispaced data. SIAM *J. Sci. Stat. Comput.*, 14:1368 – 1393, 1993.

[6] M. Fenn and G. Steidl. Fast NFFT based summation of radial functions. *Sampling Theory in Signal and Image Processing*, 3:1 – 28, 2004.

[7] M. Frigo and S. G. Johnson. FFTW, a C subroutine library. http://www.fftw.org/.

[8] L. Greengard and J. Strain. The fast Gauss transform. SIAM *J. Sci. Stat. Comput.*, 12:79 – 94, 1991.

[9] L. Greengard and X. Sun. A new version of the fast Gauss transform. *Doc. Math. J. DMV*, 3:575 – 584, 1998.

[10] S. Kunis and D. Potts. NFFT, Softwarepackage, C subroutine library. http://www.math.uni-luebeck.de/potts/nfft, 2002 – 2004.

[11] D. Potts and G. Steidl. Fast summation at nonequispaced knots by NFFTs. SIAM *J. Sci. Comput.*, 24:2013 – 2037, 2003.

[12] D. Potts, G. Steidl, and A. Nieslony. Fast convolution with radial kernels at nonequispaced knots. *Numer. Math.*, 98:329 – 351, 2004.

[13] D. Potts, G. Steidl, and M. Tasche. Fast Fourier transforms for nonequispaced data: A tutorial. In J. J. Benedetto and P. J. S. G. Ferreira, editors, *Modern Sampling Theory: Mathematics and Applications*, pages 247 – 270, Boston, 2001. Birkhäuser.

[14] G. Steidl. A note on fast Fourier transforms for nonequispaced grids. *Adv. Comput. Math.*, 9:337 – 353, 1998.

[15] J. Strain. The fast Gauss transform with variable scales. SIAM *J. Sci. Stat. Comput.*, 12:1131 – 1139, 1991.