ARTICLE IN PRESS

Computer Physics Communications ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Computer Physics Communications

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

INFFTM: Fast evaluation of 3d Fourier series in MATLAB with an application to quantum vortex reconnections☆

Marco Caliari*, Simone Zuccher

Department of Computer Science, University of Verona, Italy

ARTICLE INFO

Article history:
Received 11 May 2016
Received in revised form
1 December 2016
Accepted 5 December 2016
Available online xxxx

Keywords:
GPE
Fourier series evaluation
Time-splitting
NFFT

ABSTRACT

Although Fourier series approximation is ubiquitous in computational physics owing to the Fast Fourier Transform (FFT) algorithm, efficient techniques for the fast evaluation of a three-dimensional truncated Fourier series at a set of *arbitrary* points are quite rare, especially in MATLAB language. Here we employ the Nonequispaced Fast Fourier Transform (NFFT, by J. Keiner, S. Kunis, and D. Potts), a C library designed for this purpose, and provide a Matlab® and GNU Octave interface that makes NFFT easily available to the Numerical Analysis community. We test the effectiveness of our package in the framework of quantum vortex reconnections, where pseudospectral Fourier methods are commonly used and *local* high resolution is required in the post-processing stage. We show that the efficient evaluation of a truncated Fourier series at arbitrary points provides excellent results at a computational cost much smaller than carrying out a numerical simulation of the problem on a sufficiently fine regular grid that can reproduce comparable details of the reconnecting vortices.

Program summary

Program Title: INFFTM

Program Files doi: http://dx.doi.org/10.17632/nx5zzp5xxj.1

Licensing provisions: GNU GPLv3

Programming language: MATLAB/GNU Octave

Nature of problem: Evaluation of 3d Fourier series at arbitrary rectilinear grids or sets of arbitrary points, with application to quantum vortex reconnection.

Solution method: Fast n-dimensional linear transform of an n-d tensor (rectilinear grids), NFFT (Nonuniform Fast Fourier Transform, arbitrary points), time splitting spectral Fourier method (Gross–Pitaevskii equation for superfluids).

External routines/libraries: NFFT (optional but recommended).

1. Introduction

Fourier series approximation is a fundamental tool in computational physics. The main reason for its widespread usage is the availability of the Fast Fourier Transform (FFT) algorithm which allows to evaluate, in the three-dimensional case, a linear combination of $N_1N_2N_3$ trigonometric polynomials at a sample of $N_1N_2N_3$ points of a regular grid with a computational cost $\mathcal{O}(N_1N_2N_3(\log N_1 + \log N_2 + \log N_3))$ instead of the cost $\mathcal{O}(N_1^2N_2^2N_3^2)$ of a direct Discrete Fourier Transform.

FFT, as implemented in the FFTW [1] library, is nowadays available and easy to use on most high level computational tools. For instance, in the MATLAB[1] language the functions `ifft`, `ifft2`, and `ifftn` allow fast evaluation of trigonometric polynomials at a specific regular

☆ This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author.
E-mail address: marco.caliari@univr.it (M. Caliari).

[1] We will refer to MATLAB as the programming language used by the softwares Matlab® and GNU Octave.

http://dx.doi.org/10.1016/j.cpc.2016.12.004
0010-4655/

grid of points in one-, two- and n-dimensions, respectively. The fast evaluation of a three-dimensional truncated Fourier series at a set of *arbitrary* points is a more challenging task. NFFT [2] is a C library that approximates the evaluation of a truncated Fourier series at a set of M arbitrary points at cost $\mathcal{O}(N_1N_2N_3(\log N_1+\log N_2+\log N_3)+M|\log\varepsilon|^3)$, where ε is the desired accuracy (typically the double precision one). Although a Matlab® and GNU Octave interface is provided, to our knowledge it is not commonly used in computational science, maybe because of the NFFT algorithm formulation. For instance, given the set of coefficients $\{\hat{\psi}_k\}_{k=1}^N$, `ifft` performs the fast evaluation of

$$\frac{1}{N}\sum_{k=1}^{N}\hat{\psi}_k e^{2\pi i(k-1)x_n},\quad x_n=\frac{n-1}{N},\quad n=1,2,\ldots,N,$$

whereas, given the set of coefficients $\{\hat{\psi}_k\}_{k=-N/2}^{N/2-1}$, NFFT evaluates

$$\sum_{k=-N/2}^{N/2-1}\hat{\psi}_k e^{-2\pi ik\zeta_m},\quad \zeta_m\in\left[-\frac{1}{2},\frac{1}{2}\right),\quad m=1,2,\ldots,M$$

in an approximated and fast way.

The present work provides the package INFFTM, a MATLAB interface based on NFFT for the fast evaluation of a truncated Fourier series of a function $\psi:\Omega\to\mathbb{C}$ at a set Ξ_M of arbitrary M points in the computational domain $\Omega=\prod_{d=1}^3[a_d,b_d)$. The intermediate case of the evaluation of a truncated Fourier series at an arbitrary rectilinear grid $Y_{\boldsymbol{M}}$ with $M_1M_2M_3$ points is also addressed.

The motivation for developing this tool relies on the need for *localized* high resolution encountered in the post-processing stage of reconnecting quantum vortices [3,4]. The dynamics of quantum vortices and their possible reconnections are properly described by the Gross–Pitaevskii equation, which is normally solved by resorting to the Time Splitting pseudoSPectral (TSSP) approach. Since the details of reconnections are localized in space at scales much smaller than the vortex core size, employing standard FFT on regular grids would require an excessive, and thus infeasible, number of grid points in order to achieve the required resolution. However, in [5] it was found that, even in the presence of singular solutions, the number of Fourier coefficients required for an accurate description of the solution is not very large. The need for high local resolution at the post-processing stage, however, urgently demanded a tool for the efficient evaluation of such a *small* truncated Fourier series at a localized set of clustered points.

The paper is organized as follows. In Section 2 we present the details of Fourier series decomposition and evaluation in three dimensions, whereas in Section 3 we describe the framework for quantum fluids simulations. In Section 4 we outline the main functions of our INFFTM package and in Section 5 we show the result of the two main drivers performing a quantum vortex reconnection and some evaluations of the truncated Fourier series at different rectilinear grids and arbitrary points.

2. Fourier series decomposition and evaluation

This section, which is the core of the whole work, introduces the necessary notation and describes how the Fourier decomposition is performed, together with its successive evaluation at rectilinear grids or arbitrary points. Let

$$I_{\boldsymbol{N}}=\prod_{d=1}^{3}\{1,2,\ldots,N_d\},\quad \boldsymbol{N}=(N_1,N_2,N_3),\ N_d \text{ even}.$$

Given a complex function $\psi\in L^2(\Omega)$, with $\Omega=\prod_{d=1}^3[a_d,b_d)$, its truncated Fourier series is

$$\hat{\psi}(\boldsymbol{x})=\sum_{\boldsymbol{k}\in I_{\boldsymbol{N}}}\hat{\psi}_{\boldsymbol{k}}\mathcal{E}_{\boldsymbol{k}}(\boldsymbol{x}),\quad \hat{\psi}_{\boldsymbol{k}}\in\mathbb{C},\tag{1}$$

where $\boldsymbol{x}=(x_1,x_2,x_3)\in\Omega$, $\boldsymbol{k}=(k_1,k_2,k_3)$ is a multiindex and

$$\mathcal{E}_{\boldsymbol{k}}(\boldsymbol{x})=\prod_{d=1}^{3}\frac{e^{2\pi i(k_d-1-N_d/2)(x_d-a_d)/(b_d-a_d)}}{\sqrt{b_d-a_d}}.$$

Given the regular grid of points

$$X_{\boldsymbol{N}}=\{\boldsymbol{x}_{\boldsymbol{n}}=(x_{1,n_1},x_{2,n_2},x_{3,n_3})\}=\prod_{d=1}^{3}\{a_d+(n_d-1)h_d,\ n_d=1,2,\ldots,N_d\},$$

with $h_d=(b_d-a_d)/N_d$, the approximate Fourier coefficients $\hat{\psi}_{\boldsymbol{k}}$ are computed by the three-dimensional trapezoidal quadrature formula applied to the integral

$$\int_\Omega\psi(\boldsymbol{x})\overline{\mathcal{E}}_{\boldsymbol{k}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x},$$

where $\overline{\mathcal{E}}_{\boldsymbol{k}}(\boldsymbol{x})$ denotes the complex conjugate of $\mathcal{E}_{\boldsymbol{k}}(\boldsymbol{x})$. The function $\hat{\psi}(\boldsymbol{x})$ turns out to be an approximation of the original $\psi(\boldsymbol{x})$ which interpolates it at the points $X_{\boldsymbol{N}}$. The denominator in the basis functions $\mathcal{E}_{\boldsymbol{k}}$ assures the equivalence

$$\sum_{\boldsymbol{k}\in I_{\boldsymbol{N}}}|\hat{\psi}_{\boldsymbol{k}}|^2=\int_\Omega|\hat{\psi}(\boldsymbol{x})|^2\mathrm{d}\boldsymbol{x}=h_1h_2h_3\sum_{\boldsymbol{x}_{\boldsymbol{n}}\in X_{\boldsymbol{N}}}|\hat{\psi}(\boldsymbol{x}_{\boldsymbol{n}})|^2\approx\int_\Omega|\psi(\boldsymbol{x})|^2\mathrm{d}\boldsymbol{x}$$

for any domain $\Omega=\prod_{d=1}^3[a_d,b_d)$.

Please cite this article in press as: M. Caliari, S. Zuccher, INFFTM: Fast evaluation of 3d Fourier series in MATLAB with an application to quantum vortex reconnections, Computer Physics Communications (2016), http://dx.doi.org/10.1016/j.cpc.2016.12.004

The regular grid of points $X_{\boldsymbol{N}}$ can be represented in MATLAB by

```
[X{1:3}] = ndgrid(x{1:3})
```

where

```
x{d} = linspace(a(d),b(d),N(d)+1)'; x{d} = x{d}(1:N(d))
```

Given $\boldsymbol{n}$ = `[n(1),n(2),n(3)]`, we have

$$\begin{aligned}\boldsymbol{x_n} &= \texttt{[x\{1\}(n(1)),x\{2\}(n(2)),x\{3\}(n(3))]}\\ &= \texttt{[X\{1\}(n(1),n(2),n(3)),... X\{2\}(n(1),n(2),n(3)),... X\{3\}(n(1),n(2),n(3))]}.\end{aligned}$$

If `psi` denotes the MATLAB three-dimensional array containing the values of ψ at $X_{\boldsymbol{N}}$, then the three-dimensional array `psihat` of approximate Fourier coefficients $\hat{\psi}_{\boldsymbol{k}}$ is recovered using the fast Fourier transform

```
psihat = fftshift(fftn(psi)) * prod(sqrt(b - a) ./ N)
```

whose computational cost is $\mathcal{O}(N_1N_2N_3(\log N_1 + \log N_2 + \log N_3))$.

Given the truncated Fourier series approximation of a function, it is trivial to approximate its partial derivatives with respect to the directions x_d since

$$\partial_{x_d}\mathcal{E}_{\boldsymbol{k}}(\boldsymbol{x}) = \Lambda_{\boldsymbol{k},d}\mathcal{E}_{\boldsymbol{k}}(\boldsymbol{x}), \quad \Lambda_{\boldsymbol{k},d} = 2\pi\mathrm{i}(k_d - 1 - N_d/2)/(b_d - a_d), \tag{2}$$

which leads to

$$\partial_{x_d}\hat{\psi}(\boldsymbol{x}) = \sum_{\boldsymbol{k}\in I_{\boldsymbol{N}}} \Lambda_{\boldsymbol{k},d}\hat{\psi}_{\boldsymbol{k}}\mathcal{E}_{\boldsymbol{k}}(\boldsymbol{x}).$$

From the definition of $\Lambda_{\boldsymbol{k},d}$, it follows that

$$\begin{aligned}\int_\Omega |\nabla\hat{\psi}(\boldsymbol{x})|^2\mathrm{d}\boldsymbol{x} &= \sum_{\boldsymbol{k}\in I_{\boldsymbol{N}}}\left(\left|\Lambda_{\boldsymbol{k},1}\right|^2 + \left|\Lambda_{\boldsymbol{k},2}\right|^2 + \left|\Lambda_{\boldsymbol{k},3}\right|^2\right)|\hat{\psi}_{\boldsymbol{k}}|^2\\ &= -\sum_{\boldsymbol{k}\in I_{\boldsymbol{N}}}\left(\Lambda_{\boldsymbol{k},1}^2 + \Lambda_{\boldsymbol{k},2}^2 + \Lambda_{\boldsymbol{k},3}^2\right)|\hat{\psi}_{\boldsymbol{k}}|^2 = -\int_\Omega \nabla^2\hat{\psi}(\boldsymbol{x})\overline{\hat{\psi}(\boldsymbol{x})}\mathrm{d}\boldsymbol{x},\end{aligned}$$

where the last equivalence comes from integration by parts and taking into account the periodicity of $\hat{\psi}(\boldsymbol{x})$ in the computational domain Ω.

2.1. Evaluation of a truncated Fourier series at a rectilinear grid

The evaluation of a truncated Fourier series at the regular grid $X_{\boldsymbol{N}}$ can be implemented straightforwardly by employing the inverse fast Fourier transform

```
psihathat = ifftn(ifftshift(psihat)) / prod(sqrt(b - a) ./ N)
```

whose computational cost is $\mathcal{O}(N_1N_2N_3(\log N_1 + \log N_2 + \log N_3))$.

Given an arbitrary rectilinear grid $Y_{\boldsymbol{M}} = \prod_{d=1}^3\{y_{d,m_d},\ m_d = 1, 2, \ldots, M_d\} \subset \Omega$, we introduce the matrices

$$\mathcal{E}^d = \left(e^d_{m_dk_d}\right) = \frac{\mathrm{e}^{2\pi\mathrm{i}(k_d-1-N_d/2)(y_{d,m_d}-a_d)/(b_d-a_d)}}{\sqrt{b_d - a_d}} \in \mathbb{C}^{M_d\times N_d}, \quad d = 1, 2, 3$$

and then, for a single $\boldsymbol{y_m} \in Y_{\boldsymbol{M}}$, evaluate

$$\hat{\psi}(\boldsymbol{y_m}) = \sum_{\boldsymbol{k}\in I_{\boldsymbol{N}}}\hat{\psi}_{\boldsymbol{k}}\mathcal{E}_{\boldsymbol{k}}(\boldsymbol{y_m}) = \sum_{k_3=1}^{N_3} e^3_{m_3k_3}\left(\sum_{k_1=1}^{N_1} e^1_{m_1k_1}\left(\sum_{k_2=1}^{N_2}\hat{\psi}_{(k_1,k_2,k_3)}e^2_{m_2k_2}\right)\right).$$

We observe that the innermost and the middle sum can be evaluated, for each k_3, by a matrix–matrix product between $\hat{\psi}_{(\cdot,\cdot,k_3)}$ and the matrix $\mathcal{E}^2$ transposed, followed by a second matrix–matrix product with the matrix $\mathcal{E}^1$. The overall cost is $\mathcal{O}(N_1N_2M_2 + N_1M_2M_3)$.

If the result, i.e. a matrix of order $M_1 \times M_2$, is computed and stored for each k_3 (this cost is $\mathcal{O}(N_3(N_1N_2M_2 + N_1M_2M_3))$), then the outer sum corresponds to the multiplication of the term $e^3_{m_3k_3}$ for such matrices, for a total computational cost $\mathcal{O}(N_3M_1M_2M_3)$. A straightforward implementation in MATLAB of this strategy could be

```
for d = 1:3
  E{d} = exp(2*pi*1i * (y{d} - a(d)) / (b(d) - a(d)) ...
             * (-N(d)/2:N(d)/2 - 1)) / sqrt(b(d) - a(d));
end
psihaty = zeros(M);
for k3 = 1:N(3)
  temp = E{1} * (psihat(:,:,k3) * E{2}.'); % 2d evaluation
  for m3 = 1:M(3)
    psihaty(:,:,m3) = psihaty(:,:,m3) + temp * E{3}(m3,k3);
  end
end
```

Please cite this article in press as: M. Caliari, S. Zuccher, INFFTM: Fast evaluation of 3d Fourier series in MATLAB with an application to quantum vortex reconnections, Computer Physics Communications (2016), http://dx.doi.org/10.1016/j.cpc.2016.12.004

The routine `ndcovlt`[2] implements the same evaluation avoiding the two loops over N_3 and M_3. Both implementations do not need the explicit construction of $Y_{\boldsymbol{M}}$. In order to have an idea of the computational cost, we tested the evaluation of a series with 64^3 random complex Fourier coefficients at the regular grid $X_{\boldsymbol{M}}$ with 64^3 points and obtained what follows.

```
ifft
Elapsed time is 0.010371 seconds.
ndcovlt

error_inf =

   1.0725e-13

Elapsed time is 0.049872 seconds.
two loops

error_inf =

   1.0760e-13

Elapsed time is 0.201674 seconds.
```

in Matlab® R2014b and

```
ifft
Elapsed time is 0.0108922 seconds.
ndcovlt
error_inf =    9.5313e-14
Elapsed time is 0.0321529 seconds.
two loops
error_inf =    9.4936e-14
Elapsed time is 0.258168 seconds.
```

in GNU Octave 4.0.0. The first elapsed time is due to the inverse fast Fourier transform whose result is used to measure the error, in infinity norm, with respect to the other two methods. In the other two methods, the computational cost for the evaluation of $\mathcal{E}^d$, $d = 1, 2, 3$, is not considered. This test can be found at the end of the `igridftn.m` file and can be run, in GNU Octave, by `demo igridftn`. Due to the randomness of the Fourier coefficients the values of the errors are not perfectly reproducible. The implementation via `ndcovlt` is always much faster than the usage of nested loops; the factor is four in Matlab® R2014b and eight in GNU Octave 4.0.0, where JIT (Just-in-time accelerator) is not available. We observe that a truncated Fourier series can be evaluated at any rectilinear grid $Y_{\boldsymbol{M}}$ and that `ndcovlt` is very general as it can evaluate an n-dimensional truncated series at a rectilinear grid of points. For instance, it was used in [6] for the evaluation of truncated Hermite series.

2.2. Evaluation of a truncated Fourier series at arbitrary points

Given a set of arbitrary points $\Xi_M = \{\boldsymbol{\xi}_m = (\xi_{1m}, \xi_{2m}, \xi_{3m}),\ m = 1, 2, \ldots, M\} \subset \Omega$, it is possible to evaluate $\hat{\psi}$ at a single point $\boldsymbol{\xi}_m$ by firstly computing the vectors

$$\mathcal{E}^d = \left(e^d_{k_d}\right) = \frac{\mathrm{e}^{2\pi \mathrm{i}(k_d - 1 - N_d/2)(\xi_{dm} - a_d)/(b_d - a_d)}}{\sqrt{b_d - a_d}} \in \mathbb{C}^{N_d}, \quad d = 1, 2, 3$$

and then

$$\hat{\psi}(\boldsymbol{\xi}_m) = \sum_{\boldsymbol{k} \in I_{\boldsymbol{N}}} \hat{\psi}_{\boldsymbol{k}} \mathcal{E}_{\boldsymbol{k}}(\boldsymbol{\xi}_m) = \sum_{k_1=1}^{N_1} \sum_{k_2=1}^{N_2} \sum_{k_3=1}^{N_3} \hat{\psi}_{(k_1,k_2,k_3)} e^1_{k_1} e^2_{k_2} e^3_{k_3}. \tag{3}$$

Finally, a loop over m has to be performed. This can be done in MATLAB by the code

```
for m = 1:M
  E{1}(:,1,1) = exp(2*pi*1i * (Xi(1,m) - a(1)) / (b(1) - a(1)) ...
                    * (-N(1)/2:N(1)/2 - 1)) / sqrt(b(1) - a(1));
  E{2}(1,:,1) = exp(2*pi*1i * (Xi(2,m) - a(2)) / (b(2) - a(2)) ...
                    * (-N(2)/2:N(2)/2 - 1)) / sqrt(b(2) - a(2));
  E{3}(1,1,:) = exp(2*pi*1i * (Xi(3,m) - a(3)) / (b(3) - a(3)) ...
                    * (-N(3)/2:N(3)/2 - 1)) / sqrt(b(3) - a(3));
  EE = bsxfun(@times,E{1} * E{2},E{3});
  psihatxi(m) = sum(psihat(:) .* EE(:));
end
```

at a computational cost $\mathcal{O}(MN_1N_2N_3)$. This implementation is limited to three dimensions, but it can be extended to any n-dimensional truncated series. Unfortunately, due to the construction of the vectors $\mathcal{E}^d$ inside a loop, this implementation turns out to be quite inefficient.

[2] It was originally written by Jaroslav Hajek for the linear-algebra package of GNU Octave.

Please cite this article in press as: M. Caliari, S. Zuccher, INFFTM: Fast evaluation of 3d Fourier series in MATLAB with an application to quantum vortex reconnections, Computer Physics Communications (2016), http://dx.doi.org/10.1016/j.cpc.2016.12.004

2.2.1. NFFT

Provided the set of points $\{\boldsymbol{\zeta}_m = (\zeta_{1m}, \zeta_{2m}, \zeta_{3m}),\ m = 1, 2, \ldots, M\}$, with $-1/2 \le \zeta_{dm} < 1/2$, NFFT performs a fast approximation of

$$\hat{f}(\boldsymbol{\zeta}_m) = \sum_{\boldsymbol{k}\in I_{\boldsymbol{N}}} \left(\hat{f}_{\boldsymbol{k}} \prod_{d=1}^{3} \mathrm{e}^{-2\pi \mathrm{i}(k_d - 1 - N_d/2)\zeta_{dm}} \right).$$

Given the coefficients $\{\hat{\psi}_{\boldsymbol{k}}\}_{\boldsymbol{k}}$ and the evaluation points $\Xi_M = \{\boldsymbol{\xi}_m\}_m$, Fourier series evaluation at Ξ_M can be approximated by calling the NFFT algorithm with

$$\zeta_{dm} = \mathrm{mod}\left(\frac{\xi_{dm} - a_d}{a_d - b_d}, 1 \right) - \frac{1}{2}, \quad d = 1, 2, 3$$

and coefficients

$$\hat{f}_{\boldsymbol{k}} = \hat{\psi}_{\boldsymbol{k}} \prod_{d=1}^{3} \frac{\mathrm{e}^{\pi \mathrm{i}(k_d - 1 - N_d/2)}}{\sqrt{b_d - a_d}} = \hat{\psi}_{\boldsymbol{k}} \frac{(-1)^{k_1+k_2+k_3-3-(N_1+N_2+N_3)/2}}{\prod_{d=1}^{3} \sqrt{b_d - a_d}},$$

where $\mathrm{mod}(x, y)$ is the usual remainder of the Euclidean division of x by y, $\mathrm{mod}(x, y) = x - y\,\lfloor x/y \rfloor$.

We first checked our MATLAB interface to NFFT by evaluating a truncated series of 64^3 random complex Fourier coefficients at the regular grid $X_{\boldsymbol{M}}$ with 64^3 points (for which the inverse FFT is available) and compared the result in infinity norm obtaining

```
NFFT

error_inf =

   5.3705e-14

Elapsed time is 2.267151 seconds.
```

in Matlab® R2014b and

```
NFFT
error_inf =    6.3161e-14
Elapsed time is 2.74056 seconds.
```

in GNU Octave 4.0.0. Although the asymptotic cost of the NFFT is smaller than the evaluation at the regular grid, for this number of coefficients and points of evaluation NFFT turns out to be about 20 times slower than `ndcovlt`.

In order to have an idea of the computational cost in a real case usage, the evaluation of a series with 64^3 coefficients on $M = 1000$ random points in Ω takes

```
one-loop
Elapsed time is 3.295020 seconds.
NFFT

error_inf =

   1.2296e-13

Elapsed time is 0.130843 seconds.
```

in Matlab® R2014b and

```
one-loop
Elapsed time is 3.95735 seconds.
NFFT
error_inf =    1.2488e-13
Elapsed time is 0.0957451 seconds.
```

in GNU Octave 4.0.0. Here we observe a speed-up of about 40 of the NFFT approach over a straightforward implementation. The measured error is between the two evaluations. These tests can be found at the end of the `innft3.m` file and can be run, in GNU Octave, by `demo infft3`.

Please cite this article in press as: M. Caliari, S. Zuccher, INFFTM: Fast evaluation of 3d Fourier series in MATLAB with an application to quantum vortex reconnections, Computer Physics Communications (2016), http://dx.doi.org/10.1016/j.cpc.2016.12.004

3. Application to quantum vortex reconnections

Turbulence, ubiquitously present in nature, is dominated by reconnection of vortical structures. Examples of reconnecting vortex tubes can be found in quantum turbulence [7–9], whose dynamics is properly described by the Gross–Pitaevskii equation (GPE) [10,11]

$$\frac{\partial \psi}{\partial t} = \frac{\mathrm{i}}{2}\nabla^2\psi + \frac{\mathrm{i}}{2}\left(1 - |\psi|^2\right)\psi, \tag{4}$$

where ψ is the complex wave function. Quantum vortices are infinitesimally thin filaments of concentrated vorticity in a unitary background density, $\rho(\boldsymbol{x}) = |\psi(\boldsymbol{x})|^2 \to 1$ when $|\boldsymbol{x}| \to \infty$. On the vortex centerlines the density tends to zero and the phase of the wave function ψ is not defined. In the dimensionless units of Eq. (4), the quantum of circulation is $\Gamma = 2\pi$ and the healing length, i.e. the lengthscale of the core vortex over which reconnections occur, is $\xi = 1$. GPE admits energy

$$E = \frac{1}{2}\int |\nabla\psi(\boldsymbol{x})|^2 \,\mathrm{d}\boldsymbol{x} + \frac{1}{4}\int (1 - |\psi(\boldsymbol{x})|^2)^2 \,\mathrm{d}\boldsymbol{x}. \tag{5}$$

Time splitting Fourier methods [12,3,13,4,5] are normally used to compute the numerical solution of the GPE (4). Because these methods rely on periodic boundary conditions for the solutions restricted to a bounded physical domain, initial conditions that are not periodic must be mirrored in the directions lacking periodicity [12], with a consequent increase of the degrees of freedom and computational effort [5].

Recent studies focusing on the topological details of quantum-vortex reconnections [4] have emphasized the need for an accurate description of the vortex centerline, which can be achieved by costly high-resolution numerical simulations of Eq. (4). On the other hand, it is possible to resort to more affordable approaches combined with an *a posteriori* accurate evaluation of the solution on a finer grid, as proposed in Ref. [5].

In order to exploit the second option, following [12], we consider a fully three-dimensional reconnection originating from two perpendicular straight vortices, whose cross sections are two-dimensional vortices. The wave function of a single two-dimensional vortex in the (s_1, s_2) plane and centered in $(0, 0)$ is $\rho(\sqrt{s_1^2 + s_2^2})^{1/2}\mathrm{e}^{\mathrm{i}\theta(s_1,s_2)} = f(\sqrt{s_1^2 + s_2^2})\mathrm{e}^{\mathrm{i}\theta(s_1,s_2)}$, where $f(\sqrt{s_1^2 + s_2^2}) = f(r)$ is a function to be determined whereas the phase is $\theta(s_1, s_2) = \mathrm{atan2}(s_2, s_1)$. By requiring the wave function to be the steady solution of Eq. (4), we find [5] that $\rho(r)$ satisfies

$$\rho'' + \frac{\rho'}{r} - \frac{(\rho')^2}{2\rho} - \frac{2\rho}{r^2} + 2(1 - \rho)\rho = 0, \tag{6}$$

with boundary conditions $\rho(0) = 0$, $\rho(\infty) = 1$. Instead of computing the numerical solution of this equation, it is possible to resort to a high-order Padé approximation of $\rho(r)$ [5]. It is known [14,15] that diagonal Padé approximations of $\rho(r)$ retain only even degrees at both the numerator and denominator, that is

$$\rho(r) \approx \rho_q(r) = \frac{a_1 r^2 + a_2 r^4 + \cdots + a_q r^{2q}}{1 + b_1 r^2 + b_2 r^4 + \cdots + b_q r^{2q}}. \tag{7}$$

The coefficients of a certain approximation $\rho_q(r)$ are computed by substituting the analytic expressions $\rho_q(r)$, $\rho_q'(r)$ and $\rho_q''(r)$ in Eq. (6) and by nullifying the coefficients of the first $2q - 1$ terms r^{2k}. The choice $q = 4$ leads to an algebraic equation of degree 8 for a_1 which can be solved numerically. Once a_1 is known, all the other coefficients can be computed analytically (see [5] for the details). Their expression is reported and used in the code file `sf4pade.m`. A straight vortex in a three-dimensional domain can be obtained by the extrusion of the above two-dimensional wave function along the vortex center line. A nontrivial initial condition generated by the superimposition of multiple straight vortices is simply the product of their wave functions.

3.1. Numerical discretization

After restricting the unbounded domain $\mathbb{R}^3$ to the computational domain $\Omega = \prod_{d=1}^{3}[a_d, b_d)$ in which the initial solution is periodic, Eq. (4) can be split into the *kinetic* and *potential* parts

$$\frac{\partial u}{\partial t} = \frac{\mathrm{i}}{2}\nabla^2 u \tag{8a}$$

$$\frac{\partial v}{\partial t} = \frac{\mathrm{i}}{2}\left(1 - |v|^2\right)v, \tag{8b}$$

and the Time Splitting pseudoSPectral (TSSP) approach can be employed, as done in [5]. Eq. (8a) is solved exactly in time within the Fourier spectral space, whereas Eq. (8b) is solved exactly owing to the fact that $|v|$ is preserved by the equation. Therefore,

$$v(\tau, \boldsymbol{x}) = \exp\left(\frac{\tau\mathrm{i}}{2}\left(1 - |v(0, \boldsymbol{x})|^2\right)\right)v(0, \boldsymbol{x}) \tag{9}$$

for any $\boldsymbol{x}$ in the spatial domain. By introducing $\mathrm{e}^{\tau\mathcal{A}}u_n(\boldsymbol{x})$ and $\mathrm{e}^{\tau\mathcal{B}(v_n(\boldsymbol{x}))}v_n(\boldsymbol{x})$ to denote the two partial numerical solutions, the numerical approximation $\psi_{n+1}(\boldsymbol{x})$ of $\psi(t_{n+1}, \boldsymbol{x})$ at time $t_{n+1} = (n + 1)\tau$ is recovered by the so-called Strang splitting

$$\psi_{n+1/2}(\boldsymbol{x}) = \mathrm{e}^{\tau\mathcal{A}}\mathrm{e}^{\frac{\tau}{2}\mathcal{B}(\psi_n(\boldsymbol{x}))}\psi_n(\boldsymbol{x})$$

$$\psi_{n+1}(\boldsymbol{x}) = \mathrm{e}^{\frac{\tau}{2}\mathcal{B}(\psi_{n+1/2}(\boldsymbol{x}))}\psi_{n+1/2}(\boldsymbol{x}).$$

Strang splitting preserves the discrete finite mass in the computational domain Ω and is second order accurate in time. We refer the reader to [16] for higher-order time splitting methods.

Please cite this article in press as: M. Caliari, S. Zuccher, INFFTM: Fast evaluation of 3d Fourier series in MATLAB with an application to quantum vortex reconnections, Computer Physics Communications (2016), http://dx.doi.org/10.1016/j.cpc.2016.12.004

3.2. Other applications of the NFFT tool

The Gross–Pitaevskii equation is a model not only for superfluids but also for Bose–Einstein condensates (see [17] for a review). In the second framework, the typical formulation is

$$\mathrm{i}\frac{\partial\psi}{\partial t} = -\frac{1}{2}\nabla^2\psi + V\psi + \beta|\psi|^{2\sigma}\psi,$$

where $V\colon \mathbb{R}^3 \to \mathbb{R}$ is a scalar potential, β a real constant and $\sigma > 0$. In this case the corresponding energy is

$$E = \frac{1}{2}\int |\nabla\psi(\boldsymbol{x})|^2 \mathrm{d}\boldsymbol{x} + \int V|\psi(\boldsymbol{x})|^2 \mathrm{d}\boldsymbol{x} + \frac{\beta}{\sigma+1}\int |\psi(\boldsymbol{x})|^{2\sigma+2}\mathrm{d}\boldsymbol{x}$$

and the Strang splitting method described above can still be applied without any modification. We notice that space discretizations which are not regular (see, for instance, [5] for nonuniform finite differences and [18] for finite elements) provide results that are difficult to compare with those obtained via pseudospectral approaches, which are available only on regular grids. INFFTM allows the evaluation at arbitrary rectilinear grids and sets of arbitrary points making the comparison of these results possible.

Another interesting application where NFFT is a valuable tool is the so called *magnetic* Schrödinger equation

$$\mathrm{i}\frac{\partial\psi}{\partial t} = \frac{1}{2}(\mathrm{i}\nabla + A)^2\psi + V\psi,$$

where $A\colon \mathbb{R}^3 \to \mathbb{R}^3$ is the vector potential which can be chosen divergence free owing to Coulomb's gauge. Besides the kinetic and the potential parts, the *advection* part

$$\frac{\partial w}{\partial t} = A\cdot\nabla w$$

has to be considered and then combined with the others in a splitting scheme. The advection part can be solved, for instance, by the characteristics method and the value of $w(\tau, \boldsymbol{x})$ at the departure point of the characteristics can be recovered by NFFT. We refer to [19] for further details.

4. Description of the programs

On developing the code, we realized that some functions naturally apply to any space dimension. On the contrary, others are specific for the three-dimensional case, which is the object of the present work. Therefore, we used the following convention: function names ending in '3' are specific and for the three-dimensional case only, whereas the others can work in any space dimension. The only exception is `igridftn` which calls `ndcovlt`, originally developed by Jaroslav Hajek and not designed for the trivial one-dimensional case (see Section 4.1 and Appendix B). In what follows we describe only the implementation in three dimensions.

As written in the `README` file, before using the package, NFFT has to be installed. We refer to Appendix A for the instructions on the installation in a Linux environment. After that, the correct path to the NFFT library has to be given in the file `nfftpath.m`. If the NFFT library is not installed, the package will work anyway, but the evaluation of a three-dimensional truncated Fourier series at a set of arbitrary points will be extremely slow.

4.1. Functions for Fourier series evaluation

The two main functions are `igridftn` and `infft3`. They implement the evaluation of the truncated Fourier series (1) at a rectilinear grid (`ndgrid` format) and at an arbitrary set of points, respectively. The calls are similar

```
psi = igridftn(psihat,a,b,y)
psi = infft3(psihat,a,b,Xi)
```

`psihat` being the three-dimensional array of Fourier coefficients, `a` and `b` the limits of the physical domain Ω (in the form `[a(1),a(2),a(3)]` and `[b(1),b(2),b(3)]`), `y` a cell array containing in the column vector `y{d}` the dth projection of the points and `Xi` a two-dimensional array containing in the dth row the dth component of the points.

The simple function

```
plotiso3(x,data,iso)
```

invokes the MATLAB program `isosurface` to plot the isosurface of level `iso` of the real input `data` corresponding to `ndgrid(x{1:3})`. Since `isosurface` in Matlab® R2014b requires the data in `meshgrid` format, `plotiso3` performs the permutation

```
data = permute(data,[2,1,3]);
```

4.2. Functions for superfluid simulation by GPE

As described in Section 3, here we focus on the particular application to quantum vortex reconnections. The time integration of GPE is carried out by the main function `sfrun`. Given an initial solution as a function of `x{1:3}`, `sfrun` first computes some preliminary

Please cite this article in press as: M. Caliari, S. Zuccher, INFFTM: Fast evaluation of 3d Fourier series in MATLAB with an application to quantum vortex reconnections, Computer Physics Communications (2016), http://dx.doi.org/10.1016/j.cpc.2016.12.004

quantities (`sfpregpe`), such as $\Lambda_{\boldsymbol{k},d}$ (see Eq. (2)), then computes initial and final mass and energy of the system (`sfEm`, see Eq. (5)), and finally it performs time integration by Strang splitting method (`sfgpe`) and store the structure `sf` of the solution in a MATLAB `'-v6'` format file at `nsteps1+` equally distributed time steps. The structure `sf` contains the fields `pdb` (a row vector of length six consisting of the physical domain boundaries), `psipdb` (a complex 3d-array of the values of the wave function at the grid in the physical domain), `mirror` (a row vector of length three for the mirroring flags) and `t` (the simulation time). From the structure `sf` it is possible to recover the Fourier coefficients of the solution `psipdb` by invoking the function

```
[psihat,a,b] = sf2psihat(sf)
```

All the previous functions work in any space dimension. The functions `sfsvl3` and `sfic3` generate respectively a single straight vortex in a three-dimensional domain and the superimposition of multiple vortices. The function `sfview3` simply extracts the grid points and the density of the wave function from the structure `sf` and plots a given isosurface level through the function `plotiso3`.

4.3. Evaluation within a vortex tube

The study of vortex reconnections in quantum fluids requires high spatial resolution in order to extract the vortex centerlines with enough accuracy, and this is especially true in the neighborhood of the reconnection event (see Section 3). Instead of evaluating the physical solution at a finer rectilinear grid within the whole physical domain, it is more convenient to evaluate the solution only within vortex tubes, i.e. where high resolution is really needed.

Function `sftubeeval3` has been designed especially for this purpose. Its input arguments are the structure `sf`, which defines completely ψ on an equispaced grid in the physical domain, and `rhobar`, a vector containing the values of the density ρ that define the vortex tubes. For example, if `rhobar=0.2` (a single value), then function `sftubeeval3` first extracts points from $X_{\boldsymbol{N}}$ for which $\rho \leq 0.2$. Then, if $\boldsymbol{\xi}_m$ denotes the mth point within the vortex tube (corresponding to a certain $\boldsymbol{x}_{\boldsymbol{n}} \in X_{\boldsymbol{N}}$) and h_d the step-size of $X_{\boldsymbol{N}}$ in direction d, a small regular grid of step-size $h_d/3$ made of only 27 points centered in $\boldsymbol{\xi}_m$ is generated for each m. Finally, `sftubeeval3` returns the new set of points and $\rho = |\psi|^2$ evaluated at these points by NFFT. In order to retrieve smaller vortex tubes containing enough points, the input `rhobar` should be a vector. In this case the process described above is repeated up to the last value of ρ and the output of `sftubeeval3` is the set of points Ξ_M on successive refined grids for which $\rho \leq$ `rhobar(end)`, together with their corresponding values of ρ.

4.4. Drivers

The two drivers `sfdrv3` and `evaldrv3` were written for the convenience of the user, as they perform the numerical simulation and the visualizations exactly as described in the next section.

5. Numerical experiments

We solve the GPE equation (4) in the physical domain $[-20, 20]^3$. The initial solution is given by the superimposition of two straight vortices, passing through the points $(2, 0, 0)$ and $(-2, 0, 0)$ and oriented as $(0, 1, 0)$ and $(0, 0, 1)$, respectively. In order to make this initial condition periodic at the boundaries, the computational domain has to be set to $\Omega = [-20, 60)^3$ and the initial solution has to be mirrored along the three directions. This can be accomplished by setting the field `mirror` to `[true,true,true]` in the structure `sf` associated to the initial solution. While ensuring the periodicity of the solution, mirroring does not force the periodicity of the derivatives. In the computational domain we select $\boldsymbol{N}_1 = (80, 80, 80)$, yielding a (coarse) regular grid $X_{\boldsymbol{N}_1}$ with a constant space step size of 1 along each direction. The solution is computed up to the final time $T = 20$ with 200 time steps. The initial solution, in the original physical domain, is shown in Fig. 1 by plotting the isosurface $\rho = 0.1$ extracted from the original data at the regular grid $X_{\boldsymbol{N}_1}$, through the function `sfview3`.

The solution at the final time $T = 20$ is reported in Fig. 2. The left plot shows the isosurface $\rho = 0.1$ in the whole physical domain, whereas the right plot shows a zoom at the isolevel $\rho = 0.05$, which should guarantee a better description of the vortex centerlines ($\rho \to 0$ therein). Clearly, none of the plots in Fig. 2 allows to discriminate whether the reconnection has occurred or not. Moreover, reducing the isolevel of ρ makes things worse in that vortex tubes appear disconnected due to the low spatial resolution characterizing the original data.

In order to increase the details, we evaluate the solution at a finer Cartesian equispaced grid $Y_{\boldsymbol{M}_1}$, with $\boldsymbol{M}_1 = (321, 321, 321)$ in the physical domain $[-20, 20]^3$ by the function `igridftn`. By extracting the isosurface corresponding to $\rho = 0.0012$, the vortex tubes become much better defined (see Fig. 3, left) clearly indicating that a reconnection has occurred. Selecting the same isolevel for the original data at $X_{\boldsymbol{N}_1}$ yields an almost empty plot. Since our interest is in the neighborhood of the reconnection, instead of evaluating the solution at equally-spaced points, it is more convenient to evaluate the solution at a coarser *nonequispaced* rectilinear grid $Y_{\boldsymbol{M}_2}$, with $\boldsymbol{M}_2 = (281, 281, 281)$ points denser around the origin, always by `igridftn`. The isosurface $\rho = 0.0012$ (see Fig. 3, right) provides a much better result than the equispaced case (Fig. 3, left) in terms of clear vortex cores, which now appear completely connected.

Finally, by employing the strategy described in Section 4.3 with the sequence `rhobar=[0.2,0.05]` and then plotting points corresponding to $\rho \leq 0.0012$, we obtain the left plot in Fig. 4. The set Ξ_M obtained by `sftubeeval3` has $M = 32\,022$ points and the number of points corresponding to $\rho \leq 0.0012$ is 809. Similar vortex tubes can be obtained without evaluation at finer grids only by resorting to high-resolution simulations. An example is reported in the right plot of Fig. 4, which shows the isosurface $\rho = 0.0012$ for the solution at the original regular grid $X_{\boldsymbol{N}_2}$ with $\boldsymbol{N}_2 = (228, 228, 228)$ and 1000 time steps.

The script to run the GPE simulation with $\boldsymbol{N}_1$ is `sfdrv3` (CPU time about 14 s), whereas the script to perform evaluation is `evaldrv3` (CPU time about 17 s with NFFT installed, about 247 s without). The simulation with $\boldsymbol{N}_2$ took about 420 min.

6. Conclusions

We have developed the package INFFTM for the fast evaluation of three-dimensional truncated Fourier series at general rectilinear grids and sets of arbitrary points. The two main functions, `igridftn` and `infft3`, are written in plain MATLAB language, work in Matlab® and

Please cite this article in press as: M. Caliari, S. Zuccher, INFFTM: Fast evaluation of 3d Fourier series in MATLAB with an application to quantum vortex reconnections, Computer Physics Communications (2016), http://dx.doi.org/10.1016/j.cpc.2016.12.004

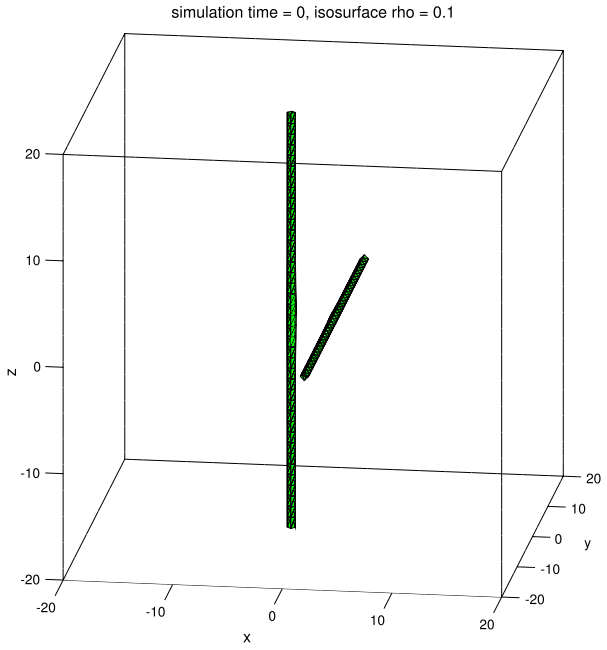

Fig. 1. Isosurface level 0.1 for the density of the initial solution at X_{N_1}.

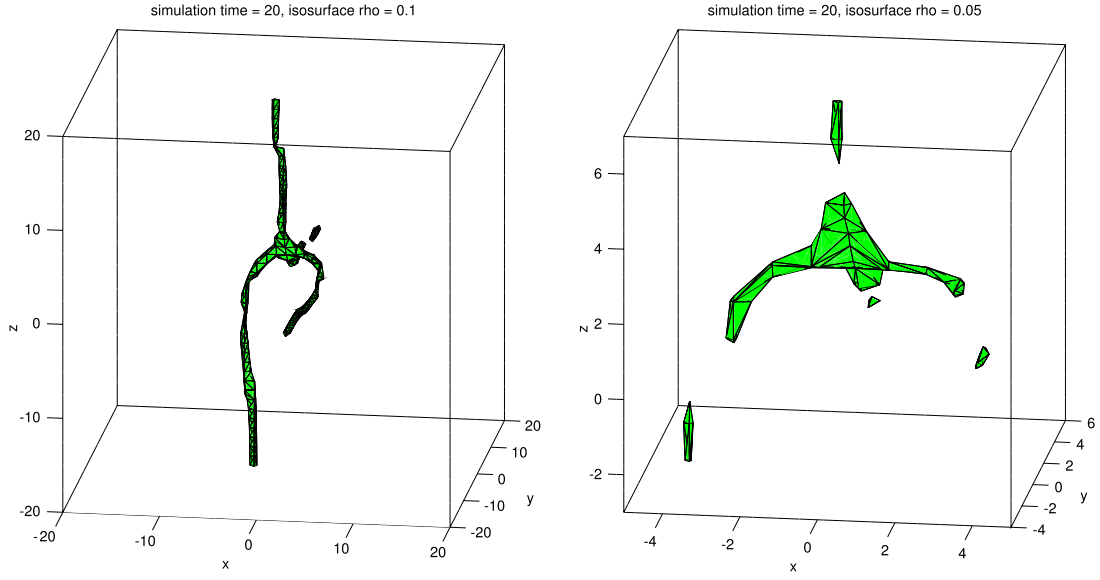

Fig. 2. Isosurface level 0.1 (left) and zoom of the isosurface level 0.05 (right) for the density of the final solution at X_{N_1}.

GNU Octave and are based on two efficient, although not widespread, tools, namely `ndcovlt` by J. Hajek and NFFT by J. Keiner, S. Kunis, and D. Potts. We have demonstrated the effectiveness of `igridftn` and `infft3` in the framework of quantum vortex reconnections. A proper post-processing of the numerical data obtained by running a cheap simulation of the vortex dynamics modeled by the Gross–Pitaevskii equation provides details on the reconnecting vortices that are comparable to costly high-resolution simulations. These promising results highlight the potential of INFFTM to become a standard MATLAB library for applications involving Fourier series approximation.

Appendix A. Installation of NFFT in a Linux environment

The necessary information for installation of NFFT is available in the file `README`. Here we briefly summarize the procedure.

On writing this paper, the latest release of NFFT was `nfft-3.3.2.tar.gz`.[3] It can be built in the usual way (`./configure` and `make`) and, in order to compile the Matlab® mex interface, it must to be configured by

```
./configure --with-matlab=/path/to/matlab --enable-openmp
```

[3] Available at https://www-user.tu-chemnitz.de/~potts/nfft/.

Please cite this article in press as: M. Caliari, S. Zuccher, INFFTM: Fast evaluation of 3d Fourier series in MATLAB with an application to quantum vortex reconnections, Computer Physics Communications (2016), http://dx.doi.org/10.1016/j.cpc.2016.12.004

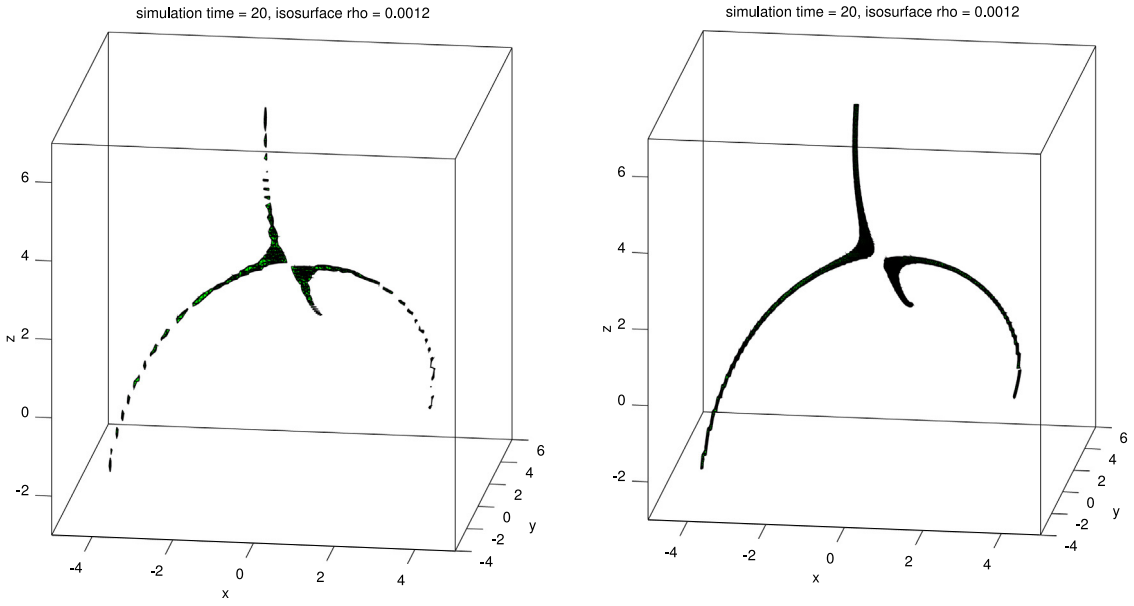

Fig. 3. Isosurface level 0.0012 for the density of the final solution evaluated at $Y_{\boldsymbol{M}_1}$ (left, equispaced grid) and at $Y_{\boldsymbol{M}_2}$ (right, nonequispaced grid).

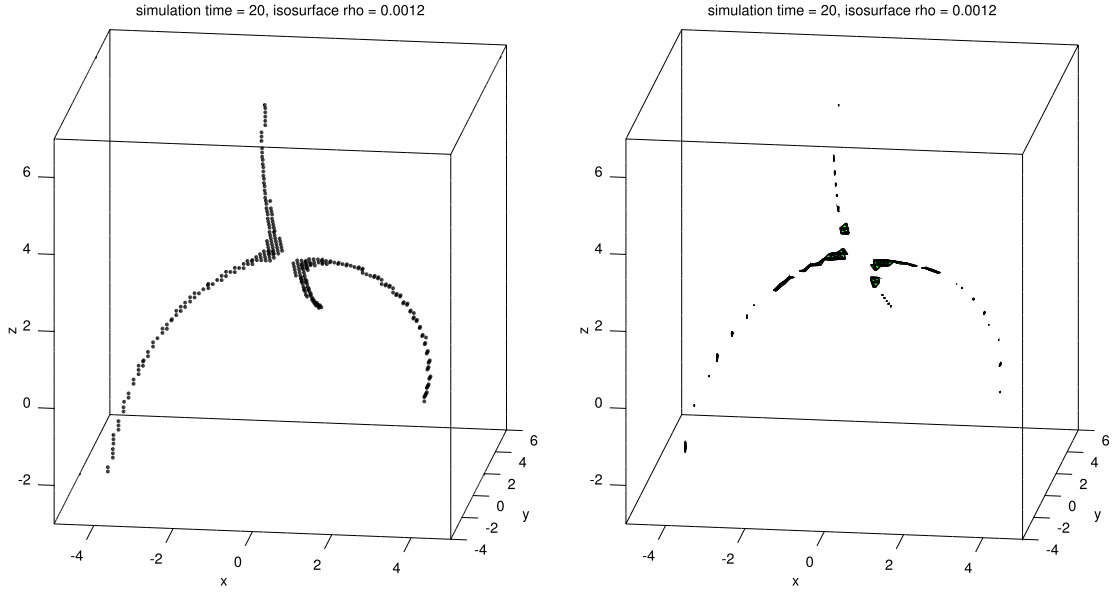

Fig. 4. Isosurface level 0.0012 for the density of the final solution evaluated at Ξ_M (left) and for the density of the final solution computed and evaluated at $X_{\boldsymbol{N}_2}$, with $\boldsymbol{N}_2 = (228, 228, 228)$ (right).

The installation stage `make install` is not necessary. Then, the correct path to `nfft-3.3.2/matlab/nfft` has to be given in the `nfftpath.m` file. The path to Matlab can be obtained by the command

```
matlab -n
```

The installation in GNU Octave is similar and requires, in the configuration stage,

```
./configure --with-octave=/path/to/octave --enable-openmp
```

If GNU Octave is installed in the system as a standard package, then it is enough to give

```
./configure --with-octave --enable-openmp
```

In order to check the correct installation of the NFFT library it is possible to run the test at the end of the `nfftpath.m` file or, in GNU Octave, to run `demo nfftpath`. The demo (provided by the original NFFT library) requires about 20 s and has to be considered passed if the string `A two dimensional example` appears. The presence of some `NaN` values in the output is *not* a symptom of a failure.

Please cite this article in press as: M. Caliari, S. Zuccher, INFFTM: Fast evaluation of 3d Fourier series in MATLAB with an application to quantum vortex reconnections, Computer Physics Communications (2016), http://dx.doi.org/10.1016/j.cpc.2016.12.004

Appendix B. Auxiliary files and workarounds

Versions of GNU Octave before 4.0.0 have no `flip` function, which is required by the code and distributed in the `aux` folder. GNU Octave 4.0.0. has a bug[4] with `fftshift` and `ifftshift` not working on three-dimensional arrays. Patched working functions are distributed in the `aux` folder.

Matlab® R2014b has a bug[5] preventing, from time to time, to load `nfftmex.mexa64`. The workaround is to load the library as soon as Matlab® R2014b is started. This can be achieved, for instance, by running the script `aux/mlloadnfft`.

In the folder `aux` we provide also the functions `igridft2`, `infft`, and `infft2`. Although `igridftn` and `ndconvlt` can work in two dimensions, in order to evaluate a two-dimensional truncated Fourier series at a rectilinear grid it is much simpler to use a double matrix–matrix product as done in `igridft2.m`. The functions `infft` and `infft2` apply NFFT to a one-dimensional and to a two-dimensional array of Fourier coefficients, respectively. A demonstration of their usage can be found at the end of the files and can be run, in GNU Octave, by `demo infft` and `demo infft2`. We notice that there is no need for a specific function `igridft1` because the evaluation of a one-dimensional truncated Fourier series at an arbitrary set Ξ_M of points is straightforwardly obtained by the matrix–vector product

```
E = exp(2*pi*1i * (Xi(:) - a) * (-N/2:N/2 - 1) / (b - a)) / ...
        sqrt(b - a);
psi = E * psihat;
```

whose computational cost is $\mathcal{O}(NM)$. However, the one-dimensional `infft` might be more convenient for large M as its computational cost is $\mathcal{O}(N \log N + M\,|\log \varepsilon|)$ and, in general, it is also more accurate.

References

[1] M. Frigo, S.G. Johnson, Proc. IEEE 93 (2) (2005) 216–231.
[2] J. Keiner, S. Kunis, D. Potts, ACM Trans. Math. Software 36 (4) (2009) 19:1–19:30.
[3] S. Zuccher, M. Caliari, A.W. Baggaley, C.F. Barenghi, Phys. Fluids 24 (125108) (2012) 1–21.
[4] S. Zuccher, R.L. Ricca, Phys. Rev. E 92 (6) (2015) 061001.
[5] M. Caliari, S. Zuccher, 2016, arXiv:1603.05022 [math.NA].
[6] M. Caliari, S. Rainer, Comput. Phys. Comm. 184 (3) (2013) 812–823.
[7] W.F. Vinen, Phil. Trans. R. Soc. A 366 (1877) (2008) 2925–2933.
[8] M.S. Paoletti, D.P. Lathrop, Ann. Rev. Cond. Mat. Phys. 2 (2011) 213–234.
[9] C.F. Barenghi, L. Skrbek, K.R. Sreenivasan, Proc. Natl. Acad. Sci. USA 111 (1) (2014) 4647–4652.
[10] L.P. Pitaevskii, Sov. Phys.—JETP 13 (1961) 451–454.
[11] E.P. Gross, J. Math. Phys. 4 (1963) 195–207.
[12] J. Koplik, H. Levine, Phys. Rev. Lett. 71 (9) (1993) 1375–1379.
[13] A.J. Allen, S. Zuccher, M. Caliari, N. Proukakis, N.G. Parker, C.F. Barenghi, Phys. Rev. A 90 (2014) 013601.
[14] N.G. Berloff, J. Phys. A: Math. Gen. 37 (2004) 1617–1632.
[15] S. Nazarenko, R. West, J. Low Temp. Phys. 132 (1) (2003) 1–10.
[16] M. Thalhammer, M. Caliari, C. Neuhauser, J. Comput. Phys. 228 (3) (2009) 822–832.
[17] W. Bao, Y. Cai, Kinet. Relat. Models 6 (1) (2013) 1–135.
[18] M. Thalhammer, J. Abhau, J. Comput. Phys. 231 (20) (2012) 6665–6681.
[19] M. Caliari, A. Ostermann, C. Piazzola, J. Comput. Appl. Math. Available online 5 September 2016.

[4] Bug no. #45207.
[5] Bug no. 961694.

Please cite this article in press as: M. Caliari, S. Zuccher, INFFTM: Fast evaluation of 3d Fourier series in MATLAB with an application to quantum vortex reconnections, Computer Physics Communications (2016), http://dx.doi.org/10.1016/j.cpc.2016.12.004