XFFT - a MATLAB toolbox for computing eXtended fast Fourier transforms

Ines Melzer

October 21, 2015

This toolbox includes the computation of the Butterfly sparse fast Fourier transform [3], the fast Laplace transform [5, 4] and a fast Fourier transform for complex evaluation nodes [4]. We give a short overview, how to use this toolbox. For readers who are not familiar with the class concept it is recommended to study the corresponding section of the MATLAB help, labeled as Object-Oriented Programming. The current release version `XFFT 1.0` of the toolbox is available at http://sines.de. under the GNU General Public License version 3 as published by the Free Software Foundation [1].

1 Butterfly sparse fast Fourier transform

An introduction to the butterfly sparse Fourier transform can be found in [6]. In the following, we use the notation of [3]. For a given space dimension $d \in \mathbb{N}$, a nonharmonic bandwidth $N = 2^L$, $L \in \mathbb{N}$, a set of frequencies $\widetilde{T} = \{\boldsymbol{\xi}_k \in [0,N]^d : k = 1,\dots,M_2\}$, a set of Fourier coefficients $\hat{u}_k \in \mathbb{C}$, $k = 1,\dots,M_2$, and a set of evaluation nodes $\widetilde{X} = \{\boldsymbol{x}_j \in [0,N]^d : j = 1,\dots,M_1\}$, the butterfly sparse Fourier transform computes the sums

$$f_j := f(\boldsymbol{x}_j) = \sum_{k=1}^{M_2} \hat{u}_k \mathrm{e}^{2\pi \mathrm{i} \boldsymbol{\xi}_k \boldsymbol{x}_j / N}, \quad j = 1,\dots,M_1. \tag{1}$$

The sums (1) with $d \geq 2$ and $M_1 = M_2 = \mathcal{O}(N^{d-1})$, and sampling sets $\widetilde{T}$, $\widetilde{X}$ on smooth $(d-1)$-dimensional manifolds can be computed in $\mathcal{O}(N^{d-1} \log N(|\log \varepsilon| + \log N)^{d+1})$ floating point operations, where $\varepsilon > 0$ denotes the target accuracy.
The butterfly sparse Fourier transform is implemented for the dimensions $d = 1,2,3,4$. The XFFT class consists of the main classes `@sparse_FFT1D` for $d = 1$, `@sparse_FFT2D` for $d = 2$, `@sparse_FFT3D` for $d = 3$, and `@sparse_FFT4D` for $d = 4$. To shorten notation, we write `sparse_FFT*D`, where the * can be chosen as 1,2,3,4. There are some classes called `@tree1D`, `@tree2D`, `@tree3D`, and `@tree4D`, which generate the trees of the dyadic decompositions of the domains X and Ω, see [3, Algorithm 1]. Note that the user must not call the `tree`-classes by oneself. The `@sparse_FFT*D` generates the tree automatically in a pre-computation step. The user has to set the properties `N`, `MX`, `MOmega`, `p`, and `option` given in Table 1. Here `MX` and `MOmega` are the spatial and frequency nodes $\widetilde{X}$ and $\widetilde{\Omega}$, respectively. Moreover, the user has to ensure that the input data fulfill the range conditions listed in Table 1, because there is no check for wrong input data implemented. If we want to choose a higher approximation rank $p \geq 6$ we should use for `option` the property `'Ltb'` or `'Ltb*'`, because these variants are more stable, see [3] for details. Then we create an object of class `@sparse_FFT*D`,

`plan` = `sparse_FFT` ∗ `D`(`MX`, `MOmega`, `p`, `N`, `option`).

Afterwards we have to set the coefficient vector `fhat` $= (\hat{f}_k)_{k=1,\dots,M_2} \in \mathbb{C}^{M_2 \times 1}$ and can compute the sums `f` $:= (f(\boldsymbol{x}_j))_{j=1}^{M_1}$ in (1) approximatively by applying

`f` = `mtimes`(`plan`, `fhat`).

All available properties and methods of an object can be listed with the MATLAB functions properties and methods,

`properties(obj)` or `methods(obj)`,

and documentation is provided by the MATLAB `help` and `doc` commands, for example

`help sparse_FFT1D` or `doc sparse_FFT1D`.

You can find all numerical tests of [3] in the directory `numerical_tests/paper_KuMe2012`. The numerical files and figures of [3] are listed in table 3. For each `compute` file exists a `show` file, which plots the result of the computations. For more details see the m-Files.

Property	**Range**	**Description**
`N`	$\mathtt{N} = 2^L, L \in \mathbb{N}$	domain parameter
`MX`	$[0,N]^{M_1 \times d}$	sampling nodes $\widetilde{X}$
`MOmega`	$[0,N]^{M_2 \times d}$	sampling nodes $\widetilde{\Omega}$
`p`	$\mathtt{p} \in \mathbb{N}$	local expansion degree
`option`	see Table 2	
`invG`	$\mathbb{C}^{p \times p}$	matrix $\boldsymbol{G}^{-1}$, see [3, section 2.3.1.]
`H`	$\mathbb{C}^{p \times p}$	matrix $\boldsymbol{H}$, see [3, section 2.3.1.]
Lleft	$\mathbb{C}^{p \times p}$	Lagrange matrix for a left son box, see [3, section 2.3.2.]
Lright	$\mathbb{C}^{p \times p}$	Lagrange matrix for a right son box, see [3, section 2.3.2.]
Laglast	$\mathbb{C}^{d(M_1 \times p)}$	function values of Lagrange polynomials in `MX` just for `option='Ltb*'`, see [3, section 2.3.2.]

Table 1: Properties of the class `sparse_FFT*D` for each dimension $d = 1, 2, 3, 4$ and their default values.

option	**Description**
`'Mtb'`	monomial-type basis, see [3, section 2.3.1.]
`'Ltb'`	Lagrange-type basis, see [3, section 2.3.2.]
`'Ltb*'`	Lagrange-type basis with precomputation of the Lagrange polynomials in `MX`.

Table 2: Possible types for the property `option` of the class `sparse_FFT`.

figure	**file**
Figure 4.1. (a), (b)	`box1D/compute1D_relative_error_box`
Figure 4.2. (a)	`box1D/show_condition_number_G`
Figure 4.2. (b)	`box1D/show_condition_number_Lag`
Figure 4.3. (a),(b)	`compute1D_relative_error`
Figure 4.3. (c),(d)	`compute2D_relative_error`
Figure 4.4.	`compute1D_relative_error_dependence_L`
Figure 4.5.	`compute1D_times`
Figure 4.6. (a),(b)	`compute2D_times`
Figure 4.6. (c),(d)	`compute3D_times`
Figure 4.6. (e),(f)	`compute4D_times`

Table 3: Files of the numerical tests in [3].

2 Fast Fourier transform for complex evaluation nodes

Let now evaluation or spatial nodes $y_1 > \ldots > y_{M_1} > 0$, frequency nodes $\xi_1 > \ldots > \xi_{M_2} > 0$, and coefficients $\hat{f}_k \in \mathbb{C}$ for $k = 1, \ldots, M_2$ be given. The fast Laplace transform is the approximate computation of sums

$$f_j := f(y_j) = \sum_{k=1}^{M_2} \hat{f}_k \mathrm{e}^{-y_j \xi_k}, \quad j = 1, \ldots, M_1 \tag{2}$$

by [4, Algorithm 1]. The XFFT class consists of the main class `@XFLT`, which includes the fast Laplace transform [4, Algorithm 1], to compute (2). Furthermore, it includes [4, Algorithm 2], the Fast Fourier transform for nonequispaced complex nodes, to compute sums of the form

$$f_j := \sum_{k=0}^{N-1} \hat{f}_k \mathrm{e}^{2\pi \mathrm{i} kj/N} \mathrm{e}^{-y_j k}, \quad j = 0, \ldots, N-1, \tag{FFLT}$$

$$f_j := \sum_{k=1}^{N} \hat{f}_k \mathrm{e}^{2\pi \mathrm{i} k x_j/N} \mathrm{e}^{-y_j k}, \quad j = 1, \ldots, M_1, \tag{NFLT}$$

$$f_j := \sum_{k=1}^{M_2} \hat{f}_k \mathrm{e}^{2\pi \mathrm{i} \xi_k x_j/N} \mathrm{e}^{-y_j \xi_k}, \quad j = 1, \ldots, M_1, . \tag{(BSFLT) or (NNFLT)}$$

For more details, we refer the reader to [4]. The first input parameter will set the form of the transform, (FLT), (NFLT), (BSFLT) or (NNFLT). The possible transforms are listed in Table 4.

transform, `Ftype`	**Description**
`'LT'`	Laplace transform
`'FFLT'`	fast Fourier Laplace transform
`'NFLT'`	nonequispaced fast Fourier Laplace transform
`'BSFLT'`	butterfly sparse fast Fourier Laplace transform
`'NNFLT'`	nonequispaced fast Fourier Laplace transform in spatial and frequency domain

Table 4: Possible transforms.

For each transform, the following values have to be set. First, the user has to set sampling nodes `MY`$= (y_1, \ldots, y_{M_1}) \in [0, \infty)^{M_1}$ in ascending order. Furthermore, the user can choose between the target accuracy ε or the approximation rank q of the Laplace transform. Finally the sampling nodes `MOmega`$= (\xi_1, \ldots, \xi_{M_2}) \in [0, \infty)^{M_2}$ has to be set in ascending order, too. For the Laplace transform, you can initialize the plan for fixed target accuracy `epsilon` by

```
plan=XFLT('LT',MY,'accuracy',epsilon,'MOmega',MOmega)
```

or for a fixed approximation rank `q` by

```
plan=XFLT('LT',MY,'rank',q,'MOmega',MOmega).
```

The computation of $\mathbf{f} = (f_j)_{j=1,\ldots,M_1}$ is done by the function call

```
f=mtimes(plan,fhat),
```

where `fhat` $= (\hat{f}_k)_{k=1,\ldots,M_2}$. For the NFLT and NNFLT the NFFT software library is needed, see [2] and

http://www-user.tu-chemnitz.de/~potts/nfft/.

For an arbitrary plan, the `'Ftype'` have to been choosen by possible values listed in Table 4. Moreover, the optional input arguments `varargin` are listed for each transformation in Table 5. The plan can be initialized by

```
plan=XFLT(Ftype,MY,option,value,varargin),
```

where `option` has to be set as `'rank'` or `'accuracy'` with the appropriate `value` `q` $\in \mathbb{N}$ or `epsilon` $\in (0, 1)$, respectively.

Ftype Ftype	**Property**	**Range**	**Description**
LT	'MOmega'	$(\xi_1, \ldots, \xi_{M_2})^\top \in [0, N]^{M_2}$, $\xi_1 < \cdots < \xi_{M_2}$	frequency nodes
FFLT			everything is automatically set
NFLT	'N'	$\mathtt{N} \in \mathbb{N}$	length of the NFFT
	'MX'	$(x_0, \ldots, x_{N-1})^\top \in [0, N]^N$	spatial nodes
	'libdir'	path	NFFT library directory
BSFLT	'N'	$\mathtt{N} = 2^L, L \in \mathbb{N}$	domain parameter
	'MX'	$(x_1, \ldots, x_{M_1})^\top \in [0, N]^{M_1}$	spatial nodes
	'MOmega'	$(\xi_1, \ldots, \xi_{M_2})^\top \in [0, N]^{M_2}$, $\xi_1 < \cdots < \xi_{M_2}$	sampling nodes in frequency domain
	'BSFFTrank'	$\mathtt{p} \in \mathbb{N}$	local expansion degree
NFFLT	'N'	$\mathtt{N} = 2^L, L \in \mathbb{N}$	domain parameter
	'MX'	$(x_1, \ldots, x_{M_1})^\top \in [0, N]^{M_1}$	spatial nodes
	'MOmega'	$(\xi_1, \ldots, \xi_{M_2})^\top \in [0, N]^{M_2}$, $\xi_1 < \cdots < \xi_{M_2}$	frequency nodes
	'libdir'	path	NFFT library directory

Table 5: Properties of the class XFLT and their default values.

References

[1] Free Software Foundation. GNU General Public License version 3 (GPLv3). http://www.gnu.org/licenses/, 6 2007.

[2] J. Keiner, S. Kunis, and D. Potts. Using NFFT 3—a software library for various nonequispaced fast Fourier transforms. *ACM Trans. Math. Software*, 36(4):Art. 19, 30, 2009.

[3] S. Kunis and I. Melzer. A stable and accurate butterfly sparse Fourier transform. *SIAM J. Numer. Anal.*, 50(3):1777–1800, 2012.

[4] S. Kunis and I. Melzer. Fast evaluation of real and complex exponential sums. *Preprint*, 2014.

[5] V. Rokhlin. A fast algorithm for the discrete Laplace transformation. *J. Complexity*, 4(1):12–32, 1988.

[6] L. Ying. Sparse Fourier transform via butterfly algorithm. *SIAM J. Sci. Comput.*, 31(3):1678–1694, 2009.