
FNFTpy Documentation

Release 0.1

Christoph Mahnke

Jul 17, 2018

CONTENTS:

1	Module overview	1
2	Korteweg-de-Fries equation	3
2.1	kdvv - calculate the Nonlinear Fourier Transform	3
2.2	kdvv_wrapper - interact with FNFT library	3
2.3	get, set and view options for kdvv_wrapper	4
2.4	options structure	4
3	Nonlinear Schroedinger equation with periodic boundaries	5
3.1	nsep - calculate the Nonlinear Fourier Transform	5
3.2	nsep_wrapper - interact with FNFT library	6
3.3	get, set and view options for nsep_wrapper	6
3.4	options structure	7
4	Nonlinear Schroedinger equation with vanishing boundaries	9
4.1	nsev - calculate the Nonlinear Fourier Transform	9
4.2	nsev_wrapper - interact with FNFT library	10
4.3	get, set and view options for nsev_wrapper	11
4.4	options structure	11
5	example functions	13
	Python Module Index	15
	Index	17

MODULE OVERVIEW

This file is part of FNFTpy. FNFTpy provides wrapper functions to interact with FNFT, a library for the numerical computation of nonlinear Fourier transforms.

For FNFTpy to work, a copy of FNFT has to be installed. For general information, source files and installation of FNFT, visit FNFT's github page: <https://github.com/FastNFT>

For information about setup and usage of FNFTpy see README.md or documentation.

FNFTpy is free software; you can redistribute it and/or modify it under the terms of the version 2 of the GNU General Public License as published by the Free Software Foundation.

FNFTpy is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Contributors:

Christoph Mahnke, 2018

KORTEWEG-DE-FRIES EQUATION

2.1 kdvv - calculate the Nonlinear Fourier Transform

`FNFTpy.fnft_kdvv_wrapper.kdvv(u, tvec, M=128, Xi1=-2, Xi2=2, dis=None)`

Calculate the Nonlinear Fourier Transform for the Korteweg-de Vries equation with vanishing boundaries.

This function is intended to be 'convenient', which means it automatically calculates some variables needed to call the C-library and uses some default options. Own options can be set by passing optional arguments (see below).

It converts all Python input into the C equivalent and returns the result from FNFT. If a more C-like interface is desired, the function 'kdvv_wrapper' can be used (see documentation there).

Arguments:

`u` : numpy array holding the samples of the field to be analyzed

`tvec` : time vector

`M` : number of samples for the continuous spectrum to calculate,

Optional arguments:

`Xi1, Xi2` [min and max frequency for the continuous spectrum] default = -2, 2

`dis` : determines the discretization, default = 15

0 = 2split1a 1 = 2split1b 2 = 2split2a 3 = 2split2b 4 = 2split3a 5 = 2split3b 6 = 2split4a
7 = 2split4b 8 = 2split5a 9 = 2split5b 10 = 2split6a 11 = 2split6b 12 = 2split7a 13 =
2split7b 14 = 2split8a 15 = 2split8b

Returns:

`rdict` : dictionary holding the fields (depending on options)

`return_value` : return value from FNFT

`cont` : continuous spectrum

2.2 kdvv_wrapper - interact with FNFT library

`FNFTpy.fnft_kdvv_wrapper.kdvv_wrapper(D, u, T1, T2, M, Xi1, Xi2, K, options)`

Calculate the Nonlinear Fourier Transform for the Korteweg-de Vries equation with vanishing boundaries.

This function's interface mimics the behavior of the function 'fnft_kdvv' of FNFT. It converts all Python input into the C equivalent and returns the result from FNFT. If a more simplified version is desired, 'kdvv' can be used (see documentation there).

Arguments:

D : number of samples

u : numpy array holding the samples of the field to be analyzed

T1, T2 : time positions of the first and the last sample

M : number of values for the continuous spectrum to calculate

Xi1, Xi2 : min and max frequency for the continuous spectrum

K : maximum number of bound states to calculate (no effect yet)

options : options for kdvv as KdvvOptionsStruct. Can be generated e.g. with 'get_kdvv_options()'

Returns:

rdict : dictionary holding the fields (depending on options)

return_value : return value from FNFT

cont : continuous spectrum

2.3 get, set and view options for kdvv_wrapper

`FNFTpy.options_handling.fnft_kdvv_default_opts_wrapper()`

Get the default options for kdvv directly from the FNFT C-library.

Returns:

options : KdvvOptionsStruct with options for kdvv_wrapper

`FNFTpy.options_handling.print_kdvv_options(opts=None)`

Print options of a KdvvOptionsStruct for kdvv.

When called without additional argument, the default options from FNFT are printed.

Optional arguments:

opts : KdvvOptionsStruct, e.g. created by get_kdvv_options()

`FNFTpy.options_handling.get_kdvv_options(dis=None)`

Get an KdvvOptionsStruct struct for use with kdvv_wrapper.

When called without additional optional arguments, the default values from FNFT are used.

Optional arguments:

dis: discretization, default = 15

0 = 2split1a 1 = 2split1b 2 = 2split2a 3 = 2split2b 4 = 2split3a 5 = 2split3b 6 = 2split4a
7 = 2split4b 8 = 2split5a 9 = 2split5b 10 = 2split6a 11 = 2split6b 12 = 2split7a 13 =
2split7b 14 = 2split8a 15 = 2split8b

Returns:

options : KdvvOptionsStruct

2.4 options structure

`class FNFTpy.typesdef.KdvvOptionsStruct`

Ctypes options struct for interfacing fnft_kdvv

NONLINEAR SCHROEDINGER EQUATION WITH PERIODIC BOUNDARIES

3.1 nsep - calculate the Nonlinear Fourier Transform

`FNFTpy.fnft_nsep_wrapper.nsep(q, T1, T2, kappa=1, loc=None, filt=None, bb=None, maxev=None, dis=None, nf=None)`

Calculate the Nonlinear Fourier Transform for the Nonlinear Schroedinger equation with periodic boundaries.

This function is intended to be 'convenient', which means it automatically calculates some variables needed to call the C-library and uses some default options. Own options can be set by passing optional arguments (see below). Options can be set by passing optional arguments (see below).

It converts all Python input into the C equivalent and returns the result from FNFT. If a more C-like interface is desired, the function 'nsep_wrapper' can be used (see documentation there).

Arguments:

`q` : numpy array holding the samples of the field to be analyzed

`T1, T2` : time positions of the first and the (d+1) sample

Optional arguments:

`kappa` : +/- 1 for focussing/defocussing nonlinearity, default = 1

`loc` : localization method for the spectrum, default = 2

0=Subsample and refine 1=Gridsearch 2=Mixed

`filt` : filtering of spectrum, default = 2

0=None 1=Manual 2=Auto

`bb`: bounding box used for manual filtering, default = [-inf, inf, -inf, inf]

`maxev` : maximum number of evaluations for root refinement, default = 20

`nf` : normalization Flag default = 1

0=off 1=on

`dis` : discretization, default = 2

0=2split2modal 1=2split2a 2=2split4a 3=2split4b 4=BO

Returns:

`rdict` : dictionary holding the fields (depending on options)

`return_value` : return value from FNFT

`K` : number of points in the main spectrum

`main` : main spectrum

m: number of points in the auxiliary spectrum

aux: auxiliary spectrum

3.2 nsep_wrapper - interact with FNFT library

`FNFTpy.fnft_nsep_wrapper.nsep_wrapper(D, q, T1, T2, kappa, options)`

Calculate the Nonlinear Fourier Transform for the Nonlinear Schroedinger equation with periodic boundaries.

This function's interface mimics the behavior of the function 'fnft_nsep' of FNFT. It converts all Python input into the C equivalent and returns the result from FNFT. If a more simplified version is desired, 'nsep' can be used (see documentation there).

Arguments:

D : number of sample points

q : numpy array holding the samples of the field to be analyzed

T1, T2 : time positions of the first and the (D+1) sample

kappa : +/- 1 for focussing/defocussing nonlinearity

options : options for nsep as NsepOptionsStruct. Can be generated e.g. with 'get_nsep_options()'

Returns:

rdict : dictionary holding the fields (depending on options)

return_value : return value from FNFT

K : number of points in the main spectrum

main : main spectrum

M: number of points in the auxiliary spectrum

aux: auxiliary spectrum

3.3 get, set and view options for nsep_wrapper

`FNFTpy.options_handling.fnft_nsep_default_opts_wrapper()`

Get the default options for nsep directly from the FNFT C-library.

Returns:

options : NsepOptionsStruct for nsep_wrapper

`FNFTpy.options_handling.print_nsep_options(opts=None)`

Print options of a NsepOptionsStruct for nsep.

When called without additional arguments, the default options from FNFT are printed.

Optional arguments:

opts : NsepOptionsStruct, e.g. created by get_nsep_options

`FNFTpy.options_handling.get_nsep_options(loc=None, filt=None, bb=None, maxev=None, dis=None, nf=None)`

Get a NsepOptionsStruct struct for use with nsep_wrapper.

When called without additional optional argument, the default values from FNFT are used.

Optional arguments:

loc : localization of spectrum, default = 2

0=Subsample and Refine 1=Gridsearch 2=Mixed

filt : filtering of spectrum, default = 2

0=None 1=Manual 2=Auto

bb : bounding box used for manual filtering, default = [-inf, inf, -inf, inf]

maxev : maximum number of evaluations for root refinement, default = 20

nf : normalization flag, default = 1

0=off 1=on

dis : discretization, default = 1

0=2split2modal 1=2split2a 2=2split4a 3=2split4b 4=BO

Returns:

options : NsepOptionsStruct with options for nsep_wrapper

3.4 options structure

```
class FNFTpy.typesdef.NsepOptionsStruct
    Ctypes options struct for interfacing fnft_nsep
```


NONLINEAR SCHROEDINGER EQUATION WITH VANISHING BOUNDARIES

4.1 nsev - calculate the Nonlinear Fourier Transform

```
FNFTpy.fnft_nsev_wrapper.nsev(q, tvec, Xi1=-2, Xi2=2, M=128, K=128, kappa=1, bsf=None,  
                               bsl=None, niter=None, dst=None, cst=None, nf=None,  
                               dis=None)
```

Calculate the Nonlinear Fourier Transform for the Nonlinear Schroedinger equation with vanishing boundaries.

This function is intended to be 'convenient', which means it automatically calculates some variables needed to call the C-library and uses some default options. Own options can be set by passing optional arguments (see below). Options can be set by passing optional arguments (see below).

It converts all Python input into the C equivalent and returns the result from FNFT. If a more C-like interface is desired, the function 'nsev_wrapper' can be used (see documentation there).

Arguments:

q : numpy array holding the samples of the field to be analyzed

tvec: time vector for q samples

Optional arguments:

Xi1, Xi2 : min and max frequency for the continuous spectrum. default = -2,2

M : number of values for the continuous spectrum to calculate default = 128

K : maximum number of bound states to calculate default = 128

kappa : +/- 1 for focussing/defocussing nonlinearity, default = 1

bsf : bound state filtering, default =2

0=none 1=basic 2=full

bsl : bound state localization, default = 0

0=Fast Eigenvalue 1=Newton 2=Subsample and Refine

niter : number of iterations for Newton bound state localization, default = 10

dst : type of discrete spectrum, default = 2

0=norming constants 1=residues 2=both

cst : type of continuous spectrum, default = 0

0=reflection coefficient 1=a and b 2=both

dis : discretization, default = 3

0=2split2modal 1=2split2a 2=2split4a 3=2split4b 4=BO

nf : normalization flag, default = 1

0=off 1=on

Returns:

rdict : dictionary holding the fields (depending on options)
 return_value : return value from FNFT
 bound_states_num : number of bound states found
 bound_states : array of bound states found
 disc_norm : discrete spectrum - norming constants
 disc_res : discrete spectrum - residues
 cont_ref : continuous spectrum - reflection coefficient
 cont_a : continuous spectrum - scattering coefficient a
 cont_b : continuous spectrum - scattering coefficient b

4.2 nsev_wrapper - interact with FNFT library

`FNFTpy.fnft_nsev_wrapper.nsev_wrapper` (*D, q, T1, T2, Xi1, Xi2, M, K, kappa, options*)

Calculate the Nonlinear Fourier Transform for the Nonlinear Schroedinger equation with vanishing boundaries.

This function's interface mimics the behavior of the function 'fnft_nsev' of FNFT. It converts all Python input into the C equivalent and returns the result from FNFT. If a more simplified version is desired, 'nsev' can be used (see documentation there).

Arguments:

D : number of sample points
 q : numpy array holding the samples of the field to be analyzed
 T1, T2 : time positions of the first and the last sample
 Xi1, Xi2 : min and max frequency for the continuous spectrum
 M : number of values for the continuous spectrum to calculate
 K : maximum number of bound states to calculate
 kappa : +/- 1 for focussing/defocussing nonlinearity
 options : options for nsev as NsevOptionsStruct

Returns:

rdict : dictionary holding the fields (depending on options)
 return_value : return value from FNFT
 bound_states_num : number of bound states found
 bound_states : array of bound states found
 disc_norm : discrete spectrum - norming constants
 disc_res : discrete spectrum - residues
 cont_ref : continuous spectrum - reflection coefficient
 cont_a : continuous spectrum - scattering coefficient a
 cont_b : continuous spectrum - scattering coefficient b

4.3 get, set and view options for nsev_wrapper

`FNFTpy.options_handling.fnft_nsev_default_opts_wrapper()`

Get the default options for nsev directly from the FNFT C-library.

Returns:

options : NsevOptionsStruct with options for nsev_wrapper

`FNFTpy.options_handling.print_nsev_options(opts=None)`

Print options of a NsevOptionsStruct for nsev.

When called without additional argument, the default options from FNFT are printed.

Optional arguments:

opts : NsevOptionsStruct, e.g. created by `get_nsev_options()`

`FNFTpy.options_handling.get_nsev_options(bsf=None, bsl=None, niter=None, dst=None, cst=None, nf=None, dis=None)`

Get a NsevOptionsStruct for use with nsev_wrapper.

When called without additional optional arguments, the default values from FNFT are used.

Optional arguments:

bsf : bound state filtering, default = 2

0=none 1=basic 2=full

bsl : bound state localization, default = 2

0=Fast Eigenvalue 1=Newton 2=Subsample and refine

niter : number of iterations for Newton bound state location, default = 10

dst : type of discrete spectrum, default = 0

0=norming constants 1=residues 2=both

cst : type of continuous spectrum, default = 0

0=reflection coefficient 1=a and b 2=both

dis : discretization, default = 3

0=2split2modal 1=2split2a 2=2split4a 3=2split4b 4=BO

nf : normalization flag, default = 1

0=off 1=on

Returns:

options : NsevOptionsStruct with options for nsev

4.4 options structure

`class FNFTpy.typesdef.NsevOptionsStruct`

Ctypes options struct for interfacing fnft_kdvv

EXAMPLE FUNCTIONS

`FNFTpy.tests.print_default_options()`
Print the default options for `kdvv`, `nsep` and `nsev`.

`FNFTpy.tests.kdvvexample()`
Mimics the C example for calling `fnft_kdvv`.

`FNFTpy.tests.nsepexample()`
Mimics the C example for calling `fnft_nsep`.

`FNFTpy.tests.nsevexample()`
Mimics the C example for calling `fnft_nsev`.

PYTHON MODULE INDEX

f

`FNFTpy`, [1](#)

F

fnft_kdvv_default_opts_wrapper() (in module FN-
FTpy.options_handling), 4
fnft_nsep_default_opts_wrapper() (in module FN-
FTpy.options_handling), 6
fnft_nsev_default_opts_wrapper() (in module FN-
FTpy.options_handling), 11
FNFTpy (module), 1

G

get_kdvv_options() (in module FN-
FTpy.options_handling), 4
get_nsep_options() (in module FN-
FTpy.options_handling), 6
get_nsev_options() (in module FN-
FTpy.options_handling), 11

K

kdvv() (in module FNFTpy.fnft_kdvv_wrapper), 3
kdvv_wrapper() (in module FN-
FTpy.fnft_kdvv_wrapper), 3
kdvvexample() (in module FNFTpy.tests), 13
KdvvOptionsStruct (class in FNFTpy.typesdef), 4

N

nsep() (in module FNFTpy.fnft_nsep_wrapper), 5
nsep_wrapper() (in module FN-
FTpy.fnft_nsep_wrapper), 6
nsepexample() (in module FNFTpy.tests), 13
NsepOptionsStruct (class in FNFTpy.typesdef), 7
nsev() (in module FNFTpy.fnft_nsev_wrapper), 9
nsev_wrapper() (in module FN-
FTpy.fnft_nsev_wrapper), 10
nsevexample() (in module FNFTpy.tests), 13
NsevOptionsStruct (class in FNFTpy.typesdef), 11

P

print_default_options() (in module FNFTpy.tests), 13
print_kdvv_options() (in module FN-
FTpy.options_handling), 4
print_nsep_options() (in module FN-
FTpy.options_handling), 6
print_nsev_options() (in module FN-
FTpy.options_handling), 11