
FNFTpy Documentation

Release 0.4.1

Christoph Mahnke

Sep 24, 2021

CONTENTS:

1	Module overview	1
2	Auxiliary functions	3
2.1	set destination of FNFT library	3
2.2	get winmode parameter (Windows only)	3
2.3	get and print FNFT version	3
3	Korteweg-de-Vries equation	5
3.1	kdvv - calculate the Nonlinear Fourier Transform	5
3.2	kdvv_wrapper - interact with FNFT library	8
3.3	get, set and print options for kdvv_wrapper	8
3.4	options KdvvOptionsStruct	11
4	Manakov equation with vanishing boundaries	13
4.1	manakovv - calculate the Nonlinear Fourier Transform	13
4.2	manakovv_wrapper - interact with FNFT library	15
4.3	get, set and print options for manakovv wrapper	16
4.4	options ManakovvOptionsStruct	16
5	Nonlinear Schroedinger equation with periodic boundaries	17
5.1	nsep - calculate the Nonlinear Fourier Transform	17
5.2	nsep_wrapper - interact with FNFT library	19
5.3	get, set and print options for nsep wrapper	19
5.4	options NsepOptionsStruct	21
6	Nonlinear Schroedinger equation with vanishing boundaries	23
6.1	nsev - calculate the Nonlinear Fourier Transform	23
6.2	nsev_wrapper - interact with FNFT library	25
6.3	get, set and print options for nsep wrapper	26
6.4	options NsevOptionsStruct	28
7	Nonlinear Schroedinger equation with vanishing boundaries - Inverse Nonlinear Fourier Transform	29
7.1	nsev_inverse_xi_wrapper	29
7.2	nsev_inverse - calculate the Inverse Nonlinear Fourier Transform	30
7.3	nsev_inverse_wrapper - interact with FNFT library	32
7.4	get, set and print options for nsev_inverse_wrapper	32
7.5	options NsevInverseOptionsStruct	34
	Python Module Index	35
	Index	37

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 - 2021

AUXILIARY FUNCTIONS

2.1 set destination of FNFT library

`FNFTpy.get_lib_path()`

Return the path of the FNFT file.

Edit this function to set the location of the compiled library for FNFT. See example strings below.

Returns:

- `libstring` : string holding library path

Example paths:

- `libstr = "C:/Libraries/local/libfnft.dll"` # example for windows
- `libstr = "/usr/local/lib/libfnft.so"` # example for linux

2.2 get winmode parameter (Windows only)

`FNFTpy.get_winmode_param()`

This function is used to allow module-wide change of the winmode parameter used by `ctypes.CDLL` function.

Background: since Python 3.8 there is a change to the `CDLL` function. Due to some hazzle in the Windows library locations, it may be difficult to address libraries by full path. The standard parameter is 'None'. Some users report import does work when it is set to 0.

Returns: winmode parameter : 0 (change manually to either None or some int if you experience problems)

2.3 get and print FNFT version

`FNFTpy.get_fnft_version()`

Get the version of FNFT used by calling `fnft_version`.

Returns:

- **rdict: dictionary holding the fields:**
 - `return_value` : return value from FNFT
 - `major` : major version number
 - `minor` : minor version number
 - `patch` : patch level
 - `suffix` : suffix string

`FNFTpy.print_fnft_version()`

Prints the path and the version of FNFT library used.

KORTEWEG-DE-VRIES EQUATION

3.1 kdvv - calculate the Nonlinear Fourier Transform

```
FNFTpy.fnft_kdvv_wrapper.kdvv(u, tvec, K=128, M=128, Xi1=-2, Xi2=2, dis=None, bsl=None,  
                               niter=None, dst=None, cst=None, nf=None, ref=None,  
                               bound_state_guesses=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

Optional arguments:

- `K` : number of bound states expected, default = 128
- `M` : number of samples for the continuous spectrum to calculate, default = 128
- `Xi1, Xi2` : min and max frequency for the continuous spectrum, default = [-2,2]
- `dis` : discretization, default = 39 (for details see FNFT documentation)
 - 0 = 2SPLIT2_MODAL_VANILLA
 - 1 = BO_VANILLA
 - 2 = 2SPLIT1A_VANILLA
 - 3 = 2SPLIT1B_VANILLA
 - 4 = 2SPLIT2A_VANILLA
 - 5 = 2SPLIT2B_VANILLA
 - 6 = 2SPLIT2S_VANILLA
 - 7 = 2SPLIT3A_VANILLA
 - 8 = 2SPLIT3B_VANILLA
 - 9 = 2SPLIT3S_VANILLA
 - 10 = 2SPLIT4A_VANILLA
 - 11 = 2SPLIT4B_VANILLA
 - 12 = 2SPLIT5A_VANILLA

- 13 = 2SPLIT5B_VANILLA
- 14 = 2SPLIT6A_VANILLA
- 15 = 2SPLIT6B_VANILLA
- 16 = 2SPLIT7A_VANILLA
- 17 = 2SPLIT7B_VANILLA
- 18 = 2SPLIT8A_VANILLA
- 19 = 2SPLIT8B_VANILLA
- 20 = 4SPLIT4A_VANILLA
- 21 = 4SPLIT4B_VANILLA
- 22 = CF4_2_VANILLA
- 23 = CF4_3_VANILLA
- 24 = CF5_3_VANILLA
- 25 = CF6_4_VANILLA
- 26 = ES4_VANILLA
- 27 = TES4_VANILLA
- 28 = 2SPLIT2_MODAL
- 29 = BO
- 30 = 2SPLIT1A
- 31 = 2SPLIT1B
- 32 = 2SPLIT2A
- 33 = 2SPLIT2B
- 34 = 2SPLIT2S
- 35 = 2SPLIT3A
- 36 = 2SPLIT3B
- 37 = 2SPLIT3S
- 38 = 2SPLIT4A
- 39 = 2SPLIT4B
- 40 = 2SPLIT5A
- 41 = 2SPLIT5B
- 42 = 2SPLIT6A
- 43 = 2SPLIT6B
- 44 = 2SPLIT7A
- 45 = 2SPLIT7B
- 46 = 2SPLIT8A
- 47 = 2SPLIT8B
- 48 = 4SPLIT4A
- 49 = 4SPLIT4B
- 50 = CF4_2
- 51 = CF4_3

- 52 = CF5_3
 - 53 = CF6_4
 - 54 = ES4
 - 55 = TES4
- bsl : bound state localization, default=1
 - 0 = NEWTON
 - 1 = GRIDSEARCH_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
 - 3 = skip computing discrete spectrum
- cst : type of continuous spectrum, default = 0
 - 0 = REFLECTION_COEFFICIENT
 - 1 = AB
 - 2 = BOTH
 - 3 = skip computing continuous spectrum
- nf : normalization flag, default = 1
 - 0 = off
 - 1 = on
- ref : richardson extrapolation flag, default = 0
 - 0 = off
 - 1 = on
- bound_state_guesses : initial guesses for bound states, only effective if bsl=0, default=None

Please note: up to now, the guesses have to be purely imaginary

Returns:

- rdict : dictionary holding the fields:
 - return_value : return value from FNFT
 - cont_ref : continuous spectrum (reflection)
 - 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
 - options : KdvvOptionsStruct with options used

3.2 kdvv_wrapper - interact with FNFT library

`FNFTpy.fnft_kdvv_wrapper.kdvv_wrapper(D, u, T1, T2, K, M, Xi1, Xi2, options, bound_state_guesses=None)`

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
- `K` : maximum number of bound states to calculate
- `M` : number of values for the continuous spectrum to calculate
- `Xi1, Xi2` : min and max frequency for the continuous spectrum
- `options`: options for kdvv as `KdvvOptionsStruct`. Can be generated e.g. with 'get_kdvv_options()'

Optional Arguments:

- **bound_state_guesses** [lost or array of bound state guesses, only effective if `bsl==1` (Newton) bound state localization) is activated. Default = None

Please note: up to now, the guesses have to be purely imaginary

Returns:

- `rdict` : dictionary holding the fields:
 - `return_value` : return value from FNFT
 - `cont_ref` : continuous spectrum (reflection)
 - `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
 - `options` : `KdvvOptionsStruct` with options used

3.3 get, set and print options for kdvv_wrapper

`FNFTpy.options_handling.fnft_kdvv_default_options_wrapper()`

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

Returns:

- `options` : `KdvvOptionsStruct` with options for kdvv_wrapper

`FNFTpy.options_handling.get_kdvv_options`(*dis=None, bsl=None, niter=None, dst=None, cst=None, nf=None, ref=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 = 39 (for details see FNFT documentation)
 - 0 = 2SPLIT2_MODAL_VANILLA
 - 1 = BO_VANILLA
 - 2 = 2SPLIT1A_VANILLA
 - 3 = 2SPLIT1B_VANILLA
 - 4 = 2SPLIT2A_VANILLA
 - 5 = 2SPLIT2B_VANILLA
 - 6 = 2SPLIT2S_VANILLA
 - 7 = 2SPLIT3A_VANILLA
 - 8 = 2SPLIT3B_VANILLA
 - 9 = 2SPLIT3S_VANILLA
 - 10 = 2SPLIT4A_VANILLA
 - 11 = 2SPLIT4B_VANILLA
 - 12 = 2SPLIT5A_VANILLA
 - 13 = 2SPLIT5B_VANILLA
 - 14 = 2SPLIT6A_VANILLA
 - 15 = 2SPLIT6B_VANILLA
 - 16 = 2SPLIT7A_VANILLA
 - 17 = 2SPLIT7B_VANILLA
 - 18 = 2SPLIT8A_VANILLA
 - 19 = 2SPLIT8B_VANILLA
 - 20 = 4SPLIT4A_VANILLA
 - 21 = 4SPLIT4B_VANILLA
 - 22 = CF4_2_VANILLA
 - 23 = CF4_3_VANILLA
 - 24 = CF5_3_VANILLA
 - 25 = CF6_4_VANILLA
 - 26 = ES4_VANILLA
 - 27 = TES4_VANILLA
 - 28 = 2SPLIT2_MODAL
 - 29 = BO
 - 30 = 2SPLIT1A
 - 31 = 2SPLIT1B
 - 32 = 2SPLIT2A
 - 33 = 2SPLIT2B

- 34 = 2SPLIT2S
- 35 = 2SPLIT3A
- 36 = 2SPLIT3B
- 37 = 2SPLIT3S
- 38 = 2SPLIT4A
- 39 = 2SPLIT4B
- 40 = 2SPLIT5A
- 41 = 2SPLIT5B
- 42 = 2SPLIT6A
- 43 = 2SPLIT6B
- 44 = 2SPLIT7A
- 45 = 2SPLIT7B
- 46 = 2SPLIT8A
- 47 = 2SPLIT8B
- 48 = 4SPLIT4A
- 49 = 4SPLIT4B
- 50 = CF4_2
- 51 = CF4_3
- 52 = CF5_3
- 53 = CF6_4
- 54 = ES4
- 55 = TES4
- bsl : bound state localization, default=1
 - 0 = NEWTON
 - 1 = GRIDSEARCH_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
 - 3 = skip computing discrete spectrum
- cst : type of continuous spectrum, default = 0
 - 0 = REFLECTION_COEFFICIENT
 - 1 = AB
 - 2 = BOTH
 - 3 = skip computing continuous spectrum
- nf : normalization flag, default = 1
 - 0 = off
 - 1 = on

- `ref` : richardson extrapolation flag, default = 0
 - 0 = off
 - 1 = on

Returns:

- `options` : `KdvvOptionsStruct`

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

Print options of a `KdvvOptionsStruct`.

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

Optional arguments:

- `options` : `KdvvOptionsStruct`, e.g. created by `get_kdvv_options()`

3.4 options `KdvvOptionsStruct`

class `FNFTpy.typesdef.KdvvOptionsStruct`

Ctypes options struct for interfacing `fnft_kdvv`.

Fields:

- `bound_state_localization`
- `niter`
- `discspec_type`
- `contspec_type`
- `normalization_flag`
- `discretization`
- `richardson_extrapolation_flag`

Options can be printed directly to screen, e.g.

```
print(get_kdvv_options())
```

String representation can be generated by

```
repr(get_kdvv_options())
```


MANAKOV EQUATION WITH VANISHING BOUNDARIES

4.1 manakovv - calculate the Nonlinear Fourier Transform

`FNFTpy.fnft_manakovv_wrapper.manakovv(q1, q2, tvec, Xi1=-1.75, Xi2=2, M=128, K=128, kappa=1, bsf=None, bsl=None, niter=None, Dsub=None, dst=None, cst=None, nf=None, dis=None, ref=None)`

Calculate the Nonlinear Fourier Transform for the Manakov equation with vanishing boundary conditions.

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 'mannakovv_wrapper' can be used (see documentation there).

Arguments:

- `q1` : numpy array holding the samples of the first input field (potential function)
- `q2` : numpy array holding the samples of the second input field (potential function)
- `tvec` : time vector

Optional Arguments:

- `Xi1, Xi2` : min and max frequency for the continuous spectrum. default = -1.75,2
- `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 = 2
 - 0 = FAST_EIGENVALUE
 - 1 = NEWTON
 - 2 = SUBSAMPLE_AND_REFINE
- `niter` : number of iterations for Newton bound state location, default = 10
- `Dsub` : number of samples used for 'subsampling and refine'-method, default = 0 (auto)
- `dst` : type of discrete spectrum, default = 0
 - 0 = NORMING_CONSTANTS

- 1 = RESIDUES
- 2 = BOTH
- 3 = skip computing discrete spectrum
- `cst` : type of continuous spectrum, default = 0
 - 0 = REFLECTION_COEFFICIENT
 - 1 = AB
 - 2 = BOTH
 - 3 = skip computing continuous spectrum
- `dis` : discretization, default = 3
 - 0 = 2SPLIT3A
 - 1 = 2SPLIT3B
 - 2 = 2SPLIT4A
 - 3 = 2SPLIT4B
 - 4 = 2SPLIT6B
 - 5 = 4SPLIT4A
 - 6 = 4SPLIT4B
 - 7 = 4SPLIT6B
 - 8 = FTES4_4A
 - 9 = FTES4_4B
 - 10 = FTES4_suzuki
 - 11 = CF4_2
 - 12 = BO
- `nf` : normalization flag, default = 1
 - 0 = off
 - 1 = on
- `ref` : richardson extrapolation flag, default = 0
 - 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_ref1` : continuous spectrum - reflection coefficients 1
 - `cont_ref2` : continuous spectrum - reflection coefficients 2
 - `cont_a` : continuous spectrum - scattering coefficient a
 - `cont_b1` : continuous spectrum - scattering coefficient b 2

- `cont_b2` : continuous spectrum - scattering coefficient b 1
- `options` : `NsevOptionsStruct` with the options used

4.2 manakovv_wrapper - interact with FNFT library

`FNFTpy.fnft_manakovv_wrapper.manakovv_wrapper(D, q1, q2, T1, T2, Xi1, Xi2, M, K, kappa, options)`

Calculate the Nonlinear Fourier Transform for the Manakov equation with vanishing boundary conditions.

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, 'manakovv' can be used (see documentation there).

Arguments:

- `D` : number of sample points
- `q1` : numpy array holding the samples of the first input field (potential function)
- `q2` : numpy array holding the samples of the second input field (potential function)
- `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 manakovv as `ManakovvOptionsStruct`

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_ref1` : continuous spectrum - reflection coefficients 1
 - `cont_ref2` : continuous spectrum - reflection coefficients 2
 - `cont_a` : continuous spectrum - scattering coefficient a
 - `cont_b1` : continuous spectrum - scattering coefficient b 2
 - `cont_b2` : continuous spectrum - scattering coefficient b 1
 - `options` : `NsevOptionsStruct` with the options used

4.3 get, set and print options for manakovv wrapper

FNFTpy.options_handling.**fnft_manakovv_default_options_wrapper()**
WRITE DOCSTRING

Returns

FNFTpy.options_handling.**get_manakovv_options**(*dis=None, bsf=None, bsl=None, Dsub=None, niter=None, dst=None, cst=None, nf=None, ref=None*)

TODO write docstring :param dis: :param bsf: :param bsl: :param Dsub: :param niter: :param dst: :param cst: :param nf: :param ref: :return:

FNFTpy.options_handling.**print_manakovv_options**(*options=None*)

Print options of a Manakovv.

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

Optional arguments:

- options : ManakovvOptionsStruct, e.g. created by get_manakovv_options()

4.4 options ManakovvOptionsStruct

class FNFTpy.typesdef.**ManakovvOptionsStruct**
Ctypes options struct for interfacing fnft_manakovv.

Fields:

- bound_state_filtering
- bound_state_localization
- niter
- Dsub
- discspec_type
- contspec_type
- normalization_flag
- discretization
- richardson_extrapolation_flag
- discretization

Options can be printed directly to screen, e.g.

```
print(get_manakovv_options())
```

String representation can be generated by

```
repr(get_manakovv_options())
```

NONLINEAR SCHROEDINGER EQUATION WITH PERIODIC BOUNDARIES

5.1 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, floq_range=None, ppspine=None, dsub=None,
                               tol=None, phase_shift=0.0)
```

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).

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 input field
- T1, T2 : time positions of the first and the (D+1) sample, where D is the number of samples

Optional arguments:

- kappa : +/- 1 for focussing/defocussing nonlinearity, default = 1
- loc : localization method for the spectrum, default = 2
 - SUBSAMPLE_AND_REFINE = 0
 - GRIDSEARCH = 1
 - MIXED = 2
- filt : filtering of spectrum, default = 2
 - NONE = 0
 - MANUAL = 1
 - AUTO = 2
- bb: bounding box used for manual filtering, default = [-inf, inf, -inf, inf]
- maxev : maximum number of evaluations for root refinement, default = 20
- dis : discretization, default = 4
 - 0 = 2SPLIT2_MODAL
 - 1 = BO
 - 2 = 2SPLIT1A
 - 4 = 2SPLIT2A

- 5 = 2SPLIT2B
- 6 = 2SPLIT2S
- 7 = 2SPLIT3A
- 8 = 2SPLIT3B
- 9 = 2SPLIT3S
- 10 = 2SPLIT4A
- 11 = 2SPLIT4B
- 12 = 2SPLIT5A
- 13 = 2SPLIT5B
- 14 = 2SPLIT6A
- 15 = 2SPLIT6B
- 16 = 2SPLIT7A
- 17 = 2SPLIT7B
- 18 = 2SPLIT8A
- 19 = 2SPLIT8B
- 20 = 4SPLIT4A
- 21 = 4SPLIT4B
- 22 = CF4_2
- 23 = CF4_3
- 24 = CF5_3
- 25 = CF6_4
- 26 = ES4
- 27 = TES4
- nf : normalization flag, default=1
 - 0 = off
 - 1 = on
- floq_range : array of two reals defining Floquet range, default = [-1, 1]
- ppspine : points per spine: defining the grid between interval set by floq_range
- dsub : approximate number of samples for ‘subsample and refine’ localization
- tol : Tolerance, for root search refinement. Can be either positive number or (default=-1 (=auto))
- phase_shift : change of the phase over one quasi-period, $\arg(q(t+(T_2-T_1))/q(t))$ (default=0)

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
 - options : NsepOptionsStruct with options used

5.2 nsep_wrapper - interact with FNFT library

`FNFTpy.fnft_nsep_wrapper.nsep_wrapper(D, q, T1, T2, phase_shift, 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 input field
- `T1, T2` : time positions of the first and the $(D+1)$ sample
- `phase_shift` : change of the phase over one quasi-period, $\arg(q(t+(T2-T1)/q(t))$
- `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
 - `options` : `NsepOptionsStruct` with options used

5.3 get, set and print options for nsep wrapper

`FNFTpy.options_handling.fnft_nsep_default_options_wrapper()`

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

Returns:

- `options` : `NsepOptionsStruct` for nsep_wrapper

`FNFTpy.options_handling.get_nsep_options(loc=None, filt=None, bb=None, maxev=None, dis=None, nf=None, floq_range=None, ppspine=None, dsub=None, tol=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 method for the spectrum, default = 2
 - `SUBSAMPLE_AND_REFINE` = 0
 - `GRIDSEARCH` = 1
 - `MIXED` = 2
- `filt` : filtering of spectrum, default = 2
 - `NONE` = 0

- MANUAL = 1
- AUTO = 2
- bb: bounding box used for manual filtering, default = [-inf, inf, -inf, inf]
- maxev : maximum number of evaluations for root refinement, default = 20
- dis : discretization, default = 4
 - 0 = 2SPLIT2_MODAL
 - 1 = BO
 - 2 = 2SPLIT1A
 - 4 = 2SPLIT2A
 - 5 = 2SPLIT2B
 - 6 = 2SPLIT2S
 - 7 = 2SPLIT3A
 - 8 = 2SPLIT3B
 - 9 = 2SPLIT3S
 - 10 = 2SPLIT4A
 - 11 = 2SPLIT4B
 - 12 = 2SPLIT5A
 - 13 = 2SPLIT5B
 - 14 = 2SPLIT6A
 - 15 = 2SPLIT6B
 - 16 = 2SPLIT7A
 - 17 = 2SPLIT7B
 - 18 = 2SPLIT8A
 - 19 = 2SPLIT8B
 - 20 = 4SPLIT4A
 - 21 = 4SPLIT4B
 - 22 = CF4_2
 - 23 = CF4_3
 - 24 = CF5_3
 - 25 = CF6_4
 - 26 = ES4
 - 27 = TES4
- nf : normalization flag, default=1
 - 0 = off
 - 1 = on
- floq_range : array of two reals defining Floquet range, default = [-1, 1]
- ppspine : points per spine: defining the grid between interval set by floq_range
- dsub : approximate number of samples for ‘subsample and refine’ localization
- tol : Tolerance, for root search refinement. Can be either positive number or (default =-1 (=auto))

Returns:

- options : NsepOptionsStruct

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

Print options of a NsepOptionsStruct.

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

Optional arguments:

- options : NsepOptionsStruct, e.g. created by `get_nsep_options`

5.4 options NsepOptionsStruct

class `FNFTpy.typesdef.NsepOptionsStruct`

Ctypes options struct for interfacing `fnft_nsep`.

Fields:

- localization
- filtering
- bounding_box
- max_evals
- discretization
- normalization_flag
- floquet_range
- points_per_spine
- dsub
- tol

Options can be printed directly to screen, e.g.

```
print(get_nsep_options())
```

String representation can be generated by

```
repr(get_nsep_options())
```


NONLINEAR SCHROEDINGER EQUATION WITH VANISHING BOUNDARIES

6.1 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, Dsub=None, dst=None, cst=None, nf=None, dis=None,
                               ref=None, bound_state_guesses=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).

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 input field
- *tvec*: time vector

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 = 2
 - 0 = FAST_EIGENVALUE
 - 1 = NEWTON
 - 2 = SUBSAMPLE_AND_REFINE
- *niter* : number of iterations for Newton bound state location, default = 10
- *Dsub* : number of samples used for ‘subsampling and refine’-method, default = 0 (auto)
- *dst* : type of discrete spectrum, default = 0
 - 0 = NORMING_CONSTANTS

- 1 = RESIDUES
- 2 = BOTH
- 3 = skip computing discrete spectrum
- `csf` : type of continuous spectrum, default = 0
 - 0 = REFLECTION_COEFFICIENT
 - 1 = AB
 - 2 = BOTH
 - 3 = skip computing continuous spectrum
- `dis` : discretization, default = 11
 - 0 = 2SPLIT2_MODAL
 - 1 = BO
 - 2 = 2SPLIT1A
 - 3 = 2SPLIT1B
 - 4 = 2SPLIT2A
 - 5 = 2SPLIT2B
 - 6 = 2SPLIT2S
 - 7 = 2SPLIT3A
 - 8 = 2SPLIT3B
 - 9 = 2SPLIT3S
 - 10 = 2SPLIT4A
 - 11 = 2SPLIT4B
 - 12 = 2SPLIT5A
 - 13 = 2SPLIT5B
 - 14 = 2SPLIT6A
 - 15 = 2SPLIT6B
 - 16 = 2SPLIT7A
 - 17 = 2SPLIT7B
 - 18 = 2SPLIT8A
 - 19 = 2SPLIT8B
 - 20 = 4SPLIT4A
 - 21 = 4SPLIT4B
 - 22 = CF4_2
 - 23 = CF4_3
 - 24 = CF5_3
 - 25 = CF6_4
 - 26 = ES4
 - 27 = TES4
- `nf` : normalization flag, default = 1
 - 0 = off

- 1 = on
- ref : richardson extrapolation flag, default = 0
 - 0 = off
 - 1 = on
- **bound_state_guesses: list or array of bound state guesses, only effective if bsl==1** (Newton bound state location is activated). Default = None

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
 - options : NsevOptionsStruct with the options used

6.2 nsev_wrapper - interact with FNFT library

`FNFTpy.fnft_nsev_wrapper.nsev_wrapper(D, q, T1, T2, Xi1, Xi2, M, K, kappa, options, bound_state_guesses=None)`

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

Optional Arguments:

- **bound_state_guesses: list or array of bound state guesses, only effective if options.bound_state_localization == 1** (Newton bound state location is activated). Default = None

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
- `options` : `NsepOptionsStruct` with the options used

6.3 get, set and print options for nsep wrapper

`FNFTpy.options_handling.fnft_nsev_default_options_wrapper()`

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

Returns:

- `options` : `NsevOptionsStruct` with options for `nsev_wrapper`

`FNFTpy.options_handling.get_nsev_options(bsf=None, bsl=None, niter=None, Dsub=None, dst=None, cst=None, nf=None, dis=None, ref=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
- `Dsub` : number of samples used for ‘subsampling and refine’-method, default = 0 (auto)
- `dst` : type of discrete spectrum, default = 0
 - 0 = NORMING_CONSTANTS
 - 1 = RESIDUES
 - 2 = BOTH
 - 3 = skip computing discrete spectrum
- `cst` : type of continuous spectrum, default = 0
 - 0 = REFLECTION_COEFFICIENT
 - 1 = AB
 - 2 = BOTH

- 3 = skip computing continuous spectrum
- dis : discretization, default = 11
 - 0 = 2SPLIT2_MODAL
 - 1 = BO
 - 2 = 2SPLIT1A
 - 3 = 2SPLIT1B
 - 4 = 2SPLIT2A
 - 5 = 2SPLIT2B
 - 6 = 2SPLIT2S
 - 7 = 2SPLIT3A
 - 8 = 2SPLIT3B
 - 9 = 2SPLIT3S
 - 10 = 2SPLIT4A
 - 11 = 2SPLIT4B
 - 12 = 2SPLIT5A
 - 13 = 2SPLIT5B
 - 14 = 2SPLIT6A
 - 15 = 2SPLIT6B
 - 16 = 2SPLIT7A
 - 17 = 2SPLIT7B
 - 18 = 2SPLIT8A
 - 19 = 2SPLIT8B
 - 20 = 4SPLIT4A
 - 21 = 4SPLIT4B
 - 22 = CF4_2
 - 23 = CF4_3
 - 24 = CF5_3
 - 25 = CF6_4
 - 26 = ES4
 - 27 = TES4
- nf : normalization flag, default = 1
 - 0 = off
 - 1 = on
- ref : richardson extrapolation flag, default = 0
 - 0 = off
 - 1 = on

Returns:

- options : NsevOptionsStruct

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

Print options of a NsevOptionsStruct.

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

Optional arguments:

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

6.4 options NsevOptionsStruct

class `FNFTpy.typesdef.NsevOptionsStruct`

Ctypes options struct for interfacing `fnft_nsev`.

Fields:

- `bound_state_filtering`
- `bound_state_localization`
- `Dsub`
- `niter`
- `discspec_type`
- `contspec_type`
- `normalization_flag`
- `discretization`
- `richardson_extrapolation_flag`

Options can be printed directly to screen, e.g.

`print(get_nsev_options())`

String representation can be generated by

`repr(get_nsev_options())`

NONLINEAR SCHROEDINGER EQUATION WITH VANISHING BOUNDARIES - INVERSE NONLINEAR FOURIER TRANSFORM

7.1 nsev_inverse_xi_wrapper

`FNFTpy.nsev_inverse_xi_wrapper(D, T1, T2, M, dis=None)`

Helper function for `nsev_inverse` to calculate the spectral borders for a given time window.

Return value is an array holding the position of the first and the last spectral sample to be used for `nsev_inverse`.

Arguments:

- D : number of sample points for the time window
- T1, T2 : borders of the time window
- M : number of samples for the continuous spectrum

Optional Arguments:

- dis : discretization, default = 4
 - 0 = 2SPLIT2_MODAL
 - 1 = BO
 - 2 = 2SPLIT1A
 - 3 = 2SPLIT1B
 - 4 = 2SPLIT2A
 - 5 = 2SPLIT2B
 - 6 = 2SPLIT2S
 - 7 = 2SPLIT3A
 - 8 = 2SPLIT3B
 - 9 = 2SPLIT3S
 - 10 = 2SPLIT4A
 - 11 = 2SPLIT4B
 - 12 = 2SPLIT5A
 - 13 = 2SPLIT5B
 - 14 = 2SPLIT6A
 - 15 = 2SPLIT6B
 - 16 = 2SPLIT7A
 - 17 = 2SPLIT7B

- 18 = 2SPLIT8A
- 19 = 2SPLIT8B
- 20 = 4SPLIT4A
- 21 = 4SPLIT4B
- 22 = CF4_2
- 23 = CF4_3
- 24 = CF5_3
- 25 = CF6_4
- 26 = ES4
- 27 = TES4

Returns:

- `rv` : return value of the C-function
- `xi` : two-element C double vector containing XI borders

7.2 `nsev_inverse` - calculate the Inverse Nonlinear Fourier Transform

`FNFTpy.fnft_nsev_inverse_wrapper.nsev_inverse(xivec, tvec, contspec, bound_states, discspec, dis=None, cst=None, csim=None, dst=None, max_iter=None, osf=None, kappa=1)`

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

This function is intended to be ‘clutter-free’, which means it automatically calculates some variables needed to call the C-library. 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 ‘`nsev_inverse_wrapper`’ can be used (see documentation there).

!!! Attention: time and frequency vector can not be choosen independently (yet). use `nsev_inverse_xi_wrapper` to calculate `xivec` from `tvec` !!!

Arguments:

- `xivec`: frequency vector
- `tvec`: time vector
- `contspec`: continuous spectrum (of `xi`). Pass `None` if for pure soliton state.
- `bound_states`: array holding the bound states. Pass `None` if no bound states present.
- `discspec`: discrete spectrum. Pass `None` if no bound states present.

Optional arguments:

- `dis` : discretization, default = 4
 - 0 = 2SPLIT2_MODAL
 - 1 = BO
 - 2 = 2SPLIT1A
 - 3 = 2SPLIT1B
 - 4 = 2SPLIT2A
 - 5 = 2SPLIT2B

- 6 = 2SPLIT2S
- 7 = 2SPLIT3A
- 8 = 2SPLIT3B
- 9 = 2SPLIT3S
- 10 = 2SPLIT4A
- 11 = 2SPLIT4B
- 12 = 2SPLIT5A
- 13 = 2SPLIT5B
- 14 = 2SPLIT6A
- 15 = 2SPLIT6B
- 16 = 2SPLIT7A
- 17 = 2SPLIT7B
- 18 = 2SPLIT8A
- 19 = 2SPLIT8B
- 20 = 4SPLIT4A
- 21 = 4SPLIT4B
- 22 = CF4_2
- 23 = CF4_3
- 24 = CF5_3
- 25 = CF6_4
- 26 = ES4
- 27 = TES4
- cst : type of continuous spectrum, default = 0
 - 0 = REFLECTION_COEFFICIENT
 - 1 = B_OF_XI
 - 2 = B_OF_TAU
- csim : inversion method for the continuous part, default = 0
 - 0 = DEFAULT
 - 1 = TFMATRIX_CONTAINS_REFL_COEFF
 - 2 = TFMATRIX_CONTAINS_AB_FROM_ITER
 - 3 = USE_SEED_POTENTIAL_INSTEAD
- dst : type of discrete spectrum, default = 0
 - 0 = NORMING_CONSTANTS
 - 1 = RESIDUES
- max_iter : maximum number of iterations for iterative methods, default = 100
- osf : oversampling factor, default = 8

7.3 nsev_inverse_wrapper - interact with FNFT library

`FNFTpy.fnft_nsev_inverse_wrapper.nsev_inverse_wrapper`(*M, contspec, Xi1, Xi2, K, bound_states, normconst_or_residues, D, T1, T2, kappa, options*)

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

This function's interface mimics the behavior of the function 'fnft_nsev_inverse' 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_inverse' can be used (see documentation there).

Arguments:

- *M* : number of sample points for continuous spectrum
- *contspec* : numpy array holding the samples of the continuous spectrum (can be None if *M*=0)
- *Xi1, Xi2* [frequencies defining the frequency range of the continuous spectrum.] ! Currently, the positions returned by `nsev_inverse_xi_wrapper` must be used !
- *K* : number of bound states
- *bound_states* : bound states (can be None if *K*=0)
- *normconst_or_residues* : bound state spectral coefficients (can be None if *K*=0)
- *D* : number of samples for the output field
- *T1, T2* : borders of the desired time window
- *kappa* : +1/-1 for focussing / defocussing NSE
- *options* : options for `nsev_inverse` as `NsevInverseOptionsStruct`

Returns:

- *rdict* : dictionary holding the fields (depending on options)
 - *return_value* : return value from FNFT
 - *q* : time field resulting from inverse transform
 - *options* : options for `nsev_inverse` as `NsevInverseOptionsStruct`

7.4 get, set and print options for nsev_inverse_wrapper

`FNFTpy.options_handling.fnft_nsev_inverse_default_options_wrapper`()

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

Returns:

- *options* : `NsevInverseOptionsStruct` with options for `nsev_inverse_wrapper`

`FNFTpy.options_handling.get_nsev_inverse_options`(*dis=None, cst=None, csim=None, dst=None, max_iter=None, osf=None*)

Get a `NsevInverseOptionsStruct` for use with `nsev_inverse_wrapper`.

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

Optional arguments:

- *dis* : discretization, default = 4
 - 0 = 2SPLIT2_MODAL
 - 1 = BO
 - 2 = 2SPLIT1A

- 3 = 2SPLIT1B
- 4 = 2SPLIT2A
- 5 = 2SPLIT2B
- 6 = 2SPLIT2S
- 7 = 2SPLIT3A
- 8 = 2SPLIT3B
- 9 = 2SPLIT3S
- 10 = 2SPLIT4A
- 11 = 2SPLIT4B
- 12 = 2SPLIT5A
- 13 = 2SPLIT5B
- 14 = 2SPLIT6A
- 15 = 2SPLIT6B
- 16 = 2SPLIT7A
- 17 = 2SPLIT7B
- 18 = 2SPLIT8A
- 19 = 2SPLIT8B
- 20 = 4SPLIT4A
- 21 = 4SPLIT4B
- 22 = CF4_2
- 23 = CF4_3
- 24 = CF5_3
- 25 = CF6_4
- 26 = ES4
- 27 = TES4
- cst : type of continuous spectrum, default = 0
 - 0 = REFLECTION_COEFFICIENT
 - 1 = B_OF_XI
 - 2 = B_OF_TAU
- csim : inversion method for the continuous part, default = 0
 - 0 = DEFAULT
 - 1 = TFMATRIX_CONTAINS_REFL_COEFF
 - 2 = TFMATRIX_CONTAINS_AB_FROM_ITER
 - 3 = USE_SEED_POTENTIAL_INSTEAD
- dst : type of discrete spectrum, default = 0
 - 0 = NORMING_CONSTANTS
 - 1 = RESIDUES
- max_iter : maximum number of iterations for iterative methods, default = 100
- osf : oversampling factor, default = 8

Returns:

- options : NsevInverseOptionsStruct

`FNFTpy.options_handling.print_nsev_inverse_options(options=None)`

Print options of a NsevInverseOptionsStruct for nsev_inverse.

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

Optional arguments:

- options : NsevInverseOptionsStruct, e.g. created by `get_nsev_options()`

7.5 options NsevInverseOptionsStruct

class `FNFTpy.typesdef.NsevInverseOptionsStruct`

Ctypes options struct for interfacing `fnft_nsev_inverse`.

Fields:

- discretization
- contspec_type
- contspec_inversion_method
- discspec_type
- max_iter
- oversampling_factor

Options can be printed directly to screen, e.g.

```
print(get_nsev_inverse_options())
```

String representation can be generated by

```
repr(get_nsev_inverse_options())
```

PYTHON MODULE INDEX

f

FNFTpy, 1

F

`fnft_kdvv_default_options_wrapper()` (in module `FNFTpy.options_handling`), 8
`fnft_manakovv_default_options_wrapper()` (in module `FNFTpy.options_handling`), 16
`fnft_nsep_default_options_wrapper()` (in module `FNFTpy.options_handling`), 19
`fnft_nsev_default_options_wrapper()` (in module `FNFTpy.options_handling`), 26
`fnft_nsev_inverse_default_options_wrapper()` (in module `FNFTpy.options_handling`), 32
`FNFTpy`
 module, 1

G

`get_fnft_version()` (in module `FNFTpy`), 3
`get_kdvv_options()` (in module `FNFTpy.options_handling`), 8
`get_lib_path()` (in module `FNFTpy`), 3
`get_manakovv_options()` (in module `FNFTpy.options_handling`), 16
`get_nsep_options()` (in module `FNFTpy.options_handling`), 19
`get_nsev_inverse_options()` (in module `FNFTpy.options_handling`), 32
`get_nsev_options()` (in module `FNFTpy.options_handling`), 26
`get_winmode_param()` (in module `FNFTpy`), 3

K

`kdvv()` (in module `FNFTpy.fnft_kdvv_wrapper`), 5
`kdvv_wrapper()` (in module `FNFTpy.fnft_kdvv_wrapper`), 8
`KdvvOptionsStruct` (class in `FNFTpy.typesdef`), 11

M

`manakovv()` (in module `FNFTpy.fnft_manakovv_wrapper`), 13
`manakovv_wrapper()` (in module `FNFTpy.fnft_manakovv_wrapper`), 15
`ManakovvOptionsStruct` (class in `FNFTpy.typesdef`), 16
`module`
`FNFTpy`, 1

N

`nsep()` (in module `FNFTpy.fnft_nsep_wrapper`), 17
`nsep_wrapper()` (in module `FNFTpy.fnft_nsep_wrapper`), 19
`NsepOptionsStruct` (class in `FNFTpy.typesdef`), 21
`nsev()` (in module `FNFTpy.fnft_nsev_wrapper`), 23
`nsev_inverse()` (in module `FNFTpy.fnft_nsev_inverse_wrapper`), 30
`nsev_inverse_wrapper()` (in module `FNFTpy.fnft_nsev_inverse_wrapper`), 32
`nsev_inverse_xi_wrapper()` (in module `FNFTpy`), 29
`nsev_wrapper()` (in module `FNFTpy.fnft_nsev_wrapper`), 25
`NsevInverseOptionsStruct` (class in `FNFTpy.typesdef`), 34
`NsepOptionsStruct` (class in `FNFTpy.typesdef`), 28

P

`print_fnft_version()` (in module `FNFTpy`), 3
`print_kdvv_options()` (in module `FNFTpy.options_handling`), 11
`print_manakovv_options()` (in module `FNFTpy.options_handling`), 16
`print_nsep_options()` (in module `FNFTpy.options_handling`), 21
`print_nsev_inverse_options()` (in module `FNFTpy.options_handling`), 34
`print_nsev_options()` (in module `FNFTpy.options_handling`), 27