
FNFTpy Documentation

Release 0.2

Christoph Mahnke

Sep 21, 2018

CONTENTS:

1	Module overview	1
2	Auxiliary functions	3
2.1	get and print FNFT version	3
3	Korteweg-de-Fries equation	5
3.1	kdvv - calculate the Nonlinear Fourier Transform	5
3.2	kdvv_wrapper - interact with FNFT library	5
3.3	get, set and view options for kdvv_wrapper	6
3.4	options KdvvOptionsStruct	6
4	Nonlinear Schroedinger equation with periodic boundaries	7
4.1	nsep - calculate the Nonlinear Fourier Transform	7
4.2	nsep_wrapper - interact with FNFT library	8
4.3	get, set and view options for nsep_wrapper	8
4.4	options NsepOptionsStruct	9
5	Nonlinear Schroedinger equation with vanishing boundaries	11
5.1	nsev - calculate the Nonlinear Fourier Transform	11
5.2	nsev_wrapper - interact with FNFT library	12
5.3	get, set and view options for nsev_wrapper	13
5.4	options NsevOptionsStruct	13
6	Nonlinear Schroedinger equation with vanishing boundaries - Inverse Nonlinear Fourier Transform	15
6.1	nsev_inverse_xi_wrapper	15
6.2	nsev_inverse - calculate the Inverse Nonlinear Fourier Transform	15
6.3	nsev_inverse_wrapper - interact with FNFT library	16
6.4	get, set and view options for nsev_inverse_wrapper	17
6.5	options NsevInverseOptionsStruct	17
7	example functions	19
	Python Module Index	21
	Index	23

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

AUXILIARY FUNCTIONS

2.1 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-FRIES EQUATION

3.1 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).

Currently, only the continuous spectrum is calculated.

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 = 17

0 = 2split1a 1 = 2split1b 2 = 2split2a 3 = 2split2b 4 = 2split2s 5 = 2split3a 6 = 2split3b
7 = 2split3s 8 = 2split4a 9 = 2split4b 10 = 2split5a 11 = 2split5b 12 = 2split6a 13 =
2split6b 14 = 2split7a 15 = 2split7b 16 = 2split8a 17 = 2split8b

Returns:

`rdict` : dictionary holding the fields:

`return_value` : return value from FNFT

`cont` : continuous spectrum

3.2 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).

Currently, only the continuous spectrum is calculated.

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 kdv as KdvOptionsStruct. Can be generated e.g. with 'get_kdv_options()'

Returns:

rdict : dictionary holding the fields:

return_value : return value from FNFT

cont : continuous spectrum

3.3 get, set and view options for kdv_wrapper

`FNFTpy.options_handling.fnft_kdv_default_opts_wrapper()`

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

Returns:

options : KdvOptionsStruct with options for kdv_wrapper

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

Print options of a KdvOptionsStruct.

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

Optional arguments:

opts : KdvOptionsStruct, e.g. created by get_kdv_options()

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

Get an KdvOptionsStruct struct for use with kdv_wrapper.

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

Optional arguments:

dis: discretization, default = 17

0 = 2split1a 1 = 2split1b 2 = 2split2a 3 = 2split2b 4 = 2split2s 5 = 2split3a 6 = 2split3b
7 = 2split3s 8 = 2split4a 9 = 2split4b 10 = 2split5a 11 = 2split5b 12 = 2split6a 13 =
2split6b 14 = 2split7a 15 = 2split7b 16 = 2split8a 17 = 2split8b

Returns:

options : KdvOptionsStruct

3.4 options KdvOptionsStruct

`class FNFTpy.typesdef.KdvOptionsStruct`

Ctypes options struct for interfacing fnft_kdv.

Fields:

discretization

NONLINEAR SCHROEDINGER EQUATION WITH PERIODIC BOUNDARIES

4.1 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 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

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

4.2 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 input field

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

4.3 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.

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

4.4 options NsepOptionsStruct

class FNFTpy.typesdef.NsepOptionsStruct

Ctypes options struct for interfacing fnft_nsep.

Fields:

localization
filtering
bounding_box
max_evals
discretization
normalization_flag

NONLINEAR SCHROEDINGER EQUATION WITH VANISHING BOUNDARIES

5.1 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)
```

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 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 = 0

0=fast eigenvalue 1=Newton 2=subsample and refine

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

Dsub : number of samples used for 'subsampling and refine'-method, default = 0 (auto)

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

5.2 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

5.3 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.

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, Dsub=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

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

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

5.4 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

NONLINEAR SCHROEDINGER EQUATION WITH VANISHING BOUNDARIES - INVERSE NONLINEAR FOURIER TRANSFORM

6.1 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 : nse discretization parameter, default = 4

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

Returns:

rv : return value of the C-function

xi : two-element C double vector containing XI borders

6.2 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. 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 vection

contspec: continuous spectrum (of xi)

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

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=Transfermatrix with reflection coefficients 2=Transfermatrix with a,b from iteration 3=seed potential

dst : type of discrete spectrum

0 = norming constants 1 = residues

max_iter : maximum number of iterations for iterative methods, default = 100

osf : oversampling factor, default = 8

6.3 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

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

6.4 get, set and view options for nsev_inverse_wrapper

`FNFTpy.options_handling.fnft_nsev_inverse_default_opts_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.print_nsev_inverse_options(opts=None)`

Print options of a NsevInverseOptionsStruct for nsev_inverse.

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

Optional arguments:

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

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

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=Transfermatrix with reflection coefficients 2=Transfermatrix with a,b from iteration 3=seed potential

dst : type of discrete spectrum

0 = norming constants 1 = residues

max_iter : maximum number of iterations for iterative methods, default = 100

osf : oversampling factor, default = 8

Returns:

options : NsevInverseOptionsStruct

6.5 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

EXAMPLE FUNCTIONS

```
FNFTpy.tests.print_default_options()  
    Print the default options for kdvv, nsep, nsev and nsev_inverse.  
  
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.  
  
FNFTpy.tests.nsevinverseexample()  
    Mimics the C example for calling fnft_nsev_inverse.  
  
FNFTpy.tests.nsevinverseexample2()  
    nsev_inverse_example: create a N=2.2 Satsuma-Yajima pulse from nonlinear spectrum.
```


PYTHON MODULE INDEX

f

`FNFTpy`, [1](#)

F

fnft_kdvv_default_opts_wrapper() (in module FN-FTpy.options_handling), 6
 fnft_nsep_default_opts_wrapper() (in module FN-FTpy.options_handling), 8
 fnft_nsev_default_opts_wrapper() (in module FN-FTpy.options_handling), 13
 fnft_nsev_inverse_default_opts_wrapper() (in module FNFTpy.options_handling), 17
 FNFTpy (module), 1

G

get_fnft_version() (in module FNFTpy), 3
 get_kdvv_options() (in module FN-FTpy.options_handling), 6
 get_nsep_options() (in module FN-FTpy.options_handling), 8
 get_nsev_inverse_options() (in module FN-FTpy.options_handling), 17
 get_nsev_options() (in module FN-FTpy.options_handling), 13

K

kdvv() (in module FNFTpy.fnft_kdvv_wrapper), 5
 kdvv_wrapper() (in module FN-FTpy.fnft_kdvv_wrapper), 5
 kdvvexample() (in module FNFTpy.tests), 19
 KdvvOptionsStruct (class in FNFTpy.typesdef), 6

N

nsep() (in module FNFTpy.fnft_nsep_wrapper), 7
 nsep_wrapper() (in module FN-FTpy.fnft_nsep_wrapper), 8
 nsepexample() (in module FNFTpy.tests), 19
 NsepOptionsStruct (class in FNFTpy.typesdef), 9
 nsev() (in module FNFTpy.fnft_nsev_wrapper), 11
 nsev_inverse() (in module FN-FTpy.fnft_nsev_inverse_wrapper), 15
 nsev_inverse_wrapper() (in module FN-FTpy.fnft_nsev_inverse_wrapper), 16
 nsev_inverse_xi_wrapper() (in module FNFTpy), 15
 nsev_wrapper() (in module FN-FTpy.fnft_nsev_wrapper), 12
 nsevexample() (in module FNFTpy.tests), 19
 nsevinverseexample() (in module FNFTpy.tests), 19
 nsevinverseexample2() (in module FNFTpy.tests), 19

NsevInverseOptionsStruct (class in FNFTpy.typesdef), 17
 NsevOptionsStruct (class in FNFTpy.typesdef), 13

P

print_default_options() (in module FNFTpy.tests), 19
 print_fnft_version() (in module FNFTpy), 3
 print_kdvv_options() (in module FN-FTpy.options_handling), 6
 print_nsep_options() (in module FN-FTpy.options_handling), 8
 print_nsev_inverse_options() (in module FN-FTpy.options_handling), 17
 print_nsev_options() (in module FN-FTpy.options_handling), 13