cython-pocketfft

Release 0.0.1

Matthew Reid

Nov 22, 2020

CONTENTS:

CHAPTER

ONE

REFERENCE

1.1 cypocketfft.wrapper

1.1.1 C-API

size_t

cfft_plan

rfft_plan

cfft_plan **_make_cfft_plan**(size_t *length*)
Prepare a plan for complex fft functions

Parameters

- **length** – The input length for the plan

Returns A C struct with plan data

Return type cfft_plan

void **_destroy_cfft_plan**(*cfft_plan plan*)

int **_cfft_backward**(*cfft_plan plan*, double *c[]*, double *fct*)

_cfft_length(*cfft_plan plan*)

rfft_plan **_make_rfft_plan**(size_t *length*)

void **_destroy_rfft_plan**(*rfft_plan plan*)

1.2 cypocketfft.fft

1.2.1 Python API

cypocketfft.fft.**fft**(*double complex[:] in_arr*, *fct=None*)
Perform the rfft for complex input

Parameters

- **in_arr** (*complex_t [:]*) – Input array or typed-memoryview (complex-valued)
- **fct** (*float, optional*) – Scaling factor to apply to the un-normalized transform. If not provided, defaults to 1.0

Returns The fft result as an array of complex128

Return type numpy.ndarray

cypocketfft.fft.**ifft**(*double complex[:] in_arr*, *fct=None*)
Perform the inverse fft

Parameters

- **in_arr** (*complex_t [:]*) – Input array or typed-memoryview (complex-valued)
- **fct** (*float, optional*) – Scaling factor to apply to the un-normalized transform. If not provided, defaults to $1.0/N$

Returns The ifft result as an array of complex128

Return type numpy.ndarray

cypocketfft.fft.**irfft**(*double complex[:] in_arr*, *fct=None*)
Perform the inverse rfft

Parameters

- **in_arr** (*complex_t [:]*) – Input array or typed-memoryview (complex-valued)
- **fct** (*float, optional*) – Scaling factor to apply to the un-normalized transform. If not provided, defaults to $1.0/N$

Returns The irfft result as an array of float64

Return type numpy.ndarray

cypocketfft.fft.**irfft_length**(*double complex[:] in_arr*)
Calculate the irfft result length for the input array

This evaluates to $(N-1)*2$

Parameters **in_arr** (*complex_t [:]*) – Input array or typed-memoryview (complex-valued)

Returns Result length

Return type Py_ssize_t

cypocketfft.fft.**rfft**(*double[:] in_arr*, *fct=None*)
Perform the rfft for real input

Parameters

- **in_arr** (double *[:]*) – Input array or typed-memoryview (real-valued)
- **fct** (*float, optional*) – Scaling factor to apply to the un-normalized transform. If not provided, defaults to 1.0

Returns The rfft result as an array of complex128

Return type numpy.ndarray

cypocketfft.fft.**rfft_length**(*double[:] in_arr*)
Calculate the rfft result length for the input array

This evaluates to $N/2+1$

Parameters **in_arr** (double *[:]*) – Input array or typed-memoryview (real-valued)

Returns Result length

Return type Py_ssize_t

1.2.2 C API

Helpers

size_t **_rfft_length**(double[:] *in_arr*)
Calculate the rfft result length for the input array

This evaluates to $N/2 + 1$

Parameters

- **in_arr** (*double [:]*) – Input array or typed-memoryview (real-valued)

Returns The result length

size_t **_irfft_length**(*complex_t*[:] *in_arr*)
Calculate the irfft result length for the input array

This evaluates to $(N - 1) * 2$

Parameters

- **in_arr** (*complex_t [:]*) – Input array (complex-valued)

Returns The result length

Real FFTs

Py_ssize_t **_rfft**(double[:] *in_arr*, *complex_t*[:] *out_arr*, double *fct*, bint *use_cache=True*)
Perform the rfft

Parameters

- **in_arr** (*double [:]*) – Input array or typed-memoryview (real-valued)
- **out_arr** (*complex_t [:]*) – Pre-allocated buffer of complex to store the result
- **fct** (*double*) – Scaling factor to apply to the un-normalized transform (typically `1.0`)
- **use_cache** (*bool*) – If `True`, use the built-in plancache

Py_ssize_t **_rfft_with_plan**(*rfft_plan** *plan*, double[:] *in_arr*, *complex_t*[:] *out_arr*, double *fct*)
Perform the rfft with an existing `cypocketfft.wrapper.rfft_plan`

Parameters

- **plan** (*rfft_plan**) – Pointer to a `cypocketfft.wrapper.rfft_plan`
- **in_arr** (*double [:]*) – Input array or typed-memoryview (real-valued)
- **out_arr** (*complex_t [:]*) – Pre-allocated buffer of complex to store the result
- **fct** (*double*) – Scaling factor to apply to the un-normalized transform (typically `1.0`)
- **use_cache** (*bool*) – If `True`, use the built-in plancache

Py_ssize_t **_irfft**(*complex_t*[:] *in_arr*, double[:] *out_arr*, double *fct*, bint *use_cache=True*)
Perform the inverse rfft

Parameters

- **in_arr** (*complex_t [:]*) – Input array or typed-memoryview (complex-valued)
- **out_arr** (*double [:]*) – Pre-allocated buffer of double to store the result
- **fct** (*double*) – Scaling factor to apply to the un-normalized transform (typically $1.0/N$)

- **use_cache** (*bool*) – If `True`, use the built-in plancache

Py_ssize_t **_irfft_with_plan**(*rfft_plan** *plan*, *complex_t*[:] *in_arr*, double[:] *out_arr*, double *fct*)
Perform the inverse rfft with an existing `cypocketfft.wrapper.rfft_plan`

Parameters

- **plan** – Pointer to a `cypocketfft.wrapper.rfft_plan`
- **in_arr** (*complex_t [:]*) – Input array or typed-memoryview (complex-valued)
- **out_arr** (*double [:]*) – Pre-allocated buffer of double to store the result
- **fct** (*double*) – Scaling factor to apply to the un-normalized transform (typically $1.0/N$)

Complex FFTs

Py_ssize_t **_cfft**(*complex_t*[:] *in_arr*, *complex_t*[:] *out_arr*, double *fct*, bint *use_cache=True*)
Perform the complex fft

Parameters

- **in_arr** (*complex_t [:]*) – Input array or typed-memoryview (complex-valued)
- **out_arr** (*complex_t [:]*) – Pre-allocated buffer of complex to store the result
- **fct** (*double*) – Scaling factor to apply to the un-normalized transform (typically `1.0`)
- **use_cache** (*bool*) – If `True`, use the built-in plancache

Py_ssize_t **_icfft**(*complex_t*[:] *in_arr*, *complex_t*[:] *out_arr*, double *fct*, bint *use_cache=True*)
Perform the inverse complex fft

Parameters

- **in_arr** (*complex_t [:]*) – Input array or typed-memoryview (complex-valued)
- **out_arr** (*complex_t [:]*) – Pre-allocated buffer of `double complex` to store the result
- **fct** (*double*) – Scaling factor to apply to the un-normalized transform (typically $1.0/N$)
- **use_cache** (*bool*) – If `True`, use the built-in plancache

Py_ssize_t **_cfft_execute**(*complex_t*[:] *in_arr*, *complex_t*[:] *out_arr*, double *fct*, bint *is_forward=True*, bint *use_cache=True*)
Helper function called by *_cfft()* and c:func:*_icfft* to perform the forward or backward transform

Parameters

- **in_arr** (*complex_t [:]*) – Input array or typed-memoryview (complex-valued)
- **out_arr** (*complex_t [:]*) – Pre-allocated buffer of `double complex` to store the result
- **fct** (*double*) – Scaling factor to apply to the un-normalized transform
- **is_forward** (*bool*) – If `True`, perform the forward fft. If `False`, the inverse
- **use_cache** (*bool*) – If `True`, use the built-in plancache

Py_ssize_t **_cfft_with_plan**(*cfft_plan** *plan*, *complex_t*[:] *in_arr*, *complex_t*[:] *out_arr*, double *fct*, bint *is_forward=True*)
Perform the forward or inverse complex fft with an existing `cypocketfft.wrapper.cfft_plan`

Parameters

- **plan** – Pointer to a `cypocketfft.wrapper.cfft_plan`
- **in_arr** (*complex_t [:]*) – Input array or typed-memoryview (complex-valued)
- **out_arr** (*complex_t [:]*) – Pre-allocated buffer of `double complex` to store the result
- **fct** (*double*) – Scaling factor to apply to the un-normalized transform
- **is_forward** (*bool*) – If `True`, perform the forward fft. If `False`, the inverse

1.3 cypocketfft.plancache

1.3.1 Python API

class cypocketfft.plancache.**PlanCache**
Storage for fft plans

1.4 Types

double
64-bit float

complex_t
128-bit `complex` type. Alias for `double complex`

pyx_memoryview
Array-like object or buffer. Can be `numpy.ndarray`, a Cython Array, a C array or anything supported by Cython Typed Memoryviews

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

C

INDEX

Symbols

C

D

F

I

P

R

S