NumC

1.0

Generated by Doxygen 1.8.14

Contents

Chapter 1

NumC Documentation

1.1 Description

A C++ implementation of the Python Numpy library

Author

David Pilger dpilger26@gmail.com

Version

1.0

1.2 License

1.3 Testing

Compiled and tested with Visual Studio 2017, and MinGW gcc-6.3.0, with Boost version 1.63.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 NumC Namespace Reference

Namespaces

- Constants

  Holds usefull constants.
- Coordinates

  A module for holding and working with coordinates in either Ra/Dec or cartesian formats.
- Filter

  Image and signal filtering.
- Rotations

  Module for dealing with rotations.

Data Structures

- struct Axis

  Enum To describe an axis.
- class BoostNdarrayHelper

  Helper class for ndarray.
- class DataCube

  Convience container for holding a uniform array of NdArrays.
- class DtypeInfo

  Holds info about the dtype.
- struct Endian

  Enum for endianess.
- class FFT

  Class for performing fast forrier tranforms.
- class Filters

  Class for performing many types of image filtering.
- class ImageProcessing

  Class for basic image processing.
- class Linalg

  Class for doing linear algebra operations.
- class Methods

Methods for working with NdArrays.

- class NdArray

  Holds 1D and 2D arrays, the main work horse of the NumC library.
- struct Order

  C or Fortran ordering from python.
- class Polynomial

  Class for dealing with common polynomials.
- class Random

  A class for generating random numbers.
- class Shape

  A Shape Class for NdArrays.
- class Slice

  A Class for slicing into NdArrays.
- class Timer

  A timer class for timing code execution.
- class Utils

  Usefull utility type functions.

Typedefs

- typedef int16_t int16
- typedef int32_t int32
- typedef int64_t int64
- typedef int8_t int8
- typedef uint16_t uint16
- typedef uint32_t uint32
- typedef uint64_t uint64
- typedef uint8_t uint8

Functions

- template< typename dtype >
  NdArray< dtype > boostToNumC (boost::python::numpy::ndarray &inArray)
- template< typename dtype >
  boost::python::numpy::ndarray numCToBoost (const NdArray< dtype > &inArray)

Variables

- boost::random::mt19937 generator_

  generator function

5.1.1 Typedef Documentation

5.1.1.1 int16

```
typedef int16_t NumC::int16
```

5.1.1.2 int32

```
typedef int32_t NumC::int32
```

5.1.1.3 int64

```
typedef int64_t NumC::int64
```

5.1.1.4 int8

```
typedef int8_t NumC::int8
```

5.1.1.5 uint16

```
typedef uint16_t NumC::uint16
```

5.1.1.6 uint32

```
typedef uint32_t NumC::uint32
```

5.1.1.7 uint64

```
typedef uint64_t NumC::uint64
```

5.1.1.8 uint8

```
typedef uint8_t NumC::uint8
```

5.1.2 Function Documentation

5.1.2.1 boostToNumC()

```
template<typename dtype >
NdArray<dtype> NumC::boostToNumC (
            boost::python::numpy::ndarray & inArray )
```

Converts from a boost ndarray to a NumC NdArray<T>

Parameters

ndarray	

Returns

NdArray<T>

5.1.2.2 numCToBoost()

```
template<typename dtype >
boost::python::numpy::ndarray NumC::numCToBoost (
            const NdArray< dtype > & inArray )
```

Converts from a NumC NdArray<T> to a boost ndarray

Parameters

NdArray<T>	

Returns

ndarray

5.1.3 Variable Documentation

5.1.3.1 generator_

```
boost::random::mt19937 NumC::generator_
```

generator function

5.2 NumC::Constants Namespace Reference

Holds usefull constants.

Variables

- const double c = 3.0e8

    speed of light
- const double DAYS_PER_WEEK = 7

    Number of days in a week.
- const double e = 2.718281828459045

    eulers number
- const double HOURS_PER_DAY = 24

    Number of hours in a day.
- const double MILLISECONDS_PER_DAY = SECONDS_PER_DAY * MILLISECONDS_PER_SECOND

    Number of milliseconds in a day.
- const double MILLISECONDS_PER_SECOND = 1000

    Number of milliseconds in a second.
- const double MINUTES_PER_DAY = HOURS_PER_DAY * MINUTES_PER_HOUR

    Number of minutes in a day.
- const double MINUTES_PER_HOUR = 60

    Number of minutes in an hour.
- const double nan = std::nan("1")

    NaN.
- const double pi = 3.14159265358979323846

    Pi.
- const double SECONDS_PER_DAY = MINUTES_PER_DAY * SECONDS_PER_MINUTE

    Number of seconds in a day.
- const double SECONDS_PER_HOUR = MINUTES_PER_HOUR * SECONDS_PER_MINUTE

    Number of seconds in an hour.
- const double SECONDS_PER_MINUTE = 60

    Number of seconds in a minute.
- const double SECONDS_PER_WEEK = SECONDS_PER_DAY * DAYS_PER_WEEK

    Number of seconds in a week.
- const std::string VERSION = "1.0"

    Current NumC version number.

5.2.1 Detailed Description

Holds usefull constants.

5.2.2 Variable Documentation

5.2.2.1 c

```
const double NumC::Constants::c = 3.0e8
```

speed of light

5.2.2.2 DAYS_PER_WEEK

```
const double NumC::Constants::DAYS_PER_WEEK = 7
```

Number of days in a week.

5.2.2.3 e

```
const double NumC::Constants::e = 2.718281828459045
```

eulers number

5.2.2.4 HOURS_PER_DAY

```
const double NumC::Constants::HOURS_PER_DAY = 24
```

Number of hours in a day.

5.2.2.5 MILLISECONDS_PER_DAY

```
const double NumC::Constants::MILLISECONDS_PER_DAY = SECONDS_PER_DAY * MILLISECONDS_PER_SECOND
```

Number of milliseconds in a day.

5.2.2.6 MILLISECONDS_PER_SECOND

```
const double NumC::Constants::MILLISECONDS_PER_SECOND = 1000
```

Number of milliseconds in a second.

5.2.2.7 MINUTES_PER_DAY

```
const double NumC::Constants::MINUTES_PER_DAY = HOURS_PER_DAY * MINUTES_PER_HOUR
```

Number of minutes in a day.

5.2.2.8 MINUTES_PER_HOUR

```
const double NumC::Constants::MINUTES_PER_HOUR = 60
```

Number of minutes in an hour.

5.2.2.9 nan

```
const double NumC::Constants::nan = std::nan("1")
```

NaN.

5.2.2.10 pi

```
const double NumC::Constants::pi = 3.14159265358979323846
```

Pi.

5.2.2.11 SECONDS_PER_DAY

```
const double NumC::Constants::SECONDS_PER_DAY = MINUTES_PER_DAY * SECONDS_PER_MINUTE
```

Number of seconds in a day.

5.2.2.12 SECONDS_PER_HOUR

```
const double NumC::Constants::SECONDS_PER_HOUR = MINUTES_PER_HOUR * SECONDS_PER_MINUTE
```

Number of seconds in an hour.

5.2.2.13 SECONDS_PER_MINUTE

```
const double NumC::Constants::SECONDS_PER_MINUTE = 60
```

Number of seconds in a minute.

5.2.2.14 SECONDS_PER_WEEK

```
const double NumC::Constants::SECONDS_PER_WEEK = SECONDS_PER_DAY * DAYS_PER_WEEK
```

Number of seconds in a week.

5.2.2.15 VERSION

```
const std::string NumC::Constants::VERSION = "1.0"
```

Current NumC version number.

5.3 NumC::Coordinates Namespace Reference

A module for holding and working with coordinates in either Ra/Dec or cartesian formats.

Data Structures

- class Coordinate

  Holds a full coordinate object.
- class Dec

  Holds a Declination object.
- class RA

  Holds a right ascension object.
- struct Sign

  Struct Enum for positive or negative Dec angle.

Functions

- template< typename dtype >
  dtype degreeSeperation (const Coordinate< dtype > &inCoordinate1, const Coordinate< dtype > &in↩ Coordinate2)
- template< typename dtype >
  dtype degreeSeperation (const NdArray< dtype > &inVector1, const NdArray< dtype > &inVector2)
- template< typename dtype >
  dtype radianSeperation (const Coordinate< dtype > &inCoordinate1, const Coordinate< dtype > &in↩ Coordinate2)
- template< typename dtype >
  dtype radianSeperation (const NdArray< dtype > &inVector1, const NdArray< dtype > &inVector2)

5.3.1 Detailed Description

A module for holding and working with coordinates in either Ra/Dec or cartesian formats.

5.3.2 Function Documentation

5.3.2.1 degreeSeperation() [1/2]

```
template<typename dtype >
dtype NumC::Coordinates::degreeSeperation (
            const Coordinate< dtype > & inCoordinate1,
            const Coordinate< dtype > & inCoordinate2 )
```

Returns the degree seperation between the two Coordinates

Parameters

Coordinate	
Coordinate	

Returns

degrees

5.3.2.2 degreeSeperation() [2/2]

```
template<typename dtype >
dtype NumC::Coordinates::degreeSeperation (
            const NdArray< dtype > & inVector1,
            const NdArray< dtype > & inVector2 )
```

Returns the degree seperation between the Coordinate and the input vector

Parameters

NdArray	
NdArray	

Returns

degrees

5.3.2.3 radianSeperation() [1/2]

```
template<typename dtype >
dtype NumC::Coordinates::radianSeperation (
```

```
            const Coordinate< dtype > & inCoordinate1,
            const Coordinate< dtype > & inCoordinate2 )
```

Returns the radian seperation between the two Coordinates

Parameters

Coordinate	
Coordinate	

Returns

radians

5.3.2.4 radianSeperation() [2/2]

```
template<typename dtype >
dtype NumC::Coordinates::radianSeperation (
            const NdArray< dtype > & inVector1,
            const NdArray< dtype > & inVector2 )
```

Returns the radian seperation between the Coordinate and the input vector

Parameters

NdArray	
NdArray	

Returns

radians

5.4 NumC::Filter Namespace Reference

Image and signal filtering.

Data Structures

- struct Boundary

  Boundary condition to apply to the image filter.

5.4.1 Detailed Description

Image and signal filtering.

5.5 NumC::Rotations Namespace Reference

Module for dealing with rotations.

Data Structures

- class DCM

  Factory methods for generating direction cosine matrices and vectors.
- class Quaternion

  Holds a unit quaternion.

5.5.1 Detailed Description

Module for dealing with rotations.

Chapter 6

Data Structure Documentation

6.1 NumC::Axis Struct Reference

Enum To describe an axis.

```
#include <Types.hpp>
```

Public Types

- enum Type { NONE = 0, ROW, COL }

6.1.1 Detailed Description

Enum To describe an axis.

6.1.2 Member Enumeration Documentation

6.1.2.1 Type

```
enum NumC::Axis::Type
```

Enumerator

NONE	
ROW	
COL	

The documentation for this struct was generated from the following file:

- Types.hpp

6.2 NumC::BoostNdarrayHelper Class Reference

Helper class for ndarray.

`#include <BoostNumpyNdarrayHelper.hpp>`

Public Member Functions

- BoostNdarrayHelper (boost::python::numpy::ndarray *inArray)
- BoostNdarrayHelper (boost::python::tuple inShape)
- const boost::python::numpy::ndarray * getArray ()
- boost::python::numpy::matrix getArrayAsMatrix ()
- uint8 numDimensions ()
- double & operator() (uint32 index)
- double & operator() (uint32 index1, uint32 index2)
- double & operator() (uint32 index1, uint32 index2, uint32 index3)
- Order::Type order ()
- void printArray1D ()
- void printArray2D ()
- void printArray3D ()
- const std::vector< Py_intptr_t > & shape ()
- bool shapeEqual (BoostNdarrayHelper &otherNdarrayHelper)
- uint32 size ()
- const std::vector< uint32 > & strides ()

6.2.1 Detailed Description

Helper class for ndarray.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 BoostNdarrayHelper() [1/2]

```
NumC::BoostNdarrayHelper::BoostNdarrayHelper (
            boost::python::numpy::ndarray * inArray )  [inline]
```

Constructor

Parameters

pointer	to an ndarray

Returns

None

6.2.2.2 BoostNdarrayHelper() [2/2]

```
NumC::BoostNdarrayHelper::BoostNdarrayHelper (
            boost::python::tuple inShape )  [inline]
```

Constructor

Parameters

pointer	to an ndarray

Returns

None

6.2.3 Member Function Documentation

6.2.3.1 getArray()

```
const boost::python::numpy::ndarray* NumC::BoostNdarrayHelper::getArray ( )  [inline]
```

Returns the internaly held ndarray

Parameters

None	

Returns

pointer to an ndarray

6.2.3.2 getArrayAsMatrix()

```
boost::python::numpy::matrix NumC::BoostNdarrayHelper::getArrayAsMatrix ( )  [inline]
```

Returns the internaly held ndarray as a numpy matrix

Parameters

None	

Returns

matrix

6.2.3.3 numDimensions()

```
uint8 NumC::BoostNdarrayHelper::numDimensions ( ) [inline]
```

Returns the number of dimensions of the array

Parameters

None	

Returns

num dimensions

6.2.3.4 operator()() [1/3]

```
double& NumC::BoostNdarrayHelper::operator() (
            uint32 index ) [inline]
```

1D access operator

Parameters

None	

Returns

double

6.2.3.5 operator()() [2/3]

```
double& NumC::BoostNdarrayHelper::operator() (
            uint32 index1,
            uint32 index2 ) [inline]
```

2D access operator

Parameters

None	

Returns

double

6.2.3.6 operator()() [3/3]

```
double& NumC::BoostNdarrayHelper::operator() (
            uint32 index1,
            uint32 index2,
            uint32 index3 )  [inline]
```

3D access operator

Parameters

None	

Returns

double

6.2.3.7 order()

```
Order::Type NumC::BoostNdarrayHelper::order ( )  [inline]
```

Returns the memory order of the array (C or Fortran)

Parameters

None	

Returns

Order

6.2.3.8 printArray1D()

```
void NumC::BoostNdarrayHelper::printArray1D ( )  [inline]
```

Prints a 1D array

Parameters

None	

Returns

None

6.2.3.9 printArray2D()

```
void NumC::BoostNdarrayHelper::printArray2D ( )  [inline]
```

Prints a 2D array

Parameters

None	

Returns

None

6.2.3.10 printArray3D()

```
void NumC::BoostNdarrayHelper::printArray3D ( )  [inline]
```

Prints a 3D array

Parameters

None	

Returns

None

6.2.3.11 shape()

```
const std::vector<Py_intptr_t>& NumC::BoostNdarrayHelper::shape ( )  [inline]
```

Returns the shape of the array

Parameters

None	

Returns

vector

6.2.3.12 shapeEqual()

```
bool NumC::BoostNdarrayHelper::shapeEqual (
            BoostNdarrayHelper & otherNdarrayHelper )  [inline]
```

Returns if the shapes of the two array helpers are equal

Parameters

None	

Returns

boolean

6.2.3.13 size()

```
uint32 NumC::BoostNdarrayHelper::size ( )  [inline]
```

Returns the size of the array

Parameters

None	

Returns

size

6.2.3.14 strides()

```
const std::vector<uint32>& NumC::BoostNdarrayHelper::strides ( )  [inline]
```

Returns the strides of the array

Parameters

None	

Returns

vector

The documentation for this class was generated from the following file:

- BoostNumpyNdarrayHelper.hpp

6.3 NumC::Filter::Boundary Struct Reference

Boundary condition to apply to the image filter.

`#include <Filter.hpp>`

Public Types

- enum Mode {
  REFLECT = 0, CONSTANT, NEAREST, MIRROR,
  WRAP }

6.3.1 Detailed Description

Boundary condition to apply to the image filter.

6.3.2 Member Enumeration Documentation

6.3.2.1 Mode

`enum NumC::Filter::Boundary::Mode`

Enumerator

REFLECT	
CONSTANT	
NEAREST	
MIRROR	
WRAP	

The documentation for this struct was generated from the following file:

- Filter.hpp

6.4 NumC::ImageProcessing< dtype >::Centroid Class Reference

holds the information for a centroid

```
#include <ImageProcessing.hpp>
```

Public Member Functions

- Centroid ()
- Centroid (const Cluster &inCluster)
- double col () const
- double eod () const
- dtype intensity () const
- bool operator!= (const Centroid &rhs) const
- bool operator< (const Centroid &rhs) const
- bool operator== (const Centroid &rhs) const
- void print () const
- double row () const
- std::string str () const

Friends

- std::ostream & operator<< (std::ostream &inStream, const Centroid &inCentriod)

6.4.1 Detailed Description

template<typename dtype>
class NumC::ImageProcessing< dtype >::Centroid

holds the information for a centroid

6.4.2 Constructor & Destructor Documentation

6.4.2.1 Centroid() [1/2]

```
template<typename dtype >
NumC::ImageProcessing< dtype >::Centroid::Centroid ( )  [inline]
```

defualt constructor needed by containers

Parameters

None	

Returns

None

6.4.2.2 Centroid() [2/2]

```
template<typename dtype >
NumC::ImageProcessing< dtype >::Centroid::Centroid (
            const Cluster & inCluster )  [inline]
```

constructor

Parameters

centroid	id,
FP	row,
FP	column,
centroid	intensity
cluster	EOD
cluster	number of pixels

Returns

None

6.4.3 Member Function Documentation

6.4.3.1 col()

```
template<typename dtype >
double NumC::ImageProcessing< dtype >::Centroid::col ( ) const  [inline]
```

gets the centroid col

Parameters

None	

Returns

centroid col

6.4.3.2 eod()

```
template<typename dtype >
double NumC::ImageProcessing< dtype >::Centroid::eod ( ) const [inline]
```

returns the estimated eod of the centroid

Parameters

None	

Returns

star id

6.4.3.3 intensity()

```
template<typename dtype >
dtype NumC::ImageProcessing< dtype >::Centroid::intensity ( ) const [inline]
```

gets the centroid intensity

Parameters

None	

Returns

centroid intensity

6.4.3.4 operator"!=()

```
template<typename dtype >
bool NumC::ImageProcessing< dtype >::Centroid::operator!= (
            const Centroid & rhs ) const [inline]
```

not equality operator

Parameters

None	

Returns

bool

6.4.3.5 operator<()

```
template<typename dtype >
bool NumC::ImageProcessing< dtype >::Centroid::operator< (
            const Centroid & rhs ) const  [inline]
```

less than operator for std::sort algorithm; NOTE: std::sort sorts in ascending order. Since I want to sort the centroids in descensing order, I am purposefully defining this operator backwards!

Parameters

None	

Returns

None

6.4.3.6 operator==()

```
template<typename dtype >
bool NumC::ImageProcessing< dtype >::Centroid::operator== (
            const Centroid & rhs ) const  [inline]
```

equality operator

Parameters

None	

Returns

bool

6.4.3.7 print()

```
template<typename dtype >
void NumC::ImageProcessing< dtype >::Centroid::print ( ) const [inline]
```

Method Description: prints the Centroid object to the console

Parameters

None	

Returns

None

6.4.3.8 row()

```
template<typename dtype >
double NumC::ImageProcessing< dtype >::Centroid::row ( ) const [inline]
```

gets the centroid row

Parameters

None	

Returns

centroid row

6.4.3.9 str()

```
template<typename dtype >
std::string NumC::ImageProcessing< dtype >::Centroid::str ( ) const [inline]
```

returns the centroid as a string representation

Parameters

None	

Returns

string

6.4.4 Friends And Related Function Documentation

6.4.4.1 operator<<

```
template<typename dtype >
std::ostream& operator<< (
            std::ostream & inStream,
            const Centroid & inCentriod )  [friend]
```

ostream operator

Parameters

std::ostream	
Centroid	

Returns

std::ostream

The documentation for this class was generated from the following file:

- ImageProcessing.hpp

6.5 NumC::ImageProcessing< dtype >::Cluster Class Reference

Holds the information for a cluster of pixels.

`#include <ImageProcessing.hpp>`

Public Types

- typedef std::vector< Pixel >::const_iterator const_iterator

Public Member Functions

- Cluster (uint32 inClusterId)
- void addPixel (const Pixel &inPixel)
- const Pixel & at (uint32 inIndex) const
- const_iterator begin () const
- uint32 clusterId () const
- uint32 colMax () const
- uint32 colMin () const
- const_iterator end () const
- double eod () const

- uint32 height () const
- dtype intensity () const
- bool operator!= (const Cluster &rhs) const
- bool operator== (const Cluster &rhs) const
- const Pixel & operator[ ] (uint32 inIndex) const
- dtype peakPixelIntensity () const
- void print () const
- uint32 rowMax () const
- uint32 rowMin () const
- uint32 size () const
- std::string str () const
- uint32 width () const

Friends

- std::ostream & operator<< (std::ostream &inStream, const Cluster &inCluster)

6.5.1 Detailed Description

template<typename dtype>
class NumC::ImageProcessing< dtype >::Cluster

Holds the information for a cluster of pixels.

6.5.2 Member Typedef Documentation

6.5.2.1 const_iterator

```
template<typename dtype >
typedef std::vector<Pixel>::const_iterator NumC::ImageProcessing< dtype >::Cluster::const_iterator
```

6.5.3 Constructor & Destructor Documentation

6.5.3.1 Cluster()

```
template<typename dtype >
NumC::ImageProcessing< dtype >::Cluster::Cluster (
            uint32 inClusterId )  [inline]
```

default constructor needed by containers

Parameters

in←→ ClusterId	

Returns

None

6.5.4 Member Function Documentation

6.5.4.1 addPixel()

```
template<typename dtype >
void NumC::ImageProcessing< dtype >::Cluster::addPixel (
            const Pixel & inPixel )  [inline]
```

adds a pixel to the cluster

Parameters

pixel	

Returns

None

6.5.4.2 at()

```
template<typename dtype >
const Pixel& NumC::ImageProcessing< dtype >::Cluster::at (
            uint32 inIndex ) const  [inline]
```

access method with bounds checking

Parameters

index	

Returns

Pixel

6.5.4.3 begin()

```
template<typename dtype >
const_iterator NumC::ImageProcessing< dtype >::Cluster::begin ( ) const [inline]
```

returns in iterator to the beginning pixel of the cluster

Parameters

None	

Returns

const_iterator

6.5.4.4 clusterId()

```
template<typename dtype >
uint32 NumC::ImageProcessing< dtype >::Cluster::clusterId ( ) const [inline]
```

returns the minimum row number of the cluster

Parameters

None	

Returns

minimum row number of the cluster

6.5.4.5 colMax()

```
template<typename dtype >
uint32 NumC::ImageProcessing< dtype >::Cluster::colMax ( ) const [inline]
```

returns the maximum column number of the cluster

Parameters

None	

Returns

maximum column number of the cluster

6.5.4.6 colMin()

```
template<typename dtype >
uint32 NumC::ImageProcessing< dtype >::Cluster::colMin ( ) const [inline]
```

returns the minimum column number of the cluster

Parameters

None	

Returns

minimum column number of the cluster

6.5.4.7 end()

```
template<typename dtype >
const_iterator NumC::ImageProcessing< dtype >::Cluster::end ( ) const [inline]
```

returns in iterator to the 1 past the end pixel of the cluster

Parameters

None	

Returns

const_iterator

6.5.4.8 eod()

```
template<typename dtype >
double NumC::ImageProcessing< dtype >::Cluster::eod ( ) const [inline]
```

returns the cluster estimated energy on detector (EOD)

Parameters

None	

Returns

eod

6.5.4.9 height()

```
template<typename dtype >
uint32 NumC::ImageProcessing< dtype >::Cluster::height ( ) const [inline]
```

returns the number of rows the cluster spans

Parameters

None	

Returns

number of rows

6.5.4.10 intensity()

```
template<typename dtype >
dtype NumC::ImageProcessing< dtype >::Cluster::intensity ( ) const [inline]
```

returns the summed intensity of the cluster

Parameters

None	

Returns

summed cluster intensity

6.5.4.11 operator"!=()

```
template<typename dtype >
bool NumC::ImageProcessing< dtype >::Cluster::operator!= (
            const Cluster & rhs ) const [inline]
```

not equality operator

Parameters

Cluster	

Returns

bool

6.5.4.12 operator==()

```
template<typename dtype >
bool NumC::ImageProcessing< dtype >::Cluster::operator== (
            const Cluster & rhs ) const [inline]
```

equality operator

Parameters

Cluster	

Returns

bool

6.5.4.13 operator[]()

```
template<typename dtype >
const Pixel& NumC::ImageProcessing< dtype >::Cluster::operator[] (
            uint32 inIndex ) const [inline]
```

access operator, no bounds checking

Parameters

index	

Returns

Pixel

6.5.4.14 peakPixelIntensity()

```
template<typename dtype >
dtype NumC::ImageProcessing< dtype >::Cluster::peakPixelIntensity ( ) const [inline]
```

returns the intensity of the peak pixel in the cluster

Parameters

None	

Returns

peak pixel intensity

6.5.4.15 print()

```
template<typename dtype >
void NumC::ImageProcessing< dtype >::Cluster::print ( ) const [inline]
```

Method Description: prints the Cluster object to the console

Parameters

None	

Returns

None

6.5.4.16 rowMax()

```
template<typename dtype >
uint32 NumC::ImageProcessing< dtype >::Cluster::rowMax ( ) const [inline]
```

returns the maximum row number of the cluster

Parameters

None	

Returns

maximum row number of the cluster

6.5.4.17 rowMin()

```
template<typename dtype >
uint32 NumC::ImageProcessing< dtype >::Cluster::rowMin ( ) const [inline]
```

returns the minimum row number of the cluster

Parameters

None	

Returns

minimum row number of the cluster

6.5.4.18 size()

```
template<typename dtype >
uint32 NumC::ImageProcessing< dtype >::Cluster::size ( ) const [inline]
```

returns the number of pixels in the cluster

Parameters

None	

Returns

number of pixels in the cluster

6.5.4.19 str()

```
template<typename dtype >
std::string NumC::ImageProcessing< dtype >::Cluster::str ( ) const [inline]
```

returns a string representation of the cluster

Parameters

None	

Returns

string

6.5.4.20 width()

```
template<typename dtype >
uint32 NumC::ImageProcessing< dtype >::Cluster::width ( ) const [inline]
```

returns the number of columns the cluster spans

Parameters

None	

Returns

number of columns

6.5.5 Friends And Related Function Documentation

6.5.5.1 operator<<

```
template<typename dtype >
std::ostream& operator<< (
            std::ostream & inStream,
            const Cluster & inCluster )   [friend]
```

osstream operator

Parameters

std::ostream	
Cluster	

Returns

std::ostream

The documentation for this class was generated from the following file:

- ImageProcessing.hpp

6.6 NumC::Coordinates::Coordinate< dtype > Class Template Reference

Holds a full coordinate object.

```
#include <Coordinates.hpp>
```

Public Member Functions

- Coordinate ()
- Coordinate (dtype inRaDegrees, dtype inDecDegrees)
- Coordinate (uint8 inRaHours, uint8 inRaMinutes, dtype inRaSeconds, Sign::Type inSign, uint8 inDec←
DegreesWhole, uint8 inDecMinutes, dtype inDecSeconds)

- Coordinate (const RA< dtype > &inRA, const Dec< dtype > &inDec)
- Coordinate (dtype inX, dtype inY, dtype inZ)
- Coordinate (const NdArray< dtype > inCartesianVector)
- template<typename dtypeOut >
  Coordinate< dtypeOut > astype ()
- const Dec< dtype > & dec () const
- dtype degreeSeperation (const Coordinate< dtype > &inOtherCoordinate) const
- dtype degreeSeperation (const NdArray< dtype > &inVector) const
- bool operator!= (const Coordinate< dtype > &inRhs) const
- bool operator== (const Coordinate< dtype > &inRhs) const
- void print () const
- const RA< dtype > & ra () const
- dtype radianSeperation (const Coordinate< dtype > &inOtherCoordinate) const
- dtype radianSeperation (const NdArray< dtype > &inVector) const
- std::string str () const
- dtype x () const
- NdArray< dtype > xyz () const
- dtype y () const
- dtype z () const

Friends

- std::ostream & operator<< (std::ostream &inStream, const Coordinate< dtype > &inCoord)

6.6.1 Detailed Description

template<typename dtype>
class NumC::Coordinates::Coordinate< dtype >

Holds a full coordinate object.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 Coordinate() [1/6]

```
template<typename dtype>
NumC::Coordinates::Coordinate< dtype >::Coordinate ( )  [inline]
```

Default Constructor, not super usefull on its own

Parameters

None	

Returns

None

6.6.2.2 Coordinate() [2/6]

```
template<typename dtype>
NumC::Coordinates::Coordinate< dtype >::Coordinate (
            dtype inRaDegrees,
            dtype inDecDegrees )  [inline]
```

Constructor

Parameters

RA	degrees
Dec	degrees

Returns

None

6.6.2.3 Coordinate() [3/6]

```
template<typename dtype>
NumC::Coordinates::Coordinate< dtype >::Coordinate (
            uint8 inRaHours,
            uint8 inRaMinutes,
            dtype inRaSeconds,
            Sign::Type inSign,
            uint8 inDecDegreesWhole,
            uint8 inDecMinutes,
            dtype inDecSeconds )  [inline]
```

Constructor

Parameters

RA	hours
RA	minutes
RA	seconds
Dec	degrees whole
Dec	minutes
Dec	seconds

Returns

None

6.6.2.4 Coordinate() [4/6]

```
template<typename dtype>
NumC::Coordinates::Coordinate< dtype >::Coordinate (
            const RA< dtype > & inRA,
            const Dec< dtype > & inDec ) [inline]
```

Constructor

Parameters

RA	
Dec	

Returns

None

6.6.2.5 Coordinate() [5/6]

```
template<typename dtype>
NumC::Coordinates::Coordinate< dtype >::Coordinate (
            dtype inX,
            dtype inY,
            dtype inZ ) [inline]
```

Constructor

Parameters

x	
y	
z	

Returns

None

6.6.2.6 Coordinate() [6/6]

```
template<typename dtype>
NumC::Coordinates::Coordinate< dtype >::Coordinate (
            const NdArray< dtype > inCartesianVector )  [inline]
```

Constructor

Parameters

NdArray	

Returns

None

6.6.3 Member Function Documentation

6.6.3.1 astype()

```
template<typename dtype>
template<typename dtypeOut >
Coordinate<dtypeOut> NumC::Coordinates::Coordinate< dtype >::astype ( )  [inline]
```

Returns a new Coordinate object with the specified type

Parameters

None	

Returns

Coordinate

6.6.3.2 dec()

```
template<typename dtype>
const Dec<dtype>& NumC::Coordinates::Coordinate< dtype >::dec ( ) const  [inline]
```

Returns the Dec object

Parameters

None	

Returns

Dec

6.6.3.3 degreeSeperation() [1/2]

```
template<typename dtype>
dtype NumC::Coordinates::Coordinate< dtype >::degreeSeperation (
            const Coordinate< dtype > & inOtherCoordinate ) const [inline]
```

Returns the degree seperation between the two Coordinates

Parameters

Coordinate	

Returns

degrees

6.6.3.4 degreeSeperation() [2/2]

```
template<typename dtype>
dtype NumC::Coordinates::Coordinate< dtype >::degreeSeperation (
            const NdArray< dtype > & inVector ) const [inline]
```

Returns the degree seperation between the Coordinate and the input vector

Parameters

NdArray	

Returns

degrees

6.6.3.5 operator"!=()

```
template<typename dtype>
bool NumC::Coordinates::Coordinate< dtype >::operator!= (
            const Coordinate< dtype > & inRhs ) const [inline]
```

Not equality operator

Parameters

None	

Returns

bool

6.6.3.6 operator==()

```
template<typename dtype>
bool NumC::Coordinates::Coordinate< dtype >::operator== (
            const Coordinate< dtype > & inRhs ) const [inline]
```

Equality operator

Parameters

None	

Returns

bool

6.6.3.7 print()

```
template<typename dtype>
void NumC::Coordinates::Coordinate< dtype >::print ( ) const [inline]
```

Prints the Coordinate object to the console

Parameters

None	

Returns

None

6.6.3.8 ra()

```
template<typename dtype>
const RA<dtype>& NumC::Coordinates::Coordinate< dtype >::ra ( ) const [inline]
```

Returns the RA object

Parameters

None	

Returns

RA

6.6.3.9 radianSeperation() [1/2]

```
template<typename dtype>
dtype NumC::Coordinates::Coordinate< dtype >::radianSeperation (
            const Coordinate< dtype > & inOtherCoordinate ) const  [inline]
```

Returns the radian seperation between the two Coordinates

Parameters

Coordinate	

Returns

radians

6.6.3.10 radianSeperation() [2/2]

```
template<typename dtype>
dtype NumC::Coordinates::Coordinate< dtype >::radianSeperation (
            const NdArray< dtype > & inVector ) const  [inline]
```

Returns the radian seperation between the Coordinate and the input vector

Parameters

NdArray	

Returns

radians

6.6.3.11 str()

```
template<typename dtype>
std::string NumC::Coordinates::Coordinate< dtype >::str ( ) const  [inline]
```

Returns coordinate as a string representation

Parameters

None	

Returns

string

6.6.3.12 x()

```
template<typename dtype>
dtype NumC::Coordinates::Coordinate< dtype >::x ( ) const  [inline]
```

Returns the cartesian x value

Parameters

None	

Returns

x

6.6.3.13 xyz()

```
template<typename dtype>
NdArray<dtype> NumC::Coordinates::Coordinate< dtype >::xyz ( ) const  [inline]
```

Returns the cartesian xyz triplet as an NdArray

Parameters

None	

Returns

NdArray

6.6.3.14 y()

```
template<typename dtype>
dtype NumC::Coordinates::Coordinate< dtype >::y ( ) const  [inline]
```

Returns the cartesian y value

Parameters

None	

Returns

y

6.6.3.15 z()

```
template<typename dtype>
dtype NumC::Coordinates::Coordinate< dtype >::z ( ) const  [inline]
```

Returns the cartesian z value

Parameters

None	

Returns

z

6.6.4 Friends And Related Function Documentation

6.6.4.1 operator<<

```
template<typename dtype>
std::ostream& operator<< (
            std::ostream & inStream,
            const Coordinate< dtype > & inCoord )  [friend]
```

Ostream operator

Parameters

None	

Returns

None

The documentation for this class was generated from the following file:

- Coordinates.hpp

6.7 NumC::DataCube< dtype > Class Template Reference

Convience container for holding a uniform array of NdArrays.

```
#include <DataCube.hpp>
```

Public Types

- typedef std::deque< NdArray< dtype > >::const_iterator const_iterator
- typedef std::deque< NdArray< dtype > >::iterator iterator

Public Member Functions

- DataCube ()
- DataCube (uint32 inSize)
- NdArray< dtype > & at (uint32 inIndex)
- const NdArray< dtype > & at (uint32 inIndex) const
- NdArray< dtype > & back ()
- iterator begin ()
- const_iterator cbegin () const
- const_iterator cend () const
- void dump (const std::string &inFilename) const
- iterator end ()
- NdArray< dtype > & front ()
- bool isempty ()
- NdArray< dtype > & operator[ ] (uint32 inIndex)
- const NdArray< dtype > & operator[ ] (uint32 inIndex) const
- void pop_back ()
- void pop_front ()
- void push_back (const NdArray< dtype > &inArray)
- void push_front (const NdArray< dtype > &inArray)
- const Shape & shape () const
- uint32 size () const

6.7.1 Detailed Description

template<typename dtype>
class NumC::DataCube< dtype >

Convience container for holding a uniform array of NdArrays.

6.7.2 Member Typedef Documentation

6.7.2.1 const_iterator

```
template<typename dtype >
typedef std::deque<NdArray<dtype> >::const_iterator NumC::DataCube< dtype >::const_iterator
```

6.7.2.2 iterator

```
template<typename dtype >
typedef std::deque<NdArray<dtype> >::iterator NumC::DataCube< dtype >::iterator
```

6.7.3 Constructor & Destructor Documentation

6.7.3.1 DataCube() [1/2]

```
template<typename dtype >
NumC::DataCube< dtype >::DataCube ( )  [inline]
```

Default Constructor

Parameters

None	

Returns

None

6.7.3.2 DataCube() [2/2]

```
template<typename dtype >
NumC::DataCube< dtype >::DataCube (
            uint32 inSize ) [inline]
```

Constructor, preallocates to the input size

Parameters

size	

Returns

None

6.7.4 Member Function Documentation

6.7.4.1 at() [1/2]

```
template<typename dtype >
NdArray<dtype>& NumC::DataCube< dtype >::at (
            uint32 inIndex ) [inline]
```

Access method, with bounds checking

Parameters

index	

Returns

NdArray

6.7.4.2 at() [2/2]

```
template<typename dtype >
const NdArray<dtype>& NumC::DataCube< dtype >::at (
            uint32 inIndex ) const [inline]
```

Const access method, with bounds checking

Parameters

index	

Returns

NdArray

6.7.4.3 back()

```
template<typename dtype >
NdArray<dtype>& NumC::DataCube< dtype >::back ( ) [inline]
```

Returns a reference to the last element of the array

Parameters

None	

Returns

NdArray&

6.7.4.4 begin()

```
template<typename dtype >
iterator NumC::DataCube< dtype >::begin ( ) [inline]
```

Returns an iterator to the beginning of the container

Parameters

None	

Returns

iterator

6.7.4.5 cbegin()

```
template<typename dtype >
const_iterator NumC::DataCube< dtype >::cbegin ( ) const [inline]
```

Returns a const_iterator to the beginning of the container

Parameters

None	

Returns

const_iterator

6.7.4.6 cend()

```
template<typename dtype >
const_iterator NumC::DataCube< dtype >::cend ( ) const [inline]
```

Returns a const_iterator to 1 past the end of the container

Parameters

None	

Returns

const_iterator

6.7.4.7 dump()

```
template<typename dtype >
void NumC::DataCube< dtype >::dump (
            const std::string & inFilename ) const [inline]
```

Outputs the DataCube as a .bin file

Parameters

None	

Returns

None

6.7.4.8 end()

```
template<typename dtype >
iterator NumC::DataCube< dtype >::end ( ) [inline]
```

Returns an iterator to 1 past the end of the container

Parameters

None	

Returns

iterator

6.7.4.9 front()

```
template<typename dtype >
NdArray<dtype>& NumC::DataCube< dtype >::front ( ) [inline]
```

returns a reference to the first element of the array

Parameters

None	

Returns

NdArray&

6.7.4.10 isempty()

```
template<typename dtype >
bool NumC::DataCube< dtype >::isempty ( ) [inline]
```

Tests whether or not the container is empty

Parameters

None	

Returns

bool

6.7.4.11 operator[]() [1/2]

```
template<typename dtype >
NdArray<dtype>& NumC::DataCube< dtype >::operator[ ] (
            uint32 inIndex ) [inline]
```

Access operator, no bounds checking

Parameters

index	

Returns

NdArray

6.7.4.12 operator[]() [2/2]

```
template<typename dtype >
const NdArray<dtype>& NumC::DataCube< dtype >::operator[ ] (
            uint32 inIndex ) const [inline]
```

Const access operator, no bounds checking

Parameters

index	

Returns

NdArray

6.7.4.13 pop_back()

```
template<typename dtype >
void NumC::DataCube< dtype >::pop_back ( ) [inline]
```

Removes the last element in the container

Parameters

None	

Returns

None

6.7.4.14 pop_front()

```
template<typename dtype >
void NumC::DataCube< dtype >::pop_front ( ) [inline]
```

Removes the first element in the container

Parameters

None	

Returns

None

6.7.4.15 push_back()

```
template<typename dtype >
void NumC::DataCube< dtype >::push_back (
            const NdArray< dtype > & inArray ) [inline]
```

Adds a new element at the end of the container

Parameters

NdArray	

Returns

None

6.7.4.16 push_front()

```
template<typename dtype >
void NumC::DataCube< dtype >::push_front (
            const NdArray< dtype > & inArray ) [inline]
```

Adds a new element at the beginning of the container

Parameters

NdArray	

Returns

None

6.7.4.17 shape()

```
template<typename dtype >
const Shape& NumC::DataCube< dtype >::shape ( ) const [inline]
```

returns the number shape of the element arrays

Parameters

None	

Returns

Shape

6.7.4.18 size()

```
template<typename dtype >
uint32 NumC::DataCube< dtype >::size ( ) const [inline]
```

Returns the size of the container array

Parameters

None	

Returns

size

The documentation for this class was generated from the following file:

- DataCube.hpp

6.8 NumC::Rotations::DCM< dtype > Class Template Reference

Factory methods for generating direction cosine matrices and vectors.

```
#include <Rotations.hpp>
```

Static Public Member Functions

- static NdArray< double > angleAxisRotation (const NdArray< dtype > &inArray, double inAngle)
- static bool isValid (const NdArray< dtype > &inArray)
- static NdArray< double > xRotation (double inAngle)
- static NdArray< double > yRotation (double inAngle)
- static NdArray< double > zRotation (double inAngle)

6.8.1 Detailed Description

template<typename dtype>
class NumC::Rotations::DCM< dtype >

Factory methods for generating direction cosine matrices and vectors.

6.8.2 Member Function Documentation

6.8.2.1 angleAxisRotation()

```
template<typename dtype >
static NdArray<double> NumC::Rotations::DCM< dtype >::angleAxisRotation (
            const NdArray< dtype > & inArray,
            double inAngle ) [inline], [static]
```

returns a direction cosine matrix that rotates about the input axis by the input angle

Parameters

NdArray,cartesian	vector with x,y,z
rotation	angle, in radians

Returns

NdArray

6.8.2.2 isValid()

```
template<typename dtype >
static bool NumC::Rotations::DCM< dtype >::isValid (
            const NdArray< dtype > & inArray ) [inline], [static]
```

returns whether the input array is a direction cosine matrix

Parameters

NdArray	

Returns

bool

6.8.2.3 xRotation()

```
template<typename dtype >
static NdArray<double> NumC::Rotations::DCM< dtype >::xRotation (
            double inAngle ) [inline], [static]
```

returns a direction cosine matrix that rotates about the x axis by the input angle

Parameters

rotation	angle, in radians

Returns

NdArray

6.8.2.4 yRotation()

```
template<typename dtype >
static NdArray<double> NumC::Rotations::DCM< dtype >::yRotation (
            double inAngle ) [inline], [static]
```

returns a direction cosine matrix that rotates about the x axis by the input angle

Parameters

rotation	angle, in radians

Returns

NdArray

6.8.2.5 zRotation()

```
template<typename dtype >
static NdArray<double> NumC::Rotations::DCM< dtype >::zRotation (
            double inAngle ) [inline], [static]
```

returns a direction cosine matrix that rotates about the x axis by the input angle

Parameters

rotation	angle, in radians

Returns

NdArray

The documentation for this class was generated from the following file:

- Rotations.hpp

6.9 NumC::Coordinates::Dec< dtype > Class Template Reference

Holds a Declination object.

```
#include <Coordinates.hpp>
```

Public Member Functions

- Dec ()
- Dec (dtype inDegrees)
- Dec (Sign::Type inSign, uint8 inDegrees, uint8 inMinutes, dtype inSeconds)
- template<typename dtypeOut >
  Dec< dtypeOut > astype ()
- dtype degrees () const
- uint8 degreesWhole () const
- uint8 minutes () const
- bool operator!= (const Dec< dtype > &inRhs) const
- bool operator== (const Dec< dtype > &inRhs) const
- void print () const
- dtype radians () const
- dtype seconds () const
- Sign::Type sign () const
- std::string str () const

Friends

- std::ostream & operator<< (std::ostream &inStream, const Dec< dtype > &inDec)

6.9.1 Detailed Description

template<typename dtype>
class NumC::Coordinates::Dec< dtype >

Holds a Declination object.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 Dec() [1/3]

```
template<typename dtype>
NumC::Coordinates::Dec< dtype >::Dec ( )  [inline]
```

Default Constructor, not super usefull on its own

Parameters

None	

Returns

None

6.9.2.2 Dec() [2/3]

```
template<typename dtype>
NumC::Coordinates::Dec< dtype >::Dec (
            dtype inDegrees )  [inline]
```

Constructor

Parameters

degrees	

Returns

None

6.9.2.3 Dec() [3/3]

```
template<typename dtype>
NumC::Coordinates::Dec< dtype >::Dec (
            Sign::Type inSign,
            uint8 inDegrees,
            uint8 inMinutes,
            dtype inSeconds )  [inline]
```

Constructor

Parameters

Sign::Type	
hours	
minutes	
seconds	

Returns

None

6.9.3 Member Function Documentation

6.9.3.1 astype()

```
template<typename dtype>
template<typename dtypeOut >
Dec<dtypeOut> NumC::Coordinates::Dec< dtype >::astype ( )  [inline]
```

Returns a copy of the Dec object as a different type

Parameters

None	

Returns

Dec

6.9.3.2 degrees()

```
template<typename dtype>
dtype NumC::Coordinates::Dec< dtype >::degrees ( ) const [inline]
```

Get the degrees value

Parameters

None	

Returns

degrees

6.9.3.3 degreesWhole()

```
template<typename dtype>
uint8 NumC::Coordinates::Dec< dtype >::degreesWhole ( ) const [inline]
```

Get the whole degrees value

Parameters

None	

Returns

whole degrees

6.9.3.4 minutes()

```
template<typename dtype>
uint8 NumC::Coordinates::Dec< dtype >::minutes ( ) const [inline]
```

Get the minute value

Parameters

None	

Returns

minutes

6.9.3.5 operator"!=()

```
template<typename dtype>
bool NumC::Coordinates::Dec< dtype >::operator!= (
            const Dec< dtype > & inRhs ) const  [inline]
```

Not equality operator

Parameters

None	

Returns

bool

6.9.3.6 operator==()

```
template<typename dtype>
bool NumC::Coordinates::Dec< dtype >::operator== (
            const Dec< dtype > & inRhs ) const  [inline]
```

Equality operator

Parameters

None	

Returns

bool

6.9.3.7 print()

```
template<typename dtype>
void NumC::Coordinates::Dec< dtype >::print ( ) const  [inline]
```

Prints the Dec object to the console

Parameters

None	

Returns

None

6.9.3.8 radians()

```
template<typename dtype>
dtype NumC::Coordinates::Dec< dtype >::radians ( ) const [inline]
```

Get the radians value

Parameters

None	

Returns

minutes

6.9.3.9 seconds()

```
template<typename dtype>
dtype NumC::Coordinates::Dec< dtype >::seconds ( ) const [inline]
```

Get the seconds value

Parameters

None	

Returns

seconds

6.9.3.10 sign()

```
template<typename dtype>
Sign::Type NumC::Coordinates::Dec< dtype >::sign ( ) const [inline]
```

Get the sign of the degrees (positive or negative)

Parameters

None	

Returns

Sign::Type

6.9.3.11 str()

```
template<typename dtype>
std::string NumC::Coordinates::Dec< dtype >::str ( ) const  [inline]
```

Return the dec object as a string representation

Parameters

None	

Returns

string

6.9.4 Friends And Related Function Documentation

6.9.4.1 operator<<

```
template<typename dtype>
std::ostream& operator<< (
            std::ostream & inStream,
            const Dec< dtype > & inDec )  [friend]
```

Ostream operator

Parameters

None	

Returns

None

The documentation for this class was generated from the following file:

- Coordinates.hpp

6.10 NumC::DtypeInfo< dtype > Class Template Reference

Holds info about the dtype.

`#include <DtypeInfo.hpp>`

Static Public Member Functions

- static constexpr dtype bits ()
- static constexpr dtype epsilon ()
- static constexpr bool isInteger ()
- static constexpr bool isSigned ()
- static constexpr dtype max ()
- static constexpr dtype min ()

6.10.1 Detailed Description

template<typename dtype>
class NumC::DtypeInfo< dtype >

Holds info about the dtype.

6.10.2 Member Function Documentation

6.10.2.1 bits()

```
template<typename dtype >
static constexpr dtype NumC::DtypeInfo< dtype >::bits ( )  [inline], [static]
```

For integer types: number of non-sign bits in the representation. For floating types : number of digits(in radix base) in the mantissa

Parameters

None	

Returns

number of bits

6.10.2.2 epsilon()

```
template<typename dtype >
static constexpr dtype NumC::DtypeInfo< dtype >::epsilon ( ) [inline], [static]
```

Machine epsilon (the difference between 1 and the least value greater than 1 that is representable).

Parameters

None	

Returns

dtype

6.10.2.3 isInteger()

```
template<typename dtype >
static constexpr bool NumC::DtypeInfo< dtype >::isInteger ( ) [inline], [static]
```

True if type is integer.

Parameters

None	

Returns

bool

6.10.2.4 isSigned()

```
template<typename dtype >
static constexpr bool NumC::DtypeInfo< dtype >::isSigned ( ) [inline], [static]
```

True if type is signed.

Parameters

None	

Returns

bool

6.10.2.5 max()

```
template<typename dtype >
static constexpr dtype NumC::DtypeInfo< dtype >::max ( ) [inline], [static]
```

Returns the maximum value of the dtype

Parameters

None	

Returns

max value

6.10.2.6 min()

```
template<typename dtype >
static constexpr dtype NumC::DtypeInfo< dtype >::min ( ) [inline], [static]
```

Returns the minimum value of the dtype

Parameters

None	

Returns

min value

The documentation for this class was generated from the following file:

- DtypeInfo.hpp

6.11 NumC::Endian Struct Reference

Enum for endianess.

```
#include <Types.hpp>
```

Public Types

- enum Type { NATIVE = 0, BIG, LITTLE }

6.11.1 Detailed Description

Enum for endianess.

6.11.2 Member Enumeration Documentation

6.11.2.1 Type

`enum NumC::Endian::Type`

Enumerator

NATIVE	
BIG	
LITTLE	

The documentation for this struct was generated from the following file:

- Types.hpp

6.12 NumC::FFT< dtype > Class Template Reference

Class for performing fast forrier tranforms.

`#include <FFT.hpp>`

6.12.1 Detailed Description

template<typename dtype>
class NumC::FFT< dtype >

Class for performing fast forrier tranforms.

The documentation for this class was generated from the following file:

- FFT.hpp

6.13 NumC::Filters< dtype > Class Template Reference

Class for performing many types of image filtering.

`#include <Filter.hpp>`

Static Public Member Functions

- static NdArray< dtype > complementaryMedianFilter (const NdArray< dtype > &inImageArray, uint32 in↩ Size, Filter::Boundary::Mode inMode=Filter::Boundary::REFLECT, dtype inConstantValue=0)
- static NdArray< dtype > complementaryMedianFilter1d (const NdArray< dtype > &inImageArray, uint32 inSize, Filter::Boundary::Mode inMode=Filter::Boundary::REFLECT, dtype inConstantValue=0)
- static NdArray< dtype > convolve (const NdArray< dtype > &inImageArray, uint32 inSize, const NdArray< dtype > &inWeights, Filter::Boundary::Mode inMode=Filter::Boundary::REFLECT, dtype inConstantValue=0)
- static NdArray< dtype > convolve1d (const NdArray< dtype > &inImageArray, const NdArray< dtype > &inWeights, Filter::Boundary::Mode inMode=Filter::Boundary::REFLECT, dtype inConstantValue=0)
- static NdArray< dtype > gaussianFilter (const NdArray< dtype > &inImageArray, double inSigma, Filter::Boundary::Mode inMode=Filter::Boundary::REFLECT, dtype inConstantValue=0)
- static NdArray< dtype > gaussianFilter1d (const NdArray< dtype > &inImageArray, double inSigma, Filter::Boundary::Mode inMode=Filter::Boundary::REFLECT, dtype inConstantValue=0)
- static NdArray< dtype > maximumFilter (const NdArray< dtype > &inImageArray, uint32 inSize, Filter::Boundary::Mode inMode=Filter::Boundary::REFLECT, dtype inConstantValue=0)
- static NdArray< dtype > maximumFilter1d (const NdArray< dtype > &inImageArray, uint32 inSize, Filter::Boundary::Mode inMode=Filter::Boundary::REFLECT, dtype inConstantValue=0)
- static NdArray< dtype > medianFilter (const NdArray< dtype > &inImageArray, uint32 inSize, Filter::Boundary::Mode inMode=Filter::Boundary::REFLECT, dtype inConstantValue=0)
- static NdArray< dtype > medianFilter1d (const NdArray< dtype > &inImageArray, uint32 inSize, Filter::Boundary::Mode inMode=Filter::Boundary::REFLECT, dtype inConstantValue=0)
- static NdArray< dtype > minimumFilter (const NdArray< dtype > &inImageArray, uint32 inSize, Filter::Boundary::Mode inMode=Filter::Boundary::REFLECT, dtype inConstantValue=0)
- static NdArray< dtype > minumumFilter1d (const NdArray< dtype > &inImageArray, uint32 inSize, Filter::Boundary::Mode inMode=Filter::Boundary::REFLECT, dtype inConstantValue=0)
- static NdArray< dtype > percentileFilter (const NdArray< dtype > &inImageArray, uint32 inSize, uint8 in↩ Percentile, Filter::Boundary::Mode inMode=Filter::Boundary::REFLECT, dtype inConstantValue=0)
- static NdArray< dtype > percentileFilter1d (const NdArray< dtype > &inImageArray, uint32 inSize, uint8 inPercentile, Filter::Boundary::Mode inMode=Filter::Boundary::REFLECT, dtype inConstantValue=0)
- static NdArray< dtype > rankFilter (const NdArray< dtype > &inImageArray, uint32 inSize, uint32 inRank, Filter::Boundary::Mode inMode=Filter::Boundary::REFLECT, dtype inConstantValue=0)
- static NdArray< dtype > rankFilter1d (const NdArray< dtype > &inImageArray, uint32 inSize, uint8 inRank, Filter::Boundary::Mode inMode=Filter::Boundary::REFLECT, dtype inConstantValue=0)
- static NdArray< dtype > uniformFilter (const NdArray< dtype > &inImageArray, uint32 inSize, Filter::Boundary::Mode inMode=Filter::Boundary::REFLECT, dtype inConstantValue=0)
- static NdArray< dtype > uniformFilter1d (const NdArray< dtype > &inImageArray, uint32 inSize, Filter::Boundary::Mode inMode=Filter::Boundary::REFLECT, dtype inConstantValue=0)

6.13.1 Detailed Description

template<typename dtype>
class NumC::Filters< dtype >

Class for performing many types of image filtering.

6.13.2 Member Function Documentation

6.13.2.1 complementaryMedianFilter()

```
template<typename dtype >
static NdArray<dtype> NumC::Filters< dtype >::complementaryMedianFilter (
            const NdArray< dtype > & inImageArray,
            uint32 inSize,
            Filter::Boundary::Mode inMode = Filter::Boundary::REFLECT,
            dtype inConstantValue = 0 )  [inline], [static]
```

Calculates a multidimensional complemenatry median filter.

Parameters

NdArray	
square	size of the kernel to apply
boundary	mode, default Reflect, options (reflect, constant, nearest, mirror, wrap)
contant	value if boundary = 'constant'

Returns

NdArray

6.13.2.2 complementaryMedianFilter1d()

```
template<typename dtype >
static NdArray<dtype> NumC::Filters< dtype >::complementaryMedianFilter1d (
            const NdArray< dtype > & inImageArray,
            uint32 inSize,
            Filter::Boundary::Mode inMode = Filter::Boundary::REFLECT,
            dtype inConstantValue = 0 )  [inline], [static]
```

Calculate a one-dimensional complemenatry median filter.

Parameters

NdArray	
size	of the kernel to apply
boundary	mode, default Reflect, options (reflect, constant, nearest, mirror, wrap)
contant	value if boundary = 'constant'

Returns

NdArray

6.13.2.3 convolve()

```
template<typename dtype >
static NdArray<dtype> NumC::Filters< dtype >::convolve (
            const NdArray< dtype > & inImageArray,
            uint32 inSize,
            const NdArray< dtype > & inWeights,
            Filter::Boundary::Mode inMode = Filter::Boundary::REFLECT,
            dtype inConstantValue = 0 )  [inline], [static]
```

Calculates a multidimensional kernel convolution.

SciPy Reference: https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.convolve.html#scipy.ndimage.convolve

Parameters

NdArray	
square	size of the kernel to apply
NdArray,weights	
boundary	mode, default Reflect, options (reflect, constant, nearest, mirror, wrap)
contant	value if boundary = 'constant'

Returns

NdArray

6.13.2.4 convolve1d()

```
template<typename dtype >
static NdArray<dtype> NumC::Filters< dtype >::convolve1d (
            const NdArray< dtype > & inImageArray,
            const NdArray< dtype > & inWeights,
            Filter::Boundary::Mode inMode = Filter::Boundary::REFLECT,
            dtype inConstantValue = 0 )  [inline], [static]
```

Calculates a one-dimensional kernel convolution.

SciPy Reference: https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.convolve1d.html#scipy.ndimage.convolve1d

Parameters

NdArray	
NdArray,weights	
boundary	mode, default Reflect, options (reflect, constant, nearest, mirror, wrap)
contant	value if boundary = 'constant'

Returns

NdArray

6.13.2.5 gaussianFilter()

```
template<typename dtype >
static NdArray<dtype> NumC::Filters< dtype >::gaussianFilter (
            const NdArray< dtype > & inImageArray,
            double inSigma,
            Filter::Boundary::Mode inMode = Filter::Boundary::REFLECT,
            dtype inConstantValue = 0 ) [inline], [static]
```

Calculates a multidimensional gaussian filter.

SciPy Reference: https://docs.scipy.org/doc/scipy/reference/generated/scipy.↩ ndimage.gaussian_filter.html#scipy.ndimage.gaussian_filter

Parameters

NdArray	
double,Standard	deviation for Gaussian kernel
boundary	mode, default Reflect, options (reflect, constant, nearest, mirror, wrap)
contant	value if boundary = 'constant'

Returns

NdArray

6.13.2.6 gaussianFilter1d()

```
template<typename dtype >
static NdArray<dtype> NumC::Filters< dtype >::gaussianFilter1d (
            const NdArray< dtype > & inImageArray,
            double inSigma,
            Filter::Boundary::Mode inMode = Filter::Boundary::REFLECT,
            dtype inConstantValue = 0 ) [inline], [static]
```

Calculate a one-dimensional gaussian filter.

SciPy Reference: https://docs.scipy.org/doc/scipy/reference/generated/scipy.↩ ndimage.generic_filter1d.html#scipy.ndimage.generic_filter1d

Parameters

NdArray	
double,Standard	deviation for Gaussian kernel
boundary	mode, default Reflect, options (reflect, constant, nearest, mirror, wrap)
contant	value if boundary = 'constant'

Returns

NdArray

6.13.2.7 maximumFilter()

```
template<typename dtype >
static NdArray<dtype> NumC::Filters< dtype >::maximumFilter (
            const NdArray< dtype > & inImageArray,
            uint32 inSize,
            Filter::Boundary::Mode inMode = Filter::Boundary::REFLECT,
            dtype inConstantValue = 0 ) [inline], [static]
```

Calculates a multidimensional maximum filter.

SciPy Reference: https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.maximum_filter.html#scipy.ndimage.maximum_filter

Parameters

NdArray	
square	size of the kernel to apply
boundary	mode, default Reflect, options (reflect, constant, nearest, mirror, wrap)
contant	value if boundary = 'constant'

Returns

NdArray

6.13.2.8 maximumFilter1d()

```
template<typename dtype >
static NdArray<dtype> NumC::Filters< dtype >::maximumFilter1d (
            const NdArray< dtype > & inImageArray,
            uint32 inSize,
            Filter::Boundary::Mode inMode = Filter::Boundary::REFLECT,
            dtype inConstantValue = 0 ) [inline], [static]
```

Calculates a one-dimensional maximum filter.

SciPy Reference: https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.maximum_filter1d.html#scipy.ndimage.maximum_filter1d

Parameters

NdArray	
size	of the kernel to apply
boundary	mode, default Reflect, options (reflect, constant, nearest, mirror, wrap)
contant	value if boundary = 'constant'

Returns

NdArray

6.13.2.9 medianFilter()

```
template<typename dtype >
static NdArray<dtype> NumC::Filters< dtype >::medianFilter (
            const NdArray< dtype > & inImageArray,
            uint32 inSize,
            Filter::Boundary::Mode inMode = Filter::Boundary::REFLECT,
            dtype inConstantValue = 0 ) [inline], [static]
```

Calculates a multidimensional median filter.

SciPy Reference: https://docs.scipy.org/doc/scipy/reference/generated/scipy.↩
ndimage.median_filter.html#scipy.ndimage.median_filter

Parameters

NdArray	
square	size of the kernel to apply
boundary	mode, default Reflect, options (reflect, constant, nearest, mirror, wrap)
contant	value if boundary = 'constant'

Returns

NdArray

6.13.2.10 medianFilter1d()

```
template<typename dtype >
static NdArray<dtype> NumC::Filters< dtype >::medianFilter1d (
            const NdArray< dtype > & inImageArray,
            uint32 inSize,
            Filter::Boundary::Mode inMode = Filter::Boundary::REFLECT,
            dtype inConstantValue = 0 ) [inline], [static]
```

Calculates a one-dimensional median filter.

SciPy Reference: https://docs.scipy.org/doc/scipy/reference/generated/scipy.↩
ndimage.median_filter.html#scipy.ndimage.median_filter

Parameters

NdArray	
linear	size of the kernel to apply
boundary	mode, default Reflect, options (reflect, constant, nearest, mirror, wrap)
contant	value if boundary = 'constant'

Returns

NdArray

6.13.2.11 minimumFilter()

```
template<typename dtype >
static NdArray<dtype> NumC::Filters< dtype >::minimumFilter (
            const NdArray< dtype > & inImageArray,
            uint32 inSize,
            Filter::Boundary::Mode inMode = Filter::Boundary::REFLECT,
            dtype inConstantValue = 0 )  [inline], [static]
```

Calculates a multidimensional minimum filter.

SciPy Reference: https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.minimum_filter.html#scipy.ndimage.minimum_filter

Parameters

NdArray	
square	size of the kernel to apply
boundary	mode, default Reflect, options (reflect, constant, nearest, mirror, wrap)
contant	value if boundary = 'constant'

Returns

NdArray

6.13.2.12 minumumFilter1d()

```
template<typename dtype >
static NdArray<dtype> NumC::Filters< dtype >::minumumFilter1d (
            const NdArray< dtype > & inImageArray,
            uint32 inSize,
            Filter::Boundary::Mode inMode = Filter::Boundary::REFLECT,
            dtype inConstantValue = 0 )  [inline], [static]
```

Calculates a one-dimensional minumum filter.

SciPy Reference: https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.minimum_filter1d.html#scipy.ndimage.minimum_filter1d

Parameters

NdArray	
size	of the kernel to apply
boundary	mode, default Reflect, options (reflect, constant, nearest, mirror, wrap)
contant	value if boundary = 'constant'

Returns

NdArray

6.13.2.13 percentileFilter()

```
template<typename dtype >
static NdArray<dtype> NumC::Filters< dtype >::percentileFilter (
            const NdArray< dtype > & inImageArray,
            uint32 inSize,
            uint8 inPercentile,
            Filter::Boundary::Mode inMode = Filter::Boundary::REFLECT,
            dtype inConstantValue = 0 ) [inline], [static]
```

Calculates a multidimensional percentile filter.

SciPy Reference: https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.percentile_filter.html#scipy.ndimage.percentile_filter

Parameters

NdArray	
square	size of the kernel to apply
percentile	[0, 100]
boundary	mode, default Reflect, options (reflect, constant, nearest, mirror, wrap)
contant	value if boundary = 'constant'

Returns

NdArray

6.13.2.14 percentileFilter1d()

```
template<typename dtype >
static NdArray<dtype> NumC::Filters< dtype >::percentileFilter1d (
            const NdArray< dtype > & inImageArray,
            uint32 inSize,
            uint8 inPercentile,
            Filter::Boundary::Mode inMode = Filter::Boundary::REFLECT,
            dtype inConstantValue = 0 ) [inline], [static]
```

Calculates a one-dimensional percentile filter.

SciPy Reference: https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.percentile_filter.html#scipy.ndimage.percentile_filter

Parameters

NdArray	
size	of the kernel to apply
percentile	[0, 100]
boundary	mode, default Reflect, options (reflect, constant, nearest, mirror, wrap)
contant	value if boundary = 'constant'

Returns

NdArray

6.13.2.15 rankFilter()

```
template<typename dtype >
static NdArray<dtype> NumC::Filters< dtype >::rankFilter (
            const NdArray< dtype > & inImageArray,
            uint32 inSize,
            uint32 inRank,
            Filter::Boundary::Mode inMode = Filter::Boundary::REFLECT,
            dtype inConstantValue = 0 ) [inline], [static]
```

Calculates a multidimensional rank filter.

SciPy Reference: https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.rank_filter.html#scipy.ndimage.rank_filter

Parameters

NdArray	
square	size of the kernel to apply
rank	[0, inSize^2 - 1]
boundary	mode, default Reflect, options (reflect, constant, nearest, mirror, wrap)
contant	value if boundary = 'constant'

Returns

NdArray

6.13.2.16 rankFilter1d()

```
template<typename dtype >
static NdArray<dtype> NumC::Filters< dtype >::rankFilter1d (
            const NdArray< dtype > & inImageArray,
```

```
            uint32 inSize,
            uint8 inRank,
            Filter::Boundary::Mode inMode = Filter::Boundary::REFLECT,
            dtype inConstantValue = 0 )  [inline], [static]
```

Calculates a one-dimensional rank filter.

SciPy Reference: https://docs.scipy.org/doc/scipy/reference/generated/scipy.↩ndimage.rank_filter.html#scipy.ndimage.rank_filter

Parameters

NdArray	
size	of the kernel to apply
rank	[0, 100]
boundary	mode, default Reflect, options (reflect, constant, nearest, mirror, wrap)
contant	value if boundary = 'constant'

Returns

NdArray

6.13.2.17 uniformFilter()

```
template<typename dtype >
static NdArray<dtype> NumC::Filters< dtype >::uniformFilter (
            const NdArray< dtype > & inImageArray,
            uint32 inSize,
            Filter::Boundary::Mode inMode = Filter::Boundary::REFLECT,
            dtype inConstantValue = 0 )  [inline], [static]
```

Calculates a multidimensional uniform filter.

SciPy Reference: https://docs.scipy.org/doc/scipy/reference/generated/scipy.↩ndimage.uniform_filter.html#scipy.ndimage.uniform_filter

Parameters

NdArray	
square	size of the kernel to apply
boundary	mode, default Reflect, options (reflect, constant, nearest, mirror, wrap)
contant	value if boundary = 'constant'

Returns

NdArray

6.13.2.18 uniformFilter1d()

```
template<typename dtype >
static NdArray<dtype> NumC::Filters< dtype >::uniformFilter1d (
            const NdArray< dtype > & inImageArray,
            uint32 inSize,
            Filter::Boundary::Mode inMode = Filter::Boundary::REFLECT,
            dtype inConstantValue = 0 )  [inline], [static]
```

Calculates a one-dimensional uniform filter.

SciPy Reference: https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.uniform_filter1d.html#scipy.ndimage.uniform_filter1d

Parameters

NdArray	
size	of the kernel to apply
boundary	mode, default Reflect, options (reflect, constant, nearest, mirror, wrap)
contant	value if boundary = 'constant'

Returns

NdArray

The documentation for this class was generated from the following file:

- Filter.hpp

6.14 NumC::ImageProcessing< dtype > Class Template Reference

Class for basic image processing.

```
#include <ImageProcessing.hpp>
```

Data Structures

- class Centroid

  holds the information for a centroid
- class Cluster

  Holds the information for a cluster of pixels.
- class Pixel

  Holds the information for a single pixel.

Static Public Member Functions

- static NdArray< bool > applyThreshold (const NdArray< dtype > &inImageArray, dtype inThreshold)
- static std::vector< Centroid > centroidClusters (const std::vector< Cluster > &inClusters)
- static std::vector< Cluster > clusterPixels (const NdArray< dtype > &inImageArray, const NdArray< bool > &inExceedances, uint8 inBorderWidth=0)
- static std::vector< Centroid > generateCentroids (const NdArray< dtype > &inImageArray, double inRate, const std::string inWindowType, uint8 inBorderWidth=0)
- static dtype generateThreshold (const NdArray< dtype > &inImageArray, double inRate)
- static NdArray< bool > windowExceedances (const NdArray< bool > &inExceedances, uint8 inBorderWidth)

6.14.1 Detailed Description

template<typename dtype>
class NumC::ImageProcessing< dtype >

Class for basic image processing.

6.14.2 Member Function Documentation

6.14.2.1 applyThreshold()

```
template<typename dtype >
static NdArray<bool> NumC::ImageProcessing< dtype >::applyThreshold (
            const NdArray< dtype > & inImageArray,
            dtype inThreshold )  [inline], [static]
```

Applies a threshold to an image

Parameters

NdArray	threshold value

Returns

NdArray of booleans of pixels that exceeded the threshold

6.14.2.2 centroidClusters()

```
template<typename dtype >
static std::vector<Centroid> NumC::ImageProcessing< dtype >::centroidClusters (
            const std::vector< Cluster > & inClusters )  [inline], [static]
```

Center of Mass centroids clusters

Parameters

NdArray	
threshold	value

Returns

std::vector<Centroid>

6.14.2.3 clusterPixels()

```
template<typename dtype >
static std::vector<Cluster> NumC::ImageProcessing< dtype >::clusterPixels (
            const NdArray< dtype > & inImageArray,
            const NdArray< bool > & inExceedances,
            uint8 inBorderWidth = 0 )  [inline], [static]
```

Clusters exceedance pixels from an image

Parameters

NdArray	
NdArray	of exceedances
border	to apply around exceedance pixels post clustering, default 0

Returns

std::vector<Cluster>

6.14.2.4 generateCentroids()

```
template<typename dtype >
static std::vector<Centroid> NumC::ImageProcessing< dtype >::generateCentroids (
            const NdArray< dtype > & inImageArray,
            double inRate,
            const std::string inWindowType,
            uint8 inBorderWidth = 0 )  [inline], [static]
```

Generates a list of centroids givin an input exceedance rate

Parameters

NdArray	
exceedance	rate
string	"pre", or "post" for where to apply the exceedance windowing
border	to apply, default 0

Returns

std::vector<Centroid>

6.14.2.5 generateThreshold()

```
template<typename dtype >
static dtype NumC::ImageProcessing< dtype >::generateThreshold (
            const NdArray< dtype > & inImageArray,
            double inRate )  [inline], [static]
```

Calculates a threshold such that the input rate of pixels exceeds the threshold. Really should only be used for integer input array values. If using floating point data, user beware...

Parameters

NdArray	
exceedance	rate

Returns

dtype

6.14.2.6 windowExceedances()

```
template<typename dtype >
static NdArray<bool> NumC::ImageProcessing< dtype >::windowExceedances (
            const NdArray< bool > & inExceedances,
            uint8 inBorderWidth )  [inline], [static]
```

Window expand around exceedance pixels

Parameters

NdArray<bool>	
border	width

Returns

NdArray<bool>

The documentation for this class was generated from the following file:

- ImageProcessing.hpp

6.15 NumC::Linalg< dtype > Class Template Reference

Class for doing linear algebra operations.

```
#include <Linalg.hpp>
```

Static Public Member Functions

- static dtype det (const NdArray< dtype > &inArray)
- static NdArray< dtype > hat (dtype inX, dtype inY, dtype inZ)
- static NdArray< dtype > hat (const NdArray< dtype > &inVec)
- static NdArray< double > inv (const NdArray< dtype > &inArray)
- static NdArray< double > lstsq (const NdArray< dtype > &inA, const NdArray< dtype > &inB, double in↩ Tolerance=1.e-12)
- template<typename dtypeOut = double>
  static NdArray< dtypeOut > matrix_power (const NdArray< dtype > &inArray, int16 inPower)
- template<typename dtypeOut = double>
  static NdArray< dtypeOut > multi_dot (const std::initializer_list< NdArray< dtype > > &inList)
- static void svd (const NdArray< dtype > &inArray, NdArray< double > &outU, NdArray< double > &outS, NdArray< double > &outVt)

6.15.1 Detailed Description

template<typename dtype>
class NumC::Linalg< dtype >

Class for doing linear algebra operations.

6.15.2 Member Function Documentation

6.15.2.1 det()

```
template<typename dtype>
static dtype NumC::Linalg< dtype >::det (
            const NdArray< dtype > & inArray )  [inline], [static]
```

matrix determinant. NOTE: can get verrrrry slow for large matrices (order > 10)

SciPy Reference: https://docs.scipy.org/doc/scipy/reference/generated/scipy.↩ linalg.det.html#scipy.linalg.det

Parameters

NdArray	

Returns

dtype

6.15.2.2 hat() [1/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Linalg< dtype >::hat (
            dtype inX,
            dtype inY,
            dtype inZ )  [inline], [static]
```

vector hat operator

Parameters

x	
y	
z	

Returns

3x3 NdArray

6.15.2.3 hat() [2/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Linalg< dtype >::hat (
            const NdArray< dtype > & inVec )  [inline], [static]
```

vector hat operator

Parameters

NdArray	3x1, or 1x3 cartesian vector

Returns

3x3 NdArray

6.15.2.4 inv()

```
template<typename dtype>
static NdArray<double> NumC::Linalg< dtype >::inv (
            const NdArray< dtype > & inArray )  [inline], [static]
```

matrix inverse

SciPy Reference: https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.inv.html#scipy.linalg.inv

Parameters

NdArray	

Returns

NdArray

6.15.2.5 lstsq()

```
template<typename dtype>
static NdArray<double> NumC::Linalg< dtype >::lstsq (
            const NdArray< dtype > & inA,
            const NdArray< dtype > & inB,
            double inTolerance = 1.e-12 ) [inline], [static]
```

Solves the equation a x = b by computing a vector x that minimizes the Euclidean 2-norm || b - a x ||^2. The equation may be under-, well-, or over- determined (i.e., the number of linearly independent rows of a can be less than, equal to, or greater than its number of linearly independent columns). If a is square and of full rank, then x (but for round-off error) is the "exact" solution of the equation.

SciPy Reference: https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.lstsq.html#scipy.linalg.lstsq

Parameters

NdArray,coefficient	matrix
NdArray,Ordinate	or "dependent variable" values
double,tolerance	

Returns

NdArray

6.15.2.6 matrix_power()

```
template<typename dtype>
template<typename dtypeOut = double>
static NdArray<dtypeOut> NumC::Linalg< dtype >::matrix_power (
            const NdArray< dtype > & inArray,
            int16 inPower ) [inline], [static]
```

Raise a square matrix to the (integer) power n.

For positive integers n, the power is computed by repeated matrix squarings and matrix multiplications. If n == 0, the identity matrix of the same shape as M is returned. If n < 0, the inverse is computed and then raised to the abs(n).

NumPy Reference: https://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.matrix_power.html#numpy.linalg.matrix_power

Parameters

NdArray	
power	

Returns

NdArray

6.15.2.7 multi_dot()

```
template<typename dtype>
template<typename dtypeOut = double>
static NdArray<dtypeOut> NumC::Linalg< dtype >::multi_dot (
            const std::initializer_list< NdArray< dtype > > & inList ) [inline], [static]
```

Compute the dot product of two or more arrays in a single function call..

NumPy Reference: https://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.multi_dot.html#numpy.linalg.multi_dot

Parameters

initializer_list<NdArray<dtype>	>, list of arrays

Returns

NdArray

6.15.2.8 svd()

```
template<typename dtype>
static void NumC::Linalg< dtype >::svd (
            const NdArray< dtype > & inArray,
            NdArray< double > & outU,
            NdArray< double > & outS,
            NdArray< double > & outVt ) [inline], [static]
```

matrix svd

NumPy Reference: https://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.svd.html#numpy.linalg.svd

Parameters

NdArray	to be SVDed
NdArray	output U
NdArray	output S
NdArray	output V transpose

Returns

NdArray

The documentation for this class was generated from the following file:

- Linalg.hpp

6.16 NumC::Methods< dtype > Class Template Reference

Methods for working with NdArrays.

```
#include <Methods.hpp>
```

Static Public Member Functions

- static dtype abs (dtype inValue)
- static NdArray< dtype > abs (const NdArray< dtype > &inArray)
- template<typename dtypeOut = double>
  static NdArray< dtypeOut > add (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- static uint32 alen (const NdArray< dtype > &inArray)
- static NdArray< bool > all (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- static bool allclose (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2, double in←Tolerance=1e-5)
- static NdArray< dtype > amax (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- static NdArray< dtype > amin (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- static NdArray< bool > any (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- static NdArray< dtype > append (const NdArray< dtype > &inArray, const NdArray< dtype > &inAppend←Values, Axis::Type inAxis=Axis::NONE)
- static NdArray< dtype > arange (dtype inStart, dtype inStop, dtype inStep=1)
- static NdArray< dtype > arange (dtype inStop)
- static double arccos (dtype inValue)
- static NdArray< double > arccos (const NdArray< dtype > &inArray)
- static double arccosh (dtype inValue)
- static NdArray< double > arccosh (const NdArray< dtype > &inArray)
- static double arcsin (dtype inValue)
- static NdArray< double > arcsin (const NdArray< dtype > &inArray)
- static double arcsinh (dtype inValue)
- static NdArray< double > arcsinh (const NdArray< dtype > &inArray)
- static double arctan (dtype inValue)
- static NdArray< double > arctan (const NdArray< dtype > &inArray)
- static double arctan2 (dtype inY, dtype inX)
- static NdArray< double > arctan2 (const NdArray< dtype > &inY, const NdArray< dtype > &inX)

- static double arctanh (dtype inValue)
- static NdArray< double > arctanh (const NdArray< dtype > &inArray)
- static NdArray< uint32 > argmax (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- static NdArray< uint32 > argmin (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- static NdArray< uint32 > argsort (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- static NdArray< uint32 > argwhere (const NdArray< dtype > &inArray)
- static dtype around (dtype inValue, uint8 inNumDecimals=0)
- static NdArray< dtype > around (const NdArray< dtype > &inArray, uint8 inNumDecimals=0)
- static bool array_equal (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- static bool array_equiv (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- static NdArray< dtype > asarray (const std::vector< dtype > &inVector)
- static NdArray< dtype > asarray (std::initializer_list< dtype > &inList)
- template<typename dtypeOut = double>
  static NdArray< dtypeOut > astype (const NdArray< dtype > inArray)
- static NdArray< double > average (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- static NdArray< double > average (const NdArray< dtype > &inArray, const NdArray< dtype > &inWeights, Axis::Type inAxis=Axis::NONE)
- static NdArray< dtype > bincount (const NdArray< dtype > &inArray, uint16 inMinLength=0)
- static NdArray< dtype > bincount (const NdArray< dtype > &inArray, const NdArray< dtype > &inWeights, uint16 inMinLength=0)
- static NdArray< dtype > bitwise_and (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- static NdArray< dtype > bitwise_not (const NdArray< dtype > &inArray)
- static NdArray< dtype > bitwise_or (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- static NdArray< dtype > bitwise_xor (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- static NdArray< dtype > byteswap (const NdArray< dtype > &inArray)
- static double cbrt (dtype inValue)
- static NdArray< double > cbrt (const NdArray< dtype > &inArray)
- static dtype ceil (dtype inValue)
- static NdArray< dtype > ceil (const NdArray< dtype > &inArray)
- static dtype clip (dtype inValue, dtype inMinValue, dtype inMaxValue)
- static NdArray< dtype > clip (const NdArray< dtype > &inArray, dtype inMinValue, dtype inMaxValue)
- static NdArray< dtype > column_stack (const std::initializer_list< NdArray< dtype > > &inArrayList)
- static NdArray< dtype > concatenate (const std::initializer_list< NdArray< dtype > > &inArrayList, Axis::Type inAxis=Axis::NONE)
- static NdArray< bool > contains (const NdArray< dtype > &inArray, dtype inValue, Axis::Type inAxis=Axis::NONE)
- static NdArray< dtype > copy (const NdArray< dtype > &inArray)
- static NdArray< dtype > copySign (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- static NdArray< dtype > & copyto (NdArray< dtype > &inDestArray, const NdArray< dtype > &inSrcArray)
- static double cos (dtype inValue)
- static NdArray< double > cos (const NdArray< dtype > &inArray)
- static double cosh (dtype inValue)
- static NdArray< double > cosh (const NdArray< dtype > &inArray)
- static NdArray< uint32 > count_nonzero (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- template<typename dtypeOut = double>
  static NdArray< dtypeOut > cross (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2, Axis::Type inAxis=Axis::NONE)
- template<typename dtypeOut = double>
  static NdArray< dtypeOut > cube (const NdArray< dtype > &inArray)
- template<typename dtypeOut = double>
  static NdArray< dtypeOut > cumprod (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- template<typename dtypeOut = double>
  static NdArray< dtypeOut > cumsum (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)

- static double deg2rad (dtype inValue)
- static NdArray< double > deg2rad (const NdArray< dtype > &inArray)
- static NdArray< dtype > deleteIndices (const NdArray< dtype > &inArray, const NdArray< uint32 > &inArrayIdxs, Axis::Type inAxis=Axis::NONE)
- static NdArray< dtype > deleteIndices (const NdArray< dtype > &inArray, const Slice &inIndicesSlice, Axis::Type inAxis=Axis::NONE)
- static NdArray< dtype > deleteIndices (const NdArray< dtype > &inArray, uint32 inIndex, Axis::Type inAxis=Axis::NONE)
- static NdArray< dtype > diagflat (const NdArray< dtype > &inArray)
- static NdArray< dtype > diagonal (const NdArray< dtype > &inArray, uint32 inOffset=0, Axis::Type inAxis=Axis::ROW)
- static NdArray< dtype > diff (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- template<typename dtypeOut = double>
  static NdArray< dtypeOut > divide (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- template<typename dtypeOut = double>
  static NdArray< dtypeOut > dot (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- static void dump (const NdArray< dtype > &inArray, const std::string &inFilename)
- static NdArray< dtype > empty (uint32 inNumRows, uint32 inNumCols)
- static NdArray< dtype > empty (const Shape &inShape)
- template<typename dtypeOut = double>
  static NdArray< dtypeOut > empty_like (const NdArray< dtype > &inArray)
- static Endian::Type endianess (const NdArray< dtype > &inArray)
- static NdArray< bool > equal (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- static double exp (dtype inValue)
- static NdArray< double > exp (const NdArray< dtype > &inArray)
- static double exp2 (dtype inValue)
- static NdArray< double > exp2 (const NdArray< dtype > &inArray)
- static double expm1 (dtype inValue)
- static NdArray< double > expm1 (const NdArray< dtype > &inArray)
- static NdArray< dtype > eye (uint32 inN, int32 inK=0)
- static NdArray< dtype > eye (uint32 inN, uint32 inM, int32 inK=0)
- static NdArray< dtype > eye (const Shape &inShape, int32 inK=0)
- static dtype fix (dtype inValue)
- static NdArray< dtype > fix (const NdArray< dtype > &inArray)
- static NdArray< uint32 > flatnonzero (const NdArray< dtype > &inArray)
- static NdArray< dtype > flatten (const NdArray< dtype > &inArray)
- static NdArray< dtype > flip (const NdArray< dtype > &inArray, Axis::Type inAxis)
- static NdArray< dtype > fliplr (const NdArray< dtype > &inArray)
- static NdArray< dtype > flipud (const NdArray< dtype > &inArray)
- static dtype floor (dtype inValue)
- static NdArray< dtype > floor (const NdArray< dtype > &inArray)
- static dtype floor_divide (dtype inValue1, dtype inValue2)
- static NdArray< dtype > floor_divide (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- static dtype fmax (dtype inValue1, dtype inValue2)
- static NdArray< dtype > fmax (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- static dtype fmin (dtype inValue1, dtype inValue2)
- static NdArray< dtype > fmin (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- static dtype fmod (dtype inValue1, dtype inValue2)
- static NdArray< dtype > fmod (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- static NdArray< dtype > fromfile (const std::string &inFilename, const std::string &inSep="")
- static NdArray< dtype > full (uint32 inSquareSize, dtype inFillValue)
- static NdArray< dtype > full (uint32 inNumRows, uint32 inNumCols, dtype inFillValue)
- static NdArray< dtype > full (const Shape &inShape, dtype inFillValue)

- template<typename dtypeOut = double>
  static NdArray< dtypeOut > full_like (const NdArray< dtype > &inArray, dtype inFillValue)
- static NdArray< bool > greater (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- static NdArray< bool > greater_equal (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- static std::pair< NdArray< uint32 >, NdArray< double > > histogram (const NdArray< dtype > &inArray, uint32 inNumBins=10)
- static NdArray< dtype > hstack (const std::initializer_list< NdArray< dtype > > &inArrayList)
- template<typename dtypeOut = double>
  static dtypeOut hypot (dtype inValue1, dtype inValue2)
- template<typename dtypeOut = double>
  static NdArray< dtypeOut > hypot (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- static NdArray< dtype > identity (uint32 inSquareSize)
- static NdArray< dtype > intersect1d (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- static NdArray< dtype > invert (const NdArray< dtype > &inArray)
- static NdArray< bool > isclose (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2, double inRtol=1e-05, double inAtol=1e-08)
- static bool isnan (dtype inValue)
- static NdArray< bool > isnan (const NdArray< dtype > &inArray)
- static dtype ldexp (dtype inValue1, uint8 inValue2)
- static NdArray< dtype > ldexp (const NdArray< dtype > &inArray1, const NdArray< uint8 > &inArray2)
- static NdArray< dtype > left_shift (const NdArray< dtype > &inArray, uint8 inNumBits)
- static NdArray< bool > less (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- static NdArray< bool > less_equal (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- static NdArray< dtype > linspace (dtype inStart, dtype inStop, uint32 inNum=50, bool endPoint=true)
- static NdArray< dtype > load (const std::string &inFilename)
- static double log (dtype inValue)
- static NdArray< double > log (const NdArray< dtype > &inArray)
- static double log10 (dtype inValue)
- static NdArray< double > log10 (const NdArray< dtype > &inArray)
- static double log1p (dtype inValue)
- static NdArray< double > log1p (const NdArray< dtype > &inArray)
- static double log2 (dtype inValue)
- static NdArray< double > log2 (const NdArray< dtype > &inArray)
- static NdArray< bool > logical_and (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- static NdArray< bool > logical_not (const NdArray< dtype > &inArray)
- static NdArray< bool > logical_or (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- static NdArray< bool > logical_xor (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- template<typename dtypeOut = double>
  static NdArray< dtypeOut > matmul (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- static NdArray< dtype > max (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- static NdArray< dtype > maximum (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- static NdArray< double > mean (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- static NdArray< dtype > median (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- static NdArray< dtype > min (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- static NdArray< dtype > minimum (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- static NdArray< dtype > mod (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- static NdArray< dtype > multiply (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- static NdArray< uint32 > nanargmax (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- static NdArray< uint32 > nanargmin (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- template<typename dtypeOut = double>
  static NdArray< dtypeOut > nancumprod (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)

- template<typename dtypeOut = double>
  static NdArray< dtypeOut > nancumsum (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- static NdArray< dtype > nanmax (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- static NdArray< double > nanmean (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- static NdArray< dtype > nanmedian (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- static NdArray< dtype > nanmin (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- template<typename dtypeOut = double>
  static NdArray< double > nanpercentile (const NdArray< dtype > &inArray, double inPercentile, Axis::Type inAxis=Axis::NONE, const std::string &inInterpMethod="linear")
- template<typename dtypeOut = double>
  static NdArray< dtypeOut > nanprod (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- static NdArray< dtype > nans (uint32 inSquareSize)
- static NdArray< dtype > nans (uint32 inNumRows, uint32 inNumCols)
- static NumC::NdArray< dtype > nans (const NumC::Shape &inShape)
- static NdArray< double > nanstd (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- template<typename dtypeOut = double>
  static NdArray< dtypeOut > nansum (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- static NdArray< double > nanvar (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- static uint64 nbytes (const NdArray< dtype > &inArray)
- template<typename dtypeOut = double>
  static NdArray< dtypeOut > negative (const NdArray< dtype > &inArray)
- static dtype newbyteorder (dtype inValue, Endian::Type inEndianess)
- static NdArray< dtype > newbyteorder (const NdArray< dtype > &inArray, Endian::Type inEndianess)
- static NdArray< uint32 > nonzero (const NdArray< dtype > &inArray)
- template<typename dtypeOut = double>
  static NdArray< dtypeOut > norm (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- static NdArray< bool > not_equal (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- static NdArray< dtype > ones (uint32 inSquareSize)
- static NdArray< dtype > ones (uint32 inNumRows, uint32 inNumCols)
- static NdArray< dtype > ones (const Shape &inShape)
- template<typename dtypeOut = double>
  static NdArray< dtypeOut > ones_like (const NdArray< dtype > &inArray)
- static NdArray< dtype > pad (const NdArray< dtype > &inArray, uint16 inPadWidth, dtype inPadValue)
- static NdArray< dtype > partition (const NdArray< dtype > &inArray, uint32 inKth, Axis::Type inAxis=Axis::NONE)
- template<typename dtypeOut = double>
  static NdArray< dtypeOut > percentile (const NdArray< dtype > &inArray, double inPercentile, Axis::Type inAxis=Axis::NONE, const std::string &inInterpMethod="linear")
- template<typename dtypeOut = double>
  static NdArray< dtypeOut > power (const NdArray< dtype > &inArray, uint8 inExponent)
- template<typename dtypeOut = double>
  static NdArray< dtypeOut > power (const NdArray< dtype > &inArray, const NdArray< uint8 > &inExponents)
- static void print (const NdArray< dtype > &inArray)
- template<typename dtypeOut = double>
  static NdArray< dtypeOut > prod (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- static NdArray< dtype > ptp (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- static NdArray< dtype > & put (NdArray< dtype > &inArray, const NdArray< uint32 > &inIndices, const NdArray< dtype > &inValues)
- static NdArray< dtype > & putmask (NdArray< dtype > &inArray, const NdArray< bool > &inMask, dtype inValue)
- static NdArray< dtype > & putmask (NdArray< dtype > &inArray, const NdArray< bool > &inMask, const NdArray< dtype > &inValues)
- static double rad2deg (dtype inValue)
- static NdArray< double > rad2deg (const NdArray< dtype > &inArray)

- template< typename dtypeOut = double >
  static NdArray< dtypeOut > reciprocal (const NdArray< dtype > &inArray)
- template< typename dtypeOut = double >
  static dtypeOut remainder (dtype inValue1, dtype inValue2)
- template< typename dtypeOut = double >
  static NdArray< dtypeOut > remainder (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- static NdArray< dtype > repeat (const NdArray< dtype > &inArray, uint32 inNumRows, uint32 inNumCols)
- static NdArray< dtype > repeat (const NdArray< dtype > &inArray, const Shape &inRepeatShape)
- static NdArray< dtype > & reshape (NdArray< dtype > &inArray, uint32 inNumRows, uint32 inNumCols)
- static NdArray< dtype > & reshape (NdArray< dtype > &inArray, const Shape &inNewShape)
- static NdArray< dtype > & resizeFast (NdArray< dtype > &inArray, uint32 inNumRows, uint32 inNumCols)
- static NdArray< dtype > & resizeFast (NdArray< dtype > &inArray, const Shape &inNewShape)
- static NdArray< dtype > & resizeSlow (NdArray< dtype > &inArray, uint32 inNumRows, uint32 inNumCols)
- static NdArray< dtype > & resizeSlow (NdArray< dtype > &inArray, const Shape &inNewShape)
- static NdArray< dtype > right_shift (const NdArray< dtype > &inArray, uint8 inNumBits)
- static dtype rint (dtype inValue)
- static NdArray< dtype > rint (const NdArray< dtype > &inArray)
- static NdArray< dtype > roll (const NdArray< dtype > &inArray, int32 inShift, Axis::Type inAxis=Axis::NONE)
- static NdArray< dtype > rot90 (const NdArray< dtype > &inArray, uint8 inK=1)
- static dtype round (dtype inValue, uint8 inDecimals)
- static NdArray< dtype > round (const NdArray< dtype > &inArray, uint8 inDecimals)
- static NdArray< dtype > row_stack (const std::initializer_list< NdArray< dtype > > &inArrayList)
- static NdArray< dtype > setdiff1d (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- static Shape shape (const NdArray< dtype > &inArray)
- static int8 sign (dtype inValue)
- static NdArray< int8 > sign (const NdArray< dtype > &inArray)
- static bool signbit (dtype inValue)
- static NdArray< bool > signbit (const NdArray< dtype > &inArray)
- static double sin (dtype inValue)
- static NdArray< double > sin (const NdArray< dtype > &inArray)
- static double sinc (dtype inValue)
- static NdArray< double > sinc (const NdArray< dtype > &inArray)
- static double sinh (dtype inValue)
- static NdArray< double > sinh (const NdArray< dtype > &inArray)
- static uint32 size (const NdArray< dtype > &inArray)
- static NdArray< dtype > sort (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- static double sqrt (dtype inValue)
- static NdArray< double > sqrt (const NdArray< dtype > &inArray)
- static dtype square (dtype inValue)
- static NdArray< dtype > square (const NdArray< dtype > &inArray)
- static NdArray< double > std (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- template< typename dtypeOut = double >
  static NdArray< dtypeOut > sum (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- static NdArray< dtype > swapaxes (const NdArray< dtype > &inArray)
- static double tan (dtype inValue)
- static NdArray< double > tan (const NdArray< dtype > &inArray)
- static double tanh (dtype inValue)
- static NdArray< double > tanh (const NdArray< dtype > &inArray)
- static NdArray< dtype > tile (const NdArray< dtype > &inArray, uint32 inNumRows, uint32 inNumCols)
- static NdArray< dtype > tile (const NdArray< dtype > &inArray, const Shape &inReps)
- static void tofile (const NdArray< dtype > &inArray, const std::string &inFilename, const std::string &inSep="")
- static std::vector< dtype > toStlVector (const NdArray< dtype > &inArray)
- template< typename dtypeOut = double >
  static dtypeOut trace (const NdArray< dtype > &inArray, uint16 inOffset=0, Axis::Type inAxis=Axis::ROW)

- static NdArray< dtype > transpose (const NdArray< dtype > &inArray)
- static NdArray< double > trapz (const NdArray< dtype > &inArray, double dx=1.0, Axis::Type in↩ Axis=Axis::NONE)
- static NdArray< double > trapz (const NdArray< dtype > &inArrayY, const NdArray< dtype > &inArrayX, Axis::Type inAxis=Axis::NONE)
- static NdArray< dtype > tri (uint32 inN, int32 inOffset=0)
- static NdArray< dtype > tri (uint32 inN, uint32 inM, int32 inOffset=0)
- static NdArray< dtype > trim_zeros (const NdArray< dtype > &inArray, const std::string inTrim="fb")
- static dtype trunc (dtype inValue)
- static NdArray< dtype > trunc (const NdArray< dtype > &inArray)
- static NdArray< dtype > union1d (const NdArray< dtype > &inArray1, const NdArray< dtype > &inArray2)
- static NdArray< dtype > unique (const NdArray< dtype > &inArray)
- static dtype unwrap (dtype inValue)
- static NdArray< dtype > unwrap (const NdArray< dtype > &inArray)
- static NdArray< double > var (const NdArray< dtype > &inArray, Axis::Type inAxis=Axis::NONE)
- static NdArray< dtype > vstack (const std::initializer_list< NdArray< dtype > > &inArrayList)
- static NdArray< dtype > zeros (uint32 inSquareSize)
- static NdArray< dtype > zeros (uint32 inNumRows, uint32 inNumCols)
- static NumC::NdArray< dtype > zeros (const NumC::Shape &inShape)

6.16.1 Detailed Description

template< typename dtype >
class NumC::Methods< dtype >

Methods for working with NdArrays.

6.16.2 Member Function Documentation

6.16.2.1 abs() [1/2]

```
template<typename dtype>
static dtype NumC::Methods< dtype >::abs (
            dtype inValue )  [inline], [static]
```

Calculate the absolute value.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩ absolute.html

Parameters

value	

Returns

value

6.16.2.2 abs() [2/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::abs (
            const NdArray< dtype > & inArray )  [inline], [static]
```

Calculate the absolute value element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.absolute.html

Parameters

NdArray	

Returns

NdArray

6.16.2.3 add()

```
template<typename dtype>
template<typename dtypeOut = double>
static NdArray<dtypeOut> NumC::Methods< dtype >::add (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 )  [inline], [static]
```

Add arguments element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.add.html

Parameters

NdArray	
NdArray	

Returns

NdArray

6.16.2.4 alen()

```
template<typename dtype>
static uint32 NumC::Methods< dtype >::alen (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Return the length of the first dimension of the input array.

Parameters

NdArray	

Returns

length uint16

6.16.2.5 all()

```
template<typename dtype>
static NdArray<bool> NumC::Methods< dtype >::all (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE ) [inline], [static]
```

Test whether all array elements along a given axis evaluate to True.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.all.↵ html

Parameters

NdArray	
Axis	

Returns

bool

6.16.2.6 allclose()

```
template<typename dtype>
static bool NumC::Methods< dtype >::allclose (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2,
            double inTolerance = 1e-5 ) [inline], [static]
```

Returns True if two arrays are element-wise equal within a tolerance. inTolerance must be a positive number

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↵ allclose.html

Parameters

NdArray	
NdArray	
(Optional)	tolerance

Returns

bool

6.16.2.7 amax()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::amax (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE ) [inline], [static]
```

Return the maximum of an array or maximum along an axis.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.amax.↵
html

Parameters

NdArray	
(Optional)	Axis

Returns

max value

6.16.2.8 amin()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::amin (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE ) [inline], [static]
```

Return the minimum of an array or minimum along an axis.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.amin.↵
html

Parameters

NdArray	
(Optional)	Axis

Returns

min value

6.16.2.9 any()

```
template<typename dtype>
static NdArray<bool> NumC::Methods< dtype >::any (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE ) [inline], [static]
```

Test whether any array element along a given axis evaluates to True.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.any.html

Parameters

NdArray	
(Optional)	Axis

Returns

NdArray

6.16.2.10 append()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::append (
            const NdArray< dtype > & inArray,
            const NdArray< dtype > & inAppendValues,
            Axis::Type inAxis = Axis::NONE ) [inline], [static]
```

Append values to the end of an array.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.append.html

Parameters

NdArray	
NdArray	append values
(Optional)	axis - The axis along which values are appended. If axis is not given, both inArray and inAppendValues are flattened before use.

Returns

NdArray

6.16.2.11 arange() [1/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::arange (
            dtype inStart,
            dtype inStop,
            dtype inStep = 1 )  [inline], [static]
```

Return evenly spaced values within a given interval.

Values are generated within the half - open interval[start, stop) (in other words, the interval including start but excluding stop). For integer arguments the function is equivalent to the Python built - in range function, but returns an ndarray rather than a list.

When using a non - integer step, such as 0.1, the results will often not be consistent.It is better to use linspace for these cases.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.arange.html

Parameters

start	value,
stop	value,
(Optional)	step value, defaults to 1

Returns

NdArray

6.16.2.12 arange() [2/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::arange (
            dtype inStop )  [inline], [static]
```

Return evenly spaced values within a given interval.

Values are generated within the half - open interval[start, stop) (in other words, the interval including start but excluding stop). For integer arguments the function is equivalent to the Python built - in range function, but returns an ndarray rather than a list.

When using a non - integer step, such as 0.1, the results will often not be consistent.It is better to use linspace for these cases.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.arange.html

Parameters

stop	value, start is 0 and step is 1

Returns

NdArray

6.16.2.13 arccos() [1/2]

```
template<typename dtype>
static double NumC::Methods< dtype >::arccos (
            dtype inValue ) [inline], [static]
```

Trigonometric inverse cosine

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.arccos.html

Parameters

value	

Returns

value

6.16.2.14 arccos() [2/2]

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::arccos (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Trigonometric inverse cosine, element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.arccos.html

Parameters

NdArray	

Returns

NdArray

6.16.2.15 arccosh() [1/2]

```
template<typename dtype>
static double NumC::Methods< dtype >::arccosh (
            dtype inValue ) [inline], [static]
```

Trigonometric inverse hyperbolic cosine.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.arccosh.html

Parameters

value	

Returns

value

6.16.2.16 arccosh() [2/2]

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::arccosh (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Trigonometric inverse hyperbolic cosine, element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.arccosh.html

Parameters

NdArray	

Returns

NdArray

6.16.2.17 arcsin() [1/2]

```
template<typename dtype>
static double NumC::Methods< dtype >::arcsin (
            dtype inValue ) [inline], [static]
```

Trigonometric inverse sine.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩
arcsin.html

Parameters

value	

Returns

value

6.16.2.18 arcsin() [2/2]

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::arcsin (
            const NdArray< dtype > & inArray )  [inline], [static]
```

Trigonometric inverse sine, element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩
arcsin.html

Parameters

NdArray	

Returns

NdArray

6.16.2.19 arcsinh() [1/2]

```
template<typename dtype>
static double NumC::Methods< dtype >::arcsinh (
            dtype inValue )  [inline], [static]
```

Trigonometric inverse hyperbolic sine.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩
arcsinh.html

Parameters

value	

Returns

value

6.16.2.20 arcsinh() [2/2]

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::arcsinh (
            const NdArray< dtype > & inArray )  [inline], [static]
```

Trigonometric inverse hyperbolic sine, element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.arcsinh.html

Parameters

NdArray	

Returns

NdArray

6.16.2.21 arctan() [1/2]

```
template<typename dtype>
static double NumC::Methods< dtype >::arctan (
            dtype inValue )  [inline], [static]
```

Trigonometric inverse tangent.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.arctan.html

Parameters

value	

Returns

value

6.16.2.22 arctan() [2/2]

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::arctan (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Trigonometric inverse tangent, element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.arctan.html

Parameters

NdArray	

Returns

NdArray

6.16.2.23 arctan2() [1/2]

```
template<typename dtype>
static double NumC::Methods< dtype >::arctan2 (
            dtype inY,
            dtype inX ) [inline], [static]
```

Trigonometric inverse tangent.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.arctan2.html

Parameters

Y	
x	

Returns

value

6.16.2.24 arctan2() [2/2]

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::arctan2 (
            const NdArray< dtype > & inY,
            const NdArray< dtype > & inX ) [inline], [static]
```

Trigonometric inverse tangent, element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.arctan2.html

Parameters

NdArray	y
NdArray	x

Returns

NdArray

6.16.2.25 arctanh() [1/2]

```
template<typename dtype>
static double NumC::Methods< dtype >::arctanh (
            dtype inValue )  [inline], [static]
```

Trigonometric inverse hyperbolic tangent.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.arctanh.html

Parameters

value	

Returns

value

6.16.2.26 arctanh() [2/2]

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::arctanh (
            const NdArray< dtype > & inArray )  [inline], [static]
```

Trigonometric inverse hyperbolic tangent, element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.arctanh.html

Parameters

NdArray	

Returns

NdArray

6.16.2.27 argmax()

```
template<typename dtype>
static NdArray<uint32> NumC::Methods< dtype >::argmax (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE ) [inline], [static]
```

Returns the indices of the maximum values along an axis.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.argmax.html

Parameters

NdArray	
(Optional)	axis

Returns

NdArray

6.16.2.28 argmin()

```
template<typename dtype>
static NdArray<uint32> NumC::Methods< dtype >::argmin (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE ) [inline], [static]
```

Returns the indices of the minimum values along an axis.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.argmin.html

Parameters

NdArray	
(Optional)	axis

Returns

NdArray

6.16.2.29 argsort()

```
template<typename dtype>
static NdArray<uint32> NumC::Methods< dtype >::argsort (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE )  [inline], [static]
```

Returns the indices that would sort an array.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.argsort.html

Parameters

NdArray	
(Optional)	axis

Returns

NdArray

6.16.2.30 argwhere()

```
template<typename dtype>
static NdArray<uint32> NumC::Methods< dtype >::argwhere (
            const NdArray< dtype > & inArray )  [inline], [static]
```

Returns the indices that would sort an array.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.argwhere.html

Parameters

NdArray	
(Optional)	axis

Returns

NdArray

6.16.2.31 around() [1/2]

```
template<typename dtype>
static dtype NumC::Methods< dtype >::around (
            dtype inValue,
            uint8 inNumDecimals = 0 ) [inline], [static]
```

Evenly round to the given number of decimals.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩ around.html

Parameters

value	
(Optional)	decimals, default = 0

Returns

value

6.16.2.32 around() [2/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::around (
            const NdArray< dtype > & inArray,
            uint8 inNumDecimals = 0 ) [inline], [static]
```

Evenly round to the given number of decimals.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩ around.html

Parameters

NdArray	
(Optional)	decimals, default = 0

Returns

NdArray

6.16.2.33 array_equal()

```
template<typename dtype>
static bool NumC::Methods< dtype >::array_equal (
```

```
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 )  [inline], [static]
```

True if two arrays have the same shape and elements, False otherwise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.array←_equal.html

Parameters

NdArray	
NdArray	

Returns

bool

6.16.2.34 array_equiv()

```
template<typename dtype>
static bool NumC::Methods< dtype >::array_equiv (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 )  [inline], [static]
```

Returns True if input arrays are shape consistent and all elements equal.

Shape consistent means they are either the same shape, or one input array can be broadcasted to create the same shape as the other one.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.array←_equiv.html

Parameters

NdArray	
NdArray	

Returns

bool

6.16.2.35 asarray() [1/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::asarray (
            const std::vector< dtype > & inVector )  [inline], [static]
```

Convert the vector to an array.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩
asarray.html

Parameters

std::vector	

Returns

NdArray

6.16.2.36 asarray() [2/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::asarray (
            std::initializer_list< dtype > & inList )  [inline], [static]
```

Convert the list initializer to an array. eg: NdArray<int> myArray = NumC::asarray<int>({1,2,3});

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩
asarray.html

Parameters

std::vector	

Returns

NdArray

6.16.2.37 astype()

```
template<typename dtype>
template<typename dtypeOut = double>
static NdArray<dtypeOut> NumC::Methods< dtype >::astype (
            const NdArray< dtype > inArray )  [inline], [static]
```

Returns a copy of the array, cast to a specified type.

Parameters

NdArray	

Returns

NdArray

6.16.2.38 average() [1/2]

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::average (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE )  [inline], [static]
```

Compute the average along the specified axis.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.average.html

Parameters

NdArray	
(Optional)	axis

Returns

NdArray

6.16.2.39 average() [2/2]

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::average (
            const NdArray< dtype > & inArray,
            const NdArray< dtype > & inWeights,
            Axis::Type inAxis = Axis::NONE )  [inline], [static]
```

Compute the weighted average along the specified axis.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.average.html

Parameters

NdArray	
NdArray	of weights, otherwise all weights = 1
(Optional)	axis

Returns

NdArray

6.16.2.40 bincount() [1/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::bincount (
            const NdArray< dtype > & inArray,
            uint16 inMinLength = 0 ) [inline], [static]
```

Count number of occurrences of each value in array of non-negative ints. Negative values will be counted in the zero bin.

The number of bins(of size 1) is one larger than the largest value in x. If minlength is specified, there will be at least this number of bins in the output array(though it will be longer if necessary, depending on the contents of x).Each bin gives the number of occurrences of its index value in x.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.bincount.html

Parameters

NdArray	
min	bin length

Returns

NdArray

6.16.2.41 bincount() [2/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::bincount (
            const NdArray< dtype > & inArray,
            const NdArray< dtype > & inWeights,
            uint16 inMinLength = 0 ) [inline], [static]
```

Count number of occurrences of each value in array of non-negative ints. Negative values will be counted in the zero bin.

The number of bins(of size 1) is one larger than the largest value in x. If minlength is specified, there will be at least this number of bins in the output array(though it will be longer if necessary, depending on the contents of x).Each bin gives the number of occurrences of its index value in x.If weights is specified the input array is weighted by it, i.e. if a value n is found at position i, out[n] += weight[i] instead of out[n] += 1. Weights array shall be of the same shape as inArray.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.bincount.html

Parameters

NdArray	
NdArray	weights
min	bin length

Returns

NdArray

6.16.2.42 bitwise_and()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::bitwise_and (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 )  [inline], [static]
```

Compute the bit-wise AND of two arrays element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩
bitwise_and.html

Parameters

NdArray	1
NdArray	2

Returns

NdArray

6.16.2.43 bitwise_not()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::bitwise_not (
            const NdArray< dtype > & inArray )  [inline], [static]
```

Compute the bit-wise NOT the input array element-wise.

NdArray

Returns

NdArray

6.16.2.44 bitwise_or()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::bitwise_or (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 )  [inline], [static]
```

Compute the bit-wise OR of two arrays element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.bitwise_or.html

Parameters

NdArray	1
NdArray	2

Returns

NdArray

6.16.2.45 bitwise_xor()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::bitwise_xor (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 )  [inline], [static]
```

Compute the bit-wise XOR of two arrays element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.bitwise_xor.html

Parameters

NdArray	1
NdArray	2

Returns

NdArray

6.16.2.46 byteswap()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::byteswap (
            const NdArray< dtype > & inArray )  [inline], [static]
```

Return a new array with the bytes of the array elements swapped.

Parameters

NdArray	

Returns

NdArray

6.16.2.47 cbrt() [1/2]

```
template<typename dtype>
static double NumC::Methods< dtype >::cbrt (
            dtype inValue ) [inline], [static]
```

Return the cube-root of an array. Not super usefull if not using a floating point type

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.cbrt.html

Parameters

value	

Returns

value

6.16.2.48 cbrt() [2/2]

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::cbrt (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Return the cube-root of an array, element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.cbrt.html

Parameters

NdArray	

Returns

NdArray

6.16.2.49 ceil() [1/2]

```
template<typename dtype>
static dtype NumC::Methods< dtype >::ceil (
            dtype inValue ) [inline], [static]
```

Return the ceiling of the input.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ceil.html

Parameters

value	

Returns

value

6.16.2.50 ceil() [2/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::ceil (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Return the ceiling of the input, element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ceil.html

Parameters

NdArray	

Returns

NdArray

6.16.2.51 clip() [1/2]

```
template<typename dtype>
static dtype NumC::Methods< dtype >::clip (
```

```
            dtype inValue,
            dtype inMinValue,
            dtype inMaxValue )  [inline], [static]
```

Clip (limit) the value.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.clip.html

Parameters

value	
min	Value
max	Value

Returns

NdArray

6.16.2.52 clip() [2/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::clip (
            const NdArray< dtype > & inArray,
            dtype inMinValue,
            dtype inMaxValue )  [inline], [static]
```

Clip (limit) the values in an array.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.clip.html

Parameters

NdArray	
min	Value
max	Value

Returns

NdArray

6.16.2.53 column_stack()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::column_stack (
            const std::initializer_list< NdArray< dtype > > & inArrayList )  [inline], [static]
```

Stack 1-D arrays as columns into a 2-D array.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.column_stack.html

Parameters

{list}	of arrays to stack

Returns

NdArray

6.16.2.54 concatenate()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::concatenate (
            const std::initializer_list< NdArray< dtype > > & inArrayList,
            Axis::Type inAxis = Axis::NONE ) [inline], [static]
```

Join a sequence of arrays along an existing axis.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.concatenate.html

Parameters

NdArray	1
NdArray	2
(Optional)	Axis (Default NONE)

Returns

NdArray

6.16.2.55 contains()

```
template<typename dtype>
static NdArray<bool> NumC::Methods< dtype >::contains (
            const NdArray< dtype > & inArray,
            dtype inValue,
            Axis::Type inAxis = Axis::NONE ) [inline], [static]
```

returns whether or not a value is included the array

Parameters

NdArray	
value	
(Optional)	axis

Returns

bool

6.16.2.56 copy()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::copy (
            const NdArray< dtype > & inArray )  [inline], [static]
```

Return an array copy of the given object.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.copy.html

Parameters

NdArray	

Returns

NdArray

6.16.2.57 copySign()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::copySign (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 )  [inline], [static]
```

Change the sign of x1 to that of x2, element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.copysign.html

Parameters

NdArray	1
NdArray	2

Returns

NdArray

6.16.2.58 copyto()

```
template<typename dtype>
static NdArray<dtype>& NumC::Methods< dtype >::copyto (
            NdArray< dtype > & inDestArray,
            const NdArray< dtype > & inSrcArray )  [inline], [static]
```

Copies values from one array to another

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.copyto.html

Parameters

NdArray	destination
NdArray	source

Returns

NdArray

6.16.2.59 cos() [1/2]

```
template<typename dtype>
static double NumC::Methods< dtype >::cos (
            dtype inValue )  [inline], [static]
```

Cosine

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.cos.html

Parameters

value	

Returns

value

6.16.2.60 cos() [2/2]

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::cos (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Cosine element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.cos.↩
html

Parameters

NdArray	

Returns

NdArray

6.16.2.61 cosh() [1/2]

```
template<typename dtype>
static double NumC::Methods< dtype >::cosh (
            dtype inValue ) [inline], [static]
```

Hyperbolic Cosine.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.cosh.↩
html

Parameters

Value	

Returns

value

6.16.2.62 cosh() [2/2]

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::cosh (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Hyperbolic Cosine element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.cosh.↩
html

Parameters

NdArray	

Returns

NdArray

6.16.2.63 count_nonzero()

```
template<typename dtype>
static NdArray<uint32> NumC::Methods< dtype >::count_nonzero (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE ) [inline], [static]
```

Counts the number of non-zero values in the array.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.count_nonzero.html

Parameters

NdArray	
(Optional)	Axis

Returns

NdArray

6.16.2.64 cross()

```
template<typename dtype>
template<typename dtypeOut = double>
static NdArray<dtypeOut> NumC::Methods< dtype >::cross (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2,
            Axis::Type inAxis = Axis::NONE ) [inline], [static]
```

Return the cross product of two (arrays of) vectors.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.cross.html

Parameters

NdArray	1
NdArray	2
(Optional)	Axis - default = row

Returns

NdArray

6.16.2.65 cube()

```
template<typename dtype>
template<typename dtypeOut = double>
static NdArray<dtypeOut> NumC::Methods< dtype >::cube (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Cubes the elements of the array

Parameters

NdArray	

Returns

NdArray

6.16.2.66 cumprod()

```
template<typename dtype>
template<typename dtypeOut = double>
static NdArray<dtypeOut> NumC::Methods< dtype >::cumprod (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE ) [inline], [static]
```

Return the cumulative product of elements along a given axis.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩cumprod.html

Parameters

NdArray	
(Optional)	Axis

Returns

NdArray

6.16.2.67 cumsum()

```
template<typename dtype>
template<typename dtypeOut = double>
static NdArray<dtypeOut> NumC::Methods< dtype >::cumsum (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE ) [inline], [static]
```

Return the cumulative sum of the elements along a given axis.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.cumsum.html

Parameters

NdArray	
(Optional)	Axis

Returns

NdArray

6.16.2.68 deg2rad() [1/2]

```
template<typename dtype>
static double NumC::Methods< dtype >::deg2rad (
            dtype inValue ) [inline], [static]
```

Convert angles from degrees to radians.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.deg2rad.html

Parameters

value	

Returns

value

6.16.2.69 deg2rad() [2/2]

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::deg2rad (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Convert angles from degrees to radians.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.deg2rad.html

Parameters

NdArray	

Returns

NdArray

6.16.2.70 deleteIndices() [1/3]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::deleteIndices (
            const NdArray< dtype > & inArray,
            const NdArray< uint32 > & inArrayIdxs,
            Axis::Type inAxis = Axis::NONE )  [inline], [static]
```

Return a new array with sub-arrays along an axis deleted.

Parameters

NdArray	
NdArray	indices to delete
(Optional)	Axis, if none the indices will be applied to the flattened array

Returns

NdArray

6.16.2.71 deleteIndices() [2/3]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::deleteIndices (
            const NdArray< dtype > & inArray,
            const Slice & inIndicesSlice,
            Axis::Type inAxis = Axis::NONE )  [inline], [static]
```

Return a new array with sub-arrays along an axis deleted.

Parameters

NdArray	
Slice	to delete
(Optional)	Axis, if none the indices will be applied to the flattened array

Returns

NdArray

6.16.2.72 deleteIndices() [3/3]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::deleteIndices (
            const NdArray< dtype > & inArray,
            uint32 inIndex,
            Axis::Type inAxis = Axis::NONE ) [inline], [static]
```

Return a new array with sub-arrays along an axis deleted.

Parameters

NdArray	
index	to delete
(Optional)	Axis, if none the indices will be applied to the flattened array

Returns

NdArray

6.16.2.73 diagflat()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::diagflat (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Create a two-dimensional array with the flattened input as a diagonal.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.diagflat.html

Parameters

NdArray	

Returns

NdArray

6.16.2.74 diagonal()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::diagonal (
            const NdArray< dtype > & inArray,
            uint32 inOffset = 0,
            Axis::Type inAxis = Axis::ROW ) [inline], [static]
```

Return specified diagonals.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.diagonal.html

Parameters

NdArray	
Offset	of the diagonal from the main diagonal. Can be both positive and negative. Defaults to 0.
(Optional)	axis the offset is applied to

Returns

NdArray

6.16.2.75 diff()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::diff (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE ) [inline], [static]
```

Calculate the n-th discrete difference along given axis. Unsigned dtypes will give you weird results...obviously.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.diff.html

Parameters

NdArray	
(Optional)	Axis

Returns

NdArray

6.16.2.76 divide()

```
template<typename dtype>
template<typename dtypeOut = double>
```

```
static NdArray<dtypeOut> NumC::Methods< dtype >::divide (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 )  [inline], [static]
```

Returns a true division of the inputs, element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.divide.html

Parameters

NdArray	1
NdArray	2

Returns

NdArray

6.16.2.77 dot()

```
template<typename dtype>
template<typename dtypeOut = double>
static NdArray<dtypeOut> NumC::Methods< dtype >::dot (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 )  [inline], [static]
```

Dot product of two arrays.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.dot.html

Parameters

NdArray	1
NdArray	2

Returns

NdArray

6.16.2.78 dump()

```
template<typename dtype>
static void NumC::Methods< dtype >::dump (
            const NdArray< dtype > & inArray,
            const std::string & inFilename )  [inline], [static]
```

Dump a binary file of the array to the specified file. The array can be read back with or NumC::load.

Parameters

NdArray	
string	filename

Returns

NdArray

6.16.2.79 empty() [1/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::empty (
            uint32 inNumRows,
            uint32 inNumCols ) [inline], [static]
```

Return a new array of given shape and type, without initializing entries.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.empty.html

Parameters

inNumRows	
inNumCols	

Returns

NdArray

6.16.2.80 empty() [2/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::empty (
            const Shape & inShape ) [inline], [static]
```

Return a new array of given shape and type, without initializing entries.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.empty.html

Parameters

Shape	

Returns

NdArray

6.16.2.81 empty_like()

```
template<typename dtype>
template<typename dtypeOut = double>
static NdArray<dtypeOut> NumC::Methods< dtype >::empty_like (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Return a new array with the same shape as a given array.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.empty_like.html

Parameters

NdArray	

Returns

NdArray

6.16.2.82 endianess()

```
template<typename dtype>
static Endian::Type NumC::Methods< dtype >::endianess (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Return the endianess of the array values.

Parameters

NdArray	

Returns

Endian::Type

6.16.2.83 equal()

```
template<typename dtype>
static NdArray<bool> NumC::Methods< dtype >::equal (
```

```
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 )  [inline], [static]
```

Return (x1 == x2) element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.equal.html

Parameters

NdArray	
NdArray	

Returns

NdArray

6.16.2.84 exp() [1/2]

```
template<typename dtype>
static double NumC::Methods< dtype >::exp (
            dtype inValue )  [inline], [static]
```

Calculate the exponential of the input value.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.exp.html

Parameters

value	

Returns

value

6.16.2.85 exp() [2/2]

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::exp (
            const NdArray< dtype > & inArray )  [inline], [static]
```

Calculate the exponential of all elements in the input array.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.exp.html

Parameters

NdArray	

Returns

NdArray

6.16.2.86 exp2() [1/2]

```
template<typename dtype>
static double NumC::Methods< dtype >::exp2 (
            dtype inValue ) [inline], [static]
```

Calculate 2**p for all p in the input value.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.exp2.html

Parameters

value	

Returns

value

6.16.2.87 exp2() [2/2]

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::exp2 (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Calculate 2**p for all p in the input array.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.exp2.html

Parameters

NdArray	

Returns

NdArray

6.16.2.88 expm1() [1/2]

```
template<typename dtype>
static double NumC::Methods< dtype >::expm1 (
            dtype inValue ) [inline], [static]
```

Calculate exp(x) - 1 for the input value.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩
expm1.html

Parameters

value	

Returns

value

6.16.2.89 expm1() [2/2]

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::expm1 (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Calculate exp(x) - 1 for all elements in the array.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩
expm1.html

Parameters

NdArray	

Returns

NdArray

6.16.2.90 eye() [1/3]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::eye (
```

```
            uint32 inN,
            int32 inK = 0 )  [inline], [static]
```

Return a 2-D array with ones on the diagonal and zeros elsewhere.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.eye.html

Parameters

number	of rows and columns (N)
K	- Index of the diagonal: 0 (the default) refers to the main diagonal, a positive value refers to an upper diagonal, and a negative value to a lower diagonal.

Returns

NdArray

6.16.2.91 eye() [2/3]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::eye (
            uint32 inN,
            uint32 inM,
            int32 inK = 0 )  [inline], [static]
```

Return a 2-D array with ones on the diagonal and zeros elsewhere.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.eye.html

Parameters

number	of rows (N)
number	of columns (M)
K	- Index of the diagonal: 0 (the default) refers to the main diagonal, a positive value refers to an upper diagonal, and a negative value to a lower diagonal.

Returns

NdArray

6.16.2.92 eye() [3/3]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::eye (
```

```
            const Shape & inShape,
            int32 inK = 0 )  [inline], [static]
```

Return a 2-D array with ones on the diagonal and zeros elsewhere.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.eye.html

Parameters

Shape	
K	- Index of the diagonal: 0 (the default) refers to the main diagonal, a positive value refers to an upper diagonal, and a negative value to a lower diagonal.

Returns

NdArray

6.16.2.93 fix() [1/2]

```
template<typename dtype>
static dtype NumC::Methods< dtype >::fix (
            dtype inValue )  [inline], [static]
```

Round to nearest integer towards zero.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.fix.html

Parameters

value	

Returns

value

6.16.2.94 fix() [2/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::fix (
            const NdArray< dtype > & inArray )  [inline], [static]
```

Round to nearest integer towards zero.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.fix.html

Parameters

NdArray	

Returns

NdArray

6.16.2.95 flatnonzero()

```
template<typename dtype>
static NdArray<uint32> NumC::Methods< dtype >::flatnonzero (
            const NdArray< dtype > & inArray )  [inline], [static]
```

Return indices that are non-zero in the flattened version of a.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.flatnonzero.html

Parameters

NdArray	

Returns

NdArray

6.16.2.96 flatten()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::flatten (
            const NdArray< dtype > & inArray )  [inline], [static]
```

Return a copy of the array collapsed into one dimension.

Parameters

NdArray	

Returns

NdArray

6.16.2.97 flip()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::flip (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis ) [inline], [static]
```

Reverse the order of elements in an array along the given axis.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.flip.html

Parameters

NdArray	
axis	

Returns

NdArray

6.16.2.98 fliplr()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::fliplr (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Flip array in the left/right direction.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.fliplr.html

Parameters

NdArray	

Returns

NdArray

6.16.2.99 flipud()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::flipud (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Flip array in the up/down direction.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩
flipud.html

Parameters

NdArray	

Returns

NdArray

6.16.2.100 floor() [1/2]

```
template<typename dtype>
static dtype NumC::Methods< dtype >::floor (
            dtype inValue ) [inline], [static]
```

Return the floor of the input.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.floor.html

Parameters

value	

Returns

value

6.16.2.101 floor() [2/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::floor (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Return the floor of the input, element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.floor.html

Parameters

NdArray	

Returns

NdArray

6.16.2.102 floor_divide() [1/2]

```
template<typename dtype>
static dtype NumC::Methods< dtype >::floor_divide (
            dtype inValue1,
            dtype inValue2 )  [inline], [static]
```

Return the largest integer smaller or equal to the division of the inputs.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.floor↩
_divide.html

Parameters

value	1
value	2

Returns

value

6.16.2.103 floor_divide() [2/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::floor_divide (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 )  [inline], [static]
```

Return the largest integer smaller or equal to the division of the inputs.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.floor↩
_divide.html

Parameters

NdArray	1
NdArray	2

Returns

NdArray

6.16.2.104 fmax() [1/2]

```
template<typename dtype>
static dtype NumC::Methods< dtype >::fmax (
            dtype inValue1,
            dtype inValue2 )  [inline], [static]
```

maximum of inputs.

Compare two value and returns a value containing the maxima

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.fmax.html

Parameters

value	1
value	2

Returns

value

6.16.2.105 fmax() [2/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::fmax (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 )  [inline], [static]
```

Element-wise maximum of array elements.

Compare two arrays and returns a new array containing the element - wise maxima

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.fmax.html

Parameters

NdArray	1
NdArray	2

Returns

NdArray

6.16.2.106 fmin() [1/2]

```
template<typename dtype>
static dtype NumC::Methods< dtype >::fmin (
            dtype inValue1,
            dtype inValue2 )  [inline], [static]
```

minimum of inputs.

Compare two value and returns a value containing the minima

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.fmin.html

Parameters

value	1
value	2

Returns

value

6.16.2.107 fmin() [2/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::fmin (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 )  [inline], [static]
```

Element-wise minimum of array elements.

Compare two arrays and returns a new array containing the element - wise minima

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.fmin.html

Parameters

NdArray	1
NdArray	2

Returns

NdArray

6.16.2.108 fmod() [1/2]

```
template<typename dtype>
static dtype NumC::Methods< dtype >::fmod (
            dtype inValue1,
            dtype inValue2 )  [inline], [static]
```

Return the remainder of division.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.fmod.html

Parameters

value	1
value	2

Returns

value

6.16.2.109 fmod() [2/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::fmod (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 )  [inline], [static]
```

Return the element-wise remainder of division.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.fmod.html

Parameters

NdArray	1
NdArray	2

Returns

NdArray

6.16.2.110 fromfile()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::fromfile (
```

```
            const std::string & inFilename,
            const std::string & inSep = "" )  [inline], [static]
```

Construct an array from data in a text or binary file.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩fromfile.html

Parameters

filename	
seperator,Separator	between items if file is a text file. Empty ("") separator means the file should be treated as binary. Right now the only supported seperators are " ", "\t", "\n"

Returns

NdArray

6.16.2.111 full() [1/3]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::full (
            uint32 inSquareSize,
            dtype inFillValue )  [inline], [static]
```

Return a new array of given shape and type, filled with inFillValue

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.full.↩html

Parameters

square	size
fill	value

Returns

NdArray

6.16.2.112 full() [2/3]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::full (
            uint32 inNumRows,
            uint32 inNumCols,
            dtype inFillValue )  [inline], [static]
```

Return a new array of given shape and type, filled with inFillValue

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.full.html

Parameters

numRows	
numCols	
fill	value

Returns

NdArray

6.16.2.113 full() [3/3]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::full (
            const Shape & inShape,
            dtype inFillValue )  [inline], [static]
```

Return a new array of given shape and type, filled with inFillValue

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.full.↩
html

Parameters

Shape	
fill	value

Returns

NdArray

6.16.2.114 full_like()

```
template<typename dtype>
template<typename dtypeOut = double>
static NdArray<dtypeOut> NumC::Methods< dtype >::full_like (
            const NdArray< dtype > & inArray,
            dtype inFillValue )  [inline], [static]
```

Return a full array with the same shape and type as a given array.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.full_↩
like.html

Parameters

NdArray	
fill	value

Returns

NdArray

6.16.2.115 greater()

```
template<typename dtype>
static NdArray<bool> NumC::Methods< dtype >::greater (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 ) [inline], [static]
```

Return the truth value of (x1 > x2) element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.greater.html

Parameters

NdArray	1
NdArray	2

Returns

NdArray

6.16.2.116 greater_equal()

```
template<typename dtype>
static NdArray<bool> NumC::Methods< dtype >::greater_equal (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 ) [inline], [static]
```

Return the truth value of (x1 >= x2) element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.greater_equal.html

Parameters

NdArray	1
NdArray	2

Returns

NdArray

6.16.2.117 histogram()

```
template<typename dtype>
static std::pair<NdArray<uint32>, NdArray<double> > NumC::Methods< dtype >::histogram (
            const NdArray< dtype > & inArray,
            uint32 inNumBins = 10 )  [inline], [static]
```

Compute the histogram of a set of data.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.histogram.html

Parameters

NdArray	
number	of bins, default 10

Returns

std::pair of NdArrays; first is histogram counts, seconds is the bin edges

6.16.2.118 hstack()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::hstack (
            const std::initializer_list< NdArray< dtype > > & inArrayList )  [inline], [static]
```

Stack arrays in sequence horizontally (column wise).

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.hstack.html

Parameters

{list}	of arrays to stack

Returns

NdArray

6.16.2.119 hypot() [1/2]

```
template<typename dtype>
template<typename dtypeOut = double>
static dtypeOut NumC::Methods< dtype >::hypot (
            dtype inValue1,
            dtype inValue2 ) [inline], [static]
```

Given the "legs" of a right triangle, return its hypotenuse.

Equivalent to sqrt(x1∗∗2 + x2 ∗ ∗2), element - wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.hypot.html

Parameters

value	1
value	2

Returns

NdArray

6.16.2.120 hypot() [2/2]

```
template<typename dtype>
template<typename dtypeOut = double>
static NdArray<dtypeOut> NumC::Methods< dtype >::hypot (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 ) [inline], [static]
```

Given the "legs" of a right triangle, return its hypotenuse.

Equivalent to sqrt(x1∗∗2 + x2 ∗ ∗2), element - wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.hypot.html

Parameters

NdArray	1
NdArray	2

Returns

NdArray

6.16.2.121 identity()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::identity (
            uint32 inSquareSize )  [inline], [static]
```

Return the identity array.

The identity array is a square array with ones on the main diagonal.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.identity.html

Parameters

matrix	square size

Returns

NdArray

6.16.2.122 intersect1d()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::intersect1d (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 )  [inline], [static]
```

Find the intersection of two arrays.

Return the sorted, unique values that are in both of the input arrays.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.intersect1d.html

Parameters

NdArray	1
NdArray	2

Returns

NdArray

6.16.2.123 invert()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::invert (
            const NdArray< dtype > & inArray )  [inline], [static]
```

Compute bit-wise inversion, or bit-wise NOT, element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩
invert.html

Parameters

NdArray	

Returns

NdArray

6.16.2.124 isclose()

```
template<typename dtype>
static NdArray<bool> NumC::Methods< dtype >::isclose (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2,
            double inRtol = 1e-05,
            double inAtol = 1e-08 )  [inline], [static]
```

Returns a boolean array where two arrays are element-wise equal within a tolerance.

For finite values, isclose uses the following equation to test whether two floating point values are equivalent. absolute(a - b) <= (atol + rtol ∗ absolute(b))

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩
isclose.html

Parameters

NdArray	1
NdArray	2
relative	tolerance
absolute	tolerance

Returns

NdArray

6.16.2.125 isnan() [1/2]

```
template<typename dtype>
static bool NumC::Methods< dtype >::isnan (
            dtype inValue )  [inline], [static]
```

Test for NaN and return result as a boolean.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.isnan.html

Parameters

value	

Returns

bool

6.16.2.126 isnan() [2/2]

```
template<typename dtype>
static NdArray<bool> NumC::Methods< dtype >::isnan (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Test element-wise for NaN and return result as a boolean array.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.isnan.html

Parameters

NdArray	

Returns

NdArray

6.16.2.127 ldexp() [1/2]

```
template<typename dtype>
static dtype NumC::Methods< dtype >::ldexp (
            dtype inValue1,
            uint8 inValue2 ) [inline], [static]
```

Returns x1 ∗ 2^x2.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ldexp.html

Parameters

value	1
value	2

Returns

value

6.16.2.128 ldexp() [2/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::ldexp (
            const NdArray< dtype > & inArray1,
            const NdArray< uint8 > & inArray2 )  [inline], [static]
```

Returns x1 $*$ 2^x2, element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ldexp.html

Parameters

NdArray	1
NdArray	2

Returns

NdArray

6.16.2.129 left_shift()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::left_shift (
            const NdArray< dtype > & inArray,
            uint8 inNumBits )  [inline], [static]
```

Shift the bits of an integer to the left.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.left_shift.html

Parameters

NdArray	
number	of bits to sift

Returns

NdArray

6.16.2.130 less()

```
template<typename dtype>
static NdArray<bool> NumC::Methods< dtype >::less (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 ) [inline], [static]
```

Return the truth value of (x1 < x2) element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.less.html

Parameters

NdArray	1
NdArray	2

Returns

NdArray

6.16.2.131 less_equal()

```
template<typename dtype>
static NdArray<bool> NumC::Methods< dtype >::less_equal (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 ) [inline], [static]
```

Return the truth value of (x1 <= x2) element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.less_equal.html

Parameters

NdArray	1
NdArray	2

Returns

NdArray

6.16.2.132 linspace()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::linspace (
```

```
dtype inStart,
dtype inStop,
uint32 inNum = 50,
bool endPoint = true )  [inline], [static]
```

Return evenly spaced numbers over a specified interval.

Returns num evenly spaced samples, calculated over the interval[start, stop].

The endpoint of the interval can optionally be excluded.

Mostly only usefull if called with a floating point type for the template argument.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.linspace.html

Parameters

start	point
end	point
number	of points, default = 50
include	endPoint, default = true

Returns

NdArray

6.16.2.133 load()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::load (
            const std::string & inFilename )  [inline], [static]
```

loads a .bin file from the dump() method into an NdArray

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.load.html

Parameters

string	filename

Returns

NdArray

6.16.2.134 log() [1/2]

```
template<typename dtype>
static double NumC::Methods< dtype >::log (
            dtype inValue ) [inline], [static]
```

Natural logarithm.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.log.↵
html

Parameters

value	

Returns

value

6.16.2.135 log() [2/2]

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::log (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Natural logarithm, element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.log.↵
html

Parameters

NdArray	

Returns

NdArray

6.16.2.136 log10() [1/2]

```
template<typename dtype>
static double NumC::Methods< dtype >::log10 (
            dtype inValue ) [inline], [static]
```

Return the base 10 logarithm of the input array.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↵
log10.html

Parameters

value	

Returns

value

6.16.2.137 log10() [2/2]

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::log10 (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Return the base 10 logarithm of the input array, element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩
log10.html

Parameters

NdArray	

Returns

NdArray

6.16.2.138 log1p() [1/2]

```
template<typename dtype>
static double NumC::Methods< dtype >::log1p (
            dtype inValue ) [inline], [static]
```

Return the natural logarithm of one plus the input array.

Calculates log(1 + x).

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩
log1p.html

Parameters

value	

Returns

value

6.16.2.139 log1p() [2/2]

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::log1p (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Return the natural logarithm of one plus the input array, element-wise.

Calculates log(1 + x).

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩
log1p.html

Parameters

NdArray	

Returns

NdArray

6.16.2.140 log2() [1/2]

```
template<typename dtype>
static double NumC::Methods< dtype >::log2 (
            dtype inValue ) [inline], [static]
```

Base-2 logarithm of x.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.log2.↩
html

Parameters

value	

Returns

value

6.16.2.141 log2() [2/2]

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::log2 (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Base-2 logarithm of x.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.log2.html

Parameters

NdArray	

Returns

NdArray

6.16.2.142 logical_and()

```
template<typename dtype>
static NdArray<bool> NumC::Methods< dtype >::logical_and (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 ) [inline], [static]
```

Compute the truth value of x1 AND x2 element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.logical_and.html

Parameters

NdArray	1
NdArray	2

Returns

NdArray

6.16.2.143 logical_not()

```
template<typename dtype>
static NdArray<bool> NumC::Methods< dtype >::logical_not (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Compute the truth value of NOT x element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.logical_not.html

Parameters

NdArray	

Returns

NdArray

6.16.2.144 logical_or()

```
template<typename dtype>
static NdArray<bool> NumC::Methods< dtype >::logical_or (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 )  [inline], [static]
```

Compute the truth value of x1 OR x2 element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.logical_or.html

Parameters

NdArray	1
NdArray	2

Returns

NdArray

6.16.2.145 logical_xor()

```
template<typename dtype>
static NdArray<bool> NumC::Methods< dtype >::logical_xor (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 )  [inline], [static]
```

Compute the truth value of x1 XOR x2 element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.logical_xor.html

Parameters

NdArray	1
NdArray	2

Returns

NdArray

6.16.2.146 matmul()

```
template<typename dtype>
template<typename dtypeOut = double>
static NdArray<dtypeOut> NumC::Methods< dtype >::matmul (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 ) [inline], [static]
```

Matrix product of two arrays.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩
matmul.html

Parameters

NdArray	1
NdArray	2

Returns

NdArray

6.16.2.147 max()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::max (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE ) [inline], [static]
```

Return the maximum of an array or maximum along an axis.

Parameters

NdArray	
(Optional)	axis

Returns

NdArray

6.16.2.148 maximum()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::maximum (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 )  [inline], [static]
```

Element-wise maximum of array elements.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.maximum.html

Parameters

NdArray	1
NdArray	2

Returns

NdArray

6.16.2.149 mean()

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::mean (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE )  [inline], [static]
```

Compute the mean along the specified axis.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.mean.html

Parameters

NdArray	
(Optional)	axis

Returns

NdArray

6.16.2.150 median()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::median (
```

```
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE )  [inline], [static]
```

Compute the median along the specified axis.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.median.html

Parameters

NdArray	
(Optional)	axis

Returns

NdArray

6.16.2.151 min()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::min (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE )  [inline], [static]
```

Return the minimum of an array or maximum along an axis.

Parameters

NdArray	
(Optional)	axis

Returns

NdArray

6.16.2.152 minimum()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::minimum (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 )  [inline], [static]
```

Element-wise minimum of array elements.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.minimum.html

Parameters

NdArray	1
NdArray	2

Returns

NdArray

6.16.2.153 mod()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::mod (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 )  [inline], [static]
```

Return element-wise remainder of division.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.mod.html

Parameters

NdArray	1
NdArray	2

Returns

NdArray

6.16.2.154 multiply()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::multiply (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 )  [inline], [static]
```

Multiply arguments element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.multiply.html

Parameters

NdArray	1
NdArray	2

Returns

NdArray

6.16.2.155 nanargmax()

```
template<typename dtype>
static NdArray<uint32> NumC::Methods< dtype >::nanargmax (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE ) [inline], [static]
```

Returns the indices of the maximum values along an axis ignoring NaNs.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.nanargmax.html

Parameters

NdArray	
(Optional)	axis

Returns

NdArray

6.16.2.156 nanargmin()

```
template<typename dtype>
static NdArray<uint32> NumC::Methods< dtype >::nanargmin (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE ) [inline], [static]
```

Returns the indices of the minimum values along an axis ignoring NaNs.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.nanargmin.html

Parameters

NdArray	
(Optional)	axis

Returns

NdArray

6.16.2.157 nancumprod()

```
template<typename dtype>
template<typename dtypeOut = double>
static NdArray<dtypeOut> NumC::Methods< dtype >::nancumprod (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE )  [inline], [static]
```

Return the cumulative product of elements along a given axis ignoring NaNs.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.nancumprod.html

Parameters

NdArray	
(Optional)	Axis

Returns

NdArray

6.16.2.158 nancumsum()

```
template<typename dtype>
template<typename dtypeOut = double>
static NdArray<dtypeOut> NumC::Methods< dtype >::nancumsum (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE )  [inline], [static]
```

Return the cumulative sum of the elements along a given axis ignoring NaNs.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.nancumsum.html

Parameters

NdArray	
(Optional)	Axis

Returns

NdArray

6.16.2.159 nanmax()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::nanmax (
```

```
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE )  [inline], [static]
```

Return the maximum of an array or maximum along an axis ignoring NaNs.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.nanmax.html

Parameters

NdArray	
(Optional)	axis

Returns

NdArray

6.16.2.160 nanmean()

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::nanmean (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE )  [inline], [static]
```

Compute the mean along the specified axis ignoring NaNs.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.nanmean.html

Parameters

NdArray	
(Optional)	axis

Returns

NdArray

6.16.2.161 nanmedian()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::nanmedian (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE )  [inline], [static]
```

Compute the median along the specified axis ignoring NaNs.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.nanmedian.html

Parameters

NdArray	
(Optional)	axis

Returns

NdArray

6.16.2.162 nanmin()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::nanmin (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE ) [inline], [static]
```

Return the minimum of an array or maximum along an axis ignoring NaNs.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.nanmin.html

Parameters

NdArray	
(Optional)	axis

Returns

NdArray

6.16.2.163 nanpercentile()

```
template<typename dtype>
template<typename dtypeOut = double>
static NdArray<double> NumC::Methods< dtype >::nanpercentile (
            const NdArray< dtype > & inArray,
            double inPercentile,
            Axis::Type inAxis = Axis::NONE,
            const std::string & inInterpMethod = "linear" ) [inline], [static]
```

Compute the qth percentile of the data along the specified axis, while ignoring nan values.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.nanpercentile.html

Parameters

NdArray	
(Optional)	Axis

Returns

NdArray

6.16.2.164 nanprod()

```
template<typename dtype>
template<typename dtypeOut = double>
static NdArray<dtypeOut> NumC::Methods< dtype >::nanprod (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE ) [inline], [static]
```

Return the product of array elements over a given axis treating Not a Numbers (NaNs) as ones.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩
nanprod.html

Parameters

NdArray	
(Optional)	axis

Returns

NdArray

6.16.2.165 nans() [1/3]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::nans (
            uint32 inSquareSize ) [inline], [static]
```

Return a new array of given shape and type, filled with nans. Only really works for dtype = float/double

Parameters

square	size

Returns

NdArray

6.16.2.166 nans() [2/3]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::nans (
            uint32 inNumRows,
            uint32 inNumCols )  [inline], [static]
```

Return a new array of given shape and type, filled with nans. Only really works for dtype = float/double

Parameters

numRows	
numCols	

Returns

NdArray

6.16.2.167 nans() [3/3]

```
template<typename dtype>
static NumC::NdArray<dtype> NumC::Methods< dtype >::nans (
            const NumC::Shape & inShape )  [inline], [static]
```

Return a new array of given shape and type, filled with nans. Only really works for dtype = float/double

Parameters

Shape	

Returns

NdArray

6.16.2.168 nanstd()

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::nanstd (
```

```
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE )  [inline], [static]
```

Compute the standard deviation along the specified axis, while ignoring NaNs.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↵
nanstd.html

Parameters

NdArray	
(Optional)	axis

Returns

NdArray

6.16.2.169 nansum()

```
template<typename dtype>
template<typename dtypeOut = double>
static NdArray<dtypeOut> NumC::Methods< dtype >::nansum (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE )  [inline], [static]
```

Return the sum of array elements over a given axis treating Not a Numbers (NaNs) as zero.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↵
nansum.html

Parameters

NdArray	
(Optional)	axis

Returns

NdArray

6.16.2.170 nanvar()

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::nanvar (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE )  [inline], [static]
```

Compute the variance along the specified axis, while ignoring NaNs.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↵
nanvar.html

Parameters

NdArray	
(Optional)	axis

Returns

NdArray

6.16.2.171 nbytes()

```
template<typename dtype>
static uint64 NumC::Methods< dtype >::nbytes (
            const NdArray< dtype > & inArray )  [inline], [static]
```

Returns the number of bytes held by the array

Parameters

None	

Returns

number of bytes

6.16.2.172 negative()

```
template<typename dtype>
template<typename dtypeOut = double>
static NdArray<dtypeOut> NumC::Methods< dtype >::negative (
            const NdArray< dtype > & inArray )  [inline], [static]
```

Numerical negative, element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.negative.html

Parameters

NdArray	

Returns

NdArray

6.16.2.173 newbyteorder() [1/2]

```
template<typename dtype>
static dtype NumC::Methods< dtype >::newbyteorder (
            dtype inValue,
            Endian::Type inEndianess ) [inline], [static]
```

Return the array with the same data viewed with a different byte order. only works for integer types, floating point types will not compile and you will be confused as to why...

Parameters

inValue	
Endianess	

Returns

inValue

6.16.2.174 newbyteorder() [2/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::newbyteorder (
            const NdArray< dtype > & inArray,
            Endian::Type inEndianess ) [inline], [static]
```

Return the array with the same data viewed with a different byte order. only works for integer types, floating point types will not compile and you will be confused as to why...

Parameters

NdArray	
Endianess	

Returns

NdArray

6.16.2.175 nonzero()

```
template<typename dtype>
static NdArray<uint32> NumC::Methods< dtype >::nonzero (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Return the indices of the flattened array of the elements that are non-zero.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩
nonzero.html

Parameters

NdArray	

Returns

NdArray

6.16.2.176 norm()

```
template<typename dtype>
template<typename dtypeOut = double>
static NdArray<dtypeOut> NumC::Methods< dtype >::norm (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE ) [inline], [static]
```

Matrix or vector norm.

Parameters

NdArray	
(Optional)	Axis

Returns

NdArray

6.16.2.177 not_equal()

```
template<typename dtype>
static NdArray<bool> NumC::Methods< dtype >::not_equal (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 ) [inline], [static]
```

Return (x1 != x2) element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.not_equal.html

Parameters

NdArray	1
NdArray	2

Returns

NdArray

6.16.2.178 ones() [1/3]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::ones (
            uint32 inSquareSize ) [inline], [static]
```

Return a new array of given shape and type, filled with ones.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ones.html

Parameters

square	size

Returns

NdArray

6.16.2.179 ones() [2/3]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::ones (
            uint32 inNumRows,
            uint32 inNumCols ) [inline], [static]
```

Return a new array of given shape and type, filled with ones.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ones.html

Parameters

numRows	
numCols	

Returns

NdArray

6.16.2.180 ones() [3/3]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::ones (
            const Shape & inShape )  [inline], [static]
```

Return a new array of given shape and type, filled with ones.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ones.html

Parameters

Shape	

Returns

NdArray

6.16.2.181 ones_like()

```
template<typename dtype>
template<typename dtypeOut = double>
static NdArray<dtypeOut> NumC::Methods< dtype >::ones_like (
            const NdArray< dtype > & inArray )  [inline], [static]
```

Return a new array of given shape and type, filled with ones.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ones_like.html

Parameters

NdArray	

Returns

NdArray

6.16.2.182 pad()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::pad (
            const NdArray< dtype > & inArray,
            uint16 inPadWidth,
            dtype inPadValue )  [inline], [static]
```

Pads an array.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.pad.↩
html

Parameters

NdArray	
pad	width
pad	value

Returns

NdArray

6.16.2.183 partition()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::partition (
            const NdArray< dtype > & inArray,
            uint32 inKth,
            Axis::Type inAxis = Axis::NONE ) [inline], [static]
```

Rearranges the elements in the array in such a way that value of the element in kth position is in the position it would be in a sorted array. All elements smaller than the kth element are moved before this element and all equal or greater are moved behind it. The ordering of the elements in the two partitions is undefined.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩
partition.html

Parameters

kth	element
(Optional)	Axis

Returns

NdArray

6.16.2.184 percentile()

```
template<typename dtype>
template<typename dtypeOut = double>
static NdArray<dtypeOut> NumC::Methods< dtype >::percentile (
            const NdArray< dtype > & inArray,
            double inPercentile,
```

```
            Axis::Type inAxis = Axis::NONE,
            const std::string & inInterpMethod = "linear" )  [inline], [static]
```

Compute the qth percentile of the data along the specified axis.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.percentile.html

Parameters

NdArray	
percentile,must	be in the range [0, 100]
(Optional)	axis
(Optional)	interpolation method linear: i + (j - i) ∗ fraction, where fraction is the fractional part of the index surrounded by i and j. lower : i. higher : j. nearest : i or j, whichever is nearest. midpoint : (i + j) / 2.

Returns

NdArray

6.16.2.185 power() [1/2]

```
template<typename dtype>
template<typename dtypeOut = double>
static NdArray<dtypeOut> NumC::Methods< dtype >::power (
            const NdArray< dtype > & inArray,
            uint8 inExponent )  [inline], [static]
```

Raises the elements of the array to the input power

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.power.html

Parameters

NdArray	
exponent	

Returns

NdArray

6.16.2.186 power() [2/2]

```
template<typename dtype>
template<typename dtypeOut = double>
```

```
static NdArray<dtypeOut> NumC::Methods< dtype >::power (
            const NdArray< dtype > & inArray,
            const NdArray< uint8 > & inExponents ) [inline], [static]
```

Raises the elements of the array to the input powers

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.power.html

Parameters

NdArray	
NdArray	

Returns

NdArray

6.16.2.187 print()

```
template<typename dtype>
static void NumC::Methods< dtype >::print (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Prints the array to the console.

Parameters

NdArray	

Returns

None

6.16.2.188 prod()

```
template<typename dtype>
template<typename dtypeOut = double>
static NdArray<dtypeOut> NumC::Methods< dtype >::prod (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE ) [inline], [static]
```

Return the product of array elements over a given axis.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.prod.html

Parameters

NdArray	
(Optional)	axis

Returns

NdArray

6.16.2.189 ptp()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::ptp (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE ) [inline], [static]
```

Range of values (maximum - minimum) along an axis.

NumPy Reference: `https://www.numpy.org/devdocs/reference/generated/numpy.ptp.html`

Parameters

NdArray	
(Optional)	axis

Returns

NdArray

6.16.2.190 put()

```
template<typename dtype>
static NdArray<dtype>& NumC::Methods< dtype >::put (
            NdArray< dtype > & inArray,
            const NdArray< uint32 > & inIndices,
            const NdArray< dtype > & inValues ) [inline], [static]
```

Replaces specified elements of an array with given values. The indexing works on the flattened target array

NumPy Reference: `https://www.numpy.org/devdocs/reference/generated/numpy.put.html`

Parameters

NdArray	
NdArray	of indices
NdArray	of values to put

Returns

NdArray

6.16.2.191 putmask() [1/2]

```
template<typename dtype>
static NdArray<dtype>& NumC::Methods< dtype >::putmask (
            NdArray< dtype > & inArray,
            const NdArray< bool > & inMask,
            dtype inValue ) [inline], [static]
```

Changes elements of an array based on conditional and input values.

Sets a.flat[n] = values[n] for each n where mask.flat[n] == True.

If values is not the same size as a and mask then it will repeat.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.putmask.html

Parameters

NdArray	
NdArray	mask
scalar	value to put

Returns

NdArray

6.16.2.192 putmask() [2/2]

```
template<typename dtype>
static NdArray<dtype>& NumC::Methods< dtype >::putmask (
            NdArray< dtype > & inArray,
            const NdArray< bool > & inMask,
            const NdArray< dtype > & inValues ) [inline], [static]
```

Changes elements of an array based on conditional and input values.

Sets a.flat[n] = values[n] for each n where mask.flat[n] == True.

If values is not the same size as a and mask then it will repeat.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.putmask.html

Parameters

NdArray	
NdArray	mask
NdArray	of values to put

Returns

NdArray

6.16.2.193 rad2deg() [1/2]

```
template<typename dtype>
static double NumC::Methods< dtype >::rad2deg (
            dtype inValue ) [inline], [static]
```

Convert angles from radians to degrees.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.rad2deg.html

Parameters

value	

Returns

value

6.16.2.194 rad2deg() [2/2]

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::rad2deg (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Convert angles from radians to degrees.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.rad2deg.html

Parameters

NdArray	

Returns

NdArray

6.16.2.195 reciprocal()

```
template<typename dtype>
template<typename dtypeOut = double>
static NdArray<dtypeOut> NumC::Methods< dtype >::reciprocal (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Return the reciprocal of the argument, element-wise.

Calculates 1 / x.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩
reciprocal.html

Parameters

NdArray	

Returns

NdArray

6.16.2.196 remainder() [1/2]

```
template<typename dtype>
template<typename dtypeOut = double>
static dtypeOut NumC::Methods< dtype >::remainder (
            dtype inValue1,
            dtype inValue2 ) [inline], [static]
```

Return remainder of division.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩
remainder.html

Parameters

value	1
value	2

Returns

NdArray

6.16.2.197 remainder() [2/2]

```
template<typename dtype>
template<typename dtypeOut = double>
static NdArray<dtypeOut> NumC::Methods< dtype >::remainder (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 )  [inline], [static]
```

Return element-wise remainder of division.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩
remainder.html

Parameters

NdArray	1
NdArray	2

Returns

NdArray

6.16.2.198 repeat() [1/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::repeat (
            const NdArray< dtype > & inArray,
            uint32 inNumRows,
            uint32 inNumCols )  [inline], [static]
```

Repeat elements of an array.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩
repeat.html

Parameters

numRows	
numCols	
Shape	

Returns

NdArray

6.16.2.199 repeat() [2/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::repeat (
            const NdArray< dtype > & inArray,
            const Shape & inRepeatShape ) [inline], [static]
```

Repeat elements of an array.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩ repeat.html

Parameters

NdArray	
Shape	

Returns

NdArray

6.16.2.200 reshape() [1/2]

```
template<typename dtype>
static NdArray<dtype>& NumC::Methods< dtype >::reshape (
            NdArray< dtype > & inArray,
            uint32 inNumRows,
            uint32 inNumCols ) [inline], [static]
```

Gives a new shape to an array without changing its data.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩ reshape.html

Parameters

numRows	
numCols	
Shape,new	Shape

Returns

NdArray

6.16.2.201 reshape() [2/2]

```
template<typename dtype>
static NdArray<dtype>& NumC::Methods< dtype >::reshape (
```

```
            NdArray< dtype > & inArray,
            const Shape & inNewShape )  [inline], [static]
```

Gives a new shape to an array without changing its data.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.reshape.html

Parameters

NdArray	
Shape,new	Shape

Returns

NdArray

6.16.2.202 resizeFast() [1/2]

```
template<typename dtype>
static NdArray<dtype>& NumC::Methods< dtype >::resizeFast (
            NdArray< dtype > & inArray,
            uint32 inNumRows,
            uint32 inNumCols )  [inline], [static]
```

Change shape and size of array in-place. All previous data of the array is lost.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.resize.html

Parameters

NdArray	
numRows	
numCols	

Returns

NdArray

6.16.2.203 resizeFast() [2/2]

```
template<typename dtype>
static NdArray<dtype>& NumC::Methods< dtype >::resizeFast (
            NdArray< dtype > & inArray,
            const Shape & inNewShape )  [inline], [static]
```

Change shape and size of array in-place. All previous data of the array is lost.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↵resize.html

Parameters

NdArray	
Shape,*new*	Shape

Returns

NdArray

6.16.2.204 resizeSlow() [1/2]

```
template<typename dtype>
static NdArray<dtype>& NumC::Methods< dtype >::resizeSlow (
            NdArray< dtype > & inArray,
            uint32 inNumRows,
            uint32 inNumCols ) [inline], [static]
```

Return a new array with the specified shape. If new shape is larger than old shape then array will be padded with zeros. If new shape is smaller than the old shape then the data will be discarded.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↵resize.html

Parameters

NdArray	
numRows	
numCols	

Returns

NdArray

6.16.2.205 resizeSlow() [2/2]

```
template<typename dtype>
static NdArray<dtype>& NumC::Methods< dtype >::resizeSlow (
            NdArray< dtype > & inArray,
            const Shape & inNewShape ) [inline], [static]
```

Return a new array with the specified shape. If new shape is larger than old shape then array will be padded with zeros. If new shape is smaller than the old shape then the data will be discarded.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↵resize.html

Parameters

NdArray	
Shape,new	Shape

Returns

NdArray

6.16.2.206 right_shift()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::right_shift (
            const NdArray< dtype > & inArray,
            uint8 inNumBits )  [inline], [static]
```

Shift the bits of an integer to the right.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.right←_shift.html

Parameters

NdArray	
number	of bits to sift

Returns

NdArray

6.16.2.207 rint() [1/2]

```
template<typename dtype>
static dtype NumC::Methods< dtype >::rint (
            dtype inValue )  [inline], [static]
```

Round value to the nearest integer.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.rint.←html

Parameters

value	

Returns

value

6.16.2.208 rint() [2/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::rint (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Round elements of the array to the nearest integer.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.rint.html

Parameters

NdArray	

Returns

NdArray

6.16.2.209 roll()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::roll (
            const NdArray< dtype > & inArray,
            int32 inShift,
            Axis::Type inAxis = Axis::NONE ) [inline], [static]
```

Roll array elements along a given axis.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.roll.html

Parameters

NdArray	
elements	to shift, positive means forward, negative means backwards
(Optional)	axis

Returns

NdArray

6.16.2.210 rot90()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::rot90 (
            const NdArray< dtype > & inArray,
            uint8 inK = 1 ) [inline], [static]
```

Rotate an array by 90 degrees counter clockwise in the plane.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.rot90.html

Parameters

NdArray	
the	number of times to rotate 90 degrees

Returns

NdArray

6.16.2.211 round() [1/2]

```
template<typename dtype>
static dtype NumC::Methods< dtype >::round (
            dtype inValue,
            uint8 inDecimals ) [inline], [static]
```

Round value to the given number of decimals.

Parameters

value	
the	number of decimals

Returns

value

6.16.2.212 round() [2/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::round (
            const NdArray< dtype > & inArray,
            uint8 inDecimals ) [inline], [static]
```

Round an array to the given number of decimals.

Parameters

NdArray	
the	number of decimals

Returns

NdArray

6.16.2.213 row_stack()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::row_stack (
            const std::initializer_list< NdArray< dtype > > & inArrayList ) [inline], [static]
```

Stack arrays in sequence vertically (row wise).

Parameters

{list}	of arrays to stack

Returns

NdArray

6.16.2.214 setdiff1d()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::setdiff1d (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 ) [inline], [static]
```

Find the set difference of two arrays.

Return the sorted, unique values in ar1 that are not in ar2.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.setdiff1d.html

Parameters

NdArray	1
NdArray	2

Returns

NdArray

6.16.2.215 shape()

```
template<typename dtype>
static Shape NumC::Methods< dtype >::shape (
            const NdArray< dtype > & inArray )  [inline], [static]
```

Return the shape of the array

Parameters

NdArray	

Returns

Shape

6.16.2.216 sign() [1/2]

```
template<typename dtype>
static int8 NumC::Methods< dtype >::sign (
            dtype inValue )  [inline], [static]
```

Returns an element-wise indication of the sign of a number.

The sign function returns - 1 if x < 0, 0 if x == 0, 1 if x > 0. nan is returned for nan inputs.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.sign.html

Parameters

NdArray	

Returns

NdArray

6.16.2.217 sign() [2/2]

```
template<typename dtype>
static NdArray<int8> NumC::Methods< dtype >::sign (
            const NdArray< dtype > & inArray )  [inline], [static]
```

Returns an element-wise indication of the sign of a number.

The sign function returns - 1 if x < 0, 0 if x == 0, 1 if x > 0. nan is returned for nan inputs.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.sign.html

Parameters

NdArray	

Returns

NdArray

6.16.2.218 signbit() [1/2]

```
template<typename dtype>
static bool NumC::Methods< dtype >::signbit (
            dtype inValue )  [inline], [static]
```

Returns element-wise True where signbit is set (less than zero).

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.signbit.html

Parameters

NdArray	

Returns

NdArray

6.16.2.219 signbit() [2/2]

```
template<typename dtype>
static NdArray<bool> NumC::Methods< dtype >::signbit (
            const NdArray< dtype > & inArray )  [inline], [static]
```

Returns element-wise True where signbit is set (less than zero).

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.signbit.html

Parameters

NdArray	

Returns

NdArray

6.16.2.220 sin() [1/2]

```
template<typename dtype>
static double NumC::Methods< dtype >::sin (
            dtype inValue )  [inline], [static]
```

Trigonometric sine.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.sin.html

Parameters

value	

Returns

value

6.16.2.221 sin() [2/2]

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::sin (
            const NdArray< dtype > & inArray )  [inline], [static]
```

Trigonometric sine, element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.sin.html

Parameters

NdArray	

Returns

NdArray

6.16.2.222 sinc() [1/2]

```
template<typename dtype>
static double NumC::Methods< dtype >::sinc (
            dtype inValue ) [inline], [static]
```

Return the sinc function.

The sinc function is sin(pi∗x) / (pi∗x).

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.sinc.html

Parameters

value	

Returns

value

6.16.2.223 sinc() [2/2]

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::sinc (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Return the sinc function.

The sinc function is sin(pi∗x) / (pi∗x).

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.sinc.html

Parameters

NdArray	

Returns

NdArray

6.16.2.224 sinh() [1/2]

```
template<typename dtype>
static double NumC::Methods< dtype >::sinh (
            dtype inValue )  [inline], [static]
```

Hyperbolic sine.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.sinh.html

Parameters

value	

Returns

value

6.16.2.225 sinh() [2/2]

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::sinh (
            const NdArray< dtype > & inArray )  [inline], [static]
```

Hyperbolic sine, element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.sinh.html

Parameters

NdArray	

Returns

NdArray

6.16.2.226 size()

```
template<typename dtype>
static uint32 NumC::Methods< dtype >::size (
            const NdArray< dtype > & inArray )  [inline], [static]
```

Return the number of elements.

Parameters

uint32	

Returns

NdArray

6.16.2.227 sort()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::sort (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE ) [inline], [static]
```

Return a sorted copy of an array.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.sort.↩
html

Parameters

NdArray	
(Optional)	Axis

Returns

NdArray

6.16.2.228 sqrt() [1/2]

```
template<typename dtype>
static double NumC::Methods< dtype >::sqrt (
            dtype inValue ) [inline], [static]
```

Return the positive square-root of a value.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.sqrt.↩
html

Parameters

value	

Returns

value

6.16.2.229 sqrt() [2/2]

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::sqrt (
            const NdArray< dtype > & inArray )  [inline], [static]
```

Return the positive square-root of an array, element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.sqrt.html

Parameters

NdArray	

Returns

NdArray

6.16.2.230 square() [1/2]

```
template<typename dtype>
static dtype NumC::Methods< dtype >::square (
            dtype inValue )  [inline], [static]
```

Return the square of an array.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.square.html

Parameters

value	

Returns

value

6.16.2.231 square() [2/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::square (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Return the square of an array, element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.square.html

Parameters

NdArray	

Returns

NdArray

6.16.2.232 std()

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::std (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE ) [inline], [static]
```

Compute the standard deviation along the specified axis.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.std.html

Parameters

NdArray	
Axis	

Returns

NdArray

6.16.2.233 sum()

```
template<typename dtype>
template<typename dtypeOut = double>
static NdArray<dtypeOut> NumC::Methods< dtype >::sum (
```

```
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE )  [inline], [static]
```

Sum of array elements over a given axis.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.sum.html

Parameters

NdArray	
Axis	

Returns

NdArray

6.16.2.234 swapaxes()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::swapaxes (
            const NdArray< dtype > & inArray )  [inline], [static]
```

Interchange two axes of an array.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.swapaxes.html

Parameters

NdArray	

Returns

NdArray

6.16.2.235 tan() [1/2]

```
template<typename dtype>
static double NumC::Methods< dtype >::tan (
            dtype inValue )  [inline], [static]
```

Compute tangent.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.tan.html

Parameters

value	

Returns

value

6.16.2.236 tan() [2/2]

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::tan (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Compute tangent element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.tan.html

Parameters

NdArray	

Returns

NdArray

6.16.2.237 tanh() [1/2]

```
template<typename dtype>
static double NumC::Methods< dtype >::tanh (
            dtype inValue ) [inline], [static]
```

Compute hyperbolic tangent.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.tanh.html

Parameters

value	

Returns

value

6.16.2.238 tanh() [2/2]

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::tanh (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Compute hyperbolic tangent element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.tanh.html

Parameters

NdArray	

Returns

NdArray

6.16.2.239 tile() [1/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::tile (
            const NdArray< dtype > & inArray,
            uint32 inNumRows,
            uint32 inNumCols ) [inline], [static]
```

Construct an array by repeating A the number of times given by reps.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.tile.html

Parameters

NdArray	
numRows	
numCols	

Returns

NdArray

6.16.2.240 tile() [2/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::tile (
            const NdArray< dtype > & inArray,
            const Shape & inReps ) [inline], [static]
```

Construct an array by repeating A the number of times given by reps.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.tile.↩html

Parameters

NdArray	
Shape	

Returns

NdArray

6.16.2.241 tofile()

```
template<typename dtype>
static void NumC::Methods< dtype >::tofile (
            const NdArray< dtype > & inArray,
            const std::string & inFilename,
            const std::string & inSep = "" ) [inline], [static]
```

Write array to a file as text or binary (default).. The data produced by this method can be recovered using the function fromfile().

Parameters

NdArray	
filename	
Separator	between array items for text output. If ŞŤ (empty), a binary file is written

Returns

None

6.16.2.242 toStlVector()

```
template<typename dtype>
static std::vector<dtype> NumC::Methods< dtype >::toStlVector (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Write flattened array to an STL vector

Parameters

NdArray	

Returns

std::vector

6.16.2.243 trace()

```
template<typename dtype>
template<typename dtypeOut = double>
static dtypeOut NumC::Methods< dtype >::trace (
            const NdArray< dtype > & inArray,
            uint16 inOffset = 0,
            Axis::Type inAxis = Axis::ROW ) [inline], [static]
```

Return the sum along diagonals of the array.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.trace.html

Parameters

NdArray	
Offset	from main diaganol, default = 0, negative=above, positve=below
Axis	

Returns

NdArray

6.16.2.244 transpose()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::transpose (
            const NdArray< dtype > & inArray ) [inline], [static]
```

Permute the dimensions of an array.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.transpose.html

Parameters

NdArray	

Returns

NdArray

6.16.2.245 trapz() [1/2]

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::trapz (
            const NdArray< dtype > & inArray,
            double dx = 1.0,
            Axis::Type inAxis = Axis::NONE ) [inline], [static]
```

Integrate along the given axis using the composite trapezoidal rule.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.trapz.html

Parameters

NdArray	
(Optional)	dx, defaults to 1.0
(Optional)	Axis, default None

Returns

NdArray

6.16.2.246 trapz() [2/2]

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::trapz (
            const NdArray< dtype > & inArrayY,
            const NdArray< dtype > & inArrayX,
            Axis::Type inAxis = Axis::NONE ) [inline], [static]
```

Integrate along the given axis using the composite trapezoidal rule.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.trapz.html

Parameters

NdArray	Y values
NdArray	X values
(Optional)	Axis

Returns

NdArray

6.16.2.247 tri() [1/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::tri (
            uint32 inN,
            int32 inOffset = 0 ) [inline], [static]
```

An array with ones at and below the given diagonal and zeros elsewhere.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.tri.↩ html

Parameters

N,number	of rows and cols
Offset,the	sub-diagonal at and below which the array is filled. k = 0 is the main diagonal, while k < 0 is below it, and k > 0 is above. The default is 0.

Returns

NdArray

6.16.2.248 tri() [2/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::tri (
            uint32 inN,
            uint32 inM,
            int32 inOffset = 0 ) [inline], [static]
```

An array with ones at and below the given diagonal and zeros elsewhere.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.tri.↩ html

Parameters

N,number	of rows
M,number	of columns
Offset,the	sub-diagonal at and below which the array is filled. k = 0 is the main diagonal, while k < 0 is below it, and k > 0 is above. The default is 0.

Returns

NdArray

6.16.2.249 trim_zeros()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::trim_zeros (
            const NdArray< dtype > & inArray,
            const std::string inTrim = "fb" )  [inline], [static]
```

Trim the leading and/or trailing zeros from a 1-D array or sequence.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.trim_zeros.html

Parameters

NdArray	
string,f	= front, "b" = back, "fb" = front and back

Returns

NdArray

6.16.2.250 trunc() [1/2]

```
template<typename dtype>
static dtype NumC::Methods< dtype >::trunc (
            dtype inValue )  [inline], [static]
```

Return the truncated value of the input.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.trunc.html

Parameters

value	

Returns

value

6.16.2.251 trunc() [2/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::trunc (
            const NdArray< dtype > & inArray )  [inline], [static]
```

Return the truncated value of the input, element-wise.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.trunc.html

Parameters

NdArray	

Returns

NdArray

6.16.2.252 union1d()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::union1d (
            const NdArray< dtype > & inArray1,
            const NdArray< dtype > & inArray2 )  [inline], [static]
```

Find the union of two arrays.

Return the unique, sorted array of values that are in either of the two input arrays.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.union1d.html

Parameters

NdArray	1
NdArray	2

Returns

NdArray

6.16.2.253 unique()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::unique (
            const NdArray< dtype > & inArray )  [inline], [static]
```

Find the unique elements of an array.

Returns the sorted unique elements of an array.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩
unique.html

Parameters

NdArray	

Returns

NdArray

6.16.2.254 unwrap() [1/2]

```
template<typename dtype>
static dtype NumC::Methods< dtype >::unwrap (
            dtype inValue )  [inline], [static]
```

Unwrap by changing deltas between values to 2∗pi complement.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩
unwrap.html

Parameters

value	

Returns

value

6.16.2.255 unwrap() [2/2]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::unwrap (
            const NdArray< dtype > & inArray )  [inline], [static]
```

Unwrap by changing deltas between values to 2∗pi complement.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩
unwrap.html

Parameters

NdArray	

Returns

NdArray

6.16.2.256 var()

```
template<typename dtype>
static NdArray<double> NumC::Methods< dtype >::var (
            const NdArray< dtype > & inArray,
            Axis::Type inAxis = Axis::NONE ) [inline], [static]
```

Compute the variance along the specified axis.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.var.html

Parameters

NdArray	
(Optional)	axis

Returns

NdArray

6.16.2.257 vstack()

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::vstack (
            const std::initializer_list< NdArray< dtype > > & inArrayList ) [inline], [static]
```

Compute the variance along the specified axis.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.vstack.html

Parameters

{list}	of arrays to stack

Returns

NdArray

6.16.2.258 zeros() [1/3]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::zeros (
            uint32 inSquareSize ) [inline], [static]
```

Return a new array of given shape and type, filled with zeros.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.zeros.html

Parameters

square	size

Returns

NdArray

6.16.2.259 zeros() [2/3]

```
template<typename dtype>
static NdArray<dtype> NumC::Methods< dtype >::zeros (
            uint32 inNumRows,
            uint32 inNumCols ) [inline], [static]
```

Return a new array of given shape and type, filled with zeros.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.zeros.html

Parameters

numRows	
numCols	

Returns

NdArray

6.16.2.260 zeros() [3/3]

```
template<typename dtype>
static NumC::NdArray<dtype> NumC::Methods< dtype >::zeros (
            const NumC::Shape & inShape ) [inline], [static]
```

Return a new array of given shape and type, filled with zeros.

NumPy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.zeros.html

Parameters

Shape	

Returns

NdArray

The documentation for this class was generated from the following file:

- Methods.hpp

6.17 NumC::NdArray< dtype > Class Template Reference

Holds 1D and 2D arrays, the main work horse of the NumC library.

```
#include <NdArray.hpp>
```

Public Types

- typedef const dtype ∗ const_iterator
- typedef dtype ∗ iterator

Public Member Functions

- NdArray ()
- NdArray (uint32 inSquareSize)
- NdArray (uint32 inNumRows, uint32 inNumCols)
- NdArray (const Shape &inShape)
- NdArray (const std::initializer_list< dtype > &inList)
- NdArray (const std::initializer_list< std::initializer_list< dtype > > &inList)
- NdArray (const std::vector< dtype > &inVector)
- NdArray (const std::set< dtype > &inSet)
- NdArray (const_iterator inFirst, const_iterator inLast)
- NdArray (const dtype ∗inBeginning, uint32 inNumBytes)
- NdArray (const NdArray< dtype > &inOtherArray)
- NdArray (NdArray< dtype > &&inOtherArray)
- ∼NdArray ()

- NdArray< bool > all (Axis::Type inAxis=Axis::NONE) const
- NdArray< bool > any (Axis::Type inAxis=Axis::NONE) const
- NdArray< uint32 > argmax (Axis::Type inAxis=Axis::NONE) const
- NdArray< uint32 > argmin (Axis::Type inAxis=Axis::NONE) const
- NdArray< uint32 > argsort (Axis::Type inAxis=Axis::NONE) const
- template<typename dtypeOut = double>
  NdArray< dtypeOut > astype () const
- dtype & at (int32 inIndex)
- const dtype & at (int32 inIndex) const
- dtype & at (int32 inRowIndex, int32 inColIndex)
- const dtype & at (int32 inRowIndex, int32 inColIndex) const
- NdArray< dtype > at (const Slice &inSlice) const
- NdArray< dtype > at (const Slice &inRowSlice, const Slice &inColSlice) const
- NdArray< dtype > at (const Slice &inRowSlice, int32 inColIndex) const
- NdArray< dtype > at (int32 inRowIndex, const Slice &inColSlice) const
- iterator begin ()
- iterator begin (uint32 inRow)
- void byteswap ()
- const_iterator cbegin () const
- const_iterator cbegin (uint32 inRow) const
- const_iterator cend () const
- const_iterator cend (uint32 inRow) const
- NdArray< dtype > clip (dtype inMin, dtype inMax) const
- NdArray< bool > contains (dtype inValue, Axis::Type inAxis=Axis::NONE) const
- NdArray< dtype > copy ()
- template<typename dtypeOut = double>
  NdArray< dtypeOut > cumprod (Axis::Type inAxis=Axis::NONE) const
- template<typename dtypeOut = double>
  NdArray< dtypeOut > cumsum (Axis::Type inAxis=Axis::NONE) const
- NdArray< dtype > diagonal (uint32 inOffset=0, Axis::Type inAxis=Axis::ROW) const
- template<typename dtypeOut = double>
  NdArray< dtypeOut > dot (const NdArray< dtype > &inOtherArray) const
- void dump (const std::string &inFilename) const
- iterator end ()
- iterator end (uint32 inRow)
- Endian::Type endianess () const
- void fill (dtype inFillValue)
- NdArray< dtype > flatten () const
- bool isempty () const
- dtype item () const
- NdArray< dtype > max (Axis::Type inAxis=Axis::NONE) const
- NdArray< double > mean (Axis::Type inAxis=Axis::NONE) const
- NdArray< dtype > median (Axis::Type inAxis=Axis::NONE) const
- NdArray< dtype > min (Axis::Type inAxis=Axis::NONE) const
- void nans ()
- uint64 nbytes () const
- NdArray< dtype > newbyteorder (Endian::Type inEndianess) const
- NdArray< uint32 > nonzero () const
- template<typename dtypeOut = double>
  NdArray< dtypeOut > norm (Axis::Type inAxis=Axis::NONE) const
- void ones ()
- NdArray< dtype > operator & (const NdArray< dtype > &inOtherArray) const
- NdArray< dtype > operator & (dtype inScalar) const
- NdArray< dtype > & operator &= (const NdArray< dtype > &inOtherArray)
- NdArray< dtype > & operator &= (dtype inScalar)

- NdArray< bool > operator!= (dtype inValue) const
- NdArray< bool > operator!= (const NdArray< dtype > &inOtherArray) const
- NdArray< dtype > operator% (const NdArray< dtype > &inOtherArray) const
- NdArray< dtype > operator% (dtype inScalar) const
- NdArray< dtype > & operator%= (const NdArray< dtype > &inOtherArray)
- NdArray< dtype > & operator%= (dtype inScalar)
- dtype & operator() (int32 inRowIndex, int32 inColIndex)
- const dtype & operator() (int32 inRowIndex, int32 inColIndex) const
- NdArray< dtype > operator() (const Slice &inRowSlice, const Slice &inColSlice) const
- NdArray< dtype > operator() (const Slice &inRowSlice, int32 inColIndex) const
- NdArray< dtype > operator() (int32 inRowIndex, const Slice &inColSlice) const
- NdArray< dtype > operator* (const NdArray< dtype > &inOtherArray) const
- NdArray< dtype > operator* (dtype inScalar) const
- NdArray< dtype > & operator*= (const NdArray< dtype > &inOtherArray)
- NdArray< dtype > & operator*= (dtype inScalar)
- NdArray< dtype > operator+ (const NdArray< dtype > &inOtherArray) const
- NdArray< dtype > operator+ (dtype inScalar) const
- NdArray< dtype > & operator++ ()
- NdArray< dtype > operator++ (int) const
- NdArray< dtype > & operator+= (const NdArray< dtype > &inOtherArray)
- NdArray< dtype > & operator+= (dtype inScalar)
- NdArray< dtype > operator- (const NdArray< dtype > &inOtherArray) const
- NdArray< dtype > operator- (dtype inScalar) const
- NdArray< dtype > & operator-- ()
- NdArray< dtype > operator-- (int) const
- NdArray< dtype > & operator-= (const NdArray< dtype > &inOtherArray)
- NdArray< dtype > & operator-= (dtype inScalar)
- NdArray< dtype > operator/ (const NdArray< dtype > &inOtherArray) const
- NdArray< dtype > operator/ (dtype inScalar) const
- NdArray< dtype > & operator/= (const NdArray< dtype > &inOtherArray)
- NdArray< dtype > & operator/= (dtype inScalar)
- NdArray< bool > operator< (const NdArray< dtype > &inOtherArray) const
- NdArray< bool > operator< (dtype inScalar) const
- NdArray< bool > operator<= (dtype inScalar) const
- NdArray< bool > operator<= (const NdArray< dtype > &inOtherArray) const
- NdArray< dtype > & operator= (const NdArray< dtype > &inOtherArray)
- NdArray< dtype > & operator= (NdArray< dtype > &&inOtherArray)
- NdArray< bool > operator== (dtype inValue) const
- NdArray< bool > operator== (const NdArray< dtype > &inOtherArray) const
- NdArray< bool > operator> (dtype inScalar) const
- NdArray< bool > operator> (const NdArray< dtype > &inOtherArray) const
- NdArray< bool > operator>= (dtype inScalar) const
- NdArray< bool > operator>= (const NdArray< dtype > &inOtherArray) const
- dtype & operator[ ] (int32 inIndex)
- const dtype & operator[ ] (int32 inIndex) const
- NdArray< dtype > operator[ ] (const Slice &inSlice) const
- NdArray< dtype > operator^ (const NdArray< dtype > &inOtherArray) const
- NdArray< dtype > operator^ (dtype inScalar) const
- NdArray< dtype > & operator^= (const NdArray< dtype > &inOtherArray)
- NdArray< dtype > & operator^= (dtype inScalar)
- NdArray< dtype > operator| (const NdArray< dtype > &inOtherArray) const
- NdArray< dtype > operator| (dtype inScalar) const
- NdArray< dtype > & operator|= (const NdArray< dtype > &inOtherArray)
- NdArray< dtype > & operator|= (dtype inScalar)
- NdArray< dtype > operator~ () const

- void partition (uint32 inKth, Axis::Type inAxis=Axis::NONE)
- void print () const
- template<typename dtypeOut = double>
  NdArray< dtypeOut > prod (Axis::Type inAxis=Axis::NONE) const
- NdArray< dtype > ptp (Axis::Type inAxis=Axis::NONE) const
- void put (int32 inIndex, dtype inValue)
- void put (int32 inRow, int32 inCol, dtype inValue)
- void put (const NdArray< uint32 > &inIndices, dtype inValue)
- void put (const NdArray< uint32 > &inIndices, const NdArray< dtype > &inValues)
- void put (const Slice &inSlice, dtype inValue)
- void put (const Slice &inSlice, const NdArray< dtype > &inValues)
- void put (const Slice &inRowSlice, const Slice &inColSlice, dtype inValue)
- void put (const Slice &inRowSlice, int32 inColIndex, dtype inValue)
- void put (int32 inRowIndex, const Slice &inColSlice, dtype inValue)
- void put (const Slice &inRowSlice, const Slice &inColSlice, const NdArray< dtype > &inValues)
- void put (const Slice &inRowSlice, int32 inColIndex, const NdArray< dtype > &inValues)
- void put (int32 inRowIndex, const Slice &inColSlice, const NdArray< dtype > &inValues)
- NdArray< dtype > repeat (uint32 inNumRows, uint32 inNumCols) const
- NdArray< dtype > repeat (const Shape &inRepeatShape) const
- void reshape (uint32 inNumRows, uint32 inNumCols)
- void reshape (const Shape &inShape)
- void resizeFast (uint32 inNumRows, uint32 inNumCols)
- void resizeFast (const Shape &inShape)
- void resizeSlow (uint32 inNumRows, uint32 inNumCols)
- void resizeSlow (const Shape &inShape)
- NdArray< dtype > round (uint8 inNumDecimals=0) const
- Shape shape () const
- uint32 size () const
- void sort (Axis::Type inAxis=Axis::NONE)
- NdArray< double > std (Axis::Type inAxis=Axis::NONE) const
- std::string str () const
- template<typename dtypeOut = double>
  NdArray< dtypeOut > sum (Axis::Type inAxis=Axis::NONE) const
- NdArray< dtype > swapaxes () const
- void tofile (const std::string &inFilename, const std::string &inSep="") const
- std::vector< dtype > toStlVector () const
- template<typename dtypeOut = double>
  dtypeOut trace (uint16 inOffset=0, Axis::Type inAxis=Axis::ROW) const
- NdArray< dtype > transpose () const
- NdArray< double > var (Axis::Type inAxis=Axis::NONE) const
- void zeros ()

Friends

- NdArray< dtype > operator<< (const NdArray< dtype > &lhs, uint8 inNumBits)
- std::ostream & operator<< (std::ostream &inOStream, const NdArray< dtype > &inArray)
- NdArray< dtype > & operator<<= (NdArray< dtype > &lhs, uint8 inNumBits)
- NdArray< dtype > operator>> (const NdArray< dtype > &lhs, uint8 inNumBits)
- NdArray< dtype > & operator>>= (NdArray< dtype > &lhs, uint8 inNumBits)

6.17.1 Detailed Description

template<typename dtype>
class NumC::NdArray< dtype >

Holds 1D and 2D arrays, the main work horse of the NumC library.

6.17.2 Member Typedef Documentation

6.17.2.1 const_iterator

```
template<typename dtype>
typedef const dtype* NumC::NdArray< dtype >::const_iterator
```

6.17.2.2 iterator

```
template<typename dtype>
typedef dtype* NumC::NdArray< dtype >::iterator
```

6.17.3 Constructor & Destructor Documentation

6.17.3.1 NdArray() [1/12]

```
template<typename dtype>
NumC::NdArray< dtype >::NdArray ( )  [inline]
```

Defualt Constructor, not very usefull...

Parameters

None	

Returns

None

6.17.3.2 NdArray() [2/12]

```
template<typename dtype>
NumC::NdArray< dtype >::NdArray (
            uint32 inSquareSize )  [inline], [explicit]
```

Constructor

Parameters

square	number of rows and columns

Returns

None

6.17.3.3 NdArray() [3/12]

```
template<typename dtype>
NumC::NdArray< dtype >::NdArray (
            uint32 inNumRows,
            uint32 inNumCols )  [inline]
```

Constructor

Parameters

number	of rows,
number	of columns

Returns

None

6.17.3.4 NdArray() [4/12]

```
template<typename dtype>
NumC::NdArray< dtype >::NdArray (
            const Shape & inShape )  [inline], [explicit]
```

Constructor

Parameters

Shape	

Returns

None

6.17.3.5 NdArray() [5/12]

```
template<typename dtype>
NumC::NdArray< dtype >::NdArray (
            const std::initializer_list< dtype > & inList )  [inline]
```

Constructor

Parameters

1D	initializer list

Returns

None

6.17.3.6 NdArray() [6/12]

```
template<typename dtype>
NumC::NdArray< dtype >::NdArray (
            const std::initializer_list< std::initializer_list< dtype > > & inList )  [inline]
```

Constructor

Parameters

2D	initializer list

Returns

None

6.17.3.7 NdArray() [7/12]

```
template<typename dtype>
NumC::NdArray< dtype >::NdArray (
            const std::vector< dtype > & inVector )  [inline], [explicit]
```

Constructor

Parameters

std::vector	

Returns

None

6.17.3.8 NdArray() [8/12]

```
template<typename dtype>
NumC::NdArray< dtype >::NdArray (
            const std::set< dtype > & inSet )  [inline], [explicit]
```

Constructor

Parameters

std::set	

Returns

None

6.17.3.9 NdArray() [9/12]

```
template<typename dtype>
NumC::NdArray< dtype >::NdArray (
            const_iterator inFirst,
            const_iterator inLast )  [inline], [explicit]
```

Constructor

Parameters

const_iterator	first
const_iterator	second

Returns

None

6.17.3.10 NdArray() [10/12]

```
template<typename dtype>
NumC::NdArray< dtype >::NdArray (
            const dtype * inBeginning,
            uint32 inNumBytes )  [inline]
```

Constructor

Parameters

*char**	to beginning of buffer
number	of bytes

Returns

None

6.17.3.11 NdArray() [11/12]

```
template<typename dtype>
NumC::NdArray< dtype >::NdArray (
            const NdArray< dtype > & inOtherArray )  [inline]
```

Copy Constructor

Parameters

NdArray	

Returns

None

6.17.3.12 NdArray() [12/12]

```
template<typename dtype>
NumC::NdArray< dtype >::NdArray (
            NdArray< dtype > && inOtherArray )  [inline]
```

Move Constructor

Parameters

NdArray	

Returns

None

6.17.3.13 ∼NdArray()

```
template<typename dtype>
NumC::NdArray< dtype >::∼NdArray ( ) [inline]
```

Destructor

Parameters

None	

Returns

None

6.17.4 Member Function Documentation

6.17.4.1 all()

```
template<typename dtype>
NdArray<bool> NumC::NdArray< dtype >::all (
            Axis::Type inAxis = Axis::NONE ) const [inline]
```

Returns True if all elements evaluate to True or non zero

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.all.html

Parameters

(Optional)	axis

Returns

NdArray

6.17.4.2 any()

```
template<typename dtype>
NdArray<bool> NumC::NdArray< dtype >::any (
            Axis::Type inAxis = Axis::NONE ) const  [inline]
```

Returns True if any elements evaluate to True or non zero

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.any.html

Parameters

(Optional)	axis

Returns

NdArray

6.17.4.3 argmax()

```
template<typename dtype>
NdArray<uint32> NumC::NdArray< dtype >::argmax (
            Axis::Type inAxis = Axis::NONE ) const  [inline]
```

Return indices of the maximum values along the given axis. Only the first index is returned.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.argmax.html

Parameters

(Optional)	axis

Returns

NdArray

6.17.4.4 argmin()

```
template<typename dtype>
NdArray<uint32> NumC::NdArray< dtype >::argmin (
            Axis::Type inAxis = Axis::NONE ) const  [inline]
```

Return indices of the minimum values along the given axis. Only the first index is returned.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.argmin.html

Parameters

(Optional)	axis

Returns

NdArray

6.17.4.5 argsort()

```
template<typename dtype>
NdArray<uint32> NumC::NdArray< dtype >::argsort (
            Axis::Type inAxis = Axis::NONE ) const  [inline]
```

Returns the indices that would sort this array.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩ ndarray.argsort.html

Parameters

(Optional)	axis

Returns

NdArray

6.17.4.6 astype()

```
template<typename dtype>
template<typename dtypeOut = double>
NdArray<dtypeOut> NumC::NdArray< dtype >::astype ( ) const  [inline]
```

Returns a copy of the array, cast to a specified type.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩ ndarray.astype.html

Parameters

None	

Returns

NdArray

6.17.4.7 at() [1/8]

```
template<typename dtype>
dtype& NumC::NdArray< dtype >::at (
            int32 inIndex )  [inline]
```

1D access method with bounds checking

Parameters

array	index

Returns

value

6.17.4.8 at() [2/8]

```
template<typename dtype>
const dtype& NumC::NdArray< dtype >::at (
            int32 inIndex ) const  [inline]
```

const 1D access method with bounds checking

Parameters

array	index

Returns

value

6.17.4.9 at() [3/8]

```
template<typename dtype>
dtype& NumC::NdArray< dtype >::at (
            int32 inRowIndex,
            int32 inColIndex )  [inline]
```

2D access method with bounds checking

Parameters

row	index
col	index

Returns

value

6.17.4.10 at() [4/8]

```
template<typename dtype>
const dtype& NumC::NdArray< dtype >::at (
            int32 inRowIndex,
            int32 inColIndex ) const  [inline]
```

const 2D access method with bounds checking

Parameters

row	index
col	index

Returns

value

6.17.4.11 at() [5/8]

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::at (
            const Slice & inSlice ) const  [inline]
```

const 1D access method with bounds checking

Parameters

Slice	

Returns

Ndarray

6.17.4.12 at() [6/8]

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::at (
            const Slice & inRowSlice,
            const Slice & inColSlice ) const [inline]
```

const 2D access method with bounds checking

Parameters

Row	Slice,
Column	Slice

Returns

Ndarray

6.17.4.13 at() [7/8]

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::at (
            const Slice & inRowSlice,
            int32 inColIndex ) const [inline]
```

const 2D access method with bounds checking

Parameters

Row	Slice,
Column	index

Returns

Ndarray

6.17.4.14 at() [8/8]

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::at (
            int32 inRowIndex,
            const Slice & inColSlice ) const [inline]
```

const 2D access method with bounds checking

Parameters

Row	index
Column	Slice

Returns

Ndarray

6.17.4.15 begin() [1/2]

```
template<typename dtype>
iterator NumC::NdArray< dtype >::begin ( )  [inline]
```

iterator to the beginning of the flattened array

Parameters

None	

Returns

iterator

6.17.4.16 begin() [2/2]

```
template<typename dtype>
iterator NumC::NdArray< dtype >::begin (
            uint32 inRow )  [inline]
```

iterator to the beginning of the input row

Parameters

row	

Returns

iterator

6.17.4.17 byteswap()

```
template<typename dtype>
void NumC::NdArray< dtype >::byteswap ( )  [inline]
```

Swap the bytes of the array elements in place

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.byteswap.html

Parameters

None	

Returns

NdArray

6.17.4.18 cbegin() [1/2]

```
template<typename dtype>
const_iterator NumC::NdArray< dtype >::cbegin ( ) const  [inline]
```

const iterator to the beginning of the flattened array

Parameters

None	

Returns

const_iterator

6.17.4.19 cbegin() [2/2]

```
template<typename dtype>
const_iterator NumC::NdArray< dtype >::cbegin (
            uint32 inRow ) const  [inline]
```

const iterator to the beginning of the input row

Parameters

row	

Returns

const_iterator

6.17.4.20 cend() [1/2]

```
template<typename dtype>
const_iterator NumC::NdArray< dtype >::cend ( ) const  [inline]
```

const iterator to 1 past the end of the flattened array

Parameters

None	

Returns

const_iterator

6.17.4.21 cend() [2/2]

```
template<typename dtype>
const_iterator NumC::NdArray< dtype >::cend (
            uint32 inRow ) const  [inline]
```

const iterator to 1 past the end of the input row

Parameters

row	

Returns

const_iterator

6.17.4.22 clip()

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::clip (
            dtype inMin,
            dtype inMax ) const  [inline]
```

Returns an array whose values are limited to [min, max].

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.clip.html

Parameters

min	value to clip to
max	value to clip to

Returns

clipped value

6.17.4.23 contains()

```
template<typename dtype>
NdArray<bool> NumC::NdArray< dtype >::contains (
            dtype inValue,
            Axis::Type inAxis = Axis::NONE ) const [inline]
```

returns whether or not a value is included the array

Parameters

value	
(Optional)	axis

Returns

bool

6.17.4.24 copy()

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::copy ( ) [inline]
```

Return a copy of the array

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩ndarray.copy.html

Parameters

None	

Returns

NdArray

6.17.4.25 cumprod()

```
template<typename dtype>
template<typename dtypeOut = double>
NdArray<dtypeOut> NumC::NdArray< dtype >::cumprod (
            Axis::Type inAxis = Axis::NONE ) const [inline]
```

Return the cumulative product of the elements along the given axis.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩
ndarray.cumprod.html

Parameters

(Optional)	axis

Returns

NdArray

6.17.4.26 cumsum()

```
template<typename dtype>
template<typename dtypeOut = double>
NdArray<dtypeOut> NumC::NdArray< dtype >::cumsum (
            Axis::Type inAxis = Axis::NONE ) const [inline]
```

Return the cumulative sum of the elements along the given axis.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩
ndarray.cumsum.html

Parameters

(Optional)	axis

Returns

NdArray

6.17.4.27 diagonal()

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::diagonal (
```

```
             uint32 inOffset = 0,
             Axis::Type inAxis = Axis::ROW ) const  [inline]
```

Return specified diagonals.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.diagonal.html

Parameters

Offset	of the diagonal from the main diagonal. Can be both positive and negative. Defaults to 0.
(Optional)	axis the offset is applied to

Returns

NdArray

6.17.4.28 dot()

```
template<typename dtype>
template<typename dtypeOut = double>
NdArray<dtypeOut> NumC::NdArray< dtype >::dot (
             const NdArray< dtype > & inOtherArray ) const  [inline]
```

Dot product of two arrays.

For 2-D arrays it is equivalent to matrix multiplication, and for 1-D arrays to inner product of vectors.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.dot.html

Parameters

NdArray	

Returns

dot product

6.17.4.29 dump()

```
template<typename dtype>
void NumC::NdArray< dtype >::dump (
             const std::string & inFilename ) const  [inline]
```

Dump a binary file of the array to the specified file. The array can be read back with or NumC::load.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.dump.html

Parameters

filename	

Returns

None

6.17.4.30 end() [1/2]

```
template<typename dtype>
iterator NumC::NdArray< dtype >::end ( )  [inline]
```

iterator to 1 past the end of the flattened array

Parameters

None	

Returns

iterator

6.17.4.31 end() [2/2]

```
template<typename dtype>
iterator NumC::NdArray< dtype >::end (
            uint32 inRow )  [inline]
```

iterator to the 1 past end of the row

Parameters

row	

Returns

iterator

6.17.4.32 endianess()

```
template<typename dtype>
Endian::Type NumC::NdArray< dtype >::endianess ( ) const  [inline]
```

Return the NdArrays endianess

Parameters

None	

Returns

Endian::Type

6.17.4.33 fill()

```
template<typename dtype>
void NumC::NdArray< dtype >::fill (
            dtype inFillValue ) [inline]
```

Fill the array with a scalar value.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.fill.html

Parameters

fill	value

Returns

None

6.17.4.34 flatten()

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::flatten ( ) const [inline]
```

Return a copy of the array collapsed into one dimension.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.flatten.html

Parameters

None	

Returns

NdArray

6.17.4.35 isempty()

```
template<typename dtype>
bool NumC::NdArray< dtype >::isempty ( ) const [inline]
```

Return if the NdArray is empty. ie the default construtor was used.

Parameters

None	

Returns

boolean

6.17.4.36 item()

```
template<typename dtype>
dtype NumC::NdArray< dtype >::item ( ) const [inline]
```

Copy an element of an array to a standard C++ scalar and return it.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.item.html

Parameters

None	

Returns

array element

6.17.4.37 max()

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::max (
            Axis::Type inAxis = Axis::NONE ) const [inline]
```

Return the maximum along a given axis.

Numpy Reference: `https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.max.html`

Parameters

(Optional)	Axis

Returns

NdArray

6.17.4.38 mean()

```
template<typename dtype>
NdArray<double> NumC::NdArray< dtype >::mean (
            Axis::Type inAxis = Axis::NONE ) const  [inline]
```

Return the mean along a given axis.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.mean.html

Parameters

(Optional)	Axis

Returns

NdArray

6.17.4.39 median()

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::median (
            Axis::Type inAxis = Axis::NONE ) const  [inline]
```

Return the median along a given axis. Does NOT average if array has even number of elements!

Parameters

(Optional)	Axis

Returns

NdArray

6.17.4.40 min()

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::min (
            Axis::Type inAxis = Axis::NONE ) const [inline]
```

Return the minimum along a given axis.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.min.html

Parameters

(Optional)	Axis

Returns

NdArray

6.17.4.41 nans()

```
template<typename dtype>
void NumC::NdArray< dtype >::nans ( ) [inline]
```

Fills the array with nans; only really works with. Only really works for dtype = float/double

Parameters

None	

Returns

None

6.17.4.42 nbytes()

```
template<typename dtype>
uint64 NumC::NdArray< dtype >::nbytes ( ) const [inline]
```

Returns the number of bytes held by the array

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.nbytes.html

Parameters

None	

Returns

number of bytes

6.17.4.43 newbyteorder()

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::newbyteorder (
            Endian::Type inEndianess ) const [inline]
```

Return the array with the same data viewed with a different byte order. only works for integer types, floating point types will not compile and you will be confused as to why...

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.newbyteorder.html

Parameters

Endian::Type	

Returns

NdArray

6.17.4.44 nonzero()

```
template<typename dtype>
NdArray<uint32> NumC::NdArray< dtype >::nonzero ( ) const [inline]
```

Return the indices of the flattened array of the elements that are non-zero.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.nonzero.html

Parameters

None	

Returns

NdArray

6.17.4.45 norm()

```
template<typename dtype>
template<typename dtypeOut = double>
NdArray<dtypeOut> NumC::NdArray< dtype >::norm (
            Axis::Type inAxis = Axis::NONE ) const  [inline]
```

Returns the norm of the array

Numpy Reference: http://www.numpy.org/devdocs/reference/generated/numpy.linalg.↩
norm.html?highlight=norm#numpy.linalg.norm

Parameters

(Optional)	Axis

Returns

norm

6.17.4.46 ones()

```
template<typename dtype>
void NumC::NdArray< dtype >::ones ( )  [inline]
```

Fills the array with ones

Parameters

None	

Returns

None

6.17.4.47 operator &() [1/2]

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::operator& (
            const NdArray< dtype > & inOtherArray ) const  [inline]
```

Takes the bitwise and of the elements of two arrays

Parameters

NdArray	

Returns

NdArray

6.17.4.48 operator &() [2/2]

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::operator& (
            dtype inScalar ) const  [inline]
```

Takes the bitwise and of the array and the scalar

Parameters

scalar	

Returns

NdArray

6.17.4.49 operator &=() [1/2]

```
template<typename dtype>
NdArray<dtype>& NumC::NdArray< dtype >::operator&= (
            const NdArray< dtype > & inOtherArray )  [inline]
```

Takes the bitwise and of the elements of two arrays

Parameters

NdArray	

Returns

NdArray

6.17.4.50 operator &=() [2/2]

```
template<typename dtype>
NdArray<dtype>& NumC::NdArray< dtype >::operator&= (
            dtype inScalar ) [inline]
```

Takes the bitwise and of the array and the scalar

Parameters

scalar	

Returns

NdArray

6.17.4.51 operator"!=() [1/2]

```
template<typename dtype>
NdArray<bool> NumC::NdArray< dtype >::operator!= (
            dtype inValue ) const [inline]
```

Returns an array of booleans of element wise comparison of two arrays

Parameters

NdArray	

Returns

NdArray

6.17.4.52 operator"!=() [2/2]

```
template<typename dtype>
NdArray<bool> NumC::NdArray< dtype >::operator!= (
            const NdArray< dtype > & inOtherArray ) const [inline]
```

Returns an array of booleans of element wise comparison of two arrays

Parameters

NdArray	

Returns

NdArray

6.17.4.53 operator%() [1/2]

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::operator% (
            const NdArray< dtype > & inOtherArray ) const  [inline]
```

Takes the modulus of the elements of two arrays

Parameters

NdArray	

Returns

NdArray

6.17.4.54 operator%() [2/2]

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::operator% (
            dtype inScalar ) const  [inline]
```

Modulus of the array and the scalar

Parameters

scalar	

Returns

NdArray

6.17.4.55 operator%=() [1/2]

```
template<typename dtype>
NdArray<dtype>& NumC::NdArray< dtype >::operator%= (
            const NdArray< dtype > & inOtherArray )  [inline]
```

Takes the modulus of the elements of two arrays

Parameters

NdArray	

Returns

NdArray

6.17.4.56 operator%=() [2/2]

```
template<typename dtype>
NdArray<dtype>& NumC::NdArray< dtype >::operator%= (
            dtype inScalar )  [inline]
```

Modulus of the array and the scalar

Parameters

scalar	

Returns

NdArray

6.17.4.57 operator()() [1/5]

```
template<typename dtype>
dtype& NumC::NdArray< dtype >::operator() (
            int32 inRowIndex,
            int32 inColIndex )  [inline]
```

2D access operator with no bounds checking

Parameters

row	index
col	index

Returns

value

6.17.4.58 operator()() [2/5]

```
template<typename dtype>
const dtype& NumC::NdArray< dtype >::operator() (
            int32 inRowIndex,
            int32 inColIndex ) const  [inline]
```

const 2D access operator with no bounds checking

Parameters

row	index
col	index

Returns

value

6.17.4.59 operator()() [3/5]

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::operator() (
            const Slice & inRowSlice,
            const Slice & inColSlice ) const  [inline]
```

2D Slicing access operator with no bounds checking. returned array is of the range [start, stop).

Parameters

Row	Slice
Col	Slice

Returns

NdArray

6.17.4.60 operator()() [4/5]

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::operator() (
            const Slice & inRowSlice,
            int32 inColIndex ) const  [inline]
```

2D Slicing access operator with no bounds checking. returned array is of the range [start, stop).

Parameters

Row	Slice
Col	index

Returns

NdArray

6.17.4.61 operator()() [5/5]

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::operator() (
            int32 inRowIndex,
            const Slice & inColSlice ) const  [inline]
```

2D Slicing access operator with no bounds checking. returned array is of the range [start, stop).

Parameters

Row	index
Col	Slice

Returns

NdArray

6.17.4.62 operator*() [1/2]

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::operator* (
            const NdArray< dtype > & inOtherArray ) const  [inline]
```

Multiplies the elements of two arrays

Parameters

NdArray	

Returns

NdArray

6.17.4.63 operator*() [2/2]

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::operator* (
            dtype inScalar ) const [inline]
```

Muliplies the scalar to the array

Parameters

scalar	

Returns

NdArray

6.17.4.64 operator*=() [1/2]

```
template<typename dtype>
NdArray<dtype>& NumC::NdArray< dtype >::operator*= (
            const NdArray< dtype > & inOtherArray ) [inline]
```

Multiplies the elements of two arrays

Parameters

NdArray	

Returns

NdArray

6.17.4.65 operator*=() [2/2]

```
template<typename dtype>
NdArray<dtype>& NumC::NdArray< dtype >::operator*= (
            dtype inScalar ) [inline]
```

Muliplies the scalar to the array

Parameters

scalar	

Returns

NdArray

6.17.4.66 operator+() [1/2]

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::operator+ (
            const NdArray< dtype > & inOtherArray ) const [inline]
```

Adds the elements of two arrays

Parameters

NdArray	

Returns

NdArray

6.17.4.67 operator+() [2/2]

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::operator+ (
            dtype inScalar ) const [inline]
```

Adds the scalar to the array

Parameters

scalar	

Returns

NdArray

6.17.4.68 operator++() [1/2]

```
template<typename dtype>
NdArray<dtype>& NumC::NdArray< dtype >::operator++ ( ) [inline]
```

prefix incraments the elements of an array

Parameters

NdArray	

Returns

NdArray

6.17.4.69 operator++() [2/2]

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::operator++ (
            int ) const [inline]
```

postfix increments the elements of an array

Parameters

NdArray	

Returns

NdArray

6.17.4.70 operator+=() [1/2]

```
template<typename dtype>
NdArray<dtype>& NumC::NdArray< dtype >::operator+= (
            const NdArray< dtype > & inOtherArray ) [inline]
```

Adds the elements of two arrays

Parameters

NdArray	

Returns

NdArray

6.17.4.71 operator+=() [2/2]

```
template<typename dtype>
NdArray<dtype>& NumC::NdArray< dtype >::operator+= (
            dtype inScalar )  [inline]
```

Adds the scalar to the array

Parameters

scalar	

Returns

NdArray

6.17.4.72 operator-() [1/2]

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::operator- (
            const NdArray< dtype > & inOtherArray ) const  [inline]
```

Subtracts the elements of two arrays

Parameters

NdArray	

Returns

NdArray

6.17.4.73 operator-() [2/2]

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::operator- (
            dtype inScalar ) const  [inline]
```

Subtracts the scalar from the array

Parameters

scalar	

Returns

NdArray

6.17.4.74 operator--() [1/2]

```
template<typename dtype>
NdArray<dtype>& NumC::NdArray< dtype >::operator-- ( ) [inline]
```

prefix decrements the elements of an array

Parameters

NdArray	

Returns

NdArray

6.17.4.75 operator--() [2/2]

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::operator-- (
            int ) const [inline]
```

postfix decrements the elements of an array

Parameters

NdArray	

Returns

NdArray

6.17.4.76 operator-=() [1/2]

```
template<typename dtype>
NdArray<dtype>& NumC::NdArray< dtype >::operator-= (
            const NdArray< dtype > & inOtherArray ) [inline]
```

Subtracts the elements of two arrays

Parameters

NdArray	

Returns

NdArray

6.17.4.77 operator-=() [2/2]

```
template<typename dtype>
NdArray<dtype>& NumC::NdArray< dtype >::operator-= (
            dtype inScalar ) [inline]
```

Subtracts the scalar from the array

Parameters

scalar	

Returns

NdArray

6.17.4.78 operator/() [1/2]

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::operator/ (
            const NdArray< dtype > & inOtherArray ) const [inline]
```

Divides the elements of two arrays

Parameters

NdArray	

Returns

NdArray

6.17.4.79 operator/() [2/2]

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::operator/ (
            dtype inScalar ) const  [inline]
```

Divides the array by the scalar

Parameters

scalar	

Returns

NdArray

6.17.4.80 operator/=() [1/2]

```
template<typename dtype>
NdArray<dtype>& NumC::NdArray< dtype >::operator/= (
            const NdArray< dtype > & inOtherArray )  [inline]
```

Divides the elements of two arrays

Parameters

NdArray	

Returns

NdArray

6.17.4.81 operator/=() [2/2]

```
template<typename dtype>
NdArray<dtype>& NumC::NdArray< dtype >::operator/= (
            dtype inScalar )  [inline]
```

Divides the array by the scalar

Parameters

scalar	

Returns

NdArray

6.17.4.82 operator<() [1/2]

```
template<typename dtype>
NdArray<bool> NumC::NdArray< dtype >::operator< (
            dtype inScalar ) const [inline]
```

Returns an array of booleans of element wise comparison the array and a scalar

Parameters

NdArray	

Returns

NdArray

6.17.4.83 operator<() [2/2]

```
template<typename dtype>
NdArray<bool> NumC::NdArray< dtype >::operator< (
            const NdArray< dtype > & inOtherArray ) const [inline]
```

Returns an array of booleans of element wise comparison of two arrays

Parameters

NdArray	

Returns

NdArray

6.17.4.84 operator<=() [1/2]

```
template<typename dtype>
NdArray<bool> NumC::NdArray< dtype >::operator<= (
            dtype inScalar ) const [inline]
```

Returns an array of booleans of element wise comparison the array and a scalar

Parameters

NdArray	

Returns

NdArray

6.17.4.85 operator<=() [2/2]

```
template<typename dtype>
NdArray<bool> NumC::NdArray< dtype >::operator<= (
            const NdArray< dtype > & inOtherArray ) const  [inline]
```

Returns an array of booleans of element wise comparison of two arrays

Parameters

NdArray	

Returns

NdArray

6.17.4.86 operator=() [1/2]

```
template<typename dtype>
NdArray<dtype>& NumC::NdArray< dtype >::operator= (
            const NdArray< dtype > & inOtherArray )  [inline]
```

Assignment operator, performs a deep copy

Parameters

NdArray	

Returns

None

6.17.4.87 operator=() [2/2]

```
template<typename dtype>
NdArray<dtype>& NumC::NdArray< dtype >::operator= (
            NdArray< dtype > && inOtherArray )  [inline]
```

Move operator, performs a deep move

Parameters

NdArray	

Returns

None

6.17.4.88 operator==() [1/2]

```
template<typename dtype>
NdArray<bool> NumC::NdArray< dtype >::operator== (
            dtype inValue ) const  [inline]
```

Returns an array of booleans of element wise comparison of two arrays

Parameters

NdArray	

Returns

NdArray

6.17.4.89 operator==() [2/2]

```
template<typename dtype>
NdArray<bool> NumC::NdArray< dtype >::operator== (
            const NdArray< dtype > & inOtherArray ) const  [inline]
```

Returns an array of booleans of element wise comparison of two arrays

Parameters

NdArray	

Returns

NdArray

6.17.4.90 operator>() [1/2]

```
template<typename dtype>
NdArray<bool> NumC::NdArray< dtype >::operator> (
            dtype inScalar ) const  [inline]
```

Returns an array of booleans of element wise comparison the array and a scalar

Parameters

NdArray	

Returns

NdArray

6.17.4.91 operator>() [2/2]

```
template<typename dtype>
NdArray<bool> NumC::NdArray< dtype >::operator> (
            const NdArray< dtype > & inOtherArray ) const  [inline]
```

Returns an array of booleans of element wise comparison of two arrays

Parameters

NdArray	

Returns

NdArray

6.17.4.92 operator>=() [1/2]

```
template<typename dtype>
NdArray<bool> NumC::NdArray< dtype >::operator>= (
            dtype inScalar ) const  [inline]
```

Returns an array of booleans of element wise comparison the array and a scalar

Parameters

NdArray	

Returns

NdArray

6.17.4.93 operator>=() [2/2]

```
template<typename dtype>
NdArray<bool> NumC::NdArray< dtype >::operator>= (
            const NdArray< dtype > & inOtherArray ) const  [inline]
```

Returns an array of booleans of element wise comparison of two arrays

Parameters

NdArray	

Returns

NdArray

6.17.4.94 operator[]() [1/3]

```
template<typename dtype>
dtype& NumC::NdArray< dtype >::operator[] (
            int32 inIndex )  [inline]
```

1D access operator with no bounds checking

Parameters

array	index

Returns

value

6.17.4.95 operator[]() [2/3]

```
template<typename dtype>
const dtype& NumC::NdArray< dtype >::operator[] (
             int32 inIndex ) const  [inline]
```

const 1D access operator with no bounds checking

Parameters

array	index

Returns

value

6.17.4.96 operator[]() [3/3]

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::operator[] (
             const Slice & inSlice ) const  [inline]
```

1D Slicing access operator with no bounds checking. returned array is of the range [start, stop).

Parameters

Slice	

Returns

NdArray

6.17.4.97 operator^() [1/2]

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::operator^ (
             const NdArray< dtype > & inOtherArray ) const  [inline]
```

Takes the bitwise xor of the elements of two arrays

Parameters

None	

Returns

None

6.17.4.98 operator^() [2/2]

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::operator^ (
            dtype inScalar ) const [inline]
```

Takes the bitwise xor of the array and the scalar

Parameters

scalar	

Returns

NdArray

6.17.4.99 operator^=() [1/2]

```
template<typename dtype>
NdArray<dtype>& NumC::NdArray< dtype >::operator^= (
            const NdArray< dtype > & inOtherArray ) [inline]
```

Takes the bitwise xor of the elements of two arrays

Parameters

None	

Returns

None

6.17.4.100 operator^=() [2/2]

```
template<typename dtype>
NdArray<dtype>& NumC::NdArray< dtype >::operator^= (
            dtype inScalar ) [inline]
```

Takes the bitwise xor of the array and the scalar

Parameters

scalar	

Returns

NdArray

6.17.4.101 operator"\|() [1/2]

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::operator| (
            const NdArray< dtype > & inOtherArray ) const  [inline]
```

Takes the bitwise or of the elements of two arrays

Parameters

NdArray	

Returns

NdArray

6.17.4.102 operator"\|() [2/2]

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::operator| (
            dtype inScalar ) const  [inline]
```

Takes the bitwise or of the array and the scalar

Parameters

scalar	

Returns

NdArray

6.17.4.103 operator"\|=() [1/2]

```
template<typename dtype>
NdArray<dtype>& NumC::NdArray< dtype >::operator|= (
            const NdArray< dtype > & inOtherArray ) [inline]
```

Takes the bitwise or of the elements of two arrays

Parameters

NdArray	

Returns

NdArray

6.17.4.104 operator"\|=() [2/2]

```
template<typename dtype>
NdArray<dtype>& NumC::NdArray< dtype >::operator|= (
            dtype inScalar ) [inline]
```

Takes the bitwise or of the array and the scalar

Parameters

scalar	

Returns

NdArray

6.17.4.105 operator∼()

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::operator~ ( ) const [inline]
```

Takes the bitwise not of the array

Parameters

None	

Returns

NdArray

6.17.4.106 partition()

```
template<typename dtype>
void NumC::NdArray< dtype >::partition (
            uint32 inKth,
            Axis::Type inAxis = Axis::NONE )  [inline]
```

Rearranges the elements in the array in such a way that value of the element in kth position is in the position it would be in a sorted array. All elements smaller than the kth element are moved before this element and all equal or greater are moved behind it. The ordering of the elements in the two partitions is undefined.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.partition.html

Parameters

kth	element
(Optional)	Axis

Returns

None

6.17.4.107 print()

```
template<typename dtype>
void NumC::NdArray< dtype >::print ( ) const  [inline]
```

Prints the array to the console.

Parameters

None	

Returns

None

6.17.4.108 prod()

```
template<typename dtype>
template<typename dtypeOut = double>
NdArray<dtypeOut> NumC::NdArray< dtype >::prod (
            Axis::Type inAxis = Axis::NONE ) const [inline]
```

Return the product of the array elements over the given axis

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩ ndarray.prod.html

Parameters

(Optional)	Axis

Returns

NdArray

6.17.4.109 ptp()

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::ptp (
            Axis::Type inAxis = Axis::NONE ) const [inline]
```

Peak to peak (maximum - minimum) value along a given axis.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩ ndarray.ptp.html

Parameters

(Optional)	Axis

Returns

NdArray

6.17.4.110 put() [1/12]

```
template<typename dtype>
void NumC::NdArray< dtype >::put (
            int32 inIndex,
            dtype inValue ) [inline]
```

set the flat index element to the value

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.↩
ndarray.put.html

Parameters

index	
value	

Returns

None

6.17.4.111 put() [2/12]

```
template<typename dtype>
void NumC::NdArray< dtype >::put (
            int32 inRow,
            int32 inCol,
            dtype inValue ) [inline]
```

set the 2D row/col index element to the value

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.put.html

Parameters

row	index
col	index
value	

Returns

None

6.17.4.112 put() [3/12]

```
template<typename dtype>
void NumC::NdArray< dtype >::put (
            const NdArray< uint32 > & inIndices,
            dtype inValue ) [inline]
```

Set a.flat[n] = values for all n in indices.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.put.html

Parameters

NdArray	of indices
value	

Returns

None

6.17.4.113 put() [4/12]

```
template<typename dtype>
void NumC::NdArray< dtype >::put (
            const NdArray< uint32 > & inIndices,
            const NdArray< dtype > & inValues )  [inline]
```

Set a.flat[n] = values[n] for all n in indices.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.put.html

Parameters

NdArray	of indices
NdArray	of values

Returns

None

6.17.4.114 put() [5/12]

```
template<typename dtype>
void NumC::NdArray< dtype >::put (
            const Slice & inSlice,
            dtype inValue )  [inline]
```

Set the slice indices to the input value.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.put.html

Parameters

Slice	1D
value	

Returns

None

6.17.4.115 put() [6/12]

```
template<typename dtype>
void NumC::NdArray< dtype >::put (
            const Slice & inSlice,
            const NdArray< dtype > & inValues )  [inline]
```

Set the slice indices to the input values.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.put.html

Parameters

Slice	1D
NdArray	of values

Returns

None

6.17.4.116 put() [7/12]

```
template<typename dtype>
void NumC::NdArray< dtype >::put (
            const Slice & inRowSlice,
            const Slice & inColSlice,
            dtype inValue )  [inline]
```

Set the slice indices to the input values.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.put.html

Parameters

Slice	rows
Slice	cols
value	

Returns

None

6.17.4.117 put() [8/12]

```
template<typename dtype>
void NumC::NdArray< dtype >::put (
            const Slice & inRowSlice,
            int32 inColIndex,
            dtype inValue )  [inline]
```

Set the slice indices to the input values.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.put.html

Parameters

Slice	rows
col	index
value	

Returns

None

6.17.4.118 put() [9/12]

```
template<typename dtype>
void NumC::NdArray< dtype >::put (
            int32 inRowIndex,
            const Slice & inColSlice,
            dtype inValue )  [inline]
```

Set the slice indices to the input values.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.put.html

Parameters

row	index
Slice	cols
value	

Returns

None

6.17.4.119 put() [10/12]

```
template<typename dtype>
void NumC::NdArray< dtype >::put (
            const Slice & inRowSlice,
            const Slice & inColSlice,
            const NdArray< dtype > & inValues ) [inline]
```

Set the slice indices to the input values.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.put.html

Parameters

Slice	rows
Slice	cols
NdArray	of values

Returns

None

6.17.4.120 put() [11/12]

```
template<typename dtype>
void NumC::NdArray< dtype >::put (
            const Slice & inRowSlice,
            int32 inColIndex,
            const NdArray< dtype > & inValues ) [inline]
```

Set the slice indices to the input values.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.put.html

Parameters

Slice	rows
col	index
NdArray	of values

Returns

None

6.17.4.121 put() [12/12]

```
template<typename dtype>
void NumC::NdArray< dtype >::put (
            int32 inRowIndex,
            const Slice & inColSlice,
            const NdArray< dtype > & inValues ) [inline]
```

Set the slice indices to the input values.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.put.html

Parameters

row	index
Slice	cols
NdArray	of values

Returns

None

6.17.4.122 repeat() [1/2]

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::repeat (
            uint32 inNumRows,
            uint32 inNumCols ) const [inline]
```

Repeat elements of an array.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.repeat.html

Parameters

numRows	
numCols	

Returns

NdArray

6.17.4.123 repeat() [2/2]

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::repeat (
            const Shape & inRepeatShape ) const [inline]
```

Repeat elements of an array.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.repeat.html

Parameters

Shape	

Returns

NdArray

6.17.4.124 reshape() [1/2]

```
template<typename dtype>
void NumC::NdArray< dtype >::reshape (
            uint32 inNumRows,
            uint32 inNumCols ) [inline]
```

Returns an array containing the same data with a new shape.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.repeat.html

Parameters

Shape	

Returns

None

6.17.4.125 reshape() [2/2]

```
template<typename dtype>
void NumC::NdArray< dtype >::reshape (
            const Shape & inShape )  [inline]
```

Returns an array containing the same data with a new shape.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.reshape.html

Parameters

Shape	

Returns

None

6.17.4.126 resizeFast() [1/2]

```
template<typename dtype>
void NumC::NdArray< dtype >::resizeFast (
            uint32 inNumRows,
            uint32 inNumCols )  [inline]
```

Change shape and size of array in-place. All previous data of the array is lost.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.resize.html

Parameters

Shape	

Returns

None

6.17.4.127 resizeFast() [2/2]

```
template<typename dtype>
void NumC::NdArray< dtype >::resizeFast (
            const Shape & inShape )  [inline]
```

Change shape and size of array in-place. All previous data of the array is lost.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.resize.html

Parameters

Shape	

Returns

None

6.17.4.128 resizeSlow() [1/2]

```
template<typename dtype>
void NumC::NdArray< dtype >::resizeSlow (
            uint32 inNumRows,
            uint32 inNumCols )  [inline]
```

Return a new array with the specified shape. If new shape is larger than old shape then array will be padded with zeros. If new shape is smaller than the old shape then the data will be discarded.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.resize.html

Parameters

num	Rows
num	Cols

Returns

None

6.17.4.129 resizeSlow() [2/2]

```
template<typename dtype>
void NumC::NdArray< dtype >::resizeSlow (
            const Shape & inShape )  [inline]
```

Return a new array with the specified shape. If new shape is larger than old shape then array will be padded with zeros. If new shape is smaller than the old shape then the data will be discarded.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.resize.html

Parameters

Shape	

Returns

None

6.17.4.130 round()

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::round (
            uint8 inNumDecimals = 0 ) const [inline]
```

Return a with each element rounded to the given number of decimals.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.round.html

Parameters

number	of decimals to round to

Returns

NdArray

6.17.4.131 shape()

```
template<typename dtype>
Shape NumC::NdArray< dtype >::shape ( ) const [inline]
```

Return the shape of the array

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.shape.html

Parameters

None	

Returns

Shape

6.17.4.132 size()

```
template<typename dtype>
uint32 NumC::NdArray< dtype >::size ( ) const [inline]
```

Return the size of the array

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.size.html

Parameters

None	

Returns

size

6.17.4.133 sort()

```
template<typename dtype>
void NumC::NdArray< dtype >::sort (
            Axis::Type inAxis = Axis::NONE )  [inline]
```

Sort an array, in-place.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.sort.html

Parameters

(Optional)	Axis

Returns

size

6.17.4.134 std()

```
template<typename dtype>
NdArray<double> NumC::NdArray< dtype >::std (
            Axis::Type inAxis = Axis::NONE ) const  [inline]
```

Return the std along a given axis.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.std.html

Parameters

(Optional)	Axis

Returns

NdArray

6.17.4.135 str()

```
template<typename dtype>
std::string NumC::NdArray< dtype >::str ( ) const [inline]
```

returns the NdArray as a string representation

Parameters

None	

Returns

string

6.17.4.136 sum()

```
template<typename dtype>
template<typename dtypeOut = double>
NdArray<dtypeOut> NumC::NdArray< dtype >::sum (
            Axis::Type inAxis = Axis::NONE ) const [inline]
```

Return the sum of the array elements over the given axis.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.sum.html

Parameters

(Optional)	Axis

Returns

NdArray

6.17.4.137 swapaxes()

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::swapaxes ( ) const [inline]
```

Interchange two axes of an array. Equivalent to transpose...

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.swapaxes.html

Parameters

None	

Returns

NdArray

6.17.4.138 tofile()

```
template<typename dtype>
void NumC::NdArray< dtype >::tofile (
            const std::string & inFilename,
            const std::string & inSep = "" ) const [inline]
```

Write array to a file as text or binary (default).. The data produced by this method can be recovered using the function fromfile().

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.tofile.html

Parameters

filename	
Separator	between array items for text output. If "" (empty), a binary file is written

Returns

None

6.17.4.139 toStlVector()

```
template<typename dtype>
std::vector<dtype> NumC::NdArray< dtype >::toStlVector ( ) const [inline]
```

Write flattened array to an STL vector

Parameters

None	

Returns

None

6.17.4.140 trace()

```
template<typename dtype>
template<typename dtypeOut = double>
dtypeOut NumC::NdArray< dtype >::trace (
            uint16 inOffset = 0,
            Axis::Type inAxis = Axis::ROW ) const [inline]
```

Return the sum along diagonals of the array.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.trace.html

Parameters

Offset	of the diagonal from the main diagonal. Can be both positive and negative. Defaults to 0.
(Optional)	Axis to offset from

Returns

None

6.17.4.141 transpose()

```
template<typename dtype>
NdArray<dtype> NumC::NdArray< dtype >::transpose ( ) const [inline]
```

Tranpose the rows and columns of an array

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.transpose.html

Parameters

None	

Returns

NdArray

6.17.4.142 var()

```
template<typename dtype>
NdArray<double> NumC::NdArray< dtype >::var (
            Axis::Type inAxis = Axis::NONE ) const [inline]
```

Returns the variance of the array elements, along given axis.

Numpy Reference: https://www.numpy.org/devdocs/reference/generated/numpy.ndarray.var.html

Parameters

(Optional)	Axes

Returns

NdArray

6.17.4.143 zeros()

```
template<typename dtype>
void NumC::NdArray< dtype >::zeros ( ) [inline]
```

Fills the array with zeros

Parameters

None	

Returns

None

6.17.5 Friends And Related Function Documentation

6.17.5.1 operator<< [1/2]

```
template<typename dtype>
NdArray<dtype> operator<< (
            const NdArray< dtype > & lhs,
            uint8 inNumBits ) [friend]
```

Bitshifts left the elements of the array

Parameters

None	

Returns

None

6.17.5.2 operator<< [2/2]

```
template<typename dtype>
std::ostream& operator<< (
            std::ostream & inOStream,
            const NdArray< dtype > & inArray )  [friend]
```

io operator for the NdArray class

Parameters

None	

Returns

None

6.17.5.3 operator<<=

```
template<typename dtype>
NdArray<dtype>& operator<<= (
            NdArray< dtype > & lhs,
            uint8 inNumBits )  [friend]
```

Bitshifts left the elements of the array

Parameters

None	

Returns

None

6.17.5.4 operator>>

```
template<typename dtype>
NdArray<dtype> operator>> (
            const NdArray< dtype > & lhs,
            uint8 inNumBits )  [friend]
```

Bitshifts right the elements of the array

Parameters

None	

Returns

None

6.17.5.5 operator>>=

```
template<typename dtype>
NdArray<dtype>& operator>>= (
            NdArray< dtype > & lhs,
            uint8 inNumBits )  [friend]
```

Bitshifts right the elements of the array

Parameters

None	

Returns

None

The documentation for this class was generated from the following file:

- NdArray.hpp

6.18 NumC::Order Struct Reference

C or Fortran ordering from python.

```
#include <BoostNumpyNdarrayHelper.hpp>
```

Public Types

- enum Type { F, C }

6.18.1 Detailed Description

C or Fortran ordering from python.

6.18.2 Member Enumeration Documentation

6.18.2.1 Type

`enum NumC::Order::Type`

Enumerator

F	
C	

The documentation for this struct was generated from the following file:

- BoostNumpyNdarrayHelper.hpp

6.19 NumC::ImageProcessing< dtype >::Pixel Class Reference

Holds the information for a single pixel.

`#include <ImageProcessing.hpp>`

Public Member Functions

- Pixel ()
- Pixel (uint32 inRow, uint32 inCol, dtype inIntensity)
- int32 clusterId () const
- uint32 col () const
- dtype intensity () const
- bool operator!= (const Pixel &rhs) const
- bool operator< (const Pixel &rhs) const
- bool operator== (const Pixel &rhs) const
- void print () const
- uint32 row () const
- void setClusterId (int32 inClusterId)
- std::string str () const

Friends

- std::ostream & operator<< (std::ostream &inStream, const Pixel &inPixel)

6.19.1 Detailed Description

template<typename dtype>
class NumC::ImageProcessing< dtype >::Pixel

Holds the information for a single pixel.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 Pixel() [1/2]

```
template<typename dtype >
NumC::ImageProcessing< dtype >::Pixel::Pixel ( )  [inline]
```

defualt constructor needed by containers

Parameters

None	

Returns

None

6.19.2.2 Pixel() [2/2]

```
template<typename dtype >
NumC::ImageProcessing< dtype >::Pixel::Pixel (
            uint32 inRow,
            uint32 inCol,
            dtype inIntensity )  [inline]
```

constructor

Parameters

pixel	row,
pixel	column,
pixel	intensity

Returns

None

6.19.3 Member Function Documentation

6.19.3.1 clusterId()

```
template<typename dtype >
int32 NumC::ImageProcessing< dtype >::Pixel::clusterId ( ) const [inline]
```

returns the cluster id that this pixel belongs to

Parameters

None	

Returns

cluster id

6.19.3.2 col()

```
template<typename dtype >
uint32 NumC::ImageProcessing< dtype >::Pixel::col ( ) const [inline]
```

returns the pixel column

Parameters

None	

Returns

column

6.19.3.3 intensity()

```
template<typename dtype >
dtype NumC::ImageProcessing< dtype >::Pixel::intensity ( ) const [inline]
```

returns the pixel intensity

Parameters

None	

Returns

intensity

6.19.3.4 operator"!=()

```
template<typename dtype >
bool NumC::ImageProcessing< dtype >::Pixel::operator!= (
            const Pixel & rhs ) const  [inline]
```

not equality operator

Parameters

None	

Returns

bool

6.19.3.5 operator<()

```
template<typename dtype >
bool NumC::ImageProcessing< dtype >::Pixel::operator< (
            const Pixel & rhs ) const  [inline]
```

less than operator for std::sort algorithm and std::set<>; NOTE: std::sort sorts in ascending order. Since I want to sort the centroids in descensing order, I am purposefully defining this operator backwards!

Parameters

None	

Returns

None

6.19.3.6 operator==()

```
template<typename dtype >
bool NumC::ImageProcessing< dtype >::Pixel::operator== (
            const Pixel & rhs ) const  [inline]
```

equality operator

Parameters

None	

Returns

bool

6.19.3.7 print()

```
template<typename dtype >
void NumC::ImageProcessing< dtype >::Pixel::print ( ) const [inline]
```

Method Description: prints the Pixel object to the console

Parameters

None	

Returns

None

6.19.3.8 row()

```
template<typename dtype >
uint32 NumC::ImageProcessing< dtype >::Pixel::row ( ) const [inline]
```

returns the pixel row

Parameters

None	

Returns

row

6.19.3.9 setClusterId()

```
template<typename dtype >
void NumC::ImageProcessing< dtype >::Pixel::setClusterId (
            int32 inClusterId ) [inline]
```

sets the cluster id that this pixel belongs to

Parameters

cluster	id

Returns

None

6.19.3.10 str()

```
template<typename dtype >
std::string NumC::ImageProcessing< dtype >::Pixel::str ( ) const [inline]
```

returns the pixel information as a string

Parameters

None	

Returns

std::string

6.19.4 Friends And Related Function Documentation

6.19.4.1 operator<<

```
template<typename dtype >
std::ostream& operator<< (
            std::ostream & inStream,
            const Pixel & inPixel ) [friend]
```

osstream operator

Parameters

std::ostream	
Pixel	

Returns

std::ostream

The documentation for this class was generated from the following file:

- ImageProcessing.hpp

6.20 NumC::Polynomial< dtype > Class Template Reference

Class for dealing with common polynomials.

```
#include <Polynomial.hpp>
```

6.20.1 Detailed Description

template<typename dtype>
class NumC::Polynomial< dtype >

Class for dealing with common polynomials.

The documentation for this class was generated from the following file:

- Polynomial.hpp

6.21 NumC::Rotations::Quaternion Class Reference

Holds a unit quaternion.

```
#include <Rotations.hpp>
```

Public Member Functions

- Quaternion ()
- Quaternion (double inI, double inJ, double inK, double inS)
- Quaternion (const NdArray< double > &inArray)
- NdArray< double > angularVelocity (const Quaternion &inQuat2, double inTime) const
- Quaternion conjugate () const
- double i () const
- Quaternion inverse () const
- double j () const
- double k () const
- Quaternion nlerp (const Quaternion &inQuat2, double inPercent) const
- bool operator!= (const Quaternion &inRhs) const
- Quaternion operator* (const Quaternion &inRhs) const
- Quaternion operator* (double inScalar) const

- template< typename dtype >
  NdArray< double > operator* (const NdArray< dtype > &inVec) const
- Quaternion & operator*= (const Quaternion &inRhs)
- Quaternion & operator*= (double inScalar)
- Quaternion operator+ (const Quaternion &inRhs) const
- Quaternion & operator+= (const Quaternion &inRhs)
- Quaternion operator- (const Quaternion &inRhs) const
- Quaternion & operator-= (const Quaternion &inRhs)
- Quaternion operator/ (const Quaternion &inRhs) const
- Quaternion & operator/= (const Quaternion &inRhs)
- bool operator== (const Quaternion &inRhs) const
- void print () const
- template< typename dtype >
  NdArray< double > rotate (const NdArray< dtype > &inVector) const
- double s () const
- Quaternion slerp (const Quaternion &inQuat2, double inPercent) const
- std::string str () const
- NdArray< double > toDCM () const
- NdArray< double > toNdArray () const

Static Public Member Functions

- template< typename dtype >
  static Quaternion angleAxisRotation (const NdArray< dtype > &inAxis, double inAngle)
- static NdArray< double > angularVelocity (const Quaternion &inQuat1, const Quaternion &inQuat2, double inTime)
- template< typename dtype >
  static Quaternion fromDCM (const NdArray< dtype > &inDcm)
- static Quaternion identity ()
- static Quaternion nlerp (const Quaternion &inQuat1, const Quaternion &inQuat2, double inPercent)
- static Quaternion slerp (const Quaternion &inQuat1, const Quaternion &inQuat2, double inPercent)
- static Quaternion xRotation (double inAngle)
- static Quaternion yRotation (double inAngle)
- static Quaternion zRotation (double inAngle)

Friends

- std::ostream & operator<< (std::ostream &inOStream, const Quaternion &inQuat)

6.21.1 Detailed Description

Holds a unit quaternion.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 Quaternion() [1/3]

`NumC::Rotations::Quaternion::Quaternion ( ) [inline]`

Default Constructor, not super usefull on its own

Parameters

None	

Returns

None

6.21.2.2 Quaternion() [2/3]

```
NumC::Rotations::Quaternion::Quaternion (
            double inI,
            double inJ,
            double inK,
            double inS ) [inline]
```

Constructor

Parameters

i	
j	
k	
s	

Returns

None

6.21.2.3 Quaternion() [3/3]

```
NumC::Rotations::Quaternion::Quaternion (
            const NdArray< double > & inArray ) [inline]
```

Constructor

Parameters

NdArray,size	= 4

Returns

None

6.21.3 Member Function Documentation

6.21.3.1 angleAxisRotation()

```
template<typename dtype >
static Quaternion NumC::Rotations::Quaternion::angleAxisRotation (
            const NdArray< dtype > & inAxis,
            double inAngle )  [inline], [static]
```

returns a quaternion to rotate about the input axis by the input angle

Parameters

NdArray,x,y,z	vector components
angle	in radians

Returns

Quaternion

6.21.3.2 angularVelocity() [1/2]

```
static NdArray<double> NumC::Rotations::Quaternion::angularVelocity (
            const Quaternion & inQuat1,
            const Quaternion & inQuat2,
            double inTime )  [inline], [static]
```

angular velocity between the two quaternions. The norm of the array is the magnitude

Parameters

Quaternion	1
Quaternion	2
seperation	time

Returns

Quaternion

6.21.3.3 angularVelocity() [2/2]

```
NdArray<double> NumC::Rotations::Quaternion::angularVelocity (
            const Quaternion & inQuat2,
            double inTime ) const  [inline]
```

angular velocity between the two quaternions. The norm of the array is the magnitude

Parameters

Quaternion	2
seperation	time

Returns

Quaternion

6.21.3.4 conjugate()

```
Quaternion NumC::Rotations::Quaternion::conjugate ( ) const  [inline]
```

quaternion conjugate

Parameters

None	

Returns

s

6.21.3.5 fromDCM()

```
template<typename dtype >
static Quaternion NumC::Rotations::Quaternion::fromDCM (
            const NdArray< dtype > & inDcm )  [inline], [static]
```

converts from a direction cosine matrix to a quaternion

Parameters

NdArray	

Returns

Quaternion

6.21.3.6 i()

```
double NumC::Rotations::Quaternion::i ( ) const [inline]
```

returns the i component

Parameters

None	

Returns

i

6.21.3.7 identity()

```
static Quaternion NumC::Rotations::Quaternion::identity ( ) [inline], [static]
```

quaternion identity (0,0,0,1)

Parameters

None	

Returns

Quaternion

6.21.3.8 inverse()

```
Quaternion NumC::Rotations::Quaternion::inverse ( ) const [inline]
```

quaternion inverse

Parameters

None	

Returns

Quaterion

6.21.3.9 j()

```
double NumC::Rotations::Quaternion::j ( ) const [inline]
```

returns the j component

Parameters

None	

Returns

j

6.21.3.10 k()

```
double NumC::Rotations::Quaternion::k ( ) const [inline]
```

returns the k component

Parameters

None	

Returns

k

6.21.3.11 nlerp() [1/2]

```
static Quaternion NumC::Rotations::Quaternion::nlerp (
            const Quaternion & inQuat1,
            const Quaternion & inQuat2,
            double inPercent ) [inline], [static]
```

linearly interpolates between the two quaternions

Parameters

Quaternion	1
Quaternion	2
percent	[0, 1]

Returns

Quaternion

6.21.3.12 nlerp() [2/2]

```
Quaternion NumC::Rotations::Quaternion::nlerp (
            const Quaternion & inQuat2,
            double inPercent ) const [inline]
```

linearly interpolates between the two quaternions

Parameters

Quaternion	2
percent	(0, 1)

Returns

Quaternion

6.21.3.13 operator"!=()

```
bool NumC::Rotations::Quaternion::operator!= (
            const Quaternion & inRhs ) const [inline]
```

equality operator

Parameters

None	

Returns

None

6.21.3.14 operator*() [1/3]

```
Quaternion NumC::Rotations::Quaternion::operator* (
            const Quaternion & inRhs ) const [inline]
```

multiplication operator

Parameters

Quaternion	

Returns

Quaternion

6.21.3.15 operator*() [2/3]

```
Quaternion NumC::Rotations::Quaternion::operator* (
            double inScalar ) const  [inline]
```

multiplication operator, only useful for multiplying by negative 1, all others will be renormalized back out

Parameters

scalar	value

Returns

Quaternion

6.21.3.16 operator*() [3/3]

```
template<typename dtype >
NdArray<double> NumC::Rotations::Quaternion::operator* (
            const NdArray< dtype > & inVec ) const  [inline]
```

multiplication operator

Parameters

NdArray	

Returns

NdArray

6.21.3.17 operator*=() [1/2]

```
Quaternion& NumC::Rotations::Quaternion::operator*= (
            const Quaternion & inRhs )  [inline]
```

multiplication assignment operator

Parameters

Quaternion	

Returns

Quaternion

6.21.3.18 operator*=() [2/2]

```
Quaternion& NumC::Rotations::Quaternion::operator*= (
            double inScalar )  [inline]
```

multiplication operator, only useful for multiplying by negative 1, all others will be renormalized back out

Parameters

scalar	value

Returns

Quaternion

6.21.3.19 operator+()

```
Quaternion NumC::Rotations::Quaternion::operator+ (
            const Quaternion & inRhs ) const  [inline]
```

addition operator

Parameters

Quaternion	

Returns

Quaternion

6.21.3.20 operator+=()

```
Quaternion& NumC::Rotations::Quaternion::operator+= (
            const Quaternion & inRhs ) [inline]
```

addition assignment operator

Parameters

Quaternion	

Returns

Quaternion

6.21.3.21 operator-()

```
Quaternion NumC::Rotations::Quaternion::operator- (
            const Quaternion & inRhs ) const [inline]
```

subtraction operator

Parameters

Quaternion	

Returns

Quaternion

6.21.3.22 operator-=()

```
Quaternion& NumC::Rotations::Quaternion::operator-= (
            const Quaternion & inRhs ) [inline]
```

subtraction assignment operator

Parameters

Quaternion	

Returns

Quaternion

6.21.3.23 operator/()

```
Quaternion NumC::Rotations::Quaternion::operator/ (
            const Quaternion & inRhs ) const [inline]
```

division operator

Parameters

Quaternion	

Returns

Quaternion

6.21.3.24 operator/=()

```
Quaternion& NumC::Rotations::Quaternion::operator/= (
            const Quaternion & inRhs ) [inline]
```

division assignment operator

Parameters

Quaternion	

Returns

Quaternion

6.21.3.25 operator==()

```
bool NumC::Rotations::Quaternion::operator== (
            const Quaternion & inRhs ) const [inline]
```

equality operator

Parameters

Quaternion	

Returns

bool

6.21.3.26 print()

```
void NumC::Rotations::Quaternion::print ( ) const [inline]
```

prints the Quaternion to the console

Parameters

None	

Returns

None

6.21.3.27 rotate()

```
template<typename dtype >
NdArray<double> NumC::Rotations::Quaternion::rotate (
            const NdArray< dtype > & inVector ) const [inline]
```

rotate a vector using the quaternion

Parameters

cartesian	vector with x,y,z components

Returns

cartesian vector with x,y,z components

6.21.3.28 s()

```
double NumC::Rotations::Quaternion::s ( ) const [inline]
```

returns the s component

Parameters

None	

Returns

s

6.21.3.29 slerp() [1/2]

```
static Quaternion NumC::Rotations::Quaternion::slerp (
            const Quaternion & inQuat1,
            const Quaternion & inQuat2,
            double inPercent )  [inline], [static]
```

spherical linear interpolates between the two quaternions

Parameters

Quaternion	1
Quaternion	2
percent	(0, 1)

Returns

Quaternion

6.21.3.30 slerp() [2/2]

```
Quaternion NumC::Rotations::Quaternion::slerp (
            const Quaternion & inQuat2,
            double inPercent ) const  [inline]
```

spherical linear interpolates between the two quaternions

Parameters

Quaternion	2
percent	(0, 1)

Returns

Quaternion

6.21.3.31 str()

```
std::string NumC::Rotations::Quaternion::str ( ) const [inline]
```

returns the quaternion as a string representation

Parameters

None	

Returns

string

6.21.3.32 toDCM()

```
NdArray<double> NumC::Rotations::Quaternion::toDCM ( ) const [inline]
```

returns the direction cosine matrix

Parameters

None	

Returns

NdArray

6.21.3.33 toNdArray()

```
NdArray<double> NumC::Rotations::Quaternion::toNdArray ( ) const [inline]
```

returns the quaternion as an NdArray

Parameters

None	

Returns

NdArray

6.21.3.34 xRotation()

```
static Quaternion NumC::Rotations::Quaternion::xRotation (
            double inAngle ) [inline], [static]
```

returns a quaternion to rotate about the x-axis by the input angle

Parameters

angle	in radians

Returns

Quaternion

6.21.3.35 yRotation()

```
static Quaternion NumC::Rotations::Quaternion::yRotation (
            double inAngle ) [inline], [static]
```

returns a quaternion to rotate about the y-axis by the input angle

Parameters

angle	in radians

Returns

Quaternion

6.21.3.36 zRotation()

```
static Quaternion NumC::Rotations::Quaternion::zRotation (
            double inAngle ) [inline], [static]
```

returns a quaternion to rotate about the y-axis by the input angle

Parameters

angle	in radians

Returns

Quaternion

6.21.4 Friends And Related Function Documentation

6.21.4.1 operator<<

```
std::ostream& operator<< (
            std::ostream & inOStream,
            const Quaternion & inQuat ) [friend]
```

IO operator for the Quaternion class

Parameters

ostream	
Quaternion	

Returns

ostream&

The documentation for this class was generated from the following file:

- Rotations.hpp

6.22 NumC::Coordinates::RA< dtype > Class Template Reference

Holds a right ascension object.

```
#include <Coordinates.hpp>
```

Public Member Functions

- RA ()
- RA (dtype inDegrees)
- RA (uint8 inHours, uint8 inMinutes, dtype inSeconds)
- template<typename dtypeOut >
  RA< dtypeOut > astype ()
- dtype degrees () const
- uint8 hours () const
- uint8 minutes () const
- bool operator!= (const RA< dtype > &inRhs) const
- bool operator== (const RA< dtype > &inRhs) const
- void print () const
- dtype radians () const
- dtype seconds () const
- std::string str () const

Friends

- std::ostream & operator<< (std::ostream &inStream, const RA< dtype > &inRa)

6.22.1 Detailed Description

template<typename dtype>
class NumC::Coordinates::RA< dtype >

Holds a right ascension object.

6.22.2 Constructor & Destructor Documentation

6.22.2.1 RA() [1/3]

```
template<typename dtype>
NumC::Coordinates::RA< dtype >::RA ( )  [inline]
```

Default Constructor, not super usefull on its own

Parameters

None	

Returns

None

6.22.2.2 RA() [2/3]

```
template<typename dtype>
NumC::Coordinates::RA< dtype >::RA (
            dtype inDegrees )  [inline]
```

Constructor

Parameters

degrees	

Returns

None

6.22.2.3 RA() [3/3]

```
template<typename dtype>
NumC::Coordinates::RA< dtype >::RA (
            uint8 inHours,
            uint8 inMinutes,
            dtype inSeconds )  [inline]
```

Constructor

Parameters

hours	
minutes	
seconds	

Returns

None

6.22.3 Member Function Documentation

6.22.3.1 astype()

```
template<typename dtype>
template<typename dtypeOut >
RA<dtypeOut> NumC::Coordinates::RA< dtype >::astype ( )  [inline]
```

Returns a copy of the RA object as a different type

Parameters

None	

Returns

RA

6.22.3.2 degrees()

```
template<typename dtype>
dtype NumC::Coordinates::RA< dtype >::degrees ( ) const [inline]
```

Get the degrees value

Parameters

None	

Returns

degrees

6.22.3.3 hours()

```
template<typename dtype>
uint8 NumC::Coordinates::RA< dtype >::hours ( ) const [inline]
```

Get the hour value

Parameters

None	

Returns

hours

6.22.3.4 minutes()

```
template<typename dtype>
uint8 NumC::Coordinates::RA< dtype >::minutes ( ) const [inline]
```

Get the minute value

Parameters

None	

Returns

minutes

6.22.3.5 operator"!=()

```
template<typename dtype>
bool NumC::Coordinates::RA< dtype >::operator!= (
            const RA< dtype > & inRhs ) const [inline]
```

Not equality operator

Parameters

None	

Returns

bool

6.22.3.6 operator==()

```
template<typename dtype>
bool NumC::Coordinates::RA< dtype >::operator== (
            const RA< dtype > & inRhs ) const [inline]
```

Equality operator

Parameters

None	

Returns

bool

6.22.3.7 print()

```
template<typename dtype>
void NumC::Coordinates::RA< dtype >::print ( ) const [inline]
```

Prints the RA object to the console

Parameters

None	

Returns

None

6.22.3.8 radians()

```
template<typename dtype>
dtype NumC::Coordinates::RA< dtype >::radians ( ) const [inline]
```

Get the radians value

Parameters

None	

Returns

radians

6.22.3.9 seconds()

```
template<typename dtype>
dtype NumC::Coordinates::RA< dtype >::seconds ( ) const [inline]
```

Get the seconds value

Parameters

None	

Returns

seconds

6.22.3.10 str()

```
template<typename dtype>
std::string NumC::Coordinates::RA< dtype >::str ( ) const [inline]
```

Return the RA object as a string representation

Parameters

None	

Returns

string

6.22.4 Friends And Related Function Documentation

6.22.4.1 operator<<

```
template<typename dtype>
std::ostream& operator<< (
            std::ostream & inStream,
            const RA< dtype > & inRa )  [friend]
```

Ostream operator

Parameters

None	

Returns

None

The documentation for this class was generated from the following file:

- Coordinates.hpp

6.23 NumC::Random< dtype > Class Template Reference

A class for generating random numbers.

```
#include <Random.hpp>
```

Static Public Member Functions

- static NdArray< dtype > bernoulli (const Shape &inShape, dtype inP)
- static NdArray< dtype > beta (const Shape &inShape, dtype inAlpha, dtype inBeta)
- static NdArray< dtype > binomial (const Shape &inShape, dtype inN, double inP=0.5)
- static NdArray< dtype > cauchy (const Shape &inShape, dtype inMean=0, dtype inSigma=1)
- static NdArray< dtype > chiSquare (const Shape &inShape, dtype inDof)

- static dtype choice (const NdArray< dtype > &inArray)
- static NdArray< dtype > discrete (const Shape &inShape, const NdArray< double > &inWeights)
- static NdArray< dtype > exponential (const Shape &inShape, dtype inScaleValue=1)
- static NdArray< dtype > extremeValue (const Shape &inShape, dtype inA=1, dtype inB=1)
- static NdArray< dtype > f (const Shape &inShape, dtype inDofN, dtype inDofD)
- static NdArray< dtype > gamma (const Shape &inShape, dtype inGammaShape, dtype inScaleValue=1)
- static NdArray< dtype > geometric (const Shape &inShape, double inP=0.5)
- static NdArray< dtype > laplace (const Shape &inShape, dtype inLoc=0, dtype inScale=1)
- static NdArray< dtype > lognormal (const Shape &inShape, dtype inMean=0, dtype inSigma=1)
- static NdArray< dtype > negativeBinomial (const Shape &inShape, dtype inN, double inP=0.5)
- static NdArray< dtype > nonCentralChiSquared (const Shape &inShape, dtype inK=1, dtype inLambda=1)
- static NdArray< dtype > normal (const Shape &inShape, dtype inMean=0, dtype inSigma=1)
- static NdArray< dtype > permutation (dtype inValue)
- static NdArray< dtype > permutation (const NdArray< dtype > &inArray)
- static NdArray< dtype > poisson (const Shape &inShape, double inMean=1)
- static NdArray< dtype > rand (const Shape &inShape)
- static NdArray< dtype > randFloat (const Shape &inShape, dtype inLow, dtype inHigh)
- static NdArray< dtype > randInt (const Shape &inShape, dtype inLow, dtype inHigh)
- static NdArray< dtype > randN (const Shape &inShape)
- static void seed (uint32 inSeed)
- static void shuffle (NdArray< dtype > &inArray)
- static NdArray< dtype > standardNormal (const Shape &inShape)
- static NdArray< dtype > studentT (const Shape &inShape, dtype inDof)
- static NdArray< dtype > triangle (const Shape &inShape, dtype inA=0, dtype inB=0.5, dtype inC=1)
- static NdArray< dtype > uniform (const Shape &inShape, dtype inLow, dtype inHigh)
- static NdArray< dtype > uniformOnSphere (uint32 inNumPoints, uint32 inDims=2)
- static NdArray< dtype > weibull (const Shape &inShape, dtype inA=1, dtype inB=1)

6.23.1 Detailed Description

template<typename dtype>
class NumC::Random< dtype >

A class for generating random numbers.

6.23.2 Member Function Documentation

6.23.2.1 bernoulli()

```
template<typename dtype >
static NdArray<dtype> NumC::Random< dtype >::bernoulli (
            const Shape & inShape,
            dtype inP )  [inline], [static]
```

Create an array of the given shape and populate it with random samples from the ŞbernoulliŤ distribution.

Parameters

Shape	
probablity	of success [0, 1]

Returns

NdArray

6.23.2.2 beta()

```
template<typename dtype >
static NdArray<dtype> NumC::Random< dtype >::beta (
            const Shape & inShape,
            dtype inAlpha,
            dtype inBeta ) [inline], [static]
```

Create an array of the given shape and populate it with random samples from the ŞbetaŤ distribution.

NumPy Reference: https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.beta.html#numpy.random.beta

Parameters

Shape	
alpha	
beta	

Returns

NdArray

6.23.2.3 binomial()

```
template<typename dtype >
static NdArray<dtype> NumC::Random< dtype >::binomial (
            const Shape & inShape,
            dtype inN,
            double inP = 0.5 ) [inline], [static]
```

Create an array of the given shape and populate it with random samples from the ŞbinomialŤ distribution.

NumPy Reference: https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.binomial.html#numpy.random.binomial

Parameters

Shape	
number	of trials
probablity	of success [0, 1]

Returns

NdArray

6.23.2.4 cauchy()

```
template<typename dtype >
static NdArray<dtype> NumC::Random< dtype >::cauchy (
            const Shape & inShape,
            dtype inMean = 0,
            dtype inSigma = 1 ) [inline], [static]
```

Create an array of the given shape and populate it with random samples from a "cauchy" distrubution.

Parameters

mean	Mean value of the underlying normal distribution. Default is 0.
sigma,Standard	deviation of the underlying normal distribution. Should be greater than zero. Default is 1.

Returns

NdArray

6.23.2.5 chiSquare()

```
template<typename dtype >
static NdArray<dtype> NumC::Random< dtype >::chiSquare (
            const Shape & inShape,
            dtype inDof ) [inline], [static]
```

Create an array of the given shape and populate it with random samples from the Şchi squareŤ distribution.

NumPy Reference: https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.chisquare.html#numpy.random.chisquare

Parameters

Shape	
df	independent random variables

Returns

NdArray

6.23.2.6 choice()

```
template<typename dtype >
static dtype NumC::Random< dtype >::choice (
            const NdArray< dtype > & inArray )  [inline], [static]
```

Generates a random sample from an input array

Parameters

NdArray	

Returns

NdArray

6.23.2.7 discrete()

```
template<typename dtype >
static NdArray<dtype> NumC::Random< dtype >::discrete (
            const Shape & inShape,
            const NdArray< double > & inWeights )  [inline], [static]
```

Create an array of the given shape and populate it with random samples from a "discrete" distrubution. It produces integers in the range [0, n) with the probability of producing each value is specified by the parameters of the distribution.

Parameters

NdArray	of weights,

Returns

NdArray

6.23.2.8 exponential()

```
template<typename dtype >
static NdArray<dtype> NumC::Random< dtype >::exponential (
```

```
            const Shape & inShape,
            dtype inScaleValue = 1 )  [inline], [static]
```

Create an array of the given shape and populate it with random samples from a "exponential" distrubution.

NumPy Reference: https://docs.scipy.org/doc/numpy/reference/generated/numpy.↩
random.exponential.html#numpy.random.exponential

Parameters

Shape	
scale	value, default 1

Returns

NdArray

6.23.2.9 extremeValue()

```
template<typename dtype >
static NdArray<dtype> NumC::Random< dtype >::extremeValue (
            const Shape & inShape,
            dtype inA = 1,
            dtype inB = 1 )  [inline], [static]
```

Create an array of the given shape and populate it with random samples from a "extreme value" distrubution.

Parameters

Shape	
a,default	1
b,default	1

Returns

NdArray

6.23.2.10 f()

```
template<typename dtype >
static NdArray<dtype> NumC::Random< dtype >::f (
            const Shape & inShape,
            dtype inDofN,
            dtype inDofD )  [inline], [static]
```

Create an array of the given shape and populate it with random samples from a "F" distrubution.

NumPy Reference: https://docs.scipy.org/doc/numpy/reference/generated/numpy.↩
random.f.html#numpy.random.f

Parameters

Shape	
Degrees	of freedom in numerator. Should be greater than zero.
Degrees	of freedom in denominator. Should be greater than zero.

Returns

NdArray

6.23.2.11 gamma()

```
template<typename dtype >
static NdArray<dtype> NumC::Random< dtype >::gamma (
            const Shape & inShape,
            dtype inGammaShape,
            dtype inScaleValue = 1 )  [inline], [static]
```

Create an array of the given shape and populate it with random samples from a "gamma" distrubution.

NumPy Reference: https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.gamma.html#numpy.random.gamma

Parameters

Shape	
Scale,default	1
Gamma	shape

Returns

NdArray

6.23.2.12 geometric()

```
template<typename dtype >
static NdArray<dtype> NumC::Random< dtype >::geometric (
            const Shape & inShape,
            double inP = 0.5 )  [inline], [static]
```

Create an array of the given shape and populate it with random samples from a "geometric" distrubution.

NumPy Reference: https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.geometric.html#numpy.random.geometric

Parameters

Shape	
probablity	of success [0, 1]

Returns

NdArray

6.23.2.13 laplace()

```
template<typename dtype >
static NdArray<dtype> NumC::Random< dtype >::laplace (
            const Shape & inShape,
            dtype inLoc = 0,
            dtype inScale = 1 ) [inline], [static]
```

Create an array of the given shape and populate it with random samples from a "laplace" distrubution.

NumPy Reference: https://docs.scipy.org/doc/numpy/reference/generated/numpy.↩ random.laplace.html#numpy.random.laplace

Parameters

inLoc	The position, mu, of the distribution peak. Default is 0.
inScale	float optional, the exponential decay. Default is 1.

Returns

NdArray

6.23.2.14 lognormal()

```
template<typename dtype >
static NdArray<dtype> NumC::Random< dtype >::lognormal (
            const Shape & inShape,
            dtype inMean = 0,
            dtype inSigma = 1 ) [inline], [static]
```

Create an array of the given shape and populate it with random samples from a "lognormal" distrubution.

NumPy Reference: https://docs.scipy.org/doc/numpy/reference/generated/numpy.↩ random.lognormal.html#numpy.random.lognormal

Parameters

mean	Mean value of the underlying normal distribution. Default is 0.
sigma,Standard	deviation of the underlying normal distribution. Should be greater than zero. Default is 1.

Returns

NdArray

6.23.2.15 negativeBinomial()

```
template<typename dtype >
static NdArray<dtype> NumC::Random< dtype >::negativeBinomial (
            const Shape & inShape,
            dtype inN,
            double inP = 0.5 )  [inline], [static]
```

Create an array of the given shape and populate it with random samples from the Şnegative BinomialŤ distribution.

NumPy Reference: https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.negative_binomial.html#numpy.random.negative_binomial

Parameters

Shape	
number	of trials
probablity	of success [0, 1]

Returns

NdArray

6.23.2.16 nonCentralChiSquared()

```
template<typename dtype >
static NdArray<dtype> NumC::Random< dtype >::nonCentralChiSquared (
            const Shape & inShape,
            dtype inK = 1,
            dtype inLambda = 1 )  [inline], [static]
```

Create an array of the given shape and populate it with random samples from a "non central chi squared" distrubution.

NumPy Reference: https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.noncentral_chisquare.html#numpy.random.noncentral_chisquare

Parameters

Shape	
k,default	1
lambda,default	1

Returns

NdArray

6.23.2.17 normal()

```
template<typename dtype >
static NdArray<dtype> NumC::Random< dtype >::normal (
            const Shape & inShape,
            dtype inMean = 0,
            dtype inSigma = 1 )  [inline], [static]
```

Create an array of the given shape and populate it with random samples from a "normal" distrubution.

NumPy Reference: https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.normal.html#numpy.random.normal

Parameters

mean	Mean value of the underlying normal distribution. Default is 0.
sigma,Standard	deviation of the underlying normal distribution. Should be greater than zero. Default is 1.

Returns

NdArray

6.23.2.18 permutation() [1/2]

```
template<typename dtype >
static NdArray<dtype> NumC::Random< dtype >::permutation (
            dtype inValue )  [inline], [static]
```

Randomly permute a sequence, or return a permuted range. If x is an integer, randomly permute np.arange(x). If x is an array, make a copy and shuffle the elements randomly.

Parameters

value	

Returns

NdArray

6.23.2.19 permutation() [2/2]

```
template<typename dtype >
static NdArray<dtype> NumC::Random< dtype >::permutation (
            const NdArray< dtype > & inArray )  [inline], [static]
```

Randomly permute a sequence, or return a permuted range. If x is an integer, randomly permute np.arange(x). If x is an array, make a copy and shuffle the elements randomly.

Parameters

NdArray	

Returns

NdArray

6.23.2.20 poisson()

```
template<typename dtype >
static NdArray<dtype> NumC::Random< dtype >::poisson (
            const Shape & inShape,
            double inMean = 1 )  [inline], [static]
```

Create an array of the given shape and populate it with random samples from the ŞpoissonŤ distribution.

NumPy Reference: https://docs.scipy.org/doc/numpy/reference/generated/numpy.↩ random.poisson.html#numpy.random.poisson

Parameters

Shape	
mean,default	1

Returns

NdArray

6.23.2.21 rand()

```
template<typename dtype >
static NdArray<dtype> NumC::Random< dtype >::rand (
            const Shape & inShape )  [inline], [static]
```

Create an array of the given shape and populate it with random samples from a uniform distribution over [0, 1).

NumPy Reference: https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand.html#numpy.random.rand

Parameters

Shape	

Returns

NdArray

6.23.2.22 randFloat()

```
template<typename dtype >
static NdArray<dtype> NumC::Random< dtype >::randFloat (
            const Shape & inShape,
            dtype inLow,
            dtype inHigh )  [inline], [static]
```

Return random floats from low (inclusive) to high (exclusive), with the given shape

NumPy Reference: https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.ranf.html#numpy.random.ranf

Parameters

Shape	
low	value
high	value

Returns

NdArray

6.23.2.23 randInt()

```
template<typename dtype >
static NdArray<dtype> NumC::Random< dtype >::randInt (
```

```
            const Shape & inShape,
            dtype inLow,
            dtype inHigh )  [inline], [static]
```

Return random integers from low (inclusive) to high (exclusive), with the given shape

NumPy Reference: https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.randint.html#numpy.random.randint

Parameters

Shape	
low	value
high	value

Returns

NdArray

6.23.2.24 randN()

```
template<typename dtype >
static NdArray<dtype> NumC::Random< dtype >::randN (
            const Shape & inShape )  [inline], [static]
```

Create an array of the given shape and populate it with random samples from the Şstandard normalŤ distribution.

NumPy Reference: https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.randn.html#numpy.random.randn

Parameters

Shape	

Returns

NdArray

6.23.2.25 seed()

```
template<typename dtype >
static void NumC::Random< dtype >::seed (
            uint32 inSeed )  [inline], [static]
```

Seeds the random number generator_

NumPy Reference: https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.seed.html#numpy.random.seed

Parameters

seed	

Returns

None

6.23.2.26 shuffle()

```
template<typename dtype >
static void NumC::Random< dtype >::shuffle (
            NdArray< dtype > & inArray )  [inline], [static]
```

Modify a sequence in-place by shuffling its contents.

Parameters

NdArray	

Returns

None

6.23.2.27 standardNormal()

```
template<typename dtype >
static NdArray<dtype> NumC::Random< dtype >::standardNormal (
            const Shape & inShape )  [inline], [static]
```

Create an array of the given shape and populate it with random samples from a "standard normal" distrubution with mean = 0 and std = 1

NumPy Reference: https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.standard_normal.html#numpy.random.standard_normal

Parameters

Shape	

Returns

NdArray

6.23.2.28 studentT()

```
template<typename dtype >
static NdArray<dtype> NumC::Random< dtype >::studentT (
            const Shape & inShape,
            dtype inDof ) [inline], [static]
```

Create an array of the given shape and populate it with random samples from the Şstudent-TŤ distribution.

NumPy Reference: https://docs.scipy.org/doc/numpy/reference/generated/numpy.↩
random.standard_t.html#numpy.random.standard_t

Parameters

Shape	
df	independent random variables

Returns

NdArray

6.23.2.29 triangle()

```
template<typename dtype >
static NdArray<dtype> NumC::Random< dtype >::triangle (
            const Shape & inShape,
            dtype inA = 0,
            dtype inB = 0.5,
            dtype inC = 1 ) [inline], [static]
```

Create an array of the given shape and populate it with random samples from the ŞtriangleŤ distribution.

NumPy Reference: https://docs.scipy.org/doc/numpy/reference/generated/numpy.↩
random.triangular.html#numpy.random.triangular

Parameters

Shape	
a	
b	
c	

Returns

NdArray

6.23.2.30 uniform()

```
template<typename dtype >
static NdArray<dtype> NumC::Random< dtype >::uniform (
            const Shape & inShape,
            dtype inLow,
            dtype inHigh ) [inline], [static]
```

Draw samples from a uniform distribution.

Samples are uniformly distributed over the half - open interval[low, high) (includes low, but excludes high)

NumPy Reference: https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.uniform.html#numpy.random.uniform

Parameters

Shape	
low	value
high	value

Returns

NdArray

6.23.2.31 uniformOnSphere()

```
template<typename dtype >
static NdArray<dtype> NumC::Random< dtype >::uniformOnSphere (
            uint32 inNumPoints,
            uint32 inDims = 2 ) [inline], [static]
```

Such a distribution produces random numbers uniformly distributed on the unit sphere of arbitrary dimension dim.

Parameters

number	of points
dimension	of the sphere, default 2

Returns

NdArray

6.23.2.32 weibull()

```
template<typename dtype >
static NdArray<dtype> NumC::Random< dtype >::weibull (
```

```
const Shape & inShape,
dtype inA = 1,
dtype inB = 1 )  [inline], [static]
```

Create an array of the given shape and populate it with random samples from the "weibull" distribution.

NumPy Reference: https://docs.scipy.org/doc/numpy/reference/generated/numpy.↩
random.weibull.html#numpy.random.weibull

Parameters

Shape	
a,default	1
b,default	1

Returns

NdArray

The documentation for this class was generated from the following file:

- Random.hpp

6.24 NumC::Shape Class Reference

A Shape Class for NdArrays.

```
#include <Shape.hpp>
```

Public Member Functions

- Shape ()
- Shape (uint32 inSquareSize)
- Shape (uint32 inRows, uint32 inCols)
- bool isnull ()
- bool operator!= (const Shape &inOtherShape) const
- bool operator== (const Shape &inOtherShape) const
- void print () const
- uint32 size () const
- std::string str () const

Data Fields

- uint32 cols
- uint32 rows

Friends

- std::ostream & operator<< (std::ostream &inOStream, const Shape &inShape)

6.24.1 Detailed Description

A Shape Class for NdArrays.

6.24.2 Constructor & Destructor Documentation

6.24.2.1 Shape() [1/3]

```
NumC::Shape::Shape ( )  [inline]
```

Constructor

Parameters

number	of rows
number	of cols

Returns

None

6.24.2.2 Shape() [2/3]

```
NumC::Shape::Shape (
            uint32 inSquareSize )  [inline], [explicit]
```

Constructor

Parameters

number	of rows and cols

Returns

None

6.24.2.3 Shape() [3/3]

```
NumC::Shape::Shape (
            uint32 inRows,
            uint32 inCols )  [inline]
```

Constructor

Parameters

number	of rows
number	of cols

Returns

None

6.24.3 Member Function Documentation

6.24.3.1 isnull()

```
bool NumC::Shape::isnull ( )  [inline]
```

Returns whether the shape is null (constructed with the default constructor).

Parameters

None	

Returns

bool

6.24.3.2 operator"!=()

```
bool NumC::Shape::operator!= (
             const Shape & inOtherShape ) const  [inline]
```

Not equality operator

Parameters

None	

Returns

None

6.24.3.3 operator==()

```
bool NumC::Shape::operator== (
            const Shape & inOtherShape ) const  [inline]
```

Equality operator

Parameters

None	

Returns

None

6.24.3.4 print()

```
void NumC::Shape::print ( ) const  [inline]
```

Prints the shape to the console

Parameters

None	

Returns

None

6.24.3.5 size()

```
uint32 NumC::Shape::size ( ) const  [inline]
```

Returns the size of the shape

Parameters

None	

Returns

size

6.24.3.6 str()

```
std::string NumC::Shape::str ( ) const  [inline]
```

Returns the shape as a string representation

Parameters

None	

Returns

string

6.24.4 Friends And Related Function Documentation

6.24.4.1 operator<<

```
std::ostream& operator<< (
            std::ostream & inOStream,
            const Shape & inShape )  [friend]
```

IO operator for the Shape class

Parameters

None	

Returns

None

6.24.5 Field Documentation

6.24.5.1 cols

```
uint32 NumC::Shape::cols
```

6.24.5.2 rows

```
uint32 NumC::Shape::rows
```

The documentation for this class was generated from the following file:

- Shape.hpp

6.25 NumC::Coordinates::Sign Struct Reference

Struct Enum for positive or negative Dec angle.

```
#include <Coordinates.hpp>
```

Public Types

- enum Type { NEGATIVE = 0, POSITIVE }

6.25.1 Detailed Description

Struct Enum for positive or negative Dec angle.

6.25.2 Member Enumeration Documentation

6.25.2.1 Type

```
enum NumC::Coordinates::Sign::Type
```

Enumerator

NEGATIVE	
POSITIVE	

The documentation for this struct was generated from the following file:

- Coordinates.hpp

6.26 NumC::Slice Class Reference

A Class for slicing into NdArrays.

```
#include <Slice.hpp>
```

Public Member Functions

- Slice ()
- Slice (int32 inStop)
- Slice (int32 inStart, int32 inStop)
- Slice (int32 inStart, int32 inStop, int32 inStep)
- void makePositiveAndValidate (uint32 inArraySize)
- uint32 numElements (uint32 inArraySize)
- void print ()
- std::string str () const

Data Fields

- int32 start
- int32 step
- int32 stop

Friends

- std::ostream & operator<< (std::ostream &inOStream, const Slice &inSlice)

6.26.1 Detailed Description

A Class for slicing into NdArrays.

6.26.2 Constructor & Destructor Documentation

6.26.2.1 Slice() [1/4]

`NumC::Slice::Slice ( ) [inline]`

Constructor

Parameters

None	

Returns

None

6.26.2.2 Slice() [2/4]

```
NumC::Slice::Slice (
            int32 inStop )  [inline], [explicit]
```

Constructor

Parameters

stop	index (not included)

Returns

None

6.26.2.3 Slice() [3/4]

```
NumC::Slice::Slice (
            int32 inStart,
            int32 inStop )  [inline]
```

Constructor

Parameters

start	index,
stop	index (not included)

Returns

None

6.26.2.4 Slice() [4/4]

```
NumC::Slice::Slice (
            int32 inStart,
            int32 inStop,
            int32 inStep )  [inline]
```

Constructor

Parameters

start	index,
stop	index (not included)
step	value

Returns

None

6.26.3 Member Function Documentation

6.26.3.1 makePositiveAndValidate()

```
void NumC::Slice::makePositiveAndValidate (
            uint32 inArraySize )  [inline]
```

Make the slice all positive and does some error checking

Parameters

The	calling array size

Returns

None

6.26.3.2 numElements()

```
uint32 NumC::Slice::numElements (
            uint32 inArraySize )  [inline]
```

Returns the number of elements that the slice contains. be aware that this method will also make the slice all positive!

Parameters

The	calling array size

Returns

None

6.26.3.3 print()

```
void NumC::Slice::print ( )  [inline]
```

Prints the shape to the console

Parameters

None	

Returns

None

6.26.3.4 str()

```
std::string NumC::Slice::str ( ) const  [inline]
```

Prints the shape to the console

Parameters

None	

Returns

None

6.26.4 Friends And Related Function Documentation

6.26.4.1 operator<<

```
std::ostream& operator<< (
            std::ostream & inOStream,
            const Slice & inSlice )  [friend]
```

IO operator for the Slice class

Parameters

None	

Returns

None

6.26.5 Field Documentation

6.26.5.1 start

`int32 NumC::Slice::start`

6.26.5.2 step

`int32 NumC::Slice::step`

6.26.5.3 stop

`int32 NumC::Slice::stop`

The documentation for this class was generated from the following file:

- Slice.hpp

6.27 NumC::Timer< TimeUnit > Class Template Reference

A timer class for timing code execution.

`#include <Timer.hpp>`

Public Types

- typedef std::chrono::high_resolution_clock ChronoClock
- typedef std::chrono::time_point< ChronoClock > TimePoint

Public Member Functions

- Timer ()
- Timer (const std::string &inName)
- void tic ()
- int64 toc ()

6.27.1 Detailed Description

template<typename TimeUnit = std::chrono::milliseconds>
class NumC::Timer< TimeUnit >

A timer class for timing code execution.

6.27.2 Member Typedef Documentation

6.27.2.1 ChronoClock

```
template<typename TimeUnit = std::chrono::milliseconds>
typedef std::chrono::high_resolution_clock NumC::Timer< TimeUnit >::ChronoClock
```

6.27.2.2 TimePoint

```
template<typename TimeUnit = std::chrono::milliseconds>
typedef std::chrono::time_point<ChronoClock> NumC::Timer< TimeUnit >::TimePoint
```

6.27.3 Constructor & Destructor Documentation

6.27.3.1 Timer() [1/2]

```
template<typename TimeUnit = std::chrono::milliseconds>
NumC::Timer< TimeUnit >::Timer ( ) [inline]
```

Constructor

Parameters

None	

Returns

None

6.27.3.2 Timer() [2/2]

```
template<typename TimeUnit = std::chrono::milliseconds>
NumC::Timer< TimeUnit >::Timer (
            const std::string & inName ) [inline]
```

Constructor

Parameters

Timer	name

Returns

None

6.27.4 Member Function Documentation

6.27.4.1 tic()

```
template<typename TimeUnit = std::chrono::milliseconds>
void NumC::Timer< TimeUnit >::tic ( )  [inline]
```

Starts the timer

Parameters

None	

Returns

None

6.27.4.2 toc()

```
template<typename TimeUnit = std::chrono::milliseconds>
int64 NumC::Timer< TimeUnit >::toc ( )  [inline]
```

Method Description: Stops the timer

Parameters

None	

Returns

ellapsed time in specified time units

The documentation for this class was generated from the following file:

- Timer.hpp

6.28 NumC::Utils< dtype > Class Template Reference

Usefull utility type functions.

```
#include <Utils.hpp>
```

Static Public Member Functions

- static dtype cube (dtype inValue)
- static std::string num2str (dtype inNumber)
- static dtype power (dtype inValue, uint8 inPower)
- static dtype sqr (dtype inValue)

6.28.1 Detailed Description

template<typename dtype>
class NumC::Utils< dtype >

Usefull utility type functions.

6.28.2 Member Function Documentation

6.28.2.1 cube()

```
template<typename dtype >
static dtype NumC::Utils< dtype >::cube (
            dtype inValue )  [inline], [static]
```

Cubes in input value

Parameters

dtype	

Returns

dtype

6.28.2.2 num2str()

```
template<typename dtype >
static std::string NumC::Utils< dtype >::num2str (
            dtype inNumber )  [inline], [static]
```

Converts the number into a string

Parameters

number	

Returns

string

6.28.2.3 power()

```
template<typename dtype >
static dtype NumC::Utils< dtype >::power (
            dtype inValue,
            uint8 inPower ) [inline], [static]
```

Raises the input value to a power

Parameters

dtype	

Returns

dtype

6.28.2.4 sqr()

```
template<typename dtype >
static dtype NumC::Utils< dtype >::sqr (
            dtype inValue ) [inline], [static]
```

Squares in input value

Parameters

dtype	

Returns

dtype

The documentation for this class was generated from the following file:

- Utils.hpp

Chapter 7

File Documentation

7.1 BoostNumpyNdarrayHelper.hpp File Reference

```
#include <NumC/NdArray.hpp>
#include <NumC/Types.hpp>
#include <cmath>
#include <vector>
#include <iostream>
#include <string>
#include <stdexcept>
#include <utility>
#include "boost/python.hpp"
#include "boost/python/numpy.hpp"
```

Data Structures

- class NumC::BoostNdarrayHelper

  Helper class for ndarray.
- struct NumC::Order

  C or Fortran ordering from python.

Namespaces

- NumC

Functions

- template< typename dtype >
  NdArray< dtype > NumC::boostToNumC (boost::python::numpy::ndarray &inArray)
- template< typename dtype >
  boost::python::numpy::ndarray NumC::numCToBoost (const NdArray< dtype > &inArray)

7.1.1 Detailed Description

Author

David Pilger dpilger26@gmail.com

Version

1.0

7.1.2 LICENSE

Copyright 2018 David Pilger

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files(the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions :

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

7.1.3 DESCRIPTION

A module for interacting with the boost numpy arrays

7.2 Constants.hpp File Reference

```
#include "NumC/Types.hpp"
#include <cmath>
#include <string>
```

Namespaces

- NumC
- NumC::Constants

    Holds usefull constants.

Variables

- const double NumC::Constants::c = 3.0e8

    speed of light
- const double NumC::Constants::DAYS_PER_WEEK = 7

    Number of days in a week.
- const double NumC::Constants::e = 2.718281828459045

    eulers number
- const double NumC::Constants::HOURS_PER_DAY = 24

    Number of hours in a day.
- const double NumC::Constants::MILLISECONDS_PER_DAY = SECONDS_PER_DAY ∗ MILLISECONDS↩_PER_SECOND

    Number of milliseconds in a day.
- const double NumC::Constants::MILLISECONDS_PER_SECOND = 1000

    Number of milliseconds in a second.
- const double NumC::Constants::MINUTES_PER_DAY = HOURS_PER_DAY ∗ MINUTES_PER_HOUR

    Number of minutes in a day.
- const double NumC::Constants::MINUTES_PER_HOUR = 60

    Number of minutes in an hour.
- const double NumC::Constants::nan = std::nan("1")

    NaN.
- const double NumC::Constants::pi = 3.14159265358979323846

    Pi.
- const double NumC::Constants::SECONDS_PER_DAY = MINUTES_PER_DAY ∗ SECONDS_PER_MINU↩TE

    Number of seconds in a day.
- const double NumC::Constants::SECONDS_PER_HOUR = MINUTES_PER_HOUR ∗ SECONDS_PER_↩MINUTE

    Number of seconds in an hour.
- const double NumC::Constants::SECONDS_PER_MINUTE = 60

    Number of seconds in a minute.
- const double NumC::Constants::SECONDS_PER_WEEK = SECONDS_PER_DAY ∗ DAYS_PER_WEEK

    Number of seconds in a week.
- const std::string NumC::Constants::VERSION = "1.0"

    Current NumC version number.

7.2.1 Detailed Description

Author

David Pilger dpilger26@gmail.com

Version

1.0

7.2.2 LICENSE

7.2.3 DESCRIPTION

Holds usefull constants

7.3 Coordinates.hpp File Reference

```
#include "NumC/DtypeInfo.hpp"
#include "NumC/NdArray.hpp"
#include "NumC/Methods.hpp"
#include "NumC/Types.hpp"
#include "NumC/Utils.hpp"
#include <iostream>
#include <stdexcept>
#include <string>
#include <utility>
```

Data Structures

- class NumC::Coordinates::Coordinate< dtype >

  Holds a full coordinate object.
- class NumC::Coordinates::Dec< dtype >

  Holds a Declination object.
- class NumC::Coordinates::RA< dtype >

  Holds a right ascension object.
- struct NumC::Coordinates::Sign

  Struct Enum for positive or negative Dec angle.

Namespaces

- NumC
- NumC::Coordinates

  A module for holding and working with coordinates in either Ra/Dec or cartesian formats.

Functions

- template< typename dtype >
  dtype NumC::Coordinates::degreeSeperation (const Coordinate< dtype > &inCoordinate1, const Coordinate< dtype > &inCoordinate2)
- template< typename dtype >
  dtype NumC::Coordinates::degreeSeperation (const NdArray< dtype > &inVector1, const NdArray< dtype > &inVector2)
- template< typename dtype >
  dtype NumC::Coordinates::radianSeperation (const Coordinate< dtype > &inCoordinate1, const Coordinate< dtype > &inCoordinate2)
- template< typename dtype >
  dtype NumC::Coordinates::radianSeperation (const NdArray< dtype > &inVector1, const NdArray< dtype > &inVector2)

7.3.1 Detailed Description

Author

David Pilger `dpilger26@gmail.com`

Version

1.0

7.3.2 LICENSE

7.3.3 DESCRIPTION

A module for holding and working with coordinates in either Ra/Dec or cartesian formats

7.4 DataCube.hpp File Reference

```
#include "NumC/NdArray.hpp"
#include "NumC/Types.hpp"
#include "boost/filesystem.hpp"
#include <deque>
#include <limits>
#include <stdexcept>
```

Data Structures

- class NumC::DataCube< dtype >

  Convience container for holding a uniform array of NdArrays.

Namespaces

- NumC

7.4.1 Detailed Description

Author

David Pilger dpilger26@gmail.com

Version

1.0

7.4.2 LICENSE

7.4.3 DESCRIPTION

Convience container for holding a uniform array of NdArrays

7.5 DtypeInfo.hpp File Reference

```
#include <limits>
```

Data Structures

- class NumC::DtypeInfo< dtype >
  Holds info about the dtype.

Namespaces

- NumC

7.5.1 Detailed Description

Author

David Pilger dpilger26@gmail.com

Version

1.0

7.5.2 LICENSE

7.5.3 DESCRIPTION

Holds info about the dtype

7.6 FFT.hpp File Reference

```
#include "NumC/NdArray.hpp"
#include "NumC/Types.hpp"
```

Data Structures

- class NumC::FFT< dtype >

  Class for performing fast forrier tranforms.

Namespaces

- NumC

7.6.1 Detailed Description

Author

David Pilger dpilger26@gmail.com

Version

1.0

7.6.2 LICENSE

7.6.3 DESCRIPTION

Class for performing fast forrier tranforms

7.7 Filter.hpp File Reference

```
#include <NumC/NdArray.hpp>
#include <NumC/Methods.hpp>
#include <NumC/Types.hpp>
#include <NumC/Utils.hpp>
#include <cmath>
#include <utility>
```

Data Structures

- struct NumC::Filter::Boundary

  Boundary condition to apply to the image filter.
- class NumC::Filters< dtype >

  Class for performing many types of image filtering.

Namespaces

- NumC
- NumC::Filter

  Image and signal filtering.

7.7.1 Detailed Description

Author

David Pilger dpilger26@gmail.com

Version

1.0

7.7.2 LICENSE

7.7.3 DESCRIPTION

Image and signal filtering

7.8 ImageProcessing.hpp File Reference

```
#include <NumC/NdArray.hpp>
#include <NumC/Methods.hpp>
#include <NumC/Types.hpp>
#include <NumC/Utils.hpp>
#include <cmath>
#include <iostream>
#include <limits>
#include <set>
#include <string>
#include <utility>
#include <vector>
```

Data Structures

- class NumC::ImageProcessing< dtype >::Centroid

  holds the information for a centroid
- class NumC::ImageProcessing< dtype >::Cluster

  Holds the information for a cluster of pixels.
- class NumC::ImageProcessing< dtype >

  Class for basic image processing.
- class NumC::ImageProcessing< dtype >::Pixel

  Holds the information for a single pixel.

Namespaces

- NumC

7.8.1 Detailed Description

Author

David Pilger dpilger26@gmail.com

Version

1.0

7.8.2 LICENSE

7.8.3 DESCRIPTION

A module for basic image processing

7.9 Linalg.hpp File Reference

```
#include "NumC/Methods.hpp"
#include "NumC/NdArray.hpp"
#include "NumC/Shape.hpp"
#include "NumC/Types.hpp"
#include <cmath>
#include <initializer_list>
#include <limits>
#include <stdexcept>
#include <utility>
```

Data Structures

- class NumC::Linalg< dtype >

  Class for doing linear algebra operations.

Namespaces

- NumC

7.9.1 Detailed Description

Author

David Pilger dpilger26@gmail.com

Version

1.0

7.9.2 LICENSE

7.9.3 DESCRIPTION

Class for doing linear algebra operations

7.10 Methods.hpp File Reference

```
#include "NumC/Constants.hpp"
#include "NumC/NdArray.hpp"
#include "NumC/Types.hpp"
#include "boost/filesystem.hpp"
#include <algorithm>
#include <cmath>
#include <fstream>
#include <initializer_list>
#include <iostream>
#include <set>
#include <sstream>
#include <stdexcept>
#include <string>
#include <utility>
#include <vector>
```

Data Structures

- class NumC::Methods< dtype >

    Methods for working with NdArrays.

Namespaces

- NumC

7.10.1 Detailed Description

Author

David Pilger dpilger26@gmail.com

Version

1.0

7.10.2 LICENSE

7.10.3 DESCRIPTION

Methods for working with NdArrays

7.11 NdArray.hpp File Reference

```
#include "NumC/DtypeInfo.hpp"
#include "NumC/Shape.hpp"
#include "NumC/Slice.hpp"
#include "NumC/Types.hpp"
#include "NumC/Utils.hpp"
#include "NumC/Constants.hpp"
#include <boost/filesystem.hpp>
#include <boost/endian/conversion.hpp>
#include <algorithm>
#include <cmath>
#include <fstream>
#include <initializer_list>
#include <iostream>
#include <numeric>
#include <set>
#include <stdexcept>
#include <string>
#include <utility>
#include <vector>
```

Data Structures

- class NumC::NdArray< dtype >

  Holds 1D and 2D arrays, the main work horse of the NumC library.

Namespaces

- NumC

7.11.1 Detailed Description

Author

David Pilger dpilger26@gmail.com

Version

1.0

7.11.2 LICENSE

7.11.3 DESCRIPTION

Holds 1D and 2D arrays, the main work horse of the NumC library

7.12 NumC.hpp File Reference

```
#include "NumC/BoostNumpyNdarrayHelper.hpp"
#include "NumC/Constants.hpp"
#include "NumC/Coordinates.hpp"
#include "NumC/DataCube.hpp"
#include "NumC/DtypeInfo.hpp"
#include "NumC/FFT.hpp"
#include "NumC/Filter.hpp"
#include "NumC/ImageProcessing.hpp"
#include "NumC/Linalg.hpp"
#include "NumC/Methods.hpp"
#include "NumC/NdArray.hpp"
#include "NumC/Polynomial.hpp"
#include "NumC/Random.hpp"
#include "NumC/Rotations.hpp"
#include "NumC/Shape.hpp"
#include "NumC/Slice.hpp"
#include "NumC/Timer.hpp"
#include "NumC/Types.hpp"
#include "NumC/Utils.hpp"
```

Macros

- #define _CRT_SECURE_NO_WARNINGS

7.12.1 Macro Definition Documentation

7.12.1.1 _CRT_SECURE_NO_WARNINGS

```
#define _CRT_SECURE_NO_WARNINGS
```

7.13 Polynomial.hpp File Reference

```
#include "NumC/Types.hpp"
#include "NumC/NdArray.hpp"
```

Data Structures

- class NumC::Polynomial< dtype >
  Class for dealing with common polynomials.

Namespaces

- NumC

7.13.1 Detailed Description

Author

David Pilger dpilger26@gmail.com

Version

1.0

7.13.2 LICENSE

7.13.3 DESCRIPTION

Class for dealing with common polynomials

7.14 Random.hpp File Reference

```
#include "NumC/Methods.hpp"
#include "NumC/NdArray.hpp"
#include "NumC/Shape.hpp"
#include "NumC/Types.hpp"
#include "boost/random.hpp"
#include <algorithm>
#include <vector>
```

Data Structures

- class NumC::Random< dtype >

  A class for generating random numbers.

Namespaces

- NumC

Variables

- boost::random::mt19937 NumC::generator_

  generator function

7.14.1 Detailed Description

Author

David Pilger dpilger26@gmail.com

Version

1.0

7.14.2 LICENSE

7.14.3 DESCRIPTION

A module for generating random numbers

7.15 Rotations.hpp File Reference

```
#include "NumC/Methods.hpp"
#include "NumC/Linalg.hpp"
#include "NumC/NdArray.hpp"
#include "NumC/Types.hpp"
#include "NumC/Utils.hpp"
#include <cmath>
#include <iostream>
#include <stdexcept>
#include <string>
```

Data Structures

- class NumC::Rotations::DCM< dtype >

  Factory methods for generating direction cosine matrices and vectors.
- class NumC::Rotations::Quaternion

  Holds a unit quaternion.

Namespaces

- NumC
- NumC::Rotations

  Module for dealing with rotations.

7.15.1 Detailed Description

Author

David Pilger dpilger26@gmail.com

Version

1.0

7.15.2 LICENSE

7.15.3 DESCRIPTION

Module for dealing with rotations

7.16 Shape.hpp File Reference

```
#include "NumC/Types.hpp"
#include "NumC/Utils.hpp"
#include <iostream>
#include <stdexcept>
#include <string>
```

Data Structures

- class NumC::Shape

  A Shape Class for NdArrays.

Namespaces

- NumC

7.16.1 Detailed Description

Author

David Pilger dpilger26@gmail.com

Version

1.0

7.16.2 LICENSE

Copyright 2018 David Pilger

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files(the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions :

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

7.16.3 DESCRIPTION

A Shape Class for NdArrays

7.17 Slice.hpp File Reference

```
#include "NumC/Types.hpp"
#include "NumC/Utils.hpp"
#include <iostream>
#include <stdexcept>
#include <string>
```

Data Structures

- class NumC::Slice

  A Class for slicing into NdArrays.

Namespaces

- NumC

7.17.1 Detailed Description

Author

David Pilger dpilger26@gmail.com

Version

1.0

7.17.2 LICENSE

7.17.3 DESCRIPTION

A Class for slicing into NdArrays

7.18 Timer.hpp File Reference

```
#include <chrono>
#include <string>
```

Data Structures

- class NumC::Timer< TimeUnit >

  A timer class for timing code execution.

Namespaces

- NumC

7.18.1 Detailed Description

Author

David Pilger dpilger26@gmail.com

Version

1.0

7.18.2 LICENSE

7.18.3 DESCRIPTION

A timer class for timing code execution

7.19 Types.hpp File Reference

```
#include <cstdint>
```

Data Structures

- struct NumC::Axis

  Enum To describe an axis.
- struct NumC::Endian

  Enum for endianess.

Namespaces

- NumC

Typedefs

- typedef int16_t NumC::int16
- typedef int32_t NumC::int32
- typedef int64_t NumC::int64
- typedef int8_t NumC::int8
- typedef uint16_t NumC::uint16
- typedef uint32_t NumC::uint32
- typedef uint64_t NumC::uint64
- typedef uint8_t NumC::uint8

7.19.1 Detailed Description

Author

David Pilger dpilger26@gmail.com

Version

1.0

7.19.2 LICENSE

7.19.3 DESCRIPTION

Usefull types

7.20 Utils.hpp File Reference

```
#include <string>
```

Data Structures

- class NumC::Utils< dtype >
  Usefull utility type functions.

Namespaces

- NumC

7.20.1 Detailed Description

Author

David Pilger dpilger26@gmail.com

Version

1.0

7.20.2 LICENSE

Copyright 2018 David Pilger

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files(the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions :

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, IN↩ CLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

7.20.3 DESCRIPTION

Usefull utility type functions

Index