DumPy

Release 0.1.0_a1

Aurora Pereira

Dec 13, 2023

`dumpy.fft`(*x: Vec_Complex*) → Vec_Complex

Radix-2 DIT FFT using list comprehension.

Parameters
x (*Vec_Complex*) – The input vector of real or complex numbers. Must have a length that is a power of 2.

Returns
The output vector after performing the FFT.

Return type
Vec_Complex

Raises
ValueError – The length of the input vector is not a power of 2.

`dumpy.identity`(*n: int*) → Matrix

Returns an identity matrix of size n x n.

Parameters
n (*int*) – The size of the identity matrix.

Returns
The identity matrix of size n x n.

Return type
Matrix

`dumpy.inner`(*v1: Vector*, *v2: Vector*) → Scalar

Computes the inner product of two vectors.

Parameters

- **v1** (*Vector*) – The first vector.
- **v2** (*Vector*) – The second vector.

Returns
The inner product of the two vectors.

Return type
Scalar

`dumpy.matadd`(*A: Matrix*, *B: Matrix*) → Matrix

Adds two matrices.

Parameters

- **A** (*Matrix*) – The first matrix.
- **B** (*Matrix*) – The second matrix.

Returns
The resulting matrix after adding A and B.

Return type
Matrix

`dumpy.matmul`(*A: Matrix*, *B: Matrix*, *mt: bool = True*, *flip: bool = True*) → Matrix

Performs a matrix multiplication on two matrices.

Parameters

- **A** (*Matrix*) – The first matrix.

- **B** (*Matrix*) – The second matrix.
- **mt** (*Bool, optional*) – Flag indicating whether to use multithreaded implementation. Defaults to True.
- **flip** (*Bool, optional*) – Flag indicating whether to transpose the second matrix. Defaults to True. Not available for multithreaded implementation.

Returns
The result of the matrix multiplication.

Return type
Matrix

dumpy.**matsub**(*A: Matrix*, *B: Matrix*) → Matrix

Subtracts two matrices.

Parameters

- **A** (*Matrix*) – The first matrix.
- **B** (*Matrix*) – The second matrix.

Returns
The resulting matrix after subtracting B from A.

Return type
Matrix

dumpy.**mvmul**(*A: Vector*, *B: Vector*) → Vector

Matrix-vector/vector-matrix multiplication.

Parameters

- **A** (*Vector or Matrix*) – The matrix represented as a list of lists.
- **B** (*Vector or Matrix*) – The vector represented as a list.

Returns
A vector represented as a list. For Matrix-Vector Multiplication: A 1-column vector represented as a list of lists.

Return type
For Vector-Matrix Multiplication

Raises
ValueError – If the matrix and vector are not of compatible sizes.

dumpy.**norm**(*v: Vector*, *p: int = 2*) → Scalar

Computes the p-norm of a vector.

Parameters

- **v** (*Vector*) – The input vector.
- **p** (*int, optional*) – The order of the norm. Default is 2 (Euclidean norm).

Returns
The p-norm of the vector.

Return type
Scalar

`dumpy.outer`(*v1: Vector*, *v2: Vector*) → Vector

Computes the outer product of two vectors.

Parameters

- **v1** (*list*) – The first vector.
- **v2** (*list*) – The second vector.

Returns
The outer product matrix.

Return type
Vector

Raises
`ValueError` – If the vectors are not of the same length.

`dumpy.printmat`(*A: Matrix*, *digits: int = 3*) → None

Prints a matrix.

Parameters

- **A** (*Matrix*) – The matrix to be printed.
- **digits** (*int, optional*) – The number of decimal places to round the vector elements to. Default is 3.

Returns
None

`dumpy.printvec`(*v: Vector*, *digits: int = 3*) → None

Prints a vector.

Parameters

- **v** (*Vector*) – The vector to be printed.
- **digits** (*int, optional*) – The number of decimal places to round the vector elements to. Default is 3.

Returns
None

`dumpy.randmat`(*r: int*, *c: int*, *lb: Scalar = 0*, *ub: Scalar = 100*, *dtype: str = 'float'*) → Matrix

Returns a random matrix of size r x c.

Please don’t use this for random ints, esecially for small ranges. Use a builtin like random.randint() instead.

Parameters

- **r** (*int*) – The number of rows of the output matrix.
- **c** (*int*) – The number of columns of the output matrix.
- **lb** (*Scalar, optional*) – The lower bound of the random values. Defaults to 0.
- **ub** (*Scalar, optional*) – The upper bound of the random values. Defaults to 100.
- **dtype** (*str, optional*) – The data type of the random values. Can be ‘int’ or ‘float’. Defaults to ‘float’.

Returns
The random matrix.

Return type
Matrix

Raises
`ValueError` – If the dtype is not ‘int’ or ‘float’.

`dumpy.`**`randrng`**`(`*lb: Scalar = 0*, *ub: Scalar = 1*, *dtype: str = 'float'*`)` → Scalar

Returns a random scalar within a specified range.

Please don’t use this for random ints, esecially for small ranges. Use a builtin like random.randint() instead.

Parameters

- **`lb`** (*`Scalar, optional`*) – The lower bound of the range (inclusive). Default is 0.
- **`ub`** (*`Scalar, optional`*) – The upper bound of the range (inclusive). Default is 1.
- **`dtype`** (*`Scalar, optional`*) – The data type of the returned scalar. Must be ‘int’ or ‘float’. Default is ‘float’.

Returns
A random scalar within the specified range.

Return type
Scalar

Raises
`ValueError` – If the dtype is not ‘int’ or ‘double’.

`dumpy.`**`randvec`**`(`*n: int*, *lb: Scalar = 0*, *ub: Scalar = 100*, *dtype: str = 'float'*`)` → Vector

Returns a random vector of size n.

Please don’t use this for random ints, esecially for small ranges. Use a builtin like random.randint() instead.

Parameters

- **`n`** (*`int`*) – The size of the vector.
- **`lb`** (*`Scalar, optional`*) – The lower bound of the random values. Defaults to 0.
- **`ub`** (*`Scalar, optional`*) – The upper bound of the random values. Defaults to 100.
- **`dtype`** (*`str, optional`*) – The data type of the random values. Can be ‘int’ or ‘float’. Defaults to ‘float’.

Returns
The random vector.

Return type
Vector

Raises
`ValueError` – If the dtype is not ‘int’ or ‘float’.

`dumpy.`**`taylor_cos`**`(`*theta: float*, *iter: int = 64*`)` → float

Calculate the cosine of an angle using Taylor series approximation.

Parameters

- **`theta`** (*`float or int`*) – The angle in radians.
- **`iter`** (*`int`*) – The number of iterations to perform in the Taylor series approximation. Default is 64.

Returns
The cosine of the angle.

Return type
float or int

`dumpy.`**`taylor_sin`**(*theta: float*, *iter: int = 64*) → float

Calculate the sine of an angle using Taylor series approximation.

Parameters

- **`theta`** (*`float or int`*) – The angle in radians.
- **`iter`** (*`int`*) – The number of iterations to perform in the Taylor series approximation. Default is 64.

Returns
The sine of the angle.

Return type
float or int

`dumpy.`**`transpose`**(*A: Matrix*) → Matrix

Transposes a matrix.

Parameters
`A` (*`Matrix`*) – The matrix to be transposed.

Returns
The transposed matrix.

Return type
Matrix