matrix_decomposition Documentation

Release 0.1

Joscha Reimer

Jul 19, 2018

Contents:

1 Functions **1**
- 1.1 decompose a matrix . . . 1
- 1.2 approximate a matrix by a decomposition . . . 2
- 1.3 examine a matrix . . . 3
- 1.4 solve system of linear equations . . . 4

2 Matrix decompositions **5**
- 2.1 LL decomposition . . . 5
- 2.2 LDL decomposition . . . 8
- 2.3 LDL decomposition compressed . . . 12
- 2.4 base decomposition . . . 16

3 Errors **21**
- 3.1 MatrixNoDecompositionPossibleError . . . 21
- 3.2 MatrixNoLDLDecompositionPossibleError . . . 21
- 3.3 MatrixNoLLDecompositionPossibleError . . . 22
- 3.4 MatrixNoDecompositionPossibleWithProblematicSubdecompositionError . . . 22
- 3.5 MatrixDecompositionNoConversionImplementedError . . . 22
- 3.6 MatrixNotSquareError . . . 22
- 3.7 MatrixNotFiniteError . . . 23
- 3.8 MatrixSingularError . . . 23
- 3.9 MatrixDecompositionNotFiniteError . . . 23
- 3.10 MatrixDecompositionSingularError . . . 23
- 3.11 MatrixError . . . 23

4 Changelog **25**
- 4.1 v0.7 . . . 25
- 4.2 v0.6 . . . 25
- 4.3 v0.5 . . . 25
- 4.4 v0.4 . . . 25
- 4.5 v0.3 . . . 25
- 4.6 v0.2 . . . 26
- 4.7 v0.1 . . . 26

5 Indices and tables **27**

CHAPTER 1

Functions

Several functions are included in this package. The most important are summarized here.

1.1 decompose a matrix

matrix.**decompose**(*A*, *permutation_method=None*, *check_finite=True*, *return_type=None*)

Computes a decomposition of a matrix.

Parameters

- **A** (*numpy.ndarray or scipy.sparse.spmatrix*) – Matrix to be decomposed. It is assumed, that A is Hermitian. The matrix must be a squared matrix.
- **permutation_method** (*str*) – The symmetric permutation method that is applied to the matrix before it is decomposed. It has to be a value in *matrix.PERMUTATION_METHODS*. If *A* is sparse, it can also be a value in *matrix.SPARSE_PERMUTATION_METHODS*. optional, default: no permutation
- **check_finite** (*bool*) – Whether to check that the input matrix contains only finite numbers. Disabling may result in problems (crashes, non-termination) if the inputs do contain infinities or NaNs. Disabling gives a performance gain. optional, default: True
- **return_type** (*str*) – The type of the decomposition that should be calculated. It has to be a value in *matrix.DECOMPOSITION_TYPES*. If return_type is None the type of the returned decomposition is chosen by the function itself. optional, default: the type of the decomposition is chosen by the function itself

Returns A decompostion of *A* of type *return_type*.

Return type *matrix.decompositions.DecompositionBase*

Raises

- *matrix.errors.MatrixNoDecompositionPossibleError* – If the decomposition of *A* is not possible.

- *matrix.errors.MatrixNotSquareError* – If *A* is not a square matrix.
- *matrix.errors.MatrixNotFiniteError* – If *A* is not a finte matrix and *check_finite* is True.

matrix.**PERMUTATION_METHODS = (None, '', 'none', 'natural', 'decreasing_diagonal_values', '**
Supported permutation methods for dense and sparse matrices.

matrix.**SPARSE_PERMUTATION_METHODS = ()**
Supported permutation methods only for sparse matrices.

matrix.**DECOMPOSITION_TYPES = ('LDL', 'LDL_compressed', 'LL')**
Supported types of decompositions.

1.2 approximate a matrix by a decomposition

matrix.**approximate**(*A*, *t=None*, *min_diag_value=None*, *max_diag_value=None*, *min_abs_value=None*, *permutation_method=None*, *check_finite=True*, *return_type=None*, *callback=None*)
Computes an approximative decomposition of a matrix.

If *A* is decomposable in a decomposition of type *return_type*, this decomposition is returned. Otherwise a decomposition of type *return_type* is retuned which represents an approximation of *A*.

Parameters

- **A** (*numpy.ndarray or scipy.sparse.spmatrix*) – The matrix that should be approximated by a decomposition. It is assumed, that A is Hermitian. The matrix must be a squared matrix.
- **t** (*numpy.ndarray*) – The targed vector used for the approximation. For each i in range(M) *min_diag_value* <= *t[i]* <= *max_diag_value* must hold. *t* and *A* must have the same length. optional, default : The diagonal of *A* is used as *t*.
- **min_diag_value** (*float*) – Each component of the diagonal of the matrix *D* in an returned *LDL* decomposition is forced to be greater or equal to *min_diag_value*. optional, default : 0.
- **max_diag_value** (*float*) – Each component of the diagonal of the matrix *D* in an returned *LDL* decomposition is forced to be lower or equal to *max_diag_value*. optional, default : No maximal value is forced.
- **min_abs_value** (*float*) – Absolute values below *min_abs_value* are considered as zero. optional, default : The resolution of the underlying data type is used.
- **permutation_method** (*str*) – The symmetric permutation method that is applied to the matrix before it is decomposed. It has to be a value in *matrix.PERMUTATION_METHODS*. If *A* is sparse, it can also be a value in *matrix.SPARSE_PERMUTATION_METHODS*. optional, default: No permutation is done.
- **check_finite** (*bool*) – Whether to check that the input matrix contains only finite numbers. Disabling may result in problems (crashes, non-termination) if the inputs do contain infinities or NaNs. Disabling gives a performance gain. optional, default: True
- **return_type** (*str*) – The type of the decomposition that should be calculated. It has to be a value in *matrix.DECOMPOSITION_TYPES*. optional, default : The type of the decomposition is chosen by the function itself.

- **callback** (*callable*) – In each iteration *callback(i, r)* is called where *i* is the index of the row and column where components of *A* are reduced by the factor *r*. optional, default : No callback function is called.

Returns An approximative decompostion of *A* of type *return_type*.

Return type *matrix.decompositions.DecompositionBase*

Raises

- *matrix.errors.MatrixNotSquareError* – If *A* is not a square matrix.
- *matrix.errors.MatrixNotFiniteError* – If *A* is not a finte matrix and *check_finite* is True.

1.3 examine a matrix

matrix.**is_invertible**(*A*, *check_finite=True*)

Returns whether the passed matrix is an invertible matrix.

Parameters

- **A** (*numpy.ndarray or scipy.sparse.spmatrix*) – The matrix that should be checked. It is assumed, that A is Hermitian. The matrix must be a squared matrix.
- **check_finite** (*bool*) – Whether to check that *A* contain only finite numbers. Disabling may result in problems (crashes, non-termination) if they contain infinities or NaNs. Disabling gives a performance gain. optional, default: True

Returns Whether *A* is invertible.

Return type bool

Raises *matrix.errors.MatrixNotFiniteError* – If *A* is not a finte matrix and *check_finite* is True.

matrix.**is_positive_semi_definite**(*A*, *check_finite=True*)

Returns whether the passed matrix is positive semi-definite.

Parameters

- **A** (*numpy.ndarray or scipy.sparse.spmatrix*) – The matrix that should be checked. It is assumed, that A is Hermitian. The matrix must be a squared matrix.
- **check_finite** (*bool*) – Whether to check that *A* contain only finite numbers. Disabling may result in problems (crashes, non-termination) if they contain infinities or NaNs. Disabling gives a performance gain. optional, default: True

Returns Whether *A* is positive semi-definite.

Return type bool

Raises *matrix.errors.MatrixNotFiniteError* – If *A* is not a finte matrix and *check_finite* is True.

matrix.**is_positive_definite**(*A*, *check_finite=True*)

Returns whether the passed matrix is positive definite.

Parameters

- **A** (*numpy.ndarray or scipy.sparse.spmatrix*) – The matrix that should be checked. It is assumed, that A is Hermitian. The matrix must be a squared matrix.

- **check_finite** (*bool*) – Whether to check that *A* contain only finite numbers. Disabling may result in problems (crashes, non-termination) if they contain infinities or NaNs. Disabling gives a performance gain. optional, default: True

Returns Whether *A* is positive definite.

Return type bool

Raises *matrix.errors.MatrixNotFiniteError* – If *A* is not a finte matrix and *check_finite* is True.

1.4 solve system of linear equations

matrix.**solve**(*A*, *b*, *overwrite_b=False*, *check_finite=True*)

Solves the equation *A x = b* regarding *x*.

Parameters

- **A** (*numpy.ndarray or scipy.sparse.spmatrix*) – The matrix that should be checked. It is assumed, that A is Hermitian. The matrix must be a squared matrix.
- **b** (*numpy.ndarray*) – Right-hand side vector or matrix in equation *A x = b*. Ii must hold *b.shape[0] == A.shape[0]*.
- **overwrite_b** (*bool*) – Allow overwriting data in *b*. Enabling gives a performance gain. optional, default: False
- **check_finite** (*bool*) – Whether to check that *A* and b‘ contain only finite numbers. Disabling may result in problems (crashes, non-termination) if they contain infinities or NaNs. Disabling gives a performance gain. optional, default: True

Returns An *x* so that *A x = b*. The shape of *x* matches the shape of *b*.

Return type numpy.ndarray

Raises

- *matrix.errors.MatrixNotSquareError* – If *A* is not a square matrix.
- *matrix.errors.MatrixNotFiniteError* – If *A* is not a finte matrix and *check_finite* is True.
- *matrix.errors.MatrixSingularError* – If *A* is singular.

CHAPTER 2

Matrix decompositions

Several matrix decompositions are supported. They are available in *matrix.decompositions*:

2.1 LL decomposition

class matrix.decompositions.**LL_Decomposition**(*L=None*, *p=None*)

Bases: *matrix.decompositions.DecompositionBase*

A matrix decomposition where LL^H is the decomposed (permuted) matrix.

L is a lower triangle matrix with ones on the diagonal. This decomposition is also called Cholesky decomposition.

Parameters

- **L** (*numpy.ndarray or scipy.sparse.spmatrix*) – The matrix *L* of the decomposition. optional, If it is not set yet, it must be set later.
- **p** (*numpy.ndarray*) – The permutation vector used for the decomposition. This decomposition is of A[p[:, np.newaxis], p[np.newaxis, :]] where A is a matrix. optional, default: no permutation

L

numpy.matrix or scipy.sparse.spmatrix – The matrix *L* of the decomposition.

P

scipy.sparse.dok_matrix – The permutation matrix. P @ A @ P.H is the matrix A permuted by the permutation of the decomposition

check_finite(*check_finite=True*)

Check if this is a decomposition representing a finite matrix.

Parameters **check_finite** (*bool*) – Whether to perform this check. default: True

Raises *matrix.errors.MatrixDecompositionNotFiniteError* – If this is a decomposition representing a non-finite matrix.

check_invertible()
Check if this is a decomposition representing an invertible matrix.

Raises *matrix.errors.MatrixDecompositionSingularError* – If this is a decomposition representing a singular matrix.

composed_matrix
numpy.matrix or scipy.sparse.spmatrix – The composed matrix represented by this decomposition.

copy()
Copy this decomposition.

Returns A copy of this decomposition.

Return type *matrix.decompositions.DecompositionBase*

decomposition_type
str – The type of this decompositon.

is_finite()
Returns whether this is a decomposition representing a finite matrix.

Returns Whether this is a decomposition representing a finite matrix.

Return type bool

is_invertible()
Returns whether this is a decomposition representing an invertible matrix.

Returns Whether this is a decomposition representing an invertible matrix.

Return type bool

is_permuted
bool – Whether this is a decompositon with permutation.

is_positive_definite()
Returns whether this is a decomposition of a positive definite matrix.

Returns Whether this is a decomposition of a positive definite matrix.

Return type bool

is_positive_semi_definite()
Returns whether this is a decomposition of a positive semi-definite matrix.

Returns Whether this is a decomposition of a positive semi-definite matrix.

Return type bool

is_singular()
Returns whether this is a decomposition representing a singular matrix.

Returns Whether this is a decomposition representing a singular matrix.

Return type bool

is_sparse()
Returns whether this is a decomposition of a sparse matrix.

Returns Whether this is a decomposition of a sparse matrix.

Return type bool

is_type(*decomposition_type*)
Whether this is a decomposition of the passed type.

Parameters **decomposition_type** (*str*) – The decomposition type according to which is checked.

Returns Whether this is a decomposition of the passed type.

Return type bool

load(*directory_name*, *filename_prefix=None*)
Loads a decomposition of this type.

Parameters

- **directory_name** (*str*) – A directory where this decomposition is saved.
- **filename_prefix** (*str*) – A prefix for the filenames of the attributes of this decomposition.

Raises FileNotFoundError – If the files are not found in the passed directory.

n
int – The dimension of the squared decomposed matrix.

p
numpy.ndarray – The permutation vector. A[p[:, np.newaxis], p[np.newaxis, :]] is the matrix A permuted by the permutation of the decomposition

p_inverse
numpy.ndarray – The permutation vector that undos the permutation.

permute_matrix(*A*)
Permute a matrix by the permutation of the decomposition.

Parameters **A** (*numpy.ndarray or scipy.sparse.spmatrix*) – The matrix that should be permuted.

Returns The matrix *A* permuted by the permutation of the decomposition.

Return type numpy.ndarray or scipy.sparse.spmatrix

save(*directory_name*, *filename_prefix=None*)
Saves this decomposition.

Parameters

- **directory_name** (*str*) – A directory where this decomposition should be saved.
- **filename_prefix** (*str*) – A prefix for the filenames of the attributes of this decomposition.

solve(*b*, *overwrite_b=False*, *check_finite=True*)
Solves the equation *A x = b* regarding *x*, where *A* is the composed matrix represented by this decomposition.

Parameters

- **b** (*numpy.ndarray*) – Right-hand side vector or matrix in equation *A x = b*. Ii must hold *b.shape[0] == self.n*.
- **overwrite_b** (*bool*) – Allow overwriting data in *b*. Enabling gives a performance gain. optional, default: False
- **check_finite** (*bool*) – Whether to check that the this decomposition and b‘ contain only finite numbers. Disabling may result in problems (crashes, non-termination) if they contain infinities or NaNs. Disabling gives a performance gain. optional, default: True

Returns An *x* so that *A x = b*. The shape of *x* matches the shape of *b*.

Return type numpy.ndarray

Raises

- *matrix.errors.MatrixDecompositionSingularError* – If this is a decomposition representing a singular matrix.
- *matrix.errors.MatrixDecompositionNotFiniteError* – If this is a decomposition representing a non-finite matrix and *check_finite* is True.

to(*decomposition_type*, *copy=False*)
Convert decomposition to passed type.

Parameters

- **decomposition_type** (*str*) – The decomposition type to which this decomposition is converted.
- **copy** (*bool*) – Whether the data of this decomposition should always be copied or only if needed.

Returns If the type of this decomposition is not *decomposition_type*, a decompostion of type *decomposition_type* is returned which represents the same decomposed matrix as this decomposition. Otherwise this decomposition or a copy of it is returned, depending on *copy*.

Return type *matrix.decompositions.DecompositionBase*

to_LDL_Decomposition()

to_any(**decomposition_types*, *copy=False*)
Convert decomposition to any of the passed types.

Parameters

- ***decomposition_types** (*str*) – The decomposition types to any of them this this decomposition is converted.
- **copy** (*bool*) – Whether the data of this decomposition should always be copied or only if needed.

Returns If the type of this decomposition is not in *decomposition_types*, a decompostion of type *decomposition_type[0]* is returned which represents the same decomposed matrix as this decomposition. Otherwise this decomposition or a copy of it is returned, depending on *copy*.

Return type *matrix.decompositions.DecompositionBase*

unpermute_matrix(*A*)
Unpermute a matrix permuted by the permutation of the decomposition.

Parameters **A** (*numpy.ndarray or scipy.sparse.spmatrix*) – The matrix that should be unpermuted.

Returns The matrix *A* unpermuted by the permutation of the decomposition.

Return type numpy.ndarray or scipy.sparse.spmatrix

2.2 LDL decomposition

class matrix.decompositions.**LDL_Decomposition**(*L=None*, *d=None*, *p=None*)
Bases: *matrix.decompositions.DecompositionBase*

A matrix decomposition where LDL^H is the decomposed (permuted) matrix.

L is a lower triangle matrix with ones on the diagonal. *D* is a diagonal matrix. Only the diagonal values of *D* are stored.

Parameters

- **L** (*numpy.ndarray or scipy.sparse.spmatrix*) – The matrix *L* of the decomposition. optional, If it is not set yet, it must be set later.
- **d** (*numpy.ndarray*) – The vector of the diagonal components of *D* of the decompositon. optional, If it is not set yet, it must be set later.
- **p** (*numpy.ndarray*) – The permutation vector used for the decomposition. This decomposition is of A[p[:, np.newaxis], p[np.newaxis, :]] where A is a matrix. optional, default: no permutation

D
scipy.sparse.dia_matrix – The permutation matrix.

L
numpy.matrix or scipy.sparse.spmatrix – The matrix *L* of the decomposition.

LD
numpy.matrix or scipy.sparse.spmatrix – A matrix whose diagonal values are the diagonal values of *D* and whose off-diagonal values are those of *L*.

P
scipy.sparse.dok_matrix – The permutation matrix. P @ A @ P.H is the matrix A permuted by the permutation of the decomposition

check_finite(*check_finite=True*)
Check if this is a decomposition representing a finite matrix.

Parameters **check_finite** (*bool*) – Whether to perform this check. default: True

Raises *matrix.errors.MatrixDecompositionNotFiniteError* – If this is a decomposition representing a non-finite matrix.

check_invertible()
Check if this is a decomposition representing an invertible matrix.

Raises *matrix.errors.MatrixDecompositionSingularError* – If this is a decomposition representing a singular matrix.

composed_matrix
numpy.matrix or scipy.sparse.spmatrix – The composed matrix represented by this decomposition.

copy()
Copy this decomposition.

Returns A copy of this decomposition.

Return type *matrix.decompositions.DecompositionBase*

d
numpy.ndarray – The diagonal vector of the matrix *D* of the decomposition.

decomposition_type
str – The type of this decompositon.

is_finite()
Returns whether this is a decomposition representing a finite matrix.

Returns Whether this is a decomposition representing a finite matrix.

Return type bool

is_invertible()
Returns whether this is a decomposition representing an invertible matrix.

Returns Whether this is a decomposition representing an invertible matrix.

Return type bool

is_permuted
bool – Whether this is a decompositon with permutation.

is_positive_definite()
Returns whether this is a decomposition of a positive definite matrix.

Returns Whether this is a decomposition of a positive definite matrix.

Return type bool

is_positive_semi_definite()
Returns whether this is a decomposition of a positive semi-definite matrix.

Returns Whether this is a decomposition of a positive semi-definite matrix.

Return type bool

is_singular()
Returns whether this is a decomposition representing a singular matrix.

Returns Whether this is a decomposition representing a singular matrix.

Return type bool

is_sparse()
Returns whether this is a decomposition of a sparse matrix.

Returns Whether this is a decomposition of a sparse matrix.

Return type bool

is_type(*decomposition_type*)
Whether this is a decomposition of the passed type.

Parameters **decomposition_type** (*str*) – The decomposition type according to which is checked.

Returns Whether this is a decomposition of the passed type.

Return type bool

load(*directory_name*, *filename_prefix=None*)
Loads a decomposition of this type.

Parameters

- **directory_name** (*str*) – A directory where this decomposition is saved.
- **filename_prefix** (*str*) – A prefix for the filenames of the attributes of this decomposition.

Raises FileNotFoundError – If the files are not found in the passed directory.

n
int – The dimension of the squared decomposed matrix.

p
numpy.ndarray – The permutation vector. A[p[:, np.newaxis], p[np.newaxis, :]] is the matrix A permuted by the permutation of the decomposition

p_inverse
numpy.ndarray – The permutation vector that undos the permutation.

permute_matrix(*A*)
Permute a matrix by the permutation of the decomposition.

Parameters **A** (*numpy.ndarray or scipy.sparse.spmatrix*) – The matrix that should be permuted.

Returns The matrix *A* permuted by the permutation of the decomposition.

Return type numpy.ndarray or scipy.sparse.spmatrix

save(*directory_name*, *filename_prefix=None*)
Saves this decomposition.

Parameters

- **directory_name** (*str*) – A directory where this decomposition should be saved.
- **filename_prefix** (*str*) – A prefix for the filenames of the attributes of this decomposition.

solve(*b*, *overwrite_b=False*, *check_finite=True*)
Solves the equation *A x = b* regarding *x*, where *A* is the composed matrix represented by this decomposition.

Parameters

- **b** (*numpy.ndarray*) – Right-hand side vector or matrix in equation *A x = b*. Ii must hold *b.shape[0] == self.n*.
- **overwrite_b** (*bool*) – Allow overwriting data in *b*. Enabling gives a performance gain. optional, default: False
- **check_finite** (*bool*) – Whether to check that the this decomposition and b‘ contain only finite numbers. Disabling may result in problems (crashes, non-termination) if they contain infinities or NaNs. Disabling gives a performance gain. optional, default: True

Returns An *x* so that *A x = b*. The shape of *x* matches the shape of *b*.

Return type numpy.ndarray

Raises

- *matrix.errors.MatrixDecompositionSingularError* – If this is a decomposition representing a singular matrix.
- *matrix.errors.MatrixDecompositionNotFiniteError* – If this is a decomposition representing a non-finite matrix and *check_finite* is True.

to(*decomposition_type*, *copy=False*)
Convert decomposition to passed type.

Parameters

- **decomposition_type** (*str*) – The decomposition type to which this decomposition is converted.
- **copy** (*bool*) – Whether the data of this decomposition should always be copied or only if needed.

Returns If the type of this decomposition is not *decomposition_type*, a decompostion of type *decomposition_type* is returned which represents the same decomposed matrix as this decomposition. Otherwise this decomposition or a copy of it is returned, depending on *copy*.

Return type *matrix.decompositions.DecompositionBase*

to_LDL_DecompositionCompressed()

to_LL_Decomposition()

to_any(**decomposition_types*, *copy=False*)
Convert decomposition to any of the passed types.

Parameters

- ***decomposition_types** (*str*) – The decomposition types to any of them this this decomposition is converted.
- **copy** (*bool*) – Whether the data of this decomposition should always be copied or only if needed.

Returns If the type of this decomposition is not in *decomposition_types*, a decompostion of type *decomposition_type[0]* is returned which represents the same decomposed matrix as this decomposition. Otherwise this decomposition or a copy of it is returned, depending on *copy*.

Return type *matrix.decompositions.DecompositionBase*

unpermute_matrix(*A*)
Unpermute a matrix permuted by the permutation of the decomposition.

Parameters **A** (*numpy.ndarray or scipy.sparse.spmatrix*) – The matrix that should be unpermuted.

Returns The matrix *A* unpermuted by the permutation of the decomposition.

Return type numpy.ndarray or scipy.sparse.spmatrix

2.3 LDL decomposition compressed

class matrix.decompositions.**LDL_DecompositionCompressed**(*LD=None*, *p=None*)
Bases: *matrix.decompositions.DecompositionBase*

A matrix decomposition where LDL^H is the decomposed (permuted) matrix.

L is a lower triangle matrix with ones on the diagonal. *D* is a diagonal matrix. *L* and *D* are stored in one matrix whose diagonal values are the diagonal values of *D* and whose off-diagonal values are those of *L*.

Parameters

- **LD** (*numpy.ndarray or scipy.sparse.spmatrix*) – A matrix whose diagonal values are the diagonal values of *D* and whose off-diagonal values are those of *L*. optional, If it is not set yet, it must be set later.
- **p** (*numpy.ndarray*) – The permutation vector used for the decomposition. This decomposition is of A[p[:, np.newaxis], p[np.newaxis, :]] where A is a matrix. optional, default: no permutation

D
scipy.sparse.dia_matrix – The permutation matrix.

L
numpy.matrix or scipy.sparse.spmatrix – The matrix *L* of the decomposition.

LD
numpy.matrix or scipy.sparse.spmatrix – A matrix whose diagonal values are the diagonal values of *D* and whose off-diagonal values are those of *L*.

P
scipy.sparse.dok_matrix – The permutation matrix. P @ A @ P.H is the matrix A permuted by the permutation of the decomposition

check_finite(*check_finite=True*)
Check if this is a decomposition representing a finite matrix.

Parameters **check_finite** (*bool*) – Whether to perform this check. default: True

Raises *matrix.errors.MatrixDecompositionNotFiniteError* – If this is a decomposition representing a non-finite matrix.

check_invertible()
Check if this is a decomposition representing an invertible matrix.

Raises *matrix.errors.MatrixDecompositionSingularError* – If this is a decomposition representing a singular matrix.

composed_matrix
numpy.matrix or scipy.sparse.spmatrix – The composed matrix represented by this decomposition.

copy()
Copy this decomposition.

Returns A copy of this decomposition.

Return type *matrix.decompositions.DecompositionBase*

d
numpy.ndarray – The diagonal vector of the matrix *D* of the decomposition.

decomposition_type
str – The type of this decompositon.

is_finite()
Returns whether this is a decomposition representing a finite matrix.

Returns Whether this is a decomposition representing a finite matrix.

Return type bool

is_invertible()
Returns whether this is a decomposition representing an invertible matrix.

Returns Whether this is a decomposition representing an invertible matrix.

Return type bool

is_permuted
bool – Whether this is a decompositon with permutation.

is_positive_definite()
Returns whether this is a decomposition of a positive definite matrix.

Returns Whether this is a decomposition of a positive definite matrix.

Return type bool

is_positive_semi_definite()
Returns whether this is a decomposition of a positive semi-definite matrix.

Returns Whether this is a decomposition of a positive semi-definite matrix.

Return type bool

is_singular()
Returns whether this is a decomposition representing a singular matrix.

Returns Whether this is a decomposition representing a singular matrix.

Return type bool

is_sparse()
Returns whether this is a decomposition of a sparse matrix.

Returns Whether this is a decomposition of a sparse matrix.

Return type bool

is_type(*decomposition_type*)
Whether this is a decomposition of the passed type.

Parameters **decomposition_type** (*str*) – The decomposition type according to which is checked.

Returns Whether this is a decomposition of the passed type.

Return type bool

load(*directory_name*, *filename_prefix=None*)
Loads a decomposition of this type.

Parameters

- **directory_name** (*str*) – A directory where this decomposition is saved.
- **filename_prefix** (*str*) – A prefix for the filenames of the attributes of this decomposition.

Raises FileNotFoundError – If the files are not found in the passed directory.

n
int – The dimension of the squared decomposed matrix.

p
numpy.ndarray – The permutation vector. A[p[:, np.newaxis], p[np.newaxis, :]] is the matrix A permuted by the permutation of the decomposition

p_inverse
numpy.ndarray – The permutation vector that undos the permutation.

permute_matrix(*A*)
Permute a matrix by the permutation of the decomposition.

Parameters **A** (*numpy.ndarray or scipy.sparse.spmatrix*) – The matrix that should be permuted.

Returns The matrix *A* permuted by the permutation of the decomposition.

Return type numpy.ndarray or scipy.sparse.spmatrix

save(*directory_name*, *filename_prefix=None*)
Saves this decomposition.

Parameters

- **directory_name** (*str*) – A directory where this decomposition should be saved.
- **filename_prefix** (*str*) – A prefix for the filenames of the attributes of this decomposition.

solve(*b*, *overwrite_b=False*, *check_finite=True*)

Solves the equation $A\ x = b$ regarding x, where A is the composed matrix represented by this decomposition.

Parameters

- **b** (*numpy.ndarray*) – Right-hand side vector or matrix in equation $A\ x = b$. Ii must hold *b.shape[0] == self.n*.
- **overwrite_b** (*bool*) – Allow overwriting data in *b*. Enabling gives a performance gain. optional, default: False
- **check_finite** (*bool*) – Whether to check that the this decomposition and b‘ contain only finite numbers. Disabling may result in problems (crashes, non-termination) if they contain infinities or NaNs. Disabling gives a performance gain. optional, default: True

Returns An x so that $A\ x = b$. The shape of x matches the shape of b.

Return type numpy.ndarray

Raises

- *matrix.errors.MatrixDecompositionSingularError* – If this is a decomposition representing a singular matrix.
- *matrix.errors.MatrixDecompositionNotFiniteError* – If this is a decomposition representing a non-finite matrix and *check_finite* is True.

to(*decomposition_type*, *copy=False*)

Convert decomposition to passed type.

Parameters

- **decomposition_type** (*str*) – The decomposition type to which this decomposition is converted.
- **copy** (*bool*) – Whether the data of this decomposition should always be copied or only if needed.

Returns If the type of this decomposition is not *decomposition_type*, a decompostion of type *decomposition_type* is returned which represents the same decomposed matrix as this decomposition. Otherwise this decomposition or a copy of it is returned, depending on *copy*.

Return type *matrix.decompositions.DecompositionBase*

to_LDL_Decomposition()

to_any(**decomposition_types*, *copy=False*)

Convert decomposition to any of the passed types.

Parameters

- ***decomposition_types** (*str*) – The decomposition types to any of them this this decomposition is converted.
- **copy** (*bool*) – Whether the data of this decomposition should always be copied or only if needed.

Returns If the type of this decomposition is not in *decomposition_types*, a decompostion of type *decomposition_type[0]* is returned which represents the same decomposed matrix as

this decomposition. Otherwise this decomposition or a copy of it is returned, depending on *copy*.

Return type *matrix.decompositions.DecompositionBase*

unpermute_matrix(*A*)
Unpermute a matrix permuted by the permutation of the decomposition.

Parameters **A** (*numpy.ndarray or scipy.sparse.spmatrix*) – The matrix that should be unpermuted.

Returns The matrix *A* unpermuted by the permutation of the decomposition.

Return type numpy.ndarray or scipy.sparse.spmatrix

2.4 base decomposition

class matrix.decompositions.**DecompositionBase**(*p=None*)
Bases: object

A matrix decomposition.

This class is a base class for matrix decompositions.

Parameters **p** (*numpy.ndarray*) – The permutation vector used for the decomposition. This decomposition is of A[p[:, np.newaxis], p[np.newaxis, :]] where A is a matrix. optional, default: no permutation

P
scipy.sparse.dok_matrix – The permutation matrix. P @ A @ P.H is the matrix A permuted by the permutation of the decomposition

check_finite(*check_finite=True*)
Check if this is a decomposition representing a finite matrix.

Parameters **check_finite** (*bool*) – Whether to perform this check. default: True

Raises *matrix.errors.MatrixDecompositionNotFiniteError* – If this is a decomposition representing a non-finite matrix.

check_invertible()
Check if this is a decomposition representing an invertible matrix.

Raises *matrix.errors.MatrixDecompositionSingularError* – If this is a decomposition representing a singular matrix.

composed_matrix
numpy.matrix or scipy.sparse.spmatrix – The composed matrix represented by this decomposition.

copy()
Copy this decomposition.

Returns A copy of this decomposition.

Return type *matrix.decompositions.DecompositionBase*

decomposition_type
str – The type of this decompositon.

is_finite()
Returns whether this is a decomposition representing a finite matrix.

Returns Whether this is a decomposition representing a finite matrix.

Return type bool

is_invertible()
Returns whether this is a decomposition representing an invertible matrix.

Returns Whether this is a decomposition representing an invertible matrix.

Return type bool

is_permuted
bool – Whether this is a decompositon with permutation.

is_positive_definite()
Returns whether this is a decomposition of a positive definite matrix.

Returns Whether this is a decomposition of a positive definite matrix.

Return type bool

is_positive_semi_definite()
Returns whether this is a decomposition of a positive semi-definite matrix.

Returns Whether this is a decomposition of a positive semi-definite matrix.

Return type bool

is_singular()
Returns whether this is a decomposition representing a singular matrix.

Returns Whether this is a decomposition representing a singular matrix.

Return type bool

is_sparse()
Returns whether this is a decomposition of a sparse matrix.

Returns Whether this is a decomposition of a sparse matrix.

Return type bool

is_type(*decomposition_type*)
Whether this is a decomposition of the passed type.

Parameters **decomposition_type** (*str*) – The decomposition type according to which is checked.

Returns Whether this is a decomposition of the passed type.

Return type bool

load(*directory_name*, *filename_prefix=None*)
Loads a decomposition of this type.

Parameters

- **directory_name** (*str*) – A directory where this decomposition is saved.
- **filename_prefix** (*str*) – A prefix for the filenames of the attributes of this decomposition.

Raises FileNotFoundError – If the files are not found in the passed directory.

n
int – The dimension of the squared decomposed matrix.

p
numpy.ndarray – The permutation vector. A[p[:, np.newaxis], p[np.newaxis, :]] is the matrix A permuted by the permutation of the decomposition

p_inverse
numpy.ndarray – The permutation vector that undos the permutation.

permute_matrix(*A*)
Permute a matrix by the permutation of the decomposition.

Parameters **A** (*numpy.ndarray or scipy.sparse.spmatrix*) – The matrix that should be permuted.

Returns The matrix *A* permuted by the permutation of the decomposition.

Return type numpy.ndarray or scipy.sparse.spmatrix

save(*directory_name*, *filename_prefix=None*)
Saves this decomposition.

Parameters

- **directory_name** (*str*) – A directory where this decomposition should be saved.
- **filename_prefix** (*str*) – A prefix for the filenames of the attributes of this decomposition.

solve(*b*, *overwrite_b=False*, *check_finite=True*)
Solves the equation *A x = b* regarding *x*, where *A* is the composed matrix represented by this decomposition.

Parameters

- **b** (*numpy.ndarray*) – Right-hand side vector or matrix in equation *A x = b*. Ii must hold *b.shape[0] == self.n*.
- **overwrite_b** (*bool*) – Allow overwriting data in *b*. Enabling gives a performance gain. optional, default: False
- **check_finite** (*bool*) – Whether to check that the this decomposition and b‘ contain only finite numbers. Disabling may result in problems (crashes, non-termination) if they contain infinities or NaNs. Disabling gives a performance gain. optional, default: True

Returns An *x* so that *A x = b*. The shape of *x* matches the shape of *b*.

Return type numpy.ndarray

Raises

- *matrix.errors.MatrixDecompositionSingularError* – If this is a decomposition representing a singular matrix.
- *matrix.errors.MatrixDecompositionNotFiniteError* – If this is a decomposition representing a non-finite matrix and *check_finite* is True.

to(*decomposition_type*, *copy=False*)
Convert decomposition to passed type.

Parameters

- **decomposition_type** (*str*) – The decomposition type to which this decomposition is converted.
- **copy** (*bool*) – Whether the data of this decomposition should always be copied or only if needed.

Returns If the type of this decomposition is not *decomposition_type*, a decompostion of type *decomposition_type* is returned which represents the same decomposed matrix as this decomposition. Otherwise this decomposition or a copy of it is returned, depending on *copy*.

Return type *matrix.decompositions.DecompositionBase*

to_any(**decomposition_types*, *copy=False*)

Convert decomposition to any of the passed types.

Parameters

- ***decomposition_types** (*str*) – The decomposition types to any of them this this decomposition is converted.
- **copy** (*bool*) – Whether the data of this decomposition should always be copied or only if needed.

Returns If the type of this decomposition is not in *decomposition_types*, a decompostion of type *decomposition_type[0]* is returned which represents the same decomposed matrix as this decomposition. Otherwise this decomposition or a copy of it is returned, depending on *copy*.

Return type *matrix.decompositions.DecompositionBase*

unpermute_matrix(*A*)

Unpermute a matrix permuted by the permutation of the decomposition.

Parameters **A** (*numpy.ndarray or scipy.sparse.spmatrix*) – The matrix that should be unpermuted.

Returns The matrix *A* unpermuted by the permutation of the decomposition.

Return type numpy.ndarray or scipy.sparse.spmatrix

CHAPTER 3

Errors

This is an overview about the exceptions that could arise in this package. They are available in *matrix.errors*:

The following exceptions can be raised if a matrix should be decomposed with *matrix.decompose* and the desired decomposition is not computable.

3.1 MatrixNoDecompositionPossibleError

class matrix.errors.**MatrixNoDecompositionPossibleError**(*matrix=None*, *decomposition_decription=None*, *message=None*)

Bases: *matrix.errors.MatrixError*

The matrix decomposition is not possible for this matrix.

3.2 MatrixNoLDLDecompositionPossibleError

class matrix.errors.**MatrixNoLDLDecompositionPossibleError**(*matrix=None*, *problematic_leading_principal_submatrix_index=None*, *subdecomposition=None*)

Bases: *matrix.errors.MatrixNoDecompositionPossibleWithProblematicSubdecompositionError*

A LDL decomposition is not possible for this matrix.

3.3 MatrixNoLLDecompositionPossibleError

class matrix.errors.**MatrixNoLLDecompositionPossibleError**(*matrix=None*, *problematic_leading_principal_submatrix_index=None*, *subdecomposition=None*)

Bases: *matrix.errors.MatrixNoDecompositionPossibleWithProblematicSubdecompositionError*

A LL decomposition is not possible for this matrix.

3.4 MatrixNoDecompositionPossibleWithProblematicSubdecompositionError

class matrix.errors.**MatrixNoDecompositionPossibleWithProblematicSubdecompositionError**(*matrix decomposition_d problematic_le subdecomposition=N*

Bases: *matrix.errors.MatrixNoDecompositionPossibleError*

The desired matrix decomposition is not possible for this matrix. Only a subdecompostion could be calculated

3.5 MatrixDecompositionNoConversionImplementedError

class matrix.errors.**MatrixDecompositionNoConversionImplementedError**(*original_decomposition=None*, *desired_decomposition_type=None*

Bases: *matrix.errors.MatrixError*

A decomposition conversion is not implemented for this type.

The following exceptions can occur if a matrix has an invalid characteristic.

3.6 MatrixNotSquareError

class matrix.errors.**MatrixNotSquareError**(*matrix=None*)

Bases: *matrix.errors.MatrixError*

A matrix is not a square matrix although this is required.

3.7 MatrixNotFiniteError

class matrix.errors.**MatrixNotFiniteError**(*matrix=None*)
Bases: *matrix.errors.MatrixError*

A matrix has non-finite entries although a finite matrix is required.

3.8 MatrixSingularError

class matrix.errors.**MatrixSingularError**(*matrix=None*)
Bases: *matrix.errors.MatrixError*

A matrix is singular although an invertible matrix is required.

The following exceptions can occur if the matrix represented by a decomposition has an invalid characteristic.

3.9 MatrixDecompositionNotFiniteError

class matrix.errors.**MatrixDecompositionNotFiniteError**(*decomposition=None*)
Bases: *matrix.errors.MatrixError*

A decomposition of a matrix has non-finite entries although a finite matrix is required.

3.10 MatrixDecompositionSingularError

class matrix.errors.**MatrixDecompositionSingularError**(*decomposition=None*)
Bases: *matrix.errors.MatrixError*

A decomposition represents a singular matrix although a non-singular matrix is required.

The following exception is the base exception from which all other exceptions in this package are derived.

3.11 MatrixError

class matrix.errors.**MatrixError**(*matrix=None*, *message=None*)
Bases: Exception

An exception related to a matrix.

This is the base exception for all exceptions in this package.

CHAPTER 4

Changelog

4.1 v0.7

- Lineare systems associated to matrices or decompositions can now be solved.
- Invertibility of matrices and decompositions can now be examined.
- Decompositions can now be examined to see if they contain only finite values.

4.2 v0.6

- Decompositions are now saveable and loadable.

4.3 v0.5

- Matrices can now be approximated by decompositions.

4.4 v0.4

- Positive definiteness and positive semi-definiteness of matrices and decompositions can now be examined.

4.5 v0.3

- Dense and sparse matrices are now decomposable into several types (LL, LDL, LDL compressed).

4.6 v0.2

- Decompositons are now convertable to other decompositon types.
- Decompositions are now comparable.

4.7 v0.1

- Several decompositions types are added (LL, LDL, LDL compressed).
- Several permutation capabilities added.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Index

A

approximate() (in module matrix), 2

C

check_finite() (matrix.decompositions.DecompositionBase method), 16
check_finite() (matrix.decompositions.LDL_Decomposition method), 9
check_finite() (matrix.decompositions.LDL_DecompositionCompressed method), 13
check_finite() (matrix.decompositions.LL_Decomposition method), 5
check_invertible() (matrix.decompositions.DecompositionBase method), 16
check_invertible() (matrix.decompositions.LDL_Decomposition method), 9
check_invertible() (matrix.decompositions.LDL_DecompositionCompressed method), 13
check_invertible() (matrix.decompositions.LL_Decomposition method), 5
composed_matrix (matrix.decompositions.DecompositionBase attribute), 16
composed_matrix (matrix.decompositions.LDL_Decomposition attribute), 9
composed_matrix (matrix.decompositions.LDL_DecompositionCompressed attribute), 13
composed_matrix (matrix.decompositions.LL_Decomposition attribute), 6
copy() (matrix.decompositions.DecompositionBase method), 16
copy() (matrix.decompositions.LDL_Decomposition method), 9
copy() (matrix.decompositions.LDL_DecompositionCompressed method), 13
copy() (matrix.decompositions.LL_Decomposition method), 6

D

D (matrix.decompositions.LDL_Decomposition attribute), 9
d (matrix.decompositions.LDL_Decomposition attribute), 9
D (matrix.decompositions.LDL_DecompositionCompressed attribute), 12
d (matrix.decompositions.LDL_DecompositionCompressed attribute), 13
decompose() (in module matrix), 1
decomposition_type (matrix.decompositions.DecompositionBase attribute), 16
decomposition_type (matrix.decompositions.LDL_Decomposition attribute), 9
decomposition_type (matrix.decompositions.LDL_DecompositionCompressed attribute), 13
decomposition_type (matrix.decompositions.LL_Decomposition attribute), 6
DECOMPOSITION_TYPES (in module matrix), 2
DecompositionBase (class in matrix.decompositions), 16

I

is_finite() (matrix.decompositions.DecompositionBase method), 16
is_finite() (matrix.decompositions.LDL_Decomposition method), 9
is_finite() (matrix.decompositions.LDL_DecompositionCompressed method), 13
is_finite() (matrix.decompositions.LL_Decomposition method), 6

is_invertible() (in module matrix), 3
is_invertible() (matrix.decompositions.DecompositionBase method), 17
is_invertible() (matrix.decompositions.LDL_Decomposition method), 10
is_invertible() (matrix.decompositions.LDL_DecompositionCompressed method), 13
is_invertible() (matrix.decompositions.LL_Decomposition method), 6
is_permuted (matrix.decompositions.DecompositionBase attribute), 17
is_permuted (matrix.decompositions.LDL_Decomposition attribute), 10
is_permuted (matrix.decompositions.LDL_DecompositionCompressed attribute), 13
is_permuted (matrix.decompositions.LL_Decomposition attribute), 6
is_positive_definite() (in module matrix), 3
is_positive_definite() (matrix.decompositions.DecompositionBase method), 17
is_positive_definite() (matrix.decompositions.LDL_Decomposition method), 10
is_positive_definite() (matrix.decompositions.LDL_DecompositionCompressed method), 13
is_positive_definite() (matrix.decompositions.LL_Decomposition method), 6
is_positive_semi_definite() (in module matrix), 3
is_positive_semi_definite() (matrix.decompositions.DecompositionBase method), 17
is_positive_semi_definite() (matrix.decompositions.LDL_Decomposition method), 10
is_positive_semi_definite() (matrix.decompositions.LDL_DecompositionCompressed method), 13
is_positive_semi_definite() (matrix.decompositions.LL_Decomposition method), 6
is_singular() (matrix.decompositions.DecompositionBase method), 17
is_singular() (matrix.decompositions.LDL_Decomposition method), 10
is_singular() (matrix.decompositions.LDL_DecompositionCompressed method), 14
is_singular() (matrix.decompositions.LL_Decomposition method), 6
is_sparse() (matrix.decompositions.DecompositionBase method), 17
is_sparse() (matrix.decompositions.LDL_Decomposition method), 10
is_sparse() (matrix.decompositions.LDL_DecompositionCompressed method), 14
is_sparse() (matrix.decompositions.LL_Decomposition method), 6
is_type() (matrix.decompositions.DecompositionBase method), 17
is_type() (matrix.decompositions.LDL_Decomposition method), 10
is_type() (matrix.decompositions.LDL_DecompositionCompressed method), 14
is_type() (matrix.decompositions.LL_Decomposition method), 6

L

L (matrix.decompositions.LDL_Decomposition attribute), 9
L (matrix.decompositions.LDL_DecompositionCompressed attribute), 12
L (matrix.decompositions.LL_Decomposition attribute), 5
LD (matrix.decompositions.LDL_Decomposition attribute), 9
LD (matrix.decompositions.LDL_DecompositionCompressed attribute), 13
LDL_Decomposition (class in matrix.decompositions), 8
LDL_DecompositionCompressed (class in matrix.decompositions), 12
LL_Decomposition (class in matrix.decompositions), 5
load() (matrix.decompositions.DecompositionBase method), 17
load() (matrix.decompositions.LDL_Decomposition method), 10
load() (matrix.decompositions.LDL_DecompositionCompressed method), 14
load() (matrix.decompositions.LL_Decomposition method), 7

M

MatrixDecompositionNoConversionImplementedError (class in matrix.errors), 22
MatrixDecompositionNotFiniteError (class in matrix.errors), 23
MatrixDecompositionSingularError (class in matrix.errors), 23
MatrixError (class in matrix.errors), 23
MatrixNoDecompositionPossibleError (class in matrix.errors), 21
MatrixNoDecompositionPossibleWithProblematicSubdecompositionError (class in matrix.errors), 22
MatrixNoLDLDecompositionPossibleError (class in matrix.errors), 21
MatrixNoLLDecompositionPossibleError (class in matrix.errors), 22

MatrixNotFiniteError (class in matrix.errors), 23
MatrixNotSquareError (class in matrix.errors), 22
MatrixSingularError (class in matrix.errors), 23

N

n (matrix.decompositions.DecompositionBase attribute), 17
n (matrix.decompositions.LDL_Decomposition attribute), 10
n (matrix.decompositions.LDL_DecompositionCompressed attribute), 14
n (matrix.decompositions.LL_Decomposition attribute), 7

P

P (matrix.decompositions.DecompositionBase attribute), 16
p (matrix.decompositions.DecompositionBase attribute), 17
P (matrix.decompositions.LDL_Decomposition attribute), 9
p (matrix.decompositions.LDL_Decomposition attribute), 10
P (matrix.decompositions.LDL_DecompositionCompressed attribute), 13
p (matrix.decompositions.LDL_DecompositionCompressed attribute), 14
P (matrix.decompositions.LL_Decomposition attribute), 5
p (matrix.decompositions.LL_Decomposition attribute), 7
p_inverse (matrix.decompositions.DecompositionBase attribute), 18
p_inverse (matrix.decompositions.LDL_Decomposition attribute), 11
p_inverse (matrix.decompositions.LDL_DecompositionCompressed attribute), 14
p_inverse (matrix.decompositions.LL_Decomposition attribute), 7
PERMUTATION_METHODS (in module matrix), 2
permute_matrix() (matrix.decompositions.DecompositionBase method), 18
permute_matrix() (matrix.decompositions.LDL_Decomposition method), 11
permute_matrix() (matrix.decompositions.LDL_DecompositionCompressed method), 14
permute_matrix() (matrix.decompositions.LL_Decomposition method), 7

S

save() (matrix.decompositions.DecompositionBase method), 18
save() (matrix.decompositions.LDL_Decomposition method), 11
save() (matrix.decompositions.LDL_DecompositionCompressed method), 14
save() (matrix.decompositions.LL_Decomposition method), 7
solve() (in module matrix), 4
solve() (matrix.decompositions.DecompositionBase method), 18
solve() (matrix.decompositions.LDL_Decomposition method), 11
solve() (matrix.decompositions.LDL_DecompositionCompressed method), 15
solve() (matrix.decompositions.LL_Decomposition method), 7
SPARSE_PERMUTATION_METHODS (in module matrix), 2

T

to() (matrix.decompositions.DecompositionBase method), 18
to() (matrix.decompositions.LDL_Decomposition method), 11
to() (matrix.decompositions.LDL_DecompositionCompressed method), 15
to() (matrix.decompositions.LL_Decomposition method), 8
to_any() (matrix.decompositions.DecompositionBase method), 19
to_any() (matrix.decompositions.LDL_Decomposition method), 12
to_any() (matrix.decompositions.LDL_DecompositionCompressed method), 15
to_any() (matrix.decompositions.LL_Decomposition method), 8
to_LDL_Decomposition() (matrix.decompositions.LDL_DecompositionCompressed method), 15
to_LDL_Decomposition() (matrix.decompositions.LL_Decomposition method), 8
to_LDL_DecompositionCompressed() (matrix.decompositions.LDL_Decomposition method), 12
to_LL_Decomposition() (matrix.decompositions.LDL_Decomposition method), 12

U

unpermute_matrix() (matrix.decompositions.DecompositionBase method), 19
unpermute_matrix() (matrix.decompositions.LDL_Decomposition method), 12

unpermute_matrix() (matrix.decompositions.LDL_DecompositionCompressed method), 16
unpermute_matrix() (matrix.decompositions.LL_Decomposition method), 8