Diffusion Maps

Generated by Doxygen 1.8.17

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 diffusion_maps Namespace Reference

Namespace for everything related to diffusion maps.

Namespaces

- kernel

  Diffusion kernels.

Classes

- class Matrix

  Matrix of doubles that may or may not own its data.
- class SparseMatrix

  Sparse matrix in the CSR format.
- class Vector

  Vector.

Functions

- template<typename R >
  Matrix diffusion_maps (const Matrix &data, std::size_t n_components, const std::function< double(const Vector &, const Vector &)> &kernel, double diffusion_time, R &rng, double kernel_↩
  epsilon=DEFAULT_KERNEL_EPSILON, double eig_solver_tol=DEFAULT_EIG_SOLVER_TOL, unsigned eig_solver_max_iter=DEFAULT_EIG_SOLVER_MAX_ITER)

  Diffusion maps.

Variables

- constexpr double DEFAULT_KERNEL_EPSILON = 1e-6

  Default kernel epsilon for the diffusion_maps() function.
- constexpr double DEFAULT_EIG_SOLVER_TOL = 1e-6

  Default tolerance of the eigendecomposition solver for the diffusion_maps() function.
- constexpr unsigned DEFAULT_EIG_SOLVER_MAX_ITER = 100000

  Default maximum number of iterations of the eigendecomposition solver for the diffusion_maps() function.

4.1.1 Detailed Description

Namespace for everything related to diffusion maps.

4.1.2 Function Documentation

4.1.2.1 diffusion_maps()

```
template<typename R >
Matrix diffusion_maps::diffusion_maps (
            const Matrix & data,
            std::size_t n_components,
            const std::function< double(const Vector &, const Vector &)> & kernel,
            double diffusion_time,
            R & rng,
            double kernel_epsilon = DEFAULT_KERNEL_EPSILON,
            double eig_solver_tol = DEFAULT_EIG_SOLVER_TOL,
            unsigned eig_solver_max_iter = DEFAULT_EIG_SOLVER_MAX_ITER )
```

Diffusion maps.

Template Parameters

R	The type of the random number generator.

Parameters

in	*data*	The data matrix where each row is a data point.
in	*n_components*	The dimension of the projected subspace.
in	*kernel*	The kernel function.
in	*diffusion_time*	The diffusion time.
in,out	*rng*	The random number generator.
in	*kernel_epsilon*	The value below which the output of the kernel would be treated as zero.
in	*eig_solver_tol*	The tolerance of the eigendecomposition solver.
in	*eig_solver_max_iter*	The maximum number of iterations of the eigendecomposition solver.

Returns

The lower-dimensional embedding of the data in the diffusion space.

Exceptions

std::invalid_argument	If `n_components` is greater than the number of data points minus 1.
std::invalid_argument	If `diffusion_time` is negative.

4.2 diffusion_maps::kernel Namespace Reference

Diffusion kernels.

Classes

- class Gaussian

  Gaussian kernel.

4.2.1 Detailed Description

Diffusion kernels.

Chapter 5

Class Documentation

5.1 diffusion_maps::kernel::Gaussian Class Reference

Gaussian kernel.

`#include <diffusion_maps/kernel.hpp>`

Public Member Functions

- Gaussian (const double gamma)

  Constructs a Gaussian kernel with the given parameter γ.
- double operator() (const Vector &x, const Vector &y) const

  Evaluates the kernel function.

Static Public Member Functions

- static Gaussian with_gamma (const double gamma)

  Constructs a Gaussian kernel with the given parameter γ.
- static Gaussian with_sigma (const double sigma)

  Constructs a Gaussian kernel with the given σ.

Public Attributes

- double gamma

  Kernel parameter γ = 1 / 2σ^2.

5.1.1 Detailed Description

Gaussian kernel.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Gaussian()

```
diffusion_maps::kernel::Gaussian::Gaussian (
            const double gamma )  [inline]
```

Constructs a Gaussian kernel with the given parameter γ.

Parameters

in	*gamma*	Kernel parameter γ = 1 / $2\sigma^2$.

5.1.3 Member Function Documentation

5.1.3.1 with_gamma()

```
static Gaussian diffusion_maps::kernel::Gaussian::with_gamma (
            const double gamma )  [inline], [static]
```

Constructs a Gaussian kernel with the given parameter γ.

Parameters

in	*gamma*	Kernel parameter γ = 1 / $2\sigma^2$.

5.1.3.2 with_sigma()

```
static Gaussian diffusion_maps::kernel::Gaussian::with_sigma (
            const double sigma )  [inline], [static]
```

Constructs a Gaussian kernel with the given σ.

Parameters

in	*sigma*	Kernel parameter σ.

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

- include/diffusion_maps/kernel.hpp

5.2 diffusion_maps::Matrix Class Reference

Matrix of doubles that may or may not own its data.

```
#include <diffusion_maps/matrix.hpp>
```

Public Member Functions

- Matrix (const std::size_t n_rows, const std::size_t n_cols)

  Constructs an owning matrix of the given dimensions with uninitialized elements.
- Matrix (double *const data, const std::size_t n_rows, const std::size_t n_cols, const std::size_t row_stride, const std::size_t col_stride)

  Constructs a non-owning matrix.
- double * data ()

  The data array.
- const double * data () const

  The data array.
- std::size_t n_rows () const

  The number of rows.
- std::size_t n_cols () const

  The number of columns.
- std::size_t row_stride () const

  The stride in the row dimension.
- std::size_t col_stride () const

  The stride in the column dimension.
- double & operator() (const std::size_t i, const std::size_t j)

  Returns the (`i` , `j`)-th element without bounds checking.
- double operator() (const std::size_t i, const std::size_t j) const

  Returns the (`i` , `j`)-th element without bounds checking.
- Vector row (const std::size_t i) const

  Returns the `i` -th row without bounds checking.

Protected Attributes

- std::variant< std::unique_ptr< double[ ]>, double * > _data

  The data array.
- std::size_t _n_rows

  The number of rows.
- std::size_t _n_cols

  The number of columns.
- std::size_t _row_stride

  The stride in the row dimension.
- std::size_t _col_stride

  The stride in the column dimension.

5.2.1 Detailed Description

Matrix of doubles that may or may not own its data.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 Matrix() [1/2]

```
diffusion_maps::Matrix::Matrix (
            const std::size_t n_rows,
            const std::size_t n_cols )  [inline]
```

Constructs an owning matrix of the given dimensions with uninitialized elements.

Parameters

in	*n_rows*	The number of rows.
in	*n_cols*	The number of columns.

5.2.2.2 Matrix() [2/2]

```
diffusion_maps::Matrix::Matrix (
            double *const data,
            const std::size_t n_rows,
            const std::size_t n_cols,
            const std::size_t row_stride,
            const std::size_t col_stride )  [inline]
```

Constructs a non-owning matrix.

Parameters

in	*data*	The data array.
in	*n_rows*	The number of rows.
in	*n_cols*	The number of columns.
in	*row_stride*	The stride in the row dimension.
in	*col_stride*	The stride in the column dimension.

5.2.3 Member Function Documentation

5.2.3.1 operator()() [1/2]

```
double& diffusion_maps::Matrix::operator() (
            const std::size_t i,
            const std::size_t j )  [inline]
```

Returns the (i , j)-th element without bounds checking.

Parameters

in	*i*	The row index.
in	*j*	The column index.

Returns

The (i , j)-th element.

5.2.3.2 operator()() [2/2]

```
double diffusion_maps::Matrix::operator() (
            const std::size_t i,
            const std::size_t j ) const  [inline]
```

Returns the (i , j)-th element without bounds checking.

Parameters

in	*i*	The row index.
in	*j*	The column index.

Returns

The (i , j)-th element.

5.2.3.3 row()

```
Vector diffusion_maps::Matrix::row (
            const std::size_t i ) const  [inline]
```

Returns the i -th row without bounds checking.

Parameters

in	*i*	The row index.

Returns

The i -th row.

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

- include/diffusion_maps/matrix.hpp

5.3 diffusion_maps::SparseMatrix Class Reference

Sparse matrix in the CSR format.

`#include <diffusion_maps/sparse_matrix.hpp>`

Classes

- class Triplet

  A non-zero element of a sparse matrix as a (i, j, value) triplet.

Public Member Functions

- SparseMatrix ()

  Constructs an empty 0×0 matrix.
- SparseMatrix (const std::size_t n_rows, const std::size_t n_cols, std::vector< Triplet > &triplets)

  Constructs a sparse matrix from a vector of triplets.
- SparseMatrix (const SparseMatrix &other)

  Copy constructor.
- SparseMatrix (SparseMatrix &&other) noexcept

  Move constructor. The moved-from matrix is set to an empty 0×0 matrix.
- virtual ~SparseMatrix ()=default

  Destructor.
- SparseMatrix & operator= (const SparseMatrix &other)

  Copy assignment operator.
- SparseMatrix & operator= (SparseMatrix &&other)

  Move assignment operator. The moved-from matrix is set to an empty 0×0 matrix.
- std::size_t n_rows () const

  The number of rows.
- std::size_t n_cols () const

  The number of columns.
- std::size_t n_nz () const

  The number of non-zero elements.
- double * data ()

  The data array.
- const double * data () const

  The data array.
- const std::size_t * col_ixs () const

  The column indices array.
- const std::size_t * row_ixs () const

  The row indices array.
- Vector operator* (const Vector &v) const

  Matrix-vector multiplication.

Protected Attributes

- std::size_t _n_rows

  The number of rows.
- std::size_t _n_cols

  The number of columns.
- std::unique_ptr< double[ ]> _data

  The data array of the matrix.
- std::unique_ptr< std::size_t[ ]> _col_ixs

  The column indices of each non-zero element.
- std::unique_ptr< std::size_t[ ]> _row_ixs

  The indices of each row.

5.3.1 Detailed Description

Sparse matrix in the CSR format.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 SparseMatrix() [1/3]

```
diffusion_maps::SparseMatrix::SparseMatrix (
            const std::size_t n_rows,
            const std::size_t n_cols,
            std::vector< Triplet > & triplets )  [inline]
```

Constructs a sparse matrix from a vector of triplets.

Parameters

in	*n_rows*	The number of rows.
in	*n_cols*	The number of columns.
in,out	*triplets*	The vector of triplets. Elements are sorted in-place.

5.3.2.2 SparseMatrix() [2/3]

```
diffusion_maps::SparseMatrix::SparseMatrix (
            const SparseMatrix & other )  [inline]
```

Copy constructor.

Parameters

in	*other*	The sparse matrix to copy.

5.3.2.3 SparseMatrix() [3/3]

```
diffusion_maps::SparseMatrix::SparseMatrix (
            SparseMatrix && other )  [inline], [noexcept]
```

Move constructor. The moved-from matrix is set to an empty 0×0 matrix.

Parameters

in,out	*other*	The sparse matrix to move.

5.3.3 Member Function Documentation

5.3.3.1 operator∗()

```
Vector diffusion_maps::SparseMatrix::operator* (
            const Vector & v ) const  [inline]
```

Matrix-vector multiplication.

Parameters

in	*v*	The vector to multiply.

Returns

The result of the multiplication.

Exceptions

std::invalid_argument	If the dimensions are incompatible.

5.3.3.2 operator=() [1/2]

```
SparseMatrix& diffusion_maps::SparseMatrix::operator= (
            const SparseMatrix & other )  [inline]
```

Copy assignment operator.

Parameters

in	*other*	The sparse matrix to copy.

Returns

A reference to this sparse matrix.

5.3.3.3 operator=() [2/2]

```
SparseMatrix& diffusion_maps::SparseMatrix::operator= (
            SparseMatrix && other )  [inline]
```

Move assignment operator. The moved-from matrix is set to an empty 0×0 matrix.

Parameters

in,out	*other*	The sparse matrix to move.

Returns

A reference to this sparse matrix.

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

- include/diffusion_maps/sparse_matrix.hpp

5.4 diffusion_maps::SparseMatrix::Triplet Class Reference

A non-zero element of a sparse matrix as a (i, j, value) triplet.

`#include <diffusion_maps/sparse_matrix.hpp>`

Public Member Functions

- bool operator< (const Triplet &other) const

  Less-than comparison operator. Compares the row and column indices.

Public Attributes

- std::size_t row

  The row index.
- std::size_t col

  The column index.
- double value

  The value.

5.4.1 Detailed Description

A non-zero element of a sparse matrix as a (i, j, value) triplet.

5.4.2 Member Function Documentation

5.4.2.1 operator<()

```
bool diffusion_maps::SparseMatrix::Triplet::operator< (
            const Triplet & other ) const  [inline]
```

Less-than comparison operator. Compares the row and column indices.

Parameters

in	*other*	The other triple.

Returns

True if this triple is less than the other triple in the aforementioned order.

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

- include/diffusion_maps/sparse_matrix.hpp

5.5 diffusion_maps::Vector Class Reference

Vector.

`#include <diffusion_maps/vector.hpp>`

Public Member Functions

- Vector ()
  Constructs a vector of size 0.
- Vector (const std::size_t size)
  Constructs a vector of size `size`.
- Vector (const std::size_t size, const double value)
  Constructs a vector of size `size` with each element set to `value`.
- Vector (const std::initializer_list< double > list)
  Constructs a vector from an initializer list.
- Vector (const Vector &other)
  Copy constructor.
- Vector (Vector &&other) noexcept
  Move constructor. The moved-from vector is set to a 0-sized vector.
- virtual ~Vector ()=default
  Destructor.
- Vector & operator= (const Vector &other)
  Copy assignment operator.
- Vector & operator= (Vector &&other)
  Move assignment operator. The moved-from vector is set to a 0-sized vector.
- std::size_t size () const noexcept
  The size of the vector.
- double * data () noexcept
  The data array of the vector.
- const double * data () const noexcept
  The data array of the vector.
- double & operator[ ] (const std::size_t index)
  Gets the element at index `index` without bounds checking.
- double operator[ ] (const std::size_t index) const
  Gets the element at index `index` without bounds checking.

- bool operator== (const Vector &other) const
  Equality operator.
- bool operator!= (const Vector &other) const
  Inequality operator.
- Vector operator- () const
  Negation operator.
- Vector operator+ (const Vector &other) const
  Addition operator.
- Vector & operator+= (const Vector &other)
  Addition assignment operator.
- Vector operator- (const Vector &other) const
  Subtraction operator.
- Vector & operator-= (const Vector &other)
  Subtraction assignment operator.
- Vector operator∗ (const double scalar) const
  Scalar multiplication operator.
- Vector & operator∗= (const double scalar)
  Scalar multiplication assignment operator.
- Vector operator/ (const double scalar) const
  Scalar division operator.
- Vector & operator/= (const double scalar)
  Scalar division assignment operator.
- double dot (const Vector &other) const
  Dot product.
- double sq_l2_norm () const
  The squared 2-norm (Euclidean norm) of the vector.
- double l2_norm () const
  The 2-norm (Euclidean norm) of the vector.
- Vector inv_sqrt () const
  Returns a vector where each element is the inverse square root of the corresponding element of this vector.

Protected Attributes

- std::size_t _size
  The size.
- std::unique_ptr< double[ ]> _data
  The data array.

5.5.1 Detailed Description

Vector.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 Vector() [1/5]

```
diffusion_maps::Vector::Vector (
            const std::size_t size )  [inline]
```

Constructs a vector of size `size`.

Parameters

in	*size*	The size of the vector.

5.5.2.2 Vector() [2/5]

```
diffusion_maps::Vector::Vector (
            const std::size_t size,
            const double value )  [inline]
```

Constructs a vector of size `size` with each element set to `value`.

Parameters

in	*size*	The size of the vector.
in	*value*	The value of each element.

5.5.2.3 Vector() [3/5]

```
diffusion_maps::Vector::Vector (
            const std::initializer_list< double > list )  [inline]
```

Constructs a vector from an initializer list.

Parameters

in	*list*	The initializer list.

5.5.2.4 Vector() [4/5]

```
diffusion_maps::Vector::Vector (
            const Vector & other )  [inline]
```

Copy constructor.

Parameters

in	*other*	The vector to copy.

5.5.2.5 Vector() [5/5]

```
diffusion_maps::Vector::Vector (
            Vector && other )  [inline], [noexcept]
```

Move constructor. The moved-from vector is set to a 0-sized vector.

Parameters

in,out	*other*	The vector to move.

5.5.3 Member Function Documentation

5.5.3.1 dot()

```
double diffusion_maps::Vector::dot (
            const Vector & other ) const  [inline]
```

Dot product.

Parameters

in	*other*	The vector to dot with.

Returns

The dot product of the vectors.

Exceptions

std::invalid_argument	If the vectors are not of the same size.

5.5.3.2 inv_sqrt()

```
Vector diffusion_maps::Vector::inv_sqrt ( ) const  [inline]
```

Returns a vector where each element is the inverse square root of the corresponding element of this vector.

Returns

The inverse square root of the vector.

5.5.3.3 operator"!=()

```
bool diffusion_maps::Vector::operator!= (
            const Vector & other ) const  [inline]
```

Inequality operator.

Parameters

in	*other*	The vector to compare with.

Returns

True if the vectors are not equal, false otherwise.

5.5.3.4 operator*()

```
Vector diffusion_maps::Vector::operator* (
            const double scalar ) const  [inline]
```

Scalar multiplication operator.

Parameters

in	*scalar*	The scalar to multiply by.

Returns

The scaled vector.

5.5.3.5 operator*=()

```
Vector& diffusion_maps::Vector::operator*= (
            const double scalar )  [inline]
```

Scalar multiplication assignment operator.

Parameters

in	*scalar*	The scalar to multiply by.

Returns

A reference to this vector.

5.5.3.6 operator+()

```
Vector diffusion_maps::Vector::operator+ (
            const Vector & other ) const  [inline]
```

Addition operator.

Parameters

in	*other*	The vector to add.

Returns

The sum of the vectors.

Exceptions

std::invalid_argument	If the vectors are not of the same size.

5.5.3.7 operator+=()

```
Vector& diffusion_maps::Vector::operator+= (
            const Vector & other )  [inline]
```

Addition assignment operator.

Parameters

in	*other*	The vector to add.

Returns

A reference to this vector.

Exceptions

std::invalid_argument	If the vectors are not of the same size.

5.5.3.8 operator-() [1/2]

```
Vector diffusion_maps::Vector::operator- ( ) const  [inline]
```

Negation operator.

Returns

The negation of the vector.

5.5.3.9 operator-() [2/2]

```
Vector diffusion_maps::Vector::operator- (
            const Vector & other ) const  [inline]
```

Subtraction operator.

Parameters

in	*other*	The vector to subtract.

Returns

The difference of the vectors.

Exceptions

std::invalid_argument	If the vectors are not of the same size.

5.5.3.10 operator-=()

```
Vector& diffusion_maps::Vector::operator-= (
            const Vector & other )  [inline]
```

Subtraction assignment operator.

Parameters

in	*other*	The vector to subtract.

Returns

A reference to this vector.

Exceptions

std::invalid_argument	If the vectors are not of the same size.

5.5.3.11 operator/()

```
Vector diffusion_maps::Vector::operator/ (
            const double scalar ) const  [inline]
```

Scalar division operator.

Parameters

in	*scalar*	The scalar to divide by.

Returns

The scaled vector.

5.5.3.12 operator/=()

```
Vector& diffusion_maps::Vector::operator/= (
            const double scalar )  [inline]
```

Scalar division assignment operator.

Parameters

in	*scalar*	The scalar to divide by.

Returns

A reference to this vector.

5.5.3.13 operator=() [1/2]

```
Vector& diffusion_maps::Vector::operator= (
            const Vector & other )  [inline]
```

Copy assignment operator.

Parameters

in	*other*	The vector to copy.

Returns

A reference to this vector.

5.5.3.14 operator=() [2/2]

```
Vector& diffusion_maps::Vector::operator= (
            Vector && other )  [inline]
```

Move assignment operator. The moved-from vector is set to a 0-sized vector.

Parameters

in,out	*other*	The vector to move.

Returns

A reference to this vector.

5.5.3.15 operator==()

```
bool diffusion_maps::Vector::operator== (
            const Vector & other ) const  [inline]
```

Equality operator.

Parameters

in	*other*	The vector to compare with.

Returns

True if the vectors are equal, false otherwise.

5.5.3.16 operator[]() [1/2]

```
double& diffusion_maps::Vector::operator[] (
            const std::size_t index )  [inline]
```

Gets the element at index `index` without bounds checking.

Parameters

in	*index*	The index of the element.

Returns

The element at index `index`.

5.5.3.17 operator[]() [2/2]

```
double diffusion_maps::Vector::operator[] (
            const std::size_t index ) const  [inline]
```

Gets the element at index `index` without bounds checking.

Parameters

in	*index*	The index of the element.

Returns

The element at index `index`.

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

- include/diffusion_maps/vector.hpp

Chapter 6

File Documentation

6.1 include/diffusion_maps/diffusion_maps.hpp File Reference

Diffusion maps algorithm.

```
#include <functional>
#include <memory>
#include <random>
#include "diffusion_maps/matrix.hpp"
#include "diffusion_maps/vector.hpp"
```

Namespaces

- diffusion_maps

  Namespace for everything related to diffusion maps.

Functions

- template<typename R >
  Matrix diffusion_maps::diffusion_maps (const Matrix &data, std::size_t n_components, const std::function< double(const Vector &, const Vector &)> &kernel, double diffusion_time, R &rng, double kernel_epsilon=DE↩ FAULT_KERNEL_EPSILON, double eig_solver_tol=DEFAULT_EIG_SOLVER_TOL, unsigned eig_solver↩ _max_iter=DEFAULT_EIG_SOLVER_MAX_ITER)

  Diffusion maps.

Variables

- constexpr double diffusion_maps::DEFAULT_KERNEL_EPSILON = 1e-6

  Default kernel epsilon for the diffusion_maps() function.
- constexpr double diffusion_maps::DEFAULT_EIG_SOLVER_TOL = 1e-6

  Default tolerance of the eigendecomposition solver for the diffusion_maps() function.
- constexpr unsigned diffusion_maps::DEFAULT_EIG_SOLVER_MAX_ITER = 100000

  Default maximum number of iterations of the eigendecomposition solver for the diffusion_maps() function.

6.1.1 Detailed Description

Diffusion maps algorithm.

6.2 include/diffusion_maps/kernel.hpp File Reference

Diffusion kernels.

```
#include "diffusion_maps/vector.hpp"
```

Classes

- class diffusion_maps::kernel::Gaussian

  Gaussian kernel.

Namespaces

- diffusion_maps

  Namespace for everything related to diffusion maps.
- diffusion_maps::kernel

  Diffusion kernels.

6.2.1 Detailed Description

Diffusion kernels.

6.3 include/diffusion_maps/matrix.hpp File Reference

Matrix.

```
#include <cstddef>
#include <memory>
#include <variant>
#include "diffusion_maps/internal/utils.hpp"
#include "diffusion_maps/vector.hpp"
```

Classes

- class diffusion_maps::Matrix

  Matrix of doubles that may or may not own its data.

Namespaces

- diffusion_maps

  Namespace for everything related to diffusion maps.

6.3.1 Detailed Description

Matrix.

6.4 include/diffusion_maps/sparse_matrix.hpp File Reference

Sparse matrix.

```
#include <algorithm>
#include <cstddef>
#include <memory>
#include <stdexcept>
#include <tuple>
#include <vector>
#include "diffusion_maps/vector.hpp"
```

Classes

- class diffusion_maps::SparseMatrix

  Sparse matrix in the CSR format.
- class diffusion_maps::SparseMatrix::Triplet

  A non-zero element of a sparse matrix as a (i, j, value) triplet.

Namespaces

- diffusion_maps

  Namespace for everything related to diffusion maps.

6.4.1 Detailed Description

Sparse matrix.

6.5 include/diffusion_maps/vector.hpp File Reference

Vector.

```
#include <algorithm>
#include <cmath>
#include <cstddef>
#include <initializer_list>
#include <memory>
#include <stdexcept>
```

Classes

- class diffusion_maps::Vector

  Vector.

Namespaces

- diffusion_maps

  Namespace for everything related to diffusion maps.

6.5.1 Detailed Description

Vector.

Index