EinSum

— Version 0.2 —

A C++ class library for index notation

Krister Ahlander

Contents

EinSum: Class support for Einsten Summation

EinSum is a class library that supports tensors with index notation, including Einstein summation. We use the following rules for index notation:

- All indices on the left hand side of an assignment are regarded as free.
- Repeated indices on the right hand side, at least once as upper and once as lower, are summation indices.

1 Indices

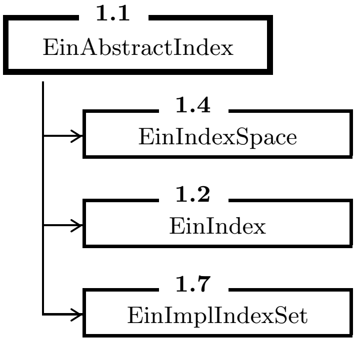
The index classes

Names

1.1 class **EinAbstractIndex**

EinAbstractIndex is an abstract base class for tensor indices or sets of tensor indices

Inheritance



Public Members

operator EinMultiInt () const
Conversion of current value to a multi int

virtual EinAbstractIndex* **virtualCopy** () const
Virtual constructor, clones the object.

virtual **~EinAbstractIndex** ()
Destructor

virtual int **containsCount** (const EinIndex&) const
counts numbers of indices with specified ID

virtual EinIndex **getIndex** (int slot) const
Returns the index in the specified slot.

virtual void **setIndex** (int slot, const EinIndex&)
Sets an index in the specified slot.

void **setIndices** (const EinAbstractIndex &M)
Sets all indices equal to those of M. Requires this and M to have the same n:o slots.

EinMultiIndex

		unrestricted () const	*Returns a multi index with no symmetries*
virtual	int	**fixed** () const	*True if the index involved in a loop.*
virtual	void	**begin** ()	*Shall be called when beginning a loop*
virtual	int	**end** () const	*Shall be true when exiting a loop*
virtual	void	**next** ()	*Iterate over the index values with next*
virtual	void	**operator++** ()	*Iterate over the index values with pre-increment*
virtual	void	**operator++** (int)	*Iterate over the index values with post-increment*
virtual	int	**current** () const	*returns the "current" value*
virtual	int	**dim** () const	*n:o total elements*
virtual	int	**size** () const	*n:o indep. elements, equals number of non-zero orbits*
virtual	int	**rank** () const	*rank, i.e., n:o slots*
virtual	ostream&	**display** (ostream&) const	*output on stream*
virtual	int	**currentSign** () const	*Returns -1, 0, or 1, depending on the current sign w.r.t. canonical element in orbit*
void		**newID** ()	*Gives all indices a new ID*
virtual	int	**currentOrbitSize** () const	*Returns n:o elements in orbit*
virtual	int	**isNonSymmetric** () const	*True for index spaces with no symmetries*
virtual	int	**isSymmetric** () const	*True for completely symmetric index spaces*
virtual	int	**isAntiSymmetric** () const	*True for completely anti-symmetric index spaces*
virtual	int	**orbit** (const EinMultiInt &m) const	*returns the orbit for given multi-int*
virtual	int	**sign** (const EinMultiInt &m) const	*Returns -1, 0, or 1, depending on the sign w.r.t. canonical element in orbit*
virtual	int	**orbitSize** (const EinMultiInt &m) const	*Returns n:o elements in orbit*

Protected Members

		EinAbstractIndex ()	*Constructor usable for heirs.*

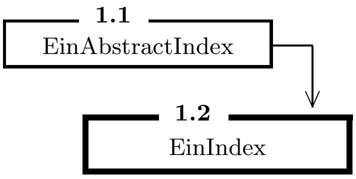
EinAbstractIndex is an abstract base class for tensor indices or sets of tensor indices. It also provides loop constructs for iterating over these ”index spaces”.

1.2

class **EinIndex** : public EinAbstractIndex

EinIndex is a one dimensional index and provides the normal way to interact with tensors

Inheritance



Public Members

	EinIndex (int dim = 3)	*Default yields a [0, 2] index.*
	EinIndex (int start, int stop)	
	EinIndex (const EinIndex&)	
	~EinIndex ()	
EinAbstractIndex*	**virtualCopy** () const	
void	**operator=** (const EinIndex&)	
int	**fixed** () const	
void	**begin** ()	
int	**end** () const	
void	**next** ()	
int	**current** () const	*Returns ”current” (storage) value*
int	**currentValue** () const	*Returns ”current” value*

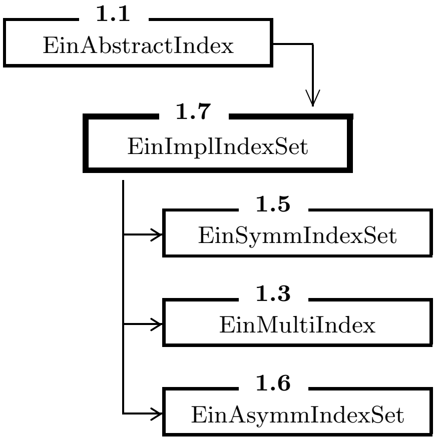
EinIndex is a one dimensional index and provides the normal way to interact with tensors. The identity of each object is used for deciding free indices and the summation indices. Note that a loop over an index which already is involved in a loop will be ignored. That is, current will not be affected by begin or next. To maintain correct loop status, it is important to break loops only when end is true.

1.7

class **EinImplIndexSet** : public EinAbstractIndex

EinImplIndexSet implements a set of abstract tensor indices, and is only used by inheritors

Inheritance

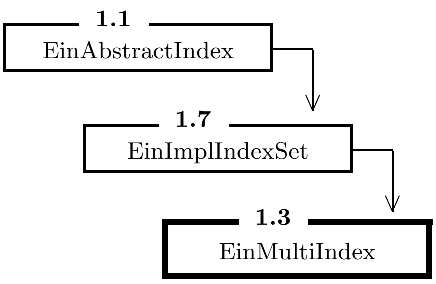


1.3

class **EinMultiIndex** : public EinImplIndexSet

EinMultiIndex is a set of abstract tensor indices

Inheritance



Public Members

friend EinMultiIndex
operator— (const EinAbstractIndex&,
const EinAbstractIndex&)

friend EinMultiIndex

operator— (const EinAbstractIndex&, const EinMultiIndex&)

friend EinMultiIndex **operator—** (const EinMultiIndex&, const EinAbstractIndex&)

friend EinMultiIndex **operator—** (const EinMultiIndex&, const EinMultiIndex&)

EinMultiIndex ()

EinMultiIndex (const EinIndex&)

EinMultiIndex (const EinMultiIndex&)

EinMultiIndex (int nr, int dim)
This constructor provides a multi-index with nr different indices of dimension dim

EinMultiIndex (const EinIndexSpace &is)
Creates an non symmetric version of is

~EinMultiIndex ()

EinAbstractIndex* **virtualCopy** () const

void **operator—=** (const EinAbstractIndex&)

void **operator—=** (const EinMultiIndex&)

void **operator=** (const EinMultiIndex&)

ostream& **display** (ostream&) const

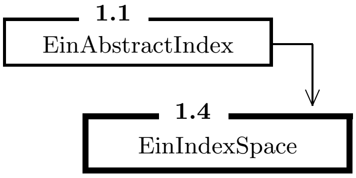
EinMultiIndex is a set of abstract tensor indices. It is normally constructed with operator bar (bitwise or)

1.4

class **EinIndexSpace** : public EinAbstractIndex

EinIndexSpace provides a concrete interface to index spaces

Inheritance



Public Members

	EinIndexSpace ()	*Default constructor yields an empty index*
	EinIndexSpace (const EinAbstractIndex &ai_)	*This constructor clones and keeps the abstract index.*
	EinIndexSpace (const EinIndexSpace &is)	*Copy constructor maintains reference counting.*
	~EinIndexSpace ()	*Destructor*
void	**operator=** (const EinAbstractIndex &ai_)	*Assignment clones and keeps the abstract index.*
void	**operator=** (const EinIndexSpace &is)	*This assignment maintains reference count.*
virtual EinAbstractIndex*	**virtualCopy** () const	*Virtual constructor, clones the object.*
virtual int	**containsCount** (const EinIndex &I) const	*counts numbers of indices with specified ID*
virtual EinIndex	**getIndex** (int slot) const	*Returns the index in the specified slot.*
virtual void	**setIndex** (int slot, const EinIndex &I)	*Sets an index in the specified slot.*
virtual int	**fixed** () const	*True if the index is set to a value or involved in a loop.*
virtual void	**begin** ()	*Shall be called when beginning a loop*
virtual int	**end** () const	*Shall be true when exiting a loop*
virtual void	**next** ()	*Iterate over the index values with next*
virtual void	**operator++** ()	*Iterate over the index values with pre-increment*
virtual void	**operator++** (int)	*Iterate over the index values with post-increment*
virtual int	**current** () const	*returns the current value for an index*
virtual int	**dim** () const	*n:o total elements*
virtual int	**size** () const	*n:o indep. elements, equals n:o orbits*
virtual int	**rank** () const	*n:o slots*
virtual ostream&	**display** (ostream &os) const	*output on stream*
virtual int	**currentSign** () const	*Returns -1, 0, or 1, depending on the current sign w.r.t. canonical element in orbit*

virtual int **currentOrbitSize** () const
Returns n:o elements in orbit

virtual int **isNonSymmetric** () const
True for index spaces with no symmetries

virtual int **isSymmetric** () const *True for completely symmetric index spaces*

virtual int **isAntiSymmetric** () const
True for completely anti-symmetric index spaces

int **orbit** (const EinMultiInt &m) const
returns the orbit for given multi-int

int **sign** (const EinMultiInt &m) const
Returns -1, 0, or 1, depending on the sign w.r.t. canonical element in orbit

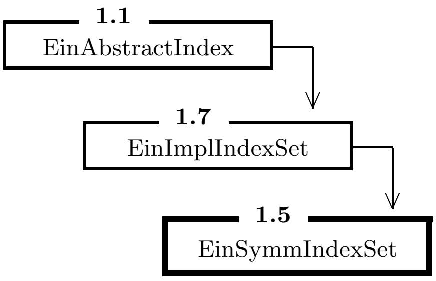
int **orbitSize** (const EinMultiInt &m) const
Returns n:o elements in orbit

1.5

class **EinSymmIndexSet** : public EinImplIndexSet

EinSymmIndexSet is a symmetric set of abstract tensor indices, and is normally constructed with ampers and operator (bitwise and)

Inheritance



Public Members

EinSymmIndexSet (int nr, int dim)
This constructor provides a completely symmetric index with nr different indices of dimension dim

EinSymmIndexSet (const EinMultiIndex &M)

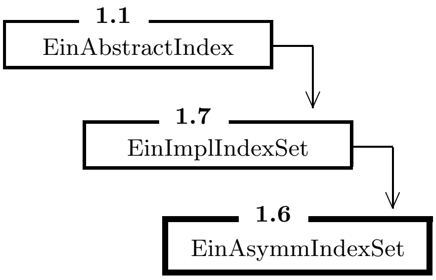
This constructor provides a completely symmetric index with same indices as M

1.6

class **EinAsymmIndexSet** : public EinImplIndexSet

EinAsymmIndexSet is a symmetric set of abstract tensor indices, and is normally constructed with wedge operator (bitwise XOR)

Inheritance



Public Members

EinAsymmIndexSet (int nr, int dim)
This constructor provides a completely anti-symmetric index with nr different indices of dimension dim

EinAsymmIndexSet (const EinMultiIndex &M)
This constructor provides a completely symmetric index with same indices as M

2 Tensors

The tensor classes

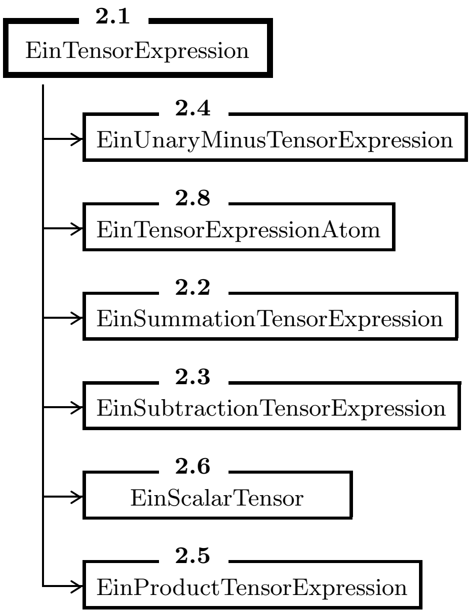
Names

2.1

template <class T> class **EinTensorExpression**

EinTensorExpression is a base class for tensor expressions, involving EinTensorExpressionAtoms, EinSummationTensorExpression, and EinProductTensorExpression objects

Inheritance



Public Members

		operator T () const
virtual		**~EinTensorExpression** ()
virtual	T	**eval** () const
virtual	void	**getMultiIndices** (EinMultiIndex &up, EinMultiIndex &lo) const
virtual	bool	**contains** (const EinTensor<T> *t) const

Protected Members

EinTensorExpression ()

EinTensorExpression (const EinTensorExpression<T>&)

EinTensorExpression is a base class for tensor expressions, involving EinTensorExpressionAtoms, EinSummationTensorExpression, and EinProductTensorExpression objects.

The eval(), getMultiIndices() and contains shall be implemented by inheritors.

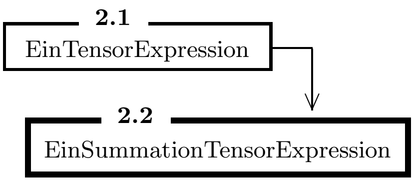
EinTensorExpression is parametrized over T. T is requested to have operators + and *.

2.2

template <class T> class **EinSummationTensorExpression** : public EinTensorExpression<T>

EinSummationTensorExpression is a binary operator that sums two TensorExpressions

Inheritance



Public Members

friend EinSummationTensorExpression <T>
operator+<T> (const EinTensorExpression<T> &left,
const EinTensorExpression<T> &right)

friend EinSummationTensorExpression <T>
operator+<T> (const T &left,
const EinTensorExpression<T> &right)

friend EinSummationTensorExpression <T>
operator+<T> (const EinTensorExpression<T> &left,
const T &right)

~EinSummationTensorExpression ()

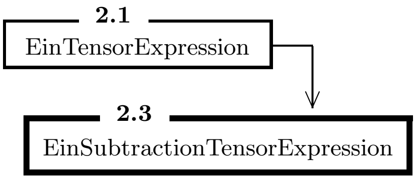
T **eval** () const

2.3

template <class T> class **EinSubtractionTensorExpression** : public EinTensorExpression<T>

EinSubtractionTensorExpression is a binary operator that subtracts two TensorExpressions

Inheritance



Public Members

friend EinSubtractionTensorExpression <T>
operator-<T> (const EinTensorExpression<T> &left,
const EinTensorExpression<T> &right)

friend EinSubtractionTensorExpression <T>
operator-<T> (const T &left,
const EinTensorExpression<T> &right)

friend EinSubtractionTensorExpression <T>
operator-<T> (const EinTensorExpression<T> &left,
const T &right)

~EinSubtractionTensorExpression ()

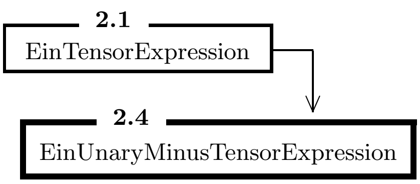
T **eval** () const

2.4

template <class T> class **EinUnaryMinusTensorExpression** : public EinTensorExpression<T>

EinUnaryMinusTensorExpression is a unary operator that negates a TensorExpression

Inheritance



Public Members

friend EinUnaryMinusTensorExpression <T>
operator-<T> (const EinTensorExpression<T> &expr)

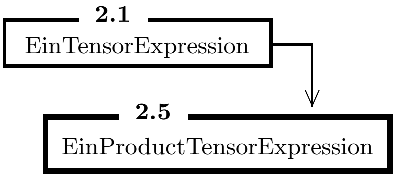
T **eval** () const

2.5

template <class T> class **EinProductTensorExpression** : public EinTensorExpression<T>

EinProductTensorExpression is a binary operator, which respects the Einstein summation convention

Inheritance



Public Members

~EinProductTensorExpression ()

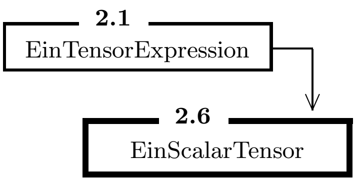
T **eval** () const

2.6

template <class T> class **EinScalarTensor** : public EinTensorExpression<T>

EinScalarTensor is a Scalar (involved in a tensor expression)

Inheritance



Public Members

	EinScalarTensor (const T &t_)
T	**eval** () const

2.7

template <class T> class **EinTensor**

EinTensor is a tensor, parametrized over the type T

Public Members

EinTensor (const EinMultiIndex &up, const EinMultiIndex &lo)
Constructs a tensor with an index space according to up and lo.

EinTensor (const EinTensor<T>&)
Copy constructor

void **operator=** (const EinTensor<T>&)
Assignment

~EinTensor () *Destructor*

EinTensorExpressionAtom <T>
operator) (const EinMultiIndex &up, const EinMultiIndex &lo)
Creates non const atom

const EinTensorExpressionAtom <T>
operator) (const EinMultiIndex &up, const EinMultiIndex &lo) const
Creates const atom

EinTensorExpressionAtom <T>
operator) (const EinMultiIndex &uplo)
If indices only up or only low, this interface provided

const EinTensorExpressionAtom <T>
operator) (const EinMultiIndex &uplo) const
A const version

PlusMinus <T>
operator[] (const EinMultiInt &m)
Returns a plus-minus-reference to a specific component

const PlusMinus <T>
operator[] (const EinMultiInt &m) const
A const version

void **setMultiIndices** (const EinMultiIndex &up, const EinMultiIndex &lo) const

Sets current associations

EinIndexSpace **getIndexSpace** () const
Returns the index space

EinIndexSpace **getUnrestrictedIndexSpace** () const
Returns the index space without symmetries

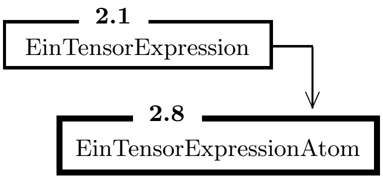
void **getMultiIndices** (EinMultiIndex &up, EinMultiIndex &lo) const
Returns current associations

2.8

template <class T> class **EinTensorExpressionAtom** : public EinTensorExpression<T>

EinTensorExpressionAtom is a Tensor involved in a tensor expression, when indexed with two multi-indices U and L

Inheritance



Public Members

void **operator=** (const EinTensorExpression<T> &)

void **operator=** (const EinScalarTensor<T> &)

void **operator=** (const EinTensorExpressionAtom<T>&)

T **eval** () const

EinTensorExpressionAtom (const EinTensorExpressionAtom&)

EinTensorExpressionAtom is a Tensor involved in a tensor expression, when indexed with two multi-indices U and L. Its eval() respects Einstein summation, and requires that no other indices are free.

When assigned to, it loops over its free indices.

Class Graph