GaBP

0.0.1

Generated by Doxygen 1.9.1

Chapter 1

Todo List

Member gmat::basematrix< T, m, n >::basematrix (const submatrix< T, m, n, M, N > &other)

Optimize this by copying rows or half-rows (need to split if the submatrix wraps)

Chapter 2

Namespace Index

2.1 Namespace List

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

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

Namespace Documentation

5.1 gmat Namespace Reference

The gmat namespace includes the linear algebra backend for GaBP.

Classes

- class matrix

  matrix class for linear algebra behind inference algorithms.
- class submatrix

  Class that shadows any matrix object and represents a rectangular selection of it (wrapping at boundaries).
- class basematrix

Functions

- template< typename T , size_t n>
  T det (std::shared_ptr< matrix< T, n, n >> mat)

  Calculates the determinant of the matrix.
- template< typename T >
  T **det** (std::shared_ptr< matrix< T, 1, 1 >> &mat)
- template< typename T , size_t n>
  bool inverse (matrix< T, n, n > &src, matrix< T, n, n > &dest)

  Calculates the inverse of a square matrix and writes it into dest.
- template< typename T , size_t m, size_t n, size_t o>
  void matmul (matrix< T, m, n > &left, matrix< T, n, o > &right, matrix< T, m, o > &dest)

  Calculates the product of two matrices and writes it into dest.
- template< typename T , size_t m, size_t n>
  void matadd (matrix< T, m, n > &left, matrix< T, m, n > &right, matrix< T, m, n > &dest)

  Calculates the entrywise sum of two matrices and writes it to dest.

5.1.1 Detailed Description

The gmat namespace includes the linear algebra backend for GaBP.

5.1.2 Function Documentation

5.1.2.1 det()

```
template<typename T , size_t n>
T gmat::det (
             std::shared_ptr< matrix< T, n, n >> mat )
```

Calculates the determinant of the matrix.

Template Parameters

T	Type of elements.
n	Number of rows and number of columns.

Parameters

mat	Shared pointer to matrix to calculate determinant of.

Returns

Determinant of type T.

5.1.2.2 inverse()

```
template<typename T , size_t n>
bool gmat::inverse (
             matrix< T, n, n > & src,
             matrix< T, n, n > & dest )
```

Calculates the inverse of a square matrix and writes it into dest.

Template Parameters

T	Type of elements.
n	Number of rows and number of columns.

Parameters

src	Reference to matrix to invert.
dest	Reference to matrix to write results.

Returns

true if src is singular (ie non-invertible). false if src is non-singular (ie invertible).

Invariant

src is unchanged.

This function writes the inverse of src into dest, unless src is singular, in which case it return true and leaves dest unchanged.

5.1.2.3 matadd()

```
template<typename T , size_t m, size_t n>
void gmat::matadd (
             matrix< T, m, n > & left,
             matrix< T, m, n > & right,
             matrix< T, m, n > & dest )
```

Calculates the entrywise sum of two matrices and writes it to dest.

Template Parameters

T	Type of elements.
m,n	Dimensions of the three matrices involved.

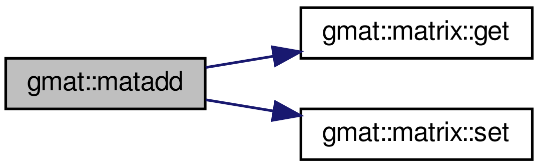
Parameters

left	Reference to left matrix to add.
right	Reference to right matrix to add.
dest	Reference to matrix to write results

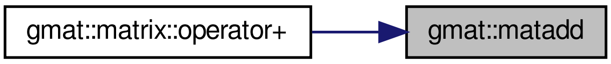
Invariant

left, right are unchanged.

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.2.4 matmul()

```
template<typename T , size_t m, size_t n, size_t o>
void gmat::matmul (
             matrix< T, m, n > & left,
             matrix< T, n, o > & right,
             matrix< T, m, o > & dest )
```

Calculates the product of two matrices and writes it into dest.

Template Parameters

T	Type of elements.
m,n,o	Dimensions of the three matrices involved. An $m*n$ matrix times an $n*o$ matrix is an $m*o$ matrix.

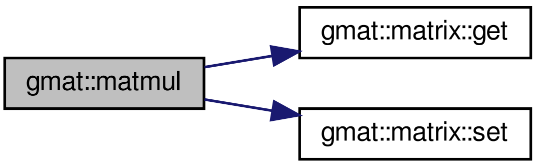
Parameters

left	Reference to left matrix to multiply.
right	Reference to right matrix to multiply.
dest	Reference to matrix to write results

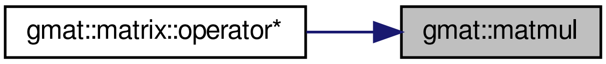
Invariant

left, right are unchanged.

Here is the call graph for this function:



Here is the caller graph for this function:

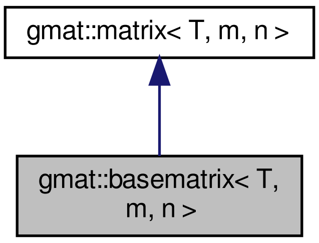


Chapter 6

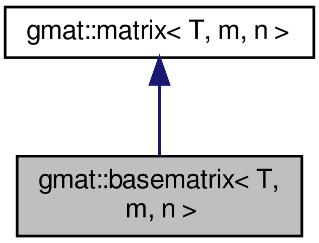
Class Documentation

6.1 gmat::basematrix< T, m, n > Class Template Reference

Inheritance diagram for gmat::basematrix< T, m, n >:



Collaboration diagram for gmat::basematrix< T, m, n >:



Public Member Functions

- basematrix ()

    Creates a basematrix object.
- basematrix (T ex)

    Creates a basematrix object with copies of an exemplar element.
- basematrix (T ∗ptr)

    Creates a basematrix object.
- basematrix (const basematrix< T, m, n > &other)

    basematrix copy constructor.
- template<size_t M, size_t N>
  basematrix (const submatrix< T, m, n, M, N > &other)

    basematrix constructor from a submatrix.
- T get (size_t i, size_t j) const override

    Gets the value of the element at coordinate i,j.
- T set (size_t i, size_t j, T value) override

    Sets the value of the element at coordinate i,j.
- template<size_t sm, size_t sn>
  submatrix< T, sm, sn, m, n > **submatrix** (size_t i, size_t j)

6.1.1 Constructor & Destructor Documentation

6.1.1.1 basematrix() [1/5]

```
template<typename T , size_t m, size_t n>
gmat::basematrix< T, m, n >::basematrix ( )  [inline]
```

Creates a basematrix object.

Warning

Not necessarily zero-valued.

This constructor does not clear or set the array.

6.1.1.2 basematrix() [2/5]

```
template<typename T , size_t m, size_t n>
gmat::basematrix< T, m, n >::basematrix (
             T ex )  [inline]
```

Creates a basematrix object with copies of an exemplar element.

Parameters

ex	Exemplar element.

This constructor fills the basematrix with $m*n$ copies of ex.

6.1.1.3 basematrix() [3/5]

```
template<typename T , size_t m, size_t n>
gmat::basematrix< T, m, n >::basematrix (
             T * ptr )  [inline]
```

Creates a basematrix object.

Parameters

ptr	Raw pointer to array of $m*n$ elements in memory.

This constructor copies the values from ptr into the basematrix.

6.1.1.4 basematrix() [4/5]

```
template<typename T , size_t m, size_t n>
gmat::basematrix< T, m, n >::basematrix (
             const basematrix< T, m, n > & other )  [inline]
```

basematrix copy constructor.

Parameters

other	Existing basematrix of identical element type and dimensions.

6.1.1.5 basematrix() [5/5]

```
template<typename T , size_t m, size_t n>
template<size_t M, size_t N>
gmat::basematrix< T, m, n >::basematrix (
             const submatrix< T, m, n, M, N > & other )  [inline]
```

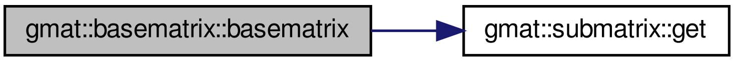
basematrix constructor from a submatrix.

Parameters

other	Existing submatrix of identical element type and dimensions.

Todo Optimize this by copying rows or half-rows (need to split if the submatrix wraps)

Here is the call graph for this function:



6.1.2 Member Function Documentation

6.1.2.1 get()

```
template<typename T , size_t m, size_t n>
T gmat::basematrix< T, m, n >::get (
             size_t i,
             size_t j ) const  [inline], [override], [virtual]
```

Gets the value of the element at coordinate i,j.

Parameters

i	Row coordinate.
j	Column coordinate.

Returns

Value at i,j.

Precondition

$i < m$

$j < n$

Reimplemented from gmat::matrix< T, m, n >.

6.1.2.2 set()

```
template<typename T , size_t m, size_t n>
T gmat::basematrix< T, m, n >::set (
             size_t i,
             size_t j,
             T value )  [inline], [override], [virtual]
```

Sets the value of the element at coordinate i,j.

Parameters

i	Row coordinate.
j	Column coordinate.
value	Value to be set.

Returns

New value.

Precondition

i < m

j < n

Reimplemented from gmat::matrix< T, m, n >.

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

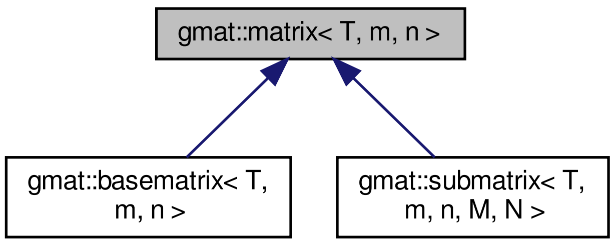
- include/gabp/matrix.hh

6.2 gmat::matrix< T, m, n > Class Template Reference

matrix class for linear algebra behind inference algorithms.

```
#include <matrix.hh>
```

Inheritance diagram for gmat::matrix< T, m, n >:



Public Member Functions

- virtual T get (size_t i, size_t j) const

    Gets the value of the element at coordinate i,j.
- virtual T set (size_t i, size_t j, T value)

    Sets the value of the element at coordinate i,j.
- template<size_t o>
  std::shared_ptr< matrix< T, m, o > > operator∗ (matrix< T, n, o > &right)

    Left matrix multiplication.
- std::shared_ptr< matrix< T, m, n > > operator+ (matrix< T, m, n > &right)

    Entryise matrix summation.
- bool operator== (matrix< T, m, n > &right)

    Compares this matrix with another.
- bool cmppred (matrix< T, m, n > &right, std::function< bool(T, T)> pred)

    Compares this matrix with another by a predicate.

Friends

- std::ostream & **operator<<** (std::ostream &out, const matrix< T, m, n > &mat)

6.2.1 Detailed Description

template<typename T, size_t m, size_t n>
class gmat::matrix< T, m, n >

matrix class for linear algebra behind inference algorithms.

Template Parameters

T	Type of elements.
m	Number of rows.
n	Number of columns.

6.2.2 Member Function Documentation

6.2.2.1 cmppred()

```
template<typename T , size_t m, size_t n>
bool gmat::matrix< T, m, n >::cmppred (
            matrix< T, m, n > & right,
            std::function< bool(T, T)> pred ) [inline]
```

Compares this matrix with another by a predicate.

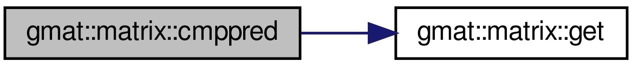
Parameters

right	Other matrix.
pred	Predicate to compare two values of type T.

Returns

true if each entry of the two matrices pass the predicate. false if any entry of the two matrices fails the predicate.

Here is the call graph for this function:



6.2.2.2 get()

```
template<typename T , size_t m, size_t n>
virtual T gmat::matrix< T, m, n >::get (
            size_t i,
            size_t j ) const  [virtual]
```

Gets the value of the element at coordinate i,j.

Parameters

i	Row coordinate.
j	Column coordinate.

Returns

Value at i,j.

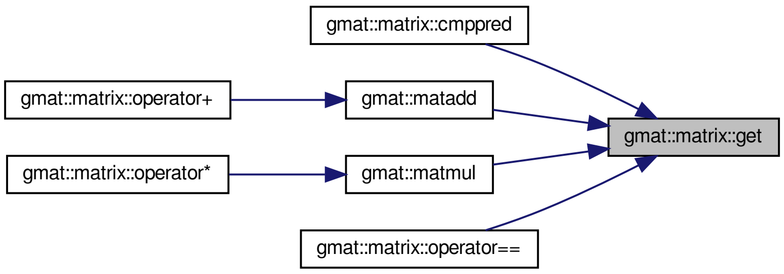
Precondition

$i < m$

$j < n$

Reimplemented in gmat::submatrix< T, m, n, M, N >, and gmat::basematrix< T, m, n >.

Here is the caller graph for this function:



6.2.2.3 operator∗()

```
template<typename T , size_t m, size_t n>
template<size_t o>
std::shared_ptr<matrix<T, m, o> > gmat::matrix< T, m, n >::operator* (
            matrix< T, n, o > & right )  [inline]
```

Left matrix multiplication.

Template Parameters

o	The width of the right matrix.

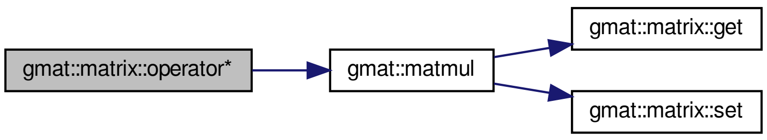
Parameters

right	An $n*o$ matrix.

Returns

The $m*o$ product of this $m*n$ matrix and the right $n*o$ matrix supplied.

Here is the call graph for this function:



6.2.2.4 operator+()

```
template<typename T , size_t m, size_t n>
std::shared_ptr<matrix<T, m, n> > gmat::matrix< T, m, n >::operator+ (
             matrix< T, m, n > & right )  [inline]
```

Entryise matrix summation.

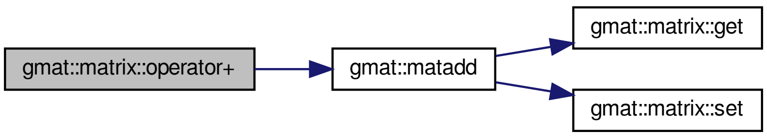
Parameters

right	Another $m*n$ matrix.

Returns

The $m*n$ sum of this $m*n$ matrix and the right $m*n$ matrix supplied.

Here is the call graph for this function:



6.2.2.5 operator==()

```
template<typename T , size_t m, size_t n>
bool gmat::matrix< T, m, n >::operator== (
             matrix< T, m, n > & right )  [inline]
```

Compares this matrix with another.

Parameters

right	Other matrix.

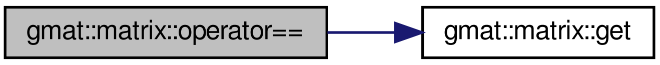
Returns

true if each entry of the two matrices are equal. false if the two matrices differ.

Warning

This function checks strict equality, and is not recommended for floating point matrices. Use comppred with a thresholding predicate instead.

Here is the call graph for this function:



6.2.2.6 set()

```
template<typename T , size_t m, size_t n>
virtual T gmat::matrix< T, m, n >::set (
            size_t i,
            size_t j,
            T value )  [virtual]
```

Sets the value of the element at coordinate i,j.

Parameters

i	Row coordinate.
j	Column coordinate.
value	Value to be set.

Returns

New value.

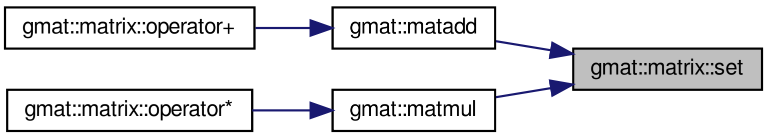
Precondition

$i < m$

$j < n$

Reimplemented in gmat::submatrix< T, m, n, M, N >, and gmat::basematrix< T, m, n >.

Here is the caller graph for this function:



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

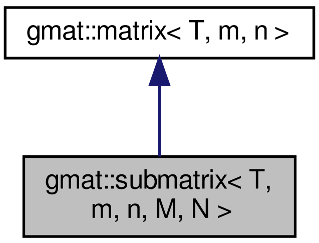
- include/gabp/matrix.hh

6.3 gmat::submatrix< T, m, n, M, N > Class Template Reference

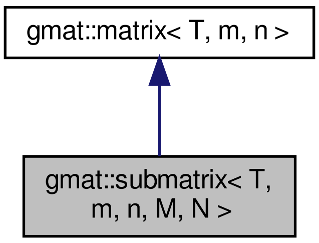
Class that shadows any matrix object and represents a rectangular selection of it (wrapping at boundaries).

`#include <matrix.hh>`

Inheritance diagram for gmat::submatrix< T, m, n, M, N >:



Collaboration diagram for gmat::submatrix< T, m, n, M, N >:



Public Member Functions

- submatrix (std::shared_ptr< matrix< T, M, N >> mat)

  Creates a submatrix object that directly mirrors a matrix.
- submatrix (std::shared_ptr< matrix< T, M, N >> mat, size_t i, size_t j)

  Creates a submatrix object that directly mirrors a matrix.
- T get (size_t i, size_t j) const override

  Gets the value of the element at coordinate i,j.
- T set (size_t i, size_t j, T value) override

  Sets the value of the element at coordinate i,j.

6.3.1 Detailed Description

template<typename T, size_t m, size_t n, size_t M, size_t N>
class gmat::submatrix< T, m, n, M, N >

Class that shadows any matrix object and represents a rectangular selection of it (wrapping at boundaries).

Template Parameters

T	Type of elements.
m	Number of rows in submatrix.
n	Number of columns in submatrix.
M	Number of rows in original matrix.
N	Number of columns in original matrix.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 submatrix() [1/2]

```
template<typename T , size_t m, size_t n, size_t M, size_t N>
gmat::submatrix< T, m, n, M, N >::submatrix (
            std::shared_ptr< matrix< T, M, N >> mat )  [inline]
```

Creates a submatrix object that directly mirrors a matrix.

Parameters

mat	shared_ptr to the matrix to be shadowed.

6.3.2.2 submatrix() [2/2]

```
template<typename T , size_t m, size_t n, size_t M, size_t N>
```

```
gmat::submatrix< T, m, n, M, N >::submatrix (
              std::shared_ptr< matrix< T, M, N >> mat,
              size_t i,
              size_t j )  [inline]
```

Creates a submatrix object that directly mirrors a matrix.

Parameters

mat	shared_ptr to the matrix to be shadowed.
i	Vertical offset from top of matrix.
j	Horizontal offset from left of matrix.

6.3.3 Member Function Documentation

6.3.3.1 get()

```
template<typename T , size_t m, size_t n, size_t M, size_t N>
T gmat::submatrix< T, m, n, M, N >::get (
              size_t i,
              size_t j ) const  [inline], [override], [virtual]
```

Gets the value of the element at coordinate i,j.

Parameters

i	Row coordinate.
j	Column coordinate.

Returns

Value at i,j.

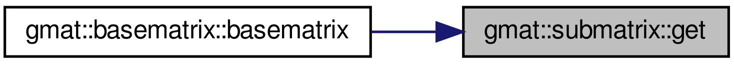
Precondition

$i < m$

$j < n$

Reimplemented from gmat::matrix< T, m, n >.

Here is the caller graph for this function:



6.3.3.2 set()

```
template<typename T , size_t m, size_t n, size_t M, size_t N>
T gmat::submatrix< T, m, n, M, N >::set (
              size_t i,
              size_t j,
              T value )  [inline], [override], [virtual]
```

Sets the value of the element at coordinate i,j.

Parameters

i	Row coordinate.
j	Column coordinate.
value	Value to be set.

Returns

New value.

Precondition

$i < m$

$j < n$

Reimplemented from gmat::matrix< T, m, n >.

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

- include/gabp/matrix.hh

Index