# frovedis::ell_matrix_local<T,I>

## NAME

`frovedis::ell_matrix_local<T,I>` - A two-dimensional non-distributed ELL sparse matrix.

## SYNOPSIS

`#include <frovedis/matrix/ell_matrix.hpp>`

### Constructors

ell_matrix_local ();
ell_matrix_local (const `ell_matrix_local<T,I>`& m);
ell_matrix_local (`ell_matrix_local<T,I>`&& m);
ell_matrix_local (const `crs_matrix_local<T,I,O>`& m);

### Overloaded Operators

`ell_matrix_local<T,I>`& operator= (const `ell_matrix_local<T,I>`& m);
`ell_matrix_local<T,I>`& operator= (`ell_matrix_local<T,I>`&& m);

### Public Member Functions

void debug_print ();
`crs_matrix_local<T,I,O>` to_crs();

### Public Data Members

`std::vector<T>` val;
`std::vector<I>` idx;
size_t local_num_row;
size_t local_num_col;

## DESCRIPTION

A ELL matrix is one of the most popular sparse matrices with elements stored in column-major order. In this matrix representation, all the non-zero elements of a row are shifted (packed) at left side and all the rows are padded with zeros on the right to give them equal length.

It has two major components while storing the non-zero elements, as explained below along with the number of rows and the number of columns in the sparse matrix.

```
val: a vector containing the left-shifted (zero-padded) non-zero elements of
the sparse matrix stored in column-major order.
idx: a vector containing the corresponding column indices of the non-zero elements.
```

For example, if we consider the below sparse matrix:

```
1 0 0 0 2 0 0 4
0 0 0 1 2 0 0 3
1 0 0 0 2 0 0 4
0 0 0 1 2 0 0 3
```

Then its ELL image can be thought of as:

```
values        indices
1 2 4          0 4 7
1 2 3   =>     3 4 7
1 2 4          0 4 7
1 2 3          3 4 7
```

And its column-major memory representation would be:

```
val: {1, 1, 1, 1, 2, 2, 2, 2, 4, 3, 4, 3}
idx: {0, 3, 0, 3, 4, 4, 4, 4, 7, 7, 7, 7}
```

`ell_matrix_local<T,I,O>` is a two-dimensional template based non-distributed sparse data storage supported by frovedis. The structure of this class is as follows:

```
template <class T, class I=size_t>
struct ell_matrix_local {
  std::vector<T> val;     // to contain non-zero elements of type "T"
  std::vector<I> idx;     // to contain column indices of type "I" (default: size_t)
  size_t local_num_row;   // number of rows in the sparse matrix
  size_t local_num_col;   // number of columns in the sparse matrix
};
```

This matrix can be loaded from a local crs matrix and also the matrix can be converted back to the local crs matrix. Thus loading/saving interfaces are not provided for local ell matrix.

## Constructor Documentation

### ell_matrix_local ()

This is the default constructor which creates an empty ell matrix with local_num_row = local_num_col = 0.

### ell_matrix_local (const `ell_matrix_local<T,I>&` m)

This is the copy constructor which creates a new ell matrix by deep-copying the contents of the input ell matrix.

**ell_matrix_local (`ell_matrix_local<T,I>`&& m)**

This is the move constructor. Instead of copying the input matrix, it moves the contents of the input rvalue matrix to the newly constructed matrix. Thus it is faster and recommended to use when input matrix will no longer be used in a user program.

**ell_matrix_local (const `crs_matrix_local<T,I,O>`& m)**

This is the implicit conversion constructor to construct a local ell matrix from the input local crs matrix of same "val" and "idx" type.

## Overloaded Operator Documentation

**`ell_matrix_local<T,I>`& operator= (const `ell_matrix_local<T,I>`& m)**

It deep-copies the input ell matrix into the left-hand side matrix of the assignment operator "=".

**`ell_matrix_local<T,I>`& operator= (`ell_matrix_local<T,I>`&& m)**

Instead of copying, it moves the contents of the input rvalue ell matrix into the left-hand side matrix of the assignment operator "=". Thus it is faster and recommended to use when input matrix will no longer be used in a user program.

## Public Member Function Documentation

**`crs_matrix_local<T,I,O>` to_crs()**

This method can be used to convert the target ell matrix into a local crs matrix of the same "val" and "idx" type.

**void debug_print ()**

It prints the information related to the ELL storage (val, idx, number of rows and number of columns) on the user terminal. It is mainly useful for debugging purpose.

## Public Data Member Documentation

**val**

An instance of `std::vector<T>` type to contain the non-zero elements of the ELL sparse matrix in column major order.

**idx**

An instance of `std::vector<I>` type to contain the column indices of the non-zero elements of the sparse matrix.

### local__num__row

A size_t attribute to contain the number of rows in the 2D matrix view.

### local__num__col

A size_t attribute to contain the number of columns in the 2D matrix view.

## Public Global Function Documentation

### `ell_matrix_local<T,I>` crs2ell(m)

**Parameters**
*m*: An object of the type `crs_matrix_local<T,I,O>`

**Purpose**
This function can be used to get a `ell_matrix_local<T,I>` from a `crs_matrix_local<T,I,O>`. Input matrix would remain unchanged.

**Return Value**
On success, it returns the created matrix of the type `ell_matrix_local<T,I>`. Otherwise, it throws an exception.

### `crs_matrix_local<T,I,O>` ell2crs(m)

**Parameters**
*m*: An object of the type `ell_matrix_local<T,I>`

**Purpose**
This function can be used to get a `crs_matrix_local<T,I,O>` from a `ell_matrix_local<T,I>`. Input matrix would remain unchanged.

**Return Value**
On success, it returns the created matrix of the type `crs_matrix_local<T,I,O>`. Otherwise, it throws an exception.

### `std::vector<T>` operator*(m,v)

**Parameters**
*m*: A const& object of the type `ell_matrix_local<T,I>`
*v*: A const& object of the type `std::vector<T>`

**Purpose**
This function performs matrix-vector multiplication between a sparse ell matrix object with a std::vector of same value (T) type. It expects the size of the input vector should be greater than or equal to the number of columns in the input ell matrix.

**Return Value**
On success, it returns the resultant vector of the type `std::vector<T>`. Otherwise, it throws an exception.

**`std::vector<T>` trans_mv(m,v)**

**Parameters**
*m*: A const& object of the type `ell_matrix_local<T,I>`
*v*: A const& object of the type `std::vector<T>`

**Purpose**
This function performs transposed matrix-vector multiplication (mT*v) between a sparse ell matrix object with a std::vector of same value (T) type. It expects the size of the input vector should be greater than or equal to the number of rows in the input ell matrix.

**Return Value**
On success, it returns the resultant vector of the type `std::vector<T>`. Otherwise, it throws an exception.

## SEE ALSO

crs_matrix_local, jds_matrix_local, ell_matrix