Webnucleo Technical Report: wn_matrix Module

David Adams

October 13, 2006

The following will provide information regarding the routines that comprise the wn_matrix Module. The wn_matrix Module is divided into two parts: routines that deal with creating and managing the WnMatrix structure, and routines for performing matrix operations with the WnMatrix structure. This report describes the function and usage of these routines, as well as the WnMatrix structure itself. The module itself can be found at:

http://nucleo.ces.clemson.edu/home/modules/wn_matrix/0.1

1 wn_matrix Structures Overview

WnMatrix is a structure for storing a matrix. Elements of the matrix are contained within an array of doubly-linked lists, which provide flexibility in adding and removing elements from the matrix. The WnMatrix structure is central to the module, as all routines contained within the module are used to either operate on the structure or retrieve information about its contents. The structure itself contains an array, the indices of which point to the first element of their respective rows, variables that store the (fixed) number of columns and rows of the matrix, and a variable with the total number of (dynamic) elements currently in the matrix. However, these structure members are not intended to be accessed directly, but rather through the helper routines. The members of the structure are not intended to be accessed directly, but the interested user may find them located in the WnMatrix header file `WnMatrix.h`.

The elements are stored within an Element structure, and are added to the matrix as instances of this structure. In addition to the variable containing the value of the element, there are also pointers to the next and previous elements of the row (if ones exist), and a variable representing the column number of that particular element.

2 wn_matrix Routines

The following is a list of the prototypes for all the routines contained within the wn_matrix module, as well as a brief description of their purpose.

- ```
  WnMatrix *WnMatrix__new(
    int *p_rows,
    int *p_columns
  );
  ```

  C routine to initialize a WnMatrix structure.

- ```
  void WnMatrix__assignElement(
    WnMatrix *self,
    int *p_row,
    int *p_col,
    double *p_val
  );
  ```

  C routine to assign an element to a WnMatrix structure. If an element already exists at that (row,col), the new value is added to the existing value.

- ```
  double WnMatrix__getElement(
    WnMatrix *self,
    int *p_row,
    int *p_col
  );
  ```

  Retrieves the value of the specified matrix element.

- ```
  void WnMatrix__removeElement(
    WnMatrix *self,
    int *p_row,
    int *p_col
  );
  ```

  Removes the specified matrix element from the matrix.

- ```
  void WnMatrix__clearRows(
    WnMatrix *self,
    int *p_n_rows
  );
  ```

  Removes the elements of the matrix up to the specified row.

- ```
  long int WnMatrix__getNumberElements(
    WnMatrix *self
  );
  ```

  Returns the number of elements in the matrix.

- ```
  int WnMatrix__getNumberRows(
    WnMatrix *self
  );
  ```

  Returns the number of rows in the matrix.
  ```

- ```
  long int WnMatrix__getNumberColumns(
    WnMatrix *self
  );
  ```

  Returns the number of columns in the matrix.

- ```
  void WnMatrix__getDiagElems(
    WnMatrix *self,
    double a_y[]
  );
  ```

  Stores the values of the diagonal elements in the double array passed into the function.

- ```
  int WnMatrix__isEmpty(
    WnMatrix *self,
    int i_row
  );
  ```

  Checks to see whether or not the given row is empty.

- ```
  void WnMatrix__getYsm(
    WnMatrix *self,
    int a_ija[],
    double a_sa[]
  );
  ```

  Stores the matrix in Yale Sparse format in the arrays passed into the function.

- ```
  void WnMatrix__getCsr(
    WnMatrix *self,
    int a_rwptr[],
    int a_col[],
    double a_val[]
  );
  ```

  A C routine to convert a WnMatrix into Compressed Sparse Row format.

- ```
  void WnMatrix__getMatrixTimesVector(
    WnMatrix *self,
    int *p_vector_size,
    double a_vector_input[],
    double a_vector_output[]
  );
  ```

  Computes the matrix equation Y = Ax given A and x.

- ```
  void WnMatrix__getTransposeMatrixTimesVector(
    WnMatrix *self,
    int *p_vector_size,
    double a_vector_input[],
  ```
  ```

```
  double a_vector_output[]
);
```

Computes the matrix equation Y = A(transpose)x given A and x.

- ```
  WnMatrix * WnMatrix__getTransferMatrix(
    WnMatrix *self
  );
  ```

  Returns the F matrix (aij/aii, but no diagonal elements).

- ```
  void WnMatrix__delete(
    WnMatrix *self
  );
  ```

  C routine to free the memory allocated for a WnMatrix structure.

- ```
  int WnMatrix__solveMatrixWithSparskit(
    WnMatrix *self,
    double a_rhs[],
    double a_sol[],
    double a_guess[]
  );
  ```

  C routine to solve matrix equation Ax = b by calling Sparskit routines.

- ```
  void WnMatrix__scaleMatrix(
    WnMatrix *self,
    double d_val
  );
  ```

  C routine to multiply all the elements in the matrix by a scalar constant.

- ```
  void WnMatrix__addElementToDiagonals(
    WnMatrix *self,
    double d_val
  );
  ```

  C routine to add a value to the diagonal elements of the matrix.

- ```
  int WnMatrix__writeMatrixToAsciiFile(
    WnMatrix *self,
    char * s_ascii_filename,
    double d_cutoff
  );
  ```

  C routine to write out the elements of a matrix stored in a WnMatrix structure.

3 Example C Code Calling WnMatrix Structure

The following code shows how to use the WnMatrix structure and routines in a C code. It is included in the distribution as `example1.c`. To compile, use

Makefile; thus, type `make example1`. Examples 2-6 are also included in the distribution, and may be compiled in the same fashion.

```
#include "WnMatrix.h"

int main() {

  int i_rows, i_cols;
  int i_row, i_col;
  double d_val;
  WnMatrix * p_my_matrix;

  /* Create a matrix called my_matrix and
  a pointer to it called p_my_matrix. */

  /* Create a 3 by 3 matrix */
  i_rows = 3;
  i_cols = 3;

  p_my_matrix = WnMatrix__new( &i_rows, &i_rows );

  /* Assign the following matrix:

     | 10.  0.   3. |
     |  0.  0.   0. |
     | -5.  2.   0. |
  */

  i_row = 1;
  i_col = 1;
  d_val = 10.;
  WnMatrix__assignElement( p_my_matrix, &i_row, &i_col, &d_val );

  i_row = 3;
  i_col = 1;
  d_val = -5.;
  WnMatrix__assignElement( p_my_matrix, &i_row, &i_col, &d_val );

  i_row = 3;
  i_col = 2;
  d_val = 2.;
  WnMatrix__assignElement( p_my_matrix, &i_row, &i_col, &d_val );

  i_row = 1;
  i_col = 3;
  d_val = 3.;
```

```
  WnMatrix__assignElement( p_my_matrix, &i_row, &i_col, &d_val );

  /* Print out the matrix */

  printf( "\nThe elements of the matrix are:\n\n" );
  printf( "Row   Column   Value\n" );
  printf( "---   ------   -----\n" );

  i_row = 1;
  i_col = 1;

  for ( i_row = 1; i_row <= 3; i_row++ ) {

    for ( i_col = 1; i_col <= 3; i_col++ ) {

      printf( "%3d   %6d   %5.1f\n",
        i_row,
        i_col,
        WnMatrix__getElement( p_my_matrix, &i_row, &i_col )
      );

    }

  }

  printf( "\n" );

  return 0;

}
```