Structural Analysis Library

1.0

Generated by Doxygen 1.5.6

Thu Oct 23 09:41:30 2008

Contents

Chapter 1

Structural Analysis Library

This document describes the application programming interface (API) of LibLA and LibStructural an open source (BSD) library for computing structural characteristics of cellular networks.

LibLA is a linear algebra library derives much of its functionality from the standard CLAPACK library with additional linear algebra functions not directly supported by CLAPACK. The libStructural library supports a range of methods for the structural analysis of cellular networks (derived either from SBML or stoichiometry matrices) and utilizes LibLA for some of its internal computations.

Installing

To make the Structural Analysis Library easily accessible we have created binary installers for Windows as wel as OS X (version 10.4 and above). We also habe a source distribution, complete with Visual Studio, XCode, Scons and Qt project files that allow to build the library on Windows, Linux and OS X. For detailed instructions on how to build the library see the file INSTALL included with the source distribution.

Dependencies

These libraries depend on two third-party libraries, LAPACK and libSBML. Both are provided with the binary installation where necessary.

This work was supported by a grant from the NIH (1R01GM0819070-01).

Author:

Frank T. Bergmann (fbergman@u.washington.edu)
Herbert M. Sauro
Ravishankar Rao Vallabhajosyula (developed a previous version of the sructural analysis code)

License

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

File Documentation

3.1 libla.h File Reference

All definitions needed for the LibLA (Linear Algebra) library.

```
#include "libutil.h"
```

Functions

- LIB_EXTERN void LibLA_freeMatrix (void **matrix, int numRows)

  Frees a matrix previously allocated by this library.

- LIB_EXTERN void LibLA_freeVector (void *vector)

  Frees a vector previously allocated by this library.

- LIB_EXTERN int LibLA_fullyPivotedGaussJordan (double **inMatrix, int numRows, int numCols, double ***outMatrix, int *outRows, int *outCols, int **oRowPivots, int *nRowLength, int **oColPivots, int *nColLength)

  This method calculates the fully pivoted Gauss Jordan form of the given matrix.

- LIB_EXTERN int LibLA_gaussJordan (double **inMatrix, int numRows, int numCols, double ***outMatrix, int *outRows, int *outCols, int **oPivots, int *nLength)

  This method calculates the Gauss Jordan or row echelon form of the given matrix.

- LIB_EXTERN int LibLA_getEigenValues (double **inMatrix, int numRows, int numCols, double **outReal, double **outImag, int *outLength)

  Calculates the eigen-values of a square real matrix.

- LIB_EXTERN int LibLA_getLU (double **inMatrix, int numRows, int numCols, double ***outL, int *outLRows, int *outLCols, double ***outU, int *outURows, int *outUCols, int ***outP, int *outPRows, int *outPCols, int *info)

  This function computes the LU factorization of the given real M-by-N matrix A.

- LIB_EXTERN int LibLA_getLUwithFullPivoting (double **inMatrix, int numRows, int numCols, double ***outL, int *outLRows, int *outLCols, double ***outU, int *outURows, int *outUCols, int ***outP, int *outPRows, int *outPCols, int ***outQ, int *outQRows, int *outQCols, int *info)

This function computes the LU factorization of the given real N-by-N matrix A using complete pivoting (with row and column interchanges).

- LIB_EXTERN int LibLA_getQR (double **inMatrix, int numRows, int numCols, double ***outQ, int *outQRows, int *outQCols, double ***outR, int *outRRows, int *outRCols)

  This function computes the QR factorization of the given real M-by-N matrix A with column pivoting.

- LIB_EXTERN int LibLA_getQRWithPivot (double **inMatrix, int numRows, int numCols, double ***outQ, int *outQRows, int *outQCols, double ***outR, int *outRRows, int *outRCols, double ***outP, int *outPRows, int *outPCols)

  This function computes the QR factorization of the given real M-by-N matrix A with column pivoting.

- LIB_EXTERN int LibLA_getRank (double **inMatrix, int numRows, int numCols)

  This method computes the rank of the given matrix.

- LIB_EXTERN int LibLA_getSingularValsBySVD (double **inMatrix, int numRows, int numCols, double **outSingularVals, int *outLength)

  This method performs the Singular Value Decomposition of the given real matrix, returning only the singular values.

- LIB_EXTERN double LibLA_getTolerance ()

  Returns the currently used tolerance.

- LIB_EXTERN int LibLA_inverse (double **inMatrix, int numRows, int numCols, double ***outMatrix, int *outRows, int *outCols)

  This function calculates the inverse of a square real matrix.

- LIB_EXTERN int LibLA_leftNullspace (double **inMatrix, int numRows, int numCols, double ***outMatrix, int *outRows, int *outCols)

  This function calculates the left null space of a given real matrix.

- LIB_EXTERN int LibLA_rightNullspace (double **inMatrix, int numRows, int numCols, double ***outMatrix, int *outRows, int *outCols)

  This function calculates the right null space of a given real matrix.

- LIB_EXTERN int LibLA_scaledLeftNullspace (double **inMatrix, int numRows, int numCols, double ***outMatrix, int *outRows, int *outCols)

  This function calculates the scaled left null space of a given real matrix.

- LIB_EXTERN int LibLA_scaledRightNullspace (double **inMatrix, int numRows, int numCols, double ***outMatrix, int *outRows, int *outCols)

  This function calculates the scaled right null space of a given real matrix.

- LIB_EXTERN void LibLA_setTolerance (const double value)

  Set user specified tolerance.

- LIB_EXTERN int LibLA_ZgetEigenValues (double **inMatrixReal, double **inMatrixImag, int numRows, int numCols, double **outReal, double **outImag, int *outLength)

  Calculates the eigen-values of a square complex matrix.

- LIB_EXTERN int LibLA_Zinverse (double **inMatrixReal, double **inMatrixImag, int numRows, int numCols, double ***outMatrixReal, double **outMatrixImag, int *outRows, int *outCols)

  This function calculates the inverse of a square complex matrix.

Variables

- BEGIN_C_DECLS
- END_C_DECLS

3.1.1 Detailed Description

All definitions needed for the LibLA (Linear Algebra) library.

The current scope of the library encompasses matrix factorizations (QR and LU factorization) as well as commonly needed matrix operations, such as calculating the inverse of a matrix, computing eigen values and singular values as well as the null space of a matrix (both left and right null space) along with a method for the computation of the row echelon or Gauss Jordan form of a matrix.

Author:

Frank T. Bergmann (fbergman@u.washington.edu)
Herbert M. Sauro
Ravishankar Rao Vallabhajosyula (developed a previous version of the sructural analysis code)

3.1.2 Function Documentation

3.1.2.1 LIB_EXTERN void LibLA_freeMatrix (void ** *matrix*, int *numRows*)

Frees a matrix previously allocated by this library.

3.1.2.2 LIB_EXTERN void LibLA_freeVector (void * *vector*)

Frees a vector previously allocated by this library.

3.1.2.3 LIB_EXTERN int LibLA_fullyPivotedGaussJordan (double ** *inMatrix*, int *numRows*, int *numCols*, double *** *outMatrix*, int * *outRows*, int * *outCols*, int ** *oRowPivots*, int * *nRowLength*, int ** *oColPivots*, int * *nColLength*)

This method calculates the fully pivoted Gauss Jordan form of the given matrix.

Fully pivoted means, that rows as well as column swaps will be used. These permutations are captured in the integer vectors rowPivots and colPivots.

If no permutations have occurred those vectors will be in ascending form [0, 1, 2, 3]; However if say row one and three would be swapped this vector would look like: [0, 3, 2, 1];

3.1.2.4 LIB_EXTERN int LibLA_gaussJordan (double ** *inMatrix*, int *numRows*, int *numCols*, double *** *outMatrix*, int * *outRows*, int * *outCols*, int ** *oPivots*, int * *nLength*)

This method calculates the Gauss Jordan or row echelon form of the given matrix.

Only row swaps are used. These permutations will be returned in the 'pivots' vector.

If no permutations have occurred this vector will be in ascending form [0, 1, 2, 3]; However if say row one and three would be swapped this vector would look like: [0, 3, 2, 1];

3.1.2.5 LIB_EXTERN int LibLA_getEigenValues (double ** *inMatrix*, int *numRows*, int *numCols*, double ** *outReal*, double ** *outImag*, int * *outLength*)

Calculates the eigen-values of a square real matrix.

This function calculates the complex eigenvalues of the given real matrix. The complex vector of eigenvalues will be returned in two real vectors, one for the real and one for the imaginary part.

Parameters:

inMatrix real matrix to calculate the eigen-values for.

numRows the number of rows of the input matrix

numCols the number of columns of the input matrix

outReal pointer to the output array for the eigenvalues (real part)

outImag pointer to the output array for the eigenvalues (imaginary part)

outLength the number of eigenvalues written to outReal and outImag

Returns:

The return value will be zero (0) when successful, and negative (-1) in case an error occurred (for example because the caller supplied a non-square matrix)

Remarks:

free outReal and outImag using LibLA_freeVector

3.1.2.6 LIB_EXTERN int LibLA_getLU (double ** *inMatrix*, int *numRows*, int *numCols*, double *** *outL*, int * *outLRows*, int * *outLCols*, double *** *outU*, int * *outURows*, int * *outUCols*, int *** *outP*, int * *outPRows*, int * *outPCols*, int * *info*)

This function computes the LU factorization of the given real M-by-N matrix A.

using partial pivoting with row interchanges. This procedure is carried out by the LAPACK method dgetrf.

A is factorized into:

A = P * L * U

Here P is the row permutation matrix.

3.1.2.7 LIB_EXTERN int LibLA_getLUwithFullPivoting (double ** *inMatrix*, int *numRows*, int *numCols*, double *** *outL*, int * *outLRows*, int * *outLCols*, double *** *outU*, int * *outURows*, int * *outUCols*, int *** *outP*, int * *outPRows*, int * *outPCols*, int *** *outQ*, int * *outQRows*, int * *outQCols*, int * *info*)

This function computes the LU factorization of the given real N-by-N matrix A using complete pivoting (with row and column interchanges).

This procedure is carried out by the LAPACK method dgetc2.

A is factorized into:

A = P * L * U * Q

Here P and Q are permutation matrices for the rows and columns respectively.

Remarks:

This function supports only square matrices (N-by-N), choose LibLA_getQRWithPivot for a stable method operating on N-by-M matrices.

3.1.2.8 LIB_EXTERN int LibLA_getQR (double ** *inMatrix*, int *numRows*, int *numCols*, double *** *outQ*, int * *outQRows*, int * *outQCols*, double *** *outR*, int * *outRRows*, int * *outRCols*)

This function computes the QR factorization of the given real M-by-N matrix A with column pivoting.

The LAPACK method dgeqp3 is used followed by an orthonormalization of Q through the use of DORGQR. The factorized form is:

A = Q * R

Parameters:

inMatrix real matrix to factorize

numRows number of rows of the matrix

numCols number of columns of the matrix

outQ pointer to a real matrix where Q will be written

outQRows number of rows of the Q matrix

outQCols number of columns of the Q matrix

outR pointer to a real matrix where R will be written

outRRows number of rows of the R matrix

outRCols number of columns of the R matrix

Returns:

The return value will be zero (0) when successful, and negative (-1) in case an error occurred

Remarks:

free outQ and outR using LibLA_freeMatrix

3.1.2.9 LIB_EXTERN int LibLA_getQRWithPivot (double ** *inMatrix*, int *numRows*, int *numCols*, double *** *outQ*, int * *outQRows*, int * *outQCols*, double *** *outR*, int * *outRRows*, int * *outRCols*, double *** *outP*, int * *outPRows*, int * *outPCols*)

This function computes the QR factorization of the given real M-by-N matrix A with column pivoting.

The LAPACK method dgeqp3 is used followed by an orthonormalization of Q through the use of DORGQR. The factorized form is:

$$A = Q * R$$

this method also returns the column pivots used in the P matrix.

Parameters:

inMatrix real matrix to factorize

numRows number of rows of the matrix

numCols number of columns of the matrix

outQ pointer to a real matrix where Q will be written

outQRows number of rows of the Q matrix

outQCols number of columns of the Q matrix

outR pointer to a real matrix where R will be written

outRRows number of rows of the R matrix

outRCols number of columns of the R matrix

outP pointer to a real matrix where P will be written

outPRows number of rows of the P matrix

outPCols number of columns of the P matrix

Returns:

The return value will be zero (0) when successful, and negative (-1) in case an error occurred

Remarks:

free outP, outQ and outR using LibLA_freeMatrix

3.1.2.10 LIB_EXTERN int LibLA_getRank (double ** *inMatrix*, int *numRows*, int *numCols*)

This method computes the rank of the given matrix.

The singular values of the matrix are calculated and the rank is determined by the number of non-zero values.

Note that zero here is defined as any value whose absolute value is bigger than the set tolerance (see LibLA_setTolerance)

3.1.2.11 LIB_EXTERN int LibLA_getSingularValsBySVD (double ** *inMatrix*, int *numRows*, int *numCols*, double ** *outSingularVals*, int * *outLength*)

This method performs the Singular Value Decomposition of the given real matrix, returning only the singular values.

This procedure is carried out by the LAPACK method dgesdd.

Parameters:

inMatrix real matrix

numRows number of rows of the matrix

numCols number of columns of the matrix

outSingularVals pointer to the double array where the singular values will be stored

outLength number of singular values

Returns:

The return value will be zero (0) when successful, and negative (-1) in case an error occurred

Remarks:

free outSingularVals using LibLA_freeVector

3.1.2.12 LIB_EXTERN double LibLA_getTolerance ()

Returns the currently used tolerance.

This function returns the tolerance currently used by the library to determine what value is considered as zero. Any value with absolute value smaller than this tolerance is considered zero and will be neglected.

3.1.2.13 LIB_EXTERN int LibLA_inverse (double ** *inMatrix*, int *numRows*, int *numCols*, double *** *outMatrix*, int * *outRows*, int * *outCols*)

This function calculates the inverse of a square real matrix.

This procedure is carried out by the LAPACK methods dgetrf and dgetri. This means that the matrix will be factorized using LU decomposition first, followed by the calculation of the inverse based on:

inv(A)*L = inv(U) for inv(A).

3.1.2.14 LIB_EXTERN int LibLA_leftNullspace (double ** *inMatrix*, int *numRows*, int *numCols*, double *** *outMatrix*, int * *outRows*, int * *outCols*)

This function calculates the left null space of a given real matrix.

That is:

null(A)*A = 0

Remarks:

This function is equivalent to returning the right null space of the transposed matrix. See LibLA_rightNullspace

3.1.2.15 LIB_EXTERN int LibLA_rightNullspace (double ** *inMatrix*, int *numRows*, int *numCols*, double *** *outMatrix*, int * *outRows*, int * *outCols*)

This function calculates the right null space of a given real matrix.

That is:

A∗null(A) = 0

In order to calculate the (right) null space, we first calculate the full singular value decomposition (employing dgesdd) of the matrix:

[U,S,V] = svd(A');

then calculate the rank:

r = rank(A)

and finally return the last columns of the U matrix (r+1...n) as the null space matrix.

3.1.2.16 LIB_EXTERN int LibLA_scaledLeftNullspace (double ** *inMatrix*, int *numRows*, int *numCols*, double *** *outMatrix*, int * *outRows*, int * *outCols*)

This function calculates the scaled left null space of a given real matrix.

This function is equivalent to calling LibLA_leftNullspace however the resulting matrix will be scaled (employing Gauss Jordan factorization) to yield whole numbered entries wherever possible.

3.1.2.17 LIB_EXTERN int LibLA_scaledRightNullspace (double ** *inMatrix*, int *numRows*, int *numCols*, double *** *outMatrix*, int * *outRows*, int * *outCols*)

This function calculates the scaled right null space of a given real matrix.

This function is equivalent to calling LibLA_rightNullspace however the resulting matrix will be scaled (employing Gauss Jordan factorization) to yield whole numbered entries wherever possible.

3.1.2.18 LIB_EXTERN void LibLA_setTolerance (const double *value*)

Set user specified tolerance.

This function sets the tolerance used by the library to determine what value is considered as zero. Any value with absolute value smaller than this tolerance is considered as zero and will be neglected.

Parameters:

value Sets the tolerance used by the library to determine a value close to zero

3.1.2.19 LIB_EXTERN int LibLA_ZgetEigenValues (double ** *inMatrixReal*, double ** *inMatrixImag*, int *numRows*, int *numCols*, double ** *outReal*, double ** *outImag*, int * *outLength*)

Calculates the eigen-values of a square complex matrix.

This function calculates the complex eigenvalues of the given complex matrix. The input matrix should be broken up into two matrices representing the real and imaginary parts respectively. The complex vector of eigenvalues will be returned in two real vectors, one for the real and one for the imaginary part.

Parameters:

inMatrixReal real part of the complex matrix to calculate the eigen-values for.

inMatrixImag imaginary part of the complex matrix to calculate the eigen-values for

numRows the number of rows of the input matrix

numCols the number of columns of the input matrix

outReal pointer to the output array for the eigenvalues (real part)

outImag pointer to the output array for the eigenvalues (imaginary part)

outLength the number of eigenvalues written to outReal and outImag

Returns:

The return value will be zero (0) when successful, and negative (-1) in case an error occurred (for example non square matrix)

Remarks:

free outReal and outImag using LibLA_freeVector

3.1.2.20 LIB_EXTERN int LibLA_Zinverse (double ** *inMatrixReal*, double ** *inMatrixImag*, int *numRows*, int *numCols*, double *** *outMatrixReal*, double ** *outMatrixImag*, int * *outRows*, int * *outCols*)

This function calculates the inverse of a square complex matrix.

This procedure is carried out by the LAPACK methods: zgetrf and zgetri. This means that the matrix will be factorized using LU decomposition first, followed by the calculation of the inverse based on:

inv(A)*L = inv(U) for inv(A).

3.1.3 Variable Documentation

3.1.3.1 BEGIN_C_DECLS

3.1.3.2 END_C_DECLS

3.2 libstructural.h File Reference

All definitions needed for the Structural Analysis Library.

```
#include "libutil.h"
```

Functions

- LIB_EXTERN int LibStructural_analyzeWithFullyPivotedLU (char **outMessage, int *nLength)

  Uses fully pivoted LU decomposition for structural analysis.

- LIB_EXTERN int LibStructural_analyzeWithFullyPivotedLUwithTests (char **outMessage, int *nLength)

  Uses fully pivoted LU decomposition for structural analysis.

- LIB_EXTERN int LibStructural_analyzeWithLU (char **outMessage, int *nLength)

  Uses LU Decomposition for structural analysis.

- LIB_EXTERN int LibStructural_analyzeWithLUandRunTests (char **outMessage, int *nLength)

  Uses LU Decomposition for structural analysis.

- LIB_EXTERN int LibStructural_analyzeWithQR (char **outMessage, int *nLength)

  Uses QR factorization for structural analysis.

- LIB_EXTERN void LibStructural_freeMatrix (void **matrix, int numRows)

  Frees a matrix previously allocated by this library.

- LIB_EXTERN void LibStructural_freeVector (void *vector)

  Frees a vector previously allocated by this library.

- LIB_EXTERN int LibStructural_getColumnReorderedNrMatrix (double ***outMatrix, int *outRows, int *outCols)

  Returns the Nr Matrix repartitioned into NIC (independent columns) and NDC (dependent columns).

- LIB_EXTERN int LibStructural_getColumnReorderedNrMatrixLabels (char ***outRowLabels, int *outRowCount, char ***outColLabels, int *outColCount)

  Returns the Nr Matrix row and column labels (repartitioned into NIC and NDC).

- LIB_EXTERN int LibStructural_getConservedLaws (char ***outArray, int *outLength)

  Returns algebraic expressions for the conservation laws.

- LIB_EXTERN int LibStructural_getConservedSums (double **outArray, int *outLength)

  Returns values for conservation laws using the current initial conditions.

- LIB_EXTERN int LibStructural_getDependentReactionIds (char ***outArray, int *outLength)

  Returns the list of dependent reaction Ids.

- LIB_EXTERN int LibStructural_getDependentSpeciesIds (char ***outArray, int *outLength)

  Returns the list of dependent species Ids.

- LIB_EXTERN int LibStructural_getFullyReorderedStoichiometryMatrix (double ***outMatrix, int *outRows, int *outCols)

  Returns the fully reordered stoichiometry matrix (row and column reordered stoichiometry matrix).

- LIB_EXTERN int LibStructural_getFullyReorderedStoichiometryMatrixLabels (char ***outRowLabels, int *outRowCount, char ***outColLabels, int *outColCount)

  Returns the row and column labels for the fully reordered stoichiometry matrix (row and column reordered stoichiometry matrix).

- LIB_EXTERN int LibStructural_getGammaMatrix (double ***outMatrix, int *outRows, int *outCols)

  Returns Gamma, the conservation law array.

- LIB_EXTERN int LibStructural_getGammaMatrixLabels (char ***outRowLabels, int *outRowCount, char ***outColLabels, int *outColCount)

  Returns the row and column labels for Gamma, the conservation law array.

- LIB_EXTERN int LibStructural_getIndependentReactionIds (char ***outArray, int *outLength)

  Returns the list of independent reaction ids.

- LIB_EXTERN int LibStructural_getIndependentSpeciesIds (char ***outArray, int *outLength)

  Returns the list of independent species ids.

- LIB_EXTERN int LibStructural_getInitialConditions (char ***outVariableNames, double **outValues, int *outLength)

  Returns the initial conditions used in the model.

- LIB_EXTERN int LibStructural_getK0Matrix (double ***outMatrix, int *outRows, int *outCols)

  Returns the K0 Matrix.

- LIB_EXTERN int LibStructural_getK0MatrixLabels (char ***outRowLabels, int *outRowCount, char ***outColLabels, int *outColCount)

  Returns the K0 Matrix row and column labels.

- LIB_EXTERN int LibStructural_getKMatrix (double ***outMatrix, int *outRows, int *outCols)

  Returns the K matrix (right nullspace of Nr) The K matrix has the structure, [I K0]'.

- LIB_EXTERN int LibStructural_getKMatrixLabels (char ***outRowLabels, int *outRowCount, char ***outColLabels, int *outColCount)

  Returns the K matrix row and column labels.

- LIB_EXTERN int LibStructural_getL0Matrix (double ***outMatrix, int *outRows, int *outCols)

  Returns the L0 Matrix.

- LIB_EXTERN int LibStructural_getL0MatrixLabels (char ***outRowLabels, int *outRowCount, char ***outColLabels, int *outColCount)

  Returns the L0 Matrix row and column labels.

- LIB_EXTERN int LibStructural_getLinkMatrix (double ***outMatrix, int *outRows, int *outCols)

  Returns L, the Link Matrix, left nullspace (aka nullspace of the transpose Nr).

- LIB_EXTERN int LibStructural_getLinkMatrixLabels (char ***outRowLabels, int *outRowCount, char ***outColLabels, int *outColCount)

  Returns the row and column labels for the Link Matrix, L.

- LIB_EXTERN int LibStructural_getModelName (char **outMessage, int *nLength)

  Returns the name of the model.

- LIB_EXTERN int LibStructural_getN0Matrix (double ***outMatrix, int *outRows, int *outCols)

  Returns the N0 Matrix.

- LIB_EXTERN int LibStructural_getN0MatrixLabels (char ***outRowLabels, int *outRowCount, char ***outColLabels, int *outColCount)

  Returns the N0 Matrix row and column labels.

- LIB_EXTERN int LibStructural_getNDCMatrix (double ***outMatrix, int *outRows, int *outCols)

  Returns the NDC Matrix (the set of linearly dependent columns of Nr).

- LIB_EXTERN int LibStructural_getNDCMatrixLabels (char ***outRowLabels, int *outRowCount, char ***outColLabels, int *outColCount)

  Returns the NDC Matrix row and column labels.

- LIB_EXTERN int LibStructural_getNICMatrix (double ***outMatrix, int *outRows, int *outCols)

  Returns the NIC Matrix (the set of linearly independent columns of Nr).

- LIB_EXTERN int LibStructural_getNICMatrixLabels (char ***outRowLabels, int *outRowCount, char ***outColLabels, int *outColCount)

  Returns the NIC Matrix row and column labels.

- LIB_EXTERN double LibStructural_getNmatrixSparsity ()

  Returns the percentage of nonzero values in the stoichiometry matrix.

- LIB_EXTERN int LibStructural_getNrMatrix (double ***outMatrix, int *outRows, int *outCols)

  Returns the Nr Matrix. The rows of the Nr matrix will be linearly independent.

- LIB_EXTERN int LibStructural_getNrMatrixLabels (char ***outRowLabels, int *outRowCount, char ***outColLabels, int *outColCount)

  Returns the Nr Matrix row and column labels.

- LIB_EXTERN int LibStructural_getNumConservedSums ()

  Returns the number of conservation laws.

- LIB_EXTERN int LibStructural_getNumDepReactions ()

  Returns the number of dependent reactions.

- LIB_EXTERN int LibStructural_getNumDepSpecies ()

  Returns the number of dependent species.

- LIB_EXTERN int LibStructural_getNumIndReactions ()

  Returns the number of independent reactions.

- LIB_EXTERN int LibStructural_getNumIndSpecies ()

  Returns the number of independent species.

- LIB_EXTERN int LibStructural_getNumReactions ()

  Returns the total number of reactions.

- LIB_EXTERN int LibStructural_getNumSpecies ()

  Returns the total number of species.

- LIB_EXTERN int LibStructural_getRank ()

  Returns the rank of the stoichiometry matrix.

- LIB_EXTERN int LibStructural_getReactionIds (char ***outArray, int *outLength)

  Returns the list of unordered Reactions. Returns the original list of reactions in the same order as when it was loaded.

- LIB_EXTERN int LibStructural_getReorderedReactionIds (char ***outArray, int *outLength)

  Returns the reordered list of reactions Ids.

- LIB_EXTERN int LibStructural_getReorderedSpeciesIds (char ***outArray, int *outLength)

  Returns the reordered list of molecular species.

- LIB_EXTERN int LibStructural_getReorderedStoichiometryMatrix (double ***outMatrix, int *outRows, int *outCols)

  Returns the reordered stoichiometry matrix (row reordered stoichiometry matrix, columns are not reordered!).

- LIB_EXTERN int LibStructural_getReorderedStoichiometryMatrixLabels (char ***outRowLabels, int *outRowCount, char ***outColLabels, int *outColCount)

  Returns the row and column labels for the reordered stoichiometry matrix (row reordered stoichiometry matrix).

- LIB_EXTERN int LibStructural_getSpeciesIds (char ***outArray, int *outLength)

  Returns the unordered list of species Ids.

- LIB_EXTERN int LibStructural_getStoichiometryMatrix (double ***outMatrix, int *outRows, int *outCols)

  Returns the original, unaltered stoichiometry matrix.

- LIB_EXTERN int LibStructural_getStoichiometryMatrixLabels (char ***outRowLabels, int *outRowCount, char ***outColLabels, int *outColCount)

  Returns the row and column labels for the original and unaltered stoichiometry matrix.

- LIB_EXTERN int LibStructural_getTestDetails (char **outMessage, int *nLength)

  Return Details about validation tests.

- LIB_EXTERN double LibStructural_getTolerance ()

Get user specified tolerance.

- LIB_EXTERN int LibStructural_loadReactionNames (const char **reactionNames, const int nLength)

  Load reaction names.

- LIB_EXTERN int LibStructural_loadSBML (const char *sSBML, char **outMessage, int *nLength)

  Load a SBML model.

- LIB_EXTERN int LibStructural_loadSBMLFromFile (const char *sFileName, char **outMessage, int *nLength)

  Load a SBML model from the specified file.

- LIB_EXTERN int LibStructural_loadSBMLwithTests (const char *sSBML, char **outMessage, int *nLength)

  Load an SBML model into the library and carry out tests using the internal test suite.

- LIB_EXTERN int LibStructural_loadSpecies (const char **speciesNames, const double *speciesValues, const int nLength)

  Load species names and initial values.

- LIB_EXTERN int LibStructural_loadStoichiometryMatrix (const double **oMatrix, const int nRows, const int nCols)

  Load a new stoichiometry matrix.

- LIB_EXTERN void LibStructural_setTolerance (const double dTolerance)

  Set user specified tolerance.

- LIB_EXTERN int LibStructural_validateStructuralMatrices (int **outResults, int *outLength)

  Validates structural matrices.

Variables

- BEGIN_C_DECLS
- END_C_DECLS

3.2.1 Detailed Description

All definitions needed for the Structural Analysis Library.

The structural analysis of stoichiometric networks is an important step in a number of computational methods in systems biology. The structure of a network based on the stoichiometry matrix is divided into two areas, structural constraints imposed by moiety conservation and constraints imposed by flux distributions at steady state. The former constraints have important applications in numerical methods for simulation and the analysis of control, while the later constraints have important applications in flux balance analysis. The LibStructural API provides a wide variety of methods that permit access to the constraint information in the stoichiometry matrix.

Stoichiometric Constraints

Moiety constraints concern the conservation of molecular subgroups in stoichiometric networks. Their existence results in dependencies among the model differential equations and the emergence of additional model parameters in the form of moiety mass totals. In the API we provide robust methods for extracting the constraint information and include specific methods to obtain for example the number of moiety cycles, the number of independent and dependent species and all the pertinent matrices such as the link matrix, reduced stoichiometry matrix etc. In addition to moiety constraints the library also provides robust methods for determining the flux constraints in a model. These include the dependent and independent flux, and the K matrix (and corresponding terms) that relates the two.

All matrices provided by the API are fully labeled with reaction and species labels. The API can accept models either directly from standard SBML or by specifying the stoichiometry matrix. In the case of SBML the species and reaction labels are obtained directly from the SBML otherwise they are entered manually.

Further and more detailed information on this work can be found in Reder (1988), Sauro and Ingalls (2004), Vallabhajosyula et al. (2005).

Author:

Frank T. Bergmann (fbergman@u.washington.edu)
Herbert M. Sauro
Ravishankar Rao Vallabhajosyula (developed a previous version of the sructural analysis code)

3.2.2 Function Documentation

3.2.2.1 LIB_EXTERN int LibStructural_analyzeWithFullyPivotedLU (char ** *outMessage*, int * *nLength*)

Uses fully pivoted LU decomposition for structural analysis.

This method performs the actual analysis of the stoichiometry matrix (loaded either via LibStructural_loadStoichiometryMatrix or LibStructural_loadSBML. Only after one of the analysis methods below has been called are the structural matrices (L0, K0...) available.

- LibStructural_analyzeWithQR,
- LibStructural_analyzeWithLU,
- LibStructural_analyzeWithLUandRunTests,
- LibStructural_analyzeWithFullyPivotedLU or
- LibStructural_analyzeWithFullyPivotedLUwithTests

Remarks:

Unlike the other methods, this method handles only square stoichiometry matrices. This method was only included for backward compatibility use LibStructural_analyzeWithQR

Parameters:

outMessage a pointer to a string where status information of the analysis will be returned.

nLength the length of the message.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand. See LibStructural_loadStoichiometryMatrix or LibStructural_loadSBML or LibStructural_loadSBMLFromFile

3.2.2.2 LIB_EXTERN int LibStructural_analyzeWithFullyPivotedLUwithTests (char ** *outMessage*, int * *nLength*)

Uses fully pivoted LU decomposition for structural analysis.

This method performs the actual analysis of the stoichiometry matrix (loaded either via LibStructural_loadStoichiometryMatrix or LibStructural_loadSBML. Only after one of the analysis methods below has been called are the structural matrices (L0, K0...) available.

- LibStructural_analyzeWithQR,
- LibStructural_analyzeWithLU,
- LibStructural_analyzeWithLUandRunTests,
- LibStructural_analyzeWithFullyPivotedLU or
- LibStructural_analyzeWithFullyPivotedLUwithTests

This method additionally performs the integrated test suite and returns those results.

Remarks:

Unlike the other methods, this method handles only square stoichiometry matrices. For non-square matrices use a method like LibStructural_analyzeWithQR.

Parameters:

outMessage a pointer to a string where status information of the analysis will be returned.

nLength the length of the message.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand see LibStructural_loadStoichiometryMatrix or LibStructural_loadSBML or LibStructural_loadSBMLFromFile

3.2.2.3 LIB_EXTERN int LibStructural_analyzeWithLU (char ** *outMessage*, int * *nLength*)

Uses LU Decomposition for structural analysis.

This method performs the actual analysis of the stoichiometry matrix (loaded either via LibStructural_loadStoichiometryMatrix or LibStructural_loadSBML. Only after one of the analysis methods below has been called are the structural matrices (L0, K0...) available.

- LibStructural_analyzeWithQR,

- LibStructural_analyzeWithLU,
- LibStructural_analyzeWithLUandRunTests,
- LibStructural_analyzeWithFullyPivotedLU or
- LibStructural_analyzeWithFullyPivotedLUwithTests

Parameters:

outMessage a pointer to a string where status information of the analysis will be returned.

nLength the length of the message.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand see LibStructural_loadStoichiometryMatrix or LibStructural_loadSBML or LibStructural_loadSBMLFromFile

3.2.2.4 LIB_EXTERN int LibStructural_analyzeWithLUandRunTests (char ** *outMessage*, int * *nLength*)

Uses LU Decomposition for structural analysis.

This method performs the actual analysis of the stoichiometry matrix (loaded either via LibStructural_loadStoichiometryMatrix or LibStructural_loadSBML. Only after one of the analysis methods below has been called are the structural matrices (L0, K0...) available.

- LibStructural_analyzeWithQR,
- LibStructural_analyzeWithLU,
- LibStructural_analyzeWithLUandRunTests,
- LibStructural_analyzeWithFullyPivotedLU or
- LibStructural_analyzeWithFullyPivotedLUwithTests

This method additionally performs the integrated test suite and returns those results.

Parameters:

outMessage a pointer to a string where status information of the analysis will be returned.

nLength the length of the message.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand see LibStructural_loadStoichiometryMatrix or LibStructural_loadSBML or LibStructural_loadSBMLFromFile

3.2.2.5 LIB_EXTERN int LibStructural_analyzeWithQR (char ** *outMessage*, int * *nLength*)

Uses QR factorization for structural analysis.

This method performs the actual analysis of the stoichiometry matrix (loaded either via LibStructural_loadStoichiometryMatrix or LibStructural_loadSBML. Only after one of the analysis methods below has been called are the structural matrices (L0, K0...) available.

- LibStructural_analyzeWithQR,
- LibStructural_analyzeWithLU,
- LibStructural_analyzeWithLUandRunTests,
- LibStructural_analyzeWithFullyPivotedLU or
- LibStructural_analyzeWithFullyPivotedLUwithTests

Remarks:

This is the prefered method for structural analysis.

Parameters:

outMessage a pointer to a string where status information of the analysis will be returned.

nLength the length of the message.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand see LibStructural_loadStoichiometryMatrix or LibStructural_loadSBML or LibStructural_loadSBMLFromFile

Examples:

examples/c/loadlabelledstoichiometry.c, and examples/c/loadstoichiometry.c.

3.2.2.6 LIB_EXTERN void LibStructural_freeMatrix (void ** *matrix*, int *numRows*)

Frees a matrix previously allocated by this library.

Examples:

examples/c/loadlabelledstoichiometry.c.

3.2.2.7 LIB_EXTERN void LibStructural_freeVector (void * *vector*)

Frees a vector previously allocated by this library.

Examples:

examples/c/loadlabelledstoichiometry.c, examples/c/loadsbmlfromfile.c, and examples/c/loadstoichiometry.c.

3.2.2.8 LIB_EXTERN int LibStructural_getColumnReorderedNrMatrix (double *** *outMatrix*, int * *outRows*, int * *outCols*)

Returns the Nr Matrix repartitioned into NIC (independent columns) and NDC (dependent columns).

Parameters:

outMatrix a pointer to a double array that holds the output

outRows will be overwritten with the number of rows

outCols will be overwritten with the number of columns.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

Remarks:

To free the returned matrix call LibStructural_freeMatrix with the outMatrix and outRows as parameter.

3.2.2.9 LIB_EXTERN int LibStructural_getColumnReorderedNrMatrixLabels (char *** *outRowLabels*, int * *outRowCount*, char *** *outColLabels*, int * *outColCount*)

Returns the Nr Matrix row and column labels (repartitioned into NIC and NDC).

Parameters:

outRowLabels a pointer to a string array where the row labels will be allocated and written.

outRowCount after the call this variable will hold the number of row labels returned.

outColLabels a pointer to a string array where the column labels will be allocated and written.

outColCount after the call this variable will hold the number of column labels returned.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

Remarks:

To free the string arrays (outRowLabels and outColLabels) call LibStructural_freeMatrix with the string array and its corresponding length (outRowCount or outColCount)

3.2.2.10 LIB_EXTERN int LibStructural_getConservedLaws (char *** *outArray*, int * *outLength*)

Returns algebraic expressions for the conservation laws.

Parameters:

outArray pointer to string array that will be allocated and filled

outLength the number of conservation laws

Remarks:

free outArray using LibStructural_freeMatrix with the outLength parameter

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

3.2.2.11 LIB_EXTERN int LibStructural_getConservedSums (double ** *outArray*, int * *outLength*)

Returns values for conservation laws using the current initial conditions.

Parameters:

outArray will be allocated and filled with a double vector of all conserved sums

outLength is the number of conserved sums

Remarks:

free outArray using LibStructural_freeVector

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

3.2.2.12 LIB_EXTERN int LibStructural_getDependentReactionIds (char *** *outArray*, int * *outLength*)

Returns the list of dependent reaction Ids.

Parameters:

outArray pointer to string array that will be allocated and filled with the dependent reaction Ids

outLength the number of dependent reactions

Remarks:

free outArray using LibStructural_freeMatrix with the outLength parameter

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

3.2.2.13 LIB_EXTERN int LibStructural_getDependentSpeciesIds (char *** *outArray*, int * *outLength*)

Returns the list of dependent species Ids.

Parameters:

outArray pointer to string array that will be allocated and filled with the dependent species Ids

outLength the number of dependent species

Remarks:

free outArray using LibStructural_freeMatrix with the outLength parameter

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

3.2.2.14 LIB_EXTERN int LibStructural_getFullyReorderedStoichiometryMatrix (double *** *outMatrix*, int * *outRows*, int * *outCols*)

Returns the fully reordered stoichiometry matrix (row and column reordered stoichiometry matrix).

Parameters:

outMatrix a pointer to a double array that holds the output

outRows will be overwritten with the number of rows

outCols will be overwritten with the number of columns.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

Remarks:

To free the returned matrix call LibStructural_freeMatrix with the outMatrix and outRows as parameter.

3.2.2.15 LIB_EXTERN int LibStructural_getFullyReorderedStoichiometryMatrixLabels (char *** *outRowLabels*, int * *outRowCount*, char *** *outColLabels*, int * *outColCount*)

Returns the row and column labels for the fully reordered stoichiometry matrix (row and column reordered stoichiometry matrix).

Parameters:

outRowLabels a pointer to a string array where the row labels will be allocated and written.

outRowCount after the call this variable will hold the number of row labels returned.

outColLabels a pointer to a string array where the column labels will be allocated and written.

outColCount after the call this variable will hold the number of column labels returned.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

Remarks:

To free the string arrays (outRowLabels and outColLabels) call LibStructural_freeMatrix with the string array and its corresponding length (outRowCount or outColCount)

3.2.2.16 LIB_EXTERN int LibStructural_getGammaMatrix (double *** *outMatrix*, int * *outRows*, int * *outCols*)

Returns Gamma, the conservation law array.

Each row represents a single conservation law where the column indicate the participating molecular species. The number of rows is therefore equal to the number of conservation laws. Columns are ordered according to the rows in the reordered stoichiometry matrix, see LibStructural_getReorderedSpeciesId and LibStructural_getReorderedStoichiometryMatrix.

Parameters:

outMatrix a pointer to a double array that holds the output

outRows will be overwritten with the number of rows

outCols will be overwritten with the number of columns.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

Remarks:

To free the returned matrix call LibStructural_freeMatrix with the outMatrix and outRows as parameter.

Examples:

examples/c/loadlabelledstoichiometry.c.

3.2.2.17 LIB_EXTERN int LibStructural_getGammaMatrixLabels (char *** *outRowLabels*, int * *outRowCount*, char *** *outColLabels*, int * *outColCount*)

Returns the row and column labels for Gamma, the conservation law array.

Parameters:

outRowLabels a pointer to a string array where the row labels will be allocated and written.

outRowCount after the call this variable will hold the number of row labels returned.

outColLabels a pointer to a string array where the column labels will be allocated and written.

outColCount after the call this variable will hold the number of column labels returned.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

Remarks:

To free the string arrays (outRowLabels and outColLabels) call LibStructural_freeMatrix with the string array and its corresponding length (outRowCount or outColCount)

Examples:

examples/c/loadlabelledstoichiometry.c.

3.2.2.18 LIB_EXTERN int LibStructural_getIndependentReactionIds (char *** *outArray*, int * *outLength*)

Returns the list of independent reaction ids.

Parameters:

outArray pointer to string array that will be allocated and filled with the independent reaction Ids

outLength the number of independent reaction

Remarks:

free outArray using LibStructural_freeMatrix with the outLength parameter

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

3.2.2.19 LIB_EXTERN int LibStructural_getIndependentSpeciesIds (char *** *outArray*, int * *outLength*)

Returns the list of independent species ids.

Parameters:

outArray pointer to string array that will be allocated and filled with the independent species Ids

outLength the number of independent species

Remarks:

free outArray using LibStructural_freeMatrix with the outLength parameter

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

3.2.2.20 LIB_EXTERN int LibStructural_getInitialConditions (char *** *outVariableNames*, double ** *outValues*, int * *outLength*)

Returns the initial conditions used in the model.

Parameters:

outVariableNames a string vector of all species Ids

outValues a double vector of corresponding initial conditions

outLength number of elements in outVariableNames and outValues (number of species)

3.2.2.21 LIB_EXTERN int LibStructural_getK0Matrix (double *** *outMatrix*, int * *outRows*, int * *outCols*)

Returns the K0 Matrix.

K0 is defined such that K0 = -(NIC)^-1 NDC, or equivalently, [NDC NIC][I K0]' = 0 where [NDC NIC] = Nr

Parameters:

outMatrix a pointer to a double array that holds the output

outRows will be overwritten with the number of rows

outCols will be overwritten with the number of columns.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

Remarks:

To free the returned matrix call LibStructural_freeMatrix with the outMatrix and outRows as parameter.

3.2.2.22 LIB_EXTERN int LibStructural_getK0MatrixLabels (char *** *outRowLabels*, int * *outRowCount*, char *** *outColLabels*, int * *outColCount*)

Returns the K0 Matrix row and column labels.

Parameters:

outRowLabels a pointer to a string array where the row labels will be allocated and written.

outRowCount after the call this variable will hold the number of row labels returned.

outColLabels a pointer to a string array where the column labels will be allocated and written.

outColCount after the call this variable will hold the number of column labels returned.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

Remarks:

To free the string arrays (outRowLabels and outColLabels) call LibStructural_freeMatrix with the string array and its corresponding length (outRowCount or outColCount)

3.2.2.23 LIB_EXTERN int LibStructural_getKMatrix (double *** *outMatrix*, int * *outRows*, int * *outCols*)

Returns the K matrix (right nullspace of Nr) The K matrix has the structure, [I K0]'.

Parameters:

outMatrix a pointer to a double array that holds the output

outRows will be overwritten with the number of rows

outCols will be overwritten with the number of columns.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

Remarks:

To free the returned matrix call LibStructural_freeMatrix with the outMatrix and outRows as parameter.

3.2.2.24 LIB_EXTERN int LibStructural_getKMatrixLabels (char *** *outRowLabels*, int * *outRowCount*, char *** *outColLabels*, int * *outColCount*)

Returns the K matrix row and column labels.

Parameters:

outRowLabels a pointer to a string array where the row labels will be allocated and written.

outRowCount after the call this variable will hold the number of row labels returned.

outColLabels a pointer to a string array where the column labels will be allocated and written.

outColCount after the call this variable will hold the number of column labels returned.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

Remarks:

To free the string arrays (outRowLabels and outColLabels) call LibStructural_freeMatrix with the string array and its corresponding length (outRowCount or outColCount)

3.2.2.25 LIB_EXTERN int LibStructural_getL0Matrix (double *** *outMatrix*, int * *outRows*, int * *outCols*)

Returns the L0 Matrix.

L0 is defined such that L0 Nr = N0. L0 forms part of the link matrix, L. N0 is the set of linear dependent rows from the lower portion of the reordered stoichiometry matrix.

Parameters:

outMatrix a pointer to a double array that holds the output

outRows will be overwritten with the number of rows

outCols will be overwritten with the number of columns.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods have been called yet.

Remarks:

To free the returned matrix call LibStructural_freeMatrix with the outMatrix and outRows as parameter.

3.2.2.26 LIB_EXTERN int LibStructural_getL0MatrixLabels (char *** *outRowLabels*, int * *outRowCount*, char *** *outColLabels*, int * *outColCount*)

Returns the L0 Matrix row and column labels.

Parameters:

outRowLabels a pointer to a string array where the row labels will be allocated and written.

outRowCount after the call this variable will hold the number of row labels returned.

outColLabels a pointer to a string array where the column labels will be allocated and written.

outColCount after the call this variable will hold the number of column labels returned.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

Remarks:

To free the string arrays (outRowLabels and outColLabels) call LibStructural_freeMatrix with the string array and its corresponding length (outRowCount or outColCount)

3.2.2.27 LIB_EXTERN int LibStructural_getLinkMatrix (double *** *outMatrix*, int * *outRows*, int * *outCols*)

Returns L, the Link Matrix, left nullspace (aka nullspace of the transpose Nr).

L will have the structure, [I L0]', such that L Nr = N

Parameters:

outMatrix a pointer to a double array that holds the output

outRows will be overwritten with the number of rows

outCols will be overwritten with the number of columns.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

Remarks:

To free the returned matrix call LibStructural_freeMatrix with the outMatrix and outRows as parameter.

3.2.2.28 LIB_EXTERN int LibStructural_getLinkMatrixLabels (char *** *outRowLabels*, int * *outRowCount*, char *** *outColLabels*, int * *outColCount*)

Returns the row and column labels for the Link Matrix, L.

Parameters:

outRowLabels a pointer to a string array where the row labels will be allocated and written.

outRowCount after the call this variable will hold the number of row labels returned.

outColLabels a pointer to a string array where the column labels will be allocated and written.

outColCount after the call this variable will hold the number of column labels returned.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

Remarks:

To free the string arrays (outRowLabels and outColLabels) call LibStructural_freeMatrix with the string array and its corresponding length (outRowCount or outColCount)

3.2.2.29 LIB_EXTERN int LibStructural_getModelName (char ** *outMessage*, int * *nLength*)

Returns the name of the model.

Returns the name of the model if SBML model has Name-tag, otherwise it returns the SBML id. If only a stoichiometry matrix was loaded 'untitled' will be returned.

Parameters:

outMessage a pointer to a string where status information of the analysis will be returned.

nLength the length of the message.

Remarks:

free outMessage using LibStructural_freeVector

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

3.2.2.30 LIB_EXTERN int LibStructural_getN0Matrix (double *** *outMatrix*, int * *outRows*, int * *outCols*)

Returns the N0 Matrix.

The N0 matrix is the set of linearly dependent rows of N where L0 Nr = N0.

Parameters:

outMatrix a pointer to a double array that holds the output

outRows will be overwritten with the number of rows

outCols will be overwritten with the number of columns.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

Remarks:

To free the returned matrix call LibStructural_freeMatrix with the outMatrix and outRows as parameter.

3.2.2.31 LIB_EXTERN int LibStructural_getN0MatrixLabels (char *** *outRowLabels*, int * *outRowCount*, char *** *outColLabels*, int * *outColCount*)

Returns the N0 Matrix row and column labels.

Parameters:

outRowLabels a pointer to a string array where the row labels will be allocated and written.

outRowCount after the call this variable will hold the number of row labels returned.

outColLabels a pointer to a string array where the column labels will be allocated and written.

outColCount after the call this variable will hold the number of column labels returned.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

Remarks:

To free the string arrays (outRowLabels and outColLabels) call LibStructural_freeMatrix with the string array and its corresponding length (outRowCount or outColCount)

3.2.2.32 LIB_EXTERN int LibStructural_getNDCMatrix (double *** *outMatrix*, int * *outRows*, int * *outCols*)

Returns the NDC Matrix (the set of linearly dependent columns of Nr).

Parameters:

outMatrix a pointer to a double array that holds the output

outRows will be overwritten with the number of rows

outCols will be overwritten with the number of columns.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

Remarks:

To free the returned matrix call LibStructural_freeMatrix with the outMatrix and outRows as parameter.

3.2.2.33 LIB_EXTERN int LibStructural_getNDCMatrixLabels (char *** *outRowLabels*, int * *outRowCount*, char *** *outColLabels*, int * *outColCount*)

Returns the NDC Matrix row and column labels.

Parameters:

outRowLabels a pointer to a string array where the row labels will be allocated and written.

outRowCount after the call this variable will hold the number of row labels returned.

outColLabels a pointer to a string array where the column labels will be allocated and written.

outColCount after the call this variable will hold the number of column labels returned.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

Remarks:

To free the string arrays (outRowLabels and outColLabels) call LibStructural_freeMatrix with the string array and its corresponding length (outRowCount or outColCount)

3.2.2.34 LIB_EXTERN int LibStructural_getNICMatrix (double *** *outMatrix*, int * *outRows*, int * *outCols*)

Returns the NIC Matrix (the set of linearly independent columns of Nr).

Parameters:

outMatrix a pointer to a double array that holds the output

outRows will be overwritten with the number of rows

outCols will be overwritten with the number of columns.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

Remarks:

To free the returned matrix call LibStructural_freeMatrix with the outMatrix and outRows as parameter.

3.2.2.35 LIB_EXTERN int LibStructural_getNICMatrixLabels (char *** *outRowLabels*, int * *outRowCount*, char *** *outColLabels*, int * *outColCount*)

Returns the NIC Matrix row and column labels.

Parameters:

outRowLabels a pointer to a string array where the row labels will be allocated and written.

outRowCount after the call this variable will hold the number of row labels returned.

outColLabels a pointer to a string array where the column labels will be allocated and written.

outColCount after the call this variable will hold the number of column labels returned.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

Remarks:

To free the string arrays (outRowLabels and outColLabels) call LibStructural_freeMatrix with the string array and its corresponding length (outRowCount or outColCount)

3.2.2.36 LIB_EXTERN double LibStructural_getNmatrixSparsity ()

Returns the percentage of nonzero values in the stoichiometry matrix.

3.2.2.37 LIB_EXTERN int LibStructural_getNrMatrix (double *** *outMatrix*, int * *outRows*, int * *outCols*)

Returns the Nr Matrix. The rows of the Nr matrix will be linearly independent.

Parameters:

outMatrix a pointer to a double array that holds the output

outRows will be overwritten with the number of rows

outCols will be overwritten with the number of columns.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

Remarks:

To free the returned matrix call LibStructural_freeMatrix with the outMatrix and outRows as parameter.

3.2.2.38 LIB_EXTERN int LibStructural_getNrMatrixLabels (char *** *outRowLabels*, int * *outRowCount*, char *** *outColLabels*, int * *outColCount*)

Returns the Nr Matrix row and column labels.

Parameters:

outRowLabels a pointer to a string array where the row labels will be allocated and written.

outRowCount after the call this variable will hold the number of row labels returned.

outColLabels a pointer to a string array where the column labels will be allocated and written.

outColCount after the call this variable will hold the number of column labels returned.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

Remarks:

To free the string arrays (outRowLabels and outColLabels) call LibStructural_freeMatrix with the string array and its corresponding length (outRowCount or outColCount)

3.2.2.39 LIB_EXTERN int LibStructural_getNumConservedSums ()

Returns the number of conservation laws.

Returns:

the number of conservation laws

3.2.2.40 LIB_EXTERN int LibStructural_getNumDepReactions ()

Returns the number of dependent reactions.

3.2.2.41 LIB_EXTERN int LibStructural_getNumDepSpecies ()

Returns the number of dependent species.

3.2.2.42 LIB_EXTERN int LibStructural_getNumIndReactions ()

Returns the number of independent reactions.

3.2.2.43 LIB_EXTERN int LibStructural_getNumIndSpecies ()

Returns the number of independent species.

3.2.2.44 LIB_EXTERN int LibStructural_getNumReactions ()

Returns the total number of reactions.

3.2.2.45 LIB_EXTERN int LibStructural_getNumSpecies ()

Returns the total number of species.

3.2.2.46 LIB_EXTERN int LibStructural_getRank ()

Returns the rank of the stoichiometry matrix.

3.2.2.47 LIB_EXTERN int LibStructural_getReactionIds (char *** *outArray*, int * *outLength*)

Returns the list of unordered Reactions. Returns the original list of reactions in the same order as when it was loaded.

Parameters:

outArray pointer to string array that will be allocated and filled with the reaction Ids

outLength the number of reactions

Remarks:

free outArray using LibStructural_freeMatrix with the outLength parameter

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

3.2.2.48 LIB_EXTERN int LibStructural_getReorderedReactionIds (char *** *outArray*, int * *outLength*)

Returns the reordered list of reactions Ids.

Parameters:

outArray pointer to string array that will be allocated and filled with the reordered reaction Ids

outLength the number of species

Remarks:

free outArray using LibStructural_freeMatrix with the outLength parameter

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

3.2.2.49 LIB_EXTERN int LibStructural_getReorderedSpeciesIds (char *** *outArray*, int * *outLength*)

Returns the reordered list of molecular species.

Parameters:

outArray pointer to string array that will be allocated and filled with the species Ids

outLength the number of species

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

Remarks:

free outArray using LibStructural_freeMatrix with the outLength parameter

3.2.2.50 LIB_EXTERN int LibStructural_getReorderedStoichiometryMatrix (double *** *outMatrix*, int * *outRows*, int * *outCols*)

Returns the reordered stoichiometry matrix (row reordered stoichiometry matrix, columns are not reordered!).

Parameters:

outMatrix a pointer to a double array that holds the output

outRows will be overwritten with the number of rows

outCols will be overwritten with the number of columns.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

Remarks:

To free the returned matrix call LibStructural_freeMatrix with the outMatrix and outRows as parameter.

Examples:

examples/c/loadlabelledstoichiometry.c.

3.2.2.51 LIB_EXTERN int LibStructural_getReorderedStoichiometryMatrixLabels (char *** *outRowLabels*, int * *outRowCount*, char *** *outColLabels*, int * *outColCount*)

Returns the row and column labels for the reordered stoichiometry matrix (row reordered stoichiometry matrix).

Parameters:

outRowLabels a pointer to a string array where the row labels will be allocated and written.

outRowCount after the call this variable will hold the number of row labels returned.

outColLabels a pointer to a string array where the column labels will be allocated and written.

outColCount after the call this variable will hold the number of column labels returned.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

Remarks:

To free the string arrays (outRowLabels and outColLabels) call LibStructural_freeMatrix with the string array and its corresponding length (outRowCount or outColCount)

Examples:

examples/c/loadlabelledstoichiometry.c.

3.2.2.52 LIB_EXTERN int LibStructural_getSpeciesIds (char *** *outArray*, int * *outLength*)

Returns the unordered list of species Ids.

Parameters:

outArray pointer to string array that will be allocated and filled with the species Ids

outLength the number of species

Remarks:

free outArray using LibStructural_freeMatrix with the outLength parameter

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

3.2.2.53 LIB_EXTERN int LibStructural_getStoichiometryMatrix (double *** *outMatrix*, int * *outRows*, int * *outCols*)

Returns the original, unaltered stoichiometry matrix.

Parameters:

outMatrix a pointer to a double array that holds the output

outRows will be overwritten with the number of rows

outCols will be overwritten with the number of columns.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

Remarks:

To free the returned matrix call LibStructural_freeMatrix with the outMatrix and outRows as parameter.

3.2.2.54 LIB_EXTERN int LibStructural_getStoichiometryMatrixLabels (char *** *outRowLabels*, int * *outRowCount*, char *** *outColLabels*, int * *outColCount*)

Returns the row and column labels for the original and unaltered stoichiometry matrix.

Parameters:

outRowLabels a pointer to a string array where the row labels will be allocated and written.

outRowCount after the call this variable will hold the number of row labels returned.

outColLabels a pointer to a string array where the column labels will be allocated and written.

outColCount after the call this variable will hold the number of column labels returned.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

Remarks:

To free the string arrays (outRowLabels and outColLabels) call LibStructural_freeMatrix with the string array and its corresponding length (outRowCount or outColCount)

3.2.2.55 LIB_EXTERN int LibStructural_getTestDetails (char ** *outMessage*, int * *nLength*)

Return Details about validation tests.

Parameters:

outMessage a pointer to a string where status information of the analysis will be returned.

nLength the length of the message.

Remarks:

free outMessage using LibStructural_freeVector

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

Examples:

examples/c/loadlabelledstoichiometry.c, examples/c/loadsbmlfromfile.c, and examples/c/loadstoichiometry.c.

3.2.2.56 LIB_EXTERN double LibStructural_getTolerance ()

Get user specified tolerance.

This function gets the tolerance used by the library to determine what value is considered as zero. Any value with absolute value smaller than this tolerance is considered as zero and will be neglected.

Returns:

the tolerance used by the library to determine a value close to zero

3.2.2.57 LIB_EXTERN int LibStructural_loadReactionNames (const char ** *reactionNames*, const int *nLength*)

Load reaction names.

This function should be used whenever labeled matrices are important as these labels will be used in labeling the structural matrices. This function sets the reaction names (ids).

Parameters:

reactionNames an array of strings of reaction names with length nLength

nLength number of elements in reactionNames

Returns:

The return value will be zero (0) when successful, and negative (-1) in case an error occurred

Remarks:

This method should only be called after LibStructural_loadStoichiometryMatrix

Examples:

examples/c/loadlabelledstoichiometry.c.

3.2.2.58 LIB_EXTERN int LibStructural_loadSBML (const char * *sSBML*, char ** *outMessage*, int * *nLength*)

Load a SBML model.

Parameters:

sSBML the SBML string to load into the library

outMessage a pointer to a string that the library can use to provide information about the loaded SBML

nLength is the length of the above message

Returns:

The return value will be zero (0) when successful, and negative (-1) in case an error occurred (invalid SBML)

3.2.2.59 LIB_EXTERN int LibStructural_loadSBMLFromFile (const char * *sFileName*, char ** *outMessage*, int * *nLength*)

Load a SBML model from the specified file.

Parameters:

sFileName the full path to the SBML file to be loaded.

outMessage a pointer to a string that the library can use to provide information about the loaded SBML

nLength is the length of the above message

Remarks:

To avoid unintentional errors be sure to pass in the full path to the SBML file.

Returns:

The return value will be zero (0) when successful, and negative (-1) in case an error occurred (invalid SBML, file not readable ...).

Examples:

examples/c/loadsbmlfromfile.c.

3.2.2.60 LIB_EXTERN int LibStructural_loadSBMLwithTests (const char * *sSBML*, char ** *outMessage*, int * *nLength*)

Load an SBML model into the library and carry out tests using the internal test suite.

Parameters:

sSBML the SBML string to load into the library

outMessage a pointer to a string that contains information about the loaded model as well as the test results of the internal test suite.

nLength is the length of the above message

Returns:

The return value will be zero (0) when successful, and negative (-1) in case an error occurred (invalid SBML)

3.2.2.61 LIB_EXTERN int LibStructural_loadSpecies (const char ** *speciesNames*, const double * *speciesValues*, const int *nLength*)

Load species names and initial values.

This function should be used whenever labeled matrices are important as these labels will be used in labeling the structural matrices. This function sets the species names (ids). It is also possible to provide an initial condition for each of the species. This will be used when calculating the conserved sums.

Parameters:

speciesNames an array of strings of species names with length nLength

speciesValues an array of real numbers of species concentrations corresponding to the speciesName with the same index

nLength number of elements in speciesNames and speciesValues

Returns:

The return value will be zero (0) when successful, and negative (-1) in case an error occurred

Remarks:

This method should only be called after LibStructural_loadStoichiometryMatrix

Examples:

examples/c/loadlabelledstoichiometry.c.

3.2.2.62 LIB_EXTERN int LibStructural_loadStoichiometryMatrix (const double ** *oMatrix*, const int *nRows*, const int *nCols*)

Load a new stoichiometry matrix.

Loads the stoichiometry matrix into the library. To analyze the stoichiometry call one of the following:

- LibStructural_analyzeWithQR,

- LibStructural_analyzeWithLU,
- LibStructural_analyzeWithLUandRunTests,
- LibStructural_analyzeWithFullyPivotedLU or
- LibStructural_analyzeWithFullyPivotedLUwithTests

Remarks:

if matrix labels are needed it is recommended to call LibStructural_loadSpecies and LibStructural_loadReactionNames after a call to this function.

Parameters:

oMatrix a pointer to a double** matrix

nRows the number of rows of the matrix

nCols the number of columns of the matrix

Returns:

The return value will be zero (0) when successful, and negative (-1) in case an error occurred

Examples:

examples/c/loadlabelledstoichiometry.c, and examples/c/loadstoichiometry.c.

3.2.2.63 LIB_EXTERN void LibStructural_setTolerance (const double *dTolerance*)

Set user specified tolerance.

This function sets the tolerance used by the library to determine what value is considered as zero. Any value with absolute value smaller than this tolerance is considered as zero and will be neglected.

Parameters:

dTolerance Sets the tolerance used by the library to determine a value close to zero

3.2.2.64 LIB_EXTERN int LibStructural_validateStructuralMatrices (int ** *outResults*, int * *outLength*)

Validates structural matrices.

Calling this method will run the internal test suite against the structural matrices those tests include:

- Test 1 : Gamma*N = 0 (Zero matrix)
- Test 2 : Rank(N) using SVD (5) is same as m0 (5)
- Test 3 : Rank(NR) using SVD (5) is same as m0 (5)
- Test 4 : Rank(NR) using QR (5) is same as m0 (5)
- Test 5 : L0 obtained with QR matches Q21*inv(Q11)
- Test 6 : N*K = 0 (Zero matrix)

Parameters:

outResults an integer vector, each element represents the result for one of the above tests (the 0th element representing the test result for test1), if the test passed the value is 1 and 0 otherwise.

outLength number of tests

Remarks:

free outResults using LibStructural_freeVector

Returns:

The return value will be zero (0) when successful, and negative (-1) in case no stoichiometry matrix was loaded beforehand or none of the analysis methods has been called yet.

3.2.3 Variable Documentation

3.2.3.1 BEGIN_C_DECLS

3.2.3.2 END_C_DECLS

Chapter 4

Example Documentation

4.1 examples/c/loadlabelledstoichiometry.c

This is an example of how to load a labeled stoichiometry matrix and read test results. The example also shows how to print the reordered stoichiometry matrix as well as the Gamma matrix.

```
#include <stdio.h>           // for printf
#include <stdlib.h>          // for malloc
#include <string.h>          // for memset
#include <libstructural.h>   // the structural analysis library

// construct simple stoichiometry matrix with labelled species and reactions
void GetMatrixFromSomeWhere(double** *oMatrix, int *nRows, int *nCols,
         char** *speciesNames, double* *initialConcentrations,
         char** *reactionNames);

// gets the reordered stoichiometry matrix from the library
void PrintReorderedStoichiometryMatrix();

// gets the gamma matrix from the librar
void PrintGammaMatrix();


int main (int argc, char** argv)
{
        int      i,j;
        int      nRows;
        int      nCols;
        double** oMatrix;
        char*    sMessage;
        char**   speciesNames;
        char**   reactionNames;
        double*  initialConcentrations;
        int      nLength;


        // get matrix to analyze from another part of the code as well
        // as species and reaction names
        GetMatrixFromSomeWhere(&oMatrix, &nRows, &nCols,
               &speciesNames, &initialConcentrations, &reactionNames);


        // load matrix into the structural analysis library
        LibStructural_loadStoichiometryMatrix (oMatrix, nRows, nCols);
        // load species names and initial concentrations
        LibStructural_loadSpecies(speciesNames, initialConcentrations, nRows);
```

```
        // load reaction names
        LibStructural_loadReactionNames(reactionNames, nCols);

        // analyze the stoichiometry matrix using the QR method
        LibStructural_analyzeWithQR( &sMessage, &nLength);

        // print model overview
        printf("%s", sMessage);

        // free the memory used by the message
        LibStructural_freeVector(sMessage);

        // obtain and print the test results
        LibStructural_getTestDetails( &sMessage, &nLength );
        printf("%s", sMessage);

        // finally free the memory used by the message
        LibStructural_freeVector(sMessage);


        // Print Reordered Stoichiometry Matrix
        PrintReorderedStoichiometryMatrix();

        // Print Gamma Matrix
        PrintGammaMatrix();


        // and free the memory used to hold the stoichiometry matrix
        for (i = 0; i < nRows; i++)
                free(oMatrix[i]);
        free(oMatrix);

        // free species names
        for (i = 0; i < nRows; i++)
                free(speciesNames[i]);
        free(speciesNames);

        // free reaction names
        for (i = 0; i < nCols; i++)
                free(reactionNames[i]);
        free(reactionNames);

        return 0;

}

void PrintReorderedStoichiometryMatrix()
{
        int      i,j;
        double** reorderedStoichiometryMatrix;
        int      reorderedNumRows;
        int      reorderedNumCols;
        char**   reorderedCols;
        char**   reorderedRows;

        printf("\nReordered Stoichiometry Matrix");

        // get Reordered Stoichiometry Matrix
        LibStructural_getReorderedStoichiometryMatrix
                (&reorderedStoichiometryMatrix, &reorderedNumRows, &reorderedNumCols);
        LibStructural_getReorderedStoichiometryMatrixLabels
                (&reorderedRows, &reorderedNumRows, &reorderedCols, &reorderedNumCols);

        // print Reordered stoichiometry matrix:
        printf("\n\t");
        for (i = 0; i < reorderedNumCols; i++)
                printf("%s\t", reorderedCols[i]);
```

```
        printf("\n");

        for (i = 0; i < reorderedNumRows; i++)
        {
                printf("%s\t", reorderedRows[i]);
                for (j = 0; j < reorderedNumCols; j++)
                        printf ("%2.1lf\t", reorderedStoichiometryMatrix[i][j]);
                printf("\n");
        }

        // free reordered stoichiometry matrix and labels
        LibStructural_freeMatrix((void**)reorderedStoichiometryMatrix, reorderedNumRows);
        LibStructural_freeMatrix((void**)reorderedCols, reorderedNumCols);
        LibStructural_freeMatrix((void**)reorderedRows, reorderedNumRows);

        printf("\n");
}

void PrintGammaMatrix()
{
        int      i,j;
        double** gammaMatrix;
        int      gammaNumRows;
        int      gammaNumCols;
        char**   gammaCols;
        char**   gammaRows;


        printf("\nGamma Matrix");

        // get Gamma Matrix and labels
        LibStructural_getGammaMatrix
                (&gammaMatrix, &gammaNumRows, &gammaNumCols);
        LibStructural_getGammaMatrixLabels
                (&gammaRows, &gammaNumRows, &gammaCols, &gammaNumCols);

        // print gamma matrix:
        printf("\n\t");
        for (i = 0; i < gammaNumCols; i++)
                printf("%s\t", gammaCols[i]);
        printf("\n");

        for (i = 0; i < gammaNumRows; i++)
        {
                printf("%s\t", gammaRows[i]);
                for (j = 0; j < gammaNumCols; j++)
                        printf ("%2.1lf\t", gammaMatrix[i][j]);
                printf("\n");
        }

        // free gamma stoichiometry matrix and labels
        LibStructural_freeMatrix((void**)gammaMatrix, gammaNumRows);
        LibStructural_freeMatrix((void**)gammaCols, gammaNumCols);
        LibStructural_freeMatrix((void**)gammaRows, gammaNumRows);

        printf("\n");

}


void GetMatrixFromSomeWhere(double** *oMatrix, int *nRows, int *nCols,
         char** *speciesNames, double* *initialConcentrations,
         char** *reactionNames)
{
        int numCols, numRows, i;
        numRows = 4; numCols = 3;
```

```
        // initialize memory needed for the stoichiometry matrix
        *oMatrix = (double**)malloc(sizeof(double*)*numRows);
        memset(*oMatrix, 0, sizeof(double*)*numRows);

        for (i = 0; i < numRows; i ++)
        {
                (*oMatrix)[i] = (double*)malloc(sizeof(double)*numCols);
                memset((*oMatrix)[i], 0, sizeof(double)*numCols);
        }

        // initialize memory needed for speciesNames
        (*speciesNames) = (char**)malloc(sizeof(char*)*numRows);
        memset(*speciesNames, 0, sizeof(char*)*numRows);

        *initialConcentrations = (double*)malloc(sizeof(double)*numRows);
        memset(*initialConcentrations, 0, sizeof(double)*numRows);

        // initialize memory needed for reactionNames
        (*reactionNames) = (char**)malloc(sizeof(char*)*numCols);
        memset(*reactionNames, 0, sizeof(char*)*numCols);

        // set non zero entries of the stoichiometry matrix
        (*oMatrix)[0][1] = -1.0;        (*oMatrix)[0][2] =  1.0;      // ES
        (*oMatrix)[1][0] =  1.0;        (*oMatrix)[1][2] = -1.0;      // S2
        (*oMatrix)[2][0] = -1.0;        (*oMatrix)[2][1] =  1.0;      // S1
        (*oMatrix)[3][1] =  1.0;        (*oMatrix)[3][2] = -1.0;      // E

        // set species names
        (*speciesNames)[0] = strdup("S2");  (*speciesNames)[1] = strdup("ES");
        (*speciesNames)[2] = strdup("S1");  (*speciesNames)[3] = strdup("E");

        // set reaction names
        (*reactionNames)[0] = strdup("J1"); (*reactionNames)[1] = strdup("J2");
        (*reactionNames)[2] = strdup("J3");

        // be sure to return number of rows and columns
        *nRows = numRows;
        *nCols = numCols;
}



//The above returns the following output:
//
//--------------------------------------------------------------------------------
//--------------------------------------------------------------------------------
//STRUCTURAL ANALYSIS MODULE : Results
//--------------------------------------------------------------------------------
//--------------------------------------------------------------------------------
//Size of Stochiometric Matrix: 4 x 3 (Rank is  2)
//Nonzero entries in Stochiometric Matrix: 8  (66.6667% full)
//
//Independent Species (2) :
//S2, ES
//
//Dependent Species (2) :
//S1, E
//
//L0 : There are 2 dependencies. L0 is a 2x2 matrix.
//
//Conserved Entities
//1:  + S2 + ES + S1
//2:  + S2 + E
//--------------------------------------------------------------------------------
//--------------------------------------------------------------------------------
//Developed by the Computational Systems Biology Group at Keck Graduate Institute
//and the Saurolab at the Bioengineering Departmant at  University of Washington.
```

```
//Contact : Frank T. Bergmann (fbergman@u.washington.edu) or Herbert M. Sauro.
//
//(previous authors) Ravishankar Rao Vallabhajosyula
//--------------------------------------------------------------------------------
//--------------------------------------------------------------------------------
//
//Testing Validity of Conservation Laws.
//
//Passed Test 1 : Gamma*N = 0 (Zero matrix)
//Passed Test 2 : Rank(N) using SVD (2) is same as m0 (2)
//Passed Test 3 : Rank(NR) using SVD (2) is same as m0 (2)
//Passed Test 4 : Rank(NR) using QR (2) is same as m0 (2)
//Passed Test 5 : L0 obtained with QR matches Q21*inv(Q11)
//Passed Test 6 : N*K = 0 (Zero matrix)
//
//Reordered Stoichiometry Matrix
//J1      J2      J3
//S2      0.0     -1.0    1.0
//ES      1.0     0.0     -1.0
//S1      -1.0    1.0     0.0
//E       0.0     1.0     -1.0
//
//
//Gamma Matrix
//S2      ES      S1      E
//0       1.0     1.0     1.0     0.0
//1       1.0     -0.0    0.0     1.0
//
```

4.2 examples/c/loadsbmlfromfile.c

This is an example of how to load a SBML file and print structural analysis test results.

```
#include <stdio.h>           // for printf
#include <stdlib.h>          // for malloc
#include <string.h>          // for memset
#include <libstructural.h>   // the structural analysis library


int main (int argc, char** argv)
{
        int             result;
        char*   message;
        int             length;

        if (argc < 2)
        {
                printf("please provide a filename as argument");
                return -1;
        }

        // load the sbml file and check the argument
        result = LibStructural_loadSBMLFromFile(argv[1], &message, &length);
        if (result != 0)
        {
                printf("the SBML file %s could not be loaded.", argv[1]);
                return -1;
        }

        // print model overview
        printf("%s", message);

        // free the memory used by the message
        LibStructural_freeVector(message);

        // obtain and print the test results
        LibStructural_getTestDetails( &message, &length );
        printf("%s", message);

        // finally free the memory used by the message
        LibStructural_freeVector(message);


        return 0;
}

// Output for model BorisEJB.xml(available with SBW distribution) passed in
//
//-------------------------------------------------------------------------------
//-------------------------------------------------------------------------------
//STRUCTURAL ANALYSIS MODULE : Results
//-------------------------------------------------------------------------------
//-------------------------------------------------------------------------------
//Size of Stochiometric Matrix: 8 x 10 (Rank is  5)
//Nonzero entries in Stochiometric Matrix: 20  (25% full)
//
//Independent Species (5) :
//MKK_P, MAPK_P, MKKK, MKK, MAPK
//
//Dependent Species (3) :
//MKK_PP, MKKK_P, MAPK_PP
//
//L0 : There are 3 dependencies. L0 is a 3x5 matrix.
//
//Conserved Entities
```

```
//1:  + MKK_P + MKK + MKK_PP
//2:  + MKKK + MKKK_P
//3:  + MAPK_P + MAPK + MAPK_PP
//----------------------------------------------------------------------------
//----------------------------------------------------------------------------
//Developed by the Computational Systems Biology Group at Keck Graduate Institute
//and the Saurolab at the Bioengineering Departmant at  University of Washington.
//Contact : Frank T. Bergmann (fbergman@u.washington.edu) or Herbert M. Sauro.
//
//(previous authors) Ravishankar Rao Vallabhajosyula
//----------------------------------------------------------------------------
//----------------------------------------------------------------------------
//
//Testing Validity of Conservation Laws.
//
//Passed Test 1 : Gamma*N = 0 (Zero matrix)
//Passed Test 2 : Rank(N) using SVD (5) is same as m0 (5)
//Passed Test 3 : Rank(NR) using SVD (5) is same as m0 (5)
//Passed Test 4 : Rank(NR) using QR (5) is same as m0 (5)
//Passed Test 5 : L0 obtained with QR matches Q21*inv(Q11)
//Passed Test 6 : N*K = 0 (Zero matrix)
```

4.3 examples/c/loadstoichiometry.c

This is an example of how to load a (unlabeled) stoichiometry matrix and read test details.

```
#include <stdio.h>           // for printf
#include <stdlib.h>          // for malloc
#include <string.h>          // for memset
#include <libstructural.h>   // the structural analysis library

// construct simple stoichiometry matrix
void GetMatrixFromSomeWhere(double** *oMatrix, int *nRows, int *nCols)
{
        int numCols, numRows, i;
        numRows = 4; numCols = 3;

        // initialize memory needed
        *oMatrix = (double**)malloc(sizeof(double*)*numRows);
        memset(*oMatrix, 0, sizeof(double*)*numRows);

        for (i = 0; i < numRows; i ++)
        {
                (*oMatrix)[i] = (double*)malloc(sizeof(double)*numCols);
                memset((*oMatrix)[i], 0, sizeof(double)*numCols);
        }

        // set non zero entries of the stoichiometry matrix
        (*oMatrix)[0][1] = -1.0;    (*oMatrix)[0][2] =  1.0;    // ES
        (*oMatrix)[1][0] =  1.0;    (*oMatrix)[1][2] = -1.0;    // S2
        (*oMatrix)[2][0] = -1.0;    (*oMatrix)[2][1] =  1.0;    // S1
        (*oMatrix)[3][1] =  1.0;    (*oMatrix)[3][2] = -1.0;    // E


        // be sure to return number of rows and columns
        *nRows = numRows;
        *nCols = numCols;
}

int main (int argc, char** argv)
{
        int      i;
        int      nRows;
        int      nCols;
        double** oMatrix;
        char*    sMessage;
        int      nLength;


        // get matrix to analyze from another part of the code
        GetMatrixFromSomeWhere(&oMatrix, &nRows, &nCols);

        // load it into the structural analysis library
        LibStructural_loadStoichiometryMatrix (oMatrix, nRows, nCols);

        // analyze the stoichiometry matrix using the QR method
        LibStructural_analyzeWithQR( &sMessage, &nLength);

        // print model overview
        printf("%s", sMessage);

        // free the memory used by the message
        LibStructural_freeVector(sMessage);

        // obtain and print the test results
        LibStructural_getTestDetails( &sMessage, &nLength );
        printf("%s", sMessage);
```

```
        // finally free the memory used by the message
        LibStructural_freeVector(sMessage);

        // and free the memory used to hold the stoichiometry matrix
        for (i = 0; i < nRows; i++)
                free(oMatrix[i]);
        free(oMatrix);

        return 0;

}

//The program above returns the following output:
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
//STRUCTURAL ANALYSIS MODULE : Results
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
//Size of Stochiometric Matrix: 4 x 3 (Rank is  2)
//Nonzero entries in Stochiometric Matrix: 8  (66.6667% full)
//
//Independent Species (2) :
//0, 1
//
//Dependent Species (2) :
//2, 3
//
//L0 : There are 2 dependencies. L0 is a 2x2 matrix.
//
//Conserved Entities
//1:  + 0 + 1 + 2
//2:  + 0 + 3
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
//Developed by the Computational Systems Biology Group at Keck Graduate Institute
//and the Saurolab at the Bioengineering Departmant at  University of Washington.
//Contact : Frank T. Bergmann (fbergman@u.washington.edu) or Herbert M. Sauro.
//
//(previous authors) Ravishankar Rao Vallabhajosyula
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
//
//Testing Validity of Conservation Laws.
//
//Passed Test 1 : Gamma*N = 0 (Zero matrix)
//Passed Test 2 : Rank(N) using SVD (2) is same as m0 (2)
//Passed Test 3 : Rank(NR) using SVD (2) is same as m0 (2)
//Passed Test 4 : Rank(NR) using QR (2) is same as m0 (2)
//Passed Test 5 : L0 obtained with QR matches Q21*inv(Q11)
//Passed Test 6 : N*K = 0 (Zero matrix)
```

Index