Algorithm for Counting Cycles in C++

Generated by Doxygen 1.8.13

Contents

1 File Index **1**

1.1 File List . . . 1

2 File Documentation **3**

2.1 cycleCount.cpp File Reference . . . 3

2.1.1 Macro Definition Documentation . . . 4

2.1.1.1 EIGEN_MAX_ITERATION . . . 4

2.1.2 Function Documentation . . . 4

2.1.2.1 countTrue() . . . 4

2.1.2.2 cycleCount() . . . 5

2.1.2.3 cycleCountFixedVertex() . . . 5

2.1.2.4 matrixAddition() . . . 6

2.1.2.5 matrixMultiplication() . . . 6

2.1.2.6 pathsCount() . . . 6

2.1.2.7 primeCount() . . . 7

2.1.2.8 primeCountDirected() . . . 8

2.1.2.9 primeCountFixedVertex() . . . 8

2.1.2.10 primeCountPath() . . . 9

2.1.2.11 primeCountUndirected() . . . 9

2.1.2.12 recursiveSubgraphs() . . . 10

2.1.2.13 recursiveSubgraphsFixedVertex() . . . 11

2.1.2.14 recursiveSubgraphsPaths() . . . 11

2.1.2.15 restrictedAdjacencyMatrix() . . . 12

2.1.2.16 squareMatrixMultiplication() . . . 12

2.1.2.17 subgraphAdjacencyArray() . . . 13

2.1.2.18 subgraphAdjacencyMatrix() . . . 13

2.1.2.19 sum() . . . 14

2.1.2.20 trace() . . . 14

2.2 eigenvalues.cpp File Reference . . . 14

2.2.1 Function Documentation . . . 15

2.2.1.1 computeEigenvalues() . . . 15

2.2.1.2 getDiagonalDoublePrecisionMatrix() . . . 16

2.2.1.3 identityDoublePrecisionMatrix() . . . 16

Index **17**

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

cycleCount.cpp 3
eigenvalues.cpp 14

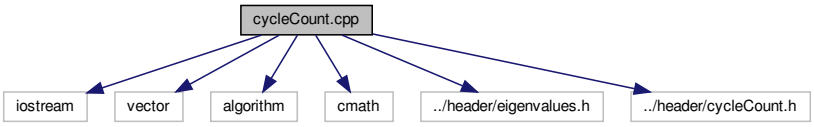
Chapter 2

File Documentation

2.1 cycleCount.cpp File Reference

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <cmath>
#include "../header/eigenvalues.h"
#include "../header/cycleCount.h"
```

Include dependency graph for cycleCount.cpp:



Macros

- #define EIGEN_MAX_ITERATION 10000

Functions

- std::vector< std::vector< std::vector< double > > > pathsCount (std::vector< std::vector< double >> &adjacencyMatrix, unsigned long length, bool directed)
- std::vector< std::vector< std::vector< double > > > recursiveSubgraphsPaths (std::vector< std::vector< double >> adjacencyMatrix, unsigned long length, std::vector< int > subgraph, std::vector< bool > allowedVertex, std::vector< std::vector< std::vector< double >>> primes, std::vector< bool > neighbourhood, bool directed)
- std::vector< std::vector< std::vector< double > > > primeCountPath (const std::vector< std::vector< double >> &adjacencyMatrix, unsigned long length, std::vector< int > subgraph, unsigned long neighbours↩Number, std::vector< std::vector< std::vector< double >>> primes)
- std::vector< double > cycleCount (std::vector< std::vector< double >> &adjacencyMatrix, unsigned long length, bool directed)

- std::vector< double > recursiveSubgraphs (std::vector< std::vector< double >> adjacencyMatrix, unsigned long length, std::vector< int > subgraph, std::vector< bool > allowedVertex, std::vector< double > primes, std::vector< bool > neighbourhood, bool directed)
- std::vector< double > primeCount (const std::vector< std::vector< double >> &adjacencyMatrix, unsigned long length, std::vector< int > subgraph, unsigned long neighboursNumber, std::vector< double > primes, bool directed)
- std::vector< double > primeCountUndirected (const std::vector< std::vector< double >> &adjacencyMatrix, unsigned long length, std::vector< int > subgraph, unsigned long neighboursNumber, std::vector< double > primes)
- std::vector< double > primeCountDirected (const std::vector< std::vector< double >> &adjacencyMatrix, unsigned long length, std::vector< int > subgraph, unsigned long neighboursNumber, std::vector< double > primes)
- std::vector< double > cycleCountFixedVertex (std::vector< std::vector< double >> &adjacencyMatrix, int vertex, unsigned long length)
- std::vector< double > recursiveSubgraphsFixedVertex (std::vector< std::vector< double >> adjacency↩ Matrix, int vertex, unsigned long length, std::vector< int > subgraph, std::vector< bool > allowedVertex, std::vector< double > primes, std::vector< bool > neighbourhood)
- std::vector< double > primeCountFixedVertex (const std::vector< std::vector< double >> &adjacency↩ Matrix, int vertex, unsigned long length, std::vector< int > subgraph, unsigned long neighboursNumber, std::vector< double > primes)
- std::vector< std::vector< double > > subgraphAdjacencyMatrix (const std::vector< std::vector< double >> &adjacencyMatrix, const std::vector< int > &subgraph)
- double * subgraphAdjacencyArray (const std::vector< std::vector< double >> &adjacencyMatrix, const std::vector< int > &subgraph)
- std::vector< std::vector< double > > restrictedAdjacencyMatrix (const std::vector< std::vector< double >> &adjacencyMatrix, const std::vector< int > &subgraph)
- std::vector< std::vector< double > > matrixMultiplication (const double x, const std::vector< std::vector< double >> &M)
- std::vector< std::vector< double > > matrixAddition (const std::vector< std::vector< double >> &A, const std::vector< std::vector< double >> &B)
- std::vector< std::vector< double > > squareMatrixMultiplication (const std::vector< std::vector< double >> &A, const std::vector< std::vector< double >> &B)
- long unsigned countTrue (const std::vector< bool > &vector)
- double sum (double *array, long unsigned n)
- double trace (const std::vector< std::vector< double >> &M)

2.1.1 Macro Definition Documentation

2.1.1.1 EIGEN_MAX_ITERATION

```
#define EIGEN_MAX_ITERATION 10000
```

2.1.2 Function Documentation

2.1.2.1 countTrue()

```
long unsigned countTrue (
            const std::vector< bool > & vector )
```

Count number of true entry in a vector

Parameters

vector	the boolean vector

Returns

true values counter

2.1.2.2 cycleCount()

```
std::vector<double> cycleCount (
            std::vector< std::vector< double >> & adjacencyMatrix,
            unsigned long length,
            bool directed )
```

Counts all simple cycle of length up to length included on adjacencyMatrix

Parameters

adjacencyMatrix	Undirected adjacency matrix of the graph G, this matrix may be weighted.
length	maximum length of the simple cyles to be counted
directed	true if the graph is directed, false if not

Returns

an array whose entry i is the number of simple cycles of length i in the graph

Remarks

Originally designed by P.-L. Giscard, N. Kriege, R. C. Wilson, July 2017

2.1.2.3 cycleCountFixedVertex()

```
std::vector<double> cycleCountFixedVertex (
            std::vector< std::vector< double >> & adjacencyMatrix,
            int vertex,
            unsigned long length )
```

Counts all simple cycle of length up to length passing through a specified vertex

Parameters

adjacencyMatrix	Undirected adjacency matrix of the graph G, this matrix may be weighted.
vertex	the fixed vertex index
length	maximum length of the simple cycles to be counted

Returns

an array whose entry i is the number of simple cycles of length i passing through the fixed vertex

Remarks

Originally designed by P.-L. Giscard, N. Kriege, R. C. Wilson, July 2017

2.1.2.4 matrixAddition()

```
std::vector<std::vector<double> > matrixAddition (
            const std::vector< std::vector< double >> & A,
            const std::vector< std::vector< double >> & B )
```

Parameters

A	first square matrix
B	second square matrix

Returns

The matrix (A + B)

2.1.2.5 matrixMultiplication()

```
std::vector<std::vector<double> > matrixMultiplication (
            const double x,
            const std::vector< std::vector< double >> & M )
```

Parameters

x	double value for multiplication
M	square matrix

Returns

The matrix (x $*$ M)

2.1.2.6 pathsCount()

```
std::vector<std::vector<std::vector<double> > > pathsCount (
            std::vector< std::vector< double >> & adjacencyMatrix,
```

```
            unsigned long length,
            bool directed )
```

Counts all simple paths and cycles of length up to length included on adjacencyMatrix

Parameters

adjacencyMatrix	Undirected adjacency matrix of the graph G, this matrix may be weighted.
length	maximum length of the simple path to be counted
directed	true if the graph is directed, false if not

Returns

an array whose entry [i][x][y] is the number of simple paths of length i from x to yin the graph

Remarks

Originally designed by P.-L. Giscard, N. Kriege, R. C. Wilson, July 2017

2.1.2.7 primeCount()

```
std::vector<double> primeCount (
            const std::vector< std::vector< double >> & adjacencyMatrix,
            unsigned long length,
            std::vector< int > subgraph,
            unsigned long neighboursNumber,
            std::vector< double > primes,
            bool directed )
```

Simple handler to split between directed and undirected function calls.

Parameters

adjacencyMatrix	adjacency matrix of the graph, must be symmetric
length	maximum subgraph size, an integer
subgraph	current subgraph, a list of vertices, further vertices are added to this list
neighboursNumber	number of neighbours from the induced subgraph to the graph
primes	current cycles count array
directed	true if the graph is directed, false if not

Returns

updated cycles count array

Remarks

Originally designed by P.-L. Giscard, N. Kriege, R. C. Wilson, July 2017

2.1.2.8 primeCountDirected()

```
std::vector<double> primeCountDirected (
            const std::vector< std::vector< double >> & adjacencyMatrix,
            unsigned long length,
            std::vector< int > subgraph,
            unsigned long neighboursNumber,
            std::vector< double > primes )  [protected]
```

Calculates the contribution to the combinatorial sieve of a given subgraph. This function is an implementation of the equation extracting prime numbers from connected induced subgraphs. do not use externally, use primeCount(_, true) instead

Parameters

adjacencyMatrix	adjacency matrix of the graph, must be symmetric
length	maximum subgraph size, an integer
subgraph	current subgraph, a list of vertices, further vertices are added to this list
neighboursNumber	number of neighbours from the induced subgraph to the graph
primes	current cycles count array

Returns

updated cycles count array

Remarks

Originally designed by P.-L. Giscard, N. Kriege, R. C. Wilson, July 2017

2.1.2.9 primeCountFixedVertex()

```
std::vector<double> primeCountFixedVertex (
            const std::vector< std::vector< double >> & adjacencyMatrix,
            int vertex,
            unsigned long length,
            std::vector< int > subgraph,
            unsigned long neighboursNumber,
            std::vector< double > primes )
```

Calculates the contribution to the combinatorial sieve of a given subgraph. This function is an implementation of the equation extracting prime numbers from connected induced subgraphs for a fixed vertex.

Parameters

adjacencyMatrix	adjacency matrix of the graph
length	maximum subgraph size, an integer
vertex	the fixed vertex
subgraph	current subgraph, a list of vertices, further vertices are added to this list
neighboursNumber	number of neighbours from the induced subgraph to the graph
primes	current cycles count array

Returns

updated cycles count array

Remarks

Originally designed by P.-L. Giscard, N. Kriege, R. C. Wilson, July 2017

2.1.2.10 primeCountPath()

```
std::vector<std::vector<std::vector<double> > > primeCountPath (
            const std::vector< std::vector< double >> & adjacencyMatrix,
            unsigned long length,
            std::vector< int > subgraph,
            unsigned long neighboursNumber,
            std::vector< std::vector< std::vector< double >>> primes )
```

Calculates the contribution to the combinatorial sieve of a given subgraph. This function is an implementation of the equation extracting prime numbers from connected induced subgraphs.

Parameters

adjacencyMatrix	adjacency matrix of the graph, must be symmetric
length	maximum subgraph size, an integer
subgraph	current subgraph, a list of vertices, further vertices are added to this list
neighboursNumber	number of neighbours from the induced subgraph to the graph
primes	current paths count array

Returns

updated paths count array

Remarks

Originally designed by P.-L. Giscard, N. Kriege, R. C. Wilson, July 2017

2.1.2.11 primeCountUndirected()

```
std::vector<double> primeCountUndirected (
            const std::vector< std::vector< double >> & adjacencyMatrix,
            unsigned long length,
            std::vector< int > subgraph,
            unsigned long neighboursNumber,
            std::vector< double > primes )  [protected]
```

Calculates the contribution to the combinatorial sieve of a given subgraph. This function is an implementation of the equation extracting prime numbers from connected induced subgraphs. It use eigenvalues to compute trace to the power recursively. do not use externally, use primeCount(_, false) instead

Parameters

adjacencyMatrix	adjacency matrix of the graph, must be symmetric
length	maximum subgraph size, an integer
subgraph	current subgraph, a list of vertices, further vertices are added to this list
neighboursNumber	number of neighbours from the induced subgraph to the graph
primes	current cycles count array

Returns

updated cycles count array

Remarks

Originally designed by P.-L. Giscard, N. Kriege, R. C. Wilson, July 2017

2.1.2.12 recursiveSubgraphs()

```
std::vector<double> recursiveSubgraphs (
            std::vector< std::vector< double >> adjacencyMatrix,
            unsigned long length,
            std::vector< int > subgraph,
            std::vector< bool > allowedVertex,
            std::vector< double > primes,
            std::vector< bool > neighbourhood,
            bool directed )
```

Finds all the connected induced subgraphs of size up "length" of a graph known through its adjacency matrix "A" and containing the subgraph "Subgraph"

Parameters

adjacencyMatrix	adjacency matrix of the graph, preferably sparse
length	maximum subgraph size, an integer
subgraph	current subgraph, a list of vertices, further vertices are added to this list
allowedVertex	indicator vector of pruned vertices that may be considered for addition to the current subgraph to form a larger one
primes	list regrouping the contribution of all the subgraphs found so far
neighbourhood	indicator vector of the vertices that are contained in the current subgraph or reachable via one edge
directed	true if the graph is directed, false if not

Returns

primes with one more subgraph

Remarks

Originally designed by P.-L. Giscard, N. Kriege, R. C. Wilson, July 2017

2.1.2.13 recursiveSubgraphsFixedVertex()

```
std::vector<double> recursiveSubgraphsFixedVertex (
            std::vector< std::vector< double >> adjacencyMatrix,
            int vertex,
            unsigned long length,
            std::vector< int > subgraph,
            std::vector< bool > allowedVertex,
            std::vector< double > primes,
            std::vector< bool > neighbourhood )
```

Finds all the connected induced subgraphs of size up "length" of a graph known through its adjacency matrix "A" and containing the subgraph "Subgraph"

Parameters

adjacencyMatrix	adjacency matrix of the graph, must be symmetric
vertex	the fixed vertex index
length	maximum subgraph size, an integer
subgraph	current subgraph, a list of vertices, further vertices are added to this list
allowedVertex	indicator vector of pruned vertices that may be considered for addition to the current subgraph to form a larger one
primes	list regrouping the contribution of all the subgraphs found so far
neighbourhood	indicator vector of the vertices that are contained in the current subgraph or reachable via one edge

Returns

updated cycles count list

Remarks

Originally designed by P.-L. Giscard, N. Kriege, R. C. Wilson, July 2017

2.1.2.14 recursiveSubgraphsPaths()

```
std::vector<std::vector<std::vector<double> > > recursiveSubgraphsPaths (
            std::vector< std::vector< double >> adjacencyMatrix,
            unsigned long length,
            std::vector< int > subgraph,
            std::vector< bool > allowedVertex,
            std::vector< std::vector< std::vector< double >>> primes,
            std::vector< bool > neighbourhood,
            bool directed )
```

Finds all the connected induced subgraphs of size up "length" of a graph known through its adjacency matrix "A" and containing the subgraph "Subgraph"

Parameters

adjacencyMatrix	adjacency matrix of the graph, preferably sparse
length	maximum subgraph size, an integer
subgraph	current subgraph, a list of vertices, further vertices are added to this list
allowedVertex	indicator vector of pruned vertices that may be considered for addition to the current subgraph to form a larger one
primes	list regrouping the contribution of all the subgraphs found so far
neighbourhood	indicator vector of the vertices that are contained in the current subgraph or reachable via one edge
directed	true if the graph is directed, false if not

Returns

primes with one more subgraph

Remarks

Originally designed by P.-L. Giscard, N. Kriege, R. C. Wilson, July 2017

2.1.2.15 restrictedAdjacencyMatrix()

```
std::vector<std::vector<double> > restrictedAdjacencyMatrix (
            const std::vector< std::vector< double >> & adjacencyMatrix,
            const std::vector< int > & subgraph )
```

The adjacency matrix restricted to subgraph

Parameters

adjacencyMatrix	the original adjacency matrix
subgraph	list of vertexes

Returns

adjacencyMatrix with (entry[i][j] = 0) if i or j not in subgraph

2.1.2.16 squareMatrixMultiplication()

```
std::vector<std::vector<double> > squareMatrixMultiplication (
            const std::vector< std::vector< double >> & A,
            const std::vector< std::vector< double >> & B )
```

Parameters

A	first square matrix
B	second square matrix

Returns

The matrix (A ∗ B)

2.1.2.17 subgraphAdjacencyArray()

```
double* subgraphAdjacencyArray (
            const std::vector< std::vector< double >> & adjacencyMatrix,
            const std::vector< int > & subgraph )
```

Get a flatten adjacency matrix induced by a vertex list

Parameters

adjacencyMatrix	original adjacency matrix
subgraph	list of index for the induced adjacency matrix

Returns

the adjacency matrix with only specified indexes

2.1.2.18 subgraphAdjacencyMatrix()

```
std::vector<std::vector<double> > subgraphAdjacencyMatrix (
            const std::vector< std::vector< double >> & adjacencyMatrix,
            const std::vector< int > & subgraph )
```

Get the adjacency matrix induced by a vertex list

Parameters

adjacencyMatrix	original adjacency matrix
subgraph	list of index for the induced adjacency matrix

Returns

the adjacency matrix with only specified indexes

2.1.2.19 sum()

```
double sum (
            double * array,
            long unsigned n )
```

Parameters

array	an array of double
n	an index

Returns

Sum of the n first values of array

2.1.2.20 trace()

```
double trace (
            const std::vector< std::vector< double >> & M )
```

Parameters

M	an array of double

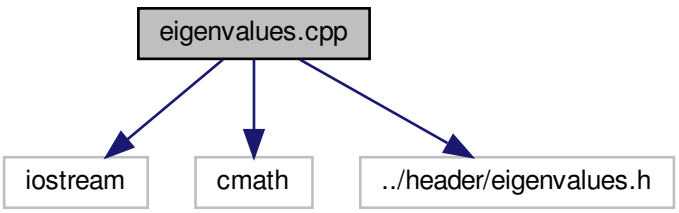
Returns

Trace of the matrix

2.2 eigenvalues.cpp File Reference

```
#include <iostream>
#include <cmath>
#include "../header/eigenvalues.h"
```

Include dependency graph for eigenvalues.cpp:



Functions

- void computeEigenvalues (int n, double ∗A, int it_max, double ∗v, double ∗d, int &it_num, int &rot_num)
- void getDiagonalDoublePrecisionMatrix (int n, double ∗A, double ∗v)
- void identityDoublePrecisionMatrix (int n, double ∗A)

2.2.1 Function Documentation

2.2.1.1 computeEigenvalues()

```
void computeEigenvalues (
            int n,
            double * A,
            int it_max,
            double * v,
            double * d,
            int & it_num,
            int & rot_num )
```

Compute the Jacobi eigenvalue iteration

This function computes the eigenvalues and eigenvectors of areal symmetric matrix, using Rutishauser's modifications of the classical Jacobi rotation method with threshold pivoting

Parameters

n	the size of the matrix
A	the matrix, which must be square, real and symmetric
it_max	maximum number of iterations
v	matrix of eigenvectors
d	eigenvalues, in descending order
it_num	total number of iterations
rot_num	total number of rotations

Author

Modified by Areski Guilhem Himeur original code by John Burkardt

Copyright

GNU LGPLv3

2.2.1.2 getDiagonalDoublePrecisionMatrix()

```
void getDiagonalDoublePrecisionMatrix (
             int n,
             double * A,
             double * v )
```

Gets the value of the diagonal of an R8MAT.

An R8MAT is A doubly dimensioned array of R8 values, stored as A vector in column-major order.

Parameters

n	the number of rows and columns of the matrix
A	the N by N matrix
v	the diagonal entries of the matrix [output]

Author

Modified by Areski Guilhem Himeur original code by John Burkardt

Copyright

GNU LGPLv3

2.2.1.3 identityDoublePrecisionMatrix()

```
void identityDoublePrecisionMatrix (
             int n,
             double * A )
```

Sets the square matrix A to the identity.

An R8MAT is A doubly dimensioned array of R8 values, stored as A vector in column-major order.

Parameters

n	the order of A
A	N by N identity matrix

Author

Modified by Areski Guilhem Himeur original code by John Burkardt

Copyright

GNU LGPLv3

Index

computeEigenvalues
    eigenvalues.cpp, 15
countTrue
    cycleCount.cpp, 4
cycleCount
    cycleCount.cpp, 5
cycleCount.cpp, 3
    countTrue, 4
    cycleCount, 5
    cycleCountFixedVertex, 5
    EIGEN_MAX_ITERATION, 4
    matrixAddition, 6
    matrixMultiplication, 6
    pathsCount, 6
    primeCount, 7
    primeCountDirected, 7
    primeCountFixedVertex, 8
    primeCountPath, 9
    primeCountUndirected, 9
    recursiveSubgraphs, 10
    recursiveSubgraphsFixedVertex, 11
    recursiveSubgraphsPaths, 11
    restrictedAdjacencyMatrix, 12
    squareMatrixMultiplication, 12
    subgraphAdjacencyArray, 13
    subgraphAdjacencyMatrix, 13
    sum, 13
    trace, 14
cycleCountFixedVertex
    cycleCount.cpp, 5

EIGEN_MAX_ITERATION
    cycleCount.cpp, 4
eigenvalues.cpp, 14
    computeEigenvalues, 15
    getDiagonalDoublePrecisionMatrix, 15
    identityDoublePrecisionMatrix, 16

getDiagonalDoublePrecisionMatrix
    eigenvalues.cpp, 15

identityDoublePrecisionMatrix
    eigenvalues.cpp, 16

matrixAddition
    cycleCount.cpp, 6
matrixMultiplication
    cycleCount.cpp, 6

pathsCount
    cycleCount.cpp, 6
primeCount
    cycleCount.cpp, 7
primeCountDirected
    cycleCount.cpp, 7
primeCountFixedVertex
    cycleCount.cpp, 8
primeCountPath
    cycleCount.cpp, 9
primeCountUndirected
    cycleCount.cpp, 9

recursiveSubgraphs
    cycleCount.cpp, 10
recursiveSubgraphsFixedVertex
    cycleCount.cpp, 11
recursiveSubgraphsPaths
    cycleCount.cpp, 11
restrictedAdjacencyMatrix
    cycleCount.cpp, 12

squareMatrixMultiplication
    cycleCount.cpp, 12
subgraphAdjacencyArray
    cycleCount.cpp, 13
subgraphAdjacencyMatrix
    cycleCount.cpp, 13
sum
    cycleCount.cpp, 13

trace
    cycleCount.cpp, 14