

CS471 Project 1-Benchmark Functions

1

Generated by Doxygen 1.8.15

| | |
|---|----------|
| 1 Class Index | 1 |
| 1.1 Class List | 1 |
| 2 File Index | 3 |
| 2.1 File List | 3 |
| 3 Class Documentation | 5 |
| 3.1 ProcessFunctions::FunctionAnalysis Struct Reference | 5 |
| 3.1.1 Detailed Description | 5 |
| 3.1.2 Member Data Documentation | 5 |
| 3.1.2.1 avgFuntionFitness | 5 |
| 3.1.2.2 functionIDs | 6 |
| 3.1.2.3 header | 6 |
| 3.1.2.4 medianFunctionFitness | 6 |
| 3.1.2.5 processTimes | 6 |
| 3.1.2.6 ranges | 6 |
| 3.2 ProcessFunctions::FunctionData Struct Reference | 6 |
| 3.2.1 Detailed Description | 7 |
| 3.2.2 Member Data Documentation | 7 |
| 3.2.2.1 fitness | 7 |
| 3.2.2.2 functionID | 7 |
| 3.2.2.3 functionMatrix | 7 |
| 3.2.2.4 timeToExecute | 7 |
| 3.3 ProcessFunctions Class Reference | 7 |
| 3.3.1 Detailed Description | 9 |
| 3.3.2 Member Function Documentation | 9 |
| 3.3.2.1 analyzeAllFunctionResults() | 9 |
| 3.3.2.2 analyzeFunctionResults() | 9 |
| 3.3.2.3 calculateAvgFitness() | 10 |
| 3.3.2.4 calculateFitnessOfAllMatrices() | 10 |
| 3.3.2.5 calculateFitnessOfMatrix() | 10 |
| 3.3.2.6 calculateFitnessOfVector() | 10 |
| 3.3.2.7 constructMatrix() [1/2] | 11 |
| 3.3.2.8 constructMatrix() [2/2] | 11 |
| 3.3.2.9 generateMatrix() | 12 |
| 3.3.2.10 getMaxFitness() | 12 |
| 3.3.2.11 getMinFitness() | 12 |
| 3.3.2.12 getNumOfDimensions() | 13 |
| 3.3.2.13 printAllFunctionIDs() | 13 |
| 3.3.2.14 printFunctionResults() | 13 |
| 3.3.2.15 printFunctionResultsAnalysis() | 13 |
| 3.3.2.16 saveAllAnalyzedDataToFile() | 13 |
| 3.3.2.17 saveAllFunctionDataToFile() | 13 |

| | | |
|----------|--|-----------|
| 3.3.2.18 | saveAllMatricesToFile() | 14 |
| 3.3.2.19 | saveAllProcessedDataToFile() | 14 |
| 3.3.2.20 | saveFunctionMatrixToFile() | 14 |
| 3.3.2.21 | setNumOfDimensions() | 14 |
| 4 | File Documentation | 17 |
| 4.1 | C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/BenchmarkFunctions.cpp File Reference | 17 |
| 4.1.1 | Detailed Description | 18 |
| 4.1.2 | Function Documentation | 18 |
| 4.1.2.1 | ackleysOneFunc() | 18 |
| 4.1.2.2 | ackleysTwoFunc() | 19 |
| 4.1.2.3 | alpineFunc() | 19 |
| 4.1.2.4 | deJongsFunc() | 19 |
| 4.1.2.5 | eggHolderFunc() | 20 |
| 4.1.2.6 | griewangkFunc() | 20 |
| 4.1.2.7 | levyFunc() | 20 |
| 4.1.2.8 | mastersCosWaveFunc() | 21 |
| 4.1.2.9 | michalewiczFunc() | 21 |
| 4.1.2.10 | pathologicalFunc() | 22 |
| 4.1.2.11 | quarticFunc() | 22 |
| 4.1.2.12 | ranaFunc() | 22 |
| 4.1.2.13 | rastriginFunc() | 23 |
| 4.1.2.14 | rosenbrockFunc() | 23 |
| 4.1.2.15 | schefelsFunc() | 23 |
| 4.1.2.16 | sineEnvelopeSineWaveFunc() | 24 |
| 4.1.2.17 | stepFunc() | 24 |
| 4.1.2.18 | stretchedVSineWaveFunc() | 25 |
| 4.2 | C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/BenchmarkFunctions.h File Reference | 25 |
| 4.2.1 | Detailed Description | 26 |
| 4.2.2 | Variable Documentation | 26 |
| 4.2.2.1 | pi | 26 |
| 4.3 | C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/FilenameConstants.h File Reference | 26 |
| 4.3.1 | Detailed Description | 27 |
| 4.4 | C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/ProcessFunctions.cpp File Reference | 27 |
| 4.5 | C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/ProcessFunctions.h File Reference | 28 |
| 4.5.1 | Detailed Description | 28 |
| 4.5.2 | Macro Definition Documentation | 28 |
| 4.5.2.1 | BOUNDARY_MAX | 29 |
| 4.5.2.2 | BOUNDARY_MIN | 29 |

| | |
|--|-----------|
| 4.5.2.3 DEFAULT_NUM_OF_DIMENSIONS | 29 |
| 4.5.2.4 DEFAULT_NUM_OF_VECTORS | 29 |
| 4.6 C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/utilities.cpp File Reference . . | 29 |
| 4.6.1 Detailed Description | 29 |
| 4.6.2 Function Documentation | 30 |
| 4.6.2.1 parseString() | 30 |
| 4.7 C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/utilities.h File Reference . . | 30 |
| 4.7.1 Detailed Description | 30 |
| 4.7.2 Function Documentation | 31 |
| 4.7.2.1 parseString() | 31 |
| Index | 33 |

Chapter 1

Class Index

1.1 Class List

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

| | |
|--|---|
| ProcessFunctions::FunctionAnalysis | |
| Function Analysis Function Analysis Structure, to keep track of the analysis performed on each FunctionData structure. Basically, it compiles and holds the averages of the calculations performed for each function | 5 |
| ProcessFunctions::FunctionData | |
| Function Data Function Data Structure, to keep track of all the data used for the Benchmark Functions | 6 |
| ProcessFunctions | |
| A class used to process matrices against Benchmark Functions and analyze the results | 7 |

Chapter 2

File Index

2.1 File List

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

| | |
|---|----|
| C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/ BenchmarkFunctions.cpp | |
| A library of benchmark functions | 17 |
| C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/ BenchmarkFunctions.h | |
| A library of benchmark functions | 25 |
| C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/ FilenameConstants.h | |
| A list of input and output filenames. The input files are where the matrices are stored. The output files are where the results from the benchmark functions are stored | 26 |
| C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/ ProcessFunctions.cpp | 27 |
| C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/ ProcessFunctions.h | |
| A class used to process matrices against Benchmark Functions and analyze the results | 28 |
| C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/ utilities.cpp | |
| This utilities file is used to create matrices using the Mersenne Twister and store them in Excel files | 29 |
| C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/ utilities.h | |
| This utilities file is used to create matrices using the Mersenne Twister and store them in Excel files | 30 |

Chapter 3

Class Documentation

3.1 ProcessFunctions::FunctionAnalysis Struct Reference

Function Analysis Function Analysis Structure, to keep track of the analysis performed on each [FunctionData](#) structure. Basically, it compiles and holds the averages of the calculations performed for each function.

Public Attributes

- std::string [header](#) = "Function ID,Avg Fitness,Range(min),Range(max),Median,Time(ms)\n"
- std::vector< int > [functionIDs](#)
- std::vector< double > [avgFuntionFitness](#)
- std::vector< std::vector< double > > [ranges](#)
- std::vector< double > [medianFunctionFitness](#)
- std::vector< double > [processTimes](#)

3.1.1 Detailed Description

Function Analysis Function Analysis Structure, to keep track of the analysis performed on each [FunctionData](#) structure. Basically, it compiles and holds the averages of the calculations performed for each function.

3.1.2 Member Data Documentation

3.1.2.1 avgFuntionFitness

```
std::vector<double> ProcessFunctions::FunctionAnalysis::avgFuntionFitness
```

List of the average fitness per [FunctionData](#) structure.

3.1.2.2 functionIDs

```
std::vector<int> ProcessFunctions::FunctionAnalysis::functionIDs
```

List of function IDs.

3.1.2.3 header

```
std::string ProcessFunctions::FunctionAnalysis::header = "Function ID,Avg Fitness,Range (min) ,Range (max) ,Median
```

Header used when saving the data.

3.1.2.4 medianFunctionFitness

```
std::vector<double> ProcessFunctions::FunctionAnalysis::medianFunctionFitness
```

List of the Median fitness from each [FunctionData](#) structure.

3.1.2.5 processTimes

```
std::vector<double> ProcessFunctions::FunctionAnalysis::processTimes
```

List of process times in ms for all functions.

3.1.2.6 ranges

```
std::vector<std::vector<double> > ProcessFunctions::FunctionAnalysis::ranges
```

List of ranges for each fitness result in resultsOfFunctions.

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

- C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/[ProcessFunctions.h](#)

3.2 ProcessFunctions::FunctionData Struct Reference

Function Data Function Data Structure, to keep track of all the data used for the Benchmark Functions.

Public Attributes

- int [functionID](#)
- std::vector< double > [fitness](#)
- std::vector< std::vector< double > > [functionMatrix](#)
- double [timeToExecute](#)

3.2.1 Detailed Description

Function Data Function Data Structure, to keep track of all the data used for the Benchmark Functions.

3.2.2 Member Data Documentation

3.2.2.1 fitness

```
std::vector<double> ProcessFunctions::FunctionData::fitness
```

The list of fitness for each vector in the matrix.

3.2.2.2 functionID

```
int ProcessFunctions::FunctionData::functionID
```

The ID used to determine which of the 18 Benchmark Functions to use.

3.2.2.3 functionMatrix

```
std::vector<std::vector<double> > ProcessFunctions::FunctionData::functionMatrix
```

The matrix of double vectors.

3.2.2.4 timeToExecute

```
double ProcessFunctions::FunctionData::timeToExecute
```

This is time in ms to process all 30 rows.

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

- C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/[ProcessFunctions.h](#)

3.3 ProcessFunctions Class Reference

A class used to process matrices against Benchmark Functions and analyze the results.

```
#include "ProcessFunctions.h"
```

Classes

- struct [FunctionAnalysis](#)
Function Analysis Function Analysis Structure, to keep track of the analysis performed on each [FunctionData](#) structure. Basically, it compiles and holds the averages of the calculations performed for each function.
- struct [FunctionData](#)
Function Data Function Data Structure, to keep track of all the data used for the Benchmark Functions.

Public Member Functions

- [ProcessFunctions](#) ()
The default constructor for the [ProcessFunctions](#) class. The default constructor only initializes the numOfDimensions variable to 0;.
- void [setNumOfDimensions](#) (int dimensions)
Sets the number of dimensions for the [ProcessFunctions](#) object.
- int [getNumOfDimensions](#) ()
Returns the number of dimensions used for the matrix.
- void [constructMatrix](#) ()
Generates a matrix using Mersenne Twister.
- void [constructMatrix](#) (int funcID, double minBoundary, double maxBoundary)
Generates a matrix using Mersenne Twister.
- void [calculateFitnessOfAllMatrices](#) ()
Calculates the fitness of all Matrices in resultsOfFunctions vector.
- void [analyzeAllFunctionResults](#) ()
- void [saveAllMatricesToFile](#) ()
Saves all the matrices in resultsOfFunctions vector to files.
- void [saveAllProcessedDataToFile](#) ()
Saves all the data in resultsOfFunctions to files.
- void [saveAllAnalyzedDataToFile](#) ()
Saves all analyzed data in analysis to file.
- void [printAllFunctionIDs](#) ()
Prints all the possible Function IDs to the screen.
- void [printFunctionResults](#) ()
- void [printFunctionResultsAnalysis](#) ()

Private Member Functions

- [FunctionData](#) [generateMatrix](#) (double minBoundary, double maxBoundary)
Generates a DEFAULT_NUM_OF_VECTORS by DEFAULT_NUM_OF_DIMENSIONS matrix using Mersenne Twister.
- double [calculateFitnessOfVector](#) (std::vector< double > &vect, int functionID)
Calculates the fitness of a vector.
- void [calculateFitnessOfMatrix](#) ([FunctionData](#) &data)
Calculates the fitness of all vectors of a matrix.
- void [analyzeFunctionResults](#) ([FunctionData](#) &data)
- double [calculateAvgFitness](#) ([FunctionData](#) &data)
Calculates the average fitness.
- double [getMinFitness](#) ([FunctionData](#) &data)
Returns the minimum fitness of the data in [FunctionData](#) struct.
- double [getMaxFitness](#) ([FunctionData](#) &data)

Returns the maximum fitness of the data in [FunctionData](#) struct.

- void [saveFunctionMatrixToFile](#) (std::string filename, [FunctionData](#) &data)
Saves the matrix of the [FunctionData](#) to file.
- void [saveAllFunctionDataToFile](#) (std::string filename, [FunctionData](#) &data)
Saves all the data of the function to file.
- void **quicksort** ([FunctionData](#) &data, int L, int R)
- void **swap** ([FunctionData](#) &data, int x, int y)

Private Attributes

- int **numOfDimensions**
- std::vector< [FunctionData](#) > **resultsOfFunctions**
- [FunctionAnalysis](#) **analysis**

3.3.1 Detailed Description

A class used to process matrices against Benchmark Functions and analyze the results.

Author

AI Timofeyev

Date

April 4, 2019

3.3.2 Member Function Documentation

3.3.2.1 analyzeAllFunctionResults()

```
void ProcessFunctions::analyzeAllFunctionResults ( )
```

Analyzes all the results from resultsOfFunctions.

3.3.2.2 analyzeFunctionResults()

```
void ProcessFunctions::analyzeFunctionResults (
    FunctionData & data ) [private]
```

Analyzes the results of the functions.

Parameters

| | |
|-------------|--|
| <i>data</i> | Analyzes the results of the functions. |
|-------------|--|

3.3.2.3 calculateAvgFitness()

```
double ProcessFunctions::calculateAvgFitness (
    FunctionData & data ) [private]
```

Calculates the average fitness.

Parameters

| | |
|-------------|--|
| <i>data</i> | The FunctionData structure that contains the list of fitness values. |
|-------------|--|

Returns

The average fitness from the list of fitness values.

3.3.2.4 calculateFitnessOfAllMatrices()

```
void ProcessFunctions::calculateFitnessOfAllMatrices ( )
```

Calculates the fitness of all Matrices in resultsOfFunctions vector.

Calculates Fitness for all matrices in resultsOfFunctions.

3.3.2.5 calculateFitnessOfMatrix()

```
void ProcessFunctions::calculateFitnessOfMatrix (
    FunctionData & data ) [private]
```

Calculates the fitness of all vectors of a matrix.

Calculates the fitness of all vectors in matrix.

Calculates the fitness of all the vectors of the matrix stored in a [FunctionData](#) structure. All the fitness results are stored in the fitness vector variable of the same [FunctionData](#) structure.

Parameters

| | |
|-------------|--|
| <i>data</i> | The FunctionData structure that contains the matrix. |
|-------------|--|

3.3.2.6 calculateFitnessOfVector()

```
double ProcessFunctions::calculateFitnessOfVector (
```



```
std::vector< double > & vect,
int functionID ) [private]
```

Calculates the fitness of a vector.

Calculates the fitness of a single vector.

The fitness of a vector is calculated by the Benchmark Function referenced by the functionID.

Parameters

| | |
|-------------------|--|
| <i>vect</i> | The vector of elements on which the Benchmark Functions operate. |
| <i>functionID</i> | The ID that references which Benchmark Function to use. |

Returns

The fitness of the vector.

3.3.2.7 constructMatrix() [1/2]

```
void ProcessFunctions::constructMatrix ( )
```

Generates a matrix using Mersenne Twister.

Uses all default constants, or previously user-set dimensions.

A matrix is constructed using the default number of dimensions, or a previously user-set number of dimensions, and the default minimum and maximum bound. Saves the constructed matrix to variable resultsOfFunctions.

3.3.2.8 constructMatrix() [2/2]

```
void ProcessFunctions::constructMatrix (
    int funcID,
    double minBoundary,
    double maxBoundary )
```

Generates a matrix using Mersenne Twister.

Uses default number of dimensions.

A matrix is constructed using the default value of 30 dimensions, or a previously user-set number of dimensions, and a user-provided minimum and maximum bound. Saves the constructed matrix to variable resultsOfFunctions.

Parameters

| | |
|--------------------------------|---|
| <i>funcID</i> | The function ID for which Benchmark Function the matrix is generated for. |
| <i>minBoundary,maxBoundary</i> | The minimum and maximum boundaries for the values in the matrix. |

3.3.2.9 generateMatrix()

```
ProcessFunctions::FunctionData ProcessFunctions::generateMatrix (
    double minBoundary,
    double maxBoundary ) [private]
```

Generates a DEFAULT_NUM_OF_VECTORS by DEFAULT_NUM_OF_DIMENSIONS matrix using Mersenne Twister.

A matrix is constructed using the default value of 30 dimensions and a user-provided minimum and maximum bound.

Note

DEFAULT_NUM_OF_VECTORS is currently set to 30 (as of April 4, 2019).
 DEFAULT_NUM_OF_DIMENSIONS is currently set to 30 (as of April 4, 2019).

Parameters

| | |
|--------------------------------|--|
| <i>minBoundary,maxBoundary</i> | The max/min boundaries are the range range in which to generate numbers. |
|--------------------------------|--|

Returns

The struct that contains the constructed matrix and an empty list of function fitness results.

3.3.2.10 getMaxFitness()

```
double ProcessFunctions::getMaxFitness (
    FunctionData & data ) [private]
```

Returns the maximum fitness of the data in [FunctionData](#) struct.

Parameters

| | |
|-------------|--|
| <i>data</i> | The FunctionData structure that contains a list of fitness values. |
|-------------|--|

Returns

The Maximum fitness in FunctionaData data structure.

3.3.2.11 getMinFitness()

```
double ProcessFunctions::getMinFitness (
    FunctionData & data ) [private]
```

Returns the minimum fitness of the data in [FunctionData](#) struct.

Parameters

| | |
|-------------|--|
| <i>data</i> | The FunctionData structure that contains a list of fitness values. |
|-------------|--|

Returns

The Minimum fitness in FunctionaData data structure.

3.3.2.12 getNumOfDimensions()

```
int ProcessFunctions::getNumOfDimensions ( )
```

Returns the number of dimensions used for the matrix.

Returns the number of dimensions.

Returns

The value stored in the numOfDimensions variable.

3.3.2.13 printAllFunctionIDs()

```
void ProcessFunctions::printAllFunctionIDs ( )
```

Prints all the possible Function IDs to the screen.

Prints all the possible Function IDs to the screen.

Prints all possible Function ID, as well as the funtions they reference, to the screen.

3.3.2.14 printFunctionResults()

```
void ProcessFunctions::printFunctionResults ( )
```

Prints all the [FunctionData](#) structures in resultsOfFunctions.

3.3.2.15 printFunctionResultsAnalysis()

```
void ProcessFunctions::printFunctionResultsAnalysis ( )
```

Prints all the Analysis Results in analysis.

3.3.2.16 saveAllAnalyzedDataToFile()

```
void ProcessFunctions::saveAllAnalyzedDataToFile ( )
```

Saves all analyzed data in analysis to file.

Saves all analyzed data in analysis to file.

3.3.2.17 saveAllFunctionDataToFile()

```
void ProcessFunctions::saveAllFunctionDataToFile (
    std::string filename,
    FunctionData & data ) [private]
```

Saves all the data of the function to file.

Saves the results of the function and it's data to file.

Parameters

| | |
|-----------------|--|
| <i>filename</i> | The filename where to store the matrix. Should be a Excel file (.csv). |
| <i>data</i> | A FunctionData struct that contains all the data of the function, including the matrix that was used as well as the fitness result of that function. |

3.3.2.18 saveAllMatricesToFile()

```
void ProcessFunctions::saveAllMatricesToFile ( )
```

Saves all the matrices in resultsOfFunctions vector to files.

Saves all the matrices in resultsOfFunctions to files.

3.3.2.19 saveAllProcessedDataToFile()

```
void ProcessFunctions::saveAllProcessedDataToFile ( )
```

Saves all the data in resultsOfFunctions to files.

Saves all the data in resultsOfFunctions to files.

3.3.2.20 saveFunctionMatrixToFile()

```
void ProcessFunctions::saveFunctionMatrixToFile (
    std::string filename,
    FunctionData & data ) [private]
```

Saves the matrix of the [FunctionData](#) to file.

Saves the matrix to file.

Parameters

| | |
|-----------------|--|
| <i>filename</i> | The filename where to store the matrix. Should be a Excel file (.csv). |
| <i>data</i> | A FunctionData struct that contains all the data of the function, including the matrix that was used as well as the fitness result of that function. |

3.3.2.21 setNumOfDimensions()

```
void ProcessFunctions::setNumOfDimensions (
    int dimensions )
```

Sets the number of dimensions for the [ProcessFunctions](#) object.

Sets the number of dimensions.

After setting the new number of dimensions, the resultsOfFunctions vector that held all the previous data, for the previous number of dimensions, is also reset to 0.

Parameters

| | |
|-------------------|---|
| <i>dimensions</i> | The number of dimensions in the matrix data (dimensions = size of each vector in the matrix). |
|-------------------|---|

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

- C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/[ProcessFunctions.h](#)
- C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/[ProcessFunctions.cpp](#)

Chapter 4

File Documentation

4.1 C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/BenchmarkFunctions.cpp File Reference

A library of benchmark functions.

```
#include "BenchmarkFunctions.h"
```

Functions

- double [schefelsFunc](#) (vector< double > &vect, int size)
Performs the Schefel's Function on a vector of elements.
- double [deJongsFunc](#) (vector< double > &vect, int size)
Performs the 1st De Jong's Function on a vector of elements.
- double [rosenbrockFunc](#) (vector< double > &vect, int size)
Performs the Rosenbrock Function on a vector of elements.
- double [rastriginFunc](#) (vector< double > &vect, int size)
Performs the Rastrigin Function on a vector of elements.
- double [griewangkFunc](#) (vector< double > &vect, int size)
Performs the Griewangk Function on a vector of elements.
- double [sineEnvelopeSineWaveFunc](#) (vector< double > &vect, int size)
Performs the Sine Envelope Sine Wave Function on a vector of elements.
- double [stretchedVSineWaveFunc](#) (vector< double > &vect, int size)
Performs the Stretched V Sine Wave Function on a vector of elements.
- double [ackleysOneFunc](#) (vector< double > &vect, int size)
Performs the Ackley's One Function on a vector of elements.
- double [ackleysTwoFunc](#) (vector< double > &vect, int size)
Performs the Ackley's Two Function on a vector of elements.
- double [eggHolderFunc](#) (vector< double > &vect, int size)
Performs the Egg Holder Function on a vector of elements.
- double [ranaFunc](#) (vector< double > &vect, int size)
Performs the Rana Function on a vector of elements.
- double [pathologicalFunc](#) (vector< double > &vect, int size)
Performs the Pathological Function on a vector of elements.

- double `michalewiczFunc` (vector< double > &vect, int size)
Performs the Michalewicz Function on a vector of elements.
- double `mastersCosWaveFunc` (vector< double > &vect, int size)
Performs the Masters Cosine Wave Function on a vector of elements.
- double `quarticFunc` (vector< double > &vect, int size)
Performs the Quartic Function on a vector of elements.
- double `levyFunc` (vector< double > &vect, int size)
Performs the Levy Function on a vector of elements.
- double `stepFunc` (vector< double > &vect, int size)
Performs the Step Function on a vector of elements.
- double `alpineFunc` (vector< double > &vect, int size)
Performs the Alpine Function on a vector of elements.

4.1.1 Detailed Description

A library of benchmark functions.

Author

AI Timofeyev

Date

March 28, 2019

4.1.2 Function Documentation

4.1.2.1 `ackleysOneFunc()`

```
double ackleysOneFunc (
    vector< double > & vect,
    int size )
```

Performs the Ackley's One Function on a vector of elements.

Parameters

| | |
|-------------|--|
| <i>vect</i> | The vector of elements on which to perform calculations. |
| <i>size</i> | The number of elements in vector. |

Returns

The results of the calculations (fitness).

4.1.2.2 ackleysTwoFunc()

```
double ackleysTwoFunc (
    vector< double > & vect,
    int size )
```

Performs the Ackley's Two Function on a vector of elements.

Parameters

| | |
|-------------|--|
| <i>vect</i> | The vector of elements on which to perform calculations. |
| <i>size</i> | The number of elements in vector. |

Returns

The results of the calculations (fitness).

4.1.2.3 alpineFunc()

```
double alpineFunc (
    vector< double > & vect,
    int size )
```

Performs the Alpine Function on a vector of elements.

Parameters

| | |
|-------------|--|
| <i>vect</i> | The vector of elements on which to perform calculations. |
| <i>size</i> | The number of elements in vector. |

Returns

The results of the calculations (fitness).

4.1.2.4 deJongsFunc()

```
double deJongsFunc (
    vector< double > & vect,
    int size )
```

Performs the 1st De Jong's Function on a vector of elements.

Parameters

| | |
|-------------|--|
| <i>vect</i> | The vector of elements on which to perform calculations. |
| <i>size</i> | The number of elements in vector. |

Returns

The results of the calculations (fitness).

4.1.2.5 eggHolderFunc()

```
double eggHolderFunc (
    vector< double > & vect,
    int size )
```

Performs the Egg Holder Function on a vector of elements.

Parameters

| | |
|-------------|--|
| <i>vect</i> | The vector of elements on which to perform calculations. |
| <i>size</i> | The number of elements in vector. |

Returns

The results of the calculations (fitness).

4.1.2.6 griewangkFunc()

```
double griewangkFunc (
    vector< double > & vect,
    int size )
```

Performs the Griewangk Function on a vector of elements.

Parameters

| | |
|-------------|--|
| <i>vect</i> | The vector of elements on which to perform calculations. |
| <i>size</i> | The number of elements in vector. |

Returns

The results of the calculations (fitness).

4.1.2.7 levyFunc()

```
double levyFunc (
    vector< double > & vect,
    int size )
```

Performs the Levy Function on a vector of elements.

Parameters

| | |
|-------------|--|
| <i>vect</i> | The vector of elements on which to perform calculations. |
| <i>size</i> | The number of elements in vector. |

Returns

The results of the calculations (fitness).

4.1.2.8 mastersCosWaveFunc()

```
double mastersCosWaveFunc (  
    vector< double > & vect,  
    int size )
```

Performs the Masters Cosine Wave Function on a vector of elements.

Parameters

| | |
|-------------|--|
| <i>vect</i> | The vector of elements on which to perform calculations. |
| <i>size</i> | The number of elements in vector. |

Returns

The results of the calculations (fitness).

4.1.2.9 michalewiczFunc()

```
double michalewiczFunc (  
    vector< double > & vect,  
    int size )
```

Performs the Michalewicz Function on a vector of elements.

Parameters

| | |
|-------------|--|
| <i>vect</i> | The vector of elements on which to perform calculations. |
| <i>size</i> | The number of elements in vector. |

Returns

The results of the calculations (fitness).

4.1.2.10 pathologicalFunc()

```
double pathologicalFunc (
    vector< double > & vect,
    int size )
```

Performs the Pathological Function on a vector of elements.

Parameters

| | |
|-------------|--|
| <i>vect</i> | The vector of elements on which to perform calculations. |
| <i>size</i> | The number of elements in vector. |

Returns

The results of the calculations (fitness).

4.1.2.11 quarticFunc()

```
double quarticFunc (
    vector< double > & vect,
    int size )
```

Performs the Quartic Function on a vector of elements.

Parameters

| | |
|-------------|--|
| <i>vect</i> | The vector of elements on which to perform calculations. |
| <i>size</i> | The number of elements in vector. |

Returns

The results of the calculations (fitness).

4.1.2.12 ranaFunc()

```
double ranaFunc (
    vector< double > & vect,
    int size )
```

Performs the Rana Function on a vector of elements.

Parameters

| | |
|-------------|--|
| <i>vect</i> | The vector of elements on which to perform calculations. |
| <i>size</i> | The number of elements in vector. |

Returns

The results of the calculations (fitness).

4.1.2.13 rastriginFunc()

```
double rastriginFunc (  
    vector< double > & vect,  
    int size )
```

Performs the Rastrigin Function on a vector of elements.

Parameters

| | |
|-------------|--|
| <i>vect</i> | The vector of elements on which to perform calculations. |
| <i>size</i> | The number of elements in vector. |

Returns

The results of the calculations (fitness).

4.1.2.14 rosenbrockFunc()

```
double rosenbrockFunc (  
    vector< double > & vect,  
    int size )
```

Performs the Rosenbrock Function on a vector of elements.

Parameters

| | |
|-------------|--|
| <i>vect</i> | The vector of elements on which to perform calculations. |
| <i>size</i> | The number of elements in vector. |

Returns

The results of the calculations (fitness).

4.1.2.15 schefelsFunc()

```
double schefelsFunc (  
    vector< double > & vect,  
    int size )
```

Performs the Schefel's Function on a vector of elements.

Parameters

| | |
|-------------|--|
| <i>vect</i> | The vector of elements on which to perform calculations. |
| <i>size</i> | The number of elements in vector. |

Returns

The results of the calculations (fitness).

4.1.2.16 sineEnvelopeSineWaveFunc()

```
double sineEnvelopeSineWaveFunc (  
    vector< double > & vect,  
    int size )
```

Performs the Sine Envelope Sine Wave Function on a vector of elements.

Parameters

| | |
|-------------|--|
| <i>vect</i> | The vector of elements on which to perform calculations. |
| <i>size</i> | The number of elements in vector. |

Returns

The results of the calculations (fitness).

4.1.2.17 stepFunc()

```
double stepFunc (  
    vector< double > & vect,  
    int size )
```

Performs the Step Function on a vector of elements.

Parameters

| | |
|-------------|--|
| <i>vect</i> | The vector of elements on which to perform calculations. |
| <i>size</i> | The number of elements in vector. |

Returns

The results of the calculations (fitness).

4.1.2.18 stretchedVSineWaveFunc()

```
double stretchedVSineWaveFunc (
    vector< double > & vect,
    int size )
```

Performs the Stretched V Sine Wave Function on a vector of elements.

Parameters

| | |
|-------------|--|
| <i>vect</i> | The vector of elements on which to perform calculations. |
| <i>size</i> | The number of elements in vector. |

Returns

The results of the calculations (fitness).

4.2 C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/BenchmarkFunctions.h File Reference

A library of benchmark functions.

```
#include <vector>
#include <math.h>
#include <cmath>
```

Functions

- double **schefelsFunc** (std::vector< double > &vect, int size)
- double **deJongsFunc** (std::vector< double > &vect, int size)
- double **rosenbrockFunc** (std::vector< double > &vect, int size)
- double **rastriginFunc** (std::vector< double > &vect, int size)
- double **griewangkFunc** (std::vector< double > &vect, int size)
- double **sineEnvelopeSineWaveFunc** (std::vector< double > &vect, int size)
- double **stretchedVSineWaveFunc** (std::vector< double > &vect, int size)
- double **ackleysOneFunc** (std::vector< double > &vect, int size)
- double **ackleysTwoFunc** (std::vector< double > &vect, int size)
- double **eggHolderFunc** (std::vector< double > &vect, int size)
- double **ranaFunc** (std::vector< double > &vect, int size)
- double **pathologicalFunc** (std::vector< double > &vect, int size)
- double **michalewiczFunc** (std::vector< double > &vect, int size)
- double **mastersCosWaveFunc** (std::vector< double > &vect, int size)
- double **quarticFunc** (std::vector< double > &vect, int size)
- double **levyFunc** (std::vector< double > &vect, int size)
- double **stepFunc** (std::vector< double > &vect, int size)
- double **alpineFunc** (std::vector< double > &vect, int size)

Variables

- const double **pi** = 3.14159265358979323846

4.2.1 Detailed Description

A library of benchmark functions.

Author

AI Timofeyev

Date

March 28, 2019

4.2.2 Variable Documentation

4.2.2.1 pi

```
const double pi = 3.14159265358979323846
```

The pi constant used for calculations.

4.3 C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/Filename↵ Constants.h File Reference

A list of input and output filenames. The input files are where the matrices are stored. The output files are where the results from the benchmark functions are stored.

```
#include <string>
```

Variables

- `std::string in_schefelsFilename` = "z_schefels-Data.csv"
- `std::string in_deJongsFilename` = "z_deJongs-Data.csv"
- `std::string in_rosenbrockFilename` = "z_rosenbrock-Data.csv"
- `std::string in_rastriginFilename` = "z_rastrigin-Data.csv"
- `std::string in_griewangkFilename` = "z_griewangk-Data.csv"
- `std::string in_sEnvSWaveFilename` = "z_sEnvSWave-Data.csv"
- `std::string in_strchVSinWaveFilename` = "z_strchVSinWave-Data.csv"
- `std::string in_ackleys1Filename` = "z_ackleys1-Data.csv"
- `std::string in_ackleys2Filename` = "z_ackleys2-Data.csv"
- `std::string in_eggHolderFilename` = "z_eggHolder-Data.csv"
- `std::string in_ranaFilename` = "z_rana-Data.csv"
- `std::string in_pathologicalFilename` = "z_pathological-Data.csv"
- `std::string in_michalewiczFilename` = "z_michalewicz-Data.csv"
- `std::string in_mastersCosWaveFilename` = "z_mastersCosWave-Data.csv"
- `std::string in_quarticFilename` = "z_quartic-Data.csv"

- `std::string in_levyFilename = "z_levy-Data.csv"`
- `std::string in_stepFilename = "z_step-Data.csv"`
- `std::string in_alpineFilename = "z_alpine-Data.csv"`
- `std::string out_schefelsFilename = "z_schefels-Output.csv"`
- `std::string out_deJongsFilename = "z_deJongs-Output.csv"`
- `std::string out_rosenbrockFilename = "z_rosenbrock-Output.csv"`
- `std::string out_rastriginFilename = "z_rastrigin-Output.csv"`
- `std::string out_griewangkFilename = "z_griewangk-Output.csv"`
- `std::string out_sEnvSWaveFilename = "z_sEnvSWave-Output.csv"`
- `std::string out_strchVSinWaveFilename = "z_strchVSinWave-Output.csv"`
- `std::string out_ackleys1Filename = "z_ackleys1-Output.csv"`
- `std::string out_ackleys2Filename = "z_ackleys2-Output.csv"`
- `std::string out_eggHolderFilename = "z_eggHolder-Output.csv"`
- `std::string out_ranaFilename = "z_rana-Output.csv"`
- `std::string out_pathologicalFilename = "z_pathological-Output.csv"`
- `std::string out_michalewiczFilename = "z_michalewicz-Output.csv"`
- `std::string out_mastersCosWaveFilename = "z_mastersCosWave-Output.csv"`
- `std::string out_quarticFilename = "z_quartic-Output.csv"`
- `std::string out_levyFilename = "z_levy-Output.csv"`
- `std::string out_stepFilename = "z_step-Output.csv"`
- `std::string out_alpineFilename = "z_alpine-Output.csv"`
- `std::string out_Fitness10Dimensions = "10DimensionFitness-ResultsAnalysis"`
- `std::string out_Fitness20Dimensions = "20DimensionFitness-ResultsAnalysis"`
- `std::string out_Fitness30Dimensions = "30DimensionFitness-ResultsAnalysis"`

4.3.1 Detailed Description

A list of input and output filenames. The input files are where the matrices are stored. The output files are where the results from the benchmark functions are stored.

Author

AI Timofeyev

Date

March 28, 2019

4.4 C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/ProcessFunctions.cpp File Reference

```
#include "FilenameConstants.h"
#include "ProcessFunctions.h"
```

4.5 C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/ProcessFunctions.h File Reference

A class used to process matrices against Benchmark Functions and analyze the results.

```
#include <iostream>
#include <fstream>
#include <random>
#include <chrono>
#include "utilities.h"
#include "BenchmarkFunctions.h"
```

Classes

- class [ProcessFunctions](#)
A class used to process matrices against Benchmark Functions and analyze the results.
- struct [ProcessFunctions::FunctionData](#)
Function Data Function Data Structure, to keep track of all the data used for the Benchmark Functions.
- struct [ProcessFunctions::FunctionAnalysis](#)
Function Analysis Function Analysis Structure, to keep track of the analysis performed on each [FunctionData](#) structure. Basically, it compiles and holds the averages of the calculations performed for each function.

Macros

- `#define` [DEFAULT_NUM_OF_DIMENSIONS](#) 30
- `#define` [DEFAULT_NUM_OF_VECTORS](#) 30
- `#define` [BOUNDARY_MIN](#) -500.0
- `#define` [BOUNDARY_MAX](#) 500.0

4.5.1 Detailed Description

A class used to process matrices against Benchmark Functions and analyze the results.

Author

Al Timofeyev

Date

April 4, 2019

4.5.2 Macro Definition Documentation

4.5.2.1 BOUNDARY_MAX

```
#define BOUNDARY_MAX 500.0
```

The default maximum boundary for the elements generated.

4.5.2.2 BOUNDARY_MIN

```
#define BOUNDARY_MIN -500.0
```

The default minimum boundary for the elements generated.

4.5.2.3 DEFAULT_NUM_OF_DIMENSIONS

```
#define DEFAULT_NUM_OF_DIMENSIONS 30
```

The default minimum number of dimensions.

4.5.2.4 DEFAULT_NUM_OF_VECTORS

```
#define DEFAULT_NUM_OF_VECTORS 30
```

The default number of vectors per matrix.

4.6 C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/utilities.cpp File Reference

This utilities file is used to create matrices using the Mersenne Twister and store them in Excel files.

```
#include "utilities.h"
```

Functions

- `std::vector< double > parseString (std::string list, char delimiter)`
Parses a string of numbers into a vector of doubles.

4.6.1 Detailed Description

This utilities file is used to create matrices using the Mersenne Twister and store them in Excel files.

Author

AI Timofeyev

Date

March 28, 2019

4.6.2 Function Documentation

4.6.2.1 `parseString()`

```
std::vector<double> parseString (
    std::string list,
    char delimiter )
```

Parses a string of numbers into a vector of doubles.

Constructs and returns a vector of doubles, given a string list of numbers and a delimiter.

Note

The input string list **MUST** be a list of doubles!

Parameters

| | |
|------------------|--|
| <i>list</i> | A string list of numbers. |
| <i>delimiter</i> | A character used to separate the numbers in the string list. |

Returns

Returns a vector filled with doubles that were extracted from the string list.

4.7 C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/utilities.h File Reference

This utilities file is used to create matrices using the Mersenne Twister and store them in Excel files.

```
#include <string>
#include <vector>
```

Functions

- `std::vector< double > parseString (std::string list, char delimiter)`
Parses a string of numbers into a vector of doubles.

4.7.1 Detailed Description

This utilities file is used to create matrices using the Mersenne Twister and store them in Excel files.

Author

AI Timofeyev

Date

March 28, 2019

4.7.2 Function Documentation

4.7.2.1 parseString()

```
std::vector<double> parseString (
    std::string list,
    char delimiter )
```

Parses a string of numbers into a vector of doubles.

Constructs and returns a vector of doubles, given a string list of numbers and a delimiter.

Note

The input string list MUST be a list of doubles!

Parameters

| | |
|------------------|--|
| <i>list</i> | A string list of numbers. |
| <i>delimiter</i> | A character used to separate the numbers in the string list. |

Returns

Returns a vector filled with doubles that were extracted from the string list.

Index

ackleysOneFunc
 BenchmarkFunctions.cpp, 18

ackleysTwoFunc
 BenchmarkFunctions.cpp, 18

alpineFunc
 BenchmarkFunctions.cpp, 19

analyzeAllFunctionResults
 ProcessFunctions, 9

analyzeFunctionResults
 ProcessFunctions, 9

avgFuntionFitness
 ProcessFunctions::FunctionAnalysis, 5

BenchmarkFunctions.cpp
 ackleysOneFunc, 18
 ackleysTwoFunc, 18
 alpineFunc, 19
 deJongsFunc, 19
 eggHolderFunc, 20
 griewangkFunc, 20
 levyFunc, 20
 mastersCosWaveFunc, 21
 michalewiczFunc, 21
 pathologicalFunc, 21
 quarticFunc, 22
 ranaFunc, 22
 rastriginFunc, 23
 rosenbrockFunc, 23
 schefelsFunc, 23
 sineEnvelopeSineWaveFunc, 24
 stepFunc, 24
 stretchedVSineWaveFunc, 24

BenchmarkFunctions.h
 pi, 26

BOUNDARY_MAX
 ProcessFunctions.h, 28

BOUNDARY_MIN
 ProcessFunctions.h, 29

C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/
 17

C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/
 25

C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/
 26

C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/
 27

C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/
 28

C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/
 29

C:/Users/altim/Documents/School/CS471/Project1/BenchmarkFunctions/
 30

calculateAvgFitness
 ProcessFunctions, 10

calculateFitnessOfAllMatrices
 ProcessFunctions, 10

calculateFitnessOfMatrix
 ProcessFunctions, 10

calculateFitnessOfVector
 ProcessFunctions, 10

constructMatrix
 ProcessFunctions, 11

DEFAULT_NUM_OF_DIMENSIONS
 ProcessFunctions.h, 29

DEFAULT_NUM_OF_VECTORS
 ProcessFunctions.h, 29

deJongsFunc
 BenchmarkFunctions.cpp, 19

eggHolderFunc
 BenchmarkFunctions.cpp, 20

fitness
 ProcessFunctions::FunctionData, 7

functionID
 ProcessFunctions::FunctionData, 7

functionIDs
 ProcessFunctions::FunctionAnalysis, 5

functionMatrix
 ProcessFunctions::FunctionData, 7

generateMatrix
 ProcessFunctions, 12

getMaxFitness
 ProcessFunctions, 12

getMinFitness
 ProcessFunctions, 12

getNumDimensions
 BenchmarkFunctions.cpp, 13
 ProcessFunctions.h, 20
 BenchmarkFunctions.cpp, 20

griewangkFunc
 BenchmarkFunctions/FilenameConstants.h, header

ProcessFunctions::FunctionAnalysis, 6

ProcessFunctions.h, 20

- mastersCosWaveFunc
 - BenchmarkFunctions.cpp, 21
- medianFunctionFitness
 - ProcessFunctions::FunctionAnalysis, 6
- michalewiczFunc
 - BenchmarkFunctions.cpp, 21
- parseString
 - utilities.cpp, 30
 - utilities.h, 31
- pathologicalFunc
 - BenchmarkFunctions.cpp, 21
- pi
 - BenchmarkFunctions.h, 26
- printAllFunctionIDs
 - ProcessFunctions, 13
- printFunctionResults
 - ProcessFunctions, 13
- printFunctionResultsAnalysis
 - ProcessFunctions, 13
- ProcessFunctions, 7
 - analyzeAllFunctionResults, 9
 - analyzeFunctionResults, 9
 - calculateAvgFitness, 10
 - calculateFitnessOfAllMatrices, 10
 - calculateFitnessOfMatrix, 10
 - calculateFitnessOfVector, 10
 - constructMatrix, 11
 - generateMatrix, 12
 - getMaxFitness, 12
 - getMinFitness, 12
 - getNumOfDimensions, 13
 - printAllFunctionIDs, 13
 - printFunctionResults, 13
 - printFunctionResultsAnalysis, 13
 - saveAllAnalyzedDataToFile, 13
 - saveAllFunctionDataToFile, 13
 - saveAllMatricesToFile, 14
 - saveAllProcessedDataToFile, 14
 - saveFunctionMatrixToFile, 14
 - setNumOfDimensions, 14
- ProcessFunctions.h
 - BOUNDARY_MAX, 28
 - BOUNDARY_MIN, 29
 - DEFAULT_NUM_OF_DIMENSIONS, 29
 - DEFAULT_NUM_OF_VECTORS, 29
- ProcessFunctions::FunctionAnalysis, 5
 - avgFunctionFitness, 5
 - functionIDs, 5
 - header, 6
 - medianFunctionFitness, 6
 - processTimes, 6
 - ranges, 6
- ProcessFunctions::FunctionData, 6
 - fitness, 7
 - functionID, 7
 - functionMatrix, 7
 - timeToExecute, 7
- processTimes
 - ProcessFunctions::FunctionAnalysis, 6
- quarticFunc
 - BenchmarkFunctions.cpp, 22
- ranaFunc
 - BenchmarkFunctions.cpp, 22
- ranges
 - ProcessFunctions::FunctionAnalysis, 6
- rastriginFunc
 - BenchmarkFunctions.cpp, 23
- rosenbrockFunc
 - BenchmarkFunctions.cpp, 23
- saveAllAnalyzedDataToFile
 - ProcessFunctions, 13
- saveAllFunctionDataToFile
 - ProcessFunctions, 13
- saveAllMatricesToFile
 - ProcessFunctions, 14
- saveAllProcessedDataToFile
 - ProcessFunctions, 14
- saveFunctionMatrixToFile
 - ProcessFunctions, 14
- schefelsFunc
 - BenchmarkFunctions.cpp, 23
- setNumOfDimensions
 - ProcessFunctions, 14
- sineEnvelopeSineWaveFunc
 - BenchmarkFunctions.cpp, 24
- stepFunc
 - BenchmarkFunctions.cpp, 24
- stretchedVSineWaveFunc
 - BenchmarkFunctions.cpp, 24
- timeToExecute
 - ProcessFunctions::FunctionData, 7
- utilities.cpp
 - parseString, 30
- utilities.h
 - parseString, 31