My Project

Generated by Doxygen 1.8.15

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

Class Documentation

2.1 FA_Params Struct Reference

Public Attributes

- matrix * **std_devs**
- double * **gBests**
- double **gamma**
- double **B0**
- double **alpha**
- double **l_b**
- double **u_b**
- int **func_id**
- int **nf**
- int **dim**
- int **num_iters**
- int **std_devs_row**

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

- FA.h

2.2 functions Class Reference

Public Member Functions

- double Schwefel (double *X, int dimension)
- double first_De_Jong (double *X, int dimension)
- double Rosenbrock (double *X, int dimension)
- double Rastrigin (double *X, int dimension)
- double Greiwangk (double *X, int dimension)
- double Sine_Envelope_Sine_Wave (double *X, int dimension)
- double Stretched_V_Sine_Wave (double *X, int dimension)
- double Ackley_One (double *X, int dimension)
- double Ackley_Two (double *X, int dimension)

- double Egg_Holder (double *X, int dimension)
- double Rana (double *X, int dimension)
- double Pathological (double *X, int dimension)
- double Michalewicz (double *X, int dimension)
- double Masters_Cosine_Wave (double *X, int dimension)
- double Quartic (double *X, int dimension)
- double Levy (double *X, int dimension)
- double Step (double *X, int dimension)
- double Alpine (double *X, int dimension)

2.2.1 Member Function Documentation

2.2.1.1 Ackley_One()

```
double functions::Ackley_One (
            double * X,
            int dimension )
```

Ackley's One function

Parameters

X	the input space
dimension	the size of the input space

Returns

: result of Ackley's One function

2.2.1.2 Ackley_Two()

```
double functions::Ackley_Two (
            double * X,
            int dimension )
```

Ackley's Two function

Parameters

X	the input space
dimension	the size of the input space

Returns

: result of Ackley's Twofunction

2.2.1.3 Alpine()

```
double functions::Alpine (
            double * X,
            int dimension )
```

Alpine's function

Parameters

X	the input space
dimension	the size of the input space

Returns

: result of Alpine's function

2.2.1.4 Egg_Holder()

```
double functions::Egg_Holder (
            double * X,
            int dimension )
```

Egg Holder's function

Parameters

X	the input space
dimension	the size of the input space

Returns

: result of Egg Holder's function

2.2.1.5 first_De_Jong()

```
double functions::first_De_Jong (
            double * X,
            int dimension )
```

1st De Jong's function

Parameters

X	the input space
dimension	the size of the input space

Returns

: result of 1st De Jong's function

2.2.1.6 Greiwangk()

```
double functions::Greiwangk (
            double * X,
            int dimension )
```

Greiwangk's function

Parameters

X	the input space
dimension	the size of the input space

Returns

: result of Greiwangk's function

2.2.1.7 Levy()

```
double functions::Levy (
            double * X,
            int dimension )
```

Levy's function

Parameters

X	the input space
dimension	the size of the input space

Returns

: result of Levy's function

2.2.1.8 Masters_Cosine_Wave()

```
double functions::Masters_Cosine_Wave (
            double * X,
            int dimension )
```

Masters Cosine Wave's function

Parameters

X	the input space
dimension	the size of the input space

Returns

: Masters Cosine Wave's function

2.2.1.9 Michalewicz()

```
double functions::Michalewicz (
            double * X,
            int dimension )
```

Michalewicz's function

Parameters

X	the input space
dimension	the size of the input space

Returns

: result of Michalewicz's function

2.2.1.10 Pathological()

```
double functions::Pathological (
            double * X,
            int dimension )
```

Pathological's function

Parameters

X	the input space
dimension	the size of the input space

Returns

: result of Pathological's function

2.2.1.11 Quartic()

```
double functions::Quartic (
            double * X,
            int dimension )
```

Quartic's function

Parameters

X	the input space
dimension	the size of the input space

Returns

: result of Quartic's function

2.2.1.12 Rana()

```
double functions::Rana (
            double * X,
            int dimension )
```

Rana's function

Parameters

X	the input space
dimension	the size of the input space

Returns

: result of Rana's function

2.2.1.13 Rastrigin()

```
double functions::Rastrigin (
            double * X,
            int dimension )
```

Rastrigin's function

Parameters

X	the input space
dimension	the size of the input space

Returns

: result of Rastrigin's function

2.2.1.14 Rosenbrock()

```
double functions::Rosenbrock (
            double * X,
            int dimension )
```

Rosenbrock's function

Parameters

X	the input space
dimension	the size of the input space

Returns

: result of Rosenbrock's function

2.2.1.15 Schwefel()

```
double functions::Schwefel (
            double * X,
            int dimension )
```

Schwefel's function

Parameters

X	the input space
dimension	the size of the input space

Returns

: result of Schwefel's function

2.2.1.16 Sine_Envelope_Sine_Wave()

```
double functions::Sine_Envelope_Sine_Wave (
            double * X,
            int dimension )
```

Sine Envelope Sine Wave's function

Parameters

X	the input space
dimension	the size of the input space

Returns

: result of Sine Envelope Sine Wave's function

2.2.1.17 Step()

```
double functions::Step (
            double * X,
            int dimension )
```

Step's function

Parameters

X	the input space
dimension	the size of the input space

Returns

: result of Step's function

2.2.1.18 Stretched_V_Sine_Wave()

```
double functions::Stretched_V_Sine_Wave (
            double * X,
            int dimension )
```

Stretched V Sine Wave's function

Parameters

X	the input space
dimension	the size of the input space

Returns

: result of Stretched V Since Wave's function

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

- functions.h
- functions.cpp

2.3 HS_Params Struct Reference

Public Attributes

- matrix * **std_devs**
- double * **gBests**
- double * **gWorsts**
- double **HMCR**
- double **PAR**
- double **HMS**
- double **EOR**
- double **l_b**
- double **u_b**
- double **bw**
- int **func_id**
- int **nh**
- int **dim**
- int **num_iters**
- int **std_devs_row**

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

- HS.h

2.4 matrix Class Reference

Public Member Functions

- matrix (int num_rows, int num_columns, int l_b, int h_b, mt19937 &mt_rand)
- matrix (int num_rows, int num_columns)

Public Attributes

- const int **num_rows**
- const int **num_columns**
- const int **l_b**
- const int **h_b**
- mt19937 **mt_rand**
- double ** **mat**

2.4.1 Constructor & Destructor Documentation

2.4.1.1 matrix() [1/2]

```
matrix::matrix (
            int num_rows,
            int num_columns,
            int l_b,
            int h_b,
            mt19937 & mt_rand )
```

generate an empty matrix and fill it up with randomly generated numbers within some range

Parameters

num_rows	integer respresenting the number of rows in the matrix
dim	integer representing the dimension or number of columns in the matrix
l_b	double representing the lowest bound for the random generator
h_b	double representing the highest bound for the random generator

Returns

: a matrix of randomly generated numbers

2.4.1.2 matrix() [2/2]

```
matrix::matrix (
            int num_rows,
            int num_columns )
```

generate an empty matrix

Parameters

num_rows	integer respresenting the number of rows in the matrix
dim	integer representing the dimension or number of columns in the matrix

Returns

: an empty matrix

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

- matrix.h
- matrix.cpp

2.5 PSO_Params Struct Reference

Public Attributes

- matrix * **std_devs**
- double * **gBests**
- double **c1**
- double **c2**
- double **k**
- double **l_b**
- double **u_b**
- int **func_id**
- int **np**
- int **dim**
- int **num_iters**
- int **std_devs_row**

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

- PSO.h

2.6 utilities Class Reference

Public Member Functions

- double * str_to_tok (char *string, char *delim, int num_tokens)
- void write_to_file (matrix *mat, string file_name)
- int **get_algorithm_id** ()
- int **get_selection_id** ()
- double **find_lowest** (const double *list, int len)
- void simulate (int dim, int ns, int num_functions, double *ranges, int algo_id, int num_iters, int num_exp, double c1, double c2, double k, double gamma, double BO, double alpha, double HMCR, double PAR, double bw, mt19937 &mt_rand)

2.6.1 Member Function Documentation

2.6.1.1 simulate()

```
void utilities::simulate (
            int dim,
            int ns,
            int num_functions,
            double * ranges,
            int algo_id,
            int num_iters,
            int num_exp,
            double c1,
            double c2,
            double k,
            double gamma,
            double B0,
            double alpha,
            double HMCR,
            double PAR,
            double bw,
            mt19937 & mt_rand )
```

simulate the Particle Swarm Optimization, the Firefly algoritm and the Harmony Search algorithm

Parameters

dim	: an integer for the dimension of the solutions
ns	: an integer the number of solutions
num_functions	: an integer for the number of objective functions to be simulated (the 18 functions)
ranges	an array of doubles containing the lower and upper bound for each of the objective functions
algo_id	an integer for the evolutionary algorithm to be simulated
num_iters	: an integer for the number of iterations for the swarm algorithms
num_exp	an integer for the number of experimentations to be run
c1,c2	doubles
k	a double
gamma	a double
B0	a double
alpha	a double
HMCR	a double
PAR	a double
bw	a double
mt_rand	a seeded random generator to generate random numbers (seeded once in main.cpp)

Returns

: None

2.6.1.2 str_to_tok()

```
double * utilities::str_to_tok (
            char * string,
```

```
            char * delim,
            int num_tokens )
```

split a string into double tokens

Parameters

string	the string to be splitted
delim	the character that separates the tokens in the string
num_tokens	number of tokens to expect

Returns

: an array of doubles

2.6.1.3 write_to_file()

```
void utilities::write_to_file (
            matrix * mat,
            string file_name )
```

write a 2d array to a csv file

Parameters

mat	a matrix containing the elements to write to the csv file
file_name	the name of the file where data will be saved

Returns

: None

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

- utilities.h
- utilities.cpp

Index