Gormanium-rush-pentlandite

Generated by Doxygen 1.9.1

Chapter 1

Gormanium-rush-pentlandite

This software aims at using in minerals processing to find a optimal circuit for two products, concentration and tailings.

1.1 Authors

Hanxiao Zhang, Jiarun Yuan, Jin Yu, Junsen Li, Yao Jiang, Yihang Liao, Yuchen Stella Wang, Yuxin Qin, Weihua Gao

1.2 Version

V1.00

1.3 Date

2021-3-26

1.4 Dependencies

python 3.7 Conda make Graphviz Use the following command to install:

```
conda install graphviz python-graphviz
```

1.5 Installation

Use make to complie

```
make all
make tests
make runtest
```

Chapter 2

README_graph

This tool provides a basic visualization method for the the vector output of the genetic algorithm code. It used the graph visualization package Graphviz https://graphviz.gitlab.io.

2.1 Installation

Install the packages

```
conda env create -f environment.yml
conda activate acse-4-p3
```

or you can (conda) install the packages yourself

```
conda install graphviz python-graphviz
```

2.2 Running the software

```
conda activate acsse-4-p3
python graph.py
```

(You will need to in the work directory /visualization)

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 graph Namespace Reference

Functions

- def loadDataset (file, k)
- def plot (output)

Variables

- string output1 = 'output.txt'
- int k = 21
- list final = [y for x in output1 for y in x]
- verbose

6.1.1 Detailed Description

```
@package docstring
File: This module is used to visualize the results of the output vector
```

6.1.2 Function Documentation

6.1.2.1 loadDataset()

```
def graph.loadDataset (
              file,
              k )

This function is used to load the previous output vector
Parameters:
    file: A txt file with output vector saved
    k: 21 for most cases
Return:
    A python vector nested anther python vector
>>> loadDataset('output.txt', 21)
[[3, 4, 5, 9, 3, 1, 11, 4, 7, 9, 8, 4, 6, 8, 2, 4, 0, 9, 1, 10, 4]]
```

Definition at line 8 of file graph.py.

6.1.2.2 plot()

```
def graph.plot (
              output )
```

```
This function is to plot a flow chart.
Parameters:
    Output: A python vector.
Return
    If the graph is plot succussfully, you can see 'Plot successfully!' in the terminal.
>>> plot([3, 4, 5, 9, 3, 1, 11, 4, 7, 9, 8, 4, 6, 8, 2, 4, 0, 9, 1, 10, 4])
Plot successfully!
```

Definition at line 32 of file graph.py.

6.1.3 Variable Documentation

6.1.3.1 final

```
list graph.final = [y for x in output1 for y in x]
```

Definition at line 80 of file graph.py.

6.1.3.2 k

```
int graph.k = 21
```

Definition at line 78 of file graph.py.

6.1.3.3 output1

```
def graph.output1 = 'output.txt'
```

Definition at line 77 of file graph.py.

6.1.3.4 verbose

```
graph.verbose
```

Definition at line 87 of file graph.py.

6.2 run_tests Namespace Reference

Variables

- string DIR = "tests/bin/∗"
- files = glob.glob(DIR)
- bool glob_fail = False
- output = subprocess.run([file], stdout=subprocess.PIPE, universal_newlines=True)
- bool test_failed = False

6.2.1 Variable Documentation

6.2.1.1 DIR

```
string run_tests.DIR = "tests/bin/∗"
```

Definition at line 8 of file run_tests.py.

6.2.1.2 files

```
run_tests.files = glob.glob(DIR)
```

Definition at line 10 of file run_tests.py.

6.2.1.3 glob_fail

```
bool run_tests.glob_fail = False
```

Definition at line 12 of file run_tests.py.

6.2.1.4 output

```
run_tests.output = subprocess.run([file], stdout=subprocess.PIPE, universal_newlines=True)
```

Definition at line 17 of file run_tests.py.

6.2.1.5 test_failed

```
bool run_tests.test_failed = False
```

Definition at line 18 of file run_tests.py.

Chapter 7

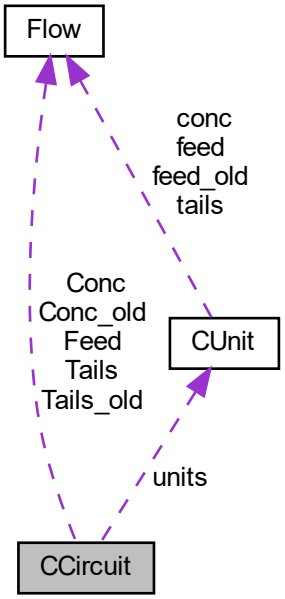
Class Documentation

7.1 CCircuit Class Reference

CCircuit A class storing various information, from the initial economic condition, number of units to the mass flows between them as well as the monetary_value calculated.

`#include <CCircuit.h>`

Collaboration diagram for CCircuit:



Public Member Functions

- CCircuit ()
- CCircuit (const CCircuit &to_copy)
- ~CCircuit ()
- void initialize_units ()

  initialize unit according to unit number
- void set_initial_flow (int unit)
- void clear ()

  clear unit, concentration, tailing
- void save_flow ()
- void clear_marks ()
- void calculate_monetary_value (bool &diverge, double tolerance=1e-12, int max_iter=500)

  Calculate monetary value, using iteration method.
- void set_units (int *vec)

  set up units for a CCircuit class according to vector value
- void initialize_vector ()

  initialize vector
- void get_circuit_vector ()

  Extract the configaration vector from a CCircuit class.
- bool Check_Validity ()

  this function is to check validility
- void mark_units (int unit_num)
- void generate_random ()

  generate a random configuration for a CCircuit object, which is likely to be valid
- void calculate_monetary_value_directly ()

  The direct linear solver to calculate the monetary value of the concentration for a CCircuit object, overwrting the monetary_value it stored.
- int * generate_vector ()

  return a copy for the configuration vector for a CCircuit object.

Static Public Member Functions

- static void setup_initial_parameters (const int num_units, const double feed_mass_gormanium, const double feed_mass_waste, const double c_rate_gormanium, const double c_rate_waste, const double t_rate_gormanium, const double t_rate_waste, const double value_gormanium, const double value_waste)

  set initial parameters
- static void setup_CUnits ()

  set CUnit parameters according to CCircuit
- static double monetary_direct_solver (int *circuit_vector, int len)
- static void Guassian_elimination (double **a, double *b, int n)

  Do Gaussian elimination, for solving the linear equation Ax = b.
- static void matmul (double **a, double *b, int n)

  performing left Matrix-vector multiplication

Public Attributes

- CUnit * units = NULL
- Flow Feed
- Flow Conc
- Flow Tails
- Flow Conc_old
- Flow Tails_old
- int source
- double monetary_value
- bool find_conc
- bool find_tails
- int * circuit_vector = nullptr

Static Public Attributes

- static int num_units
- static int cells
- static double feed_mass_gormanium
- static double feed_mass_waste
- static double c_rate_gormanium
- static double c_rate_waste
- static double t_rate_gormanium
- static double t_rate_waste
- static double value_gormanium
- static double value_waste
- static double worst_case

7.1.1 Detailed Description

CCircuit A class storing various information, from the initial economic condition, number of units to the mass flows between them as well as the monetary_value calculated.

Definition at line 11 of file CCircuit.h.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 CCircuit() [1/2]

```
CCircuit::CCircuit ( )  [inline]
```

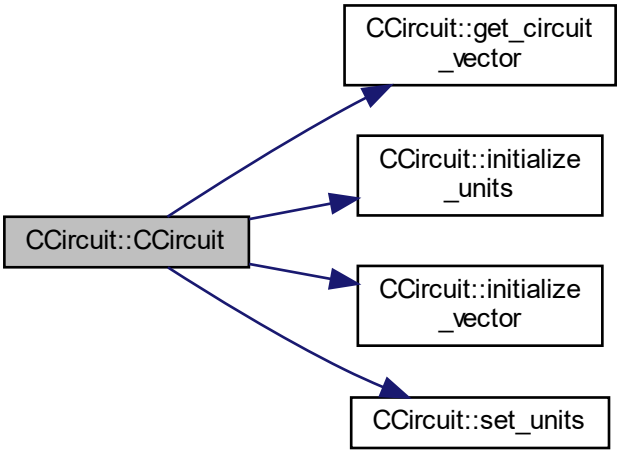
Definition at line 40 of file CCircuit.h.

7.1.2.2 CCircuit() [2/2]

```
CCircuit::CCircuit (
            const CCircuit & to_copy )  [inline]
```

Definition at line 41 of file CCircuit.h.

Here is the call graph for this function:



7.1.2.3 ~CCircuit()

```
CCircuit::~CCircuit ( )  [inline]
```

Definition at line 51 of file CCircuit.h.

7.1.3 Member Function Documentation

7.1.3.1 calculate_monetary_value()

```
void CCircuit::calculate_monetary_value (
            bool & diverge,
            double tolerance = 1e-12,
            int max_iter = 500 )
```

Calculate monetary value, using iteration method.

Parameters

diverge	boolean indicating whether flow in the circuit is diverging or not.
tolerance	which is setting for the stopping criteria, if the change over time step is small, we achieve our balance state.
max_iter	we do not want our method taking two much time, if some case converge too slow and met max iteraton times we set, the method would quit.

Returns

void

Definition at line 143 of file CCircuit.cpp.

7.1.3.2 calculate_monetary_value_directly()

```
void CCircuit::calculate_monetary_value_directly ( )
```

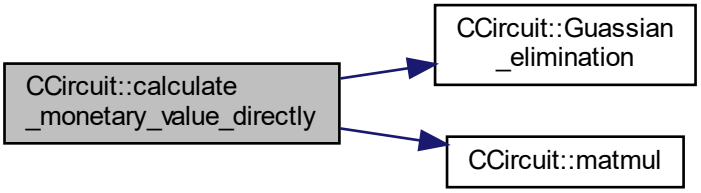
The direct linear solver to calculate the monetary value of the concentration for a CCircuit object, overwrting the monetary_value it stored.

Returns

void

Definition at line 511 of file CCircuit.cpp.

Here is the call graph for this function:



7.1.3.3 Check_Validity()

```
bool CCircuit::Check_Validity ( )
```

this function is to check validility

Returns

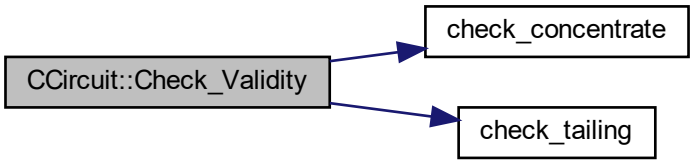
true if valid

Note

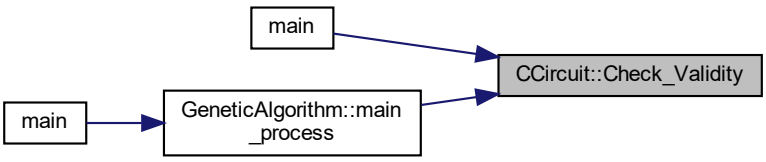
this function firsly construct the units array, if the unit is self connected, directly return false, then the function check whether all units are connected to tail and concentrate.

Definition at line 312 of file CCircuit.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



7.1.3.4 clear()

```
void CCircuit::clear ( )
```

clear unit, concentration, tailing

Returns

void

Definition at line 108 of file CCircuit.cpp.

7.1.3.5 clear_marks()

```
void CCircuit::clear_marks ( )
```

7.1.3.6 generate_random()

```
void CCircuit::generate_random ( )
```

generate a random configuration for a CCircuit object, which is likely to be valid

Returns

void

Definition at line 463 of file CCircuit.cpp.

7.1.3.7 generate_vector()

```
int * CCircuit::generate_vector ( )
```

return a copy for the configuration vector for a CCircuit object.

Returns

vector

Definition at line 492 of file CCircuit.cpp.

7.1.3.8 get_circuit_vector()

```
void CCircuit::get_circuit_vector ( )
```

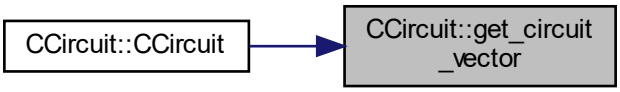
Extract the configaration vector from a CCircuit class.

Returns

void

Definition at line 253 of file CCircuit.cpp.

Here is the caller graph for this function:



7.1.3.9 Guassian_elimination()

```
void CCircuit::Guassian_elimination (
            double ** a,
            double * b,
            int n )  [static]
```

Do Gaussian elimination, for solving the linear equation Ax = b.

Parameters

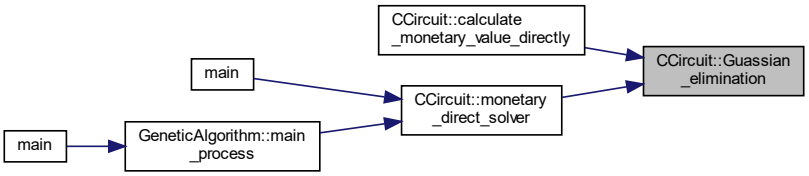
a	the matrix A on the left hand side
b	the RHS vector
n	the size of the problem

Returns

void

Definition at line 386 of file CCircuit.cpp.

Here is the caller graph for this function:



7.1.3.10 initialize_units()

```
void CCircuit::initialize_units ( )
```

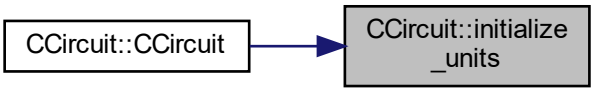
initialize unit according to unit number

Returns

void

Definition at line 90 of file CCircuit.cpp.

Here is the caller graph for this function:



7.1.3.11 initialize_vector()

```
void CCircuit::initialize_vector ( )
```

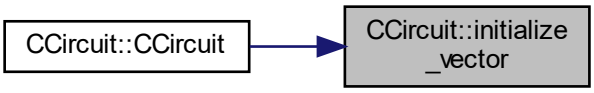
initialize vector

Returns

void

Definition at line 98 of file CCircuit.cpp.

Here is the caller graph for this function:



7.1.3.12 mark_units()

```
void CCircuit::mark_units (
            int unit_num )
```

7.1.3.13 matmul()

```
void CCircuit::matmul (
            double ** a,
            double * b,
            int n )  [static]
```

performing left Matrix-vector multiplication

Parameters

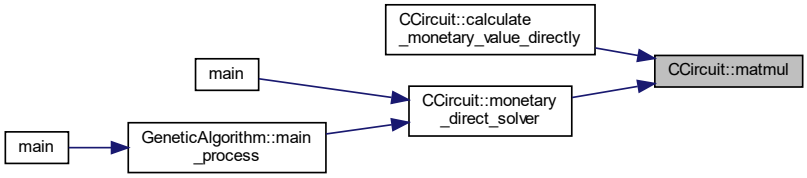
a	the matrix
b	the vector to be mutiplicated
n	the size of the problem

Returns

void

Definition at line 437 of file CCircuit.cpp.

Here is the caller graph for this function:



7.1.3.14 monetary_direct_solver()

```
double CCircuit::monetary_direct_solver (
            int * circuit_vector,
            int len )  [static]
```

Parameters

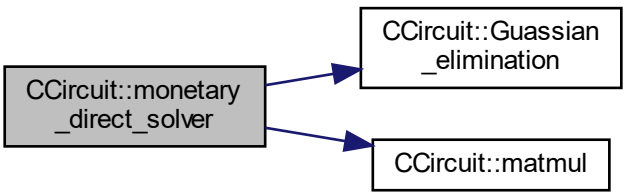
circuit_vector	The direct linear solver to calculate the monetary value of the concentration for a configuration vector, returning the monetary_value as a double.
len	The length of the Circuit configuration vector

Returns

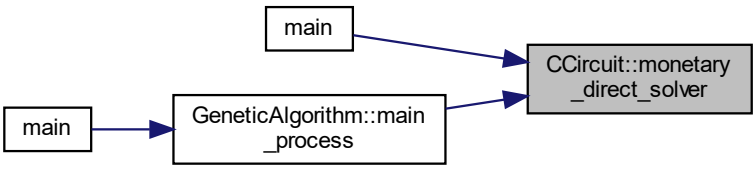
momentary value

Definition at line 561 of file CCircuit.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



7.1.3.15 save_flow()

```
void CCircuit::save_flow ( )
```

Definition at line 116 of file CCircuit.cpp.

7.1.3.16 set_initial_flow()

```
void CCircuit::set_initial_flow (
            int unit )
```

7.1.3.17 set_units()

```
void CCircuit::set_units (
            int * vec )
```

set up units for a CCircuit class according to vector value

Parameters

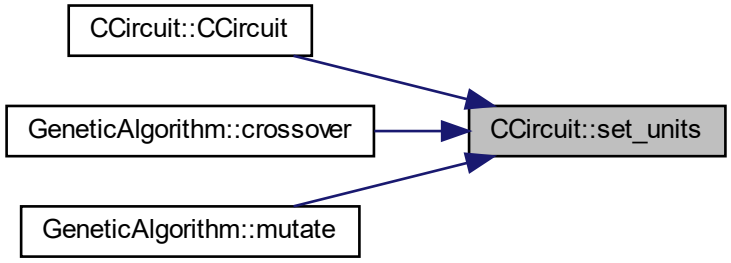
vec	the input vector, indicating a configaration.

Returns

void

Definition at line 237 of file CCircuit.cpp.

Here is the caller graph for this function:



7.1.3.18 setup_CUnits()

```
void CCircuit::setup_CUnits ( )  [static]
```

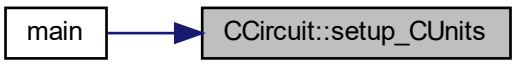
set CUnit parameters according to CCircuit

Returns

void

Definition at line 79 of file CCircuit.cpp.

Here is the caller graph for this function:



7.1.3.19 setup_initial_parameters()

```
void CCircuit::setup_initial_parameters (
            const int num_units,
            const double feed_mass_gormanium,
            const double feed_mass_waste,
            const double c_rate_gormanium,
            const double c_rate_waste,
            const double t_rate_gormanium,
            const double t_rate_waste,
            const double value_gormanium,
            const double value_waste )  [static]
```

set initial parameters

Parameters

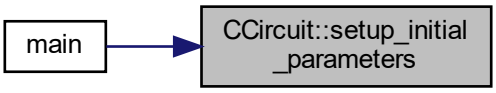
num_units	
feed_mass_gormanium	
feed_mass_waste	
c_rate_gormanium	
c_rate_waste	
t_rate_gormanium	
t_rate_waste	
value_gormanium	
value_waste	

Returns

void

Definition at line 58 of file CCircuit.cpp.

Here is the caller graph for this function:



7.1.4 Member Data Documentation

7.1.4.1 c_rate_gormanium

```
double CCircuit::c_rate_gormanium  [static]
```

Definition at line 19 of file CCircuit.h.

7.1.4.2 c_rate_waste

```
double CCircuit::c_rate_waste  [static]
```

Definition at line 20 of file CCircuit.h.

7.1.4.3 cells

```
int CCircuit::cells  [static]
```

Definition at line 14 of file CCircuit.h.

7.1.4.4 circuit_vector

```
int* CCircuit::circuit_vector = nullptr
```

Definition at line 31 of file CCircuit.h.

7.1.4.5 Conc

```
Flow CCircuit::Conc
```

Definition at line 16 of file CCircuit.h.

7.1.4.6 Conc_old

```
Flow CCircuit::Conc_old
```

Definition at line 16 of file CCircuit.h.

7.1.4.7 Feed

```
Flow CCircuit::Feed
```

Definition at line 16 of file CCircuit.h.

7.1.4.8 feed_mass_gormanium

```
double CCircuit::feed_mass_gormanium  [static]
```

Definition at line 17 of file CCircuit.h.

7.1.4.9 feed_mass_waste

```
double CCircuit::feed_mass_waste  [static]
```

Definition at line 18 of file CCircuit.h.

7.1.4.10 find_conc

```
bool CCircuit::find_conc
```

Definition at line 29 of file CCircuit.h.

7.1.4.11 find_tails

```
bool CCircuit::find_tails
```

Definition at line 29 of file CCircuit.h.

7.1.4.12 monetary_value

```
double CCircuit::monetary_value
```

Definition at line 27 of file CCircuit.h.

7.1.4.13 num_units

```
int CCircuit::num_units  [static]
```

Definition at line 13 of file CCircuit.h.

7.1.4.14 source

```
int CCircuit::source
```

Definition at line 26 of file CCircuit.h.

7.1.4.15 t_rate_gormanium

```
double CCircuit::t_rate_gormanium  [static]
```

Definition at line 21 of file CCircuit.h.

7.1.4.16 t_rate_waste

```
double CCircuit::t_rate_waste  [static]
```

Definition at line 22 of file CCircuit.h.

7.1.4.17 Tails

```
Flow CCircuit::Tails
```

Definition at line 16 of file CCircuit.h.

7.1.4.18 Tails_old

```
Flow CCircuit::Tails_old
```

Definition at line 16 of file CCircuit.h.

7.1.4.19 units

```
CUnit* CCircuit::units = NULL
```

Definition at line 15 of file CCircuit.h.

7.1.4.20 value_gormanium

```
double CCircuit::value_gormanium  [static]
```

Definition at line 23 of file CCircuit.h.

7.1.4.21 value_waste

```
double CCircuit::value_waste  [static]
```

Definition at line 24 of file CCircuit.h.

7.1.4.22 worst_case

```
double CCircuit::worst_case  [static]
```

Definition at line 25 of file CCircuit.h.

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

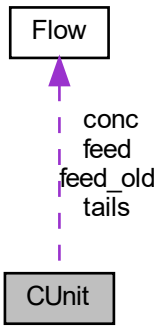
- includes/CCircuit.h
- src/CCircuit.cpp

7.2 CUnit Class Reference

CUnit The class for a certain unit in the Circuit.

```
#include <CUnit.h>
```

Collaboration diagram for CUnit:



Public Member Functions

- void calculate_outflow ()

  Calculate the mass of Gormanium and waste for the concentration and tails outflows by a given feed.
- void clearAll ()

  clear all flows stored in a certain unit.

Public Attributes

- int conc_num
- int tails_num
- bool mark
- Flow feed
- Flow feed_old
- Flow conc
- Flow tails
- bool connect_to_conc
- bool connect_to_tails
- bool connect_from_source

Static Public Attributes

- static double c_rate_gormanium
- static double c_rate_waste
- static double t_rate_gormanium
- static double t_rate_waste

7.2.1 Detailed Description

CUnit The class for a certain unit in the Circuit.

Definition at line 52 of file CUnit.h.

7.2.2 Member Function Documentation

7.2.2.1 calculate_outflow()

```
void CUnit::calculate_outflow ( )
```

Calculate the mass of Gormanium and waste for the concentration and tails outflows by a given feed.

Returns

void

Definition at line 8 of file CUnit.cpp.

7.2.2.2 clearAll()

```
void CUnit::clearAll ( )
```

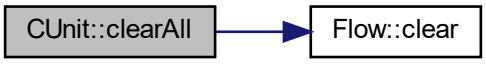
clear all flows stored in a certain unit.

Returns

void

Definition at line 20 of file CUnit.cpp.

Here is the call graph for this function:



7.2.3 Member Data Documentation

7.2.3.1 c_rate_gormanium

```
double CUnit::c_rate_gormanium  [static]
```

Definition at line 61 of file CUnit.h.

7.2.3.2 c_rate_waste

```
double CUnit::c_rate_waste  [static]
```

Definition at line 62 of file CUnit.h.

7.2.3.3 conc

```
Flow CUnit::conc
```

Definition at line 65 of file CUnit.h.

7.2.3.4 conc_num

```
int CUnit::conc_num
```

Definition at line 55 of file CUnit.h.

7.2.3.5 connect_from_source

```
bool CUnit::connect_from_source
```

Definition at line 73 of file CUnit.h.

7.2.3.6 connect_to_conc

```
bool CUnit::connect_to_conc
```

Definition at line 71 of file CUnit.h.

7.2.3.7 connect_to_tails

```
bool CUnit::connect_to_tails
```

Definition at line 72 of file CUnit.h.

7.2.3.8 feed

```
Flow CUnit::feed
```

Definition at line 65 of file CUnit.h.

7.2.3.9 feed_old

```
Flow CUnit::feed_old
```

Definition at line 65 of file CUnit.h.

7.2.3.10 mark

```
bool CUnit::mark
```

Definition at line 59 of file CUnit.h.

7.2.3.11 t_rate_gormanium

```
double CUnit::t_rate_gormanium  [static]
```

Definition at line 63 of file CUnit.h.

7.2.3.12 t_rate_waste

```
double CUnit::t_rate_waste  [static]
```

Definition at line 64 of file CUnit.h.

7.2.3.13 tails

```
Flow CUnit::tails
```

Definition at line 65 of file CUnit.h.

7.2.3.14 tails_num

```
int CUnit::tails_num
```

Definition at line 57 of file CUnit.h.

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

- includes/CUnit.h
- src/CCircuit.cpp
- src/CUnit.cpp

7.3 Flow Class Reference

Flow Representing the mass flow between units.

```
#include <CUnit.h>
```

Public Member Functions

- void set (double G, double W)
- void set (Flow flow)
- void plus (Flow feed)
- void clear ()
- double distance (Flow flow1)

Public Attributes

- double Gormanium = 0
- double waste = 0

7.3.1 Detailed Description

Flow Representing the mass flow between units.

Definition at line 9 of file CUnit.h.

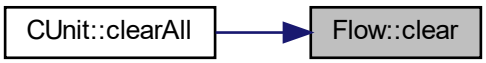
7.3.2 Member Function Documentation

7.3.2.1 clear()

```
void Flow::clear ( )  [inline]
```

Definition at line 34 of file CUnit.h.

Here is the caller graph for this function:



7.3.2.2 distance()

```
double Flow::distance (
            Flow flow1 )  [inline]
```

Definition at line 40 of file CUnit.h.

7.3.2.3 plus()

```
void Flow::plus (
            Flow feed )  [inline]
```

Definition at line 28 of file CUnit.h.

7.3.2.4 set() [1/2]

```
void Flow::set (
            double G,
            double W )  [inline]
```

Definition at line 16 of file CUnit.h.

7.3.2.5 set() [2/2]

```
void Flow::set (
            Flow flow )  [inline]
```

Definition at line 22 of file CUnit.h.

7.3.3 Member Data Documentation

7.3.3.1 Gormanium

```
double Flow::Gormanium = 0
```

Definition at line 12 of file CUnit.h.

7.3.3.2 waste

```
double Flow::waste = 0
```

Definition at line 13 of file CUnit.h.

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

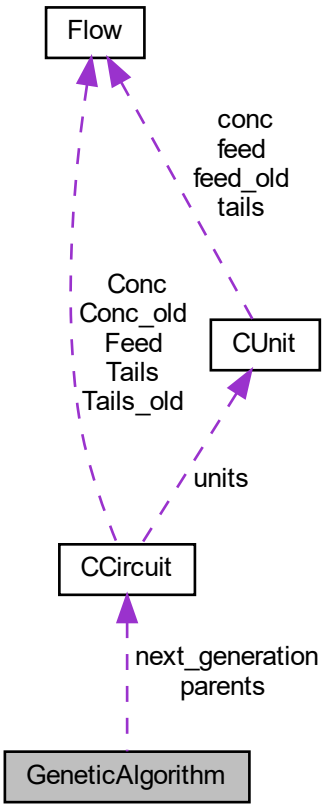
- includes/CUnit.h

7.4 GeneticAlgorithm Class Reference

Genetic algorithm Optimise a system represented by a specification vector.

`#include <Genetic_Algorithm.h>`

Collaboration diagram for GeneticAlgorithm:



Public Member Functions

- void initialize_list (int *vec)

  Generate one random CCircuit vector.
- int select_parents (CCircuit *parents, int n)

  Use tournament selection select parents.
- void find_max_index (CCircuit *parents, int &index, double &value)

  Find the best vector.
- void crossover (CCircuit *mother, CCircuit *father)

  @mother and @father are selected to cross, a random point in the vector is chosen and all of the values before that point are swapped with the corresponding points in the other vector.
- void mutate (CCircuit *parent)

  Go over each of the numbers in the vector and decide whether to mutate them.

- void main_process ()

  The main process of the basic genetic algorithm.
- void print_parents ()

  Print value in parents.
- void print_next_generation ()

  Print value in next generation.
- GeneticAlgorithm ()

  constructor
- GeneticAlgorithm (int iter_count, int threshold)

  constructor
- ~GeneticAlgorithm ()

  destructor

Static Public Member Functions

- static void setup_hyper_parameters (int parent_count, double crossover_rate, double mutation_rate)

  set up hyper parameters

Public Attributes

- CCircuit * parents = nullptr

  collection of valid circuits
- CCircuit * next_generation =nullptr

  collection of valid circuits in next generation
- int iter_count

  a set number of iterations
- double threshold

  the minimum performance gap between two generation of best vector

Static Public Attributes

- static int parent_count

  the number of offspring/parent n that are evaluated in each generation
- static double crossover_rate

  the probability of crossing selected parents rather than passing them into the mutation step unchanged
- static double mutation_rate

  the rate at which mutations are introduced

7.4.1 Detailed Description

Genetic algorithm Optimise a system represented by a specification vector.

Definition at line 6 of file Genetic_Algorithm.h.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 GeneticAlgorithm() [1/2]

```
GeneticAlgorithm::GeneticAlgorithm ( )
```

constructor

Definition at line 278 of file Genetic_Algorithm.cpp.

7.4.2.2 GeneticAlgorithm() [2/2]

```
GeneticAlgorithm::GeneticAlgorithm (
            int iter_count,
            int threshold )
```

constructor

Parameters

iter_count	max iteration
threshold	the minimum performance gap between two generation of best vector

Definition at line 289 of file Genetic_Algorithm.cpp.

7.4.2.3 ~GeneticAlgorithm()

```
GeneticAlgorithm::~GeneticAlgorithm ( )
```

destructor

Definition at line 298 of file Genetic_Algorithm.cpp.

7.4.3 Member Function Documentation

7.4.3.1 crossover()

```
void GeneticAlgorithm::crossover (
            CCircuit * mother,
            CCircuit * father )
```

@mother and @father are selected to cross, a random point in the vector is chosen and all of the values before that point are swapped with the corresponding points in the other vector.

Parameters

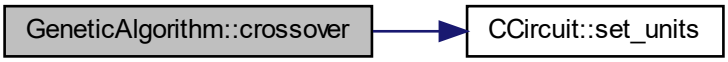
mother	selected parent
father	selected parent

Returns

void

Definition at line 100 of file Genetic_Algorithm.cpp.

Here is the call graph for this function:



7.4.3.2 find_max_index()

```
void GeneticAlgorithm::find_max_index (
            CCircuit * parents,
            int & index,
            double & value )
```

Find the best vector.

Parameters

parents	collection of valid circuits
index	index of best vector
value	value of best vector

Returns

void

Definition at line 83 of file Genetic_Algorithm.cpp.

7.4.3.3 initialize_list()

```
void GeneticAlgorithm::initialize_list (
            int * vec )
```

Generate one random CCircuit vector.

Parameters

vec	

Definition at line 41 of file Genetic_Algorithm.cpp.

7.4.3.4 main_process()

```
void GeneticAlgorithm::main_process ( )
```

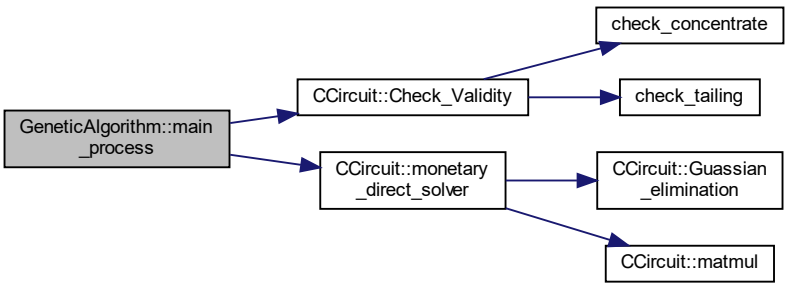
The main process of the basic genetic algorithm.

Returns

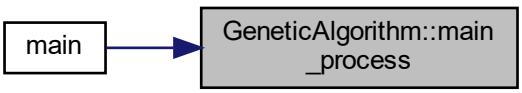
void

Definition at line 137 of file Genetic_Algorithm.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



7.4.3.5 mutate()

```
void GeneticAlgorithm::mutate (
            CCircuit * parent )
```

Go over each of the numbers in the vector and decide whether to mutate them.

Parameters

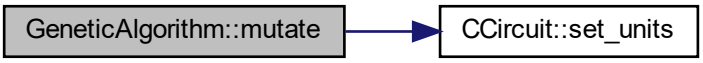
parent	

Returns

void

Definition at line 124 of file Genetic_Algorithm.cpp.

Here is the call graph for this function:



7.4.3.6 print_next_generation()

```
void GeneticAlgorithm::print_next_generation ( )
```

Print value in next generation.

Returns

void

Definition at line 263 of file Genetic_Algorithm.cpp.

7.4.3.7 print_parents()

```
void GeneticAlgorithm::print_parents ( )
```

Print value in parents.

Returns

void

Definition at line 247 of file Genetic_Algorithm.cpp.

7.4.3.8 select_parents()

```
int GeneticAlgorithm::select_parents (
            CCircuit * parents,
            int n )
```

Use tournament selection select parents.

Parameters

parents	collection of valid circuits
n	select n from parent then use largest one as parent

Returns

parent

Definition at line 61 of file Genetic_Algorithm.cpp.

7.4.3.9 setup_hyper_parameters()

```
void GeneticAlgorithm::setup_hyper_parameters (
            int parent_count,
            double crossover_rate,
            double mutation_rate )  [static]
```

set up hyper parameters

Parameters

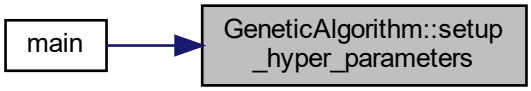
parent_count	the number of offspring n that are evaluated in each generation;
crossover_rate	the probability of crossing selected parents rather than passing them into the mutation step unchanged (a recommended range is between 0.8 and 1)
mutation_rate	the rate at which mutations are introduced (recommended probabilities of 1% or lower).

Returns

void

Definition at line 31 of file Genetic_Algorithm.cpp.

Here is the caller graph for this function:



7.4.4 Member Data Documentation

7.4.4.1 crossover_rate

```
double GeneticAlgorithm::crossover_rate  [static]
```

the probability of crossing selected parents rather than passing them into the mutation step unchanged

Definition at line 47 of file Genetic_Algorithm.h.

7.4.4.2 iter_count

```
int GeneticAlgorithm::iter_count
```

a set number of iterations

Definition at line 37 of file Genetic_Algorithm.h.

7.4.4.3 mutation_rate

```
double GeneticAlgorithm::mutation_rate  [static]
```

the rate at which mutations are introduced

Definition at line 50 of file Genetic_Algorithm.h.

7.4.4.4 next_generation

```
CCircuit* GeneticAlgorithm::next_generation =nullptr
```

collection of valid circuits in next generation

Definition at line 33 of file Genetic_Algorithm.h.

7.4.4.5 parent_count

```
int GeneticAlgorithm::parent_count  [static]
```

the number of offspring/parent n that are evaluated in each generation

Definition at line 43 of file Genetic_Algorithm.h.

7.4.4.6 parents

```
CCircuit* GeneticAlgorithm::parents = nullptr
```

collection of valid circuits

Definition at line 31 of file Genetic_Algorithm.h.

7.4.4.7 threshold

```
double GeneticAlgorithm::threshold
```

the minimum performance gap between two generation of best vector

Definition at line 40 of file Genetic_Algorithm.h.

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

- includes/Genetic_Algorithm.h
- src/Genetic_Algorithm.cpp

7.5 node Struct Reference

Public Attributes

- bool c_val = false
- bool t_val = false
- int unit
- int p_num = 0
- int * parent

7.5.1 Detailed Description

Definition at line 9 of file CCircuit.cpp.

7.5.2 Member Data Documentation

7.5.2.1 c_val

```
bool node::c_val = false
```

Definition at line 11 of file CCircuit.cpp.

7.5.2.2 p_num

```
int node::p_num = 0
```

Definition at line 14 of file CCircuit.cpp.

7.5.2.3 parent

```
int* node::parent
```

Definition at line 15 of file CCircuit.cpp.

7.5.2.4 t_val

```
bool node::t_val = false
```

Definition at line 12 of file CCircuit.cpp.

7.5.2.5 unit

```
int node::unit
```

Definition at line 13 of file CCircuit.cpp.

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

- src/CCircuit.cpp

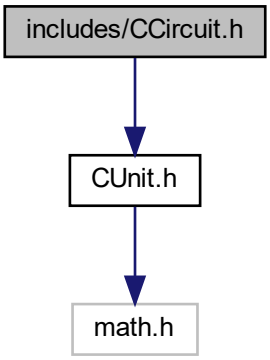
Chapter 8

File Documentation

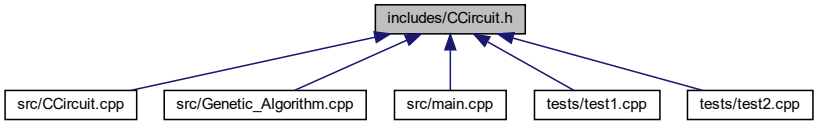
8.1 includes/CCircuit.h File Reference

```
#include "CUnit.h"
```

Include dependency graph for CCircuit.h:



This graph shows which files directly or indirectly include this file:



Classes

- class CCircuit

  CCircuit A class storing various information, from the initial economic condition, number of units to the mass flows between them as well as the monetary_value calculated.

Functions

- void mark_units (int unit_num)

8.1.1 Function Documentation

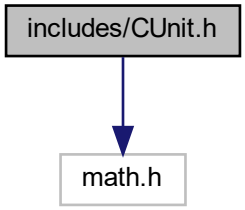
8.1.1.1 mark_units()

```
void mark_units (
            int unit_num )
```

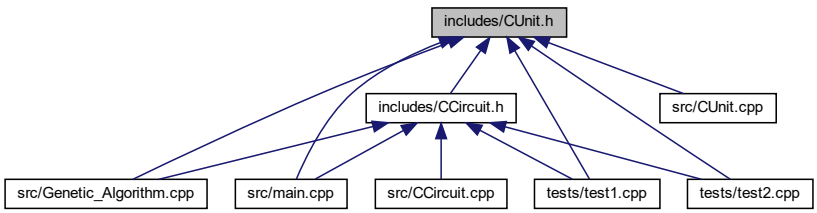
8.2 includes/CUnit.h File Reference

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

Include dependency graph for CUnit.h:



This graph shows which files directly or indirectly include this file:



Classes

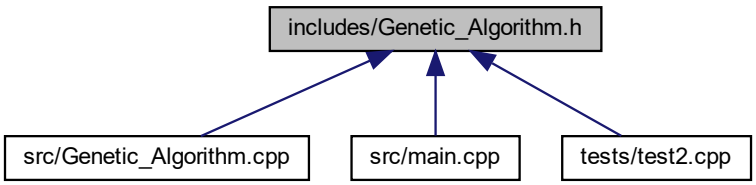
- class Flow

  Flow Representing the mass flow between units.
- class CUnit

  CUnit The class for a certain unit in the Circuit.

8.3 includes/Genetic_Algorithm.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class GeneticAlgorithm

  Genetic algorithm Optimise a system represented by a specification vector.

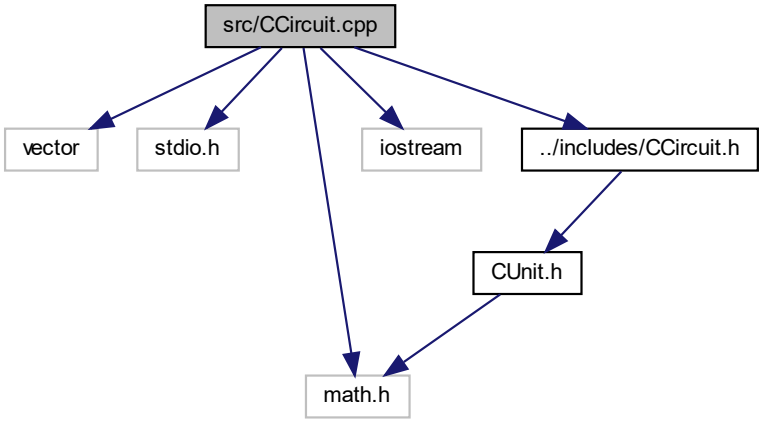
8.4 README.md File Reference

8.5 src/CCircuit.cpp File Reference

```
#include <vector>
#include <stdio.h>
#include <math.h>
#include <iostream>
```

```
#include "../includes/CCircuit.h"
```

Include dependency graph for CCircuit.cpp:



Classes

- struct node

Typedefs

- typedef struct node node

Functions

- void check_tailing (int tail, node *nodes)

  this function is to check whether the units connect to the tailing
- void check_concentrate (int con, node *nodes)

  this function is to check whether the units connect to the tailing

8.5.1 Typedef Documentation

8.5.1.1 node

```
typedef struct node node
```

8.5.2 Function Documentation

8.5.2.1 check_concentrate()

```
void check_concentrate (
            int con,
            node * nodes )
```

this function is to check whether the units connect to the tailing

Parameters

concentrate	concentrate number to locate the concentrate node
nodes	address of nodes array

Returns

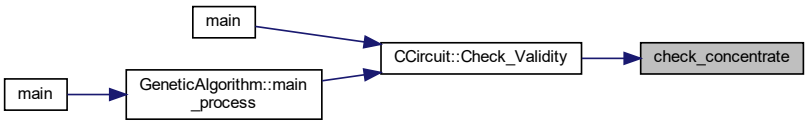
stop while backs to the concentrate

Note

this function backs recursively check the parents units, start from tail, and mark parent units true which indicates that unit has connected to concentrate

Definition at line 293 of file CCircuit.cpp.

Here is the caller graph for this function:



8.5.2.2 check_tailing()

```
void check_tailing (
            int tail,
            node * nodes )
```

this function is to check whether the units connect to the tailing

Parameters

tail	tail number to locate the tail node
nodes	address of nodes array

Returns

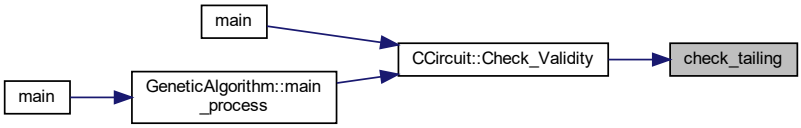
stop while backs to the tail

Note

this function backs recursively check the parents units, start from tail, and mark parent units true which indicates that unit has connected to tail

Definition at line 272 of file CCircuit.cpp.

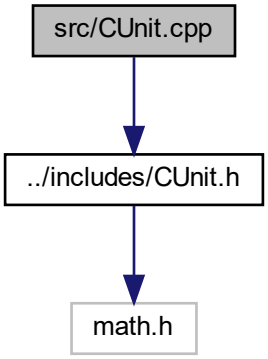
Here is the caller graph for this function:



8.6 src/CUnit.cpp File Reference

```
#include "../includes/CUnit.h"
```

Include dependency graph for CUnit.cpp:

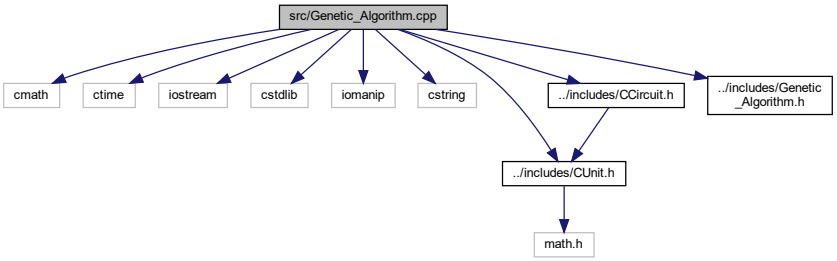


8.7 src/Genetic_Algorithm.cpp File Reference

```
#include <cmath>
#include <ctime>
```

```
#include <iostream>
#include <cstdlib>
#include <iomanip>
#include <cstring>
#include "../includes/CUnit.h"
#include "../includes/CCircuit.h"
#include "../includes/Genetic_Algorithm.h"
```

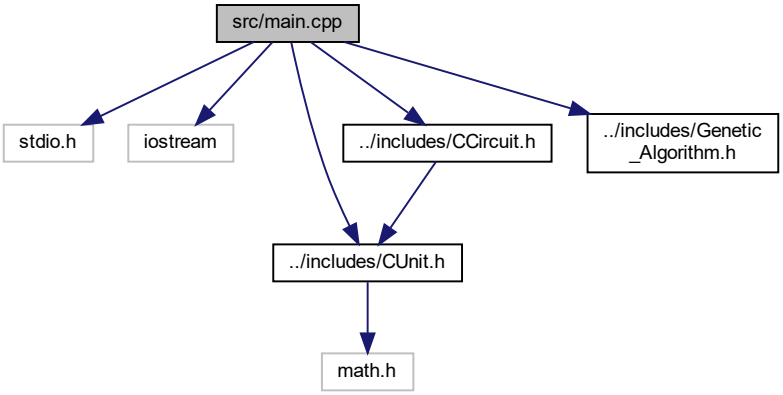
Include dependency graph for Genetic_Algorithm.cpp:



8.8 src/main.cpp File Reference

```
#include <stdio.h>
#include <iostream>
#include "../includes/CUnit.h"
#include "../includes/CCircuit.h"
#include "../includes/Genetic_Algorithm.h"
```

Include dependency graph for main.cpp:



Functions

- int main (int argc, char ∗argv[ ])

8.8.1 Function Documentation

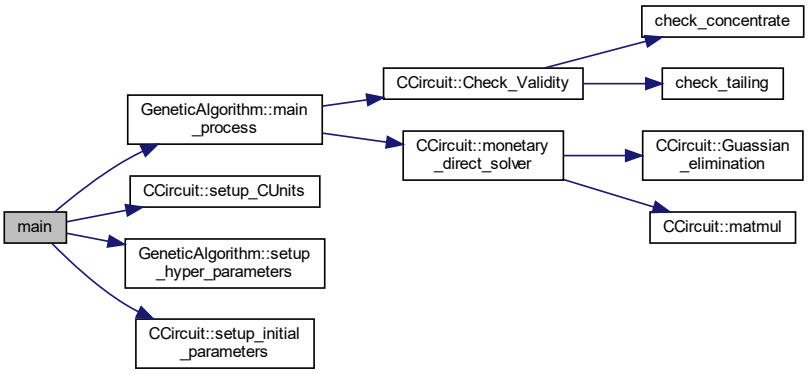
8.8.1.1 main()

```
int main (
            int argc,
            char * argv[] )
```

the number of unit

Definition at line 13 of file main.cpp.

Here is the call graph for this function:



8.9 tests/run_tests.py File Reference

Namespaces

- run_tests

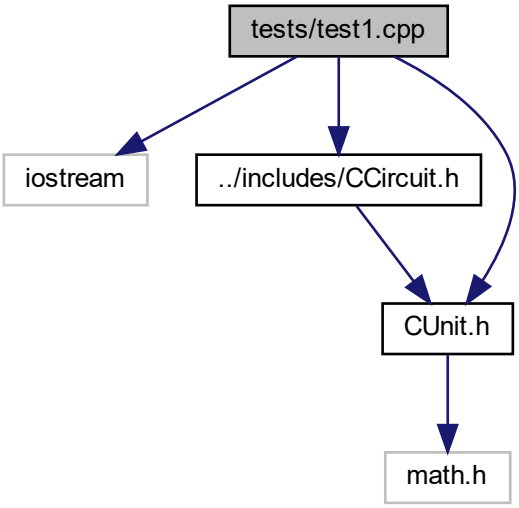
Variables

- string run_tests.DIR = "tests/bin/∗"
- run_tests.files = glob.glob(DIR)
- bool run_tests.glob_fail = False
- run_tests.output = subprocess.run([file], stdout=subprocess.PIPE, universal_newlines=True)
- bool run_tests.test_failed = False

8.10 tests/test1.cpp File Reference

```
#include <iostream>
#include "../includes/CCircuit.h"
#include "../includes/CUnit.h"
```

Include dependency graph for test1.cpp:



Functions

- int main (int argc, char ∗argv[ ])

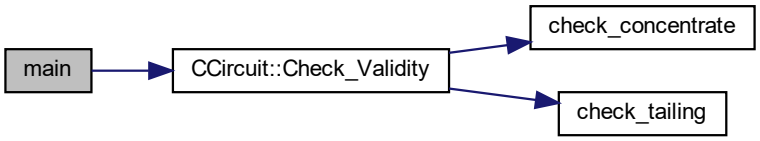
8.10.1 Function Documentation

8.10.1.1 main()

```
int main (
            int argc,
            char * argv[] )
```

Definition at line 8 of file test1.cpp.

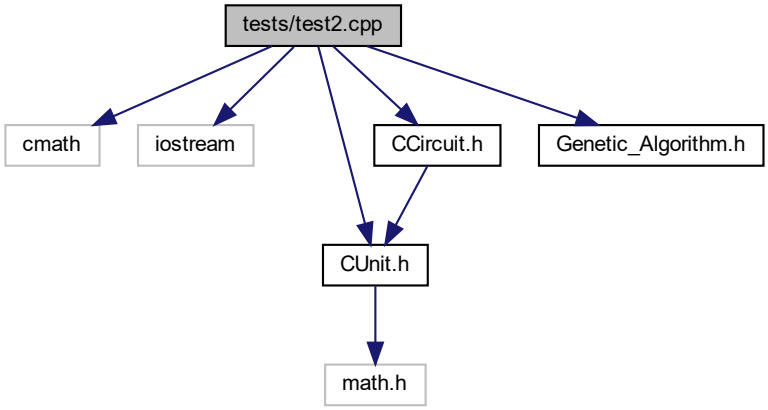
Here is the call graph for this function:



8.11 tests/test2.cpp File Reference

```
#include <cmath>
#include <iostream>
#include "CUnit.h"
#include "CCircuit.h"
#include "Genetic_Algorithm.h"
```

Include dependency graph for test2.cpp:



Functions

- int main (int argc, char ∗argv[ ])

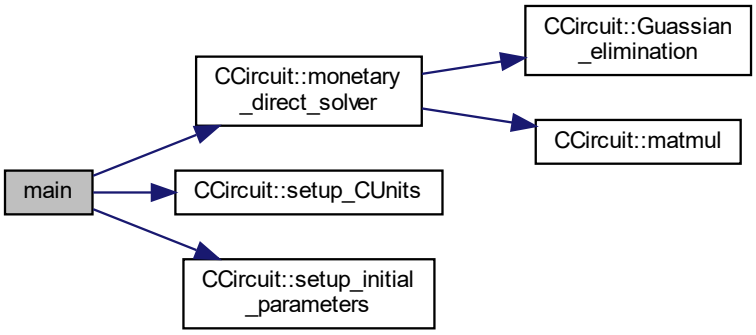
8.11.1 Function Documentation

8.11.1.1 main()

```
int main (
              int argc,
              char * argv[] )
```

Definition at line 9 of file test2.cpp.

Here is the call graph for this function:



8.12 visualization/graph.py File Reference

Namespaces

- graph

Functions

- def graph.loadDataset (file, k)
- def graph.plot (output)

Variables

- string graph.output1 = 'output.txt'
- int graph.k = 21
- list graph.final = [y for x in output1 for y in x]
- graph.verbose

8.13 visualization/README_graph.md File Reference

Index