PSS_PetlaRegulacji_Smieja

final

Generated by Doxygen 1.9.7

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

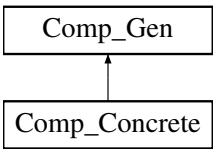
Class Documentation

4.1 Comp_Concrete Class Reference

class of concrete component (basic input) for generator decorator

`#include <comp_concrete.hpp>`

Inheritance diagram for Comp_Concrete:



Public Member Functions

- **Comp_Concrete** (double value)
- double simulate () override

- virtual double simulate ()=0

Private Attributes

- double **m_value**

4.1.1 Detailed Description

class of concrete component (basic input) for generator decorator

4.1.2 Member Function Documentation

4.1.2.1 simulate()

```
double Comp_Concrete::simulate ( )  [inline], [override], [virtual]
```

Implements Comp_Gen.

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

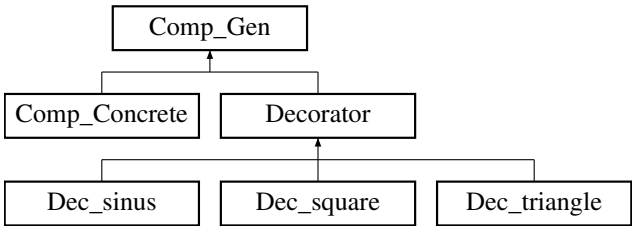
- comp_concrete/comp_concrete.hpp

4.2 Comp_Gen Class Reference

class of generator decorator component

```
#include <gen.hpp>
```

Inheritance diagram for Comp_Gen:



Public Member Functions

- virtual double simulate ()=0

4.2.1 Detailed Description

class of generator decorator component

4.2.2 Member Function Documentation

4.2.2.1 simulate()

```
virtual double Comp_Gen::simulate ( )  [pure virtual]
```

Implemented in Dec_sinus, Dec_square, and Dec_triangle.

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

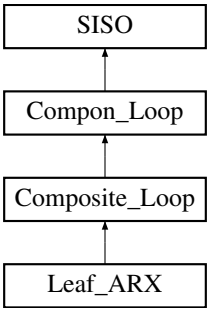
- gen/gen.hpp

4.3 Compon_Loop Class Reference

The base Component class declares common operations for composites and leafs.

`#include <compon_loop.hpp>`

Inheritance diagram for Compon_Loop:



Public Member Functions

- virtual void **add_series** (SISO ∗component)=0
- virtual void **add_pararell** (SISO ∗component)=0
- virtual void **remove_series** (SISO ∗component)=0
- virtual void **remove_pararell** (SISO ∗component)=0
- virtual double simulate (double input)=0
- virtual double **simulate_series** (double input)=0
- virtual double **simulate_pararell** (double input)=0

- virtual double **simulate** (double x)=0

4.3.1 Detailed Description

The base Component class declares common operations for composites and leafs.

4.3.2 Member Function Documentation

4.3.2.1 simulate()

```
virtual double Compon_Loop::simulate (
            double input )  [pure virtual]
```

Implements SISO.

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

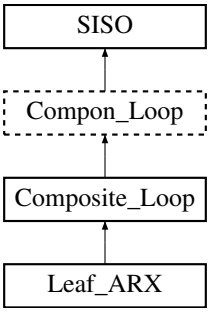
- compon_loop/compon_loop.hpp

4.4 Composite_Loop Class Reference

class od control loop composite in a Composite (structural design pattern)

`#include <composite_loop.hpp>`

Inheritance diagram for Composite_Loop:



Public Member Functions

- void add_series (SISO ∗component) override
- void add_pararell (SISO ∗component) override
- void remove_series (SISO ∗component) override
- void remove_pararell (SISO ∗component) override
- virtual double simulate (double input) override
- double simulate_pararell (double input) override
- double simulate_series (double input) override

- virtual void **add_series** (SISO ∗component)=0
- virtual void **add_pararell** (SISO ∗component)=0
- virtual void **remove_series** (SISO ∗component)=0
- virtual void **remove_pararell** (SISO ∗component)=0
- virtual double simulate (double input)=0
- virtual double **simulate_series** (double input)=0
- virtual double **simulate_pararell** (double input)=0

- virtual double **simulate** (double x)=0

Private Attributes

- double **m_input**
- double **m_out_pararell**
- std::list< SISO ∗ > **m_children_pararell**
- std::list< SISO ∗ > **m_children_series**

4.4.1 Detailed Description

class od control loop composite in a Composite (structural design pattern)

The Composite class represents the complex components that may have children. In ¨normal¨ control loop are 3 available components: main control loop object (tree), group of Leaf_ARX objects in pararell (branch), group of Leaf_ARX objects in series (branch). Each group of ARX objects (branches) can have few Leaf_ARX objects (leafs). There is also one PID controller (leaf for tree), that is the first element of the control loop

If in storage goes new group (branch), then it should be operated by add_series(), remove_series(). It's because main SISO is simulated by simulating groups in series Output can be calculated by simulate() or simulate_series() - it doesn't matter

4.4.2 Member Function Documentation

4.4.2.1 add_pararell()

```
void Composite_Loop::add_pararell (
            SISO * component )  [inline], [override], [virtual]
```

Implements Compon_Loop.

4.4.2.2 add_series()

```
void Composite_Loop::add_series (
            SISO * component )  [inline], [override], [virtual]
```

Implements Compon_Loop.

4.4.2.3 remove_pararell()

```
void Composite_Loop::remove_pararell (
            SISO * component )  [inline], [override], [virtual]
```

Implements Compon_Loop.

4.4.2.4 remove_series()

```
void Composite_Loop::remove_series (
            SISO * component )  [inline], [override], [virtual]
```

Implements Compon_Loop.

4.4.2.5 simulate()

```
double Composite_Loop::simulate (
            double input )  [override], [virtual]
```

Implements Compon_Loop.

4.4.2.6 simulate_pararell()

```
double Composite_Loop::simulate_pararell (
            double input )  [override], [virtual]
```

Implements Compon_Loop.

4.4.2.7 simulate_series()

```
double Composite_Loop::simulate_series (
            double input )  [override], [virtual]
```

Implements Compon_Loop.

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

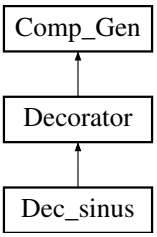
- composite_loop/composite_loop.hpp
- composite_loop/composite_loop.cpp

4.5 Dec_sinus Class Reference

class of sinus decorator for input generator

```
#include <sinus.hpp>
```

Inheritance diagram for Dec_sinus:



Public Member Functions

- **Dec_sinus** (Comp_Gen *generator, double amplitude, double period)
- double simulate () override

  return value - initial input decorated by sinus generator

Public Member Functions inherited from Decorator

- **Decorator** (Comp_Gen *generator)
- double **generate** () override
- **Decorator** (Comp_Gen *generator)
- double simulate () override

- virtual double simulate ()=0

Private Attributes

- double **m_amplitude**
- double **m_period**
- double **m_time**
- double **output**

Additional Inherited Members

Protected Attributes inherited from Decorator

- Comp_Gen * **m_generator**

4.5.1 Detailed Description

class of sinus decorator for input generator

4.5.2 Member Function Documentation

4.5.2.1 simulate()

```
double Dec_sinus::simulate ( )  [override], [virtual]
```

return value - initial input decorated by sinus generator

Returns

double

Implements Comp_Gen.

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

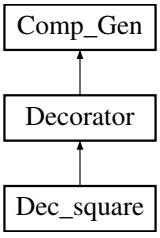
- sinus/sinus.hpp
- sinus/sinus.cpp

4.6 Dec_square Class Reference

class of square decorator for input generator

```
#include <square.hpp>
```

Inheritance diagram for Dec_square:



Public Member Functions

- **Dec_square** (Comp_Gen ∗generator, double amplitude, double period)
- double simulate () override

  return value - initial input decorated by square generator

Public Member Functions inherited from Decorator

- **Decorator** (Comp_Gen ∗generator)
- double **generate** () override
- **Decorator** (Comp_Gen ∗generator)
- double simulate () override

- virtual double simulate ()=0

Private Attributes

- double **m_amplitude**
- double **m_period**
- double **m_time**
- double **output**

Additional Inherited Members

Protected Attributes inherited from Decorator

- Comp_Gen ∗ **m_generator**

4.6.1 Detailed Description

class of square decorator for input generator

4.6.2 Member Function Documentation

4.6.2.1 simulate()

```
double Dec_square::simulate ( )  [override], [virtual]
```

return value - initial input decorated by square generator

Returns

double

Implements Comp_Gen.

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

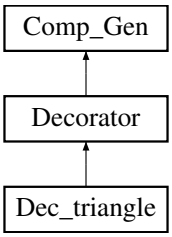
- square/square.hpp
- square/square.cpp

4.7 Dec_triangle Class Reference

class of triangle decorator for input generator

`#include <triangle.hpp>`

Inheritance diagram for Dec_triangle:



Public Member Functions

- **Dec_triangle** (Comp_Gen ∗generator, double amplitude, double period)
- double simulate () override

  return value - initial input decorated by triangle generator

Public Member Functions inherited from Decorator

- **Decorator** (Comp_Gen ∗generator)
- double **generate** () override
- **Decorator** (Comp_Gen ∗generator)
- double simulate () override

- virtual double simulate ()=0

Private Attributes

- double **m_amplitude**
- double **m_period**
- double **m_time**
- double **output**
- double **prev_output**

Additional Inherited Members

Protected Attributes inherited from Decorator

- Comp_Gen ∗ **m_generator**

4.7.1 Detailed Description

class of triangle decorator for input generator

4.7.2 Member Function Documentation

4.7.2.1 simulate()

```
double Dec_triangle::simulate ( )  [override], [virtual]
```

return value - initial input decorated by triangle generator

Returns

double

Implements Comp_Gen.

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

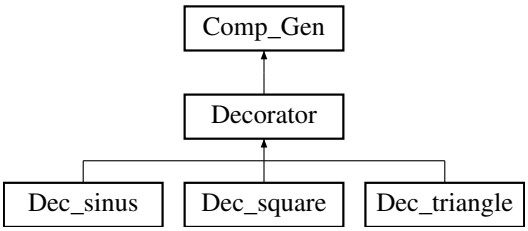
- triangle/triangle.hpp
- triangle/triangle.cpp

4.8 Decorator Class Reference

class of generator basic-decorator

```
#include <gen.hpp>
```

Inheritance diagram for Decorator:



Public Member Functions

- **Decorator** (Comp_Gen ∗generator)
- double **generate** () override
- **Decorator** (Comp_Gen ∗generator)
- double simulate () override

- virtual double simulate ()=0

Protected Attributes

- Comp_Gen ∗ **m_generator**

4.8.1 Detailed Description

class of generator basic-decorator

4.8.2 Member Function Documentation

4.8.2.1 simulate()

```
double Decorator::simulate ( )  [inline], [override], [virtual]
```

Implements Comp_Gen.

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

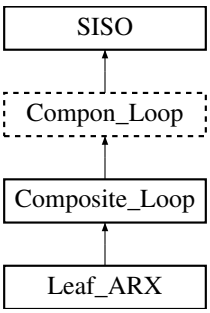
- decorator/decorator.hpp
- gen/gen.hpp

4.9 Leaf_ARX Class Reference

class of an ARX object (leaf) in a Composite (structural design pattern)

```
#include <leaf_arx.hpp>
```

Inheritance diagram for Leaf_ARX:



Public Types

- using **vector** = std::vector< double >
- using **state** = std::deque< double >

Public Member Functions

- Leaf_ARX (const vector &a, const vector &b, unsigned k=0, double evar=0.0)

  Construct a new ARX object.
- double simulate (double input) override
- void **save** (const std::string &path)

Public Member Functions inherited from Composite_Loop

- void add_series (SISO ∗component) override
- void add_pararell (SISO ∗component) override
- void remove_series (SISO ∗component) override
- void remove_pararell (SISO ∗component) override
- virtual double simulate (double input) override
- double simulate_pararell (double input) override
- double simulate_series (double input) override

- virtual void **add_series** (SISO ∗component)=0
- virtual void **add_pararell** (SISO ∗component)=0
- virtual void **remove_series** (SISO ∗component)=0
- virtual void **remove_pararell** (SISO ∗component)=0
- virtual double simulate (double input)=0
- virtual double **simulate_series** (double input)=0
- virtual double **simulate_pararell** (double input)=0

- virtual double **simulate** (double x)=0

Private Member Functions

- void **create_states** ()
- void **update_state** (state &state, double new_state)

Private Attributes

- vector **m_a**
- vector **m_b**
- state **m_x**
- state **m_y**
- double **m_evar**
- unsigned **m_k**
- unsigned **m_x_depth**
- unsigned **m_y_depth**

4.9.1 Detailed Description

class of an ARX object (leaf) in a Composite (structural design pattern)

Few Leaf_ARX objects can be grouped in pararell or in series. All of the ARX groups (composites) make one SISO object, which can be inserted into contorl loop with input generator and PID controller

4.9.2 Constructor & Destructor Documentation

4.9.2.1 Leaf_ARX()

```
Leaf_ARX::Leaf_ARX (
            const vector & a,
            const vector & b,
            unsigned k = 0,
            double evar = 0.0 )
```

Construct a new ARX object.

Parameters

a	coef's of A polynomial (denominator)
b	coef's of B polynomial (numerator)
k	delay
evar	white noise variation

4.9.3 Member Function Documentation

4.9.3.1 simulate()

```
double Leaf_ARX::simulate (
            double input )  [override], [virtual]
```

Reimplemented from Composite_Loop.

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

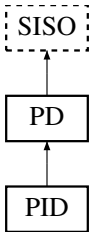
- leaf_arx/leaf_arx.hpp
- leaf_arx/leaf_arx.cpp

4.10 PD Class Reference

class for PD controller

```
#include <pd.hpp>
```

Inheritance diagram for PD:



Public Member Functions

- PD (double min=0.0, double max=100.0, double dt=0.1, double kp=1.0, double kd=0.05)
  Construct a new PD::PD object.
- virtual double simulate (double pv) override
- virtual double **calc_error** (double pv, double set_point)
- virtual double **calc_prop** (double error)
- virtual double **calc_deriv** (double error)
- virtual void **set_setPoint** (double sp)

- virtual double **simulate** (double x)=0

Private Attributes

- double **set_point**
- double **m_min**
- double **m_max**
- double **m_dt**
- double **m_Kp**
- double **m_Ki**
- double **m_Kd**
- double **integral_temp**
- double **prev_error**

4.10.1 Detailed Description

class for PD controller

4.10.2 Constructor & Destructor Documentation

4.10.2.1 PD()

```
PD::PD (
            double min = 0.0,
            double max = 100.0,
            double dt = 0.1,
            double kp = 1.0,
            double kd = 0.05 )
```

Construct a new PD::PD object.

Parameters

min	minimum controller output
max	maximum controller output
dt	time step
kp	proportional coef
kd	derivative coef

4.10.3 Member Function Documentation

4.10.3.1 simulate()

```
double PD::simulate (
            double pv )  [override], [virtual]
```

Implements SISO.

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

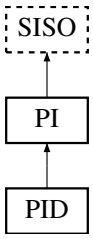
- pd/pd.hpp
- pd/pd.cpp

4.11 PI Class Reference

class for PI controller

`#include <pi.hpp>`

Inheritance diagram for PI:



Public Member Functions

- PI (double min=0.0, double max=100.0, double dt=0.1, double kp=1.0, double ki=0.5)
  Construct a new PI::PI object.
- virtual double simulate (double pv) override
- virtual double **calc_error** (double pv, double set_point)
- virtual double **calc_prop** (double error)
- virtual double **calc_integr** (double error)
- virtual void **set_setPoint** (double sp)

- virtual double **simulate** (double x)=0

Private Attributes

- double **set_point**
- double **m_min**
- double **m_max**
- double **m_dt**
- double **m_Kp**
- double **m_Ki**
- double **m_Kd**
- double **integral_temp**

4.11.1 Detailed Description

class for PI controller

4.11.2 Constructor & Destructor Documentation

4.11.2.1 PI()

```
PI::PI (
            double min = 0.0,
            double max = 100.0,
            double dt = 0.1,
            double kp = 1.0,
            double ki = 0.5 )
```

Construct a new PI::PI object.

Parameters

min	minimum controller output
max	maximum controller output
dt	time step
kp	proportional coef
ki	integral coef

4.11.3 Member Function Documentation

4.11.3.1 simulate()

```
double PI::simulate (
            double pv )  [override], [virtual]
```

Implements SISO.

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

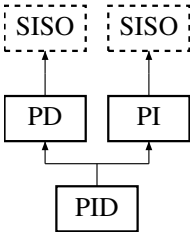
- pi/pi.hpp
- pi/pi.cpp

4.12 PID Class Reference

class for PID controller

```
#include <pid.hpp>
```

Inheritance diagram for PID:



Public Member Functions

- PID (double min=0.0, double max=100.0, double dt=0.1, double kp=1.0, double ki=0.5, double kd=0.05)
  Construct a new PID::PID object.
- double calc_error (double pv, double set_point) override
- double calc_prop (double error) override
- double calc_integr (double error) override
- double calc_deriv (double error) override
- void set_setPoint (double sp) override
- double simulate (double pv) override

Public Member Functions inherited from PD

- PD (double min=0.0, double max=100.0, double dt=0.1, double kp=1.0, double kd=0.05)

  Construct a new PD::PD object.
- virtual double simulate (double pv) override
- virtual double **calc_error** (double pv, double set_point)
- virtual double **calc_prop** (double error)
- virtual double **calc_deriv** (double error)
- virtual void **set_setPoint** (double sp)

- virtual double **simulate** (double x)=0

Public Member Functions inherited from PI

- PI (double min=0.0, double max=100.0, double dt=0.1, double kp=1.0, double ki=0.5)

  Construct a new PI::PI object.
- virtual double simulate (double pv) override
- virtual double **calc_error** (double pv, double set_point)
- virtual double **calc_prop** (double error)
- virtual double **calc_integr** (double error)
- virtual void **set_setPoint** (double sp)

Private Attributes

- double **set_point**
- double **m_min**
- double **m_max**
- double **m_dt**
- double **m_Kp**
- double **m_Ki**
- double **m_Kd**
- double **integral_temp**
- double **prev_error**

4.12.1 Detailed Description

class for PID controller

4.12.2 Constructor & Destructor Documentation

4.12.2.1 PID()

```
PID::PID (
            double min = 0.0,
            double max = 100.0,
            double dt = 0.1,
            double kp = 1.0,
            double ki = 0.5,
            double kd = 0.05 )
```

Construct a new PID::PID object.

Parameters

min	minimum controller output
max	maximum controller output
dt	time step
kp	proportional coef
ki	integral coef
kd	derivative coef

4.12.3 Member Function Documentation

4.12.3.1 calc_deriv()

```
double PID::calc_deriv (
            double error )  [override], [virtual]
```

Reimplemented from PD.

4.12.3.2 calc_error()

```
double PID::calc_error (
            double pv,
            double set_point )  [override], [virtual]
```

Reimplemented from PD.

4.12.3.3 calc_integr()

```
double PID::calc_integr (
            double error )  [override], [virtual]
```

Reimplemented from PI.

4.12.3.4 calc_prop()

```
double PID::calc_prop (
            double error )  [override], [virtual]
```

Reimplemented from PD.

4.12.3.5 set_setPoint()

```
void PID::set_setPoint (
            double sp )  [override], [virtual]
```

Reimplemented from PD.

4.12.3.6 simulate()

```
double PID::simulate (
            double pv )  [override], [virtual]
```

Reimplemented from PD.

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

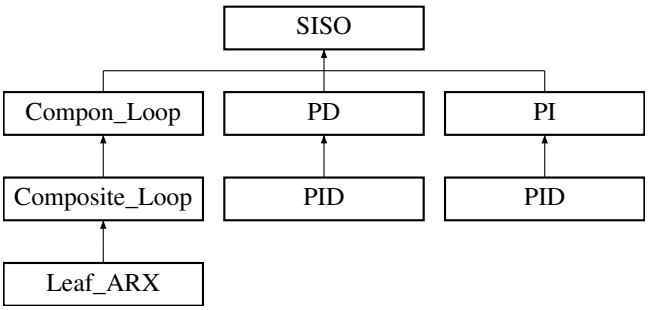
- pid/pid.hpp
- pid/pid.cpp

4.13 SISO Class Reference

basic class for SISO objects

```
#include <siso.hpp>
```

Inheritance diagram for SISO:



Public Member Functions

- virtual double **simulate** (double x)=0

4.13.1 Detailed Description

basic class for SISO objects

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

- siso/siso.hpp

Chapter 5

File Documentation

5.1 comp_concrete/comp_concrete.hpp File Reference

class of concrete component (basic input) for generator decorator

`#include ¨../gen/gen.hpp¨`

Classes

- class Comp_Concrete

  class of concrete component (basic input) for generator decorator

5.1.1 Detailed Description

class of concrete component (basic input) for generator decorator

Author

Pawel Smieja

Version

0.1

Date

2023-05-31

Copyright

Copyright (c) 2023

5.2 comp_concrete.hpp

Go to the documentation of this file.

```
00001
00011 #ifndef COMP_CONCRETE_H
00012 #define COMP_CONCRETE_H
00013
00014 #include "../gen/gen.hpp"
00015
00020 class Comp_Concrete: public Comp_Gen
00021 {
00022 public:
00023     Comp_Concrete(double value) : m_value(value) {}
00024     ~Comp_Concrete() = default;
00025
00026     double simulate() override { return m_value; }
00027
00028 private:
00029     double m_value;
00030 };
00031
00032 #endif
```

5.3 compon_loop/compon_loop.hpp File Reference

class of control loop component in a Composite (structural design pattern)

```
#include <algorithm>
#include <iostream>
#include ¨../siso/siso.hpp¨
```

Classes

- class Compon_Loop

  The base Component class declares common operations for composites and leafs.

5.3.1 Detailed Description

class of control loop component in a Composite (structural design pattern)

Author

Pawel Smieja

Version

0.1

Date

2023-06-04

Copyright

Copyright (c) 2023

5.4 compon_loop.hpp

Go to the documentation of this file.

```
00001
00011 #ifndef COMPON_LOOP_H
00012 #define COMPON_LOOP_H
00013
00014 #include <algorithm>
00015 #include <iostream>
00016 #include "../siso/siso.hpp"
00017
00022 class Compon_Loop : public SISO
00023 {
00024 public:
00025     virtual ~Compon_Loop() = default;
00026     // add new component
00027     virtual void add_series(SISO *component) = 0;
00028     virtual void add_pararell(SISO *component) = 0;
00029     // remove component
00030     virtual void remove_series(SISO *component) = 0;
00031     virtual void remove_pararell(SISO *component) = 0;
00032
00033     // get output of the component/leaf
00034     virtual double simulate(double input) = 0;
00035     // for component, where leafs are in series
00036     virtual double simulate_series(double input) = 0;
00037     // for component, where leafs are in pararell
00038     virtual double simulate_pararell(double input) = 0;
00039 };
00040
00041 #endif
```

5.5 composite_loop/composite_loop.cpp File Reference

class of control loop composite in a Composite (structural design pattern).

```
#include ¨composite_loop.hpp¨
#include <iostream>
```

5.5.1 Detailed Description

class of control loop composite in a Composite (structural design pattern).

Author

Pawel Smieja

Version

0.1

Date

2023-06-01

Copyright

5.6 composite_loop/composite_loop.hpp File Reference

class od control loop composite in a Composite (structural design pattern)

```
#include ¨../compon_loop/compon_loop.hpp¨
#include <list>
```

Classes

- class Composite_Loop

  class od control loop composite in a Composite (structural design pattern)

5.6.1 Detailed Description

class od control loop composite in a Composite (structural design pattern)

Author

Pawel Smieja

Version

0.1

Date

2023-06-01

Copyright

Copyright (c) 2023

5.7 composite_loop.hpp

Go to the documentation of this file.

```
00001
00011 #ifndef COMPOSITE_LOOP_H
00012 #define COMPOSITE_LOOP_H
00013
00014 #include "../compon_loop/compon_loop.hpp"
00015 #include <list>
00016
00033 class Composite_Loop : virtual public Compon_Loop
00034 {
00035 public:
00036     // add new component
00037     void add_series(SISO *component) override
00038     {
00039         m_children_series.push_back(component);
00040     }
00041     void add_pararell(SISO *component) override
00042     {
00043         m_children_pararell.push_back(component);
00044     }
00045     // remove component
```

```
00046     void remove_series(SISO *component) override { m_children_series.remove(component); }
00047     void remove_pararell(SISO *component) override { m_children_pararell.remove(component); }
00048
00049     // get output of the component
00050     virtual double simulate(double input) override;
00051     // for siso composite, where leafs are in pararell
00052     double simulate_pararell(double input) override;
00053     // for siso composite, where leafs are in series
00054     double simulate_series(double input) override;
00055
00056 private:
00057     double m_input, m_out_pararell;
00058
00059     std::list<SISO *> m_children_pararell;
00060     std::list<SISO *> m_children_series;
00061 };
00062
00063 #endif
```

5.8 decorator.hpp

```
00001 #ifndef DECORATOR_H
00002 #define DECORATOR_H
00003
00004 //#include "../gen/gen.hpp"
00005
00006 class Decorator : public Comp_Gen
00007 {
00008 public:
00009     Decorator(Comp_Gen *generator) : m_generator(generator) {}
00010     ~Decorator() = default;
00011
00012     double generate() override
00013     {
00014         return m_generator->generate();
00015     }
00016
00017 protected:
00018     Comp_Gen *m_generator;
00019 };
00020
00021 #endif
```

5.9 gen/gen.hpp File Reference

Classes

- class Comp_Gen

  class of generator decorator component
- class Decorator

  class of generator basic-decorator

5.9.1 Detailed Description

Author

Pawel Smieja

Version

0.1

Date

2023-05-31

Copyright

Copyright (c) 2023

5.10 gen.hpp

Go to the documentation of this file.

```
00001
00011 #ifndef GEN_H
00012 #define GEN_H
00013
00018 class Comp_Gen
00019 {
00020 public:
00021     virtual ~Comp_Gen() = default;
00022     virtual double simulate() = 0;
00023 };
00024
00029 class Decorator : public Comp_Gen
00030 {
00031 public:
00032     Decorator(Comp_Gen *generator) : m_generator(generator) {}
00033     ~Decorator() = default;
00034
00035     double simulate() override
00036     {
00037         return m_generator->simulate();
00038     }
00039
00040 protected:
00041     Comp_Gen *m_generator;
00042 };
00043
00044 #endif
```

5.11 leaf_arx/leaf_arx.cpp File Reference

class of a leaf object in a Composite (structural design pattern).

```
#include ¨leaf_arx.hpp¨
#include <numeric>
#include <fstream>
```

5.11.1 Detailed Description

class of a leaf object in a Composite (structural design pattern).

Author

Pawel Smieja

Version

0.1

Date

2023-06-01

Copyright

5.12 leaf_arx/leaf_arx.hpp File Reference

class of an ARX object (leaf) in a Composite (structural design pattern)

```
#include ¨../composite_loop/composite_loop.hpp¨
#include <vector>
#include <deque>
#include <string>
```

Classes

- class Leaf_ARX

  class of an ARX object (leaf) in a Composite (structural design pattern)

5.12.1 Detailed Description

class of an ARX object (leaf) in a Composite (structural design pattern)

Author

Pawel Smieja

Version

0.1

Date

2023-06-01

Copyright

5.13 leaf_arx.hpp

Go to the documentation of this file.

```
00001
00011 #ifndef LEAF_ARX_H
00012 #define LEAF_ARX_H
00013
00014 #include "../composite_loop/composite_loop.hpp"
00015 #include <vector>
00016 #include <deque>
00017 #include <string>
00018
00029 class Leaf_ARX : public Composite_Loop
00030 {
00031 public:
00032     using vector = std::vector<double>;
00033     using state = std::deque<double>;
00034
00043     Leaf_ARX(const vector &a, const vector &b, unsigned k = 0, double evar = 0.0);
00044     ~Leaf_ARX() = default;
```

```
00045
00046     // simulate output of the ARX object
00047     double simulate(double input) override;
00048     // save current parameters to a CSV file
00049     void save(const std::string &path);
00050
00051 private:
00052     vector m_a, m_b;
00053     state m_x, m_y;
00054     double m_evar;
00055     unsigned m_k, m_x_depth, m_y_depth;
00056
00057     void create_states();
00058     void update_state(state &state, double new_state);
00059 };
00060
00061 #endif
```

5.14 main.cpp File Reference

program for testing simulation closed control loop for: generator - PID controller - SISO composite

```
#include ¨compon_loop/compon_loop.hpp¨
#include ¨composite_loop/composite_loop.hpp¨
#include ¨pi/pi.hpp¨
#include ¨pd/pd.hpp¨
#include ¨pid/pid.hpp¨
#include ¨leaf_arx/leaf_arx.hpp¨
#include ¨comp_concrete/comp_concrete.hpp¨
#include ¨square/square.hpp¨
#include ¨sinus/sinus.hpp¨
#include ¨triangle/triangle.hpp¨
#include <iostream>
#include <iomanip>
```

Functions

- int main ()

  program for testing simulation closed control loop for: generator - PID controller - SISO composite

5.14.1 Detailed Description

program for testing simulation closed control loop for: generator - PID controller - SISO composite

Author

Pawel Smieja

Version

0.1

Date

2023-05-31

Copyright

Copyright (c) 2023

5.14.2 Function Documentation

5.14.2.1 main()

```
int main ( )
```

program for testing simulation closed control loop for: generator - PID controller - SISO composite

Returns

int

5.15 pd/pd.cpp File Reference

```
#include ¨pd.hpp¨
#include <cmath>
```

5.15.1 Detailed Description

Author

Pawel Smieja

Version

0.1

Date

2023-05-10

Copyright

5.16 pd.hpp

```
00001 #ifndef PD_H
00002 #define PD_H
00003
00004 #include "../siso/siso.hpp"
00005
00010 class PD : virtual public SISO
00011 {
00012 public:
00013     PD(double min = 0.0, double max = 100.0, double dt = 0.1, double kp = 1.0, double kd = 0.05);
00014     ~PD() = default;
00015
00016     virtual double simulate(double pv) override;
00017
00018     virtual double calc_error(double pv, double set_point);
00019     virtual double calc_prop(double error);
00020     virtual double calc_deriv(double error);
00021     virtual void set_setPoint(double sp);
00022
00023 private:
00024     double set_point, m_min, m_max, m_dt, m_Kp, m_Ki, m_Kd, integral_temp, prev_error;
00025 };
00026
00027 #endif
```

5.17 pi/pi.cpp File Reference

```
#include ¨pi.hpp¨
#include <cmath>
```

5.17.1 Detailed Description

Author

Pawel Smieja

Version

0.1

Date

2023-05-10

Copyright

Copyright (c) 2023

5.18 pi/pi.hpp File Reference

```
#include ¨../siso/siso.hpp¨
```

Classes

- class PI
  class for PI controller

5.18.1 Detailed Description

Author

Pawel Smieja

Version

0.1

Date

2023-05-10

Copyright

Copyright (c) 2023

5.19 pi.hpp

Go to the documentation of this file.

```
00001
00011 #ifndef PI_H
00012 #define PI_H
00013
00014 #include "../siso/siso.hpp"
00015
00020 class PI : virtual public SISO
00021 {
00022 public:
00023     PI(double min = 0.0, double max = 100.0, double dt = 0.1, double kp = 1.0, double ki = 0.5);
00024     ~PI() = default;
00025
00026     virtual double simulate(double pv) override;
00027
00028     virtual double calc_error(double pv, double set_point);
00029     virtual double calc_prop(double error);
00030     virtual double calc_integr(double error);
00031     virtual void set_setPoint(double sp);
00032
00033 private:
00034     double set_point, m_min, m_max, m_dt, m_Kp, m_Ki, m_Kd, integral_temp;
00035 };
00036
00037 #endif
```

5.20 pid/pid.cpp File Reference

```
#include ¨pid.hpp¨
#include <cmath>
```

5.20.1 Detailed Description

Author

Pawel Smieja

Version

0.1

Date

2023-05-10

Copyright

5.21 pid.hpp

```
00001 #ifndef PID_H
00002 #define PID_H
00003
00004 #include "../pd/pd.hpp"
00005 #include "../pi/pi.hpp"
00006
00011 class PID : public PD, public PI
00012 {
00013 public:
00014     PID(double min = 0.0, double max = 100.0, double dt = 0.1, double kp = 1.0, double ki = 0.5,
      double kd = 0.05);
00015     ~PID() = default;
00016
00017     double calc_error(double pv, double set_point) override;
00018     double calc_prop(double error) override;
00019     double calc_integr(double error) override;
00020     double calc_deriv(double error) override;
00021     void set_setPoint(double sp) override;
00022     double simulate(double pv) override;
00023
00024 private:
00025     double set_point, m_min, m_max, m_dt, m_Kp, m_Ki, m_Kd, integral_temp, prev_error;
00026 };
00027
00028 #endif
```

5.22 sinus/sinus.cpp File Reference

```
#include <cmath>
#include ¨sinus.hpp¨
```

5.22.1 Detailed Description

Author

Pawel Smieja

Version

0.1

Date

2023-05-31

Copyright

Copyright (c) 2023

5.23 sinus/sinus.hpp File Reference

class of sinus decorator for input generator

```
#include ¨../gen/gen.hpp¨
```

Classes

- class Dec_sinus

  class of sinus decorator for input generator

5.23.1 Detailed Description

class of sinus decorator for input generator

Author

Pawel Smieja

Version

0.1

Date

2023-05-31

Copyright

Copyright (c) 2023

5.24 sinus.hpp

Go to the documentation of this file.

```
00001
00011 #ifndef DEC_SINUS_H
00012 #define DEC_SINUS_H
00013
00014 #include "../gen/gen.hpp"
00015
00020 class Dec_sinus : public Decorator
00021 {
00022 public:
00023     Dec_sinus(Comp_Gen *generator, double amplitude, double period)
00024         : Decorator(generator), m_amplitude(amplitude), m_period(period), m_time(0.0) {}
00025     double simulate() override;
00026
00027 private:
00028     double m_amplitude, m_period, m_time, output;
00029 };
00030
00031 #endif
```

5.25 siso.hpp

```
00001 #ifndef SISO_H
00002 #define SISO_H
00003
00008 class SISO
00009 {
00010 public:
00011     SISO() = default;
00012     virtual ~SISO() = default;
00013
00014     virtual double simulate(double x) = 0;
00015 };
00016
00017 #endif
```

5.26 square/square.cpp File Reference

```
#include ¨square.hpp¨
```

5.26.1 Detailed Description

Author

Pawel Smieja

Version

0.1

Date

2023-05-31

Copyright

Copyright (c) 2023

5.27 square/square.hpp File Reference

class of square decorator for input generator

```
#include ¨../gen/gen.hpp¨
```

Classes

- class Dec_square
  class of square decorator for input generator

5.27.1 Detailed Description

class of square decorator for input generator

Author

Pawel Smieja

Version

0.1

Date

2023-05-31

Copyright

Copyright (c) 2023

5.28 square.hpp

Go to the documentation of this file.

```
00001
00011 #ifndef DEC_SQUARE_H
00012 #define DEC_SQUARE_H
00013
00014 #include "../gen/gen.hpp"
00015
00020 class Dec_square : public Decorator
00021 {
00022 public:
00023     Dec_square(Comp_Gen *generator, double amplitude, double period)
00024         : Decorator(generator), m_amplitude(amplitude), m_period(period), m_time(0.0) {}
00025     double simulate() override;
00026
00027 private:
00028     double m_amplitude, m_period, m_time, output;
00029 };
00030
00031 #endif
```

5.29 triangle/triangle.cpp File Reference

```
#include <cmath>
#include ¨triangle.hpp¨
```

5.29.1 Detailed Description

Author

Pawel Smieja

Version

0.1

Date

2023-05-31

Copyright

Copyright (c) 2023

5.30 triangle/triangle.hpp File Reference

class of triangle decorator for input generator

```
#include ¨../gen/gen.hpp¨
```

Classes

- class Dec_triangle

  class of triangle decorator for input generator

5.30.1 Detailed Description

class of triangle decorator for input generator

Author

Pawel Smieja

Version

0.1

Date

2023-05-31

Copyright

Copyright (c) 2023

5.31 triangle.hpp

Go to the documentation of this file.

```
00001
00011 #ifndef DEC_TRIANGLE_H
00012 #define DEC_TRIANGLE_H
00013
00014 #include "../gen/gen.hpp"
00015
00020 class Dec_triangle : public Decorator
00021 {
00022 public:
00023     Dec_triangle(Comp_Gen *generator, double amplitude, double period)
00024         : Decorator(generator), m_amplitude(amplitude), m_period(period), m_time(0.0), output(0.0),
      prev_output(-m_amplitude) {}
00025     double simulate() override;
00026
00027 private:
00028     double m_amplitude, m_period, m_time, output, prev_output;
00029 };
00030
00031 #endif
```

Index