

---

**nodedge**

*Release 0.2.0*

**Anthony De Bortoli**

**Nov 23, 2022**



# CONTENTS

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Installation . . . . .	3
1.2	Getting started . . . . .	3
<b>2</b>	<b>Library conventions</b>	<b>5</b>
2.1	LTI system representation . . . . .	5
<b>3</b>	<b>nodedge package</b>	<b>7</b>
3.1	Subpackages . . . . .	7
3.2	Submodules . . . . .	9
3.3	Module contents . . . . .	11
<b>4</b>	<b>Examples</b>	<b>13</b>
4.1	Python scripts . . . . .	13
<b>5</b>	<b>Evaluation</b>	<b>15</b>
5.1	Evaluation Functions . . . . .	15
5.2	Node Flags . . . . .	15
<b>6</b>	<b>Event system</b>	<b>17</b>
6.1	Events used in Nodedge: . . . . .	17
<b>7</b>	<b>Serialization</b>	<b>19</b>
	<b>Python Module Index</b>	<b>21</b>
	<b>Index</b>	<b>23</b>



Nodedge is a free and open-source software for graphical programming. Try it out and contribute to its development!

## **Features**

- Create your simulation easily thanks to a large variety of customizable built-in blocks.
- Access your models anywhere anytime. Nodedge is free for any purpose.
- Contribute to Nodedge development. The software is based on Python.

## **Documentation**



## INTRODUCTION

Welcome to the Nodedge User's Manual. This manual contains information on how to use the Nodedge software and includes the documentation for all the modules of the package and examples illustrating their use.

### 1.1 Installation

The *nodedge* package can be installed using pip or the standard distutils/setuptools mechanisms.

To install using pip:

```
pip install nodedge
```

Alternatively, to use setuptools, first [clone or download the source](#). To install in your default python interpreter, use:

```
git clone git@github.com:nodedge/nodedge.git
cd nodedge
python setup.py install --user
```

### 1.2 Getting started

To use nodedge simply import the package as follows:

```
>>> import nodedge
```





## LIBRARY CONVENTIONS

The nodedge library uses a set of standard conventions for the way that different types of standard information used by the library.

### 2.1 LTI system representation

Linear time invariant (LTI) systems are represented in nodedge in state space, transfer function, or frequency response data (FRD) form. Most functions in the toolbox will operate on any of these data types and functions for converting between compatible types is provided.

#### 2.1.1 State space systems

$$\begin{aligned}\frac{dx}{dt} &= Ax + Bu \\ y &= Cx + Du\end{aligned}$$

where  $u$  is the input,  $y$  is the output, and  $x$  is the state.

#### 2.1.2 Transfer functions

$$G(s) = \frac{\text{num}(s)}{\text{den}(s)} = \frac{a_0 s^m + a_1 s^{m-1} + \cdots + a_m}{b_0 s^n + b_1 s^{n-1} + \cdots + b_n},$$

where  $n$  is generally greater than or equal to  $m$  (for a proper transfer function).



## NODEGE PACKAGE

### 3.1 Subpackages

#### 3.1.1 nodedge.blocks package

##### Subpackages

nodedge.blocks.autogen package

##### Subpackages

nodedge.blocks.autogen.operator package

##### Submodules

nodedge.blocks.autogen.operator.add\_block

nodedge.blocks.autogen.operator.eq\_block

nodedge.blocks.autogen.operator.ge\_block

nodedge.blocks.autogen.operator.gt\_block

nodedge.blocks.autogen.operator.le\_block

nodedge.blocks.autogen.operator.lt\_block

nodedge.blocks.autogen.operator.mod\_block

nodedge.blocks.autogen.operator.mul\_block

nodedge.blocks.autogen.operator.pow\_block

nodedge.blocks.autogen.operator.sub\_block

`nodedge.blocks.autogen.operator.truediv_block`

Module contents

Module contents

`nodedge.blocks.custom package`

Submodules

`nodedge.blocks.custom.input_block`

`nodedge.blocks.custom.output_block`

Module contents

Submodules

`nodedge.blocks.block`

`nodedge.blocks.block_config`

`nodedge.blocks.block_exception`

Block exception module containing `EvaluationError`, `MissInputError`, and `RedundantInputError` classes.

**exception** `nodedge.blocks.block_exception.EvaluationError`

Bases: `Exception`

`EvaluationError` class

If a block cannot be evaluated, raise this error.

**exception** `nodedge.blocks.block_exception.MissInputError`

Bases: `EvaluationError`

`MissInputError` class

If an input is missing to a block, preventing it to be evaluated, raise this error.

**exception** `nodedge.blocks.block_exception.RedundantInputError`

Bases: `EvaluationError`

`RedundantInputError` class

If two different inputs are connected to a single input socket of a block, raise this error.

**nodedge.blocks.graphics\_block**

**nodedge.blocks.graphics\_block\_content**

**nodedge.blocks.graphics\_input\_block\_content**

**nodedge.blocks.graphics\_output\_block\_content**

**Module contents**

## **3.2 Submodules**

**3.2.1 nodedge.application\_styler**

**3.2.2 nodedge.connector**

**3.2.3 nodedge.edge**

**3.2.4 nodedge.edge\_dragging**

**3.2.5 nodedge.edge\_validators**

**3.2.6 nodedge.editor\_widget**

**3.2.7 nodedge.editor\_window**

**3.2.8 nodedge.graphics\_cut\_line**

**3.2.9 nodedge.graphics\_edge**

**3.2.10 nodedge.graphics\_node**

**3.2.11 nodedge.graphics\_node\_content**

**3.2.12 nodedge.graphics\_node\_title\_label**

**3.2.13 nodedge.graphics\_scene**

**3.2.14 nodedge.graphics\_socket**

**3.2.15 nodedge.graphics\_view**

**3.2.16 nodedge.history\_list\_widget**

**3.2.17 nodedge.logger**

Logger module containing function to set up logging functionalities.

```
nodedge.logger.setupLogging(defaultPath: str = './logging.yaml', defaultLevel: int = 10, envKey: str = 'LOG_CFG')
```

Logging Setup

```
nodedge.logger.highlightLoggingSetup()
```

### 3.2.18 nodedge.mdi\_area

### 3.2.19 nodedge.mdi\_widget

### 3.2.20 nodedge.mdi\_window

### 3.2.21 nodedge.node

### 3.2.22 nodedge.node\_tree\_widget

### 3.2.23 nodedge.scene

### 3.2.24 nodedge.scene\_clipboard

### 3.2.25 nodedge.scene\_coder

### 3.2.26 nodedge.scene\_history

### 3.2.27 nodedge.scene\_item\_detail\_widget

### 3.2.28 nodedge.scene\_items\_table\_widget

### 3.2.29 nodedge.serializable

Serializable “interface” module. It is an abstract class.

```
class nodedge.serializable.Serializable
```

Bases: object

*Serializable* class

Create data which are common to any serializable object.

It stores the id of the object used in the SceneHistory, the SceneClipboard, and the file structure.

```
serialize() → OrderedDict
```

Serialization method to serialize this class data into OrderedDict which can be stored in memory or file easily.

**Returns**

data serialized in OrderedDict

**Return type**

OrderedDict

**deserialize**(*data: dict, hashmap: Optional[dict] = None, restoreId: bool = True, \*args, \*\*kwargs*) → bool

Deserialization method which take data in python dict format with helping *hashmap* containing references to existing entities.

#### Parameters

- **data** (dict) – dictionary containing serialized data
- **hashmap** (dict) – helper dictionary containing references (by id == key) to existing objects
- **restoreId** (bool) – True if we are creating new sockets. False is useful when loading existing sockets which we want to keep the existing object's *id*

#### Returns

True if deserialization was successful, False otherwise

#### Return type

bool

### 3.2.30 nodedge.socket\_type

<ModuleName> module containing ~nodedge. class.

**class** nodedge.socket\_type.MyClass

Bases: object

~nodedge. class .

**class** nodedge.socket\_type.SocketType(*value*)

Bases: Enum

An enumeration.

**Any** = 0

**Number** = 1

**String** = 2

### 3.2.31 nodedge.utils

## 3.3 Module contents





## **EXAMPLES**

The source code for the examples below are available in the *examples/* subdirectory of the source code distribution. They can also be accessed online via the [nodedge GitHub repository](#).

### **4.1 Python scripts**



## EVALUATION

The evaluation system uses `eval()` and `evalChildren()`. `eval` is supposed to be overridden by your own implementation. The evaluation logic uses Flags for marking the *Nodes* as *Dirty* and/or *Invalid*.

### 5.1 Evaluation Functions

There are 2 main methods used for the evaluation:

- `eval()`
- `evalChildren()`

These functions are mutually exclusive. That means that `evalChildren` does **not** eval the current *Node*, but only children of the current *Node*.

By default the implementation of `eval()` is “empty” and return 0. However, if successful, `eval` resets the *Node* not to be *Dirty* nor *Invalid*. This method is supposed to be overridden by your own implementation. If you look for examples, please check out `examples/example_calculator` to get inspiration on how to setup your own *Node* evaluation.

The evaluation takes advantage of the *Node* flags described below.

### 5.2 Node Flags

Each *Node* has 2 flags:

- *Dirty*
- *Invalid*

The *Invalid* flag has always higher priority over *Dirty*. This means that if the *Node* is *Invalid* it does not matter whether it is *Dirty* or not.

To mark a node as *Dirty* or *Invalid* there are respective methods `markDirty()` and `markInvalid()`. Both methods take *bool* parameter for the new state. You can mark *Node* dirty by setting the parameter to `True`. Also you can un-mark the state by passing `False` value.

For both flags there are 3 methods available:

- `markInvalid()` - to mark only the *Node*
- `markChildrenInvalid()` - to mark only the direct (first level) children of the *Node*
- `markDescendantsInvalid()` - to mark it self and all descendant children of the *Node*

The same applies to the *Dirty* flag:

- `markDirty()` - to mark only the *Node*
- `markChildrenDirty()` - to mark only the direct (first level) children of the *Node*
- `markDescendantsDirty()` - to mark it self and all descendant children of the *Node*

Descendants or Children are always connected to the Output(s) of the current *Node*.

When a node is marked as *Dirty* or *Invalid* one of the two event methods, `onMarkedInvalid()` or `onMarkedDirty()`, is called. By default, these methods do nothing. However, they are implemented for you to override and use them in your own evaluation system.

## EVENT SYSTEM

Nodedge uses its own events (and tries to avoid using `Signal`) to handle logic happening inside the *Scene*. If a class does handle some events, they are usually described at the top of the page in this documentation.

Any of the events is subscribable to and the methods for registering callback are called:

`add<EventName>Listener(callback)`

You can register to any of these events any time.

### 6.1 Events used in Nodedge:

#### 6.1.1 Scene

***Has Been Modified***

when something has changed in the *Scene*

***Item Selected***

when *Node* or *Edge* is selected

***Items Deselected***

when deselect everything appears

***Drag Enter***

when something is Dragged onto the *Scene*. Here we do allow or deny the drag

***Drop***

when we Drop something into the *Scene*

#### 6.1.2 SceneHistory

***History Modified***

after *History Stamp* has been stored or restored

***History Stored***

after *History Stamp* has been stored

***History Restored***

after *History Stamp* has been restored



## SERIALIZATION

All of serializable classes derive from *Serializable* class. *Serializable* does create commonly used parameters for our classes. In our case it is just `id` attribute.

*Serializable* defines two methods which should be overridden in child classes:

- *serialize()*
- *deserialize()*

According to coding-standards we keep these two functions on the bottom of the class source code.

To contain all of the data we use `OrderedDict` instead of regular *dict*. Mainly because we want to retain the order of parameters serialized in files.

Classes which derive from *Serializable*:

- `Scene`
- `Node`
- `QDMNodeContentWidget`
- `Edge`
- `Socket`
- `genindex`

### Development

You can check out the latest version of the source code with the command:

```
git clone https://github.com/nodedge/nodedge.git
```

You can run a set of unit tests to make sure that everything is working correctly. After installation, run:

```
python setup.py test
```

Your contributions are welcome! Simply fork the [GitHub repository](#) and send a [pull request](#).

## **Links**

- Issue tracker: <https://github.com/nodedge/nodedge/issues>



## PYTHON MODULE INDEX

### n

- `nodedge`, [11](#)
- `nodedge.blocks.block_exception`, [8](#)
- `nodedge.logger`, [9](#)
- `nodedge.serializable`, [10](#)
- `nodedge.socket_type`, [11](#)



## INDEX

### A

Any (*nodedge.socket\_type.SocketType* attribute), 11

### D

deserialize() (*nodedge.serializable.Serializable*  
method), 10

### E

EvaluationError, 8

### H

highlightLoggingSetup() (*in module nodedge.logger*), 10

### M

MissInputError, 8

module

    nodedge, 11

    nodedge.blocks.block\_exception, 8

    nodedge.logger, 9

    nodedge.serializable, 10

    nodedge.socket\_type, 11

MyClass (*class in nodedge.socket\_type*), 11

### N

nodedge

    module, 11

nodedge.blocks.block\_exception

    module, 8

nodedge.logger

    module, 9

nodedge.serializable

    module, 10

nodedge.socket\_type

    module, 11

Number (*nodedge.socket\_type.SocketType* attribute), 11

### R

RedundantInputError, 8

### S

Serializable (*class in nodedge.serializable*), 10

serialize() (*nodedge.serializable.Serializable*  
method), 10

setupLogging() (*in module nodedge.logger*), 9

SocketType (*class in nodedge.socket\_type*), 11

String (*nodedge.socket\_type.SocketType* attribute), 11