

---

# **LOICA**

***Release 1.0.6***

**Gonzalo Vidal**

**Jul 25, 2023**



# CONTENTS

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Installation</b>	<b>3</b>
<b>3</b>	<b>LOICA allows you to:</b>	<b>5</b>
<b>4</b>	<b>Tutorials</b>	<b>7</b>
<b>5</b>	<b>API Reference</b>	<b>9</b>
5.1	loica.....	9
<b>6</b>	<b>Introduction</b>	<b>39</b>
<b>7</b>	<b>Installation</b>	<b>41</b>
<b>8</b>	<b>LOICA allows you to:</b>	<b>43</b>
<b>9</b>	<b>Tutorials</b>	<b>45</b>
	<b>Python Module Index</b>	<b>47</b>
	<b>Index</b>	<b>49</b>



## INTRODUCTION

Welcome to the LOICA (Logical Operators for Integrated Cell Algorithms) repository, our Python package for designing, modeling and characterizing genetic networks.

As you may have noticed, our logo features a beautiful bird—loica ([Leistes loyca](#)); a bird native to Chile known for its particular red chest and legendary kindness, with which we share name.



## INSTALLATION

Installing LOICA is way easier than pronouncing it!

```
pip install loica
```

For more details please refer to our [Wiki](#) for installation instructions and developer guides.





## **LOICA ALLOWS YOU TO:**

- Compile Code into DNA fragments that execute Cell Algorithms
- Easy programation of genetic network models
- Generation of synthetic data
- Communicate with Flapjack
- Use and output SBOL files
- Use all sorts of cellular computation
- Easy, fluid and customisable DNA design



## TUTORIALS

Now that you have LOICA installed you can familiarize yourself with the tool using the [Jupyter notebook](#) tutorials designed for this purpose.



## API REFERENCE

This page contains auto-generated API reference documentation<sup>1</sup>.

### 5.1 loica

#### 5.1.1 Subpackages

`loica.operators`

##### Submodules

`loica.operators.hill1`

##### Module Contents

##### Classes

---

<i>Hill1</i>	A class that represents a DNA fragment that encode a genetic operator.
--------------	--

---

**class** `Hill1`(*input*, *output*, *alpha*, *K*, *n*, *name=None*, *uri=None*, *sbol\_comp=None*, *color='skyblue'*)

Bases: `loica.operators.operator.Operator`

A class that represents a DNA fragment that encode a genetic operator. The Hill1 Operator is an abstraction of a repressible or inducible promoter that maps an input into an output using a Hill function.

...

**input** [Regulator | Supplement] The input of the operator that regulates the expression of the output

**output** [Regulator | Reporter] The output of the operator that is regulated by the input

**alpha** [List] [Basal expression rate, Regulated expression rate in MEFL/second]

**K** [int | float] Half expression input concentration in Molar

**n** [int | float] Hill coefficient, cooperative degree (unitless)

**uri** [str, optional] SynBioHub URI

---

<sup>1</sup> Created with `sphinx-autoapi`

**sbol\_comp** [SBOL Component, optional] SBOL Component

**name** [str, optional] Name of the operator displayed on the network representation

**color: str, optional** Color displayed on the network representation

**characterize(flappjack, receiver, inverter, media, strain, signal, biomass\_signal, gamma)** Parameterize the Operator model that maps Input concentration into Output expression rate

**\_\_str\_\_(self)**  
Return str(self).

**characterize(self, flappjack, receiver, inverter, media, strain, signal, biomass\_signal, gamma)**

**expression\_rate(self, t, dt)**

**forward\_model(self, a\_j, b\_j, n\_i=2, K\_i=1, a\_A=100.0, b\_A=0, K\_A=1, n\_A=2, Dt=0.05, sim\_steps=10, A=0, odval=[1] \* 100, gamma=0, p0\_1=0, p0\_2=0, nt=100)**

**residuals(self, df, oddf, a\_A, b\_A, K\_A, n\_A, gamma)**

## loica.operators.hill2

### Module Contents

#### Classes

---

<i>Hill2</i>	A class that represents a DNA fragment that encode a genetic operator.
--------------	--

---

**class Hill2**(input, output, alpha, K, n, name=None, uri=None, sbol\_comp=None, color='orange')

Bases: loica.operators.receiver.Operator

A class that represents a DNA fragment that encode a genetic operator. The Hill2 Operator is an abstraction of a set of two repressible or inducible promoters that maps an 2 inputs into an output using a Hill function.

...

**input** [List [Regulator | Supplement]] The inputs of the operator that regulates the expression of the output

**output** [Regulator | Reporter | List] The output of the operator that is regulated by the input

**alpha** [List] [Basal expression rate, Regulated expression rate in MEFL/second]

**K** [int | float] Half expression input concentration in Molar

**n** [int | float] Hill coefficient, cooperative degree (unitless)

**uri** [str, optional] SynBioHub URI

**sbol\_comp** [SBOL Component, optional] SBOL Component

**name** [str, optional] Name of the operator displayed on the network representation

**color: str, optional** Color displayed on the network representation

**characterize(flappjack, receiver, inverter, media, strain, signal, biomass\_signal, gamma)** Parameterize the Operator model that maps Input concentration into Output expression rate

```

__str__(self)
    Return str(self).

characterize(self, flapjack, receiver1, receiver2, chemical1, chemical2, nor_inverter, media, strain, signal,
             biomass_signal, gamma, lower_bounds=[0] * 8, upper_bounds=[100000000.0, 8,
             100000000.0, 8, 100000000.0, 100000000.0, 100000000.0, 100000000.0], init_x=[1, 2, 1, 2,
             1, 0, 0, 0])

expression_rate(self, t, dt)

forward_model(self, rep1_K=1, rep1_n=2, rep2_K=1, rep2_n=2, alpha0=1, alpha1=0, alpha2=0,
             alpha3=0, a_A=100.0, b_A=0, K_A=1, n_A=2, a_B=100.0, b_B=0, K_B=1, n_B=2,
             Dt=0.05, sim_steps=10, A=0, B=0, odval=[1] * 100, gamma=0, rep1_0=0, rep2_0=0,
             fp_0=0, nt=100)

residuals(self, df, oddf, a_A, b_A, K_A, n_A, a_B, b_B, K_B, n_B, chem1, chem2, gamma)

```

**loica.operators.operator**

## Module Contents

### Classes

<i>Operator</i>	A class that represents a DNA fragment that encode a genetic operator.
-----------------	--

**class Operator**(output, name=None, uri=None, sbol\_comp=None, color='skyblue')

A class that represents a DNA fragment that encode a genetic operator.

...

**output** [Regulator | Reporter] The output of the operator that is regulated by the input

**uri** [str, optional] SynBioHub URI

**sbol\_comp** [SBOL Component, optional] SBOL Component

**name** [str, optional] Name of the operator displayed on the network representation

**color: str, optional** Color displayed on the network representation

```

__str__(self)
    Return str(self).

```

**loica.operators.receiver**

## Module Contents

### Classes

<i>Receiver</i>	A class that represents a DNA fragment that encode a genetic operator.
-----------------	--

**class Receiver**(*input, output, alpha, K, n, name=None, uri=None, sbol\_comp=None, color='skyblue'*)

Bases: [loica.operators.operator.Operator](#)

A class that represents a DNA fragment that encode a genetic operator. The Receiver Operator is an abstraction of an inducible promoter that maps an external input into an output using a Hill function.

...

**input** [Regulator | Supplement] The input of the operator that regulates the expression of the output

**output** [Regulator | Reporter] The output of the operator that is regulated by the input

**alpha** [List] [Basal expression rate, Regulated expression rate in MEFL/second]

**K** [int | float] Half expression input concentration in Molar

**n** [int | float] Hill coefficient, cooperative degree (unitless)

**uri** [str, optional] SynBioHub URI

**sbol\_comp** [SBOL Component, optional] SBOL Component

**name** [str, optional] Name of the operator displayed on the network representation

**color: str, optional** Color displayed on the network representation

**unit: str, optional** Units of the characterization data

**characterize(flappjack, receiver, inverter, media, strain, signal, biomass\_signal, gamma)** Parameterize the Operator model that maps Input concentration into Output expression rate

**\_\_str\_\_(self)**  
Return str(self).

**characterize(self, flappjack, vector, media, strain, signal, biomass\_signal)**

**expression\_rate(self, t, dt)**

**forward\_model(self, a=0, b=1, K\_A=1, n\_A=2, Dt=0.05, sim\_steps=10, A=[0], odval=[1] \* 100, gamma=0, p0=0, nt=100)**

**residuals(self, df, oddf)**

[loica.operators.source](#)

## Module Contents

### Classes

---

[Source](#)

A class that represents a DNA fragment that encode a genetic operator.

---

**class Source**(*output, rate, uri=None, sbol\_comp=None, color='blue', name=None*)

Bases: [Operator](#)

A class that represents a DNA fragment that encode a genetic operator. The Source Operator is an abstraction of a constitutive promoter that produces output.

...

**output** [Regulator | Reporter] The output of the operator that is constitutively expressed



**rate** [float] Output constitutive expression rate in MEFL/second

**uri** [str, optional] SynBioHub URI

**sbol\_comp** [SBOL Component, optional] SBOL Component

**name** [str, optional] Name of the operator displayed on the network representation

**color: str, optional** Color displayed on the network representation

**characterize(flappjack, receiver, inverter, media, strain, signal, biomass\_signal, gamma)** Parameterize the Operator model that maps Input concentration into Output expression rate

**\_\_str\_\_(self)**

**characterize(self, flappjack, vector, media, strain, signal, biomass\_signal)**

**expression\_rate(self, t, dt)**

**forward\_model(self, Dt=0.25, sim\_steps=10, odval=[1] \* 97, rate=1, gamma=0, p0=0, nt=100)**

**residuals(self, df, oddf)**

`loica.operators.sum`

## Module Contents

### Classes

---

*Sum*

---

```
class Sum(input, output, alpha, K, n, uri=None, sbol_comp=None)
    Bases: Operator
    color = skyblue
    shape = s
    __str__(self)
    expression_rate(self, t, dt)
```

## Package Contents

### Classes

<i>Hill1</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Hill2</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Operator</i>	A class that represents a DNA fragment that encode a genetic operator.

continues on next page

Table 7 – continued from previous page

<i>Operator</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Operator</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Operator</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Operator</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Operator</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Receiver</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Receiver</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Receiver</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Source</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Source</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Source</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Source</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Sum</i>	

**class Hill1**(*input, output, alpha, K, n, name=None, uri=None, sbol\_comp=None, color='skyblue'*)

Bases: *loica.operators.operator.Operator*

A class that represents a DNA fragment that encode a genetic operator. The Hill1 Operator is an abstraction of a repressible or inducible promoter that maps an input into an output using a Hill function.

...

**input** [Regulator | Supplement] The input of the operator that regulates the expression of the output

**output** [Regulator | Reporter] The output of the operator that is regulated by the input

**alpha** [List] [Basal expression rate, Regulated expression rate in MEFL/second]

**K** [int | float] Half expression input concentration in Molar

**n** [int | float] Hill coefficient, cooperative degree (unitless)

**uri** [str, optional] SynBioHub URI

**sbol\_comp** [SBOL Component, optional] SBOL Component

**name** [str, optional] Name of the operator displayed on the network representation

**color: str, optional** Color displayed on the network representation

**characterize(flappjack, receiver, inverter, media, strain, signal, biomass\_signal, gamma)** Parameterize the Operator model that maps Input concentration into Output expression rate

**\_\_str\_\_(self)**  
Return str(self).

```

characterize(self, flapjack, receiver, inverter, media, strain, signal, biomass_signal, gamma)
expression_rate(self, t, dt)
forward_model(self, a_j, b_j, n_i=2, K_i=1, a_A=100.0, b_A=0, K_A=1, n_A=2, Dt=0.05, sim_steps=10,
                A=0, odval=[1] * 100, gamma=0, p0_1=0, p0_2=0, nt=100)
residuals(self, df, oddf, a_A, b_A, K_A, n_A, gamma)

```

**class Hill2**(input, output, alpha, K, n, name=None, uri=None, sbol\_comp=None, color='orange')

Bases: `loica.operators.receiver.Operator`

A class that represents a DNA fragment that encode a genetic operator. The Hill2 Operator is an abstraction of a set of two repressible or inducible promoters that maps an 2 inputs into an output using a Hill function.

...

**input** [List [Regulator | Supplement]] The inputs of the operator that regulates the expression of the output

**output** [Regulator | Reporter | List] The output of the operator that is regulated by the input

**alpha** [List] [Basal expression rate, Regulated expression rate in MEFL/second]

**K** [int | float] Half expression input concentration in Molar

**n** [int | float] Hill coefficient, cooperative degree (unitless)

**uri** [str, optional] SynBioHub URI

**sbol\_comp** [SBOL Component, optional] SBOL Component

**name** [str, optional] Name of the operator displayed on the network representation

**color: str, optional** Color displayed on the network representation

**characterize**(flapjack, receiver, inverter, media, strain, signal, biomass\_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

**\_\_str\_\_**(self)  
Return str(self).

```

characterize(self, flapjack, receiver1, receiver2, chemical1, chemical2, nor_inverter, media, strain, signal,
                biomass_signal, gamma, lower_bounds=[0] * 8, upper_bounds=[100000000.0, 8,
                100000000.0, 8, 100000000.0, 100000000.0, 100000000.0, 100000000.0], init_x=[1, 2, 1, 2,
                1, 0, 0, 0])
expression_rate(self, t, dt)
forward_model(self, rep1_K=1, rep1_n=2, rep2_K=1, rep2_n=2, alpha0=1, alpha1=0, alpha2=0,
                alpha3=0, a_A=100.0, b_A=0, K_A=1, n_A=2, a_B=100.0, b_B=0, K_B=1, n_B=2,
                Dt=0.05, sim_steps=10, A=0, B=0, odval=[1] * 100, gamma=0, rep1_0=0, rep2_0=0,
                fp_0=0, nt=100)
residuals(self, df, oddf, a_A, b_A, K_A, n_A, a_B, b_B, K_B, n_B, chem1, chem2, gamma)

```

**class Operator**(output, name=None, uri=None, sbol\_comp=None, color='skyblue')

A class that represents a DNA fragment that encode a genetic operator.

...

**output** [Regulator | Reporter] The output of the operator that is regulated by the input

**uri** [str, optional] SynBioHub URI

**sbol\_comp** [SBOL Component, optional] SBOL Component

**name** [str, optional] Name of the operator displayed on the network representation

**color: str, optional** Color displayed on the network representation

**\_\_str\_\_(self)**  
Return str(self).

**class Operator**(output, name=None, uri=None, sbol\_comp=None, color='skyblue')

A class that represents a DNA fragment that encode a genetic operator.

...

**output** [Regulator | Reporter] The output of the operator that is regulated by the input

**uri** [str, optional] SynBioHub URI

**sbol\_comp** [SBOL Component, optional] SBOL Component

**name** [str, optional] Name of the operator displayed on the network representation

**color: str, optional** Color displayed on the network representation

**\_\_str\_\_(self)**  
Return str(self).

**class Operator**(output, name=None, uri=None, sbol\_comp=None, color='skyblue')

A class that represents a DNA fragment that encode a genetic operator.

...

**output** [Regulator | Reporter] The output of the operator that is regulated by the input

**uri** [str, optional] SynBioHub URI

**sbol\_comp** [SBOL Component, optional] SBOL Component

**name** [str, optional] Name of the operator displayed on the network representation

**color: str, optional** Color displayed on the network representation

**\_\_str\_\_(self)**  
Return str(self).

**class Operator**(output, name=None, uri=None, sbol\_comp=None, color='skyblue')

A class that represents a DNA fragment that encode a genetic operator.

...

**output** [Regulator | Reporter] The output of the operator that is regulated by the input

**uri** [str, optional] SynBioHub URI

**sbol\_comp** [SBOL Component, optional] SBOL Component

**name** [str, optional] Name of the operator displayed on the network representation

**color: str, optional** Color displayed on the network representation

**\_\_str\_\_(self)**  
Return str(self).

**class Operator**(output, name=None, uri=None, sbol\_comp=None, color='skyblue')

A class that represents a DNA fragment that encode a genetic operator.

...

**output** [Regulator | Reporter] The output of the operator that is regulated by the input

**uri** [str, optional] SynBioHub URI

**sbol\_comp** [SBOL Component, optional] SBOL Component

**name** [str, optional] Name of the operator displayed on the network representation

**color: str, optional** Color displayed on the network representation

**\_\_str\_\_(self)**  
Return str(self).

**class Operator**(*output, name=None, uri=None, sbol\_comp=None, color='skyblue'*)  
A class that represents a DNA fragment that encode a genetic operator.

...

**output** [Regulator | Reporter] The output of the operator that is regulated by the input

**uri** [str, optional] SynBioHub URI

**sbol\_comp** [SBOL Component, optional] SBOL Component

**name** [str, optional] Name of the operator displayed on the network representation

**color: str, optional** Color displayed on the network representation

**\_\_str\_\_(self)**  
Return str(self).

**class Receiver**(*input, output, alpha, K, n, name=None, uri=None, sbol\_comp=None, color='skyblue'*)  
Bases: [loica.operators.operator.Operator](#)

A class that represents a DNA fragment that encode a genetic operator. The Receiver Operator is an abstraction of an inducible promoter that maps an external input into an output using a Hill function.

...

**input** [Regulator | Supplement] The input of the operator that regulates the expression of the output

**output** [Regulator | Reporter] The output of the operator that is regulated by the input

**alpha** [List] [Basal expression rate, Regulated expression rate in MEFL/second]

**K** [int | float] Half expression input concentration in Molar

**n** [int | float] Hill coefficient, cooperative degree (unitless)

**uri** [str, optional] SynBioHub URI

**sbol\_comp** [SBOL Component, optional] SBOL Component

**name** [str, optional] Name of the operator displayed on the network representation

**color: str, optional** Color displayed on the network representation

**unit: str, optional** Units of the characterization data

**characterize(flapjack, receiver, inverter, media, strain, signal, biomass\_signal, gamma)** Parameterize the Operator model that maps Input concentration into Output expression rate

**\_\_str\_\_(self)**  
Return str(self).

**characterize(self, flapjack, vector, media, strain, signal, biomass\_signal)**

**expression\_rate(self, t, dt)**

**forward\_model(self, a=0, b=1, K\_A=1, n\_A=2, Dt=0.05, sim\_steps=10, A=[0], odval=[1] \* 100, gamma=0, p0=0, nt=100)**

**residuals**(*self*, *df*, *oddf*)

**class Receiver**(*input*, *output*, *alpha*, *K*, *n*, *name=None*, *uri=None*, *sbol\_comp=None*, *color='skyblue'*)  
Bases: [loica.operators.operator.Operator](#)

A class that represents a DNA fragment that encode a genetic operator. The Receiver Operator is an abstraction of an inducible promoter that maps an external input into an output using a Hill function.

...

**input** [Regulator | Supplement] The input of the operator that regulates the expression of the output

**output** [Regulator | Reporter] The output of the operator that is regulated by the input

**alpha** [List] [Basal expression rate, Regulated expression rate in MEFL/second]

**K** [int | float] Half expression input concentration in Molar

**n** [int | float] Hill coefficient, cooperative degree (unitless)

**uri** [str, optional] SynBioHub URI

**sbol\_comp** [SBOL Component, optional] SBOL Component

**name** [str, optional] Name of the operator displayed on the network representation

**color: str, optional** Color displayed on the network representation

**unit: str, optional** Units of the characterization data

**characterize**(*flapjack*, *receiver*, *inverter*, *media*, *strain*, *signal*, *biomass\_signal*, *gamma*) Parameterize the Operator model that maps Input concentration into Output expression rate

**\_\_str\_\_**(*self*)  
Return str(*self*).

**characterize**(*self*, *flapjack*, *vector*, *media*, *strain*, *signal*, *biomass\_signal*)

**expression\_rate**(*self*, *t*, *dt*)

**forward\_model**(*self*, *a=0*, *b=1*, *K\_A=1*, *n\_A=2*, *Dt=0.05*, *sim\_steps=10*, *A=[0]*, *odval=[1] \* 100*, *gamma=0*, *p0=0*, *nt=100*)

**residuals**(*self*, *df*, *oddf*)

**class Receiver**(*input*, *output*, *alpha*, *K*, *n*, *name=None*, *uri=None*, *sbol\_comp=None*, *color='skyblue'*)  
Bases: [loica.operators.operator.Operator](#)

A class that represents a DNA fragment that encode a genetic operator. The Receiver Operator is an abstraction of an inducible promoter that maps an external input into an output using a Hill function.

...

**input** [Regulator | Supplement] The input of the operator that regulates the expression of the output

**output** [Regulator | Reporter] The output of the operator that is regulated by the input

**alpha** [List] [Basal expression rate, Regulated expression rate in MEFL/second]

**K** [int | float] Half expression input concentration in Molar

**n** [int | float] Hill coefficient, cooperative degree (unitless)

**uri** [str, optional] SynBioHub URI

**sbol\_comp** [SBOL Component, optional] SBOL Component

**name** [str, optional] Name of the operator displayed on the network representation

**color: str, optional** Color displayed on the network representation

**unit: str, optional** Units of the characterization data

**characterize(flappjack, receiver, inverter, media, strain, signal, biomass\_signal, gamma)** Parameterize the Operator model that maps Input concentration into Output expression rate

**\_\_str\_\_(self)**  
Return str(self).

**characterize(self, flappjack, vector, media, strain, signal, biomass\_signal)**

**expression\_rate(self, t, dt)**

**forward\_model(self, a=0, b=1, K\_A=1, n\_A=2, Dt=0.05, sim\_steps=10, A=[0], odval=[1] \* 100, gamma=0, p0=0, nt=100)**

**residuals(self, df, oddf)**

**class Source**(output, rate, uri=None, sbol\_comp=None, color='blue', name=None)  
Bases: [Operator](#)

A class that represents a DNA fragment that encode a genetic operator. The Source Operator is an abstraction of a constitutive promoter that produces output.

...

**output** [Regulator | Reporter] The output of the operator that is constitutively expressed

**rate** [float] Output constitutive expression rate in MEFL/second

**uri** [str, optional] SynBioHub URI

**sbol\_comp** [SBOL Component, optional] SBOL Component

**name** [str, optional] Name of the operator displayed on the network representation

**color: str, optional** Color displayed on the network representation

**characterize(flappjack, receiver, inverter, media, strain, signal, biomass\_signal, gamma)** Parameterize the Operator model that maps Input concentration into Output expression rate

**\_\_str\_\_(self)**

**characterize(self, flappjack, vector, media, strain, signal, biomass\_signal)**

**expression\_rate(self, t, dt)**

**forward\_model(self, Dt=0.25, sim\_steps=10, odval=[1] \* 97, rate=1, gamma=0, p0=0, nt=100)**

**residuals(self, df, oddf)**

**class Source**(output, rate, uri=None, sbol\_comp=None, color='blue', name=None)  
Bases: [Operator](#)

A class that represents a DNA fragment that encode a genetic operator. The Source Operator is an abstraction of a constitutive promoter that produces output.

...

**output** [Regulator | Reporter] The output of the operator that is constitutively expressed

**rate** [float] Output constitutive expression rate in MEFL/second

**uri** [str, optional] SynBioHub URI

**sbol\_comp** [SBOL Component, optional] SBOL Component

**name** [str, optional] Name of the operator displayed on the network representation

**color: str, optional** Color displayed on the network representation

**characterize(flapjack, receiver, inverter, media, strain, signal, biomass\_signal, gamma)** Parameterize the Operator model that maps Input concentration into Output expression rate

**\_\_str\_\_(self)**

**characterize(self, flapjack, vector, media, strain, signal, biomass\_signal)**

**expression\_rate(self, t, dt)**

**forward\_model(self, Dt=0.25, sim\_steps=10, odval=[1] \* 97, rate=1, gamma=0, p0=0, nt=100)**

**residuals(self, df, oddf)**

**class Source**(output, rate, uri=None, sbol\_comp=None, color='blue', name=None)

Bases: [Operator](#)

A class that represents a DNA fragment that encode a genetic operator. The Source Operator is an abstraction of a constitutive promoter that produces output.

...

**output** [Regulator | Reporter] The output of the operator that is constitutively expressed

**rate** [float] Output constitutive expression rate in MEFL/second

**uri** [str, optional] SynBioHub URI

**sbol\_comp** [SBOL Component, optional] SBOL Component

**name** [str, optional] Name of the operator displayed on the network representation

**color: str, optional** Color displayed on the network representation

**characterize(flapjack, receiver, inverter, media, strain, signal, biomass\_signal, gamma)** Parameterize the Operator model that maps Input concentration into Output expression rate

**\_\_str\_\_(self)**

**characterize(self, flapjack, vector, media, strain, signal, biomass\_signal)**

**expression\_rate(self, t, dt)**

**forward\_model(self, Dt=0.25, sim\_steps=10, odval=[1] \* 97, rate=1, gamma=0, p0=0, nt=100)**

**residuals(self, df, oddf)**

**class Source**(output, rate, uri=None, sbol\_comp=None, color='blue', name=None)

Bases: [Operator](#)

A class that represents a DNA fragment that encode a genetic operator. The Source Operator is an abstraction of a constitutive promoter that produces output.

...

**output** [Regulator | Reporter] The output of the operator that is constitutively expressed

**rate** [float] Output constitutive expression rate in MEFL/second

**uri** [str, optional] SynBioHub URI

**sbol\_comp** [SBOL Component, optional] SBOL Component



**name** [str, optional] Name of the operator displayed on the network representation

**color: str, optional** Color displayed on the network representation

**characterize(flappjack, receiver, inverter, media, strain, signal, biomass\_signal, gamma)** Parameterize the Operator model that maps Input concentration into Output expression rate

**\_\_str\_\_**(self)

**characterize**(self, flappjack, vector, media, strain, signal, biomass\_signal)

**expression\_rate**(self, t, dt)

**forward\_model**(self, Dt=0.25, sim\_steps=10, odval=[1] \* 97, rate=1, gamma=0, p0=0, nt=100)

**residuals**(self, df, oddf)

**class Sum**(input, output, alpha, K, n, uri=None, sbol\_comp=None)

Bases: [Operator](#)

**color** = skyblue

**shape** = s

**\_\_str\_\_**(self)

**expression\_rate**(self, t, dt)

## 5.1.2 Submodules

[loica.assay](#)

### Module Contents

#### Classes

<a href="#">Assay</a>	Assay measures a set of samples in parallel at a set of timepoints.
-----------------------	---

**class Assay**(samples, n\_measurements, interval, name='Loica assay', description="", biomass\_signal\_id=None)

Assay measures a set of samples in parallel at a set of timepoints. Connects to flappjack to generate data, and to fit parameters to data.

...

**samples** [List[Sample]] List of Samples that belongs to the Assay

**n\_measurements** [int] Number of measurements to take

**interval** [int] Time in hours between each measurements

**name** [str] Name of the Assay

**description: str** Descriptioin of the Assay

**biomass\_signal\_id** [int] Flappjack ID of the Assay that is associated with the Assay

**run(substeps=10, nsr=0, biomass\_bg=0, fluo\_bg=0)** Runs the Assay time series

**upload(flappjack, study)** Upload the data produced by running the Assay to Flappjack into the Study

Assay measures a set of samples in parallel at a set of timepoints Connects to flapjack to generate data, and to fit parameters to data

```
run(self, substeps=10, nsr=0, biomass_bg=0, fluo_bg=0, stochastic=False)
```

Run the assay measuring at specified time points, with simulation time step dt

```
upload(self, flapjack, study)
```

**loica.colony**

## Module Contents

### Classes

---

*Colony*

---

```
class Colony(circuit=None, r0=1, mu0=1)
```

```
    fun(self, x)
```

```
    kymograph(self, nx, t0, tmax)
```

```
    map_kymo(self, kymo)
```

```
    norm_kymo(self, kymo)
```

**loica.geneproduct**

## Module Contents

### Classes

<i>GeneProduct</i>	A class that represents a gene product, protein or RNA.
<i>Regulator</i>	Representation of a regulatory gene product.
<i>Reporter</i>	Representation of a regulatory gene product.

```
class GeneProduct(name, init_concentration=0, degradation_rate=0, uri=None, sbol_comp=None,  
                  type_='PRO', color='silver')
```

A class that represents a gene product, protein or RNA.

...

**name** [str] Name of the gene product

**init\_concentration** [int | float] Initial concentration of the gene product in Molar

**degradation\_rate** [int | float] Degradation rate of the gene product

**type\_** [str, optional] Molecular type of the gene product, could be 'PRO' or 'RNA'

**uri** [str, optional] SynBioHub URI

**sbol\_comp** [SBOL Component, optional] SBOL Component

**shape** = ^

```

    __str__(self)
        Return str(self).

    express(self, rate)

    initialize(self)

    step(self, growth_rate, dt)

class Regulator(name, init_concentration=0, degradation_rate=0, sbol_comp=None, color='lightgreen')
    Bases: GeneProduct

    Representation of a regulatory gene product. Child of GeneProduct.

class Reporter(name, init_concentration=0, degradation_rate=0, signal_id=None, color='w',
               sbol_comp=None)
    Bases: GeneProduct

    Representation of a regulatory gene product.

    signal_id [str, optional] Flapjack ID of the signal that the reporter is associated with.

    color [str, optional] Color of the reporter

```

loica.genetic\_network

## Module Contents

### Classes

<i>GeneticNetwork</i>	Representation of a genetic netowrk composed by a set of Operators, Regulators and Reporters.
-----------------------	---

```

class GeneticNetwork(vector=None)
    Representation of a genetic netowrk composed by a set of Operators, Regulators and Reporters.

    ...

    operators [List[Operator]] List of Operators that are part of the genetic network
    regulators [List[Regulator]] List of Regulators that are part of the genetic network
    reporters [List[Reporter]] List of Reporters that are part of the genetic network
    vector [int] Flapjack ID of the vector that is associated with the genetic network

    to_graph() Builds a graph representation of the genetic network
    draw() Generates a plot of the graph representation builded by to_graph()
    to_sbol(sbol_doc=None) Generates a SBOL3 Document representation of the genetic network on sbol_doc

    add_operator(self, ops)
    add_regulator(self, regs)
    add_reporter(self, reps)

    draw(self, node_shape='o', node_size=500, linewidths=0, alpha=0.5, arrowsize=10, font_size=6,
          font_family='Tahoma', font_weight='bold', pos=nx.kamada_kawai_layout, contracted=False)

    initialize(self)

```

```
step(self, growth_rate=1, t=0, dt=0.1)
step_stochastic(self, growth_rate=1, t=0, dt=0.1)
substep_stochastic(self, t=0, dt=0.1, growth_rate=1)
to_contracted_graph(self)
to_graph(self)
to_sbml(self, sbml_doc: sbml3.Document = None) → sbml3.Document
    Convert the genetic network to SBML. :param sbml_doc: The SBML document to add the genetic network
    to.
```

`loica.metabolism`

## Module Contents

### Classes

<i>DataMetabolism</i>	Characterized context for gene expression, incorporates biomass and growth rate.
<i>Metabolism</i>	Context for gene expression, incorporates biomass and growth rate.
<i>SimulatedMetabolism</i>	Simulated context for gene expression, incorporates biomass and growth rate.

### Functions

<i>gomPERTZ</i> (t, y0, ymax, um, l)
<i>gomPERTZ_growth_rate</i> (t, y0, ymax, um, l)
<i>ramp_biomass</i> (t, od0, start, slope)
<i>ramp_growth_rate</i> (t, start, slope)
<i>step_biomass</i> (t, od0, start)
<i>step_growth_rate</i> (t, start)

```
class DataMetabolism(name, fj, media, strain, vector, biomass_signal)
    Bases: Metabolism
    Characterized context for gene expression, incorporates biomass and growth rate. ...
    name [str, optional] Name of the metabolism or corresponding strain
    fj [Flapjack] Flapjack instance used to fetch data from
    media [str] Name of the media to query
    strain [str] Name of the strain to query
```

**vector** [str] Name of the vector to query

**biomass\_signal** [str] Name of signal to query and use as biomass

**biomass(t)** Return biomass at a given time from characterization data

**growth\_rate(t)** Return growth rate at a given time from characterization data

**biomass**(*self*, *t*)

**growth\_rate**(*self*, *t*)

**class Metabolism**(*name=None*)

Context for gene expression, incorporates biomass and growth rate. ...

**name** [str, optional] Name of the metabolism or corresponding strain

**class SimulatedMetabolism**(*name, biomass, growth\_rate*)

Bases: [Metabolism](#)

Simulated context for gene expression, incorporates biomass and growth rate. ...

**name** [str, optional] Name of the metabolism or corresponding strain

**biomass** A function of time that describes biomass  $f(t)=\text{biomass}$

**growth\_rate** A function of time that describes the growth rate  $f(t)=\text{growth rate}$

**gompertz**(*t, y0, ymax, um, l*)

**gompertz\_growth\_rate**(*t, y0, ymax, um, l*)

**ramp\_biomass**(*t, od0, start, slope*)

**ramp\_growth\_rate**(*t, start, slope*)

**step\_biomass**(*t, od0, start*)

**step\_growth\_rate**(*t, start*)

[loica.sample](#)

## Module Contents

### Classes

<a href="#">Sample</a>	Representation of a sample that encapsulates Genetic-Network and Metabolism.
------------------------	--

**class Sample**(*genetic\_network=None, metabolism=None, assay=None, media=None, strain=None*)

Representation of a sample that encapsulates GeneticNetwork and Metabolism. Incorporate environment information such as Supplements or chemicals, strain and media. Ex: 1 well in a plate, single cell. ...

**genetic\_network** [GeneticNetwork] genetic network that is part of the sample

**metabolism** [Metabolism] metabolism that drives the genetic network in the sample

**assay** [Assay] assay to which this sample belongs

**media** [str] Name of the media in the sample

**strain** [str]

Name of the strain in the sample

Methods

**add\_supplement(supplement, concentration)** establishes the concentration of Supplement

**initialize(self)**

**set\_regulator(self, name, concentration)**

**set\_reporter(self, name, concentration)**

**set\_supplement(self, supplement, concentration, profile=None)**

**step(self, t, dt, stochastic=False)**

**loica.supplement**

## Module Contents

### Classes

---

<i>Supplement</i>	Representation of a chemical
-------------------	------------------------------

---

**class Supplement**(name, pubchemid=None, supplier\_id=None, sbol\_comp=None, color='pink')

Representation of a chemical

...

**name** [str] Name of the supplement

**concentration** [int | float] concentration of the supplement in Molar

**pubchemid** [str] PubChemID URI of the supplement

**supplier\_id** [str] Supplier ID of the supplement. An URL of the product that you acquire. Accepts list of the form [product URL, catalog number, batch].

**sbol\_comp** [str] SBOL component of the supplement.

**\_\_str\_\_(self)**

Return str(self).

**loica.util**

## Module Contents

### Functions

---

*characterize\_growth*(flapjack, vector, media, strain, biomass\_signal, n\_gaussians, epsilon)

---

*forward\_model\_growth*(Dt=0.05, sim\_steps=10, mu\_val=[0] \* 100, od0=0, nt=100)

---

continues on next page

Table 16 – continued from previous page

<i>load_loica</i> (filename)
<i>residuals_growth</i> (data, epsilon, dt, t, n_gaussians)
<i>save_loica</i> (obj, filename)
<b>characterize_growth</b> (flapjack, vector, media, strain, biomass_signal, n_gaussians, epsilon)
<b>forward_model_growth</b> (Dt=0.05, sim_steps=10, muval=[0] * 100, od0=0, nt=100)
<b>load_loica</b> (filename)
<b>residuals_growth</b> (data, epsilon, dt, t, n_gaussians)
<b>save_loica</b> (obj, filename)

### 5.1.3 Package Contents

#### Classes

<i>Assay</i>	Assay measures a set of samples in parallel at a set of timepoints.
<i>Colony</i>	
<i>DataMetabolism</i>	Characterized context for gene expression, incorporates biomass and growth rate.
<i>GeneProduct</i>	A class that represents a gene product, protein or RNA.
<i>GeneticNetwork</i>	Representation of a genetic network composed by a set of Operators, Regulators and Reporters.
<i>Hill1</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Hill1</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Hill2</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Hill2</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Metabolism</i>	Context for gene expression, incorporates biomass and growth rate.
<i>Operator</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Operator</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Receiver</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Receiver</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Regulator</i>	Representation of a regulatory gene product.
<i>Regulator</i>	Representation of a regulatory gene product.
<i>Reporter</i>	Representation of a regulatory gene product.

continues on next page

Table 17 – continued from previous page

<i>Reporter</i>	Representation of a regulatory gene product.
<i>Sample</i>	Representation of a sample that encapsulates Genetic-Network and Metabolism.
<i>SimulatedMetabolism</i>	Simulated context for gene expression, incorporates biomass and growth rate.
<i>Source</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Source</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Sum</i>	
<i>Supplement</i>	Representation of a chemical
<i>Supplement</i>	Representation of a chemical

## Functions

*characterize\_growth*(flapjack, vector, media, strain, biomass\_signal, n\_gaussians, epsilon)

*forward\_model\_growth*(Dt=0.05, sim\_steps=10, mu\_val=[0] \* 100, od0=0, nt=100)

*gompertz*(t, y0, ymax, um, l)

*gompertz\_growth\_rate*(t, y0, ymax, um, l)

*load\_loica*(filename)

*ramp\_biomass*(t, od0, start, slope)

*ramp\_growth\_rate*(t, start, slope)

*residuals\_growth*(data, epsilon, dt, t, n\_gaussians)

*save\_loica*(obj, filename)

*step\_biomass*(t, od0, start)

*step\_growth\_rate*(t, start)

**class** *Assay*(*samples*, *n\_measurements*, *interval*, *name*='Loica assay', *description*='', *biomass\_signal\_id*=None)

Assay measures a set of samples in parallel at a set of timepoints. Connects to flapjack to generate data, and to fit parameters to data.

...

**samples** [List[Sample]] List of Samples that belongs to the Assay

**n\_measurements** [int] Number of measurements to take

**interval** [int] Time in hours between each measurements

**name** [str] Name of the Assay



**description:** `str` Description of the Assay

**biomass\_signal\_id** `[int]` Flapjack ID of the Assay that is associated with the Assay

**run(substeps=10, nsr=0, biomass\_bg=0, fluo\_bg=0)** Runs the Assay time series

**upload(flapjack, study)** Upload the data produced by running the Assay to Flapjack into the Study

Assay measures a set of samples in parallel at a set of timepoints Connects to flapjack to generate data, and to fit parameters to data

**run(self, substeps=10, nsr=0, biomass\_bg=0, fluo\_bg=0, stochastic=False)**

Run the assay measuring at specified time points, with simulation time step dt

**upload(self, flapjack, study)**

**class Colony(circuit=None, r0=1, mu0=1)**

**fun(self, x)**

**kymograph(self, nx, t0, tmax)**

**map\_kymo(self, kymo)**

**norm\_kymo(self, kymo)**

**class DataMetabolism(name, fj, media, strain, vector, biomass\_signal)**

Bases: [Metabolism](#)

Characterized context for gene expression, incorporates biomass and growth rate. ...

**name** `[str, optional]` Name of the metabolism or corresponding strain

**fj** `[Flapjack]` Flapjack instance used to fetch data from

**media** `[str]` Name of the media to query

**strain** `[str]` Name of the strain to query

**vector** `[str]` Name of the vector to query

**biomass\_signal** `[str]` Name of signal to query and use as biomass

**biomass(t)** Return biomass at a given time from characterization data

**growth\_rate(t)** Return growth rate at a given time from characterization data

**biomass(self, t)**

**growth\_rate(self, t)**

**class GeneProduct(name, init\_concentration=0, degradation\_rate=0, uri=None, sbol\_comp=None, type\_='PRO', color='silver')**

A class that represents a gene product, protein or RNA.

...

**name** `[str]` Name of the gene product

**init\_concentration** `[int | float]` Initial concentration of the gene product in Molar

**degradation\_rate** `[int | float]` Degradation rate of the gene product

**type\_** `[str, optional]` Molecular type of the gene product, could be 'PRO' or 'RNA'

**uri** `[str, optional]` SynBioHub URI

**sbol\_comp** `[SBOL Component, optional]` SBOL Component

```
shape = ^
__str__(self)
    Return str(self).
express(self, rate)
initialize(self)
step(self, growth_rate, dt)
class GeneticNetwork(vector=None)
    Representation of a genetic netowrk composed by a set of Operators, Regulators and Reporters.
    ...
    operators [List[Operator]] List of Operators that are part of the genetic network
    regulators [List[Regulator]] List of Regulators that are part of the genetic network
    reporters [List[Reporter]] List of Reporters that are part of the genetic network
    vector [int] Flapjack ID of the vector that is associated with the genetic network

    to_graph() Builds a graph representation of the genetic network
    draw() Generates a plot of the graph representation builded by to_graph()
    to_sbol(sbol_doc=None) Generates a SBOL3 Document representation of the genetic network on sbol_doc

    add_operator(self, ops)
    add_regulator(self, regs)
    add_reporter(self, reps)

    draw(self, node_shape='o', node_size=500, linewidths=0, alpha=0.5, arrowsize=10, font_size=6,
          font_family='Tahoma', font_weight='bold', pos=nx.kamada_kawai_layout, contracted=False)
    initialize(self)
    step(self, growth_rate=1, t=0, dt=0.1)
    step_stochastic(self, growth_rate=1, t=0, dt=0.1)
    substep_stochastic(self, t=0, dt=0.1, growth_rate=1)
    to_contracted_graph(self)
    to_graph(self)
    to_sbol(self, sbol_doc: sbol3.Document = None) → sbol3.Document
        Convert the genetic network to SBOL. :param sbol_doc: The SBOL document to add the genetic network
        to.
class Hill1(input, output, alpha, K, n, name=None, uri=None, sbol_comp=None, color='skyblue')
    Bases: loica.operators.operator.Operator

    A class that represents a DNA fragment that encode a genetic operator. The Hill1 Operator is an abstraction of
    a repressible or inducible promoter that maps an input into an output using a Hill function.
    ...
    input [Regulator | Supplement] The input of the operator that regulates the expression of the output
    output [Regulator | Reporter] The output of the operator that is regulated by the input
    alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]
```

**K** [int | float] Half expression input concentration in Molar

**n** [int | float] Hill coefficient, cooperative degree (unitless)

**uri** [str, optional] SynBioHub URI

**sbol\_comp** [SBOL Component, optional] SBOL Component

**name** [str, optional] Name of the operator displayed on the network representation

**color: str, optional** Color displayed on the network representation

**characterize(flappjack, receiver, inverter, media, strain, signal, biomass\_signal, gamma)** Parameterize the Operator model that maps Input concentration into Output expression rate

**\_\_str\_\_(self)**  
Return str(self).

**characterize(self, flappjack, receiver, inverter, media, strain, signal, biomass\_signal, gamma)**

**expression\_rate(self, t, dt)**

**forward\_model(self, a\_j, b\_j, n\_i=2, K\_i=1, a\_A=100.0, b\_A=0, K\_A=1, n\_A=2, Dt=0.05, sim\_steps=10, A=0, odval=[1] \* 100, gamma=0, p0\_1=0, p0\_2=0, nt=100)**

**residuals(self, df, oddf, a\_A, b\_A, K\_A, n\_A, gamma)**

**class Hill1(input, output, alpha, K, n, name=None, uri=None, sbol\_comp=None, color='skyblue')**  
Bases: [loica.operators.operator.Operator](#)

A class that represents a DNA fragment that encode a genetic operator. The Hill1 Operator is an abstraction of a repressible or inducible promoter that maps an input into an output using a Hill function.

...

**input** [Regulator | Supplement] The input of the operator that regulates the expression of the output

**output** [Regulator | Reporter] The output of the operator that is regulated by the input

**alpha** [List] [Basal expression rate, Regulated expression rate in MEFL/second]

**K** [int | float] Half expression input concentration in Molar

**n** [int | float] Hill coefficient, cooperative degree (unitless)

**uri** [str, optional] SynBioHub URI

**sbol\_comp** [SBOL Component, optional] SBOL Component

**name** [str, optional] Name of the operator displayed on the network representation

**color: str, optional** Color displayed on the network representation

**characterize(flappjack, receiver, inverter, media, strain, signal, biomass\_signal, gamma)** Parameterize the Operator model that maps Input concentration into Output expression rate

**\_\_str\_\_(self)**  
Return str(self).

**characterize(self, flappjack, receiver, inverter, media, strain, signal, biomass\_signal, gamma)**

**expression\_rate(self, t, dt)**

**forward\_model(self, a\_j, b\_j, n\_i=2, K\_i=1, a\_A=100.0, b\_A=0, K\_A=1, n\_A=2, Dt=0.05, sim\_steps=10, A=0, odval=[1] \* 100, gamma=0, p0\_1=0, p0\_2=0, nt=100)**

```
residuals(self, df, oddf, a_A, b_A, K_A, n_A, gamma)
```

```
class Hill2(input, output, alpha, K, n, name=None, uri=None, sbol_comp=None, color='orange')
```

```
    Bases: loica.operators.receiver.Operator
```

A class that represents a DNA fragment that encode a genetic operator. The Hill2 Operator is an abstraction of a set of two repressible or inducible promoters that maps an 2 inputs into an output using a Hill function.

```
...
```

```
input [List [Regulator | Supplement]] The inputs of the operator that regulates the expression of the output
```

```
output [Regulator | Reporter | List] The output of the operator that is regulated by the input
```

```
alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]
```

```
K [int | float] Half expression input concentration in Molar
```

```
n [int | float] Hill coefficient, cooperative degree (unitless)
```

```
uri [str, optional] SynBioHub URI
```

```
sbol_comp [SBOL Component, optional] SBOL Component
```

```
name [str, optional] Name of the operator displayed on the network representation
```

```
color: str, optional Color displayed on the network representation
```

```
characterize(flapjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the  
Operator model that maps Input concentration into Output expression rate
```

```
__str__(self)
```

```
    Return str(self).
```

```
characterize(self, flapjack, receiver1, receiver2, chemical1, chemical2, nor_inverter, media, strain, signal,  
              biomass_signal, gamma, lower_bounds=[0] * 8, upper_bounds=[100000000.0, 8,  
100000000.0, 8, 100000000.0, 100000000.0, 100000000.0, 100000000.0], init_x=[1, 2, 1, 2,  
1, 0, 0, 0])
```

```
expression_rate(self, t, dt)
```

```
forward_model(self, rep1_K=1, rep1_n=2, rep2_K=1, rep2_n=2, alpha0=1, alpha1=0, alpha2=0,  
              alpha3=0, a_A=100.0, b_A=0, K_A=1, n_A=2, a_B=100.0, b_B=0, K_B=1, n_B=2,  
              Dt=0.05, sim_steps=10, A=0, B=0, odval=[1] * 100, gamma=0, rep1_0=0, rep2_0=0,  
              fp_0=0, nt=100)
```

```
residuals(self, df, oddf, a_A, b_A, K_A, n_A, a_B, b_B, K_B, n_B, chem1, chem2, gamma)
```

```
class Hill2(input, output, alpha, K, n, name=None, uri=None, sbol_comp=None, color='orange')
```

```
    Bases: loica.operators.receiver.Operator
```

A class that represents a DNA fragment that encode a genetic operator. The Hill2 Operator is an abstraction of a set of two repressible or inducible promoters that maps an 2 inputs into an output using a Hill function.

```
...
```

```
input [List [Regulator | Supplement]] The inputs of the operator that regulates the expression of the output
```

```
output [Regulator | Reporter | List] The output of the operator that is regulated by the input
```

```
alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]
```

```
K [int | float] Half expression input concentration in Molar
```

```
n [int | float] Hill coefficient, cooperative degree (unitless)
```

**uri** [str, optional] SynBioHub URI

**sbol\_comp** [SBOL Component, optional] SBOL Component

**name** [str, optional] Name of the operator displayed on the network representation

**color: str, optional** Color displayed on the network representation

**characterize(flappjack, receiver, inverter, media, strain, signal, biomass\_signal, gamma)** Parameterize the Operator model that maps Input concentration into Output expression rate

**\_\_str\_\_(self)**  
Return str(self).

**characterize(self, flappjack, receiver1, receiver2, chemical1, chemical2, nor\_inverter, media, strain, signal, biomass\_signal, gamma, lower\_bounds=[0] \* 8, upper\_bounds=[100000000.0, 8, 100000000.0, 8, 100000000.0, 100000000.0, 100000000.0, 100000000.0], init\_x=[1, 2, 1, 2, 1, 0, 0, 0])**

**expression\_rate(self, t, dt)**

**forward\_model(self, rep1\_K=1, rep1\_n=2, rep2\_K=1, rep2\_n=2, alpha0=1, alpha1=0, alpha2=0, alpha3=0, a\_A=100.0, b\_A=0, K\_A=1, n\_A=2, a\_B=100.0, b\_B=0, K\_B=1, n\_B=2, Dt=0.05, sim\_steps=10, A=0, B=0, odval=[1] \* 100, gamma=0, rep1\_0=0, rep2\_0=0, fp\_0=0, nt=100)**

**residuals(self, df, oddf, a\_A, b\_A, K\_A, n\_A, a\_B, b\_B, K\_B, n\_B, chem1, chem2, gamma)**

**class Metabolism(name=None)**  
Context for gene expression, incorporates biomass and growth rate. ...

**name** [str, optional] Name of the metabolism or corresponding strain

**class Operator(output, name=None, uri=None, sbol\_comp=None, color='skyblue')**  
A class that represents a DNA fragment that encode a genetic operator.

...

**output** [Regulator | Reporter] The output of the operator that is regulated by the input

**uri** [str, optional] SynBioHub URI

**sbol\_comp** [SBOL Component, optional] SBOL Component

**name** [str, optional] Name of the operator displayed on the network representation

**color: str, optional** Color displayed on the network representation

**\_\_str\_\_(self)**  
Return str(self).

**class Operator(output, name=None, uri=None, sbol\_comp=None, color='skyblue')**  
A class that represents a DNA fragment that encode a genetic operator.

...

**output** [Regulator | Reporter] The output of the operator that is regulated by the input

**uri** [str, optional] SynBioHub URI

**sbol\_comp** [SBOL Component, optional] SBOL Component

**name** [str, optional] Name of the operator displayed on the network representation

**color: str, optional** Color displayed on the network representation

```
__str__(self)
    Return str(self).
```

```
class Receiver(input, output, alpha, K, n, name=None, uri=None, sbol_comp=None, color='skyblue')
    Bases: loica.operators.operator.Operator
```

A class that represents a DNA fragment that encode a genetic operator. The Receiver Operator is an abstraction of an inducible promoter that maps an external input into an output using a Hill function.

...

**input** [Regulator | Supplement] The input of the operator that regulates the expression of the output

**output** [Regulator | Reporter] The output of the operator that is regulated by the input

**alpha** [List] [Basal expression rate, Regulated expression rate in MEFL/second]

**K** [int | float] Half expression input concentration in Molar

**n** [int | float] Hill coefficient, cooperative degree (unitless)

**uri** [str, optional] SynBioHub URI

**sbol\_comp** [SBOL Component, optional] SBOL Component

**name** [str, optional] Name of the operator displayed on the network representation

**color: str, optional** Color displayed on the network representation

**unit: str, optional** Units of the characterization data

**characterize(flappjack, receiver, inverter, media, strain, signal, biomass\_signal, gamma)** Parameterize the Operator model that maps Input concentration into Output expression rate

```
__str__(self)
    Return str(self).
```

```
characterize(self, flappjack, vector, media, strain, signal, biomass_signal)
```

```
expression_rate(self, t, dt)
```

```
forward_model(self, a=0, b=1, K_A=1, n_A=2, Dt=0.05, sim_steps=10, A=[0], odval=[1] * 100,
               gamma=0, p0=0, nt=100)
```

```
residuals(self, df, oddf)
```

```
class Receiver(input, output, alpha, K, n, name=None, uri=None, sbol_comp=None, color='skyblue')
    Bases: loica.operators.operator.Operator
```

A class that represents a DNA fragment that encode a genetic operator. The Receiver Operator is an abstraction of an inducible promoter that maps an external input into an output using a Hill function.

...

**input** [Regulator | Supplement] The input of the operator that regulates the expression of the output

**output** [Regulator | Reporter] The output of the operator that is regulated by the input

**alpha** [List] [Basal expression rate, Regulated expression rate in MEFL/second]

**K** [int | float] Half expression input concentration in Molar

**n** [int | float] Hill coefficient, cooperative degree (unitless)

**uri** [str, optional] SynBioHub URI

**sbol\_comp** [SBOL Component, optional] SBOL Component

**name** [str, optional] Name of the operator displayed on the network representation

**color: str, optional** Color displayed on the network representation

**unit: str, optional** Units of the characterization data

**characterize(flappjack, receiver, inverter, media, strain, signal, biomass\_signal, gamma)** Parameterize the Operator model that maps Input concentration into Output expression rate

**\_\_str\_\_(self)**  
Return str(self).

**characterize(self, flappjack, vector, media, strain, signal, biomass\_signal)**

**expression\_rate(self, t, dt)**

**forward\_model(self, a=0, b=1, K\_A=1, n\_A=2, Dt=0.05, sim\_steps=10, A=[0], odval=[1] \* 100, gamma=0, p0=0, nt=100)**

**residuals(self, df, oddf)**

**class Regulator(name, init\_concentration=0, degradation\_rate=0, sbol\_comp=None, color='lightgreen')**  
Bases: [GeneProduct](#)

Representation of a regulatory gene product. Child of GeneProduct.

**class Regulator(name, init\_concentration=0, degradation\_rate=0, sbol\_comp=None, color='lightgreen')**  
Bases: [GeneProduct](#)

Representation of a regulatory gene product. Child of GeneProduct.

**class Reporter(name, init\_concentration=0, degradation\_rate=0, signal\_id=None, color='w', sbol\_comp=None)**  
Bases: [GeneProduct](#)

Representation of a regulatory gene product.

**signal\_id** [str, optional] Flappjack ID of the signal that the reporter is associated with.

**color** [str, optional] Color of the reporter

**class Reporter(name, init\_concentration=0, degradation\_rate=0, signal\_id=None, color='w', sbol\_comp=None)**  
Bases: [GeneProduct](#)

Representation of a regulatory gene product.

**signal\_id** [str, optional] Flappjack ID of the signal that the reporter is associated with.

**color** [str, optional] Color of the reporter

**class Sample(genetic\_network=None, metabolism=None, assay=None, media=None, strain=None)**  
Representation of a sample that encapsulates GeneticNetwork and Metabolism. Incorporate environment information such as Supplements or chemicals, strain and media. Ex: 1 well in a plate, single cell. ...

**genetic\_network** [GeneticNetwork] genetic network that is part of the sample

**metabolism** [Metabolism] metabolism that drives the genetic network in the sample

**assay** [Assay] assay to which this sample belongs

**media** [str] Name of the media in the sample

**strain** [str]  
Name of the strain in the sample

## Methods

**add\_supplement(supplement, concentration)** establishes the concentration of Supplement

**initialize(self)**

**set\_regulator(self, name, concentration)**

**set\_reporter(self, name, concentration)**

**set\_supplement(self, supplement, concentration, profile=None)**

**step(self, t, dt, stochastic=False)**

**class SimulatedMetabolism(name, biomass, growth\_rate)**

Bases: [Metabolism](#)

Simulated context for gene expression, incorporates biomass and growth rate. ...

**name** [str, optional] Name of the metabolism or corresponding strain

**biomass** A function of time that describes biomass  $f(t)=\text{biomass}$

**growth\_rate** A function of time that describes the growth rate  $f(t)=\text{growth rate}$

**class Source(output, rate, uri=None, sbol\_comp=None, color='blue', name=None)**

Bases: [Operator](#)

A class that represents a DNA fragment that encode a genetic operator. The Source Operator is an abstraction of a constitutive promoter that produces output.

...

**output** [Regulator | Reporter] The output of the operator that is constitutively expressed

**rate** [float] Output constitutive expression rate in MEFL/second

**uri** [str, optional] SynBioHub URI

**sbol\_comp** [SBOL Component, optional] SBOL Component

**name** [str, optional] Name of the operator displayed on the network representation

**color: str, optional** Color displayed on the network representation

**characterize(flapjack, receiver, inverter, media, strain, signal, biomass\_signal, gamma)** Parameterize the Operator model that maps Input concentration into Output expression rate

**\_\_str\_\_(self)**

**characterize(self, flapjack, vector, media, strain, signal, biomass\_signal)**

**expression\_rate(self, t, dt)**

**forward\_model(self, Dt=0.25, sim\_steps=10, odval=[1] \* 97, rate=1, gamma=0, p0=0, nt=100)**

**residuals(self, df, oddf)**

**class Source(output, rate, uri=None, sbol\_comp=None, color='blue', name=None)**

Bases: [Operator](#)

A class that represents a DNA fragment that encode a genetic operator. The Source Operator is an abstraction of a constitutive promoter that produces output.

...

**output** [Regulator | Reporter] The output of the operator that is constitutively expressed



**rate** [float] Output constitutive expression rate in MEFL/second

**uri** [str, optional] SynBioHub URI

**sbol\_comp** [SBOL Component, optional] SBOL Component

**name** [str, optional] Name of the operator displayed on the network representation

**color: str, optional** Color displayed on the network representation

**characterize(flappjack, receiver, inverter, media, strain, signal, biomass\_signal, gamma)** Parameterize the Operator model that maps Input concentration into Output expression rate

**\_\_str\_\_(self)**

**characterize(self, flappjack, vector, media, strain, signal, biomass\_signal)**

**expression\_rate(self, t, dt)**

**forward\_model(self, Dt=0.25, sim\_steps=10, odval=[1] \* 97, rate=1, gamma=0, p0=0, nt=100)**

**residuals(self, df, oddf)**

**class Sum(input, output, alpha, K, n, uri=None, sbol\_comp=None)**  
 Bases: *Operator*

**color = skyblue**

**shape = s**

**\_\_str\_\_(self)**

**expression\_rate(self, t, dt)**

**class Supplement(name, pubchemid=None, supplier\_id=None, sbol\_comp=None, color='pink')**  
 Representation of a chemical

...

**name** [str] Name of the supplement

**concentration** [int | float] concentration of the supplement in Molar

**pubchemid** [str] PubChemID URI of the supplement

**supplier\_id** [str] Supplier ID of the supplement. An URL of the product that you aquire. Accepts list of the form [product URL, catalog number, batch].

**sbol\_comp** [str] SBOL component of the supplement.

**\_\_str\_\_(self)**  
 Return str(self).

**class Supplement(name, pubchemid=None, supplier\_id=None, sbol\_comp=None, color='pink')**  
 Representation of a chemical

...

**name** [str] Name of the supplement

**concentration** [int | float] concentration of the supplement in Molar

**pubchemid** [str] PubChemID URI of the supplement

**supplier\_id** [str] Supplier ID of the supplement. An URL of the product that you aquire. Accepts list of the form [product URL, catalog number, batch].

**sbol\_comp** [str] SBOL component of the supplement.

**\_\_str\_\_(self)**  
Return str(self).

**characterize\_growth**(*flapjack, vector, media, strain, biomass\_signal, n\_gaussians, epsilon*)

**forward\_model\_growth**(*Dt=0.05, sim\_steps=10, muval=[0] \* 100, od0=0, nt=100*)

**gompertz**(*t, y0, ymax, um, l*)

**gompertz\_growth\_rate**(*t, y0, ymax, um, l*)

**load\_loica**(*filename*)

**ramp\_biomass**(*t, od0, start, slope*)

**ramp\_growth\_rate**(*t, start, slope*)

**residuals\_growth**(*data, epsilon, dt, t, n\_gaussians*)

**save\_loica**(*obj, filename*)

**step\_biomass**(*t, od0, start*)

**step\_growth\_rate**(*t, start*)

## INTRODUCTION

Welcome to the LOICA (Logical Operators for Integrated Cell Algorithms) repository, our Python package for designing, modeling and characterizing genetic networks.

As you may have noticed, our logo features a beautiful bird—loica ([Leistes loyca](#)); a bird native to Chile known for its particular red chest and legendary kindness, with which we share name.



## INSTALLATION

Installing LOICA is way easier than pronouncing it!

```
pip install loica
```

For more details please refer to our [Wiki](#) for installation instructions and developer guides.



## LOICA ALLOWS YOU TO:

- Compile Code into DNA fragments that execute Cell Algorithms
- Easy programation of genetic network models
- Generation of synthetic data
- Communicate with Flapjack
- Use and output SBOL files
- Use all sorts of cellular computation
- Easy, fluid and customisable DNA design





## **TUTORIALS**

Now that you have LOICA installed you can familiarize yourself with the tool using the [Jupyter notebook](#) tutorials designed for this purpose.



## PYTHON MODULE INDEX

|

- loica, 9
- loica.assay, 21
- loica.colony, 22
- loica.geneproduct, 22
- loica.genetic\_network, 23
- loica.metabolism, 24
- loica.operators, 9
  - loica.operators.hill1, 9
  - loica.operators.hill2, 10
  - loica.operators.operator, 11
  - loica.operators.receiver, 11
  - loica.operators.source, 12
  - loica.operators.sum, 13
- loica.sample, 25
- loica.supplement, 26
- loica.util, 26



## Symbols

\_\_str\_\_() (*GeneProduct* method), 22, 30  
 \_\_str\_\_() (*Hill1* method), 10, 14, 31  
 \_\_str\_\_() (*Hill2* method), 10, 15, 32, 33  
 \_\_str\_\_() (*Operator* method), 11, 16, 17, 33  
 \_\_str\_\_() (*Receiver* method), 12, 17–19, 34, 35  
 \_\_str\_\_() (*Source* method), 13, 19–21, 36, 37  
 \_\_str\_\_() (*Sum* method), 13, 21, 37  
 \_\_str\_\_() (*Supplement* method), 26, 37

## A

add\_operator() (*GeneticNetwork* method), 23, 30  
 add\_regulator() (*GeneticNetwork* method), 23, 30  
 add\_reporter() (*GeneticNetwork* method), 23, 30  
 Assay (class in *loica*), 28  
 Assay (class in *loica.assay*), 21

## B

biomass() (*DataMetabolism* method), 25, 29

## C

characterize() (*Hill1* method), 10, 14, 31  
 characterize() (*Hill2* method), 11, 15, 32, 33  
 characterize() (*Receiver* method), 12, 17–19, 34, 35  
 characterize() (*Source* method), 13, 19–21, 36, 37  
 characterize\_growth() (in module *loica*), 38  
 characterize\_growth() (in module *loica.util*), 27  
 Colony (class in *loica*), 29  
 Colony (class in *loica.colony*), 22  
 color (*Sum* attribute), 13, 21, 37

## D

DataMetabolism (class in *loica*), 29  
 DataMetabolism (class in *loica.metabolism*), 24  
 draw() (*GeneticNetwork* method), 23, 30

## E

express() (*GeneProduct* method), 23, 30  
 expression\_rate() (*Hill1* method), 10, 15, 31  
 expression\_rate() (*Hill2* method), 11, 15, 32, 33

expression\_rate() (*Receiver* method), 12, 17–19, 34, 35  
 expression\_rate() (*Source* method), 13, 19–21, 36, 37  
 expression\_rate() (*Sum* method), 13, 21, 37

## F

forward\_model() (*Hill1* method), 10, 15, 31  
 forward\_model() (*Hill2* method), 11, 15, 32, 33  
 forward\_model() (*Receiver* method), 12, 17–19, 34, 35  
 forward\_model() (*Source* method), 13, 19–21, 36, 37  
 forward\_model\_growth() (in module *loica*), 38  
 forward\_model\_growth() (in module *loica.util*), 27  
 fun() (*Colony* method), 22, 29

## G

GeneProduct (class in *loica*), 29  
 GeneProduct (class in *loica.geneproduct*), 22  
 GeneticNetwork (class in *loica*), 30  
 GeneticNetwork (class in *loica.genetic\_network*), 23  
 gompertz() (in module *loica*), 38  
 gompertz() (in module *loica.metabolism*), 25  
 gompertz\_growth\_rate() (in module *loica*), 38  
 gompertz\_growth\_rate() (in module *loica.metabolism*), 25  
 growth\_rate() (*DataMetabolism* method), 25, 29

## H

Hill1 (class in *loica*), 30, 31  
 Hill1 (class in *loica.operators*), 14  
 Hill1 (class in *loica.operators.hill1*), 9  
 Hill2 (class in *loica*), 32  
 Hill2 (class in *loica.operators*), 15  
 Hill2 (class in *loica.operators.hill2*), 10

## I

initialize() (*GeneProduct* method), 23, 30  
 initialize() (*GeneticNetwork* method), 23, 30  
 initialize() (*Sample* method), 26, 36

## K

kymograph() (*Colony* method), 22, 29

## L

`load_loica()` (in module *loica*), 38  
`load_loica()` (in module *loica.util*), 27  
*loica*  
  module, 9  
*loica.assay*  
  module, 21  
*loica.colony*  
  module, 22  
*loica.geneproduct*  
  module, 22  
*loica.genetic\_network*  
  module, 23  
*loica.metabolism*  
  module, 24  
*loica.operators*  
  module, 9  
*loica.operators.hill1*  
  module, 9  
*loica.operators.hill2*  
  module, 10  
*loica.operators.operator*  
  module, 11  
*loica.operators.receiver*  
  module, 11  
*loica.operators.source*  
  module, 12  
*loica.operators.sum*  
  module, 13  
*loica.sample*  
  module, 25  
*loica.supplement*  
  module, 26  
*loica.util*  
  module, 26

## M

`map_kymo()` (Colony method), 22, 29  
*Metabolism* (class in *loica*), 33  
*Metabolism* (class in *loica.metabolism*), 25  
module  
  *loica*, 9  
  *loica.assay*, 21  
  *loica.colony*, 22  
  *loica.geneproduct*, 22  
  *loica.genetic\_network*, 23  
  *loica.metabolism*, 24  
  *loica.operators*, 9  
  *loica.operators.hill1*, 9  
  *loica.operators.hill2*, 10  
  *loica.operators.operator*, 11  
  *loica.operators.receiver*, 11  
  *loica.operators.source*, 12  
  *loica.operators.sum*, 13

*loica.sample*, 25  
  *loica.supplement*, 26  
  *loica.util*, 26

## N

`norm_kymo()` (Colony method), 22, 29

## O

*Operator* (class in *loica*), 33  
*Operator* (class in *loica.operators*), 15–17  
*Operator* (class in *loica.operators.operator*), 11

## R

`ramp_biomass()` (in module *loica*), 38  
`ramp_biomass()` (in module *loica.metabolism*), 25  
`ramp_growth_rate()` (in module *loica*), 38  
`ramp_growth_rate()` (in module *loica.metabolism*), 25  
*Receiver* (class in *loica*), 34  
*Receiver* (class in *loica.operators*), 17, 18  
*Receiver* (class in *loica.operators.receiver*), 11  
*Regulator* (class in *loica*), 35  
*Regulator* (class in *loica.geneproduct*), 23  
*Reporter* (class in *loica*), 35  
*Reporter* (class in *loica.geneproduct*), 23  
`residuals()` (Hill1 method), 10, 15, 31  
`residuals()` (Hill2 method), 11, 15, 32, 33  
`residuals()` (Receiver method), 12, 17–19, 34, 35  
`residuals()` (Source method), 13, 19–21, 36, 37  
`residuals_growth()` (in module *loica*), 38  
`residuals_growth()` (in module *loica.util*), 27  
`run()` (Assay method), 22, 29

## S

*Sample* (class in *loica*), 35  
*Sample* (class in *loica.sample*), 25  
`save_loica()` (in module *loica*), 38  
`save_loica()` (in module *loica.util*), 27  
`set_regulator()` (Sample method), 26, 36  
`set_reporter()` (Sample method), 26, 36  
`set_supplement()` (Sample method), 26, 36  
`shape` (GeneProduct attribute), 22, 29  
`shape` (Sum attribute), 13, 21, 37  
*SimulatedMetabolism* (class in *loica*), 36  
*SimulatedMetabolism* (class in *loica.metabolism*), 25  
*Source* (class in *loica*), 36  
*Source* (class in *loica.operators*), 19, 20  
*Source* (class in *loica.operators.source*), 12  
`step()` (GeneProduct method), 23, 30  
`step()` (GeneticNetwork method), 24, 30  
`step()` (Sample method), 26, 36  
`step_biomass()` (in module *loica*), 38  
`step_biomass()` (in module *loica.metabolism*), 25  
`step_growth_rate()` (in module *loica*), 38

`step_growth_rate()` (*in module loica.metabolism*), 25  
`step_stochastic()` (*GeneticNetwork method*), 24, 30  
`substep_stochastic()` (*GeneticNetwork method*), 24, 30

`Sum` (*class in loica*), 37

`Sum` (*class in loica.operators*), 21

`Sum` (*class in loica.operators.sum*), 13

`Supplement` (*class in loica*), 37

`Supplement` (*class in loica.supplement*), 26

## T

`to_contracted_graph()` (*GeneticNetwork method*), 24, 30

`to_graph()` (*GeneticNetwork method*), 24, 30

`to_sbol()` (*GeneticNetwork method*), 24, 30

## U

`upload()` (*Assay method*), 22, 29