# User Documentation

1 Getting Started
   1.1 Basic example ......................................................... 3
   1.2 Advanced example .................................................. 5
   1.3 Runner example .................................................... 6

2 Guides
   2.1 Git Beginners Guide ............................................... 9
   2.2 MinGW Installation Guide - Windows ............................. 12

3 Support
   3.1 Usage Questions .................................................... 13
   3.2 Reporting bugs ..................................................... 13

4 Changelog
   4.1 2.0.0rc2 (Aug 30, 2018) ....................................... 15
   4.2 2.0.0rc1 (Aug 30, 2018) ....................................... 15
   4.3 1.0.1 (Mar 21, 2018) ........................................... 16
   4.4 1.0.0 (Feb 28, 2018) ........................................... 16
   4.5 1.0.0rc2 (Feb 28, 2018) ....................................... 16
   4.6 1.0.0rc1 (Feb 28, 2018) ....................................... 17

5 Installation
   5.1 Setup development environment .................................. 19

6 Testing ................................................................. 21

7 Documentation .......................................................... 23

8 API
   8.1 NiaPy ................................................................. 25
   8.2 NiaPy.algorithms .................................................. 26
   8.3 NiaPy.benchmarks ................................................ 56

9 About
   9.1 Mission .............................................................. 83
   9.2 Licence ............................................................... 83
   9.3 Disclaimer ........................................................... 83
10 Contributing to NiaPy .......................... 85
  10.1 Code of Conduct .......................................... 85
  10.2 How Can I Contribute? ................................. 85

11 Code of Conduct ........................................ 87
  11.1 Our Pledge ............................................... 87
  11.2 Our Standards .......................................... 87
  11.3 Our Responsibilities ................................. 88
  11.4 Scope ................................................. 88
  11.5 Enforcement ........................................ 88
  11.6 Attribution ........................................... 88

Python Module Index .................................... 89
Python micro framework for building nature-inspired algorithms.

Nature-inspired algorithms are a very popular tool for solving optimization problems. Since the beginning of their era, numerous variants of nature-inspired algorithms were developed. To prove their versatility, those were tested in various domains on various applications, especially when they are hybridized, modified or adapted. However, implementation of nature-inspired algorithms is sometimes difficult, complex and tedious task. In order to break this wall, NiaPy is intended for simple and quick use, without spending a time for implementing algorithms from scratch.

The main documentation is organized into a couple sections:

- User Documentation
- Developer Documentation
- About NiaPy
It’s time to write your first NiaPy example. Firstly, if you haven’t already, install NiaPy package on your system using following command:

```
pip install NiaPy
```

When package is successfully installed you are ready to write you first example.

### 1.1 Basic example

In this example, let’s say, we want to try out Gray Wolf Optimizer algorithm against Pintér benchmark function. Firstly, we have to create new file, with name, for example `basic_example.py`. Then we have to import chosen algorithm from NiaPy, so we can use it. Afterwards we initialize GreyWolfOptimizer class instance and run the algorithm. Given bellow is complete source code of basic example.

```python
from NiaPy.algorithms.basic import GreyWolfOptimizer

# we will run 10 repetitions of Grey Wolf Optimizer against Pinter benchmark function
for i in range(10):
    # first parameter takes dimension of problem
    # second parameter is population size
    # third parameter takes the number of function evaluations
    # fourth parameter is benchmark function
    algorithm = GreyWolfOptimizer(10, 20, 10000, 'pinter')

    # running algorithm returns best found minimum
    best = algorithm.run()

    # printing best minimum
    print(best)
```

Given example can be run with `python basic_example.py` command and should give you similar output as following:
1.1.1 Customize benchmark bounds

By default, Pintér benchmark has the bound set to -10 and 10. We can simply override those predefined values very easily. We will modify our basic example to run Grey Wolf Optimizer against Pintér benchmark function with custom benchmark bounds set to -5 and 5. Given below is complete source code of customized basic example.

```python
from NiaPy.algorithms.basic import GreyWolfOptimizer
from NiaPy.benchmarks import Pinter

# initialize Pinter benchmark with custom bound
pinterCustom = Pinter(-5, 5)

# we will run 10 repetitions of Grey Wolf Optimizer against Pinter benchmark function
for i in range(10):
    # first parameter takes dimension of problem
    # second parameter is population size
    # third parameter takes the number of function evaluations
    # fourth parameter is benchmark function
    algorithm = GreyWolfOptimizer(10, 20, 10000, pinterCustom)

    # running algorithm returns best found minimum
    best = algorithm.run()

    # printing best minimum
    print(best)
```

Given example can be run with `python basic_example.py` command and should give you similar output as following:

7.43266143347e-64
1.45053917474e-58
1.01835349035e-55
6.50410738064e-59
2.18186445002e-61
3.20274657669e-63
3.23728585089e-62
1.78481271215e-63
7.81043837076e-66
7.3094339032e-64
1.2 Advanced example

In this example we will show you how to implement your own benchmark function and use it with any of implemented algorithms. First let’s create new file named advanced_example.py. As in the previous examples we will import algorithm we want to use from NiaPy module.

For our custom benchmark function, we have to create new class. Let’s name it MyBenchmark. In the initialization method of MyBenchmark class we have to set Lower and Upper bounds of the function. Afterwards we have to implement a function which returns evaluation function which takes two parameters D (as dimension of problem) and sol (as solution of problem). Now we should have something similar as is shown in code snippet below.

```python
from NiaPy.algorithms.basic import GreyWolfOptimizer

# our custom benchmark class
class MyBenchmark(object):
    def __init__(self):
        # define lower bound of benchmark function
        self.Lower = -11
        # define upper bound of benchmark function
        self.Upper = 11

    # function which returns evaluate function
    def function(self):
        def evaluate(D, sol):
            val = 0.0
            for i in range(D):
                val = val + sol[i] * sol[i]
            return val
        return evaluate
```

Now, all we have to do is to initialize our algorithm as in previous examples and pass as benchmark parameter, instance of our MyBenchmark class.

```python
for i in range(10):
    algorithm = GreyWolfOptimizer(10, 20, 10000, MyBenchmark())
    best = algorithm.run()
    print(best)
```

Now we can run our advanced example with following command python advanced_example.py. The results should be similar to those bellow.

```
1.99601075063e-63
1.03831459307e-65
6.76105610278e-63
2.39738295065e-64
1.11826744557e-46
1.95914350691e-65
6.33575259075e-58
9.84100808621e-68
2.62423542073e-66
4.20503964752e-64
```
1.3 Runner example

For easier comparison between many different algorithms and benchmarks, we developed a useful feature called Runner. Runner can take an array of algorithms and an array of benchmarks to compare and run all combinations for you. We also provide an extra feature, which lets you easily exports those results in many different formats (LaTeX, Excell, JSON).

Below is given a usage example of our Runner, which will run three given algorithms and four given benchmark functions. Results will be exported as JSON.

```python
import NiaPy

class MyBenchmark(object):
    def __init__(self):
        self.Lower = -5.12
        self.Upper = 5.12

    def function(self):
        def evaluate(D, sol):
            val = 0.0
            for i in range(D):
                val = val + sol[i] * sol[i]
            return val
        return evaluate

algorithms = ['DifferentialEvolutionAlgorithm',
              'ArtificialBeeColonyAlgorithm',
              'GreyWolfOptimizer']
benchmarks = ['ackley', 'whitley', 'alpine2', MyBenchmark()]

NiaPy.Runner(10, 40, 10000, 3, algorithms, benchmarks).run(export='json',
                                                           verbose=True)
```

Output of running above example should look like something as following.

```
Running DifferentialEvolutionAlgorithm...
Running DifferentialEvolutionAlgorithm algorithm on ackley benchmark...
Running DifferentialEvolutionAlgorithm algorithm on whitley benchmark...
Running DifferentialEvolutionAlgorithm algorithm on alpine2 benchmark...
Running DifferentialEvolutionAlgorithm algorithm on MyBenchmark benchmark...
---------------------------------------------------
Running ArtificialBeeColonyAlgorithm...
Running ArtificialBeeColonyAlgorithm algorithm on ackley benchmark...
Running ArtificialBeeColonyAlgorithm algorithm on whitley benchmark...
Running ArtificialBeeColonyAlgorithm algorithm on alpine2 benchmark...
Running ArtificialBeeColonyAlgorithm algorithm on MyBenchmark benchmark...
---------------------------------------------------
Running GreyWolfOptimizer...
Running GreyWolfOptimizer algorithm on ackley benchmark...
Running GreyWolfOptimizer algorithm on whitley benchmark...
Running GreyWolfOptimizer algorithm on alpine2 benchmark...
Running GreyWolfOptimizer algorithm on MyBenchmark benchmark...
---------------------------------------------------
Export to JSON completed!
```

Results exported as JSON should look like this.
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Benchmark 1</th>
<th>Benchmark 2</th>
<th>Benchmark 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>GreyWolfOptimizer</td>
<td>6.766062076017854e-46, 2.642653581097554e-43, 8.658015542865062e-44</td>
<td>4.440892098500626e-16, 4.440892098500626e-16, 4.440892098500626e-16</td>
<td>41.15672884009374, 45.405829107898754, 45.285854036223746</td>
</tr>
<tr>
<td>Ackley</td>
<td>4.440892098500626e-16, 4.440892098500626e-16, 4.440892098500626e-16</td>
<td>0.0001596817850928467, 0.0017004800794961916, 0.00018082865898749745</td>
<td>20.622549664235308, 14.085647205633876, 1.838650658412531</td>
</tr>
<tr>
<td>Whitley</td>
<td>41.15672884009374, 45.405829107898754, 45.285854036223746</td>
<td>20.622549664235308, 14.085647205633876, 1.838650658412531</td>
<td>-23686.224202267975, -23678.92101630358, -14320.040364388877</td>
</tr>
</tbody>
</table>

1.3. Runner example
59.35951990376928, 58.805393587160424, 63.532977687055386 },
  "alpine2": [ -23698.80535644514, -19925.409402805282, -23500.48062034027 ]
]
})
}
Here are gathered together user guides.

2.1 Git Beginners Guide

Beginner’s guide on how to contribute to open source community

**Note:** If you don’t have any previous experience with using Git, we recommend you take a [15 minutes long Git Tutorial](#).

Whether you’re trying to give back to the open source community or collaborating on your own projects, knowing how to properly fork and generate pull requests is essential. Unfortunately, it’s quite easy to make mistakes or not know what you should do when you’re initially learning the process. I know that I certainly had considerable initial trouble with it, and I found a lot of the information on GitHub and around the internet to be rather piecemeal and incomplete - part of the process described here, another there, common hang-ups in a different place, and so on.

This short tutorial is fairly standard procedure for creating a fork, doing your work, issuing a pull request, and merging that pull request back into the original project.

### 2.1.1 Create a fork

Just head over to the our [GitHub page](#) and click the “Fork” button. It’s just that simple. Once you’ve done that, you can use your favorite git client to clone your repo or just head straight to the command line:

```
git clone git@github.com:<your-username>/<fork-project>
```
Keep your fork up to date

In most cases you’ll probably want to make sure you keep your fork up to date by tracking the original “upstream” repo that you forked. To do this, you’ll need to add a remote if not already added:

```bash
# Add 'upstream' repo to list of remotes
git remote add upstream git://github.com/NiaOrg/NiaPy.git

# Verify the new remote named 'upstream'
git remote -v
```

Whenever you want to update your fork with the latest upstream changes, you’ll need to first fetch the upstream repo’s branches and latest commits to bring them into your repository:

```bash
# Fetch from upstream remote
git fetch upstream
```

Now, checkout your own master branch and rebase with the upstream repo’s master branch:

```bash
# Checkout your master branch and merge upstream

git checkout master

git merge upstream/master
```

If there are no unique commits on the local master branch, git will simply perform a fast-forward. However, if you have been making changes on master (in the vast majority of cases you probably shouldn’t be - see the next section Doing your work, you may have to deal with conflicts. When doing so, be careful to respect the changes made upstream.

Now, your local master branch is up-to-date with everything modified upstream.

2.1.2 Doing your work

Create a Branch

Whenever you begin work on a new feature or bug fix, it’s important that you create a new branch. Not only is it proper git workflow, but it also keeps your changes organized and separated from the master branch so that you can easily submit and manage multiple pull requests for every task you complete.

To create a new branch and start working on it:

```bash
# Checkout the master branch - you want your new branch to come from master

git checkout master

# Create a new branch named newfeature (give your branch its own simple informative name)

git branch newfeature

# Switch to your new branch

git checkout newfeature

# Last two commands can be joined as following: git checkout -b newfeature
```

Now, go to town hacking away and making whatever changes you want to
### 2.1.3 Submitting a Pull Request

#### Cleaning Up Your Work

Prior to submitting your pull request, you might want to do a few things to clean up your branch and make it as simple as possible for the original repo’s maintainer to test, accept, and merge your work.

If any commits have been made to the upstream master branch, you should rebase your development branch so that merging it will be a simple fast-forward that won’t require any conflict resolution work.

```bash
# Fetch upstream master and merge with your repo’s master branch
git fetch upstream
git checkout master
git merge upstream/master

# If there were any new commits, rebase your development branch
git checkout newfeature
git rebase master
```

Now, it may be desirable to squash some of your smaller commits down into a small number of larger more cohesive commits. You can do this with an interactive rebase:

```bash
# Rebase all commits on your development branch
git checkout
git rebase -i master
```

This will open up a text editor where you can specify which commits to squash.

#### Submitting

Once you’ve committed and pushed all of your changes to GitHub, go to the page for your fork on GitHub, select your development branch, and click the pull request button. If you need to make any adjustments to your pull request, just push the updates to GitHub. Your pull request will automatically track the changes on your development branch and update.

When pull request is successfully created, make sure you follow activity on your pull request. It may occur that the maintainer of project will ask you to do some more changes or fix something on your pull request before merging it to master branch.

After maintainer merges your pull request to master, you’re done with development on this branch, so you’re free to delete it.

```bash
git branch -d newfeature
```

### 2.1.4 Copyright

This guide is modified version of original one, written by Chase Pettit.

**Copyright**

Copyright 2017, Chase Pettit

**MIT License**

**Additional Reading**

- Atlassian - Merging vs. Rebasings
Sources

- GitHub - Fork a Repo
- GitHub - Syncing a Fork
- GitHub - Checking Out a Pull Request

2.2 MinGW Installation Guide - Windows

Download MinGW installer from [here](#).

**Warning:** Important! Before running the MinGW installer disable any running antivirus and firewall. Afterwards run MinGW installer as Administrator.

Follow the installation wizard clicking **Continue**.

After the installation procedure is completed MinGW Installation Manager is opened.

In tree navigation on the left side of window select **All Packages > MSYS** like is shown in figure below.

![MinGW Installation Manager](#)

On the right side of window, search for packages **msys-make** and **msys-bash**. Right click on each package and select **Mark for installation** from context menu.

Next click on the **Installation** in top menu and select **Apply Changes** and again **Apply**.

The last thing is to add binaries to system variables. Go to **Control panel > System and Security > System** and click on **Advanced system settings**. Then click on **Environment Variables...** button and on list in new window mark entry with variable **Path**. Next, click on **Edit...** button and create new entry with value equal to: `<MinGW_install_path>\msys\1.0\bin` (by default it is: `C:\MinGW\msys\1.0\bin`). Click **OK** on every window.

That's it! You are ready to contribute to our project!
3.1 Usage Questions

If you have questions about how to use Niapy or have an issue that isn’t related to a bug, you can place a question on StackOverflow.

You can also join us at our Slack Channel or seek support via niapy.organization@gmail.com.

NiaPy is a community supported package, nobody is paid to develop package nor to handle NiaPy support.

All people answering your questions are doing it with their own time, so please be kind and provide as much information as possible.

3.2 Reporting bugs

Check out Reporting bugs section in Contributing to NiaPy
We are using semantic versioning.

### 4.1 2.0.0rc2 (Aug 30, 2018)

- fix PyPI build

### 4.2 2.0.0rc1 (Aug 30, 2018)

Changes included in release:

- **Added algorithms:**
  - **basic:**
    - Camel algorithm
    - Evolution Strategy
    - Fireworks algorithm
    - Glowworm swarm optimization
    - Harmony search algorithm
    - Krill Herd Algorithm
    - Monkey King Evolution
    - Multiple trajectory search
    - Sine Cosine Algorithm
  - **modified:**
    - Dynamic population size self-adaptive differential evolution algorithm
NiaPy Documentation, Release 0.0.0.

- other:
  - Anarchic society optimization algorithm
  - Hill climbing algorithm
  - Multiple trajectory search
  - Nelder mead method or downhill simplex method or amoeba method
  - Simulated annealing algorithm

- Added benchmarks functions:
  - Discus
  - Dixon-Price
  - Elliptic
  - HGBat
  - Katsuura
  - Levy
  - Michalewicz
  - Perm
  - Powell
  - Sphere2 -> Sphere with different powers
  - Sphere3 -> Rotated hyper-ellipsoid
  - Trid
  - Weierstrass
  - Zakharov

- breaking changes in algorithms structure
- various bugfixes

4.3 1.0.1 (Mar 21, 2018)

This release reflects the changes from Journal of Open Source Software (JOSS) review: - Better API Documentation
- Clarification of set-up requirements in README - Improved paper

4.4 1.0.0 (Feb 28, 2018)

- stable release 1.0.0

4.5 1.0.0rc2 (Feb 28, 2018)

- fix PyPI build
4.6 1.0.0rc1 (Feb 28, 2018)

- version 1.0.0 release candidate 1
- added 10 algorithms
- added 26 benchmark functions
- added Runner utility with export functionality
CHAPTER 5

Installation

5.1 Setup development environment

5.1.1 Requirements

• Python: download (at least version 2.7.14, preferable 3.6.x)
• Pip: installation docs
• Make
  – Mac: download
  – Linux: download
• pipenv: docs (run pip install pipenv command)
• Pandoc: installation docs * optional
• Graphviz: download * optional

To confirm these system dependencies are configured correctly:

```make
doctor```

5.1.2 Installation of development dependencies

List of NiaPy’s dependencies:
List of development dependencies:

<table>
<thead>
<tr>
<th>Package</th>
<th>Version</th>
<th>Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>pylint</td>
<td>Any</td>
<td>Any</td>
</tr>
<tr>
<td>pycodestyle</td>
<td>Any</td>
<td>Any</td>
</tr>
<tr>
<td>pydocstyle</td>
<td>Any</td>
<td>Any</td>
</tr>
<tr>
<td>pytest</td>
<td>~=3.3</td>
<td>Any</td>
</tr>
<tr>
<td>pytest-describe</td>
<td>Any</td>
<td>Any</td>
</tr>
<tr>
<td>pytest-expecter</td>
<td>Any</td>
<td>Any</td>
</tr>
<tr>
<td>pytest-random</td>
<td>Any</td>
<td>Any</td>
</tr>
<tr>
<td>pytest-cov</td>
<td>Any</td>
<td>Any</td>
</tr>
<tr>
<td>freezegun</td>
<td>Any</td>
<td>Any</td>
</tr>
<tr>
<td>coverage-space</td>
<td>Any</td>
<td>Any</td>
</tr>
<tr>
<td>docutils</td>
<td>Any</td>
<td>Any</td>
</tr>
<tr>
<td>pygments</td>
<td>Any</td>
<td>Any</td>
</tr>
<tr>
<td>wheel</td>
<td>Any</td>
<td>Any</td>
</tr>
<tr>
<td>pyinstaller</td>
<td>Any</td>
<td>Any</td>
</tr>
<tr>
<td>twine</td>
<td>Any</td>
<td>Any</td>
</tr>
<tr>
<td>sniffer</td>
<td>Any</td>
<td>Any</td>
</tr>
<tr>
<td>macfsevents</td>
<td>Any</td>
<td>darwin</td>
</tr>
<tr>
<td>enum34</td>
<td>Any</td>
<td>Any</td>
</tr>
<tr>
<td>singledispatch</td>
<td>Any</td>
<td>Any</td>
</tr>
<tr>
<td>backports.functools-lru-cache</td>
<td>Any</td>
<td>Any</td>
</tr>
<tr>
<td>configparser</td>
<td>Any</td>
<td>Any</td>
</tr>
<tr>
<td>sphinx</td>
<td>Any</td>
<td>Any</td>
</tr>
<tr>
<td>sphinx-rtd-theme</td>
<td>Any</td>
<td>Any</td>
</tr>
<tr>
<td>funcsigs</td>
<td>Any</td>
<td>Any</td>
</tr>
<tr>
<td>futures</td>
<td>Any</td>
<td>Any</td>
</tr>
<tr>
<td>autopep8</td>
<td>Any</td>
<td>Any</td>
</tr>
<tr>
<td>sphinx-autobuild</td>
<td>Any</td>
<td>Any</td>
</tr>
</tbody>
</table>

Install project dependencies into a virtual environment:

```
make install
```

To enter created virtual environment with all installed development dependencies run:

```
pipenv shell
```
CHAPTER 6

Testing

Note:  We suppose that you already followed the Installation guide. If not, please do so before you continue to read this section.

Before making a pull request, if possible provide tests for added features or bug fixes.

We have an automated building system which also runs all of provided tests. In case any of the test cases fails, we are notified about failing tests. Those should be fixed before we merge your pull request to master branch.

For the purpose of checking if all test are passing locally you can run following command:

```
make test
```

If all tests passed running this command it is most likely that the tests would pass on our build system to.
Note: We suppose that you already followed the *Installation* guide. If not, please do so before you continue to read this section.

To locally generate and preview documentation run the following command in the project root folder:

```
sphinx-autobuild docs/source docs/build/html
```

If the build of the documentation is successful, you can preview the documentation by navigating to the `http://127.0.0.1:8000`.
This is the NiaPy API documentation, auto generated from the source code.

8.1 NiaPy

Python micro framework for building nature-inspired algorithms.

```python
class NiaPy.Runner(D, NP, nFES, nRuns, useAlgorithms, useBenchmarks, **kwargs)
Runner utility feature.

  Feature which enables running multiple algorithms with multiple benchmarks. It also support exporting results in various formats (e.g. LaTeX, Excel, JSON)

  Initialize Runner.

  __init__(self, D, NP, nFES, nRuns, useAlgorithms, useBenchmarks, …)
Arguments: D {integer} – dimension of problem
NP {integer} – population size
nFES {integer} – number of function evaluations
nRuns {integer} – number of repetitions
useAlgorithms [] – array of algorithms to run
useBenchmarks [] – array of benchmarks to run
A {decimal} – laudness
r {decimal} – pulse rate
Qmin {decimal} – minimum frequency
Qmax {decimal} – maximum frequency
Pa {decimal} – probability
```
F {decimal} – scaling factor
F_l {decimal} – lower limit of scaling factor
F_u {decimal} – upper limit of scaling factor
CR {decimal} – crossover rate
alpha {decimal} – alpha parameter
betamin {decimal} – betamin parameter
gamma {decimal} – gamma parameter
p {decimal} – probability switch
Ts {decimal} – tournament selection
Mr {decimal} – mutation rate
C1 {decimal} – cognitive component
C2 {decimal} – social component
w {decimal} – inertia weight
vMin {decimal} – minimal velocity
vMax {decimal} – maximal velocity
Tao1 {decimal} –
Tao2 {decimal} –
n {integer} – number of sparks
mu {decimal} – mu parameter
omega {decimal} – TODO
S_init {decimal} – initial supply for camel
E_init {decimal} – initial endurance for camel
T_min {decimal} – minimal temperature
T_max {decimal} – maximal temperature
C_a {decimal} – Amplification factor
C_r {decimal} – Reduction factor
Limit {integer} – Limit
k {integer} – Number of runs before adaptive

8.2 NiaPy.algorithms

Module with implementations of basic and hybrid algorithms.

class NiaPy.algorithms.Algorithm(**kwargs)
   Bases: object

   Class for implementing algorithms.

   Data: 2018

   Author: Klemen Berkovič
License: MIT

Initialize algorithm and create name for an algorithm.

**Arguments:**

- `name {string}` – Full name of algorithm
- `shortName {string}` – Short name of algorithm
- `NP {integer}` – Population size
- `D {integer}` – Dimension of problem
- `nGEN {integer}` – Number of generations/iterations
- `nFES {integer}` – Number of function evaluations
- `benchmark {object}` – Benchmark implementation object
- `task {Task}` – Task to perform optimization on

**Raises:**

- `TypeError` – Raised when given benchmark function which does not exists.

See: Algorithm.setParameters(self, **kwargs)

`normal(loc, scale, D=None)`

Get D shape random normal distributed numbers.

**Arguments:**

- `loc {}` –
- `scale {}` –
- `D {array} or {int}` – Shape of returned random uniform numbers

`rand(D=1)`

Get random numbers of shape D in range from 0 to 1.

**Arguments:**

- `D {array} or {int}` – Shape of return random numbers

`randint(Nmax, D=1, Nmin=0, skip=[])`

Get D shape random full numbers in range Nmin to Nmax.

**Arguments:**

- `Nmin {integer}` –
- `Nmax {integer}` –
- `D {array} or {int}` – Shape of return random uniform numbers
- `skip {array}` – Numbers to skip

`run()`

Start the optimization.

See: Algorithm.runTask(self, taks)

`runTask(task)`

Start the optimization.

**Arguments:**

- `task {Task}` – Task with bounds and objective function for optimization
Return:
- solution {array} – point of best solution
- fitness {real} – fitness value of best solution

runYield(task)
Run the algorithm for only one iteration and return the best solution.

Arguments:
- task {Task} – Task with bounds and objective function for optimization

Return:
- solution {array} – point of best solution
- fitness {real} – fitness value of the best solution

setParameters(**kwargs)
Set the parameters/arguments of the algorithm.

Arguments:
- kwargs {dict} – Dictionary with values of the parameters

uniform(Lower, Upper, D=None)
Get D shape random uniform numbers in range from Lower to Upper.

Arguments:
- Lower {array} or {real} or {int} – Lower bound
- Upper {array} or {real} or {int} – Upper bound
- D {array} or {int} – Shape of returned random uniform numbers

class NiaPy.algorithms.Individual(**kwargs)
Bases: object
Class that represent one solution in population of solutions.

Date: 2018

Author: Klemen Berkovič

License: MIT

evaluate(task)
Evaluate the solution.

Arguments:
- task {Task} – Object with objective function for optimization

generateSolution(task, rnd=<module 'numpy.random' from '/home/docs/checkouts/readthedocs.org/user_builds/niapy/envs/stable/lib/python3.6/site-packages/numpy/random/__init__.py'>)
Generate new solution.

Arguments:
- task {Task}
- e {bool} – Eval the solution
- rnd {random} – Object for generating random numbers
repair(task)
    Reper solution and put the solution in the bounds of problem.

    Arguments:
    task {Task}

# 8.2.1 NiaPy.algorithms.basic

Implementation of basic nature-inspired algorithms.

class NiaPy.algorithms.basic.BatAlgorithm(**kwargs)
    Bases: NiaPy.algorithms.algorithm.Algorithm
    Implementation of Bat algorithm.

    Algorithm: Bat algorithm
    Date: 2015
    Authors: Iztok Fister Jr., Marko Burjek and Klemen Berkovič
    License: MIT


    __init__(self, D, NP, nFES, A, r, Qmin, Qmax, benchmark)
    See: Algorithm.__init__(self, **kwargs)

    runTask(task)
    Run algorithm with initialized parameters.

    Return:
    {decimal} – coordinates of minimal found objective function
    {decimal} – minimal value found of objective function

    setParameters(NP, A, r, Qmin, Qmax, **ukwargs)
    Set the parameters of the algorithm.

    Arguments:
    NP {integer} – population size
    A {decimal} – loudness
    r {decimal} – pulse rate
    Qmin {decimal} – minimum frequency
    Qmax {decimal} – maximum frequency

class NiaPy.algorithms.basic.FireflyAlgorithm(**kwargs)
    Bases: NiaPy.algorithms.algorithm.Algorithm
    Implementation of Firefly algorithm.

    Algorithm: Firefly algorithm
    Date: 2016
    Authors: Iztok Fister Jr, Iztok Fister and Klemen Berkovič
    License: MIT

\[
\text{alpha}_\text{new}(a, \alpha)
\]
Optional recalculate the new alpha value.

**getBest**(\(x_b, x_b_f, \text{Fireflies}, \text{Intensity}\))

**move_ffa**(\(i, \text{Fireflies}, \text{Intensity}, o\text{Fireflies}, \alpha, \text{task}\))
Move fireflies.

**runTask**(\(\text{task}\))
Run.

**setParameters**(\(NP=20, \alpha=1, \beta\text{amin}=1, \gamma=2, **\text{ukwargs}\))
Set the parameters of the algorithm.

**Arguments:**
- \(NP\) \{integer\} – population size
- \(\alpha\) \{decimal\} – alpha parameter
- \(\beta\text{amin}\) \{decimal\} – betamin parameter
- \(\gamma\) \{decimal\} – gamma parameter

**class** NiaPy.algorithms.basic.DifferentialEvolutionAlgorithm(**\text{kwargs}\)
Bases: NiaPy.algorithms.algorithm.Algorithm
Implementation of Differential evolution algorithm.

**Algorithm:** Differential evolution algorithm
**Date:** 2018
**Author:** Uros Mlakar and Klemen Berkovič
**License:** MIT


**evalPopulation**(\(x, x_\text{old}, \text{task}\))
Evaluate element.

**runTask**(\(\text{task}\))
Run.

**selectBetter**(\(x, y\))

**setParameters**(\(NP=25, F=2, CR=0.2, \text{CrossMutt=\langle function CrossRand1 \rangle}, **\text{ukwargs}\))
Set the algorithm parameters.

**Arguments:**
- \(NP\) \{integer\} – population size
- \(F\) \{decimal\} – scaling factor
- \(CR\) \{decimal\} – crossover rate
- \(\text{CrossMutt}\) \{function\} – crossover and mutation strategy

**class** NiaPy.algorithms.basic.FlowerPollinationAlgorithm(**\text{kwargs}\)
Bases: NiaPy.algorithms.algorithm.Algorithm
Implementation of Flower Pollination algorithm.
Algorithm: Flower Pollination algorithm

Date: 2018

Authors: Dusan Fister, Iztok Fister Jr. and Klemen Berkovič

License: MIT


levy()

repair(x, task)
    Find limits.

runTask(task)
    Start the optimization.

Arguments:
    task {Task} – Task with bounds and objective function for optimization

Return:
    solution {array} – point of best solution
    fitness {real} – fitness value of best solution

setParameters(NP=25, p=0.35, beta=1.5, **kwargs)

__init__(self, D, NP, nFES, p, benchmark).

Arguments:
    NP {integer} – population size
    p {decimal} – probability switch
    beta {real} –

class NiaPy.algorithms.basic.GreyWolfOptimizer(**kwargs)
Bases: NiaPy.algorithms.algorithms.Algorithm

Implementation of Grey wolf optimizer.

Algorithm: Grey wolf optimizer

Date: 2018

Author: Iztok Fister Jr. and Klemen Berkovič

License: MIT


repair(x, task)
    Find limits.

runTask(task)
    Run.
setParameters(NP=25, **kwargs)
Set the algorithm parameters.

Arguments:
NP {integer} – Number of individuals in population

class NiaPy.algorithms.basic.GeneticAlgorithm(**kwargs)
Bases: NiaPy.algorithms.algorithm.Algorithm
Implementation of Genetic algorithm.
Algorithm: Genetic algorithm
Date: 2018
Author: Uros Mlakar and Klemen Berkovič
License: MIT

evolve(pop, x_b, task)
runTask(task)
Start the optimization.

Arguments:
task {Task} – Task with bounds and objective function for optimization

Return:
solution {array} – point of best solution
fitness {real} – fitness value of best solution

setParameters(NP=25, Ts=5, Mr=0.25, Cr=0.25, Selection=<function TurnamentSelection>,
Crossover=<function UniformCrossover>, Mutation=<function UniformMutation>, **kwargs)
Set the parameters of the algorithm.

Arguments:
NP {integer} – population size
Ts {integer} – tournament selection
Mr {decimal} – mutation rate
Cr {decimal} – crossover rate

class NiaPy.algorithms.basic.ArtificialBeeColonyAlgorithm(**kwargs)
Bases: NiaPy.algorithms.algorithm.Algorithm
Implementation of Artificial Bee Colony algorithm.
Algorithm: Artificial Bee Colony algorithm
Date: 2018
Author: Uros Mlakar and Klemen Berkovič
License: MIT


__init__(self, D, NP, nFES, benchmark).
See: Algorithm.__init__(self, **kwargs)
CalculateProbs()
    Calculate probs.

checkForBest(Solution)
    Check best solution.

init(task)
    Initialize positions.

runTask(task)
    Run.

setParameters(NP=10, Limit=100, **ukwargs)
    Set the arguments of an algorithm.

Arguments:
    NP {integer} – population size
    Limit {integer} – Limit

class NiaPy.algorithms.basic.ParticleSwarmAlgorithm(**kwargs)
    Bases: NiaPy.algorithms.algorithm.Algorithm

    Implementation of Particle Swarm Optimization algorithm.

    Algorithm: Particle Swarm Optimization algorithm

    Date: 2018

    Authors: Lucija Brezočnik, Grega Vrbančič, Iztok Fister Jr. and Klemen Berkovič

    License: MIT


    init(task)
    repair(x, l, u)

runTask(task)
    Move particles in search space.

setParameters(NP=25, C1=2.0, C2=2.0, w=0.7, vMin=-4, vMax=4, **ukwargs)
    Set the parameters for the algorithm.

Arguments:
    NP {integer} – population size
    C1 {decimal} – cognitive component
    C2 {decimal} – social component
    w {decimal} – inertia weight
    vMin {decimal} – minimal velocity
    vMax {decimal} – maximal velocity

class NiaPy.algorithms.basic.BareBonesFireworksAlgorithm(**kwargs)
    Bases: NiaPy.algorithms.algorithm.Algorithm

    Implementation of bare bone fireworks algorithm.

    Algorithm: Bare Bones Fireworks Algorithm
runTask (task)
   Start the optimization.
   
   Arguments:
   task {Task} – Task with bounds and objective function for optimization
   
   Return:
   solution {array} – point of best solution
   fitness {real} – fitness value of best solution

setParameters (n=10, C_a=1.5, C_r=0.5, **ukwargs)
   Set the arguments of an algorithm.
   
   Arguments:
   n {integer} – number of sparks $\in [1, \infty)$
   C_a {real} – amplification coefficient $\in [1, \infty)$
   C_r {real} – reduction coefficient $\in (0, 1)$

class NiaPy.algorithms.basic.CamelAlgorithm(**kwargs)
   Bases: NiaPy.algorithms.algorithm.Algorithm

   Implementation of Camel traveling behavior.
   
   Algorithm: Camel algorithm
   
   Date: 2018
   
   Authors: Klemen Berkovič
   
   License: MIT

   Reference URL: https://www.iasj.net/iasj?func=fulltext&aId=118375


lifeCycle (c, fit, fitn, mu, task)
oasis (c, rn, fit, fitn, alpha)
runTask (task)
   Start the optimization.
   
   Arguments:
   task {Task} – Task with bounds and objective function for optimization
   
   Return:
   solution {array} – point of best solution
fitness {real} – fitness value of best solution

**setParameters**(NP=50, omega=0.25, mu=0.5, alpha=0.5, S_init=10, E_init=10, T_min=-10, T_max=10, **kwargs)

Set the arguments of an algorithm.

**Arguments:**

NP {integer} – population size \( \in [1, \infty) \)

T_min {real} – minimum temperature, must be true \( T_{\text{min}} < T_{\text{max}} \)

T_max {real} – maximum temperature, must be true \( T_{\text{min}} < T_{\text{max}} \)

omega {real} – burden factor \( \in [0, 1] \)

mu {real} – dying rate \( \in [0, 1] \)

S_init {real} – initial supply \( \in (0, \infty) \)

E_init {real} – initial endurance \( \in (0, \infty) \)

**walk** (c, fit, task, omega, c_best)

**class** NiaPy.algorithms.basic.MonkeyKingEvolutionV1(**kwargs)

**Bases:** NiaPy.algorithms.algorithm.Algorithm

Implementation of monkey king evolution algorithm version 1.

**Algorithm:** Monkey King Evolution version 1

**Date:** 2018

**Authors:** Klemen Berkovič

**License:** MIT

**Reference URL:** https://www.sciencedirect.com/science/article/pii/S0950705116000198


**moveMK** (x, task)

**moveMokeyKingPartice** (p, task)

**moveP** (x, x_pb, x_b, task)

**movePartice** (p, p_b, task)

**movePopulation** (pop, p_b, task)

**repair** (x, task)

**runTask** (task)

Start the optimization.

**Arguments:**

task {Task} – Task with bounds and objective function for optimization

**Return:**

solution {array} – point of best solution

fitness {real} – fitness value of best solution
setParameters \((NP=40, F=0.7, R=0.3, C=3, FC=0.5, **ukwargs)\)

Set the algorithm parameters.

**Arguments:**

NP {integer} – Size of population
F {real} – param
R {real} – param
C {real} – param
FC {real} – param

**class** NiaPy.algorithms.basic.MonkeyKingEvolutionV2(**kwargs)

Implementation of monkey king evolution algorithm version 2.

---

**class** NiaPy.algorithms.basic.MonkeyKingEvolutionV3(**kwargs)

Implementation of monkey king evolution algorithm version 3.

---

**eval** \((X, x, x_f, task)\)

**neg** \((x)\)

**runTask** \((task)\)

Start the optimization.

**Arguments:**

task {Task} – Task with bounds and objective function for optimization
class NiaPy.algorithms.basic.EvolutionStrategy1p1(**kwargs)
Bases: NiaPy.algorithms.algorithm.Algorithm

Implementation of (1 + 1) evolution strategy algorithm. Uses just one individual.

Algorithm: (1 + 1) Evolution Strategy Algorithm

Date: 2018

Authors: Klemen Berkovič

License: MIT

Reference URL:
Reference paper:

mutate (x, rho)

runTask (task)
    Start the optimization.
    
    Arguments:
    task {Task} – Task with bounds and objective function for optimization
    
    Return:
    solution {array} – point of best solution
    fitness {real} – fitness value of best solution

setParameters (mu=1, k=10, c_a=1.1, c_r=0.5, epsilon=1e-20, **ukwargs)
    Set the arguments of an algorithm.
    
    Arguments:
    mu {integer} –
    k {integer} –
    c_a {real} –
    c_r {real} –

updateRho (rho, k)

class NiaPy.algorithms.basic.EvolutionStrategyMp1(**kwargs)
Bases: NiaPy.algorithms.basic.es.EvolutionStrategy1p1

Implementation of (mu + 1) evolution strategy algorithm. Algorithm creates mu mutants but into new generation goes only one individual.

Algorithm: (\mu + 1) Evolution Strategy Algorithm

Date: 2018

Authors: Klemen Berkovič

License: MIT

Reference URL:
Reference paper:
**setParameters** (**kwargs**)

Set the arguments of an algorithm.

**Arguments:**
- mu {integer} –
- k {integer} –
- c_a {real} –
- c_r {real} –

**class** NiaPy.algorithms.basic.EvolutionStrategyMpL (**kwargs**)

**Bases:** NiaPy.algorithms.basic.es.EvolutionStrategy1p1

Implementation of (mu + lambda) evolution strategy algorithm. Mutation creates lambda individual. Lambda individual compete with mu individuals for survival, so only mu individual go to new generation.

**Algorithm:** ($mu$ + $lambda$) Evolution Strategy Algorithm

**Date:** 2018

**Authors:** Klemen Berkovič

**License:** MIT

**Reference URL:**

**Reference paper:**

**changeCount** (a, b)

**mutate** (x, rho)

**mutateRepair** (pop, task)

**runTask** (task)

Start the optimization.

**Arguments:**
- task {Task} – Task with bounds and objective function for optimization

**Return:**
- solution {array} – point of best solution
- fitness {real} – fitness value of best solution

**setParameters** (**kwargs**)

Set the arguments of an algorithm.

**Arguments:**
- mu {integer} –
- k {integer} –
- c_a {real} –
- c_r {real} –

**updateRho** (pop, k)

**class** NiaPy.algorithms.basic.EvolutionStrategyML (**kwargs**)

**Bases:** NiaPy.algorithms.basic.es.EvolutionStrategyMpL

38 Chapter 8. API
Implementation of (mu, lambda) evolution strategy algorithm. Algorithm is good for dynamic environments. Mu individual create lambda shields. Only best mu shields go to new generation. Mu parents are discarded.

**Algorithm:** ($mu$ + $lambda$) Evolution Strategy Algorithm

**Date:** 2018

**Authors:** Klemen Berkovič

**License:** MIT

**Reference URL:**

```
newPop (pop)
```

```
runTask (task)
```

Start the optimization.

**Arguments:**

```
task {Task} – Task with bounds and objective function for optimization
```

**Return:**

```
solution {array} – point of best solution
fitness {real} – fitness value of best solution
```

``` python
class NiaPy.algorithms.basic.SineCosineAlgorithm(**kwargs)

Bases: NiaPy.algorithms.algorithm.Algorithm

Implementation of sine cosine algorithm.

**Algorithm:** Sine Cosine Algorithm

**Date:** 2018

**Authors:** Klemen Berkovič

**License:** MIT

**Reference URL:** https://www.sciencedirect.com/science/article/pii/S0950705115005043


```
nextPos (x, x_b, r1, r2, r3, r4, task)
```

```
runTask (task)
```

Start the optimization.

**Arguments:**

```
task {Task} – Task with bounds and objective function for optimization
```

**Return:**

```
solution {array} – point of best solution
fitness {real} – fitness value of best solution
```

```
setParameters (NP=25, a=3, Rmin=0, Rmax=2, **ukwargs)
```

Set the arguments of an algorithm.

**Arguments:**
NP {integer} – number of individual in population
a {real} – parameter for control on $r_1$ value
rmin {integer} – miniimum value for $r_3$ value
rmax {integer} – maximum value for $r_3$ value

```python
class NiaPy.algorithms.basic.GlowwormSwarmOptimization(**kwargs)
    Bases: NiaPy.algorithms.algorithm.Algorithm
```

Implementation of glowwarm swarm optimization.

**Algorithm:** Glowwarm Swarm Optimization Algorithm

**Date:** 2018

**Authors:** Klemen Berkovič

**License:** MIT

**Reference URL:** https://www.springer.com/gp/book/9783319515946


```python
calcLuciferin(L, GS_f)
getBest(GS, GS_f, xb, xb_f)
getNeighbors(i, r, GS, L)
moveSelect(pb, i)
probabilityes(i, N, L)
randMove(i)
rangleUpdate(R, N, rs)
runTask(task)
```

**Start the optimization.**

**Arguments:**

task {Task} – Task with bounds and objective function for optimization

**Return:**
	solution {array} – point of best solution

fitness {real} – fitness value of best solution

```python
setParameters(n=25, l0=5, nt=5, rho=0.4, gamma=0.6, beta=0.08, s=0.03, **ukwargs)
```

Set the arguments of an algorithm.

**Arguments:**

n {integer} – number of glowworms in population
l0 {real} – initial luciferin quantity for each glowworm
nt {real} –
r {real} – maximum sensing range
rho {real} – luciferin decay constant
gamma {real} – luciferin enhancement constant

---

Chapter 8. API
beta {real} –
s {real} –

class NiaPy.algorithms.basic.GlowwormSwarmOptimizationV1(**kwargs)
    Bases: NiaPy.algorithms.basic.gso.GlowwormSwarmOptimization

    Implementation of glowworm swarm optimization.

    Algorithm: Glowworm Swarm Optimization Algorithm
    Date: 2018
    Authors: Klemen Berkovič
    License: MIT

    Reference URL: https://www.springer.com/gp/book/9783319515946


    calcLuciferin(L, GS_f)
    rangeUpdate(R, N, rs)

    setParameters(**kwargs)
        Set the arguments of an algorithm.

        Arguments:
            n {integer} – number of glowworms in population
            l0 {real} – initial luciferin quantity for each glowworm
            nt {real} –
            rs {real} – maximum sensing range
            rho {real} – luciferin decay constant
            gamma {real} – luciferin enhancement constant
            beta {real} –
            s {real} –

class NiaPy.algorithms.basic.GlowwormSwarmOptimizationV2(**kwargs)
    Bases: NiaPy.algorithms.basic.gso.GlowwormSwarmOptimization

    Implementation of glowworm swarm optimization.

    Algorithm: Glowworm Swarm Optimization Algorithm
    Date: 2018
    Authors: Klemen Berkovič
    License: MIT

    Reference URL: https://www.springer.com/gp/book/9783319515946


    rangeUpdate(P, N, rs)
**setParameters(**kwargs\)**

Set the arguments of an algorithm.

**Arguments:**

n {integer} – number of glowworms in population
l0 {real} – initial luciferin quantity for each glowworm
nt {real} –
rs {real} – maximum sensing range
rho {real} – luciferin decay constant
gamma {real} – luciferin enhancement constant
beta {real} –
s {real} –

**class** NiaPy.algorithms.basic.GlowwormSwarmOptimizationV3(**kwargs\)**

```
Bases: NiaPy.algorithms.basic.gso.GlowwormSwarmOptimization
```

Implementation of glowwarm swarm optimization.

**Algorithm:** Glowwarm Swarm Optimization Algorithm

**Date:** 2018

**Authors:** Klemen Berkovič

**License:** MIT

**Reference URL:** https://www.springer.com/gp/book/9783319515946


**rangeUpdate** (R, N, rs)

**setParameters(**kwargs\)**

Set the arguments of an algorithm.

**Arguments:**

n {integer} – number of glowworms in population
l0 {real} – initial luciferin quantity for each glowworm
nt {real} –
rs {real} – maximum sensing range
rho {real} – luciferin decay constant
gamma {real} – luciferin enhancement constant
beta {real} –
s {real} –

**class** NiaPy.algorithms.basic.HarmonySearch(**kwargs\)**

```
Bases: NiaPy.algorithms.algorithm.Algorithm
```

Implementation of harmony search algorithm.

**Algorithm:** Harmony Search Algorithm

**Date:** 2018
Authors: Klemen Berkovič
License: MIT
Reference URL: https://link.springer.com/chapter/10.1007/978-3-642-00185-7_1

**adjustment** *(x, task)*

**bw** *(task)*

**improvis** *(HM, task)*

**runTask** *(task)*

Start the optimization.

**Arguments:**

task {Task} – Task with bounds and objective function for optimization

**Return:**

solution {array} – point of best solution

fitness {real} – fitness value of best solution

**setParameters** *(HMS=30, r_accept=0.7, r_pa=0.35, b_range=1.42, **kwargs)*

Set the arguments of the algorithm.

**Arguments:**

HMS {integer} – Number of harmonys in the memory

r_accept {real} –

r_pa {real} –

b_range {real} –

**class** NiaPy.algorithms.basic.HarmonySearchV1(**kwargs)

Bases: NiaPy.algorithms.basic.hs.HarmonySearch

Implementation of harmony search algorithm.

**Algorithm:** Harmony Search Algorithm

**Date:** 2018

**Authors:** Klemen Berkovič

**License:** MIT

Reference URL: https://link.springer.com/chapter/10.1007/978-3-642-00185-7_1


**bw** *(task)*

**setParameters** *(bw_min=1, bw_max=2, **kwargs)*

Set the parameters of the algorithm.

**Arguments:**

bw_min {real} – Minimal bandwidth

bw_max {real} – Maximal bandwidth
class NiaPy.algorithms.basic.KrillHerdV1(**kwargs)
Bases: NiaPy.algorithms.basic.kh.KrillHerd

Implementation of krill herd algorithm.

Algorithm: Krill Herd Algorithm

Date: 2018

Authors: Klemen Berkovič

License: MIT


crossover (x, xo, Cr)

mutate (x, x_b, Mu)

class NiaPy.algorithms.basic.KrillHerdV2(**kwargs)
Bases: NiaPy.algorithms.basic.kh.KrillHerd

Implementation of krill herd algorithm.

Algorithm: Krill Herd Algorithm

Date: 2018

Authors: Klemen Berkovič

License: MIT


mutate (x, x_b, Mu)

class NiaPy.algorithms.basic.KrillHerdV3(**kwargs)
Bases: NiaPy.algorithms.basic.kh.KrillHerd

Implementation of krill herd algorithm.

Algorithm: Krill Herd Algorithm

Date: 2018

Authors: Klemen Berkovič

License: MIT


crossover (x, xo, Cr)

class NiaPy.algorithms.basic.KrillHerdV4(**kwargs)
Bases: NiaPy.algorithms.basic.kh.KrillHerd
Implementation of krill herd algorithm.

**Algorithm:** Krill Herd Algorithm

**Date:** 2018

**Authors:** Klemen Berkovič

**License:** MIT


**setParameters** (**kwargs**)

Set the arguments of an algorithm.

**Arguments:**

- **NP** (integer) – Number of krill herds in population
- **N_max** (real) – maximum induced speed
- **V_f** (real) – foraging speed
- **D_max** (real) – maximum diffusion speed
- **C_t** (real) – constant $\in [0, 2]$ $\sin(2\pi)\$
- **W_n** (real) or {array} – inertia weights of the motion induced from neighbors $\in [0, 1]$ $\sin(2\pi)$ $\$
- **W_f** (real) or {array} – inertia weights of the motion induced from foraging $\in [0, 1]$ $\sin(2\pi)$ $\$
- **d_s** (real) – maximum euclidean distance for neighbors
- **nn** (integer) – maximum neighbors for neighbors effect
- **Cr** (real) – Crossover rate
- **Mu** (real) – Mutation rate
- **epsilon** (real) – Small numbers for division

**class** `NiaPy.algorithms.basic.KrillHerdV11(**kwargs**)`

**Bases:** `NiaPy.algorithms.basic.kh.KrillHerd`

Implementation of krill herd algorithm.

**Algorithm:** Krill Herd Algorithm

**Date:** 2018

**Authors:** Klemen Berkovič

**License:** MIT

**Reference URL:**

**Reference paper:**

**Cr** (*KH_f*, *KH_b_f*, *KH_w_f*)

**ElitistSelection** (*KH*, *KH_f*, *KHo*, *KHo_f*)

**Foraging** (*KH*, *KH_f*, *KHo*, *KHo_f*, *W_f*, *F*, *KH_wf*, *KH_bf*, *x_food*, *x_food_f*, *task*)

**Neighbors** (*i*, *KH*, *KH_f*, *iw*, *ib*, *N*, *W_n*, *task*)

---

8.2. NiaPy.algorithms 45
runTask(task)

Start the optimization.

Arguments:

task {Task} – Task with bounds and objective function for optimization

Return:

solution {array} – point of best solution

fitness {real} – fitness value of best solution

class NiaPy.algorithms.basic.FireworksAlgorithm(**kwargs)

Bases: NiaPy.algorithms.algorithm.Algorithm

Implementation of fireworks algorithm.

Algorithm: Fireworks Algorithm

Date: 2018

Authors: Klemen Berkovič

License: MIT

Reference URL: https://www.springer.com/gp/book/9783662463529


ExplodeSpark(x, A, task)

ExplosionAmplitude(x_f, xb_f, A, As)

GaussianSpark(x, task)

Mapping(x, task)

NextGeneration(FW, FW_f, FWn, task)

R(x, FW)

SparsksNo(x_f, xw_f, Ss)

initAmplitude(task)

p(r, Rs)

runTask(task)

Start the optimization.

Arguments:

task {Task} – Task with bounds and objective function for optimization

Return:

solution {array} – point of best solution

fitness {real} – fitness value of best solution

setParameters(N=40, m=40, a=1, b=2, A=40, epsilon=1e-31, **ukwargs)

Set the arguments of an algorithm.

Arguments:

N {integer} – number of Fireworks

m {integer} – number of sparks
a {integer} – limitation of sparks
b {integer} – limitation of sparks
A {real} –
epsilon {real} – Small number for non 0 devision

class NiaPy.algorithms.basic.EnhancedFireworksAlgorithm(**kwargs)
Bases: NiaPy.algorithms.basic.fwa.FireworksAlgorithm
Implementation of enganced fireworks algorithm.
Algorithm: Enhanced Fireworks Algorithm
Date: 2018
Authors: Klemen Berkovič
License: MIT
Reference URL: https://ieeexplore.ieee.org/document/6557813/

ExplosionAmplitude (x_f, xb_f, A_min, Ah, As, task)
GaussianSpark (x, xb, task)
NextGeneration (FW, FW_f, FWn, task)
initRanges (task)
runTask (task)
Start the optimization.
Arguments:
task {Task} – Task with bounds and objective function for optimization
Return:
solution {array} – point of best solution
fitness {real} – fitness value of best solution

setParameters (Ainit=20, Afinal=5, **ukwargs)
Set the arguments of an algorithm.
Arguments:
N {integer} – number of Fireworks
m {integer} – number of sparks
a {integer} – limitation of sparks
b {integer} – limitation of sparks
A {real} –
epsilon {real} – Small number for non 0 devision

uAmin (Ainit, Afinal, task)

class NiaPy.algorithms.basic.DynamicFireworksAlgorithm(**kwargs)
Bases: NiaPy.algorithms.basic.fwa.DynamicFireworksAlgorithmGauss
Implementation of dynamic fireworks algorithm.
Algorithm: Dynamic Fireworks Algorithm
Date: 2018
Authors: Klemen Berković
License: MIT

Reference URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6900485&isnumber=6900223


runTask (task)
Start the optimization.

Arguments:

- task {Task} – Task with bounds and objective function for optimization

Return:

- solution {array} – point of best solution
- fitness {real} – fitness value of best solution

class NiaPy.algorithms.basic.DynamicFireworksAlgorithmGauss(**kwargs)

Bases: NiaPy.algorithms.basic.fwa.EnhancedFireworksAlgorithm

Implementation of dynamic fireworks algorithm.

Algorithm: Dynamic Fireworks Algorithm
Date: 2018
Authors: Klemen Berković
License: MIT

Reference URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6900485&isnumber=6900223


ExplosionAmplitude (x_f, xb_f, A, As)

Mapping (x, task)

NextGeneration (FW, FW_f, FWn, task)
Elitism Random Selection.

initAmplitude (task)

repair (x, d, epsilon)

runTask (task)
Start the optimization.

Arguments:

- task {Task} – Task with bounds and objective function for optimization

Return:

- solution {array} – point of best solution
- fitness {real} – fitness value of best solution
**setParameters** *(A_cf=20, C_a=1.2, C_r=0.9, epsilon=1e-08, **ukwargs)*

Set the arguments of an algorithm.

**Arguments:**

N {integer} – number of Fireworks
m {integer} – number of sparks
a {integer} – limitation of sparks
b {integer} – limitation of sparks
A {real} –
epsilon {real} – Small number for non 0 devision

**uCF** *(xnb, xcb, xcb_f, xb, xb_f, Acf, task)*

**class** `NiaPy.algorithms.basic.GravitationalSearchAlgorithm(**kwargs)`

Bases: `NiaPy.algorithms.algorithm.Algorithm`

Implementation of gravitational search algorithm.

**Algorithm:** Gravitational Search Algorithm

**Date:** 2018

**Author:** Klemen Berkoivč

**License:** MIT

**Reference URL:** [https://doi.org/10.1016/j.ins.2009.03.004](https://doi.org/10.1016/j.ins.2009.03.004)


**G** *(t)*

**d** *(x, y, ln=2)*

**runTask**(task)

Start the optimization.

**Arguments:**

task {Task} – Task with bounds and objective function for optimization

**Return:**

solution {array} – point of best solution
fitness {real} – fitness value of best solution

**setParameters** *(NP=40, G_0=2.467, epsilon=1e-17, **ukwargs)*

Set the algorithm parameters.

**Arguments:**

NP {integer} – number of planets in population
G_0 {real} – starting gravitational constant
8.2.2 `NiaPy.algorithms.modified`

Implementation of modified nature-inspired algorithms.

```python
class NiaPy.algorithms.modified.HybridBatAlgorithm(**kwargs):
    Bases: NiaPy.algorithms.basic.ba.BatAlgorithm

    Implementation of Hybrid bat algorithm.

    Algorithm: Hybrid bat algorithm

    Date: 2018

    Author: Grega Vrbancic and Klemen Berkovic

    License: MIT


def runTask(task):
    Run algorithm with initialized parameters.

    Return:
    {decimal} – coordinates of minimal found objective function
    {decimal} – minimal value found of objective function

setParameters(**kwargs)

    Set the parameters of the algorithm.

    Arguments:
    NP {integer} – population size
    A {decimal} – loudness
    r {decimal} – pulse rate
    Qmin {decimal} – minimum frequency
    Qmax {decimal} – maximum frequency
```

```python
class NiaPy.algorithms.modified.SelfAdaptiveDifferentialEvolutionAlgorithm(**kwargs):
    Bases: NiaPy.algorithms.basic.de.DifferentialEvolutionAlgorithm

    Implementation of Self-adaptive differential evolution algorithm.

    Algorithm: Self-adaptive differential evolution algorithm

    Date: 2018

    Author: Uros Mlakar and Klemen Berkovic

    License: MIT


def AdaptiveGen(x):
    runTask(task)
    Run.
**setParameters** \((F_l=0.0, F_u=2.0, Tao1=0.4, Tao2=0.6, **ukwargs\))

Set the parameters of an algorithm.

**Arguments:**
- \(F_l\) (decimal) – scaling factor lower limit
- \(F_u\) (decimal) – scaling factor upper limit
- \(Tao1\) (decimal) – change rate for \(F\) parameter update
- \(Tao2\) (decimal) – change rate for \(CR\) parameter update

```python
class NiaPy.algorithms.modified.DynNPSelfAdaptiveDifferentialEvolutionAlgorithm(**kwargs)
Bases: NiaPy.algorithms.modified.jde.SelfAdaptiveDifferentialEvolutionAlgorithm


**Algorithm:** Dynamic population size self-adaptive differential evolution algorithm

**Date:** 2018

**Author:** Jan Popiˇc

**License:** MIT

**Reference URL:** https://link.springer.com/article/10.1007/s10489-007-0091-x


**AdaptiveGen** \((x)\)

**runTask** \((task)\)

Run.

**setParameters** \((rp=0, pmax=4, **ukwargs\))

Set the parameters of an algorithm.

**Arguments:**
- \(rp\) (integer) – small non-negative number which is added to value of genp (if it’s not divisible)
- \(pmax\) (integer) – number of population reductions

### 8.2.3 NiaPy.algorithms.other

Implementation of basic nature-inspired algorithms.

```python
class NiaPy.algorithms.other.NelderMeadMethod(**kwargs)
Bases: NiaPy.algorithms.algorithm.Algorithm

Implementation of Nelder Mead method or downhill simplex method or amoeba method.

**Algorithm:** Nelder Mead Method

**Date:** 2018

**Authors:** Klemen Berkoviˇc

**License:** MIT

**Reference URL:** https://en.wikipedia.org/wiki/Nelder%E2%80%93Mead_method

**init** \((task)\)

**method** \((X, X_f, task)\)
runTask(task)
Start the optimization.

Arguments:
- task {Task} – Task with bounds and objective function for optimization

Return:
- solution {array} – point of best solution
- fitness {real} – fitness value of best solution

setParameters(alpha=1.0, gamma=2.0, rho=-0.5, sigma=0.5, **ukwargs)
Set the arguments of an algorithm.

Arguments:
- alpha {real} – Reflection coefficient parameter
- gamma {real} – Expansion coefficient parameter
- rho {real} – Contraction coefficient parameter
- sigma {real} – Shrink coefficient parameter

class NiaPy.algorithms.other.HillClimbAlgorithm(**kwargs)
Bases: NiaPy.algorithms.algorithm.Algorithm
Implementation of iterative hill climbing algorithm.

Algorithm: Hill Climbing Algorithm
Date: 2018
Authors: Jan Popič
License: MIT
Reference URL:
Reference paper:
Initialize Iterative Hillclimb algorithm class.
See: Algorithm.__init__(self, **kwargs)
runTask(task)
Start the optimization.

Arguments:
- task {Task} – Task with bounds and objective function for optimization

Return:
- solution {array} – point of best solution
- fitness {real} – fitness value of best solution

setParameters(delta=0.5, Neighborhood=<function Neighborhood>, **ukwargs)
Set the algorithm parameters/arguments.

See: HillClimbAlgorithm.__setparams(self, delta=0.5, Neighborhood=Neighborhood, **ukwargs)

class NiaPy.algorithms.other.SimulatedAnnealing(**kwargs)
Bases: NiaPy.algorithms.algorithm.Algorithm
Implementation of Simulated Annealing Algorithm.
**Algorithm:** Simulated Annealing Algorithm  
**Date:** 2018  
**Authors:** Jan Popič  
**License:** MIT  
**Reference URL:**  
**Reference paper:** Init Simulated Annealing Algorithm.

```python
runTask(task)
    Start the optimization.

Arguments:
    task {Task} – Task with bounds and objective function for optimization

Return:
    solution {array} – point of best solution
    fitness {real} – fitness value of best solution
```

```python
setParameters(delta=0.5, T=20, deltaT=0.8, coolingMethod=<function coolDelta>, **ukwargs)
    Set the algorithm parameters/arguments.
```

```python
class NiaPy.algorithms.other.MultipleTrajectorySearch(**kwargs)
    Implementation of Multiple trajectory search.
```

**Algorithm:** Multiple trajectory search  
**Date:** 2018  
**Authors:** Klemen Berkovic  
**License:** MIT  
**Reference URL:** https://ieeexplore.ieee.org/document/4631210/

**Reference paper:** Lin-Yu Tseng and Chun Chen, “Multiple trajectory search for Large Scale Global Optimization,” 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), Hong Kong, 2008, pp. 3052-3059. doi: 10.1109/CEC.2008.4631210

**BONUS1 = 10**

**Algorithm:** Multiple trajectory search  
**Date:** 2018  
**Authors:** Klemen Berkovic  
**License:** MIT  
**Reference URL:** https://ieeexplore.ieee.org/document/4631210/

**Reference paper:**

GradingRun \((x, x_f, xb, xb_f, improve, SR, task)\)

LsRun \((k, x, x_f, xb, xb_f, improve, SR, g, task)\)

getBest \((X, X_f)\)

runTask \((task)\)

Start the optimization.

Arguments:

\- task \{Task\} – Task with bounds and objective function for optimization

Return:

\- solution \{array\} – point of best solution
\- fitness \{real\} – fitness value of best solution

setParameters \((NP=40, NoLsTests=5, NoLs=5, NoLsBest=5, NoEnabled=17, **ukwargs)\)

Set the arguments of the algorithm.

Arguments:

\- NP, M \{integer\} – population size
\- NoLsTests \{integer\} – number of test runs on local search algorithms
\- NoLs \{integer\} – number of local search algorithm runs
\- NoLsBest \{integer\} – number of local search algorithm runs on best solution
\- NoEnabled \{integer\} – number of best solution for testing

class NiaPy.algorithms.other.MultipleTrajectorySearchV1(**kwargs)

Bases: NiaPy.algorithms.other.mts.MultipleTrajectorySearch

Implementation of Multiple trajectory search.

Algorithm: Multiple trajectory search

Date: 2018

Authors: Klemen Berkovic

License: MIT

Reference URL: https://ieeexplore.ieee.org/document/4983179/


runTask \((task)\)

Start the optimization.

Arguments:

\- task \{Task\} – Task with bounds and objective function for optimization

Return:

\- solution \{array\} – point of best solution
\- fitness \{real\} – fitness value of best solution
NiaPy.algorithms.other.MTS_LS1 \( (X_k, X_k, X_b, X_b, \text{improve}, \text{SR}, \text{task}, \text{BONUS1}=10, \text{BONUS2}=1, \text{rnd}=<\text{module 'numpy.random' from '/home/docs/checkouts/readthedocs.org/user_builds/niapy/envs/stable/lib/python3.6/site-packages/numpy/random/__init__.py'>) \)

NiaPy.algorithms.other.MTS_LS2 \( (X_k, X_k, X_b, X_b, \text{improve}, \text{SR}, \text{task}, \text{BONUS1}=10, \text{BONUS2}=1, \text{rnd}=<\text{module 'numpy.random' from '/home/docs/checkouts/readthedocs.org/user_builds/niapy/envs/stable/lib/python3.6/site-packages/numpy/random/__init__.py'>) \)

NiaPy.algorithms.other.MTS_LS3 \( (X_k, X_k, X_b, X_b, \text{improve}, \text{SR}, \text{task}, \text{BONUS1}=10, \text{BONUS2}=1, \text{rnd}=<\text{module 'numpy.random' from '/home/docs/checkouts/readthedocs.org/user_builds/niapy/envs/stable/lib/python3.6/site-packages/numpy/random/__init__.py'>) \)

NiaPy.algorithms.other.MTS_LS1v1 \( (X_k, X_k, X_b, X_b, \text{improve}, \text{SR}, \text{task}, \text{BONUS1}=10, \text{BONUS2}=1, \text{rnd}=<\text{module 'numpy.random' from '/home/docs/checkouts/readthedocs.org/user_builds/niapy/envs/stable/lib/python3.6/site-packages/numpy/random/__init__.py'>) \)

NiaPy.algorithms.other.MTS_LS3v1 \( (X_k, X_k, X_b, X_b, \text{improve}, \text{SR}, \text{task}, \text{phi}=3, \text{BONUS1}=10, \text{BONUS2}=1, \text{rnd}=<\text{module 'numpy.random' from '/home/docs/checkouts/readthedocs.org/user_builds/niapy/envs/stable/lib/python3.6/site-packages/numpy/random/__init__.py'>) \)

class NiaPy.algorithms.other.AnarchicSocietyOptimization (**kwargs)
Bases: NiaPy.algorithms.algorithm.Algorithm

Implementation of Anarchic Society Optimization algorithm.

Algorithm: Particle Swarm Optimization algorithm

Date: 2018

Authors: Klemen Berkovič

License: MIT


EI \( (x_f, xnb_f, \text{gamma}) \)

Get external irregularity index.

FI \( (x_f, xpb_f, xb_f, \text{alpha}) \)

Get fickleness index.

II \( (x_f, xpb_f, \text{theta}) \)

Get internal irregularity index.

getBestNeighbors \( (i, X, x_f, \text{rs}) \)

init (task)

runTask (task)

Start the optimization.

Arguments:

 task {Task} – Task with bounds and objective function for optimization

Return:

 solution {array} – point of best solution

 fitness {real} – fitness value of best solution
```python
def setParameters(NP=43, alpha=[1, 0.83], gamma=[1.17, 0.56], theta=[0.932, 0.832], d=<function euclidean>, dn=<function euclidean>, nl=1, F=1.2, CR=0.25, Combination=<function Elitism>, **ukwargs):
    Set the parameters for the algorithm.
    
    **Arguments:**
    
    NP {integer} – population size
    alpha {array} – factor for fickleness index function $in [0, 1]$
    gamma {array} – factor for external irregularity index function $in [0, infty)$
    theta {array} – factor for internal irregularity index function $in [0, infty)$
    d {function} – function that takes two arguments that are function values and calcs the distance between them
    dn {function} – function that takes two arguments that are points in function landscape and calcs the distance between them
    nl {real} – normalized range for neighborhood search $in (0, 1]$
    F {real} – mutation parameter
    CR {real} – crossover parameter $in [0, 1]$
    Combination {function} – Function that combines movement strategies
```

8.3 NiaPy.benchmarks

Module with implementations of benchmark functions.

class NiaPy.benchmarks.Rastrigin(Lower=-5.12, Upper=5.12)
    Implementation of Rastrigin benchmark function.
    Date: 2018
    Authors: Lucija Brezočnik and Iztok Fister Jr.
    License: MIT

Function: Rastrigin function

$$f(x) = 10D + \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i)\right)$$

**Input domain:** The function can be defined on any input domain but it is usually evaluated on the hypercube $x_i[-5.12, 5.12]$, for all $i = 1, 2, ..., D$.

**Global minimum:** $f(x^*) = 0$, at $x^* = (0, ..., 0)$

LaTeX formats:

**Inline:** $f(\mathbf{x}) = 10D + \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i)\right)$

**Equation:**

```
\begin{equation}
    f(\mathbf{x}) = 10D + \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i)\right)
\end{equation}
```

**Domain:** $-5.12 \leq x_i \leq 5.12$
Reference: https://www.sfu.ca/~ssurjano/rastr.html

classmethod function()

class NiaPy.benchmarks.Rosenbrock(Lower=-30.0, Upper=30.0)

Bases: object

Implementation of Rosenbrock benchmark function.

Date: 2018

Authors: Iztok Fister Jr. and Lucija Brezočnik

License: MIT

Function: Rosenbrock function

\[ f(x) = \sum_{i=1}^{D-1} \left( 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right) \]

Input domain: The function can be defined on any input domain but it is usually evaluated on the hypercube \( x_i \in [-30, 30] \), for all \( i = 1, 2, ..., D \).

Global minimum: \( f(x^*) = 0 \), at \( x^* = (1, ..., 1) \)

LaTeX formats:

Inline: \( f(\mathbf{x}) = \sum_{i=1}^{D-1} \left( 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right) \)

Equation: \begin{equation} f(\mathbf{x}) = \sum_{i=1}^{D-1} \left( 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right) \end{equation}

Domain: \(-30 \leq x_i \leq 30\)


classmethod function()

class NiaPy.benchmarks.Griewank(Lower=-100.0, Upper=100.0)

Bases: object

Implementation of Griewank function.

Date: 2018

Authors: Iztok Fister Jr. and Lucija Brezočnik

License: MIT

Function: Griewank function

\[ f(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \]

Input domain: The function can be defined on any input domain but it is usually evaluated on the hypercube \( x_i \in [-100, 100] \), for all \( i = 1, 2, ..., D \).

Global minimum: \( f(x^*) = 0 \), at \( x^* = (0, ..., 0) \)

LaTeX formats:

Inline: \( f(\mathbf{x}) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \)

Equation: \begin{equation} f(\mathbf{x}) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \end{equation}

Domain: \(-100 \leq x_i \leq 100\)

```python
class NiaPy.benchmarks.ExpandedGriewankPlusRosenbrock(Lower=-100.0, Upper=100.0):
    Bases: object

    Implementation of Expanded Griewank’s plus Rosenbrock function.
    Date: 2018
    Author: Klemen Berkovič
    License: MIT

    Function: Expanded Griewank’s plus Rosenbrock function
    
    $f(x) = h(g(x_D, x_1)) + \sum_{i=2}^{D} h(g(x_{i-1}, x_i))$
    $g(x, y) = 100(x^2 - y)^2 + (x - 1)^2$
    $h(z) = \frac{z^2}{4000} - \cos\left(\frac{z}{\sqrt{1}}\right) + 1$

    Input domain: The function can be defined on any input domain but it is usually evaluated on the hypercube $x_i[-100, 100]$, for all $i = 1, 2, ..., D$.

    Global minimum: $f(x^*) = 0$, at $x^* = (420.968746, ..., 420.968746)$

    LaTeX formats:
    Inline: $f(\mathbf{x}) = h(g(x_D, x_1)) + \sum_{i=2}^{D} h(g(x_{i-1}, x_i))$
    $g(x, y) = 100(x^2 - y)^2 + (x - 1)^2$
    $h(z) = \frac{z^2}{4000} - \cos\left(\frac{z}{\sqrt{1}}\right) + 1$

    Equation: $f(\mathbf{x}) = h(g(x_D, x_1)) + \sum_{i=2}^{D} h(g(x_{i-1}, x_i))$
    $g(x, y) = 100(x^2 - y)^2 + (x - 1)^2$
    $h(z) = \frac{z^2}{4000} - \cos\left(\frac{z}{\sqrt{1}}\right) + 1$

    Domain: $-100 \leq x_i \leq 100$

    Reference: http://www5.zzu.edu.cn/__local/A/69/BC/D3B5DFE94CD2574B38AD7CD1D12_C802DAFE_BC0C0.pdf
```

```python
class NiaPy.benchmarks.Sphere(Lower=-5.12, Upper=5.12):
    Bases: object

    Implementation of Sphere functions.
    Date: 2018
    Authors: Iztok Fister Jr.
    License: MIT

    Function: Sphere function
    
    $f(x) = \sum_{i=1}^{D} x_i^2$

    Input domain: The function can be defined on any input domain but it is usually evaluated on the hypercube $x_i[0, 10]$, for all $i = 1, 2, ..., D$.

    Global minimum: $f(x^*) = 0$, at $x^* = (0, ..., 0)$
```
LaTeX formats:

**Inline:** $f(\mathbf{x}) = \sum_{i=1}^D x_i^2$

**Equation:**

\[
\begin{equation}
    f(\mathbf{x}) = \sum_{i=1}^D x_i^2
\end{equation}
\]

**Domain:** $0 \leq x_i \leq 10$


```
classmethod function()

class NiaPy.benchmarks.Ackley(Lower=-32.768, Upper=32.768)

    Bases: object

    Implementation of Ackley function.

    Date: 2018
    Author: Lucija Brezočnik
    License: MIT

    Function: Ackley function

    \[
    f(\mathbf{x}) = -a \exp \left( -b \sqrt{\frac{1}{D} \sum_{i=1}^D x_i^2} \right) - \exp \left( \frac{1}{D} \sum_{i=1}^D \cos(c \cdot x_i) \right) + a + \exp(1)
    \]

    **Input domain:** The function can be defined on any input domain but it is usually evaluated on the hypercube $x_i[-32.768, 32.768]$, for all $i = 1, 2, ..., D$.

    **Global minimum:** $f(\mathbf{x}^*) = 0$, at $\mathbf{x}^* = (0, ..., 0)$

LaTeX formats:

**Inline:**

\[
    f(\mathbf{x}) = -a \exp \left( -b \sqrt{\frac{1}{D} \sum_{i=1}^D x_i^2} \right) - \exp \left( \frac{1}{D} \sum_{i=1}^D \cos(c \cdot x_i) \right) + a + \exp(1)
\]

**Domain:** $-32.768 \leq x_i \leq 32.768$

Reference: https://www.sfu.ca/~ssurjano/ackley.html

```
classmethod function()

Return benchmark evaluation function.

```
class NiaPy.benchmarks.Schwefel(Lower=-500.0, Upper=500.0)

    Bases: object

    Implementation of Schewel function.

    Date: 2018
    Author: Lucija Brezočnik
    License: MIT

    Function: Schwefel function

```

8.3. NiaPy.benchmarks
\[ f(x) = 418.9829d - \sum_{i=1}^{D} x_i \sin(\sqrt{|x_i|}) \]

**Input domain:** The function can be defined on any input domain but it is usually evaluated on the hypercube \( x_i [-500, 500] \), for all \( i = 1, 2, ..., D \).

**Global minimum:** \( f(x^*) = 0 \), at \( x^* = (420.968746, ..., 420.968746) \)

**LaTeX formats:**

**Inline:** 
\[
f(\textbf{x}) = 418.9829d - \sum_{i=1}^{D} x_i \sin(\sqrt{|x_i|})
\]

**Equation:**
\[
f(\textbf{x}) = 418.9829d - \sum_{i=1}^{D} x_i \sin(\sqrt{|x_i|})
\]

**Domain:** \(-500 \leq x_i \leq 500\)

Reference: https://www.sfu.ca/~ssurjano/schwef.html

class method function()

class NiaPy.benchmarks.Schwefel221 (Lower=-100.0, Upper=100.0)

Bases: object

Schwefel 2.21 function implementation.

Date: 2018

Author: Grega Vrbančič

Licence: MIT

Function: **Schwefel 2.21 function**

\[ f(x) = \max_{i=1, \ldots, D} |x_i| \]

**Input domain:** The function can be defined on any input domain but it is usually evaluated on the hypercube \( x_i [-100, 100] \), for all \( i = 1, 2, ..., D \).

**Global minimum:** \( f(x^*) = 0 \), at \( x^* = (0, ..., 0) \)

**LaTeX formats:**

**Inline:**
\[
f(\textbf{x}) = \max_{i=1, \ldots, D} |x_i|
\]

**Equation:**
\[
f(\textbf{x}) = \max_{i=1, \ldots, D} |x_i|
\]

**Domain:** \(-100 \leq x_i \leq 100\)


class method function()

class NiaPy.benchmarks.Schwefel222 (Lower=-100.0, Upper=100.0)

Bases: object

Schwefel 2.22 function implementation.

Date: 2018

Author: Grega Vrbančič

Licence: MIT

Function: **Schwefel 2.22 function**
\( f(x) = \sum_{i=1}^{D} |x_i| + \prod_{i=1}^{D} |x_i| \)

**Input domain:** The function can be defined on any input domain but it is usually evaluated on the hypercube \( x_i[-100, 100] \), for all \( i = 1, 2, ..., D \).

**Global minimum:** \( f(x^*) = 0 \), at \( x^* = (0, ..., 0) \)

LaTeX formats:

- **Inline:** \( f(\textbf{x}) = \sum_{i=1}^{D} |x_i| + \prod_{i=1}^{D} |x_i| \)

- **Equation:**

\[
\begin{equation} f(\textbf{x}) = \sum_{i=1}^{D} |x_i| + \prod_{i=1}^{D} |x_i| \end{equation}
\]

- **Domain:** \(-100 \leq x_i \leq 100\)


```python
class NiaPy.benchmarks.ExpandedScaffer(Lower=-100.0, Upper=100.0)
    Bases: object

    Implementations of High Conditioned Elliptic functions.

    Date: 2018
    Author: Klemen Berkovič
    License: MIT

    Function: **High Conditioned Elliptic Function**

    \[
    f(x) = g(x_D, x_1) + \sum_{i=2}^{D} g(x_{i - 1}, x_i) \\
    g(x, y) = 0.5 + \sin\left(\sqrt{x^2 + y^2}\right)^2 - 0.5 \left( 1 + 0.001 \left( x^2 + y^2 \right) \right)^2
    \]

    **Input domain:** The function can be defined on any input domain but it is usually evaluated on the hypercube \( x_i[-100, 100] \), for all \( i = 1, 2, ..., D \).

    **Global minimum:** \( f(x^*) = 0 \), at \( x^* = (420.968746, ..., 420.968746) \)

    LaTeX formats:

    - **Inline:**

    \[
    f(\textbf{x}) = g(x_D, x_1) + \sum_{i=2}^{D} g(x_{i - 1}, x_i) \\
    g(x, y) = 0.5 + \sin\left(\sqrt{x^2 + y^2}\right)^2 - 0.5 \left( 1 + 0.001 \left( x^2 + y^2 \right) \right)^2
    \]

    - **Equation:**

    \[
    \begin{equation} f(\textbf{x}) = g(x_D, x_1) + \sum_{i=2}^{D} g(x_{i - 1}, x_i) \\
    g(x, y) = 0.5 + \sin\left(\sqrt{x^2 + y^2}\right)^2 - 0.5 \left( 1 + 0.001 \left( x^2 + y^2 \right) \right)^2 \end{equation}
    \]

    - **Domain:** \(-100 \leq x_i \leq 100\)

    Reference: http://www5.zzu.edu.cn/__local/A/69/BC/D3B5DFE94CD2574B38AD7CD1D12_C802DAFE_BC0C0.pdf

```

```python
class NiaPy.benchmarks.ModifiedSchwefel(Lower=-100.0, Upper=100.0)
    Bases: object

    Implementations of Modified Schwefel functions.

    Date: 2018
    ```
Function: **Modified Schwefel Function**

\[ f(x) = 418.9829 \cdot D - \sum_{i=1}^{D} h(x_i) \]

\[ h(x) = g(x + 420.9687462275036) \]

\[ g(z) = \begin{cases} 
  z \sin \left( \left| z \right|^{\frac{1}{2}} \right) & \text{if } |z| \leq 500 \\
  \left( \left| z \right| - 500 \right) \sin \left( \sqrt{\left| \left| z \right| - 500 \right|} \right) - \frac{\left( z - 500 \right)^2}{10000D} & \text{if } z > 500 \\
  \left( \left| z \right| - 500 \right) \sin \left( \sqrt{\left| \left| z \right| - 500 \right|} \right) + \frac{\left( z - 500 \right)^2}{10000D} & \text{if } z < -500 
\end{cases} \]

**Input domain:** The function can be defined on any input domain but it is usually evaluated on the hypercube \(x_i[-100, 100]\), for all \(i = 1, 2, ..., D\).

**Global minimum:** \(f(x^*) = 0\), at \(x^* = (420.968746, ..., 420.968746)\)

**LaTeX formats:**

**Inline:** \(f(\textbf{x}) = 418.9829 \cdot D - \sum_{i=1}^{D} h(x_i)\)

\(h(x) = g(x + 420.9687462275036)\)

\(g(z) = \begin{cases} 
  z \sin \left( \left| z \right|^{\frac{1}{2}} \right) & \text{if } |z| \leq 500 \\
  \left( \left| z \right| - 500 \right) \sin \left( \sqrt{\left| \left| z \right| - 500 \right|} \right) - \frac{\left( z - 500 \right)^2}{10000D} & \text{if } z > 500 \\
  \left( \left| z \right| - 500 \right) \sin \left( \sqrt{\left| \left| z \right| - 500 \right|} \right) + \frac{\left( z - 500 \right)^2}{10000D} & \text{if } z < -500 
\end{cases} \)

**Equation:** \(f(\textbf{x}) = 418.9829 \cdot D - \sum_{i=1}^{D} h(x_i)\)

\(h(x) = g(x + 420.9687462275036)\)

\(g(z) = \begin{cases} 
  z \sin \left( \left| z \right|^{\frac{1}{2}} \right) & \text{if } |z| \leq 500 \\
  \left( \left| z \right| - 500 \right) \sin \left( \sqrt{\left| \left| z \right| - 500 \right|} \right) - \frac{\left( z - 500 \right)^2}{10000D} & \text{if } z > 500 \\
  \left( \left| z \right| - 500 \right) \sin \left( \sqrt{\left| \left| z \right| - 500 \right|} \right) + \frac{\left( z - 500 \right)^2}{10000D} & \text{if } z < -500 
\end{cases} \)

**Domain:** \(-100 \leq x_i \leq 100\)

Reference: [http://www5.zzu.edu.cn/__local/A/69/BC/D3B5DFE94CD2574B38AD7CD1D12_C802DAFE_BC0C0.pdf](http://www5.zzu.edu.cn/__local/A/69/BC/D3B5DFE94CD2574B38AD7CD1D12_C802DAFE_BC0C0.pdf)

**classmethod function()**

```python
class NiaPy.benchmarks.Whitley(Lower=-10.24, Upper=10.24):
    Bases: object
    Implementation of Whitley function.
    Date: 2018
    Authors: Grega Vrbančič and Lucija Brezočnik
    License: MIT
    Function: Whitley function
    \[ f(x) = \sum_{i=1}^{D} \sum_{j=1}^{D} \left( \frac{100(x_i^2 - x_j)^2 + (1-x_j)^2}{4000} \right) - \cos(100(x_i^2 - x_j)^2 + (1-x_j)^2) + 1 \]
    **Input domain:** The function can be defined on any input domain but it is usually evaluated on the hypercube \(x_i[-10.24, 10.24]\), for all \(i = 1, 2, ..., D\).
    **Global minimum:** \(f(x^*) = 0\), at \(x^* = (1, ..., 1)\)

**LaTeX formats:**
\[ f(\mathbf{x}) = \sum_{i=1}^{D} \sum_{j=1}^{D} \left( \frac{(100(x_i^2-x_j)^2 + (1-x_j)^2)^2}{4000} - \cos(100(x_i^2-x_j)^2 + (1-x_j)^2)+1 \right) \]

**Equation:**

\[
\begin{equation}
    f(\mathbf{x}) = \sum_{i=1}^{D} \sum_{j=1}^{D} \left( \frac{(100(x_i^2-x_j)^2 + (1-x_j)^2)^2}{4000} - \cos(100(x_i^2-x_j)^2 + (1-x_j)^2)+1 \right)
\end{equation}
\]

**Domain:** $-10.24 \leq x_i \leq 10.24$


```python
classmethod function()
class NiaPy.benchmarks.Alpine1(Lower=-10.0, Upper=10.0)
    Bases: object
    Implementation of Alpine1 function.
    Date: 2018
    Author: Lucija Brezočnik
    License: MIT
    Function: Alpine1 function
    \[
    f(\mathbf{x}) = \sum_{i=1}^{D} |x_i \sin(x_i) + 0.1x_i|
    \]
    **Input domain:** The function can be defined on any input domain but it is usually evaluated on the hypercube $x_i[-10, 10]$, for all $i = 1, 2, ..., D$.
    **Global minimum:** $f(\mathbf{x}^*) = 0$, at $\mathbf{x}^* = (0, ..., 0)$

**LaTeX formats:**

\[ f(\mathbf{x}) = \sum_{i=1}^{D} |x_i \sin(x_i) + 0.1x_i| \]

**Inline:** \( f(\mathbf{x}) = \sum_{i=1}^{D} |x_i \sin(x_i) + 0.1x_i| \)

**Equation:**

\[
\begin{equation}
    f(\mathbf{x}) = \sum_{i=1}^{D} |x_i \sin(x_i) + 0.1x_i|
\end{equation}
\]

**Domain:** $-10 \leq x_i \leq 10$


```python
classmethod function()
class NiaPy.benchmarks.Alpine2(Lower=0.0, Upper=10.0)
    Bases: object
    Implementation of Alpine2 function.
    Date: 2018
    Author: Lucija Brezočnik
    License: MIT
    Function: Alpine2 function
    \[
    f(\mathbf{x}) = \prod_{i=1}^{D} \sqrt{x_i} \sin(x_i)
    \]
    **Input domain:** The function can be defined on any input domain but it is usually evaluated on the hypercube $x_i[0, 10]$, for all $i = 1, 2, ..., D$. 

8.3. NiaPy.benchmarks
Global minimum: $f(x^*) = 2.808^D$, at $x^* = (7.917, ..., 7.917)$

LaTeX formats:

Inline: $f(\mathbf{x}) = \prod_{i=1}^{D} \sqrt{x_i} \sin(x_i)$

Equation: \begin{equation} f(\mathbf{x}) = \prod_{i=1}^{D} \sqrt{x_i} \sin(x_i) \end{equation}

Domain: $0 \leq x_i \leq 10$


\textbf{classmethod function()}

class NiaPy.benchmarks.HappyCat (Lower=-100.0, Upper=100.0)

Bases: object

Implementation of Happy cat function.

Date: 2018

Author: Lucija Brezočnik

License: MIT

Function: Happy cat function

\[ f(x) = \left| \sum_{i = 1}^{D} x_i^2 - D \right|^{1/4} + (0.5 \sum_{i = 1}^{D} x_i^2 + \sum_{i = 1}^{D} x_i) / D + 0.5 \]

Input domain: The function can be defined on any input domain but it is usually evaluated on the hypercube $x_i [-100, 100]$, for all $i = 1, 2, ..., D$.

Global minimum: $f(x^*) = 0$, at $x^* = (-1, ..., -1)$

LaTeX formats:

Inline: $f(\mathbf{x}) = \{\left|\sum_{i = 1}^{D} x_i^2 - D \right|^{1/4} + (0.5 \sum_{i = 1}^{D} x_i^2 + \sum_{i = 1}^{D} x_i) / D + 0.5\}$

Equation: \begin{equation} f(\mathbf{x}) = \{\left|\sum_{i = 1}^{D} x_i^2 - D \right|^{1/4} + (0.5 \sum_{i = 1}^{D} x_i^2 + \sum_{i = 1}^{D} x_i) / D + 0.5\} \end{equation}

Domain: $-100 \leq x_i \leq 100$


\textbf{classmethod function()}

class NiaPy.benchmarks.Ridge (Lower=-64.0, Upper=64.0)

Bases: object

Implementation of Ridge function.

Date: 2018

Author: Lucija Brezočnik

License: MIT

Function: Ridge function
\[ f(x) = \sum_{i=1}^{D} (\sum_{j=1}^{i} x_j)^2 \]

**Input domain:** The function can be defined on any input domain but it is usually evaluated on the hypercube \( x_i \in [-64, 64] \), for all \( i = 1, 2, ..., D \).

**Global minimum:** \( f(x^*) = 0 \), at \( x^* = (0, ..., 0) \)

LaTeX formats:

- **Inline:** \( f(\mathbf{x}) = \sum_{i=1}^{D} (\sum_{j=1}^{i} x_j)^2 \)
- **Equation:**

\[
\begin{equation}
 f(\mathbf{x}) = \left(\sum_{i=1}^{D} x_i^2\right)^2
\end{equation}
\]

**Domain:** \(-64 \leq x_i \leq 64\)


```python
class NiaPy.benchmarks.ChungReynolds(Lower=-100.0, Upper=100.0):
    Bases: object
    Implementation of Chung Reynolds functions.
    Date: 2018
    Authors: Lucija Brezočnik
    License: MIT
    Function: Chung Reynolds function
    \[ f(x) = \left(\sum_{i=1}^{D} x_i^2\right)^2 \]
    **Input domain:** The function can be defined on any input domain but it is usually evaluated on the hypercube \( x_i \in [-100, 100] \), for all \( i = 1, 2, ..., D \)
    **Global minimum:** \( f(x^*) = 0 \), at \( x^* = (0, ..., 0) \)
    LaTeX formats:
    - **Inline:** \( f(\mathbf{x}) = left(\sum_{i=1}^{D} x_i^2right)^2 \)
    - **Equation:**

\[
\begin{equation}
 f(\mathbf{x}) = left(\sum_{i=1}^{D} x_i^2right)^2
\end{equation}
\]

**Domain:** \(-100 \leq x_i \leq 100\)


```python
class NiaPy.benchmarks.Csendes(Lower=-1.0, Upper=1.0):
    Bases: object
    Implementation of Csendes function.
    Date: 2018
    Author: Lucija Brezočnik
    License: MIT
    Function: Csendes function
```
\( f(x) = \sum_{i=1}^{D} x_i^6 \left( 2 + \sin \frac{1}{x_i} \right) \)

**Input domain:** The function can be defined on any input domain but it is usually evaluated on the hypercube \( x_i[-1, 1] \), for all \( i = 1, 2, \ldots, D \).

**Global minimum:** \( f(x^*) = 0 \), at \( x^* = (0, \ldots, 0) \)

**LaTeX formats:**

**Inline:** \( \text{\$} f(\mathbf{x}) = \sum_{i=1}^{D} x_i^6 \left( 2 + \sin \frac{1}{x_i} \right) \text{\$} \)

**Equation:** \[
\begin{equation}
 f(\mathbf{x}) = \sum_{i=1}^{D} x_i^6 \left( 2 + \sin \frac{1}{x_i} \right)
\end{equation}
\]

**Domain:** \( -1 \leq x_i \leq 1 \)


```python
class NiaPy.benchmarks.Pinter(Lower=-10.0, Upper=10.0):
    Bases: object

    Implementation of Pintér function.

    Date: 2018

    Author: Lucija Brezočnik

    License: MIT

    Function: **Pintér function**

\[
\begin{align*}
 f(x) &= \sum_{i=1}^{D} ix_i^2 + \sum_{i=1}^{D} 20i \sin^2 A + \sum_{i=1}^{D} i \log_{10}(1 + iB^2); \\
 A &= (x_{i-1} \sin(x_i) + \sin(x_{i+1})) \quad \text{and} \quad B = (x_{i-1}^2 - 2x_i + 3x_{i+1} - \cos(x_i) + 1)
\end{align*}
\]

**Input domain:** The function can be defined on any input domain but it is usually evaluated on the hypercube \( x_i[-10, 10] \), for all \( i = 1, 2, \ldots, D \).

**Global minimum:** \( f(x^*) = 0 \), at \( x^* = (0, \ldots, 0) \)

**LaTeX formats:**

**Inline:** \( \text{\$} f(\mathbf{x}) = \sum_{i=1}^{D} ix_i^2 + \sum_{i=1}^{D} 20i \sin^2 A + \sum_{i=1}^{D} i \log_{10}(1 + iB^2); \\
A = (x_{i-1} \sin(x_i) + \sin(x_{i+1})) \quad \text{and} \quad B = (x_{i-1}^2 - 2x_i + 3x_{i+1} - \cos(x_i) + 1) \text{\$} \)

**Equation:** \[
\begin{equation}
 f(\mathbf{x}) = \sum_{i=1}^{D} ix_i^2 + \sum_{i=1}^{D} 20i \sin^2 A + \sum_{i=1}^{D} i \log_{10}(1 + iB^2); \\
A = (x_{i-1} \sin(x_i) + \sin(x_{i+1})) \quad \text{and} \quad B = (x_{i-1}^2 - 2x_i + 3x_{i+1} - \cos(x_i) + 1)
\end{equation}
\]

**Domain:** \( -10 \leq x_i \leq 10 \)

class NiaPy.benchmarks.Qing(Lower=-500.0, Upper=500.0)

Bases: object

Implementation of Qing function.

Date: 2018

Author: Lucija Brezočnik

License: MIT

Function: Qing function

\[ f(x) = \sum_{i=1}^{D} \left( x_i^2 - i \right)^2 \]

Input domain: The function can be defined on any input domain but it is usually evaluated on the hypercube \( x_i[-500, 500] \), for all \( i = 1, 2, ..., D \).

Global minimum: \( f(x^*) = 0 \), at \( x^* = (\pm i) \)

LaTeX formats:

- Inline: \( f(\mathbf{x}) = \sum_{i=1}^{D} \left( x_i^2 - i \right)^2 \)

- Equation: \[ f(\mathbf{x}) = \sum_{i=1}^{D} \left( x_i^2 - i \right)^2 \]

Domain: \(-500 \leq x_i \leq 500\)


classmethod function()

class NiaPy.benchmarks.Quintic(Lower=-10.0, Upper=10.0)

Bases: object

Implementation of Quintic function.

Date: 2018

Author: Lucija Brezočnik

License: MIT

Function: Quintic function

\[ f(x) = \sum_{i=1}^{D} \left| x_i^5 - 3x_i^4 + 4x_i^3 + 2x_i^2 - 10x_i - 4 \right| \]

Input domain: The function can be defined on any input domain but it is usually evaluated on the hypercube \( x_i[-10, 10] \), for all \( i = 1, 2, ..., D \).

Global minimum: \( f(x^*) = 0 \), at \( x^* = f(-1 \text{ or } 2) \)

LaTeX formats:

- Inline: \( f(\mathbf{x}) = \sum_{i=1}^{D} \left| x_i^5 - 3x_i^4 + 4x_i^3 + 2x_i^2 - 10x_i - 4 \right| \)

- Equation: \[ f(\mathbf{x}) = \sum_{i=1}^{D} \left| x_i^5 - 3x_i^4 + 4x_i^3 + 2x_i^2 - 10x_i - 4 \right| \]

Domain: \(-10 \leq x_i \leq 10\)

classmethod function()
class NiaPy.benchmarks.Salomon(Lower=-100.0, Upper=100.0)
Bases: object
Implementation of Salomon function.
Date: 2018
Author: Lucija Brezočnik
License: MIT
Function: Salomon function
\[ f(x) = 1 - \cos\left(2\pi \sqrt{\sum_{i=1}^{D} x_i^2}\right) + 0.1 \sqrt{\sum_{i=1}^{D} x_i^2} \]
Input domain: The function can be defined on any input domain but it is usually evaluated on the hypercube \( x_i [-100, 100] \), for all \( i = 1, 2, ..., D \).
Global minimum: \( f(x^*) = 0 \), at \( x^* = (0, 0) \)

\( \) \( \) \( \) \( \) 

LaTeX formats:
\begin{itemize}
  \item Inline: \$f(\mathbf{x}) = 1 - \cos(2\pi \sqrt{\sum_{i=1}^{D} x_i^2}) + 0.1 \sqrt{\sum_{i=1}^{D} x_i^2}$
  \item Equation: \begin{equation} f(\mathbf{x}) = 1 - \cos(2\pi \sqrt{\sum_{i=1}^{D} x_i^2}) + 0.1 \sqrt{\sum_{i=1}^{D} x_i^2} \end{equation}
  \item Domain: \$-100 \leq x_i \leq 100$\$
\end{itemize}


classmethod function()
class NiaPy.benchmarks.SchumerSteiglitz(Lower=-100.0, Upper=100.0)
Bases: object
Implementation of Schumer Steiglitz function.
Date: 2018
Author: Lucija Brezočnik
License: MIT
Function: Schumer Steiglitz function
\[ f(x) = \sum_{i=1}^{D} x_i^4 \]
Input domain: The function can be defined on any input domain but it is usually evaluated on the hypercube \( x_i [-100, 100] \), for all \( i = 1, 2, ..., D \).
Global minimum \( f(x^*) = 0 \), at \( x^* = (0, ..., 0) \)

\( \) \( \) \( \) \( \) 

LaTeX formats:
\begin{itemize}
  \item Inline: \$f(\mathbf{x}) = \sum_{i=1}^{D} x_i^4$
  \item Equation: \begin{equation} f(\mathbf{x}) = \sum_{i=1}^{D} x_i^4 \end{equation}
  \item Domain: \$-100 \leq x_i \leq 100$\$
\end{itemize}

classmethod function()

class NiaPy.benchmarks.Step (Lower=-100.0, Upper=100.0)
Bases: object

Implementation of Step function.

Date: 2018

Author: Lucija Brezočnik

License: MIT

Function: Step function

\[ f(x) = \sum_{i=1}^{D} \left( \lfloor \left| x_i \right| \right) \]

Input domain: The function can be defined on any input domain but it is usually evaluated on the hypercube \( x_i \in [-100, 100] \), for all \( i = 1, 2, ..., D \).

Global minimum: \( f(x^*) = 0 \), at \( x^* = (0, ..., 0) \)

LaTeX formats:

Inline: \$f(\mathbf{x}) = \sum_{i=1}^{D} \left( \lfloor \left| x_i \right| \right)\$

Equation: \begin{equation} f(\mathbf{x}) = \sum_{i=1}^{D} \left( \lfloor \left| x_i \right| \right) \end{equation}

Domain: \(-100 \leq x_i \leq 100\)


classmethod function()

class NiaPy.benchmarks.Step2 (Lower=-100.0, Upper=100.0)
Bases: object

Step2 function implementation.

Date: 2018

Author: Lucija Brezočnik

Licence: MIT

Function: Step2 function

\[ f(x) = \sum_{i=1}^{D} \left( \lfloor x_i + 0.5 \right) \right)^2 \]

Input domain: The function can be defined on any input domain but it is usually evaluated on the hypercube \( x_i \in [-100, 100] \), for all \( i = 1, 2, ..., D \).

Global minimum: \( f(x^*) = 0 \), at \( x^* = (-0.5, ..., -0.5) \)

LaTeX formats:

Inline: \$f(\mathbf{x}) = \sum_{i=1}^{D} \left( \lfloor x_i + 0.5 \right)^2 \$

8.3. NiaPy.benchmarks
Equation: \begin{equation} f(\mathbf{x}) = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2 \end{equation}

Domain: \(-100 \leq x_i \leq 100\)


classmethod function()

class NiaPy.benchmarks.Step3 (Lower=-100.0, Upper=100.0)

Bases: object

Step3 function implementation.

Date: 2018

Author: Lucija Brezočnik

License: MIT

Function: Step3 function

\[ f(\mathbf{x}) = \sum_{i=1}^{D} (\lfloor x_i^2 \rfloor) \]

Input domain: The function can be defined on any input domain but it is usually evaluated on the hypercube \(x_i[-100, 100]\), for all \(i = 1, 2, \ldots, D\).

Global minimum: \(f(x^*) = 0\), at \(x^* = (0, \ldots, 0)\)

LaTeX formats:

Inline: \( f(\mathbf{x}) = \sum_{i=1}^{D} (\lfloor x_i^2 \rfloor) \)

Equation: \begin{equation} f(\mathbf{x}) = \sum_{i=1}^{D} (\lfloor x_i^2 \rfloor) \end{equation}

Domain: \(-100 \leq x_i \leq 100\)


classmethod function()

class NiaPy.benchmarks.Stepint (Lower=-5.12, Upper=5.12)

Bases: object

Implementation of Stepint functions.

Date: 2018

Author: Lucija Brezočnik

License: MIT

Function: Stepint function

\[ f(\mathbf{x}) = \sum_{i=1}^{D} x_i^2 \]

Input domain: The function can be defined on any input domain but it is usually evaluated on the hypercube \(x_i[-5.12, 5.12]\), for all \(i = 1, 2, \ldots, D\).

Global minimum: \(f(x^*) = 0\), at \(x^* = (-5.12, \ldots, -5.12)\)

LaTeX formats:
**Inline:** $f(\mathbf{x}) = \sum_{i=1}^D x_i^2$

**Equation:** \begin{equation} f(\mathbf{x}) = \sum_{i=1}^D x_i^2 \end{equation}

**Domain:** $0 \leq x_i \leq 10$


```python
class NiaPy.benchmarks.SumSquares(Lower=-10.0, Upper=10.0)
Bases: object
Implementation of Sum Squares functions.
Date: 2018
Authors: Lucija Brezočnik
License: MIT
Function: Sum Squares function
\[ f(\mathbf{x}) = \sum_{i=1}^D i x_i^2 \]

**Input domain:** The function can be defined on any input domain but it is usually evaluated on the hypercube $x_i[-10, 10]$, for all $i = 1, 2, ..., D$.

**Global minimum:** $f(x^*) = 0$, at $x^* = (0, ..., 0)$

**LaTeX formats:**

**Inline:** $f(\mathbf{x}) = \sum_{i=1}^D i x_i^2$

**Equation:** \begin{equation} f(\mathbf{x}) = \sum_{i=1}^D i x_i^2 \end{equation}

**Domain:** $0 \leq x_i \leq 10$


```python
class NiaPy.benchmarks.StyblinskiTang(Lower=-5.0, Upper=5.0)
Bases: object
Implementation of Styblinski-Tang functions.
Date: 2018
Authors: Lucija Brezočnik
License: MIT
Function: Styblinski-Tang function
\[ f(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^D \left( x_i^4 - 16x_i^2 + 5x_i \right) \]

**Input domain:** The function can be defined on any input domain but it is usually evaluated on the hypercube $x_i[-5, 5]$, for all $i = 1, 2, ..., D$.

**Global minimum:** $f(x^*) = -78.332$, at $x^* = (-2.903534, ..., -2.903534)$
LaTeX formats:

**Inline:** \( f(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^{D} \left( x_i^4 - 16x_i^2 + 5x_i \right) \)

**Equation:**

\[
\begin{equation}
 f(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^{D} \left( x_i^4 - 16x_i^2 + 5x_i \right)
\end{equation}
\]

**Domain:** \(-5 \leq x_i \leq 5\)


classmethod function()

class NiaPy.benchmarks.BentCigar (Lower=\text{100.0}, Upper=\text{100.0})

Bases: object

Implementations of Bent Cigar functions.

Date: 2018

Author: Klemen Berkovič

License: MIT

Function: **Bent Cigar Function**

\[
f(x) = x_1^2 + 10^6 \sum_{i=2}^{D} x_i^2
\]

**Input domain:** The function can be defined on any input domain but it is usually evaluated on the hypercube \( x_i [-100, 100] \), for all \( i = 1, 2, ..., D \).

**Global minimum:** \( f(x^*) = 0 \), at \( x^* = (420.968746, ..., 420.968746) \)

LaTeX formats:

**Inline:** \( f(\textbf{x}) = x_1^2 + 10^6 \sum_{i=2}^{D} x_i^2 \)

**Equation:**

\[
\begin{equation}
 f(\textbf{x}) = x_1^2 + 10^6 \sum_{i=2}^{D} x_i^2
\end{equation}
\]

**Domain:** \(-100 \leq x_i \leq 100\)

Reference: [http://www5.zzu.edu.cn/__local/A/69/BC/D3B5DFE94CD2574B38AD7CD1D12_C802DAFE_BOC0C0.pdf](http://www5.zzu.edu.cn/__local/A/69/BC/D3B5DFE94CD2574B38AD7CD1D12_C802DAFE_BOC0C0.pdf)

classmethod function()

class NiaPy.benchmarks.Weierstrass (Lower=\text{100.0}, Upper=\text{100.0}, a=0.5, b=3, k_{max}=20)

Bases: object

Implementations of Weierstrass functions.

Date: 2018

Author: Klemen Berkovič

License: MIT

Function: **Weierstass Function**

\[
f(x) = \sum_{i=1}^{D} \left( \sum_{k=0}^{k_{max}} a^k \cos \left( 2\pi b^k (x_1 + 0.5) \right) \right) - D \sum_{k=0}^{k_{max}} a^k \cos \left( 2\pi b^k \cdot 0.5 \right)
\]

**Input domain:** The function can be defined on any input domain but it is usually evaluated on the hypercube \( x_i [-100, 100] \), for all \( i = 1, 2, ..., D \). Default value of \( a = 0.5, b = 3 \) and \( k_{max} = 20 \).

**Global minimum:** \( f(x^*) = 0 \), at \( x^* = (420.968746, ..., 420.968746) \)
LaTeX formats:

**Inline:** \$f(\textbf{x}) = \sum_{i=1}^D \left( \sum_{k=0}^{k_{\text{max}}} a^k \cos(2 \pi b^k (x_i + 0.5)) \right) - D \sum_{k=0}^{k_{\text{max}}} a^k \cos(2 \pi b^k \cdot 0.5)\$

**Equation:** \[
\begin{equation}
\begin{align*}
f(\textbf{x}) &= \sum_{i=1}^D \left( \sum_{k=0}^{k_{\text{max}}} a^k \cos(2 \pi b^k (x_i + 0.5)) \right) - D \sum_{k=0}^{k_{\text{max}}} a^k \cos(2 \pi b^k \cdot 0.5) \\
\end{align*}
\end{equation}
\]

**Domain:** $-100 \leq x_i \leq 100$

Reference: [http://www5.zzu.edu.cn/__local/A/69/BC/D3B5DFE94CD2574B38AD7CD1D12_C802DAFE_BCO0C0.pdf](http://www5.zzu.edu.cn/__local/A/69/BC/D3B5DFE94CD2574B38AD7CD1D12_C802DAFE_BCO0C0.pdf)

\(a = 0.5\)

\(b = 3\)

\[\text{classmethod function()}\]

\[k_{\text{max}} = 20\]

class \texttt{NiaPy.benchmarks.HGBat} (\texttt{Lower=-100.0, Upper=100.0})

Bases: \texttt{object}

Implementations of HGBat functions.

Date: 2018

Author: Klemen Berkovič

License: MIT

Function: \texttt{HGBat Function}

:math: f(\textbf{x}) = \left| \left( \sum_{i=1}^D x_i^2 \right)^2 - \left( \sum_{i=1}^D x_i \right)^2 \right|^{\frac{1}{2}} + \frac{0.5 \sum_{i=1}^D x_i^2 + \sum_{i=1}^D x_i}{D} + 0.5

**Input domain:** The function can be defined on any input domain but it is usually evaluated on the hypercube \(x_i[-100, 100]\), for all \(i = 1, 2, ..., D\).

**Global minimum:** \(f(x^*) = 0\), at \(x^* = (420.968746, ..., 420.968746)\)

LaTeX formats:

**Inline:** \$f(\textbf{x}) = \left| \left( \sum_{i=1}^D x_i^2 \right)^2 - \left( \sum_{i=1}^D x_i \right)^2 \right|^{\frac{1}{2}} + \frac{0.5 \sum_{i=1}^D x_i^2 + \sum_{i=1}^D x_i}{D} + 0.5\$

**Equation:** \[
\begin{align*}
f(\textbf{x}) &= \left| \left( \sum_{i=1}^D x_i^2 \right)^2 - \left( \sum_{i=1}^D x_i \right)^2 \right|^{\frac{1}{2}} + \frac{0.5 \sum_{i=1}^D x_i^2 + \sum_{i=1}^D x_i}{D} + 0.5 \\
\end{align*}
\]

**Domain:** $-100 \leq x_i \leq 100$

Reference: [http://www5.zzu.edu.cn/__local/A/69/BC/D3B5DFE94CD2574B38AD7CD1D12_C802DAFE_BCO0C0.pdf](http://www5.zzu.edu.cn/__local/A/69/BC/D3B5DFE94CD2574B38AD7CD1D12_C802DAFE_BCO0C0.pdf)

\[\text{classmethod function()}\]

class \texttt{NiaPy.benchmarks.Katsuura} (\texttt{Lower=-100.0, Upper=100.0, **kwargs})

Bases: \texttt{NiaPy.benchmarks.benchmark.Benchmark}

Implementations of Katsuura functions.

Date: 2018

8.3. \texttt{NiaPy.benchmarks} 73
Author: Klemen Berkovič
License: MIT
Function: **Katsuura Function**

\[ f(x) = \frac{10}{D^2} \prod_{i=1}^{D} \left( 1 + i \sum_{j=1}^{32} \frac{|2^j x_i - \text{round}(2^j x_i)|}{2^j} \right)^{10/D^{1.2}} - \frac{10}{D^2} \]

**Input domain:** The function can be defined on any input domain but it is usually evaluated on the hypercube \( x_i \in [-100, 100] \), for all \( i = 1, 2, ..., D \).

**Global minimum:** \( f(x^*) = 0 \), at \( x^* = (420.968746, ..., 420.968746) \)

**LaTeX formats:**

*Inline:* $f(\textbf{x}) = \frac{10}{D^2} \prod_{i=1}^{D} \left( 1 + i \sum_{j=1}^{32} \frac{|2^j x_i - \text{round}(2^j x_i)|}{2^j} \right)^{10/D^{1.2}} - \frac{10}{D^2}$

*Equation:* \[
\begin{equation}
  f(\textbf{x}) = \frac{10}{D^2} \prod_{i=1}^{D} \left( 1 + i \sum_{j=1}^{32} \frac{|2^j x_i - \text{round}(2^j x_i)|}{2^j} \right)^{10/D^{1.2}} - \frac{10}{D^2}
\end{equation}
\]

**Domain:** \(-100 \leq x_i \leq 100\)

Reference: [Link to Reference](http://www5.zzu.edu.cn/__local/A/69/BC/D3B5DFE94CD2574B38AD7CD1D12_C802DAFE_BC0C0.pdf)

**classmethod function()**
Get the optimization function.

class **NiaPy.benchmarks.Elliptic** *(Lower=-100.0, Upper=100.0)*

Bases: NiaPy.benchmarks.benchmark.Benchmark

Implementations of High Conditioned Elliptic functions.

Date: 2018

Author: Klemen Berkovič

License: MIT

Function: **High Conditioned Elliptic Function**

\[ f(x) = \sum_{i=1}^{D} \left( 10^6 \right)^{\frac{i-1}{D-1}} x_i^2 \]

**Input domain:** The function can be defined on any input domain but it is usually evaluated on the hypercube \( x_i \in [-100, 100] \), for all \( i = 1, 2, ..., D \).

**Global minimum:** \( f(x^*) = 0 \), at \( x^* = (420.968746, ..., 420.968746) \)

**LaTeX formats:**

*Inline:* $f(\textbf{x}) = \sum_{i=1}^{D} \left( 10^6 \right)^{\frac{i-1}{D-1}} x_i^2$

*Equation:* \[
\begin{equation}
  f(\textbf{x}) = \sum_{i=1}^{D} \left( 10^6 \right)^{\frac{i-1}{D-1}} x_i^2
\end{equation}
\]

**Domain:** \(-100 \leq x_i \leq 100\)

Reference: [Link to Reference](http://www5.zzu.edu.cn/__local/A/69/BC/D3B5DFE94CD2574B38AD7CD1D12_C802DAFE_BC0C0.pdf)

**classmethod function()**
Get the optimization function.
class NiaPy.benchmarks.Discus(Lower=-100.0, Upper=100.0)
    Bases: object

Implementations of Discus functions.

Date: 2018
Author: Klemen Berkovič
License: MIT

Function: Discus Function
\[ f(x) = x_1^2 10^6 + \sum_{i=2}^{D} x_i^2 \]

Input domain: The function can be defined on any input domain but it is usually evaluated on the hypercube \( x_i[-100, 100] \), for all \( i = 1, 2, ..., D \).

Global minimum: \( f(x^*) = 0 \), at \( x^* = (420.968746, ..., 420.968746) \)

LaTeX formats:
- Inline: \( f(\textbf{x}) = x_1^2 10^6 + \sum_{i=2}^{D} x_i^2 \)
- Equation: \[ f(\textbf{x}) = x_1^2 10^6 + \sum_{i=2}^{D} x_i^2 \]
- Domain: \(-100 \leq x_i \leq 100\)

Reference: http://www5.zzu.edu.cn/__local/A/69/BC/D3B5DFE94CD2574B38AD7CD1D12_C802DAFE_BC0C0.pdf

classmethod function()

class NiaPy.benchmarks.Michalewicz(Lower=0.0, Upper=3.141592653589793, m=10)
    Bases: object

Implementations of Michalewichz’s functions.

Date: 2018
Author: Klemen Berkovič
License: MIT

Function: High Conditioned Elliptic Function
\[ f(x) = \sum_{i=1}^{D} (10^{6})^{(i-1)/(D-1)} x_i^2 \]

Input domain: The function can be defined on any input domain but it is usually evaluated on the hypercube \( x_i[0, \pi] \), for all \( i = 1, 2, ..., D \).

Global minimum: at \( d = 2 \) \( f(x^*) = -1.8013 \) at \( x^* = (2.20, 1.57) \) at \( d = 5 \) \( f(x^*) = -4.687658 \) at \( d = 10 \) \( f(x^*) = -9.66015 \)

LaTeX formats:
- Inline: \( f(\textbf{x}) = \sum_{i=1}^{D} \sin(x_i) \sinleft( \frac{ix_i^2}{\pi} \right)^{2m} \)
- Equation: \[ f(\textbf{x}) = \sum_{i=1}^{D} \sin(x_i) \sinleft( \frac{ix_i^2}{\pi} \right)^{2m} \]
- Domain: \( 0 \leq x_i \leq \pi \)

Reference URL: https://www.sfu.ca/~ssurjano/michal.html

classmethod function()
class NiaPy.benchmarks.Levy (Lower=0.0, Upper=3.141592653589793)
Bases: object

Implementations of Levy functions.

Date: 2018
Author: Klemen Berković
License: MIT

Function: Levy Function
\[ f(x) = \sin^2(\pi w_1) + \sum_{i=1}^{D-1} (w_i - 1)^2 \left( 1 + 10 \sin^2(\pi w_i + 1) \right) + (w_d - 1)^2 \left( 1 + \sin^2(2\pi w_d) \right) \]
\[ w_i = 1 + \frac{x_i - 1}{4} \]

Input domain: The function can be defined on any input domain but it is usually evaluated on the hypercube \( x_i \in [-10, 10] \), for all \( i = 1, 2, ..., D \).

Global minimum: \( f(x^*) = 0 \) at \( x^* = (1, \cdots, 1) \)

LaTeX formats:

Inline: \$f(\textbf{x}) = \sin^2(\pi w_1) + \sum_{i=1}^{D-1} (w_i - 1)^2 \left( 1 + 10 \sin^2(\pi w_i + 1) \right) + (w_d - 1)^2 \left( 1 + \sin^2(2\pi w_d) \right) \]
\[ w_i = 1 + \frac{x_i - 1}{4} \]

Equation:
\[
\begin{equation}
 f(\textbf{x}) = \sin^2(\pi w_1) + \sum_{i=1}^{D-1} (w_i - 1)^2 \left( 1 + 10 \sin^2(\pi w_i + 1) \right) + (w_d - 1)^2 \left( 1 + \sin^2(2\pi w_d) \right) \]
\[ w_i = 1 + \frac{x_i - 1}{4} \]
\end{equation}

Domain: \(-10 \leq x_i \leq 10\)

Reference: https://www.sfu.ca/~ssurjano/levy.html

\[ \text{classmethod function()} \]

class NiaPy.benchmarks.Sphere (Lower=-5.12, Upper=5.12)
Bases: object

Implementation of Sphere functions.

Date: 2018
Authors: Iztok Fister Jr.
License: MIT

Function: Sphere function
\[ f(x) = \sum_{i=1}^{D} x_i^2 \]

Input domain: The function can be defined on any input domain but it is usually evaluated on the hypercube \( x_i \in [0, 10] \), for all \( i = 1, 2, ..., D \).

Global minimum: \( f(x^*) = 0 \), at \( x^* = (0, \cdots, 0) \)

LaTeX formats:

Inline: \$f(\textbf{x}) = \sum_{i=1}^{D} x_i^2\$

Equation:
\[
\begin{equation}
 f(\textbf{x}) = \sum_{i=1}^{D} x_i^2 \end{equation}
\]

Domain: \(0 \leq x_i \leq 10\)

classmethod function()
class NiaPy.benchmarks.Sphere2(Lower=-1.0, Upper=1.0)
    Bases: object
    Implementation of Sphere with different powers function.
    Date: 2018
    Authors: Klemen Berkovič
    License: MIT
    Function: Sun of different powers function
    \[ f(x) = \sum_{i=1}^{D} |x_i|^{i+1} \]
    Input domain: The function can be defined on any input domain but it is usually evaluated on the hypercube \[x_i[-1, 1], \text{for all } i = 1, 2, \ldots, D.\]
    Global minimum: \[f(x^*) = 0, \text{ at } x^* = (0, \ldots, 0)\]
    LaTeX formats:
    \begin{itemize}
    \item Inline: \[f(x) = \sum_{i=1}^{D} |x_i|^{i+1}\]
    \item Equation: \begin{equation} f(x) = \sum_{i=1}^{D} |x_i|^{i+1} \end{equation}
    \end{itemize}
    Domain: \[-1 \leq x_i \leq 1\]
    Reference URL: https://www.sfu.ca/~ssurjano/sumpow.html

classmethod function()
class NiaPy.benchmarks.Sphere3(Lower=-65.536, Upper=65.536)
    Bases: object
    Implementation of rotated hyper-ellipsoid function.
    Date: 2018
    Authors: Klemen Berkovič
    License: MIT
    Function: Sun of rotated hyper-ellipsoid function
    \[ f(x) = \sum_{i=1}^{D} \sum_{j=1}^{i} x_j^2 \]
    Input domain: The function can be defined on any input domain but it is usually evaluated on the hypercube \[x_i[-65.536, 65.536], \text{for all } i = 1, 2, \ldots, D.\]
    Global minimum: \[f(x^*) = 0, \text{ at } x^* = (0, \ldots, 0)\]
    LaTeX formats:
    \begin{itemize}
    \item Inline: \[f(x) = \sum_{i=1}^{D} \sum_{j=1}^{i} x_j^2\]
    \item Equation: \begin{equation} f(x) = \sum_{i=1}^{D} \sum_{j=1}^{i} x_j^2 \end{equation}
    \end{itemize}
    Domain: \[-65.536 \leq x_i \leq 65.536\]
    Reference URL: https://www.sfu.ca/~ssurjano/rothyp.html

8.3. NiaPy.benchmarks
classmethod function()
class NiaPy.benchmarks.Trid(D=2)
Bases: object
Implementations of Trid functions.
Date: 2018
Author: Klemen Berkovič
License: MIT
Function: Levy Function
\[ f(x) = \sum_{i=1}^{D} (x_i - 1)^2 - \sum_{i=2}^{D} x_i x_{i-1} \]
Input domain: The function can be defined on any input domain but it is usually evaluated on the hypercube \( x_i [-D, D] \), for all \( i = 1, 2, ..., D \).
Global minimum: \( f(x^*) = -\frac{D(D+4)(D-1)}{6} \) at \( x^* = (1(D + 1 - 1), \ldots, i(D + 1 - i), \ldots, D(D + 1 - D)) \)
LaTeX formats:

\begin{itemize}
  \item Inline: $f(\textbf{x}) = \sum_{i=1}^{D} (x_i - 1)^2 - \sum_{i=2}^{D} x_i x_{i-1}$
  \item Equation: \begin{equation} f(\textbf{x}) = \sum_{i=1}^{D} (x_i - 1)^2 - \sum_{i=2}^{D} x_i x_{i-1} \end{equation}
\end{itemize}

Domain: $-D^2 \leq x_i \leq D^2$

Reference: https://www.sfu.ca/~ssurjano/trid.html

classmethod function()
class NiaPy.benchmarks.Perm(D=10.0, beta=0.5)
Bases: object
Implementations of Perm functions.
Date: 2018
Author: Klemen Berkovič
License: MIT
Arguments: beta {real} – value added to inner sum of function
Function: Perm Function
\[ f(x) = \sum_{i=1}^{D} \left( \sum_{j=1}^{D} (j - \beta) \left( x_j^i - \frac{1}{j} \right) \right)^2 \]
Input domain: The function can be defined on any input domain but it is usually evaluated on the hypercube \( x_i [-D, D] \), for all \( i = 1, 2, ..., D \).
Global minimum: \( f(x^*) = 0 \) at \( x^* = (1, \frac{1}{2}, \ldots, \frac{1}{7}, \ldots, \frac{1}{D}) \)
LaTeX formats:

\begin{itemize}
  \item Inline: $f(\textbf{x}) = \sum_{i=1}^{D} \left( \sum_{j=1}^{D} (j - \beta) \left( x_j^i - \frac{1}{j} \right) \right)^2$
  \item Equation: \begin{equation} f(\textbf{x}) = \sum_{i=1}^{D} \left( \sum_{j=1}^{D} (j - \beta) \left( x_j^i - \frac{1}{j} \right) \right)^2 \end{equation}
\end{itemize}
Domain: $-D \leq x_i \leq D$

Reference: https://www.sfu.ca/~ssurjano/perm0db.html

\texttt{classmethod function()}

\texttt{class NiaPy.benchmarks.Zakharov(Lower=-5.0, Upper=10.0)}

\texttt{Bases: object}

Implementations of Zakharov functions.

Date: 2018

Author: Klemen Berkovič

License: MIT

Function: \textbf{Levy Function}

\[
f(x) = \sum_{i=1}^{D} x_i^2 + \left( \sum_{i=1}^{D} 0.5i x_i \right)^2 + \left( \sum_{i=1}^{D} 0.5i x_i \right)^4
\]

\textbf{Input domain}: The function can be defined on any input domain but it is usually evaluated on the hypercube $x_i[-5, 10]$, for all $i = 1, 2, ..., D$.

\textbf{Global minimum}: $f(x^*) = 0$ at $x^* = (0, \cdots, 0)$

\textbf{LaTeX formats}:

\textbf{Inline}: $f(x) = \sum_{i=1}^{D} x_i^2 + \left( \sum_{i=1}^{D} 0.5i x_i \right)^2 + \left( \sum_{i=1}^{D} 0.5i x_i \right)^4$

\textbf{Equation}: \begin{equation} f(x) = \sum_{i=1}^{D} x_i^2 + \left( \sum_{i=1}^{D} 0.5i x_i \right)^2 + \left( \sum_{i=1}^{D} 0.5i x_i \right)^4 \end{equation}

\textbf{Domain}: $-5 \leq x_i \leq 10$

Reference: https://www.sfu.ca/~ssurjano/levy.html

\texttt{classmethod function()}

\texttt{class NiaPy.benchmarks.DixonPrice(Lower=-10.0, Upper=10)}

\texttt{Bases: object}

Implementations of Dixon Price function.

Date: 2018

Author: Klemen Berkovič

License: MIT

Function: \textbf{Levy Function}

\[
f(x) = (x_1 - 1)^2 + \sum_{i=2}^{D} i(2x_i^2 - x_{i-1})^2
\]

\textbf{Input domain}: The function can be defined on any input domain but it is usually evaluated on the hypercube $x_i[-10, 10]$, for all $i = 1, 2, ..., D$.

\textbf{Global minimum}: $f(x^*) = 0$ at $x^* = (2 - \frac{2}{5^2}, \cdots, 2 - \frac{2}{5^2}, \cdots, 2 - \frac{2D^2}{5^2})$

\textbf{LaTeX formats}:

\textbf{Inline}: $f(x) = (x_1 - 1)^2 + \sum_{i=2}^{D} i(2x_i^2 - x_{i-1})^2$

Reference: https://www.sfu.ca/~ssurjano/perm0db.html
Equation:

\[
f(\textbf{x}) = (x_1 - 1)^2 + \sum_{i = 2}^D i (2x_i^2 - x_{i - 1})^2
\]

Domain: \(-10 \leq x_i \leq 10\)

Reference: https://www.sfu.ca/~ssurjano/dixonpr.html

classmethod function()

class NiaPy.benchmarks.Powell(Lower=-4.0, Upper=5.0)
Bases: object
Implementations of Powell functions.
Date: 2018
Author: Klemen Berkovič
License: MIT

Function: Levy Function

\[
f(x) = \sum_{i=1}^{D/4} \left( (x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + 10(x_{4i-3} - x_{4i})^4 \right)
\]

Input domain: The function can be defined on any input domain but it is usually evaluated on the hypercube \(x_i [-4, 5]\), for all \(i = 1, 2, ..., D\).

Global minimum: \(f(x^*) = 0\) at \(x^* = (0, \cdots, 0)\)

LaTeX formats:

Inline: $f(\textbf{x}) = \sum_{i=1}^{D/4} \left( (x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + 10(x_{4i-3} - x_{4i})^4 \right)$

Equation:

\[
f(\textbf{x}) = \sum_{i=1}^{D/4} \left( (x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + 10(x_{4i-3} - x_{4i})^4 \right)
\]

Domain: \(-4 \leq x_i \leq 5\)

Reference: https://www.sfu.ca/~ssurjano/levy.html

classmethod function()

class NiaPy.benchmarks.CosineMixture(Lower=-1.0, Upper=1.0)
Bases: object
Implementations of Cosine mixture function.
Date: 2018
Author: Klemen Berkovič
License: MIT

Function: Cosine Mixture Function

\[
f(x) = -0.1 \sum_{i=1}^D \cos(5\pi x_i) - \sum_{i=1}^D x_i^2
\]

Input domain: The function can be defined on any input domain but it is usually evaluated on the hypercube \(x_i [-1, 1]\), for all \(i = 1, 2, ..., D\).

Global maximum: \(f(x^*) = -0.1D\), at \(x^* = (0, 0, ..., 0)\)

LaTeX formats:
Inline: $f(\mathbf{x}) = -0.1 \sum_{i=1}^{D} \cos (5 \pi x_i) - \sum_{i=1}^{D} x_i^2$

Equation: begin{equation} f(\mathbf{x}) = -0.1 \sum_{i=1}^{D} \cos (5 \pi x_i) - \sum_{i=1}^{D} x_i^2 end{equation}

Domain: $-1 \leq x_i \leq 1$

Reference: http://infinity77.net/global_optimization/test_functions_nd_C.html#go_benchmark.CosineMixture

classmethod function()

class NiaPy.benchmarks.Infinity (Lower=-1.0, Upper=1.0)

Bases: object

Implementations of Infinity function.

Date: 2018

Author: Klemen Berković

License: MIT

Function: Infinity Function

\[ f(\mathbf{x}) = \sum_{i=1}^{D} x_i^6 \left( \sin \left( \frac{1}{x_i} \right) + 2 \right) \]

Input domain: The function can be defined on any input domain but it is usually evaluated on the hypercube \(x_i[-1, 1]\), for all \(i = 1, 2, ..., D\).

Global minimum: \(f(x^*) = 0\), at \(x^* = (420.968746, ..., 420.968746)\)

LaTeX formats:

Inline: $f(\mathbf{x}) = \sum_{i=1}^{D} x_i^6 \left( \sin \left( \frac{1}{x_i} \right) + 2 \right)$

Equation: begin{equation} f(\mathbf{x}) = \sum_{i=1}^{D} x_i^6 \left( \sin \left( \frac{1}{x_i} \right) + 2 \right) end{equation}

Domain: $-1 \leq x_i \leq 1$

Reference: http://infinity77.net/global_optimization/test_functions_nd_I.html#go_benchmark.Infinity

classmethod function()

class NiaPy.benchmarks.Benchmark (Lower, Upper, **kwargs)

Bases: object

function()

Get the optimization function.

plot2d()

plot3d(scale=0.32)
Nature-inspired algorithms are a very popular tool for solving optimization problems. Since the beginning of their era, numerous variants of nature-inspired algorithms were developed. To prove their versatility, those were tested in various domains on various applications, especially when they are hybridized, modified or adapted. However, implementation of nature-inspired algorithms is sometimes difficult, complex and tedious task. In order to break this wall, NiaPy is intended for simple and quick use, without spending a time for implementing algorithms from scratch.

9.1 Mission

Our mission is to build a collection of nature-inspired algorithms and create a simple interface for managing the optimization process along with statistical evaluation. NiaPy will offer:

- numerous benchmark functions implementations,
- use of various nature-inspired algorithms without struggle and effort with a simple interface and
- easy comparison between nature-inspired algorithms

9.2 Licence

This package is distributed under the MIT License.

9.3 Disclaimer

This framework is provided as-is, and there are no guarantees that it fits your purposes or that it is bug-free. Use it at your own risk!
First off, thanks for taking the time to contribute!

10.1 Code of Conduct

This project and everyone participating in it is governed by the Code of Conduct. By participating, you are expected to uphold this code. Please report unacceptable behavior to niapy.organization@gmail.com.

10.2 How Can I Contribute?

10.2.1 Reporting Bugs

Before creating bug reports, please check existing issues list as you might find out that you don’t need to create one. When you are creating a bug report, please include as many details as possible. Fill out the required template, the information it asks for helps us resolve issues faster.

10.2.2 Suggesting Enhancements

- Open new issue
- Write in details what enhancement would you like to see in the future
- If you have technical knowledge, propose solution on how to implement enhancement

10.2.3 Pull requests (PR)

Note: If you are not so familiar with Git or/and GitHub, we suggest you take a look at our Git Beginners Guide.
Note: Firstly follow the developers *Installation* guide to install needed software in order to contribute to our source code.

- Fill in the required template
- Document new code
- Make sure all the code goes through Pylint without problems (run `make check` command)
- Run tests (run `make test` command)
- Make sure PR builds (Travis and AppVeyor) goes through
- Follow discussion in opened PR for any possible needed changes and/or fixes
11.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.

11.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others’ private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting
11.3 Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

11.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

11.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at niapy.organization@gmail.com. The project team will review and investigate all complaints, and will respond in a way that it deems appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project’s leadership.

11.6 Attribution

This Code of Conduct is adapted from the homepage, version 1.4, available at http://contributor-covenant.org/version/1/4.
Python Module Index

N
NiaPy, ??
NiaPy.algorithms, 26
NiaPy.algorithms.basic, 29
NiaPy.algorithms.modified, 50
NiaPy.algorithms.other, 51
NiaPy.benchmarks, 56
Index

A

a (NiaPy.benchmarks.Weierstrass attribute), 73
Ackley (class in NiaPy.benchmarks), 59
AdaptiveGen() (NiaPy.algorithms.modified.DynNPSelfAdaptiveDifferentialEvolutionAlgorithm method), 51
AdaptiveGen() (NiaPy.algorithms.modified.SelfAdaptiveDifferentialEvolutionAlgorithm method), 50
adjustment() (NiaPy.algorithms.basic.HarmonySearch method), 43
Algorithm (class in NiaPy.algorithms), 26
alpha_new() (NiaPy.algorithms.basic.FireflyAlgorithm method), 30
Alpine1 (class in NiaPy.benchmarks), 63
Alpine2 (class in NiaPy.benchmarks), 63
AnarchicSocietyOptimization (class in NiaPy.algorithms.other), 55
ArtificialBeeColonyAlgorithm (class in NiaPy.algorithms.basic), 32
calcLuciferin() (NiaPy.algorithms.basic.GlowwormSwarmOptimizationV1 method), 41
CalculateProbs() (NiaPy.algorithms.basic.ArtificialBeeColonyAlgorithm method), 32
CameAlgorithm (class in NiaPy.algorithms.basic), 34
crossover() (NiaPy.algorithms.basic.EvolutionStrategyMpL method), 38
checkForBest() (NiaPy.algorithms.basic.ArtificialBeeColonyAlgorithm method), 33
ChungReynolds (class in NiaPy.benchmarks), 65
CosineMixture (class in NiaPy.benchmarks), 79
Crank() (NiaPy.algorithms.basic.KrillHerdV1 method), 45
crossover() (NiaPy.algorithms.basic.KrillHerdV3 method), 44
crossover() (NiaPy.algorithms.basic.KrillHerdV4 method), 44
Csendes (class in NiaPy.benchmarks), 65

D

d() (NiaPy.algorithms.basic.GravitationalSearchAlgorithm method), 49
DifferentialEvolutionAlgorithm (class in NiaPy.algorithms.basic), 30
Discus (class in NiaPy.benchmarks), 75
DixonPrice (class in NiaPy.benchmarks), 79
DynamicFireworksAlgorithm (class in NiaPy.algorithms.basic), 47
DynamicFireworksAlgorithmGauss (class in NiaPy.algorithms.basic), 48
DynNPSelfAdaptiveDifferentialEvolutionAlgorithm (class in NiaPy.algorithms.modified), 51

E
eI() (NiaPy.algorithms.other.AnarchicSocietyOptimization method), 55
ElitistSelection() (NiaPy.algorithms.basic.KrillHerdV1 method), 45
Elliptic (class in NiaPy.benchmarks), 74
EnhancedFireworksAlgorithm (class in NiaPy.algorithms.basic), 47
eval() (NiaPy.algorithms.basic.MonkeyKingEvolutionV3 method), 36
evalPopulation() (NiaPy.algorithms.basic.DifferentialEvolutionAlgorithm method), 30
evaluate() (NiaPy.algorithms.Individual method), 28
EvolutionStrategy1p1 (class in NiaPy.algorithms.basic), 37
EvolutionStrategyML (class in NiaPy.algorithms.basic), 38
EvolutionStrategyMp1 (class in NiaPy.algorithms.basic), 37
EvolutionStrategyMpL (class in NiaPy.algorithms.basic), 38
evolve() (NiaPy.algorithms.basic.GeneticAlgorithm method), 32
ExpandedGriewankPlusRosenbrock (class in NiaPy.benchmarks), 58
ExpandedScaffer (class in NiaPy.benchmarks), 61
ExplodeSpark() (NiaPy.algorithms.basic.FireworksAlgorithm method), 46
ExplosionAmplitude() (NiaPy.algorithms.basic.DynamicFireworksAlgorithmGauss method), 48
ExplosionAmplitude() (NiaPy.algorithms.basic.EnhancedFireworksAlgorithm method), 47
ExplosionAmplitude() (NiaPy.algorithms.basic.FireworksAlgorithm method), 46
function() (NiaPy.benchmarks.ExpandedScaffer class method), 61
function() (NiaPy.benchmarks.Griewank class method), 58
function() (NiaPy.benchmarks.HappyCat class method), 64
function() (NiaPy.benchmarks.HGBat class method), 73
function() (NiaPy.benchmarks.Infinity class method), 81
function() (NiaPy.benchmarks.Katsuura class method), 75
function() (NiaPy.benchmarks.Levy class method), 76
function() (NiaPy.benchmarks.Michalewicz class method), 75
function() (NiaPy.benchmarks.ModifiedSchwefel class method), 62
function() (NiaPy.benchmarks.Perms class method), 79
function() (NiaPy.benchmarks.Pinter class method), 66
function() (NiaPy.benchmarks.Powell class method), 77
function() (NiaPy.benchmarks.Quintic class method), 67
function() (NiaPy.benchmarks.Rastrigin class method), 57
function() (NiaPy.benchmarks.Ridge class method), 65
function() (NiaPy.benchmarks.Rosenbrock class method), 57
function() (NiaPy.benchmarks.Salomon class method), 68
function() (NiaPy.benchmarks.SchumerSteiglitz class method), 69
function() (NiaPy.benchmarks.Schwef class method), 60
function() (NiaPy.benchmarks.Schwefel221 class method), 60
function() (NiaPy.benchmarks.Schwefel222 class method), 61
function() (NiaPy.benchmarks.Sphere class method), 59
function() (NiaPy.benchmarks.Sphere2 class method), 77
function() (NiaPy.benchmarks.Sphere3 class method), 77
function() (NiaPy.benchmarks.Step class method), 69
function() (NiaPy.benchmarks.Step2 class method), 70
function() (NiaPy.benchmarks.Step3 class method), 71
function() (NiaPy.benchmarks.StyblinskiTang class method), 72
function() (NiaPy.benchmarks.SumSquares class method), 71
function() (NiaPy.benchmarks.Trid class method), 78
function() (NiaPy.benchmarks.Weierstrass class method), 73
function() (NiaPy.benchmarks.Whitley class method), 63
function() (NiaPy.benchmarks.Zakharov class method), 79
index

G

G() (NiaPy.algorithms.basic.GravitationalSearchAlgorithm method), 49
GaussianSpark() (NiaPy.algorithms.basic.EnhancedFireworksAlgorithm method), 47
GaussianSpark() (NiaPy.algorithms.basic.FireworksAlgorithm method), 46
generateSolution() (NiaPy.algorithms.Individual method), 28
GeneticAlgorithm (class in NiaPy.algorithms.basic), 32
generateBest() (NiaPy.algorithms.basic.FireflyAlgorithm method), 30
getBest() (NiaPy.algorithms.basic.GlowwormSwarmOptimization method), 40
getBest() (NiaPy.algorithms.other.AnarchicSocietyOptimization method), 55
getBest() (NiaPy.algorithms.basic.FireflyAlgorithm method), 30
GaussianSpark() (NiaPy.algorithms.basic.EnhancedFireworksAlgorithm method), 47
GaussianSpark() (NiaPy.algorithms.basic.FireworksAlgorithm method), 46
GaussianSpark() (NiaPy.algorithms.basic.EnhancedFireworksAlgorithm method), 47
GaussianSpark() (NiaPy.algorithms.basic.FireworksAlgorithm method), 46
GaussianSpark() (NiaPy.algorithms.basic.EnhancedFireworksAlgorithm method), 47

K

k_max (NiaPy.benchmarks.Weierstrass attribute), 73
Katsuura (class in NiaPy.benchmarks), 73
KrillHerdV1 (class in NiaPy.algorithms.basic), 43
KrillHerdV11 (class in NiaPy.algorithms.basic), 45
KrillHerdV2 (class in NiaPy.algorithms.basic), 44
KrillHerdV3 (class in NiaPy.algorithms.basic), 44
KrillHerdV4 (class in NiaPy.algorithms.basic), 44

L

Levy (class in NiaPy.benchmarks), 75
levy() (NiaPy.algorithms.basic.FlowerPollinationAlgorithm method), 31
lifeCycle() (NiaPy.algorithms.basic.CamelAlgorithm method), 34
lifeCycle() (NiaPy.algorithms.basic.FireflyAlgorithm method), 34

M

Mapping() (NiaPy.algorithms.basic.DynamicFireworksAlgorithmGauss method), 48
Mapping() (NiaPy.algorithms.basic.FireworksAlgorithmGauss method), 46
method() (NiaPy.algorithms.other.NelderMeadMethod method), 51
Michalewicz (class in NiaPy.benchmarks), 75
ModifiedSchwefel (class in NiaPy.benchmarks), 61
MonkeyKingEvolutionV1 (class in NiaPy.algorithms.basic), 35
MonkeyKingEvolutionV2 (class in NiaPy.algorithms.basic), 36
MonkeyKingEvolutionV3 (class in NiaPy.algorithms.basic), 36

I

II() (NiaPy.algorithms.other.AnarchicSocietyOptimization method), 55
improvize() (NiaPy.algorithms.basic.HarmonySearch method), 43
Individual (class in NiaPy.algorithms), 28
Infinity (class in NiaPy.benchmarks), 81
init() (NiaPy.algorithms.basic.ArtificialBeeColonyAlgorithm method), 33
init() (NiaPy.algorithms.basic.ParticleSwarmAlgorithm method), 33
init() (NiaPy.algorithms.other.AnarchicSocietyOptimization method), 55
init() (NiaPy.algorithms.other.NelderMeadMethod method), 51
initAmplitude() (NiaPy.algorithms.basic.DynamicFireworksAlgorithmGauss method), 48
initAmplitude() (NiaPy.algorithms.basic.FireworksAlgorithm method), 46
initRanges() (NiaPy.algorithms.basic.EnhancedFireworksAlgorithm method), 47

Index
Index 95

repair() (NiaPy.algorithms.Individual method), 28
Ridge (class in NiaPy.benchmarks), 64
Rosenbrock (class in NiaPy.benchmarks), 57
run() (NiaPy.algorithms.Algorithm method), 27
Runner (class in NiaPy), 25
runTask() (NiaPy.algorithms.Algorithm method), 27
runTask() (NiaPy.algorithms.basic.ArtificialBeeColonyAlgorithm method), 33
runTask() (NiaPy.algorithms.basic.BareBonesFireworksAlgorithm method), 34
runTask() (NiaPy.algorithms.basic.BatAlgorithm method), 29
runTask() (NiaPy.algorithms.basic.CamelAlgorithm method), 34
runTask() (NiaPy.algorithms.basic.DifferentialEvolutionAlgorithm method), 30
runTask() (NiaPy.algorithms.basic.DynamicFireworksAlgorithmGauss method), 48
runTask() (NiaPy.algorithms.basic.EnhancedFireworksAlgorithm method), 47
runTask() (NiaPy.algorithms.basic.EvolutionStrategy1p1 method), 37
runTask() (NiaPy.algorithms.basic.EvolutionStrategyML method), 39
runTask() (NiaPy.algorithms.basic.EvolutionStrategyMpL method), 38
runTask() (NiaPy.algorithms.basic.FireflyAlgorithm method), 30
runTask() (NiaPy.algorithms.basic.FireworksAlgorithm method), 46
runTask() (NiaPy.algorithms.basic.FlowerPollinationAlgorithm method), 31
runTask() (NiaPy.algorithms.basic.GeneticAlgorithm method), 32
runTask() (NiaPy.algorithms.basic.GlowwormSwarmOptimization method), 40
runTask() (NiaPy.algorithms.basic.GravitationalSearchAlgorithm method), 49
runTask() (NiaPy.algorithms.basic.GreyWolfOptimizer method), 31
runTask() (NiaPy.algorithms.basic.HarmonySearch method), 43
runTask() (NiaPy.algorithms.basic.KrillHerdV11 method), 45
runTask() (NiaPy.algorithms.basic.MonkeyKingEvolutionV1 method), 35
runTask() (NiaPy.algorithms.basic.MonkeyKingEvolutionV3 method), 36
runTask() (NiaPy.algorithms.basic.ParticleSwarmAlgorithm method), 33
runTask() (NiaPy.algorithms.basic.SineCosineAlgorithm method), 39
runTask() (NiaPy.algorithms.modified.DynNPSelfAdaptiveDifferentialEvolutionAlgorithm method), 51
runTask() (NiaPy.algorithms.modified.HybridBatAlgorithm method), 50
runTask() (NiaPy.algorithms.modified.SelfAdaptiveDifferentialEvolutionAlgorithm method), 50
runTask() (NiaPy.algorithms.other.AnarchicSocietyOptimization method), 55
runTask() (NiaPy.algorithms.other.HillClimbAlgorithm method), 51
runTask() (NiaPy.algorithms.other.MultipleTrajectorySearch method), 52
runTask() (NiaPy.algorithms.other.MultipleTrajectorySearchV1 method), 54
runTask() (NiaPy.algorithms.other.NelderMeadMethod method), 54
runTask() (NiaPy.algorithms.other.SimulatedAnnealing method), 53
runTask() (NiaPy.algorithms.Algorithm method), 28
Salomon (class in NiaPy.benchmarks), 68
SchumerSteiglitz (class in NiaPy.benchmarks), 68
Schwefel (class in NiaPy.benchmarks), 59
Schwefel221 (class in NiaPy.benchmarks), 60
Schwefel222 (class in NiaPy.benchmarks), 60
selectBetter() (NiaPy.algorithms.basic.DifferentialEvolutionAlgorithm method), 30
SelfAdaptiveDifferentialEvolutionAlgorithm (class in NiaPy.algorithms.modified), 50
setParameters() (NiaPy.algorithms.Algorithm method), 28
setParameters() (NiaPy.algorithms.basic.ArtificialBeeColonyAlgorithm method), 33
setParameters() (NiaPy.algorithms.basic.BareBonesFireworksAlgorithm method), 34
setParameters() (NiaPy.algorithms.basic.BatAlgorithm method), 29
setParameters() (NiaPy.algorithms.basic.CamelAlgorithm method), 35
setParameters() (NiaPy.algorithms.basic.DifferentialEvolutionAlgorithm method), 30
setParameters() (NiaPy.algorithms.basic.DifferentialEvolutionAlgorithmGauss method), 48
setParameters() (NiaPy.algorithms.basic.EnhancedFireworksAlgorithm method), 47
setParameters() (NiaPy.algorithms.basic.FontanaFireflyAlgorithm method), 30
setParameters() (NiaPy.algorithms.basic.FleaAlgorithm method), 30
setParameters() (NiaPy.algorithms.basic.FireworksAlgorithm method), 46
setParameters() (NiaPy.algorithms.basic.FlowerPollinationAlgorithm method), 31
setParameters() (NiaPy.algorithms.basic.GeneticAlgorithm method), 32
setParameters() (NiaPy.algorithms.basic.GlowwormSwarmOptimization method), 40
setParameters() (NiaPy.algorithms.basic.GlowwormSwarmOptimizationV1 method), 41
setParameters() (NiaPy.algorithms.basic.GlowwormSwarmOptimizationV2 method), 41
setParameters() (NiaPy.algorithms.basic.GlowwormSwarmOptimizationV3 method), 42
setParameters() (NiaPy.algorithms.basic.GravitationalSearchAlgorithm method), 49
setParameters() (NiaPy.algorithms.basic.GreyWolfOptimizer method), 31
setParameters() (NiaPy.algorithms.basic.HarmonySearch method), 43
setParameters() (NiaPy.algorithms.basic.HarmonySearchV1 method), 43
setParameters() (NiaPy.algorithms.basic.KrillHerdV4 method), 45
setParameters() (NiaPy.algorithms.basic.MonkeyKingEvolutionV1 method), 35
setParameters() (NiaPy.algorithms.basic.ParticleSwarmAlgorithm method), 33
setParameters() (NiaPy.algorithms.basic.SineCosineAlgorithm method), 39
setParameters() (NiaPy.algorithms.modified.DynNPSelfAdaptiveDifferentialEvolutionAlgorithm method), 51
setParameters() (NiaPy.algorithms.modified.HybridBatAlgorithm method), 50
setParameters() (NiaPy.algorithms.modified.SelfAdaptiveDifferentialEvolutionAlgorithm method), 50
setParameters() (NiaPy.algorithms.other.AnarchicSocietyOptimization method), 55
setParameters() (NiaPy.algorithms.other.HillClimbAlgorithm method), 52
setParameters() (NiaPy.algorithms.other.MultipleTrajectorySearch method), 54
setParameters() (NiaPy.algorithms.other.NelderMeadMethod method), 52
setParameters() (NiaPy.algorithms.other.SimulatedAnnealing method), 53
SimulatedAnnealing (class in NiaPy.algorithms.other), 52
SineCosineAlgorithm (class in NiaPy.algorithms.basic), 39
SparksNo() (NiaPy.algorithms.basic.FireworksAlgorithm method), 46
Sphere (class in NiaPy.benchmarks), 58, 76
Sphere2 (class in NiaPy.benchmarks), 77
Sphere3 (class in NiaPy.benchmarks), 77
Step (class in NiaPy.benchmarks), 69
Step2 (class in NiaPy.benchmarks), 69
StepH (class in NiaPy.benchmarks), 70
Stepint (class in NiaPy.benchmarks), 71
StyblinskiTang (class in NiaPy.benchmarks), 70
SumSquares (class in NiaPy.benchmarks), 71
T
U
uAmin() (NiaPy.algorithms.basic.EnhancedFireworksAlgorithm method), 47
uBestAndPBest() (NiaPy.algorithms.other.AnarchicSocietyOptimization method), 56
uCF() (NiaPy.algorithms.basic.DynamicFireworksAlgorithmGauss method), 49
uniform() (NiaPy.algorithms.Algorithm method), 28
updateRho() (NiaPy.algorithms.basic.EvolutionStrategy1p1 method), 37
updateRho() (NiaPy.algorithms.basic.EvolutionStrategyMpL method), 38
W
walk() (NiaPy.algorithms.basic.CamelAlgorithm method), 35
Weierstrass (class in NiaPy.benchmarks), 72
Whitley (class in NiaPy.benchmarks), 62
Z
Zakharov (class in NiaPy.benchmarks), 79