`nIOp` User Guide

D c mb r 19, 2013

([n]Dimensional [I]nput/[O]utput [p]rocessing)

Contents

Part I

General Information

This manual should enable the user to use existing and to create new sessions for the software nIOp. For more information concerning the code see [Bed13].

Chapter 1

Area Of Application

`nIOp` is a open-source software. Its name is a shortcut for '**n**-Dimensional **I**nput/**O**utput **p**rocesing'.

By using it you can:

- Create, read, process, view and write multidimensional data. There is no restriction to 2- or 3-dimensional problems.
- Handle structured/unstructured data of unlimited size
- Read-from and write-to a unlimited number of sources
- Visualize the readout-process
- Compair and connect multiple source-files (also of different file-types like plainText or libreOfficeCalc)

Some of the possibilities to process data are:

- Inter-/extrapolation between given points
- Filter via IF/ELSE-conditions
- Manipulation through given or self-defined functions

Through an easy-to-use interface to Python it is also possible to create, edit and automate problems without the need of a graphical interface. See 5.11 for further information.

1.1 Impressions:

HIER KOMMEN SCHÖNE BILDER REIN

Chapter 2

Requirements

nIOp is fully programmed in Python2.7 :

- http://www.python.org/download/releases/2.7.4/

... and uses the following Python modules:

- pyqtgraph v0.9.7: for Plotting
  - http://www.pyqtgraph.org/
  - this module needs either pyQt4 or pySide to work
  - if you are using windows, install pySide http://qt-project.org/wiki/PySide_Binaries_Windows (pyQt4 wont support all features)
- ooolib: to read from open-/libreOffice-calc
  - https://sourceforge.net/projects/ooolib/
- numpy/scipy: Python modules for numeric and scientific problems
  - http://www.scipy.org/Download
- bottleneck: superfast NaN-array-handling
  - https://pypi.python.org/pypi/Bottleneck
- automodinit: Solves the problem of forgetting to keep __init__.py files up to date
  - https://pypi.python.org/pypi/automodinit/0.12

Chapter 3

Installation

Because nIOp uses Python it does not has to be compiled. The installation procedure only implies:

1. The check, download and installation of all required modules.
2. The merge of the open- and restricted (if available) source.
3. The copy of the merged code to the default python directory.
4. The creation of an entry in the start-menu.

3.1 Linux

To install nIOp on Linux-based systems you only have to open a shell (terminal), go to the nIOp-directory and execute the installer via:

```
sudo su
. LINUX_INSTALLER.sh
```

3.2 Windows

There is no windows installer at the moment, so you have to install all required modules (see chapter 2) by yourself. Please ensure that windows know python (see <http://stackoverflow.com/questions/4621255/how-do-i-run-a-python-program-in-the-command-prompt-in-windows-7>).

After this simply double-click on

```
setup.py
```

If everything went well you may see an entry of nIOp in the start-menu.

Chapter 4

Support

The core of nIOp it is published under GPLv3[1] and can be downloaded at https://github.com/radjkarl/nIOp, so feel free to fork the code and collaborate.

All bugs and issues found in this part can be posted at
https://github.com/radjkarl/nIOp/issues

Proposed improvements and questions can be registered at
https://github.com/radjkarl/nIOp/wiki

[1] General Public Licen e, ee http://de.wikipedia.org/wiki/GNU_General_Public_Licen e

Chapter 5

Start nIOp

5.1 The nIOp-Launcher

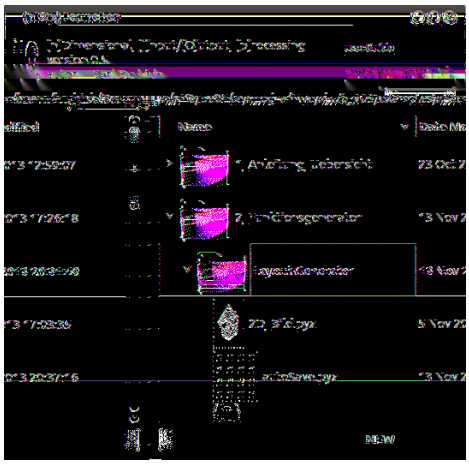

Figure 5.1.1: The nIOp-Launcher

After your first start of nIOp you will be asked to enter your project directory. All created procedure will be stored there. If this is done you will see the launcher (figure 5.1.1). Press 'NEW' to start a new session in a independent window. Part III will give you more information on this.

The start screen is more or less a file manager showing you all folders and session-files (ending with *.pyz) including an image of a screen (if existant) and the last date of change. You have the following possibilities:

- double-klick on a session-file:
  - Start this nIOp-session

- right-klick on a file or direktory:
  - opens menu to delete or rename the file or folder
  - create a new folder
  - delete the file or folder
- right-klick on a session-file:
  - edit the start-script (see section §5.11) before starting the session
  - run the session in debug mode (see chapter 8)
- drag and drop files or directories to move them

5.2 From Desktop (Windows only)

To start a nIOp-session in Windows you can simply doubleclick on a *.pyz file.

5.3 From Terminal

1. Open a terminal
   a) Linux: use a terminal-client like 'xTerm' or 'gnome-terminal'
   b) Windows: 'Start' -> 'Run' (ger:'Ausführen') -> type: '`cmd`'
2. Go the directory containing the session
   a) Linux: `cd Path/to/Session/`
   b) Windows: `cd Path\To\Session`
3. Command to start the session through python:
   a) `python mySession.pyz`
   b) Windows: If Python is not known to the system, let it knot were to find the Python-executable ensure that windows know python[1].

[1] ee http:// tackoverflow.com/que tion /4621255/how-do-i-run-a-python-program-in-the-command-prompt-in-window -7

Part II

Introduction

5.4 The Readout Procedure

After starting the readout procedure several validy checks are done to ensure the sucess of the following steps e.g:

- are dimension defined
- are those dimensions connected to at least one input source.
- has every input and output source at least one assigned dimension

If at least one basisDimension includes all values (see XXX), a first readout-process will initiated and the min/max values were set to the range of that dimension. The whole readoutprocess consists of the following steps described in pseudo code:

```
new sample = (, ,...)
for every input source:
    for every dimension in that source:

        assign a value from the source to the dimension
    for every dimension in that source:

        process the value of the dimension with through its appendend generators
    sample = (dimValue1, dimValue2 ,...)
    fill and plot the target matrixes with the sample
    for every output source:
        write sample to source
```

5.5 Input sources

Using more source-classes can be usefull to:
* combine (**same merge-dimensions**) * compare (**different merge-dimensions**)

values. When more sources are used each source has to have **the same basis (with same names and units)** Different ranges and resolutions will be fitted. If different sources includes mergeDimensions of the same name (**and unit**) their values are assembled.

5.5.1 PlainText

5.5.2 LibreOfficeCalc

5.5.3 Stream

5.6 Output sources

5.6.1 PlainText

5.7 Dimensions

5.7.1 Basisdimension

5.7.2 MergeDimension

5.7.2.1 Mergemethods

5.7.3 Helpdimension

5.8 Target Matrices

Depending on the number of used basis- and mergeDimensions all matrix-targets create three different matrices:

* basisMatrix (1D-numpy.array) * mergeMatrix (nD-numpy.array) * densityMatrix (nD-numpy.array)

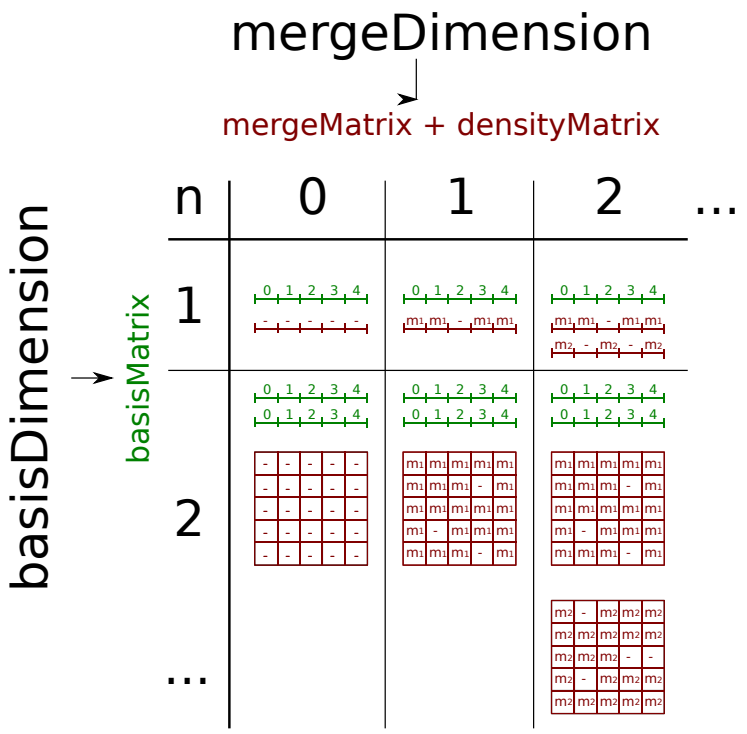

Figure 5.8.1:

5.8.1 basismatrix

A basisMatrix in a range of values from the smallest to the highest possible value of its basisDimension. Fig. XXX shows this for a range from 0 to 4. The type of the range depends, whether its a linear, a logarythmic or other sorting.

5.8.2 mergeMatrix

The position of a merge value in it's merge-, density- and varianceMatrix is defined by the value of it's basisDimensions.

One example:

You have one mergeDimension, called 'y' and one basisDimension, called 'x' which is defined between 0-1 and has a resolution of 5. This defines a basisMatrix 'X' of: [0.1, 0.3, 0.5, 0.7, 0.9]

We have one sample:$s = (v_x, v_y) = (0.55, 1.3)$

5.8.3 coarse sorting

In a coarseMatrix a merge value will be inserted at the nearest position of the basisMatrix given by the following fomular:

$$p_0 = min(abs(X - v_x)) \tag{5.8.1}$$

Each merge value counts entirely, so the value of densityMatrix at p_0 increases about one (see figure 5.8.2).

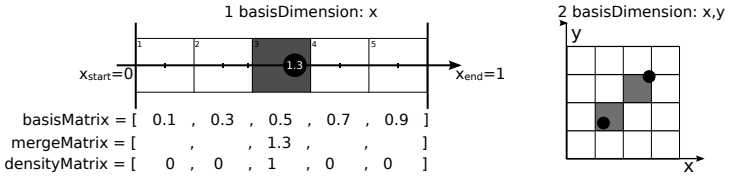

Figure 5.8.2: Example of coarse sorting

5.8.4 fine sorting

In addition to the determining of the nearest position described in equation (5.8.1) the fine sorting specify the second nearest position p_1 and disperse the point-density to d_0 and d_{01} according to:

$$i = \frac{\triangle v_1 - \triangle v_0}{\triangle v_1} \tag{5.8.2}$$

$$\triangle v = v_x - X[p] \tag{5.8.3}$$

$$d_0 = 1 - i \tag{5.8.4}$$

$$d_{01} = i \tag{5.8.5}$$

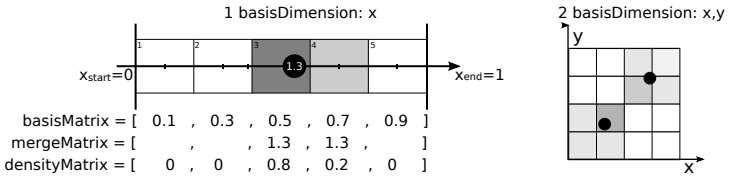

Figure 5.8.3: Example of coarse sorting

5.9 Generators

5.9.1 Channels

5.9.2 Processes

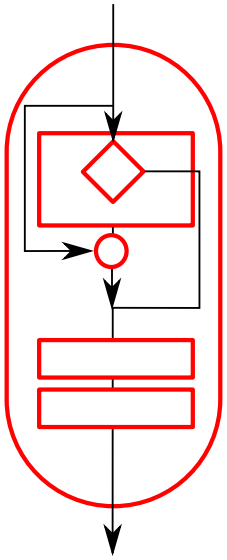

Figure 5.9.1:

drag/drop , menu right mouse

5.10 PostProcesses

5.10.1 InterpolateFine

5.11 Storage format

Every stored nIOp-session will be stored in a compressed zip-container. The file Filetype is PYZ ('python zip'). Every container-file contains at least a file named '__main__.py'. This file includes every step that is needed to rebuild the session. See XX for more information on that.

- __main__.py
- Source/
- Target/

5.12 The Python Interface

As already described every stored nIOp-session needs one file that contains all comands to rebuild it, called '__main__.py'.

This file is indeed a simple python script consisting of the following comand-types:

- Imports
  - comands like 'import nIOp' or 'from nIOp.target.coarseMatrix import coarseMatrix' imports the mentiones modules and whith this make them usable (known) for the following procedures.
- Instance creation
  - nIOp is fully object-orientated. Thats why .....
- setParam-Comands
  - +++++ GUI.setParam

Beside that the file includes comands to extract the pyz-file into a hidden folder (starting with '.') in the same directory and to delete this folder when the session will be closed. Because of the pythonic-character of the file you are able to directly manipulate the file. Examples for this are given in XXXXXXXX.

Part III

The Graphical Interface

Chapter 6

Introduction

figure 6.0.1 comment the Graphical User Interface (GUI) of a nIOp Session. The delimited sections will be described in section §6.2

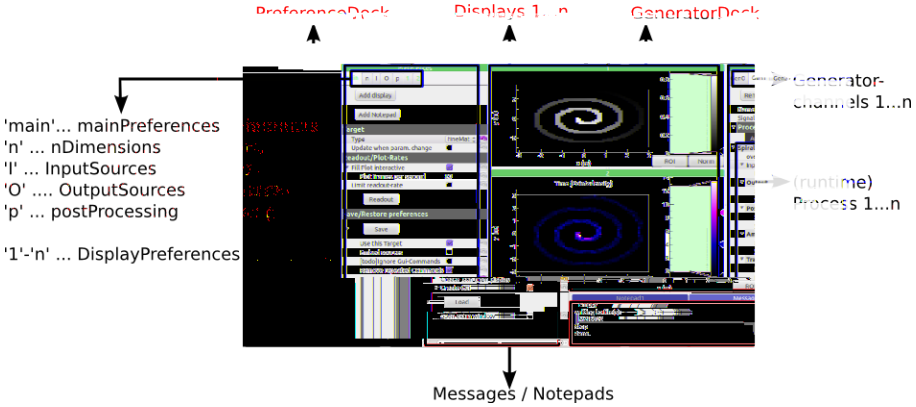

Figure 6.0.1: The Graphical User Interface of a nIOp Session

6.1 Shortcuts

nIOp uses a shortcut function delivered by application framework Qt[1]

[1] Cla name: 'QKeySeqence', ee http:// rinikom.github.io/py ide-doc /PySide/QtGui/QKeySequence.html for Shortcut in other OS

Name	Shortcut (under Linux)	Description
Save	[Ctrl]+[s]	Save the Session. Ovewrite an existing one
SaveAs	[Ctrl]+[Shift]+[s]	Save a session unter new name
Open	[Ctrl]+[o]	Open a new session
Fullscreen	[F11]	Switch to fullscreen
Fullscreen for Display no. 'n'	[Ctrl]+[1]...[9]	Show the chosen display in fullscreen
Exit	[Alt]+[F4]	Exit the nIOp session
Toggle Readout	[Esc]	Start the readout-procedure as described in 5.4
Expand, Fold	[+], [-]	Expand, Fold parameters in the Gui

Table 6.1: Shortcuts used in nIOp

6.2 Features

The window of a nIOp session is build by movable widgets, called 'Docks'. It is possible to drag and drop every dock side by side and above other docks. It is also possible to showing and hiding docks right at the start of the session.

6.2.1 Preference Dock

This dock contains every parameter that is necassary to read and write multi-dimensional samples to sources.

6.2.1.1 'main' Tab

Target

- Type [list]
  - coarseMatrix, see figure 5.8.2
  - fineMatrix, see figure 5.8.3
- Update, when parameters change
  - xxxxxxxxxxxxxxx

Readout/Plot-Rates

- Fill plot interactive [boolean]
  - True: a plot is drawn with a specific refresh-rate while the readout process

  - False: a plot will be drawn at the end of the readout process

- Plot frames per second [integer]
  - the refresh-rate of the interactive plot
- Limit readout-rate [boolean]
  - True: limit the number of samples per second that will be read and write from and to the sources
  - False: the readout rate will be as fast as possible
- Lines per second [integer]
  - Numbers of samples per second to read/write from/to sources
- Readout [button]
  - Press this button to toggle the readout process

Save/Restore-Preferences

- Save [button]
  - Press this button to save your session
- Use this target [boolean]
  - XXXXXXX
- Embedd sources [boolean]
  - True: The files of all input sources will be stored in the session file. This allows you to share your session with others even when they dont have your sources. Beware: Huge source files couse huge session files
- Ignore Gui commands [boolean]
  - XXXXXXXXX
- Remove repeated commands [boolean]
  - False: Thus every manipulation of a parameter of the GUI is tracked to recreate a saved session long works and cause very long start-scripts that need a long time to load.
  - True: In case the same parameter will be maniplutated multiple times only the last time will be stored into the start-script.

- Create icon from display [list]
  - Define a display to take an icon from. This icom will be shown for the actual session in the nIOp launcher.
- Create GUI [boolean]
  - True: The GUI will be started when the session will be loaded.
  - False: The GUI is only one medium for the purpose of setting up a session. Due it is also possible to create and control everything through a Python interface (see section §5.12) you don't necassarily need the GUI to solve your problems.
- Load [button]
  - Press this button to load a new session
- Open extra window [boolean]
  - True: The loaded session will be opened in e new window
  - False: The loaded session will replace the actual one.

6.2.1.2 'n'-Dimension Tab

6.2.1.3 'I'-Input Tab

6.2.1.4 'O'-Output Tab

6.2.1.5 'p'-Postprocessing Tab

6.2.1.6 '1-...'-Display Tab

6.2.2 Displays

6.2.3 Message Dock

6.2.4 Notepad Dock

6.2.5 Generator Dock

6.2.5.1 Channel Tab

6.2.5.2 Processses

Chapter 7

Tutorials

7.1 Visualitze a macromolecule

7.1.1 Situation

Some scientist from the department for molecule and boring calculations gave a file not just containing the positions of several atoms building a new fancy macromolecule but the positions were those atoms were most likely. Hey the file in named 'XXXXX.pdd', has a size of 700kb and contains XXX samples.

Your task will be to create an image of that molecule.

#ATOM#COUNTER#ATOMART N... STICKSTOFF,POS.#METIONIN zug. aminosäure#A...k.p.#1... 1. aminosäure#x#y#z#menge#ausgangspn# ATOM 1 N MET A 1 0.000 0.000 0.000 1.00 0.00 ^M ATOM 2 CA MET A 1 1.458 0.000 0.000 1.00 0.00 ^M

7.1.2 Procedure:

1. Have a look at that file. If you are a linux user the best will be to:
   a) open a terminal
   b) go the the directory containing the file - and open is typing vi XXX.XXX
   c) you can close is again typing ':q'
2. If you are a windows user... well ... try to open the file anyway. You will see that it takes some time to load and that it is very hard to create a plot from this file. If you are lucky to plot that file with a conventional tool than just imagine that the real file containing all positions of the molecule will be 100 times bigger that this file.
3. Open the nIOp launcher by clicking on the nIOp-button in your Start-menu (sorted under Office and Graphics)
4. Click on the 'NEW' button at the bottom. That will open a new session.

5. Click on the 'n' tab at the upper left. The 'n'-tab ('n'-dimensional) contains all dimensions used in our session. At first you only will see one 'mergeDimension'.
6. Give that dimension a name by clicking on the 'name'-parameter and typing 'Atoms'.
7. Now create a new basisDimension and name it 'x', for the x-direction.
8. Change the tab by clicking on the 'I'-tab at the right of it. This tab ('I'...Input) contains all input-sources needed in our session and is empty at the moment.
9. We add a new source by clicking on 'add' and choosing 'plainText' because that molecule-file is just a simple plain-text document.
10. I you want you can give this source a name by right-clicking in the 'plainText'-parameter and choosing 'rename'.
11. Now allocate the dimensions the that source ...
12. ################
13. If proove whether you were sucessfull go back to the nIOp-launcher and open (FINAL###).

7.2 Create a 1D function generator

7.2.1 Situation

Your assistant for communication-technics came to your and asked you to generate some samples of a sine-like signal. Hey want's you to sand hin the signal in a plain text format including samplenumber, sampletime and signal. Hey described the sine-like-signal like follows:

- ...

7.2.2 Procedure

1. Open the nIOp-launcher, start a new session by clicking on 'NEW' at the bottom
2. Go to the 'n' tab and change the name of the first mergeDimension to 'Signal'
3. Due this is a 1-dimensional problem we will need one basisDimension on which the values of the signals will be merged.
4. Got to the new basisDimension, rename it to 'Time' and define 's' as it's unit.

5. Since your assistant needs a very long and high frequency signal you would need very high resolution for your basis to plot everything well. Therefore its better to plot only the history of the last - let's say 100 - signal values. Now: Go to the parameter 'include' and choose 'chronic' (see XXX for more information on this feature). To plot the last 100 values change 'resolution' to '100'.

6. To assign values to the dimensions we need to add a new source. On our case we wand to create all values by our self. That's why nee need to add in the 'i' Input-Tab a new 'empty' source.

7. We want that this empty source has access to both of the dimensions, so we 'add Dimensions' and klick on both entries.

8. Now click on [Esc] - the shortcut to toggle the readout procedure. You will see ... nothing. Because there are no values in an empty source. Click [Esc] again to the the readout.

9. To generate values we need to one 'Generator' for both of the dimensions through clicking on 'add Generator'->'NEW'

10. This opens a new dock a the right of our window. This is the generator-Dock. Click on the 'Gen1' tab. You will see that these generator gets its signals from the 'signals'

7.3 Create a 2D function generator

Chapter 8

Debugging

8.1 General

There are two ways to run nIOp-sessions in the debug mode:

- graphical:
  1. open the nIOp launcher
  2. right click on your session and click on 'Run in debug mode'
- from terminal:
  1. add '-d' or '–debug' to your command, e.g.:

     ```
     python mySavedSession.pyz -d
     ```

This will open your session from an extra terminal and will in addition create a log-file located in the same directory o the session. Using the debug mode you can reproduce the source of your problem.

If you have to change commands on the start-script of your session you can eighter:

- using nIOp:
  1. open the nIOp launcher
  2. right click on your session and click on 'Edit start script'
  3. An editor will open showing the start-script. Edit and save the script as desired (see 5.12 for more information of the structure of the start-script). Close the editor.
  4. Click on 'Send' in the popup-window of the launcher.
- using filemanager:

1. open your session through an archivemanager.
2. open, edit and save start-script, called '__main__.py'. Update the archive.

If you found bugs, please in any case send me an mail including the log-file or post the bug on the website of the project (WERE??)

8.2 'My saved session won't start'

The reasons for this problem are most likely:

- changed names of the parameters (of the gui) because of version-conflicts of nIOp. Here you can:
  - change the names of that parameters according to the new ones (found in the Gui) in the start-script
- some commands of the start-script raise errors that were not handled properly. In this case you can:
  - fix the problem in the code :-)
  - delete or uncommend (add '#' infront of the line) the appropriate commands.

Bibliography

[Bed13] Bedrich, Karl: *nIOp Programmers Guide*. 2013 www.xxxxxx.de I