
PyNomo Documentation

Release 0.3.0

Ron Doerfler, Leif Roschier

September 07, 2015

1	Installation	3
1.1	OSX Installation	3
1.2	Linux installation	4
1.3	Windows installation	4
1.4	Docker installation	4
2	Software documentation	7
2.1	Basic blocks	7
2.2	Axes	8
2.3	Combination of blocks	8
2.4	Transformations	8
2.5	Manual	8
3	Axes	9
3.1	Common axis params	9
4	Main params	13
4.1	List of main params	13
5	Type reference	15
5.1	Type 1	15
5.2	Type 2	18
5.3	Type 3	21
5.4	Type 4	24
5.5	Type 5	28
5.6	Type 6	34
5.7	Type 7	37
5.8	Type 8	39
5.9	Type 9	42
5.10	Type 10	45
6	Examples	49
6.1	Example: Amortized loan calculator	49
6.2	Example Photography exposure	53
7	License	65

pyNomo is a python library for making nomographs (or nomograms) that are graphical calculators. Nomographs are defined as a python script that consists in most part of dictionaries.

INSTALLATION

pyNomo is a [python2](#) library and thus requires working python installation on the computer. pyNomo stands on the shoulders of (read: requires) the python packages: [numpy](#), [scipy](#) and [pyx](#) that requires LaTeX-installation.

For editing pyNomo scripts any text browser works but integrated development environment (IDE) for python can speed up developments. Good free IDE alternatives are for example [PyCharm community edition](#) and [spyder](#).

1.1 OSX Installation

In OSX [Macports](#) is an effective tool to manage open-source software. In the following a MacPorts environment is set for Python and pyNomo. *sudo* runs the commands as super-user and requires it's password to be given.

First install python 2.7

```
$ sudo port install python27
```

One can list available python versions on the system with command

```
$ sudo port select --list python
```

Select MacPorts python 2.7

```
$ sudo port select --set python python27
```

Install python package index tool (pip)

```
$ sudo port install py27-pip
```

and set it active

```
$ port select --set pip pip27
```

Now python environment should be correct to be run from `/opt/local/Library/....` Now install other required packages.

```
$ sudo port install py27-numpy
$ sudo port install py27-scipy
$ sudo port install py27-pyx
$ sudo pip install pynomo
```

1.2 Linux installation

In [Debian](#) Linux distribution and in its [derivatives](#) (for example [Ubuntu](#) and [Raspbian](#)) `pynomo` can be installed using `apt-get` with the following commands. `sudo` runs the commands as super-user and requires it's password to be given.

```
$ sudo apt-get -y install python
$ sudo apt-get -y install python-pyx
$ sudo apt-get -y install python-pip
$ sudo apt-get -y install python-numpy
$ sudo apt-get -y install python-scipy
$ sudo pip install pynomo
```

1.3 Windows installation

1. Download and install [python 2.7.x](#) from [www.python.org/downloads/](#) . `pyNomo` is not yet compatible with python 3.x.x
2. Download and install [MIKTeX LaTeX](#) -distribution from [http://miktex.org/download](#).
3. Download and install [numpy](#) from [sourceforge.net/projects/numpy](#).
4. Download and install [scipy](#) from [sourceforge.net/projects/scipy](#).
5. Download and install [PIL \(python imaging library\)](#) from [http://effbot.org/downloads/](#). `PIL` is required by some `pyx` packages. `pynomo` might work without `PIL`.

`pyx` (python graphics package) installation is more tricky. Either

- Download [pyx 0.12.1](#) (python graphics package) from [http://sourceforge.net/projects/pyx/files/pyx/0.12.1/PyX-0.12.1.tar.gz/download](#)
- Uncompress the file `PyX-0.12.1.tar.gz` using for example `7-zip`.
- Open command prompt (cmd) and go to the uncompressed folder that contains file `setup.py`.
- run command `python setup.py install`

or cross your fingers and just run:

```
> pip install --allow-external pyx pyx
```

on command prompt with administrative rights.

Finally `pyNomo` is installed either by downloading installer from [http://sourceforge.net/projects/pynomo/](#) and by running it. Other choice to try is to run:

```
> pip install pynomo
```

on command line. Tedious, huh! If you find simpler Windows recipe, please email it to the maintainer of the project.

1.4 Docker installation

[Docker](#) is a platform to create a sandboxed virtualized environments. In the following example

Dockerfile a virtualized **Ubuntu** is created that has pyNomo installed with all requirements:

```
FROM ubuntu

# Install required packages:
# python, pyx, pip, numpy, scipy, pynomo and their requirements
RUN apt-get update
RUN apt-get -y upgrade
RUN DEBIAN_FRONTEND=noninteractive apt-get -y install python
RUN DEBIAN_FRONTEND=noninteractive apt-get -y install python-pyx
RUN DEBIAN_FRONTEND=noninteractive apt-get -y install python-pip
RUN DEBIAN_FRONTEND=noninteractive apt-get -y install python-numpy
RUN DEBIAN_FRONTEND=noninteractive apt-get -y install python-scipy
RUN DEBIAN_FRONTEND=noninteractive pip install pynomo

# Add /app directory and make it working dir
RUN mkdir -p /app
ADD . /app
WORKDIR /app

# Set the default command to execute -> "python my_pynomo_file.py"
CMD ["python", "my_pynomo_file.py"]
```

Docker container (environment) *my_pynomo_docker* is built in the directory */my_directory_path* that has the file *Dockerfile* with command

```
$ docker build -t my_pynomo_docker .
```

Once environment is built and *my_pynomo_file.py* is in directory *'/my_directory_path/pdf_py_dir/* one can run

```
$ docker run -i -v /my_directory_path/pdf_py_dir:/app my_pynomo_docker
```

that runs command `python my_pynomo_file.py` inside */app* directory of container that is mapped to directory */my_directory_path/pdf_py_dir* of the host system. That way a folder is used to share the script file and the generated pdf file between host system and the container (virtualized Linux environment).

SOFTWARE DOCUMENTATION

Nomographs of PyNomo are constructed by writing a python script that defines the nomograph and calls class Nomographer to build the nomograph.

Nomograph is constructed by defining axis parameters that are used to build a block. Many blocks are possibly aligned with each other and construct the nomograph.

A simple example of pseudocode of typical PyNomo structure is the following:

```
from pynomo.nomographer import * # this loads the needed pynomo class
# define block 1
axis_params_1_for_block_1 = {...}
axis_params_2_for_block_1 = {...}
axis_params_3_for_block_1 = {...}
block_1 = {...}

# define block 2
axis_params_1_for_block_2 = {...}
axis_params_2_for_block_2 = {...}
axis_params_3_for_block_2 = {...}
block_2 = {...}

# define nomograph
main_params={
    'filename':'filename_of_nomograph.pdf', # filename of output
    'block_params':[block_1,block_2],      # the blocks make the nomograph
    'transformations':[('scale paper',)],  # these make (projective) transformations for the canves
}
# create nomograph
Nomographer(main_params)
```

It is to be noted that nomograph is defined as python dicts that constitute one dict that is passed to Nomographer class.

2.1 Basic blocks

The following blocks are the core of PyNomo. These are used as easy building blocks for nomograph construction. If these do not suffice one can build as complex nomograph as one wishes by using determinants in type 9.

Type 1	$F_1(u_1) + F_2(u_2) + F_3(u_3) = 0$	Three parallel lines
Type 2	$F_1(u_1) = F_2(u_2) F_3(u_3)$	“N” or “Z”
Type 3	$F_1(u_1) + F_2(u_2) + \cdots + F_N(u_N) = 0$	N parallel lines
Type 4	$\frac{F_1(u_1)}{F_2(u_2)} = \frac{F_3(u_3)}{F_4(u_4)}$	“Proportion”
Type 5	$F_1(v) = F_2(x, u).$	“Contour”
Type 6	$u = u$	“Ladder”
Type 7	$\frac{1}{F_1(u_1)} + \frac{1}{F_2(u_2)} = \frac{1}{F_3(u_3)}$	“Angle”
Type 8	$y = F(u)$	“Single”
Type 9	$\begin{vmatrix} F_1(u_1[v_1]) & G_1(u_1[v_1]) & H_1(u_1[v_1]) \\ F_2(u_2[v_2]) & G_2(u_2[v_2]) & H_2(u_2[v_2]) \\ F_3(u_3[v_3]) & G_3(u_3[v_3]) & H_3(u_3[v_3]) \end{vmatrix} = 0$	“General”
Type 10	$F_1(u) + F_2(v) F_3(w) + F_4(w) = 0$	One curved line

2.2 Axes

Defining axes and their appearance is major work in nomograph construction. Different possibilities are illustrated in examples of axes parameters.

2.3 Combination of blocks

If a nomograph consists of many equations that are aligned, a compound nomograph is constructed.

2.4 Transformations

Scales shall be transformed in order to tune the appearance.

2.5 Manual

Article [“Creating Nomograms with the PyNomo Software”](#) by Ron Doerfler is a detailed manual for using PyNomo.

3.1 Common axis params

Table 3.1: Common axis params

parameter	default value	explanation
'ID'	'none'	String. To identify the axis.
'tag'	'none'	String. To align blocks w.r.t each other along axes with same tag.
'dtag'	'none'	String. To double-align blocks w.r.t each other along axes with same tag.
'title'	''	String. Axis title.
'title_x_shift'	0.0	Float. Title shift in x-direction.
'title_y_shift'	0.25	Float. Title shift in y-direction.
'scale_type'	'linear'	String. Scale type. Can be 'linear': linear scale. 'log': logarithmic scale. 'smart linear': linear scale with equal spacings. 'smart log': logarithmic scale with equal spacings, can also have negative values. 'manual point': Points and corresponding text positions are given manually in 'manual axis data'. No line is drawn. 'manual line': Ticks and corresponding text positions are given manually in 'manual axis data'.
'tick_levels'	4	Integer. How many levels (minor, minor-minor, etc.) of ticks are drawn. Largest effect to 'linear' scale.
'tick_text_levels'	'3'	Integer. How many levels (minor, minor-minor, etc.) of texts are drawn. Largest effect to 'linear' scale.
'tick_side'	'right'	String. Tick and text side in final paper. Can be: 'right' or 'left'
'reference'	False	Boolean. If axis is treated as reference line that is a turning point.
'reference_padding'	'0.2'	Float. Fraction of reference line over other lines.

Continued on next page

Table 3.1 – continued from previous page

parameter	default value	explanation
'manual_axis_data'	{}	Dict. Manually set tick/point positions and text positions. Could be for example: <code>{1:'1', 3.14:r'\$\pi\$', 5:'5', 7:'seven', 10:'10'}</code>
'title_draw_center'	False	Boolean. Title is drawn to center of line.
'title_distance_center'	'type_9'	String. To double-align blocks w.r.t each other along axes with same tag.
'title_opposite_tick'	True	Boolean. Title in opposite direction w.r.t ticks.
'align_func'	lambda u:u	func(u). function to align different scales.
'align_x_offset'	0.0	Float. If axis is aligned with other axis, this value x offsets final scale.
'align_y_offset'	0.0	Float. If axis is aligned with other axis, this value y offsets final scale.
'text_format'	r'\$%4.4g\$ '	String. Format for numbers in scale.
'extra_params'	[{}], ...]	Array of Dicts. List of dictionary of params to be drawn additionally.
'text_distance_#'	x.x	Float. where $\# = 0, 1, 2, 3$ or 4. Distance of text from scale line. Number corresponds to the level, where 0 is the major tick and 4 is the most minor ticks.
'grid_length_#'	x.x	Float. where $\# = 0, 1, 2, 3$ or 4. Length of the tick. Number corresponds to the level, where 0 is the major tick and 4 is the most minor ticks.
'text_size_#'	x.x	Float. where $\# = 0, 1, 2, 3$ or 4. Text size. For example: <code>text.size.small</code> , <code>text.size.scriptsize</code> or <code>text.size.tiny</code> . Number corresponds to the level, where 0 is the major tick and 4 is the most minor ticks.
'text_size_log_#'	x.x	Float. where $\# = 0, 1$ or 2. Text size. For example: <code>text.size.small</code> , <code>text.size.scriptsize</code> or <code>text.size.tiny</code> . Number corresponds to the level, where 0 is the major tick and 2 is the most minor ticks.
'full_angle'	False	Boolean. If true, text can be upside down, otherwise ± 90 degrees from horizontal. Good for example for full circle scales.
'extra_angle'	0.0	Boolean. Title is drawn to center of line.
'title_draw_center'	False	Float. Angle to rotate tick text from horizontal along tick.
'text_horizontal_align_center'	center	Boolean. Aligns tick text horizontally to center. Good when text rotated 90 degrees.

Continued on next page

Table 3.1 – continued from previous page

parameter	default value	explanation
'turn_relative'	False	Boolean. Side left or right is relative according to traveling of scale from min to max.
'arrow_size'	0.2	Float. Used with arrow scale.
'arrow_length'	1.0	Float. Used with arrow scale..
'arrow_color'	color.rgb.black	Color. Used with arrow scale.
'axis_color'	color.rgb.black	Color. Color of axis.
'text_color'	color.rgb.black	Color. Color of tick texts.
'extra_titles'	[]	Array. List of extra title dicts for scale. Could be i.e. “[{‘dx’:1.0, ‘dy’:1.0, ‘text’:‘extra title 1’, ‘width’:5, ‘pyx_extra_defs’: [color.rgb.red,text.size.Huge]}, {‘text’: ‘extra title 2’}]”.
'base_start'	None	None/Float. Defines number with 'base_stop' (instead of 'u_min' or 'u_max') to find major tick decades.
'base_stop'	None	None/Float. Defines number with 'base_start' (instead of 'u_min' or 'u_max') to find major tick decades.

MAIN PARAMS

Main params define the top level properties of the nomograph.

4.1 List of main params

Table 4.1: General params

parameter	default value	explanation
'filename'	'pynomo_default.pdf'	String. Filename of generated file. .pdf and .eps formats supported.
'paper_height'	20.0	String. Height of paper (roughly, ticks and texts extend this).
'paper_width'	20.0	String. Width of paper (roughly, ticks and texts extend this).
'block_params'		Array of Blocks. List of blocks that make the nomograph.
'transformations'	[('rotate', 0.01), ('scale paper')]	Array of tuples. List of transformations to transform nomograph.
'title_str'	' '	String. Title string of nomograph.
'title_x'	paper_width/2.0	Float. Title x-position.
'title_y'	paper_height	Float. Title y-position.
'title_box_width'	paper_width/2.2	Float. Title box width.
'title_color'	'color.rgb.black'	Color. Title color.
'make_grid'	False	Boolean. If True, draws grid to help position texts, etc.
'pre_func'	None	func(context). PyX function(canvas) to draw under nomograph. Function definition could be:
'post_func'	None	func(context). PyX function(canvas) to draw over nomograph. Definiton same as for 'pre_func'.
'debug'	False	Boolean. If True, prints dicts of definions.
'extra_texts'	[]	List of Dicts defining texts. Defines extra texts. Could be for example:
'isopleth_params'	[{}]	List of Dicts. Defines appearance of isopleths. Could be for example:

TYPE REFERENCE

In the following is a reference for the types used in pyNomo.

5.1 Type 1

Type 1 is three parallel lines that have functional relationship:

$$F_1(u_1) + F_2(u_2) + F_3(u_3) = 0$$

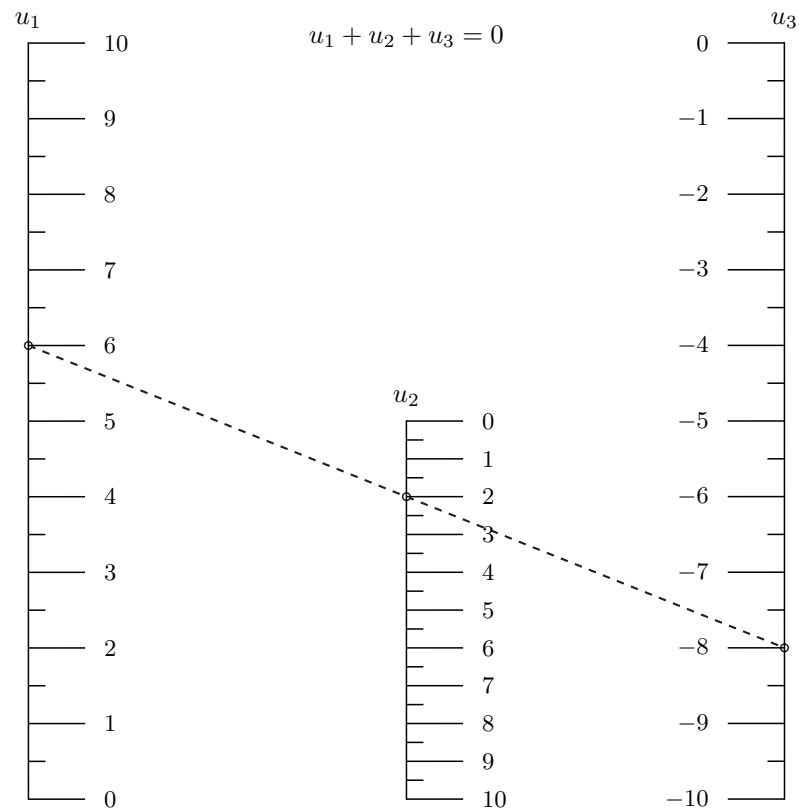
Note, that this kind of function can be transformed to many forms by using type 8 that is a equation given in determinant form. Use of this nomograph is given by the following simple example.

5.1.1 Simple example

This simple example plots nomograph for equation:

$$u_1 + u_2 + u_3 = 0.$$

Generated nomograph



Source code of simple example of type 1

```

1  """
2      ex_type1_nomo_1.py
3
4      Simple nomogram of type 1: F1+F2+F3=0
5  """
6  import sys
7  sys.path.insert(0, "..")
8  #sys.path[:0] = [".."]
9  from pynomo.nomographer import *
10
11  N_params_1={
12      'u_min':0.0,
13      'u_max':10.0,
14      'function':lambda u:u,
15      'title':r'$u_1$',
16      'tick_levels':2,
17      'tick_text_levels':1,
18  }
19
20  N_params_2={
21      'u_min':0.0,
22      'u_max':10.0,
23      'function':lambda u:u,
24      'title':r'$u_2$',
25      'tick_levels':2,
26      'tick_text_levels':1,
27  }
28
29  N_params_3={
30      'u_min':0.0,
31      'u_max':-10.0,
32      'function':lambda u:u,
33      'title':r'$u_3$',

```

```

34     'tick_levels':2,
35     'tick_text_levels':1,
36     }
37
38
39 block_1_params={
40     'block_type':'type_1',
41     'width':10.0,
42     'height':10.0,
43     'f1_params':N_params_1,
44     'f2_params':N_params_2,
45     'f3_params':N_params_3,
46     'isopleth_values':[[6,2,'x']],
47     }
48
49 main_params={
50     'filename':'ex_type1_nomo_1.pdf',
51     'paper_height':10.0,
52     'paper_width':10.0,
53     'block_params':[block_1_params],
54     'transformations':[(('rotate',0.01),('scale paper',))],
55     'title_str':r'$u_1+u_2+u_3=0$',
56     'debug':False,
57     }
58 Nomographer(main_params)

```

5.1.2 Parameters for type 1

Axis parameters

Table 5.1: Specific axis parameters for type 1

parameter key	default value	type, explanation
'function'	—	func(u) . Function in equation For example lambda u: u
'u_min'	—	Float . Minimum value of function variable.
'u_max'	—	Float . Maximum value of function variable.

See *Common axis params* for other parameters.

Block parameters

Table 5.2: Specific block parameters for type 9

parameter	default value	explanation
'block_type'	'type_1'	String. This is type 1 block
'width'	10.0	Float. Block width (to be scaled)
'height'	10.0	Float. Block height (to be scaled)
'f1_params'	–	Axis params Dict. Axis params for function f1
'f2_params'	–	Axis params Dict. Axis params for function f2
'f3_params'	–	Axis params Dict. Axis params for function f3
'mirror_x'	False	Boolean. If x-axis is mirrored
'mirror_y'	False	Boolean. If y-axis is mirrored
'proportion'	1.0	Float. Factor for spacings between lines
'isopleth_values'	[[[]]]	** List of list of isopleth values.** Unknown values are given with strings, e.g. 'x'. An example: <code>[[0.8, 0.1, 'x'], ['x', 0.2, 1.0]]</code>

General parameters

See [List of main params](#) for top level main parameters.

5.2 Type 2

Type 1 is “N” or “Z” nomograph that have functional relationship:

$$F_1(u_1) = F_2(u_2)F_3(u_3)$$

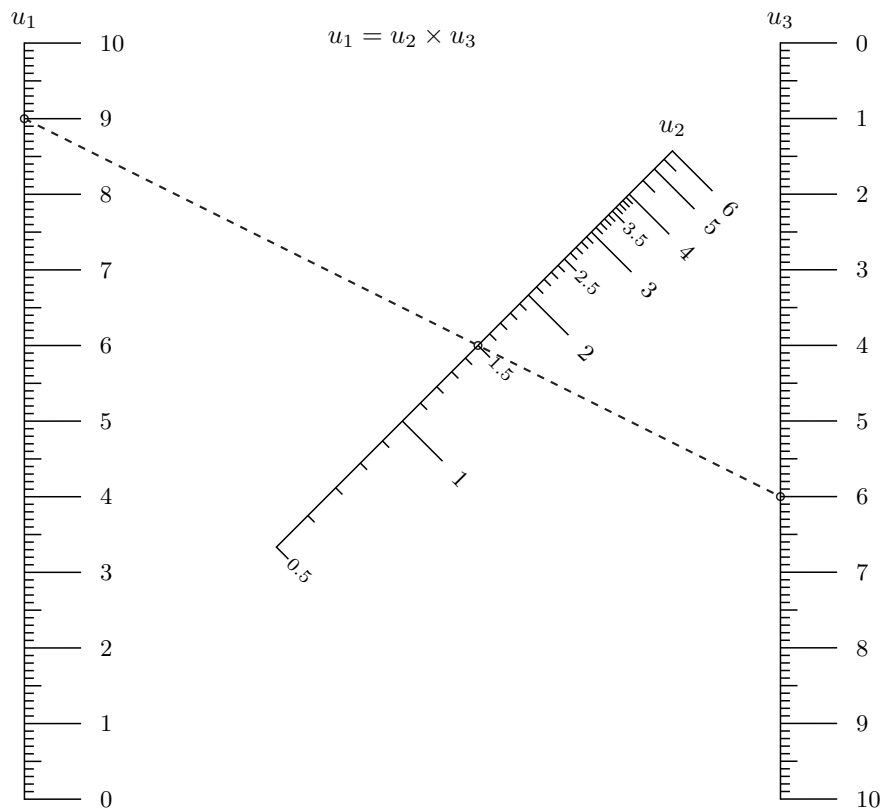
Use of this nomograph is given by the following simple example.

5.2.1 Simple example

This simple example plots nomograph for equation:

$$u_1 = u_2 u_3$$

Generated nomograph



Source code of simple example of type 2

```

1  """
2      ex_type2_nomo_1.py
3
4      Simple nomogram of type 2: F1=F2*F3
5  """
6  import sys
7  sys.path.insert(0, "..")
8  from pynomo.nomographer import *
9
10 N_params_1={
11     'u_min':0.0,
12     'u_max':10.0,
13     'function':lambda u:u,
14     'title':r'$u_1$',
15     'tick_levels':3,
16     'tick_text_levels':1,
17 }
18
19 N_params_2={
20     'u_min':0.5,
21     'u_max':6.0,
22     'function':lambda u:u,
23     'title':r'$u_2$',
24     'tick_levels':3,
25     'tick_text_levels':2,
26     'scale_type':'linear smart',
27 }
28
29 N_params_3={
30     'u_min':0.0,
31     'u_max':10.0,
32     'function':lambda u:u,
33     'title':r'$u_3$',

```

```

34     'tick_levels':3,
35     'tick_text_levels':1,
36     }
37
38
39 block_1_params={
40     'block_type':'type_2',
41     'width':10.0,
42     'height':10.0,
43     'f1_params':N_params_1,
44     'f2_params':N_params_2,
45     'f3_params':N_params_3,
46     'isopleth_values':[[9,1.5,'x']],
47     }
48
49 main_params={
50     'filename':'ex_type2_nomo_1.pdf',
51     'paper_height':10.0,
52     'paper_width':10.0,
53     'block_params':[block_1_params],
54     'transformations':[(('rotate',0.01),('scale paper',))],
55     'title_str':r'$u_1=u_2\times u_3$'
56     }
57 Nomographer(main_params)

```

5.2.2 Parameters for type 2

Axis parameters

Table 5.3: Specific axis parameters for type 2

parameter key	default value	type, explanation
'function'	—	func(u) . Function in equation For example $\lambda u: u$
'u_min'	—	Float . Minimum value of function variable.
'u_max'	—	Float . Maximum value of function variable.

See *Common axis params* for other parameters.

Block parameters

Table 5.4: Specific block parameters for type 2

parameter	default value	explanation
'block_type'	'type_2'	String. This is type 2 block
'width'	10.0	Float. Block width (to be scaled)
'height'	10.0	Float. Block height (to be scaled)
'f1_params'	–	Axis params Dict. Axis params for function f1
'f2_params'	–	Axis params Dict. Axis params for function f2
'f3_params'	–	Axis params Dict. Axis params for function f3
'mirror_x'	False	Boolean. If x-axis is mirrored
'mirror_y'	False	Boolean. If y-axis is mirrored
'proportion'	1.0	Float. Factor for spacings between lines
'isopleth_values'	[[[]]]	** List of list of isopleth values.** Unknown values are given with strings, e.g. 'x'. An example: <code>[[0.8, 0.1, 'x'], ['x', 0.2, 1.0]]</code>

General parameters

See *List of main params* for top level main parameters.

5.3 Type 3

Type 3 has N parallel lines that have functional relationship:

$$F_1(u_1) + F_2(u_2) + \cdots + F_N(u_N) = 0$$

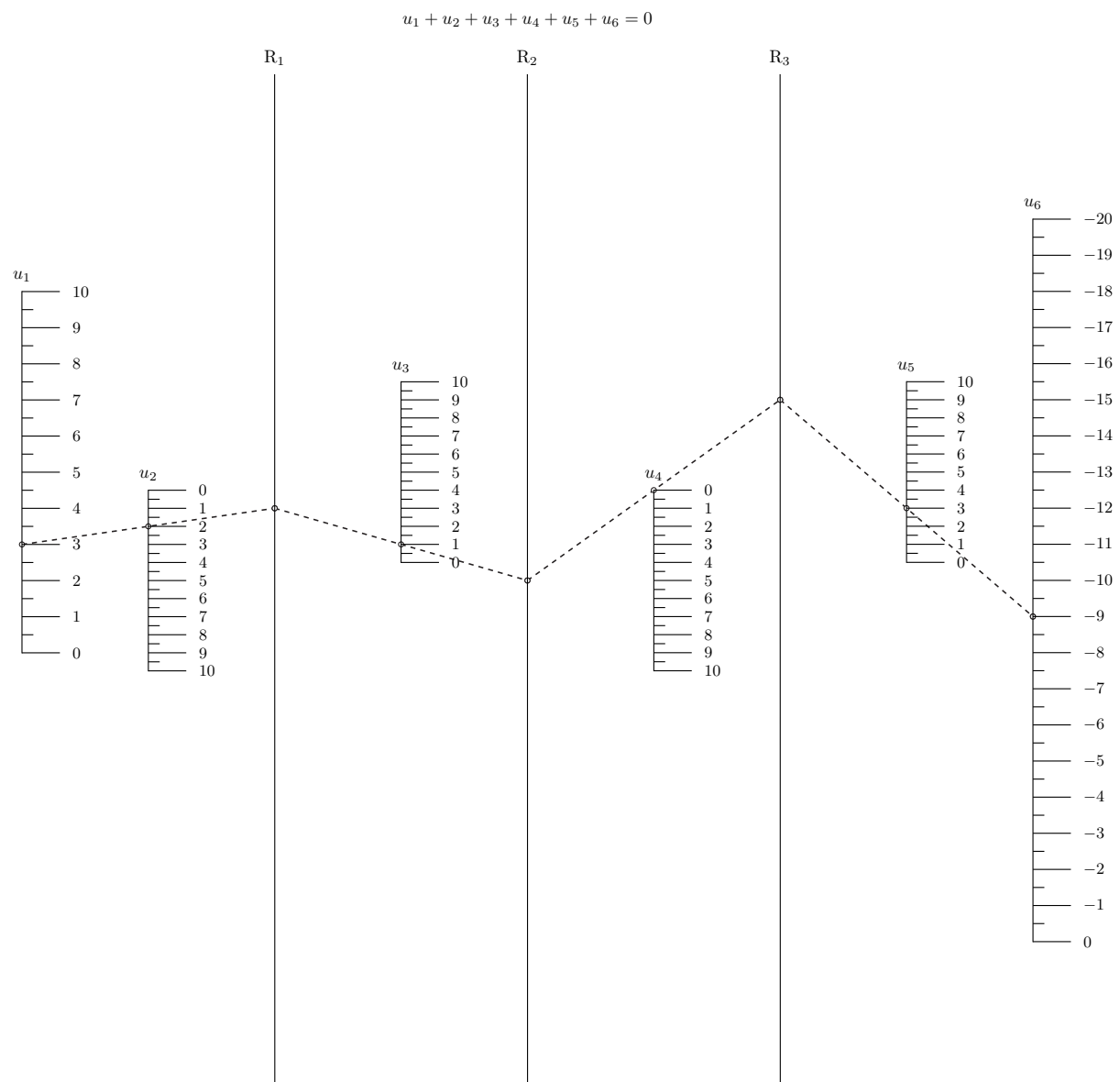
Use of this nomograph is given by the following simple example.

5.3.1 Simple example

This simple example plots nomograph for equation:

$$u_1 + u_2 + u_3 + u_4 + u_5 + u_6 = 0$$

Generated nomograph



Source code of simple example of type 2

```

1 """
2     ex_type3_nomo_1.py
3
4     Simple nomogram of type 3: F1+F2+...+FN=0
5     You should have received a copy of the GNU General Public License
6     along with this program. If not, see <http://www.gnu.org/licenses/>.
7 """
8 import sys
9 sys.path.insert(0, "..")
10 from pynomo.nomographer import *
11
12 N_params_1={
13     'u_min':0.0,
14     'u_max':10.0,
15     'function':lambda u:u,
16     'title':r'$u_1$',
17     'tick_levels':2,
18     'tick_text_levels':1,
19 }
20 N_params_2={

```

```

21     'u_min':0.0,
22     'u_max':10.0,
23     'function':lambda u:u,
24     'title':r'$u_2$',
25     'tick_levels':2,
26     'tick_text_levels':1,
27     }
28 N_params_3={
29     'u_min':0.0,
30     'u_max':10.0,
31     'function':lambda u:u,
32     'title':r'$u_3$',
33     'tick_levels':2,
34     'tick_text_levels':1,
35     }
36 N_params_4={
37     'u_min':0.0,
38     'u_max':10.0,
39     'function':lambda u:u,
40     'title':r'$u_4$',
41     'tick_levels':2,
42     'tick_text_levels':1,
43     }
44 N_params_5={
45     'u_min':0.0,
46     'u_max':10.0,
47     'function':lambda u:u,
48     'title':r'$u_5$',
49     'tick_levels':2,
50     'tick_text_levels':1,
51     }
52 N_params_6={
53     'u_min':-20.0,
54     'u_max':0.0,
55     'function':lambda u:u,
56     'title':r'$u_6$',
57     'tick_levels':2,
58     'tick_text_levels':1,
59     'tick_side':'right',
60     }
61
62 block_1_params={
63     'block_type':'type_3',
64     'width':10.0,
65     'height':10.0,
66     'f_params':[N_params_1,N_params_2,N_params_3,
67                 N_params_4,N_params_5,N_params_6],
68     'isopleth_values':[[3,2,1,0,3,'x']],
69     }
70
71 main_params={
72     'filename':'ex_type3_nomo_1.pdf',
73     'paper_height':20.0,
74     'paper_width':20.0,
75     'block_params':[block_1_params],
76     'transformations':[('rotate',0.01),('scale paper',)],
77     'title_str':r'$u_1+u_2+u_3+u_4+u_5+u_6=0$',
78     'title_y':21.0,
79     }
80 Nomographer(main_params)

```

5.3.2 Parameters for type 3

Axis parameters

Table 5.5: Specific axis parameters for type 3

parameter key	default value	type, explanation
'function'	–	func(u) . Function in equation For example <code>lambda u: u</code>
'u_min'	–	Float . Minimum value of function variable.
'u_max'	–	Float . Maximum value of function variable.

See *Common axis params* for other parameters.

Block parameters

Table 5.6: Specific block parameters for type 3

parameter	default value	explanation
'block_type'	'type_3'	String . This is type 3 block
'width'	10.0	Float . Block width (to be scaled)
'height'	10.0	Float . Block height (to be scaled)
'f_params'	–	List of Axis params Dict . List of Axis params.
'mirror_x'	False	Boolean . If x-axis is mirrored
'mirror_y'	False	Boolean . If y-axis is mirrored
'reference_padding'	0.2	Float . Additional length to reference axes.
'reference_titles'	[]	Array of Strings . List of reference line titles. For example <code>['\$R_1\$', '\$R_2\$', '\$R_3\$']</code> .
'reference_color'	<code>color.rgb.black</code>	Color . Color of reference lines.
'isopleth_values'	[[[]]]	** List of list of isopleth values.** Unknown values are given with strings, e.g. 'x'. An example: <code>[[0.8, 'x', 0.7, 7.0, 9.0], [0.7, 0.8, 'x', 5.0, 4.44]]</code>

General parameters

See *List of main params* for top level main parameters.

5.4 Type 4

Type 4 is proportion nomograph that have functional relationship:

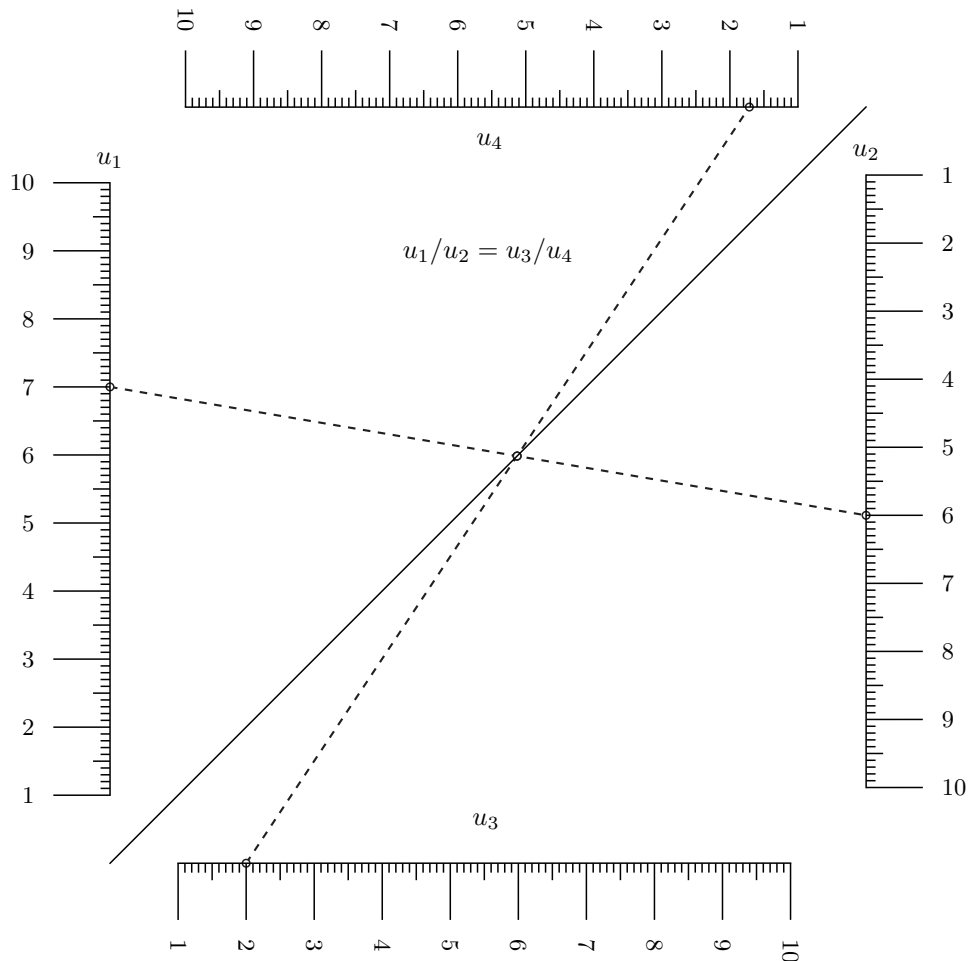
$$\frac{F_1(u_1)}{F_2(u_2)} = \frac{F_3(u_3)}{F_4(u_4)}$$

5.4.1 Simple example

This simple example plots nomograph for equation:

$$u_1/u_2 = u_3/u_4$$

Generated nomograph



Source code of simple example of type 4

```

1  """
2      ex_type4_nomo_1.py
3
4      Simple nomogram of type 4: F1/F2=F3/F4
5  """
6  import sys
7  sys.path.insert(0, "..")
8  from pynomo.nomographer import *
9
10 N_params_1={
11     'u_min':1.0,
12     'u_max':10.0,
13     'function':lambda u:u,
14     'title':r'$u_1$',
15     'tick_levels':3,
16     'tick_text_levels':1,
17     'tick_side':'left',

```

```
18         }
19 N_params_2={
20     'u_min':1.0,
21     'u_max':10.0,
22     'function':lambda u:u,
23     'title':r'$u_2$',
24     'tick_levels':3,
25     'tick_text_levels':1,
26     'tick_side':'right',
27     }
28 N_params_3={
29     'u_min':1.0,
30     'u_max':10.0,
31     'function':lambda u:u,
32     'title':r'$u_3$',
33     'tick_levels':3,
34     'tick_text_levels':1,
35     'tick_side':'right',
36     'title_draw_center':True,
37     'title_opposite_tick':False,
38     }
39 N_params_4={
40     'u_min':1.0,
41     'u_max':10.0,
42     'function':lambda u:u,
43     'title':r'$u_4$',
44     'tick_levels':3,
45     'tick_text_levels':1,
46     'tick_side':'left',
47     'title_draw_center':True,
48     'title_opposite_tick':False,
49     }
50
51 block_1_params={
52     'block_type':'type_4',
53     'f1_params':N_params_1,
54     'f2_params':N_params_2,
55     'f3_params':N_params_3,
56     'f4_params':N_params_4,
57     'isopleth_values':[[7,6,2,'x']],
58     }
59
60 main_params={
61     'filename':'ex_type4_nomo_1.pdf',
62     'paper_height':10.0,
63     'paper_width':10.0,
64     'block_params':[block_1_params],
65     'transformations':[( 'rotate',0.01),('scale paper',)],
66     'title_str':r'$u_1/u_2=u_3/u_4$',
67     'title_y':8.0,
68     }
69 Nomographer(main_params)
```

5.4.2 Parameters for type 4

Axis parameters

Table 5.7: Specific axis parameters for type 4

parameter key	default value	type, explanation
'function'	–	func(u) . Function in equation For example <code>lambda u: u</code>
'u_min'	–	Float . Minimum value of function variable.
'u_max'	–	Float . Maximum value of function variable.

See *Common axis params* for other parameters.

Block parameters

Table 5.8: Specific block parameters for type 4

parameter	default value	explanation
'block_type'	'type_4'	String . This is type 4 block
'width'	10.0	Float . Block width (to be scaled)
'height'	10.0	Float . Block height (to be scaled)
'f1_params'	–	Axis params Dict . Axis params for function f1
'f2_params'	–	Axis params Dict . Axis params for function f2
'f3_params'	–	Axis params Dict . Axis params for function f3
'f4_params'	–	Axis params Dict . Axis params for function f4
'mirror_x'	False	Boolean . If x-axis is mirrored
'mirror_y'	False	Boolean . If y-axis is mirrored
'padding'	0.9	Float . How much axis extend w.r.t. width/height.
'float_axis'	'F1 or F2'	Strings . If given 'F1 or F2', then scaling is according to them, otherwise according to F3 and F4.
'reference_color'	<code>color.rgb.black</code>	Color . Color of reference lines.
'isopleth_values'	[[[]]]	** List of list of isopleth values.** Unknown values are given with strings, e.g. 'x'. An example: <code>[[0.8, 'x', 0.7, 0.5], [0.7, 0.8, 'x', 0.3]]</code>

General parameters

See *List of main params* for top level main parameters.

5.5 Type 5

Type 5 is graphing block that has functional relationship:

$$F_1(u) = F_2(x, v).$$

This type of block is used commonly in nomographs that have an equation in form

$$f_a(a_1, a_2, a_3, \dots) = f_b(u, v)$$

and `math:f_b(u,v)` cannot be represented as line-nomograph. Typically equation above is written as pair of equations:

$$f_a(a_1, a_2, a_3, \dots) = x$$

and

$$f_b(u, v) = x.$$

This equation is written in form

$$F_1(u) = F_2(x, v).$$

in order to construct this contour block. In reality block consists of horizontal lines:

$$F_1(u) = y$$

and contour lines

$$F_2(x, v) = y,$$

where x and y are the coordinates of canvas. Coordinate x is reference with name `wd` in block parameters and it holds

$$x = f_{wd}(wd).$$

Note: Type 5 is a very complex (say stupid) way to make basic graphs. In the future versions of pynomo a more simple way for graphs will be implemented.

5.5.1 Simple example

In the following example

$$F_1(u) = u$$

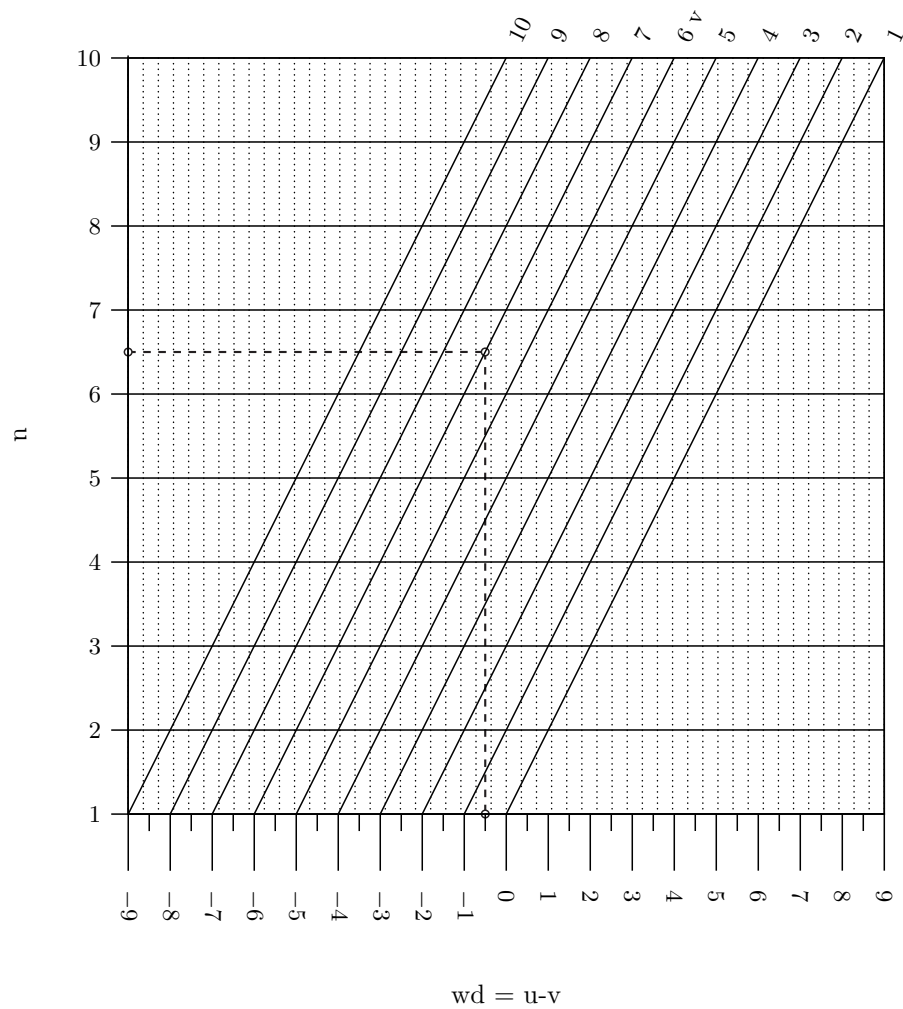
and

$$F_2(wd, v) = wd + v.$$

Thus the original equation is

$$wd = u - v.$$

Generated nomograph



Source code of simple example of type 5

```

1  """
2      ex_type5_nomo_1.py
3
4      Simple nomogram of type 5.
5  """
6  import sys
7  sys.path.insert(0, "..")
8  from pynomo.nomographer import *
9
10
11
12  block_params={
13      'block_type': 'type_5',
14      'u_func': lambda u:u,
15      'v_func': lambda x,v:x+v,
16      'u_values':[1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0],
17      'v_values':[1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0],
18      'wd_tick_levels':2,
19      'wd_tick_text_levels':1,
20      'wd_tick_side':'right',
21      'wd_title':'wd = u-v',
22      'u_title':'u',
23      'v_title':'v',
24      'wd_title_opposite_tick':True,
25      'wd_title_distance_center':2.5,

```

```

26     'isopleth_values':[[6.5,7,'x']],
27 }
28
29
30 main_params={
31     'filename':'ex_type5_nomo_1.pdf',
32     'paper_height':10.0,
33     'paper_width':10.0,
34     'block_params':[block_params],
35     'transformations':[('rotate',0.01),('scale paper',)]
36 }
37
38 Nomographer(main_params)

```

5.5.2 Parameters for type 5

Axis parameters

No specific axis parameters. Everything is defined in block.

Block parameters

Table 5.9: Specific block parameters for type 4

parameter	default value	explanation
'block_type'	'type_5'	String. This is type 5 block.
'width'	10.0	Float. Block width (to be scaled)
'height'	10.0	Float. Block height (to be scaled)
'mirror_x'	False	Boolean. If x-axis is mirrored
'mirror_y'	False	Boolean. If y-axis is mirrored
'u_func'	–	func(u). u function. For example <code>lambda u:u</code>
'v_func'	–	func(u,v). v function. For example <code>lambda x,v: x+v</code>
'wd_func'	–	func(wd). wd func. For example <code>lambda wd: wd</code>
'wd_func_inv'	–	func(wd). Inverse of wd-func. For example <code>lambda wd: wd</code>
'u_values'	–	List of Floats. List of plotted u values. For example <code>[1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0]</code> .
'u_tag'	'none'	String. To align blocks w.r.t each other along axes with same tag.
'u_title'	' '	String. Axis title.
'u_title_x_shift'	0.0	Float. Title shift in x-direction.
'u_title_y_shift'	0.25	Float. Title shift in y-direction.

Continued on next page

Table 5.9 – continued from previous page

parameter	default value	explanation
'u_scale_type'	'linear'	String. Scale type. Can be 'linear': linear scale. 'log': logarithmic scale. 'smart linear': linear scale with equal spacings. 'smart log': logarithmic scale with equal spacings, can also have negative values. 'manual point': Points and corresponding text positions are given manually in 'manual axis data'. No line is drawn. 'manual line': Ticks and corresponding text positions are given manually in 'manual axis data'.
'u_tick_levels'	4	Integer. How many levels (minor, minor-minor, etc.) of ticks are drawn. Largest effect to 'linear' scale.
'u_tick_text_levels'	'3'	Integer. How many levels (minor, minor-minor, etc.) of texts are drawn. Largest effect to 'linear' scale.
'u_tick_side'	'right'	String. Tick and text side in final paper. Can be: 'right' or 'left'
'u_reference'	False	Boolean. If axis is treated as reference line that is a turning point.
'u_reference_padding'	'0.2'	Float. Fraction of reference line over other lines.
'u_manual_axis_data'	{}	Dict. Manually set tick/point positions and text positions. Could be for example: <code>{1:'1', 3.14:r'\$\pi\$', 5:'5', 7:'seven', 10:'10'}</code>
'u_title_draw_center'	False	Boolean. Title is drawn to center of line.
'u_title_distance_center'	'type_9'	String. To double-align blocks w.r.t each other along axes with same tag.
'u_title_opposite_tick'	True	Boolean. Title in opposite direction w.r.t ticks.
'u_align_func'	lambda u:u	func(u). function to align different scales.
'u_align_x_offset'	0.0	Float. If axis is aligned with other axis, this value x offsets final scale.
'u_align_y_offset'	0.0	Float. If axis is aligned with other axis, this value y offsets final scale.
'u_text_format'	r'\$%4.4g\$ '	String. Format for numbers in scale.
'u_extra_params'	[{}, ...]	Array of Dicts. List of dictionary of params to be drawn additionally.
'u_text_distance_#'	x.x	Float. where # = 0, 1, 2, 3 or 4. Distance of text from scale line. Number corresponds to the level, where 0 is the major tick and 4 is the most minor ticks.
'u_grid_length_#'	x.x	Float. where # = 0, 1, 2, 3 or 4. Length of the tick. Number corresponds to the level, where 0 is the major tick and 4 is the most minor ticks.

Continued on next page

Table 5.9 – continued from previous page

parameter	default value	explanation
'u_text_size_#'	x.x	Float. where #=0,1,2,3 or 4. Text size. For example: text.size.small, text.size.scriptsize or text.size.tiny. Number corresponds to the level, where 0 is the major tick and 4 is the most minor ticks.
'u_text_size_log_#'	x.x	Float. where #=0,1 or 2. Text size. For example: text.size.small, text.size.scriptsize or text.size.tiny. Number corresponds to the level, where 0 is the major tick and 2 is the most minor ticks.
'u_full_angle'	False	Boolean. If true, text can be upside down, otherwise +- 90 degrees from horizontal. Good for example for full circle scales.
'u_extra_angle'	0.0	Boolean. Title is drawn to center of line.
'u_text_horizontal_align'	center	Boolean. Aligns tick text horizontally to center. Good when text rotated 90 degrees.
'u_axis_color'	color.rgb.black	Color. Color of axis.
'u_text_color'	color.rgb.black	Color. Color of tick texts.
'v_values'	–	List of Floats. List of plotted v values. For example [1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0]’.
'v_title'	''	String. Axis title.
'v_title_draw_center'	False	Boolean. Title is drawn to center of line.
'v_title_distance_center'	type_9	String. To double-align blocks w.r.t each other along axes with same tag.
'v_title_opposite_tick'	True	Boolean. Title in opposite direction w.r.t ticks.
'wd_tag'	'none'	String. To align blocks w.r.t each other along axes with same tag.
'wd_title'	''	String. Axis title.
'wd_title_x_shift'	0.0	Float. Title shift in x-direction.
'wd_title_y_shift'	0.25	Float. Title shift in y-direction.
'wd_scale_type'	'linear'	String. Scale type. Can be 'linear': linear scale. 'log': logarithmic scale. 'smart linear': linear scale with equal spacings. 'smart log': logarithmic scale with equal spacings, can also have negative values. 'manual point': Points and corresponding text positions are given manually in 'manual axis data'. No line is drawn. 'manual line': Ticks and corresponding text positions are given manually in 'manual axis data'.

Continued on next page

Table 5.9 – continued from previous page

parameter	default value	explanation
'wd_tick_levels'	4	Integer. How many levels (minor, minor-minor, etc.) of ticks are drawn. Largest effect to 'linear' scale.
'wd_tick_text_levels'	'3'	Integer. How many levels (minor, minor-minor, etc.) of texts are drawn. Largest effect to 'linear' scale.
'wd_tick_side'	'right'	String. Tick and text side in final paper. Can be: 'right' or 'left'
'wd_reference'	False	Boolean. If axis is treated as reference line that is a turning point.
'wd_reference_padding'	'0.2'	Float. Fraction of reference line over other lines.
'wd_manual_axis_data'	{}	Dict. Manually set tick/point positions and text positions. Could be for example: <code>{1:'1', 3.14:r'\$\pi\$', 5:'5', 7:'seven', 10:'10'}</code>
'wd_title_draw_center'	False	Boolean. Title is drawn to center of line.
'wd_title_distance_center_type_9'		String. To double-align blocks w.r.t each other along axes with same tag.
'wd_title_opposite_ticks'	True	Boolean. Title in opposite direction w.r.t ticks.
'wd_align_func'	lambda u:u	func(u). function to align different scales.
'wd_align_x_offset'	0.0	Float. If axis is aligned with other axis, this value x offsets final scale.
'wd_align_y_offset'	0.0	Float. If axis is aligned with other axis, this value y offsets final scale.
'wd_text_format'	r'\${4.4g}\$ '	String. Format for numbers in scale.
'wd_extra_params'	[{}, ...]	Array of Dicts. List of dictionary of params to be drawn additionally.
'wd_text_distance_#'	x.x	Float. where # = 0, 1, 2, 3 or 4. Distance of text from scale line. Number corresponds to the level, where 0 is the major tick and 4 is the most minor ticks.
'wd_grid_length_#'	x.x	Float. where # = 0, 1, 2, 3 or 4. Length of the tick. Number corresponds to the level, where 0 is the major tick and 4 is the most minor ticks.
'wd_text_size_#'	x.x	Float. where # = 0, 1, 2, 3 or 4. Text size. For example: <code>text.size.small</code> , <code>text.size.scriptsize</code> or <code>text.size.tiny</code> . Number corresponds to the level, where 0 is the major tick and 4 is the most minor ticks.

Continued on next page

Table 5.9 – continued from previous page

parameter	default value	explanation
'wd_text_size_log_#'	x.x	Float. where #=0,1 or 2. Text size. For example: text.size.small, text.size.scriptsize or text.size.tiny. Number corresponds to the level, where 0 is the major tick and 2 is the most minor ticks.
'wd_full_angle'	False	Boolean. If true, text can be upside down, otherwise +- 90 degrees from horizontal. Good for example for full circle scales.
'wd_extra_angle'	0.0	Boolean. Title is drawn to center of line.
'wd_text_horizontal_align'	center	Boolean. Aligns tick text horizontally to center. Good when text rotated 90 degrees.
'wd_axis_color'	color.rgb.black	Color. Color of axis.
'wd_text_color'	color.rgb.black	Color. Color of tick texts.
'isopleth_values'	[[[]]]	** List of list of isopleth values.** Unknown values are given with strings, e.g. 'x'. An example: <code>[[0.8, 'x', 0.7], [0.7, 0.8, 'x']]</code>

General parameters

See *List of main params* for top level main parameters.

5.6 Type 6

Type 6 is ladder nomograph:

$$u = u.$$

In practice this means that if one axis has for example y-position as

$$y = f_1(u)$$

and it was desirable to have

$$y = f_2(u)$$

in order to connect blocks together, one uses ladder to make the transformation.

Note: Ladders are not beautiful and should be used only when no other solution exist.

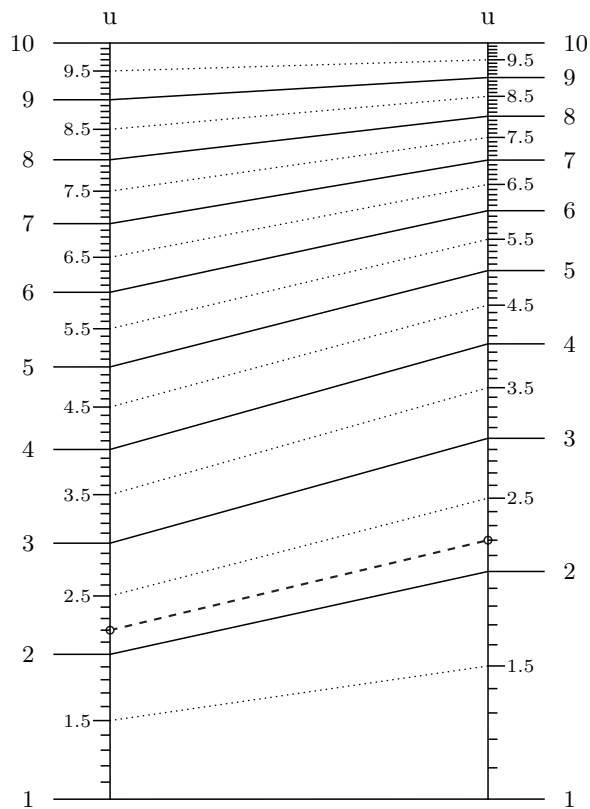
5.6.1 Simple example

This simple example plots nomograph for equation:

$$u = u,$$

where linear scale is converted to a logarithmic scale.

Generated nomograph



Source code of simple example of type6

```

1  """
2      ex_type6_nomo_1.py
3
4      Simple nomogram of type 6.
5  """
6  import sys
7  sys.path.insert(0, "..")
8  from pynomo.nomographer import *
9
10 N_params_1={
11     'u_min':1.0,
12     'u_max':10.0,
13     'function':lambda u:u**0.5,
14     'title':'u',
15     'tick_levels':3,
16     'tick_text_levels':2,
17     'tick_side':'left',
18 }
19
20 N_params_2={
21     'u_min':1.0,
22     'u_max':10.0,
23     'function':lambda u:log(u),
24     'title':'u',
25     'tick_levels':3,
26     'tick_text_levels':2,
27 }
28
29 block_params={
30     'block_type':'type_6',
31     'f1_params':N_params_1,
32     'f2_params':N_params_2,
33     'width':5.0,

```

```
34         'height':10.0,
35         'isopleth_values':[[2.2,'x']],
36         #'curve_const':0.01
37     }
38
39 main_params={
40     'filename':'ex_type6_nomo_1.pdf',
41     'paper_height':10.0,
42     'paper_width':5.0,
43     'block_params':[block_params],
44     'transformations':[('rotate',0.01),('scale paper',)]
45 }
46
47 Nomographer(main_params)
```

5.6.2 Parameters for type 6

Axis parameters

Table 5.10: Specific axis parameters for type 6

parameter key	default value	type, explanation
'function'	—	func(u). Function in equation For example $\lambda u: u$
'u_min'	—	Float. Minimum value of function variable.
'u_max'	—	Float. Maximum value of function variable.

See *Common axis params* for other parameters.

Block parameters

Table 5.11: Specific block parameters for type 6

parameter	default value	explanation
'block_type'	'type_6'	String. This is type 6 block.
'type'	'parallel'	String. Can be either 'parallel' or 'orthogonal'.
'x_empty'	0.2	Float. If orthogonal, how much fractional space before start of x-axis.
'y_empty'	0.2	Float. If orthogonal, how much fractional space before start of y-axis.
'curve_const'	0.0	Float. Sets the lenght of angle of Bezier curve. low value = straigh line, high value = curved line.
'width'	10.0	Float. Block width (to be scaled)
'height'	10.0	Float. Block height (to be scaled)
'f1_params'	–	Axis params Dict. Axis params for function f1
'f2_params'	–	Axis params Dict. Axis params for function f2
'mirror_x'	False	Boolean. If x-axis is mirrored
'mirror_y'	False	Boolean. If y-axis is mirrored
'ladder_color'	color.rgb.black	Color. Ladder color.
'isopleth_values'	[[[]]]	** List of list of isopleth values.** Unknown values are given with strings, e.g. 'x'. An example: <code>[[0.8, 'x'], [0.7, 'x']]</code>

General parameters

See [List of main params](#) for top level main parameters.

5.7 Type 7

Type 7 is “angle” nomograph that has functional relationship:

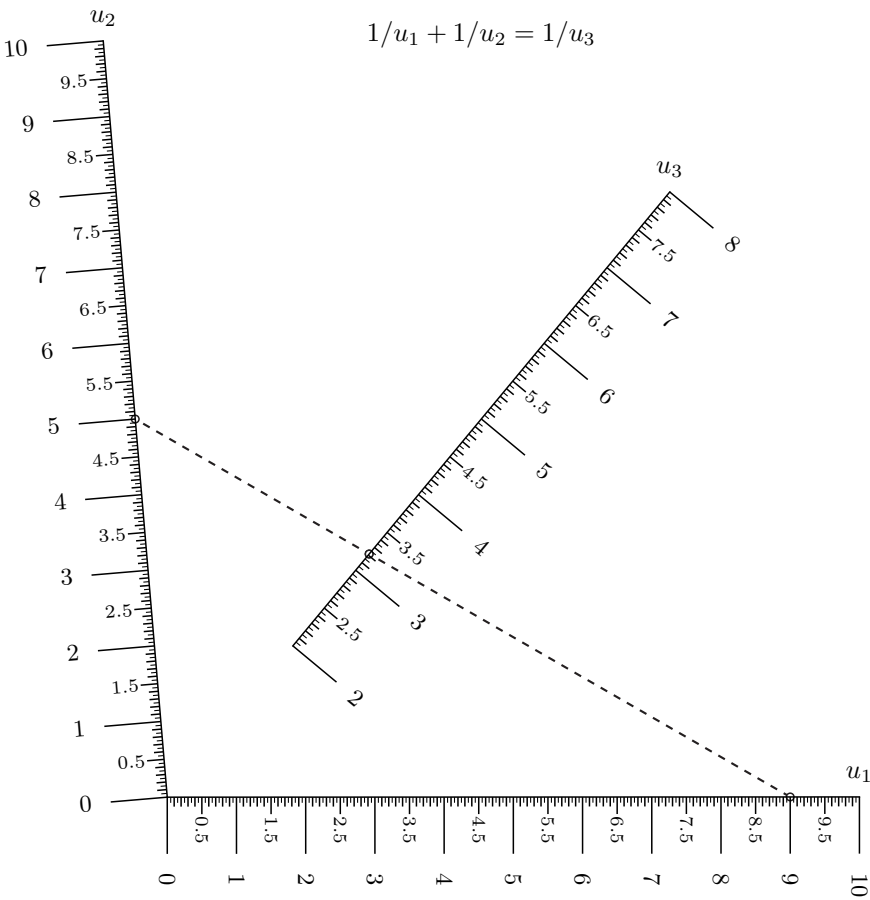
$$\frac{1}{F_1(u_1)} + \frac{1}{F_2(u_2)} = \frac{1}{F_3(u_3)}$$

5.7.1 Simple example

This simple example plots nomograph for equation:

$$1/u_1 + 1/u_2 = 1/u_3$$

Generated nomograph



Source code of simple example of type 2

5.7.2 Parameters for type 7

Axis parameters

Table 5.12: Specific axis parameters for type 7

parameter key	default value	type, explanation
'function'	—	func(u) . Function in equation For example <code>lambda u: u</code>
'u_min'	—	Float . Minimum value of function variable.
'u_max'	—	Float . Maximum value of function variable.

See *Common axis params* for other parameters.

Block parameters

Table 5.13: Specific block parameters for type 7

parameter	default value	explanation
'block_type'	'type_4'	String. This is type 7 block
'width'	10.0	Float. Block width (to be scaled)
'height'	10.0	Float. Block height (to be scaled)
'f1_params'	–	Axis params Dict. Axis params for function f1
'f2_params'	–	Axis params Dict. Axis params for function f2
'f3_params'	–	Axis params Dict. Axis params for function f3
'mirror_x'	False	Boolean. If x-axis is mirrored
'mirror_y'	False	Boolean. If y-axis is mirrored
'angle_u'	45.0	Float. Angle between u1 and u3. Note: later transformations may alter the angle.
'angle_v'	45.0	Float. Angle between u2 and u3. Note: later transformations may alter the angle.
'isopleth_values'	[[[]]]	** List of list of isopleth values.** Unknown values are given with strings, e.g. 'x'. An example: <code>[[0.8, 'x', 0.7], [0.7, 0.8, 'x']]</code>

General parameters

See *List of main params* for top level main parameters.

5.8 Type 8

Type 8 is single nomograph:

$$y = F(u)$$

or

$$x = F_x(u),$$

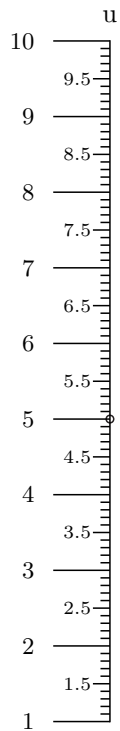
$$y = F_y(u).$$

x and y are coordinates of canvas. Often this block is used for construction of dual-scales to existing scales.

5.8.1 Simple example

This simple example plots single vertical scale.

Generated nomograph



Source code of simple example of type 8

```
1 """
2     ex_type8_nomo_1.py
3
4     Simple nomogram of type 8.
5 """
6 import sys
7 sys.path.insert(0, "..")
8 from pynomo.nomographer import *
9
10 N_params_1={
11     'u_min':1.0,
12     'u_max':10.0,
13     'function':lambda u:u,
14     'title':'u',
15     'tick_levels':3,
16     'tick_text_levels':2,
17     'tick_side':'left',
18 }
19
20 block_params={
21     'block_type':'type_8',
22     'f_params':N_params_1,
23     'width':5.0,
24     'height':10.0,
25     'isopleth_values':[[5]]
26 }
27
28 main_params={
29     'filename':'ex_type8_nomo_1.pdf',
30     'paper_height':10.0,
31     'paper_width':5.0,
32     'block_params':[block_params],
33     'transformations':[]
34 }
35
36 Nomographer(main_params)
```

5.8.2 Parameters for type 8

Axis parameters

Table 5.14: Specific axis parameters for type 8

parameter key	default value	type, explanation
'function'	–	func(u) . Function in equation. For example <code>lambda u: u</code> .
'u_min'	–	Float . Minimum value of function variable.
'u_max'	–	Float . Maximum value of function variable.
'function_x'	–	func(u) . x-position in function. If used 'function_y' must be defined. For example <code>lambda u: u</code> .
'function_y'	–	func(u) . y-position in function. If used 'function_x' must be defined. Overrides 'function'. For example <code>lambda u: u</code> .

See [Common axis params](#) for other parameters.

Block parameters

Table 5.15: Specific block parameters for type 8

parameter	default value	explanation
'block_type'	'type_8'	String . This is type 8 block
'width'	10.0	Float . Block width (to be scaled)
'height'	10.0	Float . Block height (to be scaled)
'f1_params'	–	Axis params Dict . Axis params for function f1
'f2_params'	–	Axis params Dict . Axis params for function f2
'f3_params'	–	Axis params Dict . Axis params for function f3
'f4_params'	–	Axis params Dict . Axis params for function f4
'mirror_x'	False	Boolean . If x-axis is mirrored
'mirror_y'	False	Boolean . If y-axis is mirrored
'padding'	0.9	Float . How much axis extend w.r.t. width/height.
'float_axis'	'F1 or F2'	Strings . If given 'F1 or F2', then scaling is according to them, otherwise according to F3 and F4.
'reference_color'	<code>color.rgb.black</code>	Color . Color of reference lines.
'isopleth_values'	<code>[[[]]]</code>	** List of list of isopleth values.** Unknown values are given with strings, e.g. 'x'. An example: <code>[[0.8, 'x', 0.7, 0.5], [0.7, 0.8, 'x', 0.3]]</code>

General parameters

See [List of main params](#) for top level main parameters.

5.9 Type 9

Type 9 is “general determinant” nomograph that has functional relationship:

$$\begin{vmatrix} F_1(u_1[, v_1]) & G_1(u_1[, v_1]) & H_1(u_1[, v_1]) \\ F_2(u_2[, v_2]) & G_2(u_2[, v_2]) & H_2(u_2[, v_2]) \\ F_3(u_3[, v_3]) & G_3(u_3[, v_3]) & H_3(u_3[, v_3]) \end{vmatrix} = 0.$$

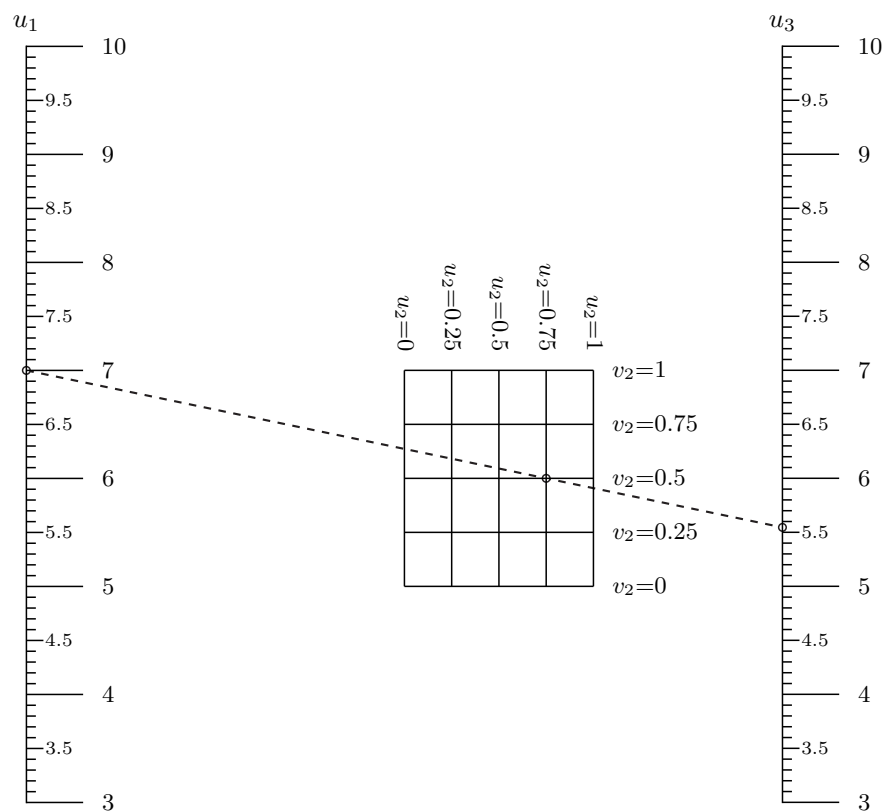
This is the basic building block for line nomographs. Notation $u[, v]$ is to be understood such that if v is defined, a grid is constructed for the row, otherwise a normal scale with variable u .

5.9.1 Simple example

This simple example plots nomograph for equation in determinant form:

$$\begin{vmatrix} 0 & u_1 & 1 \\ u_2 + 2 & 2v_2 + 5 & 1 \\ 4 & u_3 & 1 \end{vmatrix} = 0$$

Generated nomograph



Source code of simple example of type 9

```

1  """
2      ex_type9_nomo_1.py
3
4      Simple nomogram of type 9: determinant
5  """
6  import sys
7  sys.path.insert(0, "..")
8  from pynomo.nomographer import *
9
10 N_params_1={
11     'u_min':3.0,
12     'u_max':10.0,
13     'f':lambda u:0,
14     'g':lambda u:u,
15     'h':lambda u:1.0,
16     'title':r'$u_1$',
17     'scale_type':'linear',
18     'tick_levels':3,
19     'tick_text_levels':2,
20     'grid':False}
21
22 N_params_2={
23     'u_min':0.0, # for alignment
24     'u_max':1.0, # for alignment
25     'f_grid':lambda u,v:u+2.0,
26     'g_grid':lambda u,v:2*v+5.0,
27     'h_grid':lambda u,v:1.0,
28     'u_start':0.0,
29     'u_stop':1.0,
30     'v_start':0.0,
31     'v_stop':1.0,
32     'u_values':[0.0,0.25,0.5,0.75,1.0],
33     'v_values':[0.0,0.25,0.5,0.75,1.0],
34     'grid':True,
35     'text_prefix_u':r'$u_2$=',
36     'text_prefix_v':r'$v_2$=',
37 }
38
39 N_params_3={
40     'u_min':3.0,
41     'u_max':10.0,
42     'f':lambda u:4.0,
43     'g':lambda u:u,
44     'h':lambda u:1.0,
45     'title':r'$u_3$',
46     'scale_type':'linear',
47     'tick_levels':3,
48     'tick_text_levels':2,
49     'grid':False
50 }
51
52 block_params={
53     'block_type':'type_9',
54     'f1_params':N_params_1,
55     'f2_params':N_params_2,
56     'f3_params':N_params_3,
57     'transform_ini':False,
58     'isopleth_values':[[7,[0.75,0.5], 'x']]
59 }
60
61 main_params={
62     'filename':'ex_type9_nomo_1.pdf',
63     'paper_height':10.0,
64     'paper_width':10.0,
65     'block_params':[block_params],
66     'transformations':[('rotate',0.01),('scale paper',)]
67 }
68 Nomographer(main_params)

```

5.9.2 Parameters for type 9

Axis parameters

Table 5.16: Specific axis parameters for type 9 grid axis

parameter key	default value	type, explanation
'grid'	–	Bool. True because this is grid.
'f'	–	func(u,v). F function in determinant. For example <code>lambda u,v:u+v</code>
'g'	–	func(u,v). G function in determinant. For example <code>lambda u,v:u+v</code>
'h'	–	func(u,v). H function in determinant. For example <code>lambda u,v:u+v</code>
'u_start'	–	u start when drawing $v=\text{const}$ line
'u_stop'	–	u stop when drawing $v=\text{const}$ line
'v_start'	–	v start when drawing $u=\text{const}$ line
'v_stop'	–	v stop when drawing $u=\text{const}$ line
'u_values'	–	List of grid lines $u=\text{const}$. For example <code>[0.0,0.25,0.5,0.75,1.0]</code>
'v_values'	–	List of grid lines $v=\text{const}$. For example <code>“[0.0,0.25,0.5,0.75,1.0]”</code>
'text_prefix_u'	–	Text prefix for u before value
'text_prefix_v'	–	Text prefix for v before value
'v_texts_u_start'	False	If v-texts are in u start side
'v_texts_u_stop'	True	If v-texts are in u stop side
'u_texts_v_start'	False	If u-texts are in v start side
'u_texts_v_stop'	True	If u-texts are in v stop side
'u_line_color'	<code>color.rgb.black</code>	Color. u line color
'v_line_color'	<code>color.rgb.black</code>	Color. v line color
'u_text_color'	<code>color.rgb.black</code>	Color. u text color
'v_text_color'	<code>color.rgb.black</code>	Color. v text color
'text_distance'	0.25	Float. Text distance
'circles'	False	Boolean. If marker circles to crossings
'extra_params'	–	List of Dicts. List of params to be drawn.

See *Common axis params* for other parameters.

Block parameters

Table 5.17: Specific block parameters for type 9

parameter	default value	explanation
'block_type'	'type_9'	String. This is type 9 block
'width'	10.0	Float. Block width (to be scaled)
'height'	10.0	Float. Block height (to be scaled)
'f1_params'	–	Axis params Dict. Axis params for function f1
'f2_params'	–	Axis params Dict. Axis params for function f2
'f3_params'	–	Axis params Dict. Axis params for function f3
'mirror_x'	False	Boolean. If x-axis is mirrored
'mirror_y'	False	Boolean. If y-axis is mirrored
'transform_ini'	False	Boolean. If row 1 and row 3 end and start are to be transformed to be in rectangle corners. If True, be sure that 'u_min_trafo' and 'u_max_trafo' are defined.
'isopleth_values'	[[[]]]	** List of list of isopleth values.** Grid values are given with tuple (a,b) and are not solved. Unknown values are given with strings, e.g. 'x'. An example: <code>[[0.8, (0.1, 0.2), 'x'], ['x', (0.1, 0.2), 1.0]]</code>

General parameters

See [List of main params](#) for top level main parameters.

5.10 Type 10

Type 10 is nomograph that has one curved line. It has functional relationship:

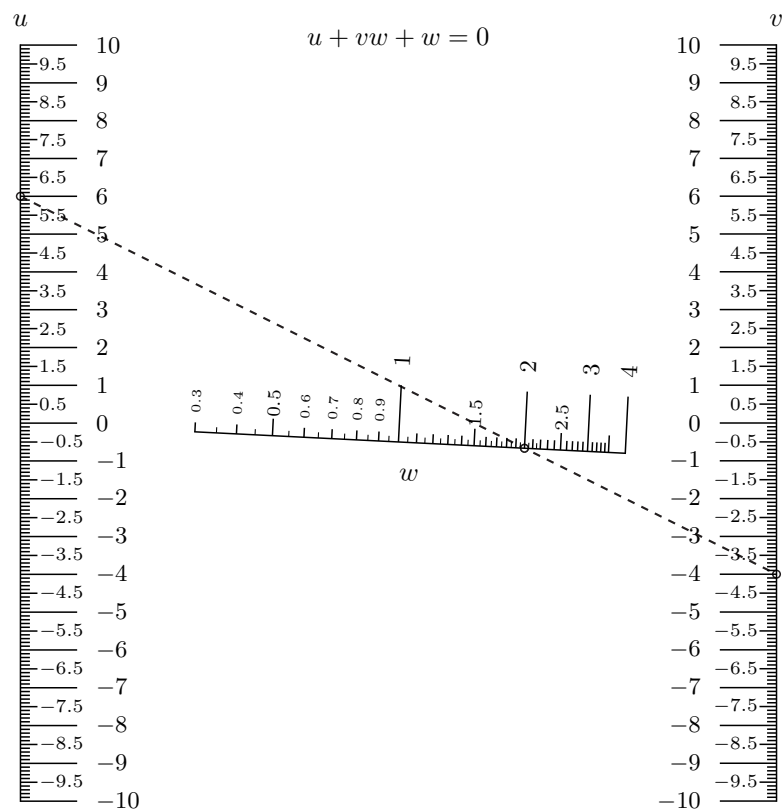
$$F_1(u) + F_2(v)F_3(w) + F_4(w) = 0.$$

5.10.1 Simple example

This simple example plots nomograph for equation:

$$u + vw + w = 0.$$

Generated nomograph



Source code of simple example of type 10

```

1  """
2      ex_type10_nomo_1.py
3
4      Simple nomogram of type 7: F1(u)+F2(v)*F3(w)+F4(w)=0
5      along with this program. If not, see <http://www.gnu.org/licenses/>.
6  """
7  import sys
8  sys.path.insert(0, "..")
9  from pynomo.nomographer import *
10
11  N_params_1={
12      'u_min':-10.0,
13      'u_max':10.0,
14      'function':lambda u:u,
15      'title':r'$u$',
16      'tick_levels':3,
17      'tick_text_levels':2,
18  }
19
20  N_params_2={
21      'u_min':-10.0,
22      'u_max':10.0,
23      'function':lambda u:u,
24      'title':r'$v$',
25      'tick_levels':3,
26      'tick_text_levels':2,
27      'tick_side':'left',
28  }
29
30  N_params_3={
31      'u_min':0.3,
32      'u_max':4.0,
33      'function_3':lambda u:u,

```

```

34     'function_4': lambda u: u,
35     'title': r'$w$',
36     'tick_levels': 4,
37     'tick_text_levels': 3,
38     'scale_type': 'linear smart',
39     'title_draw_center': True,
40     }
41
42 block_1_params={
43     'block_type': 'type_10',
44     'width': 10.0,
45     'height': 10.0,
46     'f1_params': N_params_1,
47     'f2_params': N_params_2,
48     'f3_params': N_params_3,
49     'isopleth_values': [[6, -4, 'x']]
50 }
51
52 main_params={
53     'filename': 'ex_type10_nomo_1.pdf',
54     'paper_height': 10.0,
55     'paper_width': 10.0,
56     'block_params': [block_1_params],
57     'transformations': [('rotate', 0.01), ('scale paper',)],
58     'title_str': r'$u+vw+w=0$'
59 }
60 Nomographer(main_params)

```

5.10.2 Parameters for type 10

Axis parameters

Table 5.18: Specific axis parameters for type 10

parameter key	default value	type, explanation
'function'	—	func(u) . Function in the equation for F_1 and F_2 . For example <i>"lambda u : u"</i>
'function_3'	—	func(u) . Function in the equation for F_3 . For example <i>lambda u: u</i>
'function_4'	—	func(u) . Function in the equation for F_4 . For example <i>lambda u: u</i>
'u_min'	—	Float . Minimum value of function variable.
'u_max'	—	Float . Maximum value of function variable.

See *Common axis params* for other parameters.

Block parameters

Table 5.19: Specific block parameters for type 10

parameter	default value	explanation
'block_type'	'type_10'	String. This is type 10 block
'width'	10.0	Float. Block width (to be scaled)
'height'	10.0	Float. Block height (to be scaled)
'f1_params'	–	Axis params Dict. Axis params for function f1
'f2_params'	–	Axis params Dict. Axis params for function f2
'f3_params'	–	Axis params Dict. Axis params for function f3
'f4_params'	–	Axis params Dict. Axis params for function f4
'mirror_x'	False	Boolean. If x-axis is mirrored
'mirror_y'	False	Boolean. If y-axis is mirrored
'padding'	0.9	Float. How much axis extend w.r.t. width/height.
'float_axis'	'F1 or F2'	Strings. If given 'F1 or F2', then scaling is according to them, otherwise according to F3 and F4.
'reference_color'	color.rgb.black	Color. Color of reference lines.
'isopleth_values'	[[[]]]	** List of list of isopleth values.** Unknown values are given with strings, e.g. 'x'. An example: <code>[[0.8, 'x', 0.7, 0.5], [0.7, 0.8, 'x', 0.3]]</code>

General parameters

See *List of main params* for top level main parameters.

EXAMPLES

In the following are listed examples to show nomographs possibilities. Also is explained the background for the cases and underlying math for the nomograph construction. Source code shows the implementation.

6.1 Example: Amortized loan calculator

6.1.1 Theory and background

This approach of constructing an amortized loan calculator is similar to one in Ref. [1]_ Equation for amortized loan [2]_ is:

$$\frac{a}{A} = \frac{\frac{p}{100 \times 12}}{1 - \frac{1}{(1 + \frac{p}{100 \times 12})^{12n}}},$$

where A is the amount of loan, a is monthly payment amount, p interest rate per year (monthly interest rate is taken as $p/12$)¹ and n is number of years for payment.

This equation of four variables is probably impossible to present with line and grid nomographs. For this reason a “Type 5” contour nomogram is constructed of the right hand side of the equation and left hand equation is just N-nomogram (Type 2). The two equations for nomogram construction are:

$$x = \frac{a}{A}$$

and

$$x = \frac{\frac{p}{100 \times 12}}{1 - \frac{1}{(1 + \frac{p}{100 \times 12})^{12n}}}.$$

In practice x is the x-coordinate of the canvas where nomogram is constructed.

Right hand side of equation

By defining coordinates x and y :

$$x = \frac{\frac{p}{100 \times 12}}{1 - \frac{1}{(1 + \frac{p}{100 \times 12})^{12n}}},$$

$y = 12n$, we may solve y in terms of x and n :

¹ http://en.wikipedia.org/wiki/Annual_percentage_rate#Does_not_represent_the_total_cost_of_borrowing

$$y = \frac{\log\left(\frac{x}{x - \frac{p}{100 \times 12}}\right)}{\log\left(1 + \frac{p}{100 \times 12}\right)}$$

The previous two equations are of correct form

$$y = f_1(v)$$

and

$$y = f_2(x, u)$$

for type 5 nomogram. For compressing time axis (y -axis), we transform $y \rightarrow \log y$ and find

$$y = \log\left(\frac{\log\left(\frac{x}{x - \frac{p}{100 \times 12}}\right)}{\log\left(1 + \frac{p}{100 \times 12}\right)}\right)$$

$$y = \log(12n).$$

Left hand side of equation

Left hand side of equation

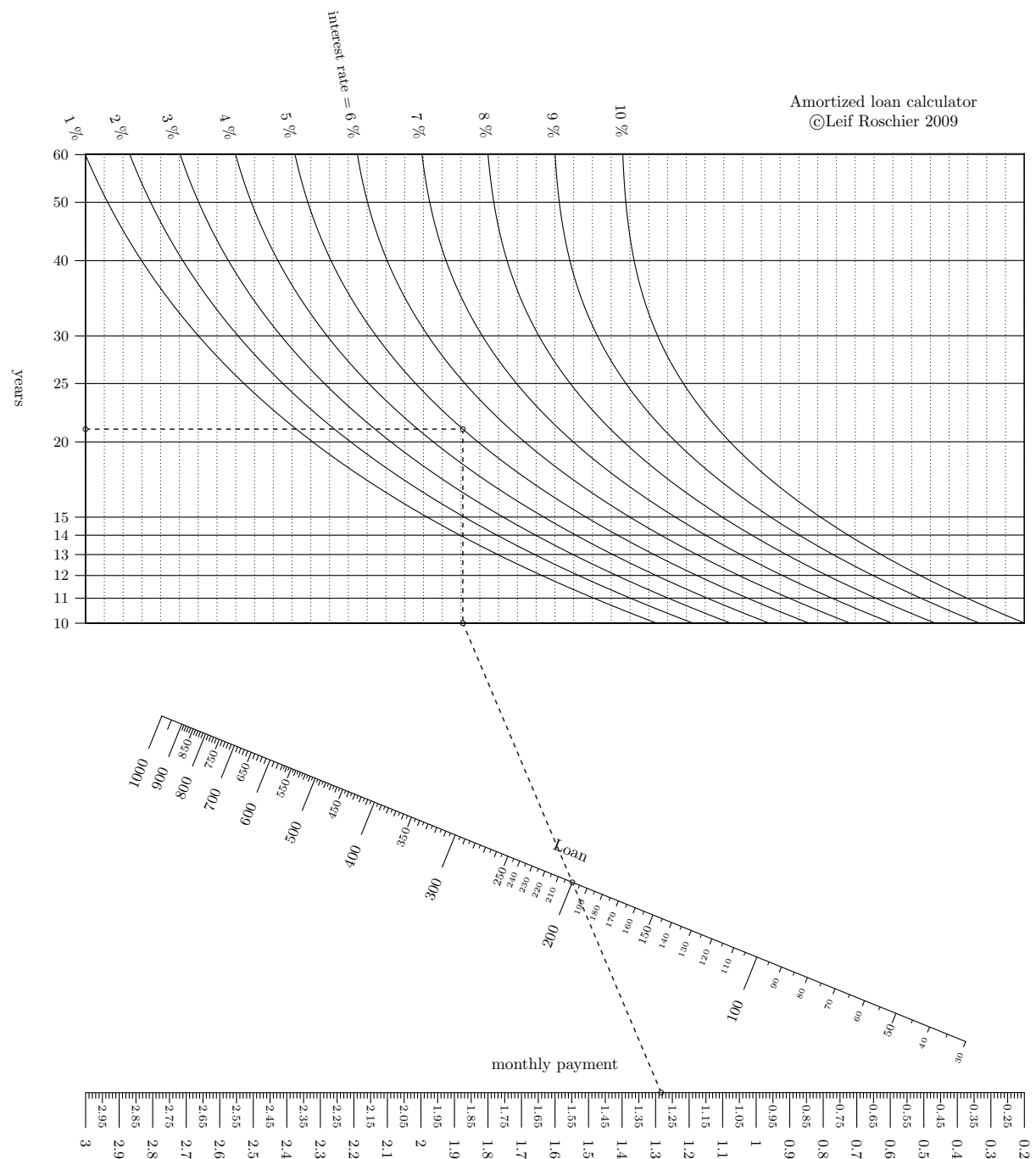
$$x = \frac{a}{A}$$

is just N-nomogram

$$F_1(u_1) = F_2(u_2)F_3(u_3)$$

References

6.1.2 Generated nomograph



6.1.3 Source code

```

1  """
2      ex_amortized_loan.py
3
4      Amortized loan calculator
5  """
6  import sys
7  sys.path.insert(0, "..")
8  from pynomo.nomographer import *
9
10 # Type 5 contour

```

```

11 def f1(x,u):
12     return log(log(x/(x-u/(100.0*12.0)))/log(1+u/(100.0*12.0)))
13
14 block1_params={
15     'width':10.0,
16     'height':5.0,
17     'block_type':'type_5',
18     'u_func':lambda u:log(u*12.0),
19     'v_func':f1,
20     'u_values':[10.0,11.0,12.0,13.0,14.0,15.0,20.0,25.0,30.0,40.0,50.0,60.0],
21     'v_values':[1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0],
22     'wd_tag':'A',
23     'u_title':'years',
24     'v_title':r'interest rate = ',
25     'u_text_format':r"$%3.0f$ ",
26     'v_text_format':r"$%3.0f$ \%",
27     'isopleth_values':[[21,5,'x']]
28 }
29
30 # this is non-obvious trick to find bottom edge coordinates of the grid in order
31 # to align it with N nomogram
32 block1_dummy=Nomo_Block_Type_5(mirror_x=False)
33 block1_dummy.define_block(block1_params)
34 block1_dummy.set_block()
35
36 # Let's define the N-nomogram
37 N_params_3={
38     'u_min':block1_dummy.grid_box.params_wd['u_min'],
39     'u_max':block1_dummy.grid_box.params_wd['u_max'],
40     'function':lambda u:u,
41     'title':'',
42     'tag':'A',
43     'tick_side':'right',
44     'tick_levels':2,
45     'tick_text_levels':2,
46     'reference':False,
47     'tick_levels':0,
48     'tick_text_levels':0,
49     'title_draw_center':True
50 }
51 N_params_2={
52     'u_min':30.0,
53     'u_max':1000.0,
54     'function':lambda u:u,
55     'title':'Loan',
56     'tag':'none',
57     'tick_side':'left',
58     'tick_levels':4,
59     'tick_text_levels':3,
60     'title_draw_center':True,
61     #'text_format':r"$%3.0f$ ",
62     'scale_type':'linear smart',
63 }
64 N_params_1={
65     'u_min':0.2,
66     'u_max':3.0,
67     'function':lambda u:u,
68     'title':'monthly payment',
69     'tag':'none',
70     'tick_side':'right',
71     'tick_levels':3,
72     'tick_text_levels':2,
73     'title_draw_center':True
74 }
75
76 block2_params={
77     'block_type':'type_2',
78     'width':10.0,
79     'height':20.0,
80     'f1_params':N_params_1,
81     'f2_params':N_params_2,
82     'f3_params':N_params_3,

```



```

83     'isopleth_values':[['x',200,'x']]
84     }
85
86 main_params={
87     'filename':'amortized_loan.pdf',
88     'paper_height':20.0,
89     'paper_width':20.0,
90     'block_params':[block_1_params,block_2_params],
91     'transformations':[('rotate',0.01),('scale paper',)],
92     'title_str':r'Amortized loan calculator \copyright Leif Roschier 2009',
93     'title_x': 17,
94     'title_y': 21,
95     'title_box_width': 5
96     }
97 Nomographer(main_params)

```

6.2 Example Photography exposure

6.2.1 Theory and background

This example illustrates how exposure in photography depends on factors: latitude, time of day, day of year, weather, composition. It relates these to camera settings: film speed (e.g. ISO 100), aperture and shutter speed. The mathematical approach and model is taken from book written by V. Setälä. [\[1\]](#) This book illustrates the approach as nomographs but they are different compared with the one generated here. Book uses shadow length, but we break shadow length into time, date and latitude via solar zenith angle.

The basic equation in Setälä (pp.492-494) can be extracted and written as

$$FS - L - A - W + C + T = 0 \quad (6.1)$$

where parameters of (6.1) are listed below:

FS	Film speed	DIN value that equals $10 \log(S) + 1$, where S is ISO FILM speed
T	shutter time	$10 \log \left(\frac{t}{1/10} \right)$
A	aperture	$10 \log \left(\frac{N^2}{3.2^2} \right)$
L	shadow length (in steps)	two times (shadow length)/(person length) = $2 \arctan(\phi)$, where ϕ is solar zenith angle.
W	weather	Clear sky, Cumulus clouds: 0, Clear sky: 1, Sun through clouds: 3, Sky light gray: 6, Sky dark gray: 9, Thunder-clouds cover sky: 12
C	Composi-tion	Person under trees: -6, Inside forest : -4, Person in shadow of wall : -1, Person at open place; alley under trees : 2, Buildings; street : 5, Landscape and front matter : 7, Open landscape : 9, Snow landscape and front matter; beach : 11, Snow field; open sea : 13, Clouds : 15

It is to be noted that Setälä has stops ten times base-10 logarithmic. Today we think stops in base-2 logarithmic.

Shadow length

Calculation of shadow length as a function of day of year, time of day and latitude is according to [\[2\]](#). Following equations are used. For fractional year (without time information) we take

$$\gamma = (day - 1 + 0.5)2\pi/365.$$

For time offset (eqtime) we use equation (in minutes)

$$TO = 229.18(0.000075 + 0.001868 \cos(\gamma) - 0.032077 \sin(\gamma) - 0.014615 \cos(2\gamma) - 0.040849 \sin(2\gamma))$$

to calculate that error is below 17 minutes for time axis. We assume that sun is at highest point at noon and this is the error and approximation. We calculate stops in logarithmic scale and in this case we do not need very accurate equations for time. For declination we use equation

and for hour angle

$$ha = (60h + \overline{TO})/4 - 180.$$

Solar zenith angle (ϕ), latitude (LAT), declination (D) and hour angle (ha) are connected with equation:

$$\cos(\phi) = \sin(LAT)\sin(D) + \cos(LAT)\cos(D)\cos(ha).$$

This is in our desired form as a function of hour (h), day (day), latitude (LAT), solar zenith angle (ϕ):

$$\cos(\phi) = \sin(LAT)\sin(D(\gamma(day))) + \cos(LAT)\cos(D(\gamma(day)))\cos(ha(h)).$$

In practice illuminance of flat surface on earth depends on solar zenith angle as $\cos(\phi)$. Setälä uses shadow length that is easily measurable, but scales incorrectly, as value is proportional to $\tan(\phi)$. Also Setälä sums linear value with logarithmic ones as a practical approximation. To correct these assumptions, here we assume that values for shadow length 1 and 10 for Setälä are reasonable, and an equation that scales logarithmically is found:

$$L = 0.33766 - 13.656 \log_{10}(\cos(\phi))$$

that gives $L = 1$ for $\phi = 26.565 = \arctan(1/2)$ and $L = 10$ for $\phi = 78.69 = \arctan(10/2)$.

6.2.2 Construction of the nomograph

The presented equation is the following:

$$FS - \{0.33766 - 13.656 \log_{10}[\sin(LAT)\sin(D(\gamma(day))) + \cos(LAT)\cos(D(\gamma(day)))\cos(ha(h))]\} \\ - A - W + C + T = 0.$$

In order to construct the nomograph, we split the equation into four blocks and an additional block to present values as EV100.

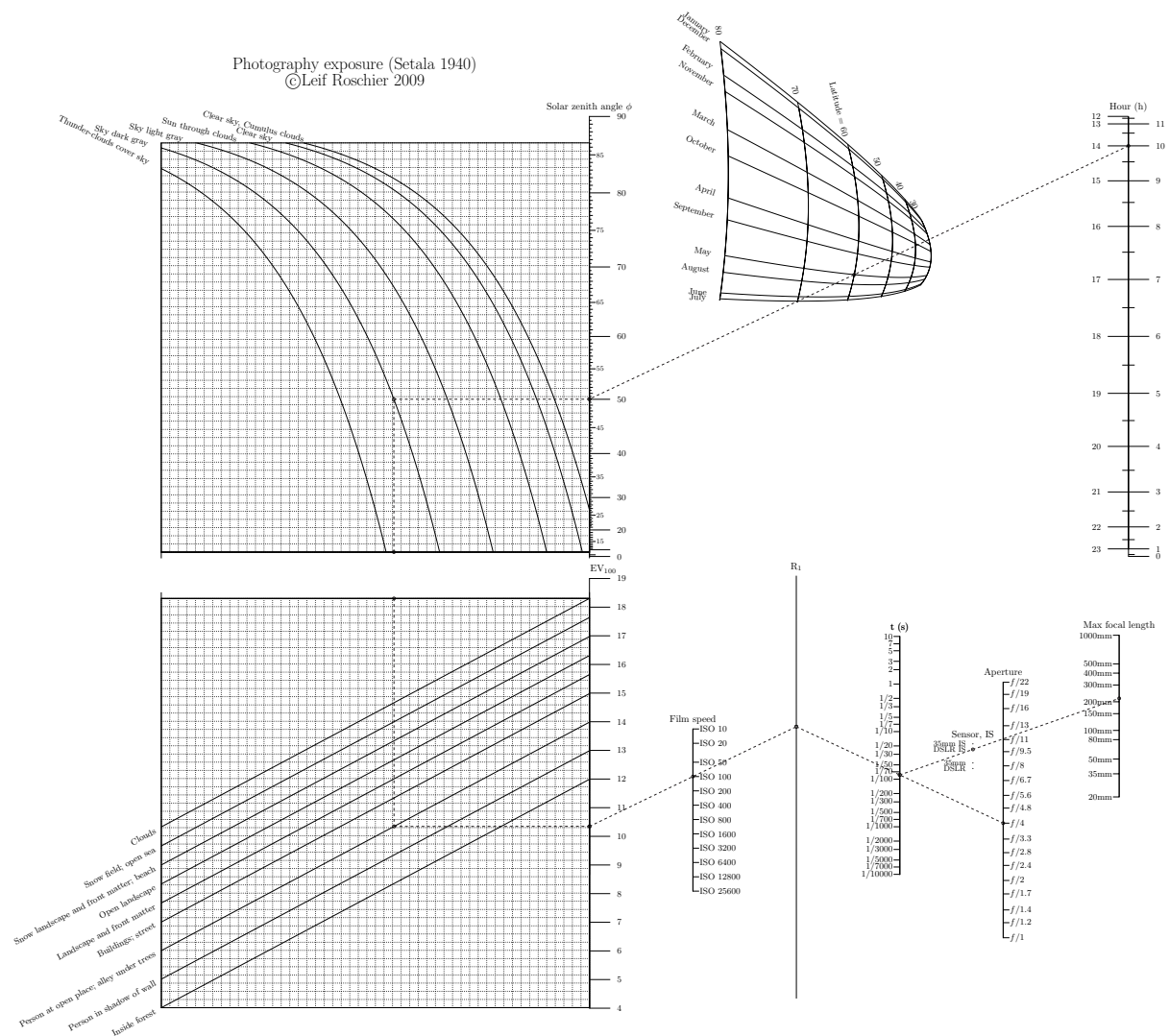
Table 6.1: Main equation split into blocks for the nomograph.

Explanation	Type
$x_1 \equiv \cos(\phi) = \sin(LAT) \sin(D(\gamma(day))) + \cos(LAT) \cos(D(\gamma(day))) \cos(ha(h))$ <p>formed into determinant:</p> $\begin{vmatrix} 0 & \cos(\phi) & 1 \\ \frac{\cos(LAT) \cos(D(\gamma(day)))}{1 + (\cos(LAT) \cos(D(\gamma(day))))} & \frac{\sin(LAT) \sin(D(\gamma(day)))}{1 + (\cos(LAT) \cos(D(\gamma(day))))} & 1 \\ 1 & -\cos(ha(h)) & 1 \end{vmatrix} = 0$	Type 9
$C_1 \equiv L + W = 0.006918 - 13.656 \log_{10}(x_1) + W$ <p>split into two equations for contour construction:</p> $y_1 = C_1$ $y_1 = 0.006918 - 13.656 \log_{10}(x_1) + W$	Type 5
$C_2 \equiv L + W + C = C_1 + C$ <p>split into two equations for contour construction:</p> $y_2 = C_2$ $y_2 = C_1 + C$	Type 5
$C_2 = FS - A + T$ <p>equals</p> $C_2 - (10 \log_{10}(S) + 1.0) + 10 \log_{10} \left(\frac{N^2}{3.2^2} \right) - 10 \log_{10} \left(\frac{1/t_i}{1/10} \right) = 0,$ <p>where</p> $t_i \equiv 1/t$ <p>is inverse shutter time.</p>	Type 3
Continued on next page	

Table 6.1 – continued from previous page

Explanation	Type
<p>Additional EV100 scale by using relation</p> $C_2 = (-EV_{100} + 13.654)/0.3322$	Type 8
<p>Maximum focal length calculator according to equation</p> $t_i/f = FL$ <p>written as</p> $-10 \log_{10} \left(\frac{1/t_i}{1/10} \right) - 10 \log_{10} \left(\frac{f}{10} \right) - 10 \log_{10} (FL) = 0$ <p>in order to align correctly with previous equation. The values for the factor f are: DSLR (3/2), 35mm (1), DSLR image stabilization (3/8) and 35mm image stabilization (1/8).</p>	Type 1

6.2.3 Generated nomograph



6.2.4 Source code

```

1  """
2      ex_photo_exposure.py
3
4      Photograph exposure.
5  """
6  import sys
7  sys.path.insert(0, "..")
8  from pynomo.nomographer import *
9  """
10 functions for solartime taken from solareqns.pdf from
11 http://www.srrb.noaa.gov/highlights/sunrise/solareqns.PDF
12 """
13 # fractional year
14 def gamma(day):
15     return 2*pi/365.0*(day-1+0.5)
16 # equation of time
17 def eq_time(day):
18     gamma0=gamma(day)
19     return 229.18*(0.000075+0.001868*cos(gamma0)-0.032077*sin(gamma0)\
20         -0.014615*cos(2*gamma0)-0.040849*sin(2*gamma0))
21
22 # mean correction, with constant correction we make less than 17 minutes error
23 # in time axis
24 temp_a=arange(0,365.0,0.1)

```

```

25 temp_b=eq_time(temp_a)
26 correction=mean(temp_b) # this is 0.0171885 minutes
27 #print "maximum time errors: %g %g"%(min(temp_b),max(temp_b))
28
29 # declination
30 def eq_declination(day):
31     g0=gamma(day)
32     return 0.006918-0.399912*cos(g0)+0.070257*sin(g0)-0.006758*cos(2*g0)\
33         +0.000907*sin(2*g0)-0.002697*cos(3*g0)+0.00148*sin(3*g0)
34
35 def f1(dummy):
36     return 0.0
37 def g1(fii):
38     return cos(fii*pi/180.0)
39
40 def f2(lat,day):
41     dec=eq_declination(day)
42     return (cos(lat*pi/180.0)*cos(dec))/(1.0+(cos(lat*pi/180.0)*cos(dec)))
43 def g2(lat,day):
44     dec=eq_declination(day) # in radians
45     return (sin(lat*pi/180.0)*sin(dec))/(1.0+(cos(lat*pi/180.0)*cos(dec)))
46
47 def f3(dummy):
48     return 1
49 def g3(h):
50     hr=(h*60.0+correction)/4.0-180.0
51     return -1.0*cos(hr*pi/180.0)
52
53 days_in_month = (31,28,31,30,31,30,31,31,30,31,30,31)
54 times1=[]
55 for idx in range(0,12):
56     times1.append(sum(days_in_month[0:idx])+1)
57
58
59 #times=linspace(0,350,10)
60 #times=arange(0.0,360.0,10.0, dtype=double).tolist()
61 time_titles=['January','February','March','April','May','June',
62             'July','August','September','October','November','December']
63
64 phi_params={
65     'u_min':0.0,
66     'u_max':90.0,
67     'u_min_trafo':0.0,
68     'u_max_trafo':90.0,
69     'f':f1,
70     'g':g1,
71     'h':lambda u:1.0,
72     'title':r'Solar zenith angle $\phi$',
73     'title_x_shift':0.0,
74     'title_y_shift':0.25,
75     'scale_type':'linear smart',
76     'tick_levels':4,
77     'tick_text_levels':2,
78     'tick_side':'right',
79     'tag':'phi',
80     'grid':False,
81     # 'extra_params':[{'u_min':20.0,
82     #                  'u_max':90.0,
83     #                  'tick_levels':4,
84     #                  'tick_text_levels':2,
85     #                  }]
86 }
87
88 time_params={
89     'u_min':0.0,
90     'u_max':23.0,
91     'u_min_trafo':0.0,
92     'u_max_trafo':12.0,
93     'f':f3,
94     'g':g3,
95     'h':lambda u:1.0,
96     'title':r'Hour (h)',

```

```

97     'title_x_shift':0.0,
98     'title_y_shift':0.25,
99     'scale_type':'linear',
100    'tick_levels':2,
101    'tick_text_levels':1,
102    'tick_side':'right',
103    'tag':'none',
104    'grid':False,
105    }
106
107    lat_day_params={
108        'ID':'none', # to identify the axis
109        'tag':'none', # for aligning block wrt others
110        'title':'Grid',
111        'title_x_shift':0.0,
112        'title_y_shift':0.25,
113        'title_distance_center':0.5,
114        'title_opposite_tick':True,
115        'u_min':20.0, # for alignment
116        'u_max':80.0, # for alignment
117        'f_grid':f2,
118        'g_grid':g2,
119        'h_grid':lambda u,v:1.0,
120        'u_start':30.0,
121        'u_stop':80.0,
122        'v_start':times1[0], # day
123        'v_stop':times1[-1],
124        'u_values':[30.0,40.0,50.0,60.0,70.0,80.0],
125        'u_texts':['30','40','50','Latitude = 60','70','80'],
126        'v_values':times1,
127        'v_texts':time_titles,
128        'grid':True,
129        'text_prefix_u':r'',
130        'text_prefix_v':r'',
131        'text_distance':0.5,
132        'v_texts_u_start':False,
133        'v_texts_u_stop':True,
134        'u_texts_v_start':False,
135        'u_texts_v_stop':True,
136    }
137
138    block_params={
139        'block_type':'type_9',
140        'f1_params':phi_params,
141        'f2_params':lat_day_params,
142        'f3_params':time_params,
143        'transform_ini':True,
144        'isopleth_values':[['x',[60,times1[4]],14.0]],
145    }
146
147    # limiting functions are to avoid NaN in contour construction that uses optimization
148    def limit_xx(x):
149        x1=x
150        # if x1>1.0:
151        #     x1=1.0
152        # if x1<-1.0:
153        #     x1=-1.0
154        return x1
155
156    def limit_x(x):
157        x1=x
158        # if not x1>0.0:
159        #     x1=0.0001
160        return x1
161
162    const_A=0.33766
163    const_B=-13.656
164
165    block_params_weather={
166        'block_type':'type_5',
167        'u_func':lambda u:u,
168        'v_func':lambda x,v:const_A+const_B*log10(limit_x(x))+v,

```

```

169 # 'u_values': [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 15.0, 20, 25],
170 'u_values': [1.0, 25.0],
171 'u_manual_axis_data': {1.0: '',
172                        25.0: ''},
173 'v_values': [0.0, 1.0, 3.0, 6.0, 9.0, 12.0],
174 'v_manual_axis_data': {0.0: ['Clear sky, Cumulus clouds', {'x_corr': 0.5,
175                                                            'y_corr': 0.0,
176                                                            'draw_line': False}],
177                        1.0: 'Clear sky',
178                        3.0: 'Sun through clouds',
179                        6.0: 'Sky light gray',
180                        9.0: 'Sky dark gray',
181                        12.0: 'Thunder-clouds cover sky',
182                        },
183 'v_text_distance': 0.5,
184 'wd_tick_levels': 0,
185 'wd_tick_text_levels': 0,
186 'wd_tick_side': 'right',
187 'wd_title': '',
188 'manual_x_scale': True,
189 'x_min': 0.06,
190 'x_max': 0.99,
191 'u_title': '',
192 'v_title': '',
193 'wd_title_opposite_tick': True,
194 'wd_title_distance_center': 2.5,
195 'wd_align_func': lambda L: acos(limit_xx(10.0**((L-const_A)/const_B))) * 180.0/pi, # phi as L
196 'wd_func': lambda L: 10.0**((L-const_A)/const_B), # x as L
197 'wd_func_inv': lambda x: const_A + const_B * log10(x), # L as x
198 'wd_tag': 'phi',
199 'mirror_y': True,
200 'mirror_x': False,
201 'width': 10.0,
202 'height': 10.0,
203 'u_scale_opposite': True,
204 'u_tag': 'AA',
205 'horizontal_guides': True,
206 'isopleth_values': [['x', 9.0, 'x']],
207 }
208
209 block_params_scene = {
210     'block_type': 'type_5',
211     'u_func': lambda u: u,
212     'v_func': lambda x, v: x + v,
213     'u_values': [1.0, 25.0],
214     'u_manual_axis_data': {1.0: '',
215                            25.0: ''},
216     'u_tag': 'AA',
217     'wd_tag': 'EV',
218     'v_values': [-4.0, -1.0, 2.0, 5.0, 7.0, 9.0, 11.0, 13.0, 15.0],
219     'v_manual_axis_data': {-6.0: 'Person under trees',
220                            -4.0: 'Inside forest',
221                            -1.0: 'Person in shadow of wall',
222                            2.0: 'Person at open place; alley under trees',
223                            5.0: 'Buildings; street',
224                            7.0: 'Landscape and front matter',
225                            9.0: 'Open landscape',
226                            11.0: 'Snow landscape and front matter; beach',
227                            13.0: 'Snow field; open sea',
228                            15.0: 'Clouds',
229                            },
230     'wd_tick_levels': 0,
231     'wd_tick_text_levels': 0,
232     'wd_tick_side': 'right',
233     'wd_title': '',
234     'u_title': '',
235     'v_title': '',
236     'wd_title_opposite_tick': True,
237     'wd_title_distance_center': 2.5,
238     'mirror_x': True,
239     'horizontal_guides': True,
240     'u_align_y_offset': -0.9,

```



```

241     'isopleth_values':[['x',2.0,'x']],
242 }
243
244
245 camera_params_1={
246     'u_min':-10.0,
247     'u_max':15.0,
248     'function':lambda u:u,
249     'title':r'',
250     'tick_levels':0,
251     'tick_text_levels':0,
252     'tag':'EV',
253 }
254 camera_params_2={
255     'u_min':10.0,
256     'u_max':25600.0,
257     'function':lambda S:-(10*log10(S)+1.0),
258     'title':r'Film speed',
259     'manual_axis_data': {10.0:'ISO 10',
260                          20.0:'ISO 20',
261                          40.0:'ISO 40',
262                          50.0:'ISO 50',
263                          100.0:'ISO 100',
264                          200.0:'ISO 200',
265                          400.0:'ISO 400',
266                          800.0:'ISO 800',
267                          1600.0:'ISO 1600',
268                          3200.0:'ISO 3200',
269                          6400.0:'ISO 6400',
270                          12800.0:'ISO 12800',
271                          25600.0:'ISO 25600',
272                      },
273     'scale_type':'manual line'
274 }
275 camera_params_3={
276     'u_min':0.1,
277     'u_max':10000.0,
278     'function':lambda t:-10*log10((1.0/t)/(1.0/10.0))-30,
279     'manual_axis_data': {1/10.0:'10',
280                          1/7.0:'7',
281                          1/5.0:'5',
282                          1/3.0:'3',
283                          1/2.0:'2',
284                          1.0:'1',
285                          2.0:'1/2',
286                          3.0:'1/3',
287                          5.0:'1/5',
288                          7.0:'1/7',
289                          10.0:'1/10',
290                          20.0:'1/20',
291                          30.0:'1/30',
292                          50.0:'1/50',
293                          70.0:'1/70',
294                          100.0:'1/100',
295                          200.0:'1/200',
296                          300.0:'1/300',
297                          500.0:'1/500',
298                          700.0:'1/700',
299                          1000.0:'1/1000',
300                          2000.0:'1/2000',
301                          3000.0:'1/3000',
302                          5000.0:'1/5000',
303                          7000.0:'1/7000',
304                          10000.0:'1/10000',
305                      },
306     'scale_type':'manual line',
307     'title':r't (s)',
308     'text_format':r"1/%3.0f s",
309     'tag':'shutter',
310     'tick_side':'left',
311 }
312

```

```

313 camera_params_4={
314     'u_min':1.0,
315     'u_max':22.0,
316     'function':lambda N:10*log10((N/3.2)**2)+30,
317     'manual_axis_data': {1.0:'f$/1',
318                          1.2:'f$/1.2',
319                          1.4:'f$/1.4',
320                          1.7:'f$/1.7',
321                          2.0:'f$/2',
322                          2.4:'f$/2.4',
323                          2.8:'f$/2.8',
324                          3.3:'f$/3.3',
325                          4.0:'f$/4',
326                          4.8:'f$/4.8',
327                          5.6:'f$/5.6',
328                          6.7:'f$/6.7',
329                          8.0:'f$/8',
330                          9.5:'f$/9.5',
331                          11.0:'f$/11',
332                          13.0:'f$/13',
333                          16.0:'f$/16',
334                          19.0:'f$/19',
335                          22.0:'f$/22',
336                      },
337     'scale_type':'manual line',
338     'title':r'Aperture',
339 }
340
341 block_params_camera={
342     'block_type':'type_3',
343     'width':10.0,
344     'height':10.0,
345     'f_params':[camera_params_1,camera_params_2,camera_params_3,
346                 camera_params_4],
347     'mirror_x':True,
348     'isopleth_values':[['x',100.0,'x',4.0]],
349 }
350
351 def old_EV(EV): # C2(EV100) in wiki
352     return (-EV+13.654)/0.3322
353
354 EV_para={
355     'tag':'EV',
356     'u_min':4.0,
357     'u_max':19.0,
358     'function':lambda u:old_EV(u),
359     'title':r'EV$({100})$',
360     'tick_levels':1,
361     'tick_text_levels':1,
362     'align_func':old_EV,
363     'title_x_shift':0.5,
364     'tick_side':'right',
365 }
366 EV_block={
367     'block_type':'type_8',
368     'f_params':EV_para,
369     'isopleth_values':[['x']],
370 }
371
372 # maximum focal length
373 FL_t_para={
374     'u_min':0.1,
375     'u_max':10000.0,
376     'function':lambda t:-10*log10((1.0/t)/(1.0/10.0))-30,
377     'scale_type':'linear',
378     'tick_levels':0,
379     'tick_text_levels':0,
380     'title':r't (s)',
381     'text_format':r"1/%3.0f s",
382     'tag':'shutter',
383 }
384

```

```

385 FL_factor_params_2={
386     'u_min':1.0/4.0,
387     'u_max':3.0/2.0,
388     'function':lambda factor:-10*log10(factor/10.0)+0,
389     'title':r'Sensor, IS',
390     'scale_type':'manual point',
391     'manual_axis_data': {1.0/(2.0/3.0):'DSLR',
392                         1.0/(1.0):'35mm',
393                         1.0/(8.0/3.0):'DSLR IS',
394                         1.0/(4.0):'35mm IS',
395                     },
396     'tick_side':'left',
397     'text_size_manual':text.size.footnotesize, # pyx directive
398 }
399
400 FL_fl_params={
401     'u_min':20.0,
402     'u_max':1000.0,
403     'function':lambda FL:-10*log10(FL)+30,
404     'title':r'Max focal length',
405     'tick_levels':3,
406     'tick_text_levels':2,
407     'tick_side':'left',
408     'scale_type':'manual line',
409     'manual_axis_data': {20.0:'20mm',
410                         35.0:'35mm',
411                         50.0:'50mm',
412                         80.0:'80mm',
413                         100.0:'100mm',
414                         150.0:'150mm',
415                         200.0:'200mm',
416                         300.0:'300mm',
417                         400.0:'400mm',
418                         500.0:'500mm',
419                         1000.0:'1000mm'}
420 }
421
422
423
424 FL_block_params={
425     'block_type':'type_1',
426     'width':12.0,
427     'height':10.0,
428     'f1_params':FL_t_para,
429     'f2_params':FL_factor_params_2,
430     'f3_params':FL_fl_params,
431     'mirror_x':True,
432     'proportion':0.5,
433     'isopleth_values':[['x',1.0/(8.0/3.0),'x']],
434 }
435
436
437 main_params={
438     'filename':['ex_photo_exposure.pdf','ex_photo_exposure.eps'],
439     'paper_height':35.0,
440     'paper_width':35.0,
441     'block_params':[block_params,block_params_weather,block_params_scene,block_params_camera,EV_block,FL_block_params],
442     #'block_params':[block_params_weather,block_params_scene,block_params_camera,EV_block,FL_block_params],
443     'transformations': [('rotate',0.01),('scale paper',)],
444     'title_x': 7,
445     'title_y': 34,
446     'title_box_width': 10,
447     'title_str':r'\LARGE Photography exposure (Setala 1940) \par \copyright Leif Roschier 2009 '
448 }
449 b=Nomographer(main_params)

```

CHAPTER SEVEN

LICENSE

PyNomo is open source and free software licensed under the GNU GPL Version 3.