Creating Nomograms with the *PyNomo* Software

Version 1.1 for PyNomo Release 0.2.2

Ron Doerfler
http://www.myreckonings.com/wordpress
October 19, 2009

PYNOMO IS A POWERFUL, FREE SOFTWARE PACKAGE FOR DRAWING PRECISION NOMOGRAMS. It is written by Leif Roschier, and it provides a solution to one of the most vexing of problems in nomography: actually drawing the nomogram once the mathematical layout is derived. The output is in vector form in a PDF or EPS file, so it can be enlarged for printing and still retain its sharpness. If additional artwork is desired, the vector file can be edited in vector form in, say, Adobe Illustrator, Microsoft Visio or LaTeX. PyNomo directly supports nine basic types of nomograms based simply on the format of the equation, so for these types there is no need to convert the equation to the standard nomographic determinant or use geometric relations. But it also supports more complicated equations that have been cast into general determinant form, so it can produce output for any equation that can be plotted as a nomogram. In fact, PyNomo is an excellent tool for experimenting with parameters to get the best nomogram for the problem at hand. This essay discusses the use of PyNomo for creating a variety of sophisticated types of nomograms.[1]

1 Changes in Version 1.1 compared to Version 1.0

This version 1.1 introduces these significant changes to the PyNomo software:

1. Smart Tick Spacing on Scales ("smart axes") — Nonlinear scales generally need different tick levels for different parts of the scale to provide the greatest accuracy to the user. Areas where the major tick marks are more spaced out allow additional tick marks of lower levels to be inserted. In the past this has been accomplished by specifying an initial range and tick level for a scale, and then adding additional ranges with different tick levels using the scale parameter *extra_params*. The "smart axes" feature introduces scale types of *linear smart* or *log smart* that will automatically determine appropriate tick levels over the entire scale. It even adjusts labels on the scale to avoid overlap. Perhaps the most striking difference offered by this feature is seen by comparing the circular dials between this version and the previous version of this document.
2. Isopleths — To draw sample isopleths, or index lines, on a nomogram previously required drawing lines between grid coordinates on the overall nomogram. The main parameter *make_grid* could be set to True to draw a numbered grid over the nomogram, and then the coordinates would be determined manually and entered as the line coordinates. Thereafter, any change in the nomogram size or layout

[1] I would like to thank Leif Roschier, Joe Marasco and Scott Finegan for reviewing drafts of this essay and providing suggestions that have considerably improved it. If you have any questions or suggestions, please feel free to email me at doerfpub@myreckonings.com

required this process to be repeated. The "Isopleth" feature draws lines directly between specified values on the scales, calculating the points on intermediate scales as needed. As before, various styles and colors of lines are supported.

3. Text format of scale values — The previous version of PyNomo assigned the value $%3.2f$ as the default *text_format* for label values (as in `'text_format':r"$%3.2f$",`), which displayed up to 3 digits to the left of the decimal point, and in all cases 2 digits to the right of the decimal point even if the last digit were 0. The trailing zeros on the text labels are now eliminated by changing the default text format to $%4.4g$. The *g* print format produces a decimal number of up to 4 digits left of the decimal point and 4 to the right only as needed, unless the number is 0.0001 or less in which case it is printed in exponential form (e.g., 1e-4). This provides a cleaner nomogram, as unnecessary digits are not shown. As before, this default can be changed by specifying a different text format.

2 What is a Nomogram and Why Would It Interest Me?

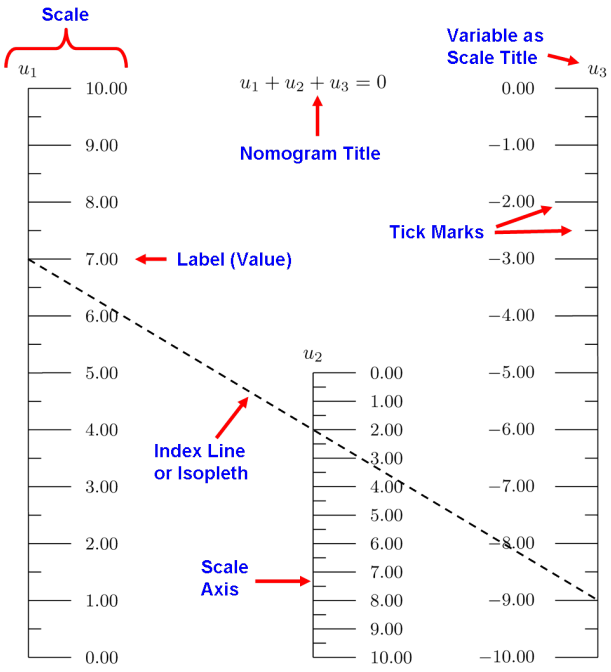
A nomogram is a diagram that provides a graphical way of calculating the result of a mathematical formula. Sometimes called an alignment chart, a nomogram consists of a set of numbered scales, usually one for each variable in the formula, arranged so that a straightedge can be placed across known values to find the unknown value that solves the formula. Since an equation in two variables is usually represented by a graph, most nomograms represent formulas that involve three or more variables. These graphical calculators were invented in 1880 by Philbert Maurice d'Ocagne and used extensively for many years to provide engineers with fast graphical calculations of complicated formulas to a practical precision. Electronic calculators and computers have made nomograms much less common today, but when a fast, handy calculator of a particular formula is needed they can be very useful. The cost to produce one is a sheet of paper, and they are fun to design, easy to use, and can be beautiful designs that engage people.[2]

The simplest nomogram, one we often encounter, is a single scale with numbers on both sides of it. This is called a conversion scale, and some people would hesitate to call it a nomogram at all. It provides conversion between two functions such as units of measurement. The most frequently seen example is probably a Fahrenheit to Celsius scale. We will inspect a single-scale nomogram as our first example.

The most common type of multi-scale nomogram consists of three parallel straight scales. It is used to solve an equation in which functions of three variables are added. The simplest such formula is $u_1 + u_2 + u_3 = 0$ for the three variables u_1, u_2 and u_3. An example of this type of nomogram from the PyNomo website[3] is shown below, annotated with terms used to describe the parts of a nomogram. A line (called an *index line* or *isopleth*) or a straightedge will cross the three scales at values that solve this equation, so if you know the values of any two of the variables you can find the third very easily.

[2] For more details on nomograms see *http://myreckonings.com/wordpress/2008/01/09/the-art-of-nomography-i-geometric-design/*, *http://www.pynomo.org/wiki/index.php/Basics*, *http://www.projectrho.com/nomogram/* and *http://www.barbecuejoe.com/scan.htm*

[3] *http://www.pynomo.org/wiki/index.php/Type_1*



There are many (out-of-print) books on the mathematics of nomography, but PyNomo allows us to design nearly all nomograms with very little mathematics background. A knowledge of algebra is necessary in order to arrange the formula into a standard type of equation that PyNomo supports. Then a PyNomo script for creating the nomogram can be copied from a standard example of that type and edited to match the terms in the formula. The script is run and a PDF file is created with the nomogram laid out for printing. Copying and modifying a standard example is easy and fast, and I do it all the time. Once you start making nomograms you may want to customize how they look—the spacing of tick marks on the scales, the scale titles, the location of the nomogram title, and so forth. You may want to draw a sample isopleth and add color to the scales and their labels. PyNomo offers many such features, and this paper tries to cover them all, but don't be put off by these extra details sprinkled in the examples here. They may make the scripts appear more complicated, but they are totally optional and can be ignored until the day you decide you really would like that one scale to be red. That's the point where you look in the index at the end of this essay for scale parameters that involve color.

A knowledge of logarithms is sometimes helpful to convert your formula to a standard type recognized by PyNomo. For example, suppose you have a typical formula in which three variables are multiplied or divided, and maybe one or two are raised to a power. You can convert it to a formula of simple additions by taking the logarithms of both sides of the equation. You only need to know a few properties of logarithms:

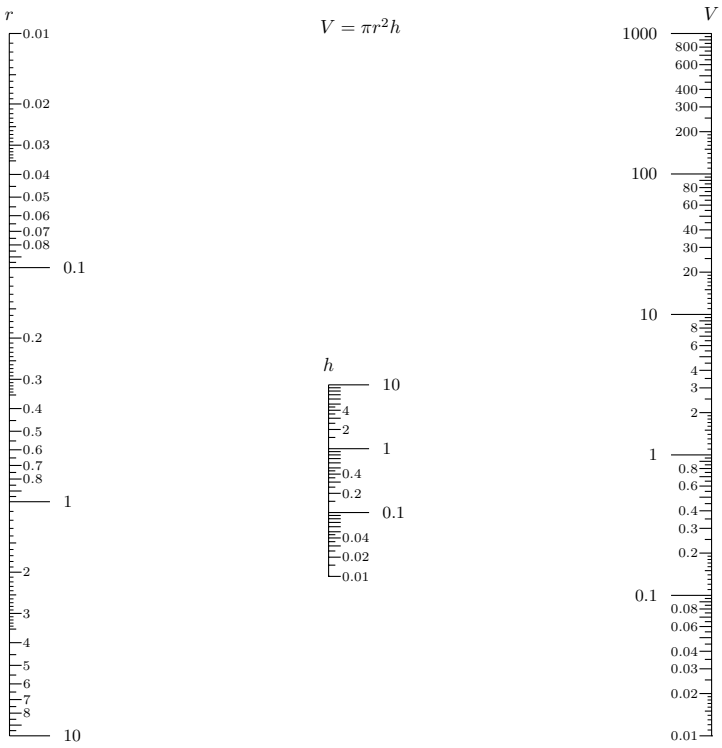
$$\log(a \times b) = \log a + \log b$$
$$\log(a/b) = \log a - \log b$$
$$\log(a^b) = b \log a$$

So a formula such as $V = \pi r^2 h$ for the volume of a cylinder can be expressed as an equation with simple

sums as

$$\log V = \log(\pi r^2 h)$$
$$\log V = \log(\pi r^2) + \log h$$
$$\log V = \log(\pi) + \log(r^2) + \log h$$
$$\log V = \log(\pi) + 2\log(r) + \log h$$
$$\log V - \log(\pi) - 2\log(r) - \log h = 0$$

So now we have an equation of a sum of terms, resulting in a parallel three-scale Type 1 nomogram as in our previous example. We replace u_1, u_2 and u_3 in our script with the functions $\log V$, $\{-\log(\pi) - 2\log(r)\}$ and $\log h$, and this creates the scales for V, r and h that form the nomogram. The output of the PyNomo script for a Type 1 nomogram with these variable functions looks like the one to the right. The scales have logarithmic spacing now, where the powers of 10 are the major tick marks and the intermediate values are spaced unevenly. The volume is easily found from this nomogram for a given radius and height, and the radius corresponding to a given volume and height is found just as easily. Try it—it works!



If a formula cannot be manipulated algebraically into one of the standard equations supported by PyNomo, it can be created using the Type 9 Determinant form supported by PyNomo. Sufficient information on determinants is provided in the Type 9 section of this essay to use this form without any prior knowledge of determinants.

3 Finding and Installing PyNomo

PyNomo can be found at *http://www.pynomo.org*. The software is OS-independent in principle, as Python interpreters exist for all major operating systems. It runs at least on Windows XP/Vista, Mac OS X and Linux. It requires installation of several other software packages, including the Python language, the PyNomo package for Python, four other Python add-on packages, and the MiKTeX distribution of the typesetting language LaTeX/TeX. Detailed instructions for the installation on Windows platforms are found on the PyNomo site[4].

[4] *http://pynomo.org/wiki/index.php?title=Installation* (click on the link at the top of this page for step-by-step instructions)

It is worth the time to install, as it produces beautiful nomograms without all the effort required to draw them by other means.

This article is based on Release 0.2.2 of PyNomo, the latest release as of this writing. There are two ways to install any files updated since the last release. First, you can go to the PyNomo development webpage[5]. Then click on the revision numbers in low-to-high order. For each filename listed for a revision, click on the filename, click on the *download* link along the top, and save the file to the folder containing the PyNomo files, choosing to replace the file when prompted. The PyNomo files are located in the Lib\site-packages\pynomo subfolder under the folder where you installed Python. Alternatively, on a Linux/OS X operating system or on a Windows system with cygwin installed, you can fetch all the revisions with the *svn* command:

```
svn co https://pynomo.svn.sourceforge.net/svnroot/pynomo pynomo
```

[5] *http://sourceforge.net/projects/pynomo/develop*

4 Supported Nomogram Types

If you are not familiar with PyNomo, you really should see the *Basics*[6] and *Examples*[7] links off the main PyNomo webpage. You will be amazed at what this program can do. If you go to the *Software Documentation* page[8] you will find 10 different general types of nomograms it can create, including the most common one of 3 parallel scales for addition or multiplication, "N" or "Z" for division, proportions, sum of reciprocals, and so forth. It can also handle general determinant forms, even with more than 3 variables, so it can really handle any type of nomogram. The types of nomograms supported by PyNomo and the sections that these are covered in here are shown in the table below. Section 10 deals exclusively with adding color to your nomogram. This paper concludes with Section 18 discussing some ways you can edit your PyNomo nomogram with external image tools if you wish, followed by an index of terms and parameters in this essay.

Type	**Form of Equation**	**Form of Nomogram**	**Covered In**
Type 1	$f_1(u_1)+f_2(u_2)+f_3(u_3)=0$	3 Parallel Scales	Section 6
Type 2	$f_1(u_1)=f_2(u_2)\times f_3(u_3)$ or $f_3(u_3)=f_1(u_1)/f_2(u_2)$	N or Z	Section 7
Type 3	$f_1(u_1)+f_2(u_2)+\ldots+f_n(u_n)=0$	Compound Parallel Scales	Section 8
—	$f_1(u_1)+f_2(u_2)+f_3(u_3)=0$ and $f_3(u_3)+f_4(u_4)+f_5(u_5)=0$	3 Parallel Scales Compounded	Section 11
Type 4	$\frac{f_1(u_1)}{f_2(u_2)}=\frac{f_3(u_3)}{f_4(u_4)}$	Proportion	Section 13
Type 5	$f_1(v)=f_2(x,u)$	Contour	Section 17
Type 6	$u=u$	Ladder	Section 12
Type 7	$\frac{1}{f_3(u_3)}=\frac{1}{f_1(u_1)}+\frac{1}{f_2(u_2)}$	Reciprocal/Angle	Section 14
Type 8	$y=f(u)$	Single Scale	Sections 4 & 9
Type 9	Determinant	General Nomogram	Section 16
Type 10	$f_1(u)+f_2(v)f_3(w)+f_4(w)=0$	One Curved Line	Section 15

Nomogram Types Supported by PyNomo

[6] *http://www.pynomo.org/wiki/index.php/Basics*
[7] *http://www.pynomo.org/wiki/index.php/Examples*
[8] *http://pynomo.org/wiki/index.php?title=Software_documentation*

5 The Structure of a PyNomo Script

Here we will inspect a simple PyNomo script for overall structure without delving into the details. More complicated scripts are just variations on this basic structure of scales and their parameters.

A script specifies the type of nomogram and the functions, titles and tick levels of each scale. Let's look at a script to create a simple conversion scale between miles and kilometers. This nomogram consists of a pair of scales that are assigned tags to cause them to overlay each other. A single scale is a Type 8 nomogram in PyNomo, so we will create two Type 8 nomogram blocks with the same tag name.

It's usually best to copy an existing script for a nomogram of the same type into a new text file and edit it for our needs. Each type of nomogram has an example on the PyNomo website, and there are also a number of examples presented in this paper. Each *scale* has its own set of parameters (function, title, etc.), and each set of scales that make up a nomogram type have a set of *block* parameters (the scale names, type and size). Compound nomograms will have multiple blocks. Finally, there is a set of *main* parameters (the block names, title, paper size and transformations to apply).

Below is the PyNomo script and output PDF graphic for the conversion scale. The top line is a comment line as it starts with a # sign. The next line and the last line are common to all scripts. A set of scale parameters is provided first for the Miles scale. This includes a tag name to share with the Kilometers scale so the two scale axes line up on top of one other. Then the minimum and maximum values are provided, the function is provided for the scale in terms of u, the title is given and shifted over to that side of the scale, the levels of tick marks to draw and the level of tick labels to display are indicated, and finally the side of the scale axis to draw the tick marks and labels is given. A set of block parameters identifies the PyNomo type for the scale and the name of the set of scale parameters.

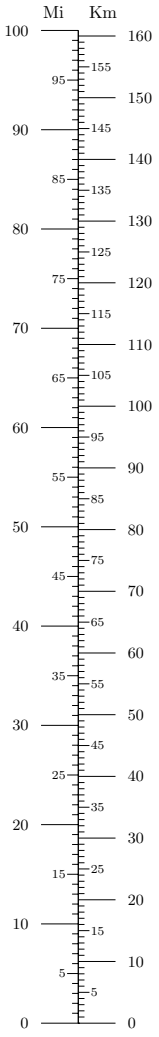
This is repeated for the Kilometers scale, but the ranges are obviously different by a factor of 1.609 and its function u is multiplied by 1.609. A new parameter provides the alignment function so the scales will overlay, which is the inverse or $u/1.609$. These ticks are placed on the right side of the scale axis.

The main parameters consist of the output filename, the paper height and width, and transformations to apply, here simply to scale the nomogram to the specified paper size.

If you want to make a conversion scale, just copy this script, change the titles, range values and functions, and run the script as described in the next section. Copying the script directly out of the PDF form of this essay will cause the indents to disappear and the script to fail, but the script and PDF output are available online for download.[9]

[9] *http://www.myreckonings.com/pynomo11/Type8-Sample.py* and *http://www.myreckonings.com/pynomo11/Type8-Sample.pdf*

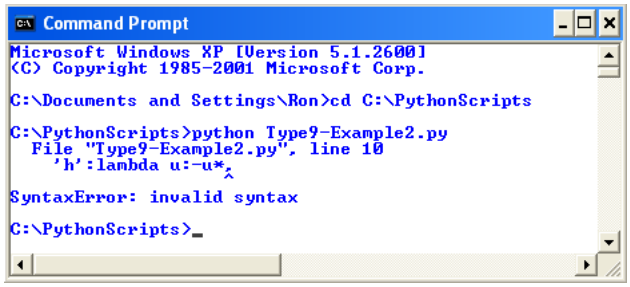
```
### Type8-Sample.py ###

from pynomo.nomographer import *

Miles_params={
        'tag':'myscale',
        'u_min':0,
        'u_max':100,
        'function':lambda u:u,
        'title':r'Mi',
        'title_x_shift':-0.5,
        'tick_levels':3,
        'tick_text_levels':2,
        'tick_side':'left',
                }

block_1_params={
        'block_type':'type_8',
        'f_params':Miles_params,
                }

Kilometers_params={
        'tag':'myscale',
        'u_min':0.0,
        'u_max':1.609*100.0,
        'function':lambda u:1.609*u,
        'align_func':lambda u:u/1.609,
        'title':r'Km',
        'title_x_shift':0.5,
        'tick_levels':3,
        'tick_text_levels':2,
        'tick_side':'right',
                }

block_2_params={
        'block_type':'type_8',
        'f_params':Kilometers_params,
                }

main_params={
        'filename':'Type8-Sample.pdf',
        'paper_height':20.0,
        'paper_width':0.5,
        'block_params':[block_1_params,block_2_params],
        'transformations':[('scale paper',)],
                }

Nomographer(main_params)
```



6 Basic Things to Know About Scripts

PyNomo scripts are written in text files saved with the suffix .py since they are interpreted by Python. Any text editor such as WordPad can be used as long as you save the output as a plain text file.

You can double-click on the .py file to build and display the output. A black command window appears for a little bit, and about 20 seconds later the PDF (or EPS) output file is written in your folder, and you can double-click on it to open it in your default PDF viewer (say, Adobe Acrobat). This is a vector-drawn nomogram that can be printed out or converted to another image format and inserted into a document. Note that you have to exit a previously-built PDF file of the same name before you build another or it will fail in overwriting the file. If the output file fails to materialize, there is an error in the script.



In order to see any error messages, it's much better to open a Command Prompt window in Windows or a Terminal application in Linux or OS X. For Windows, you can launch a Command Prompt window from the Start menu on the taskbar (Start–>Programs–>Accessories–>Command Prompt). Then navigate to the folder with your PyNomo file (say, *cd C:\PyNomoScripts*). Then enter *python <filename>.py* and a carriage return. Note that you enter *python*, not *pynomo*. Now as the build is occurring any error message will appear in the Command Prompt window. You should build the output file often as you write a PyNomo script in order to isolate errors as soon as they occur. Here is a typical Command Prompt window showing the navigation to the folder containing scripts (using the *cd* command to change the directory). In this case a syntax error is noted in the script because $-u*$ should be followed by a number that $-u$ is multiplied by.

As you use PyNomo more, you may be interested in using a text editor designed for writing and building Python files, such as EasyEclipse for Python.[10] I have only used WordPad and a Command Prompt window so far, though.

I don't know Python, but there are just a few things to remember about Python scripts. First, we are mostly filling in a "dictionary" of *key*:*value* parameters, where the *key* is a string in single quotes and the *value* can be a string in single quotes, a True or False, or a number. The parameters do not need to be in a particular order and most have default values. Also, you need to indent the list of parameters after the initial line containing the parameter set name, as Python uses indentation to determine the scope of a code construct. The amount of indentation is not important. Don't mix tabs and spaces—I would suggest using spaces only. Finally, a simple scale function is usually expressed as a local "lambda" function, but you will see that when a function is long it's cleaner to define it at the top of the script using the "def" command and give the function name in the scale parameters.

If you're like me, these are by far the most common errors you will encounter in a build:

- The output file cannot be opened, because I didn't close the PDF output file from the previous run.
- A syntax error where the offending line shown appears just fine, because the problem is really that I forgot to add a comma at the end of the previous line.

[10] *http://www.easyeclipse.org/site/distributions/python.html*

- A divide-by-zero error, because my scale range included a value that produces zero in a denominator in the scale function. This can sometimes occur due to a similar internal problem resolving the curves in a Type 5 Contour nomogram.
- A general hang-up where the program does not stop working and no error messages are output. You can end the program with a Ctrl-c key combination. I've found that this usually means I've made some sort of mistake in entering my scale function. It also occurred when I had a singularity in a particular Type 9 Determinant nomogram (see the Type 9 section).
- An undefined scale or block name, because I had copied and pasted a parameter section from another example and I forgot to change the name to the one called for in the block or main parameter section.

One solution to the divide-by-zero error is shown in an online PyNomo example[11], where the variable *u* in the scale function can be replaced by a function of *u*, say *limit_u(u)*, that makes sure the divide-by-zero situation does not occur. This function can be defined at the beginning of the script as

```
def limit_u(u):
    u1=u
    if not u1>0.0:
        u1=0.0001
    return u1
```

Formats of Title Text

The underlying typesetting engine used by PyNomo is a language called TeX. It handles text strings differently from mathematical expressions, and if you understand a few of the things you can do with these you will have more options for formatting them.

First, make sure you add the letter `r` right before the string in any title, even the extra titles, as in `'title_str':r'Equivalent Resistance'`, an indication of "raw" text that allows TeX formatting commands to be inserted in the title. It's possible in any title to precede the text with a TeX font size command such as `\Large` as in `'title_str':r'\Large Equivalent Resistance'`, which will print a larger title than normal. The options are

\tiny \scriptsize \footnotesize \small \normalsize \large
\Large \LARGE \huge \Huge

If a title string spans more than one line of your script, you have to add a backslash \ at the end of the line to indicate that the string continues on the next line (the \ produces a space, so place it immediately at the end without a space). To get a carriage return in a title you have three options:

1. Size the title box to force a return at the right place
2. Insert the \par command (does not work in the primary scale title *title_str*)
3. Use separate title texts placed to simulate a carriage return

If you precede a title with `\bf` then any title text will be printed in bold. Preceding it with `\it` will print it in italic, but both of these revert to normalsize font. A copyright symbol is generated by \copyright.

[11] *http://www.pynomo.org/wiki/index.php/Photography_exposure*

The following symbols in title text are special characters and must be preceded by a backslash, as in \$

$ & % # _ { } ’

TeX takes care of spaces between letters words, so even if you leave a few spaces between words in a title or between symbols in a math expression you will find that no extra space appears in the output nomogram. To add space you can use one of these commands that add 1/12em, 1/2em, 1em and 2em, respectively:

\thinspace \enspace \quad \qquad

Formats of Title Math Expressions

Mathematical expressions, such as formulas appearing in the title text, must begin and end in dollar signs ($), indicating the entry and exit of Math Mode. In this mode letters are printed in italic font as variables and the typesetting is optimized for math. For example, the title in our first example is given as `$(S+0.64)^{0.58}(0.74V) = P$` in which the `^` symbol indicates that the next character, or the next set of characters in curly braces, is a superscript. The title will appear in the output as:

$$(S+0.64)^{0.58}(0.74V) = P$$

You will see in some of the examples here that the title is placed in $ signs even though it may be just a letter. This simply prints the letter in italic. For titles of more than one word you will have to add spacing macros as described above, as the math mode removes spaces in your text.

Subscripts are similar but use an underscore rather than a caret: a_{12} in a math expression will appear as a_{12}. Note that you cannot have superscripts or subscripts in text without treating it as a mathematical expression by surrounding it with $ signs. Within a mathematical expression you can also use \pi to print π, and in general precede any Greek letter with \ to print the symbol, as \theta for θ and \Theta for the capital version Θ. For the times symbol you should use \times rather than x or *, as you get a larger symbol $\times$. The expression \div will produce a division sign $\div$ if you prefer this to a / symbol. The expression \approx will produce an approximation symbol $\approx$. Other common math symbols include ^ \circ for a degree sign $^\circ$ and \sqrt{} for a square root of the text in the curly braces, as \sqrt{L/m^2} produces $\sqrt{L/m^2}$. Other symbols are also available in math mode (i.e., within $ signs).[12]

Formats of Math Functions

Each scale has a mathematical function associated with it, and these should **not** be confused with the mathematical expressions we discussed above for displaying in title text. In functions the variable is generally u, so for example you should write 3*u for multiplying u by 3 and u**3 or pow(u,3) for cubing u. A significant number of math functions are available in Python.[13] The main ones, where x and y will be functions of u, are

exp(x), log(x[, base]), log(x), log10(x), pow(x, y), sqrt(x), acos(x), asin(x), atan(x), atan2(y, x), cos(x), hypot(x, y), sin(x), tan(x), degrees(x), radians(x), acosh(x), asinh(x), atanh(x), cosh(x), sinh(x), tanh(x)

[12] See the list at *http://abel.math.harvard.edu/computing/latex/manual/node21.html*

[13] For details see *http://docs.python.org/library/math.html*

Constants pi and e can also be used.

We often use log10(u) for the common logarithm of u. We will see this in our very first example, as we need to take the logarithm of both sides of an equation to convert it to the simple addition of logs to plot as a Type 1 nomogram, just as we described in Section 1. It's possible to use the natural logarithm log(u) everywhere since we usually scale the nomogram to the size of the paper, but this can cause problems when some functions of the nomogram include logs and some do not, so we will consistently use log10(u).

These are the basic things to know about a PyNomo script. The rest is diving in and learning by example, and that's what this paper is all about. Remember that it's easiest to start by copying and pasting existing examples and then editing them, as well as copying and pasting additional parameter and block sections when adding new scales.

7 Creating a Type 1 (Three Parallel Scales) Nomogram

The vast majority of nomograms are of Type 1, that is, composed of three parallel scales. The form of the equation for this nomogram is $f_1(u_1)+f_2(u_2)+f_3(u_3)=0$, so it can be used for addition of individual functions of three variables. But as we described in Section 1, it can also be used to multiply functions when they are expressed as logarithms. As our example, we will use this type of nomogram to graph the equation

$$(S+0.64)^{0.58}(0.74V)=P$$

We also assume that the engineering ranges we are interested in are $1.0<S<3.5$ and $1.0<V<2.0$. From the equation above we calculate the minimum P as $(1.0+0.64)^{0.58}(0.74\times 1.0)=0.986$ and the maximum P as $(3.5+0.64)^{0.58}(0.74\times 2.0)=3.374$. Discussions of plotting this same nomogram by geometrical means and by the use of determinants have been detailed elsewhere.[14] [15] Now we will see how it can be drawn using PyNomo.

Taking the logarithm of both sides of the equation under consideration and re-arranging the terms, we can express it as a simple sum of functions of the variables.

$$\begin{aligned} 0.58\log(S+0.64)+\log 0.74V &= \log P \\ 0.58\log(S+0.64)+\log 0.74+\log V &= \log P \\ 0.58\log(S+0.64)+\log V-\log P+\log 0.74 &= 0 \end{aligned}$$

From the general form above, you can see that this is a Type 1 nomogram where

$$\begin{aligned} f_1(S) &= 0.58\log(S+0.64) \\ f_2(V) &= \log V \\ f_3(P) &= -\log P+\log 0.74 \end{aligned}$$

We could have included the $\log 0.74$ term in $f_1(S)$ or $f_2(V)$ instead of $f_3(P)$ if we had wished. The *Type 1* link on the Software Documentation page of the PyNomo site[16] shows an example script for this type of nomogram for the equation $u_1+u_2+u_3=0$. This is the case $f_1(u_1)=u_1$, $f_2(u_2)=u_2$ and $f_3(u_3)=u_3$, so it's a simple matter to modify this example for what we need. We find that if we plot the nomogram scales in the left-to-right order SVP, the V-scale is quite small even after the polygon transformation, so we plot the scales in the order SPV.

The PyNomo script and output PDF graphic are shown on page 16. The PyNomo script and the PDF output are available online for download.[17]

Let's look over the PyNomo script. First, a set of parameters for each of the three scales is defined. There are quite a number of parameters available, as shown on Software Documentation page for this type, but the ones we mention here are

[14] *http://myreckonings.com/wordpress/2008/01/09/the-art-of-nomography-i-geometric-design/*

[15] *http://myreckonings.com/wordpress/2008/01/09/the-art-of-nomography-ii-designing-with-determinants/*

[16] *http://www.pynomo.org/wiki/index.php/Type_1*

[17] *http://www.myreckonings.com/pynomo11/Type1-Example1.py* and *http://www.myreckonings.com/pynomo11/Type1-Example1.pdf*

- *u_min* and *u_max*: the minimum and maximum values for the scale
- *function*: the function of the variable for the scale
- *title*: the title to place above the scale (remember to add the `r` between the colon and the string)
- *tick_levels*: the level to which the tick marks are printed
- *tick_text_levels*: the level to which the tick labels are printed
- *tick_side*: the side of the scale to print the tick marks and labels (right side by default, but I like to place them on the side with the most space available)
- *text_format*: the number of digits to the left and right of the decimal point in the tick labels (`r"$%4.4g$"` is the default, displaying up to 4 digits left of the decimal point and 4 to the right only as needed, unless the number is 0.0001 or less in which case it is printed in exponential form (e.g., 1e-4)

There are a few other options for scale titles that are not used here but will be seen later in our Type 4 example:

- *title_draw_center* (True or False): locate the title at the center of the scale point of the scale
- *title_distance_center*: distance of the title from the scale center (0.5cm by default)
- *title_opposite_tick* (True or False): place the centered title on the opposite side from the tick marks

These sets of scale parameters are followed by a set of parameters for the whole *block* of scales. The block parameters include

- *block_type*: the PyNomo Type of the block as defined in the previous section
- *height* and *width*: the height and width of the block, the overall size of which will be changed if the *scale paper* transformation is chosen in the main parameters (this is a rough value, as tick marks and texts may extend this)
- *f#_params*: the previously defined scales to include in the block, with # being nothing or a number for each scale included, depending on the Type of the nomogram
- *mirror_x*: if True, flip the scales of the finished block left-to-right to change their order (False by default)
- *mirror_y*: if True, flip the scales of the finished block top-to-bottom to change the direction that the scales increase or decrease (False by default)

In this simple nomogram there is only one block, but a compound nomogram can contain additional blocks as we will see later.

We can also specify a proportional spacing between the scales in the Type 1 block parameters if we don't use the scaling transformation, by using the *proportion* parameter with a default of 1.0, as in `'proportion':1.0`.

A set of overall *main* parameters for the nomogram appears at the end. These parameters include

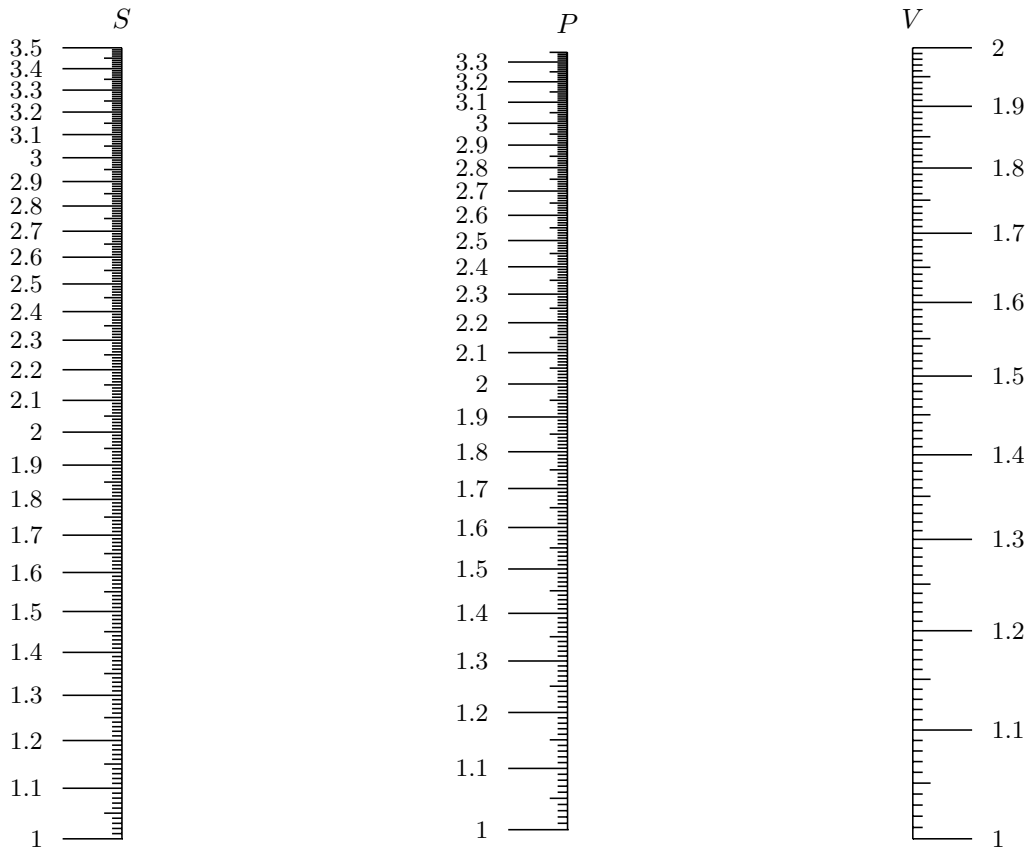
- *filename*: the output filename with a .pdf or .eps suffix
- *paper_height* and *paper_width*: the paper size to scale the nomogram to if scaling is specified (this is a rough value, as tick marks and texts may extend this)

- *block_params*: a list of defined nomogram blocks to include in the nomogram
- *transformations*: rotation, scaling, polygon transformations to apply as described below
- *title_x* and *title_y*: the absolute (x,y) coordinate in cm of the center of the nomogram title from the lower left
- *title_box_width*: the width of the invisible text box surrounding the title, useful to force a carriage return at the right place in the title
- *title_str*: the nomogram title (remember to add the `r` between the colon and the string)

There are a few transformations to the nomogram that can be included in the main parameters. One is a very tiny *rotation*—some rotation is mandatory, so we just rotate it an unnoticeable amount. The nomogram is also scaled to just fit the paper size defined earlier. Finally, I've chosen the *polygon* transformation—this adjusts the nomogram so that the outline of the scales fits in a rectangle in order to maximize their readability on the paper. I would suggest trying the polygon transformation after the nomogram is drawn to your satisfaction without it, as you may not prefer the skewing it can do to your scales. In this case it moves the middle P-scale away from the center so that both outer scales (and that scale, incidentally) end up the same height, which is the size of the paper since we are using the *scale paper* option. The fact that PyNomo does this scaling and polygonal transformation is a major benefit of using it—the polygonal transformation is a very tedious process to do manually.

Once we have created a Type 1 nomogram, modifying it for another equation that can be put in this form is really a very easy process.

```
### Type1-Example1.py ###

from pynomo.nomographer import *

S_params={
        'u_min':1.0,
        'u_max':3.5,
        'function':lambda u:0.58*log10(u+0.64),
        'title':r'$S$',
        'tick_levels':3,
        'tick_text_levels':1,
        'tick_side':'left',
                }

P_params={
        'u_min':1.0,
        'u_max':3.35,
        'function':lambda u:-log10(u)+log10(0.74),
        'title':r'$P$',
        'tick_levels':3,
        'tick_text_levels':1,
        'tick_side':'left',
                }

V_params={
        'u_min':1.0,
        'u_max':2.0,
        'function':lambda u:log10(u),
        'title':r'$V$',
        'tick_levels':3,
        'tick_text_levels':1,
                }

block_1_params={
        'block_type':'type_1',
        'width':10.0,
        'height':10.0,
        'f1_params':S_params,
        'f2_params':P_params,
        'f3_params':V_params,
                }

main_params={
        'filename':'Type1-Example1.pdf',
        'paper_height':10.0,
        'paper_width':10.0,
        'block_params':[block_1_params],
        'transformations':[('rotate',0.01),('scale paper',),
             ('polygon',)],
        'title_x':5.0,
        'title_y':-1.0,
        'title_box_width': 10.0,
        'title_str':r'$(S+0.64)^{0.58}(0.74V) = P$'
                }
Nomographer(main_params)
```



$$(S + 0.64)^{0.58}(0.74V) = P$$

Changing Relative Sizes

The previous example displays nicely-sized text for the scale of the nomogram. If we want smaller text or numbers, say if the tick labels were overlapping, we could decrease the size of these by specifying a larger paper size in the main parameters (*paper_height* and *paper_width*), say 20.0×20.0 cm instead of 10.0×10.0 cm. When the nomogram is scaled to the larger paper size, the scale ticks spread out but the text and numbers stay the same size, or in other words they will be smaller in comparison to the overall nomogram. Since the output is a vector image that can be scaled to any size for display or printing, the only effect is to decrease the relative size of the text and numbers, and make the scale lines thinner. If we want any titles to remain large, we can precede their text by a TeX font size macro such as \Large as described in the previous section. Since we are scaling to the paper size, the block size is only important in compound nomograms, as the relative sizes of the individual blocks give us flexibility in shaping the interior of the nomogram.

The next example is our previous example scaled to a larger paper size (but reduced here when imported to this document). The numbers and tick marks are reduced in scale, but the titles and overall title are increased in size to compensate by adding the \Large command to them. The title also had to be moved to the right 6cm to center it in the larger width. The scale values are still readable but greater accuracy is obtained with this nomogram. Sometimes scaling up to a larger paper size also allows us to add additional tick marks in the increased spacing between the original tick marks. Scaling up to larger paper sizes is particularly nice when the nomogram is printed on larger paper. If you are reading this on a monitor, zoom in to see the details of what this nomogram would look like on larger paper.

Adding Isopleths

Here we also add a new feature, a guide to the use of the nomogram in the form of a sample *isopleth* or *index line*. This has always been a very common feature of nomograms, and it is even more important today when most people, even engineers, have never seen a nomogram and have no idea how to use it. The more complicated nomograms we will see later really benefit from an isopleth or a picture key, although for simplicity not all the nomograms in this essay have them.

Version 1.0 of this manual used the main parameter *line_params* to draw an isopleth between to points on the nomogram, where the coordinates of the start and end were found by overlaying a grid onto the finished nomogram. Starting with Version 1.1, the following parameters are introduced specifically for drawing isopleths:

- *isopleth_values*: a block parameter providing the values on the scales in a block that are crossed by isopleths as described below
- *isopleth_params*: an optional main parameter that lists characteristics of each isopleth in order of appearance in the blocks (line style, line width, color, circles on endpoints) as described below

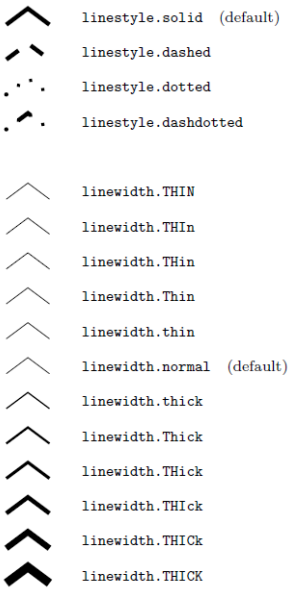
The *isopleth_values* block parameter consists of a list of scale values to cross with isopleths, where a character (or any set of characters) is used for values on scales that should be computed by PyNomo. For example, to draw an isopleth between the value of 6.0 on the first scale in a Type 1 block (*f1_params*) and the value 3.0 on the third scale (*f3_params*), plus another isopleth from a value of 2.0 on the first scale through the value 8.5 on the second scale, we would add the block parameter `'isopleth_values':[[6.0,'x',3.0],[2.0,8.5,'x']],` . There are variations of this format for different types of nomograms as we will see in those examples—for example, compound-scale Type 3 blocks include the multiple scale values for the zigzag isopleth within one set of square brackets.

The *isopleth_params* parameter is optionally included in the main parameters and applies the following characteristics to each isopleth calculated in order in the blocks listed in the main parameters. For example, if a compound nomogram of several blocks had one isopleth calculated through all the blocks, the first *isopleth_params* parameter would apply to this entire set. However, if another set were begun and calculated (with a different character string to differentiate them), the second *isopleth_params* parameter would apply to this second set.

- *linestyle*: the style of the isopleth, consisting of *solid*, *dashed*, *dotted* or *dashdotted* as illustrated in the figure below, with a default of dashed, e.g., `'linestyle':'dashed',`
- *linewidth*: the style of the isopleth, consisting of the values illustrated in the figure below), with a default of thick, e.g., `'linewidth':'THick',`
- *transparency*: the transparency of the isopleth, consisting of a value between 0.0 and 1.0, with a default of 0.0, e.g., `'transparency':0.4,`
- *circle_size*: the radius in cm of circles on the isopleth where it intersects the scales, with 0.0 producing no circles and a default of 0.05, e.g., `'circle\_size':0.2,`
- *color*: the color of the isopleth as listed on page 37, with a default of black, e.g., `'color':'OrangeRed',`

The set of characteristics for each isopleth is listed in *isopleth_params* within a set of braces, and the sets are listed in order within square brackets, as shown in the next script. Only isopleth parameters that are different from the default values need to be listed. Also, there is no reason to repeat the isopleth parameters if they are unchanged between isopleths, as the previous values are maintained. The linewidth and linestyle names are provided in the figure below.[18] The example script on the next page shows values for all the isopleth parameters to demonstrate their use.

This PyNomo script along with its PDF output are available online for download.[19]

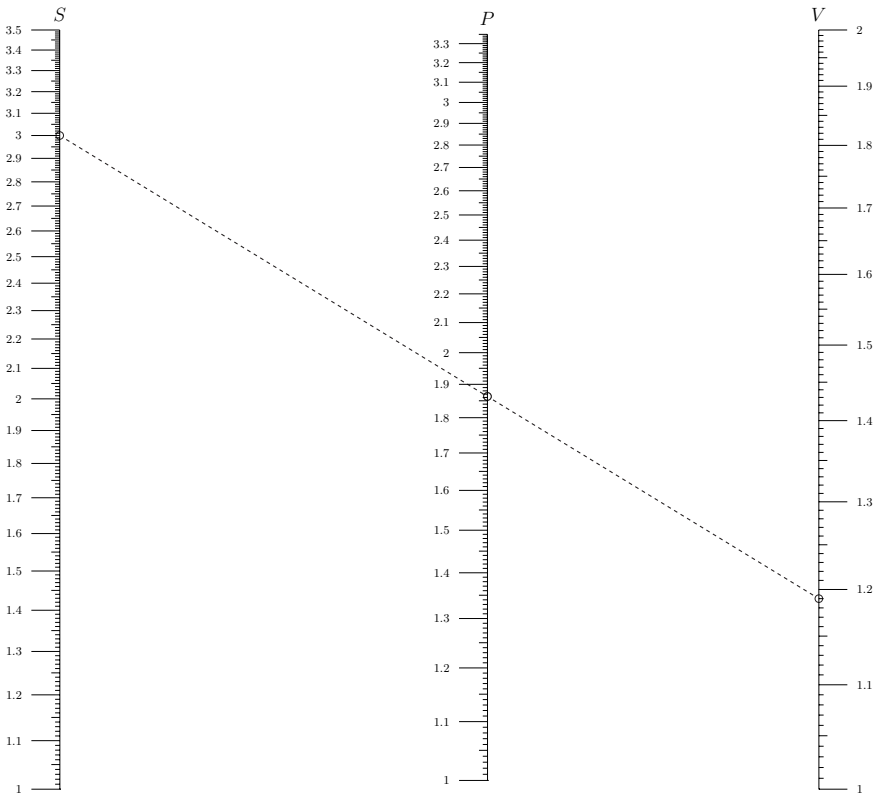


[18] From the Pyx Reference Manual, page 71, at *http://pyx.sourceforge.net/manual.pdf*

[19] *http://www.myreckonings.com/pynomo11/Type1-Isopleth.py* and *http://www.myreckonings.com/pynomo11/Type1-Isopleth.pdf*

```
### Type1-Isopleth.py ###

from pynomo.nomographer import *

S_params={
        'u_min':1.0,
        'u_max':3.5,
        'function':lambda u:0.58*log10(u+0.64),
        'title':r'\Large $S$',
        'tick_levels':3,
        'tick_text_levels':1,
        'tick_side':'left',
                }

P_params={
        'u_min':1.0,
        'u_max':3.35,
        'function':lambda u:-log10(u)+log10(0.74),
        'title':r'\Large $P$',
        'tick_levels':3,
        'tick_text_levels':1,
        'tick_side':'left',
                }

V_params={
        'u_min':1.0,
        'u_max':2.0,
        'function':lambda u:log10(u),
        'title':r'\Large $V$',
        'tick_levels':3,
        'tick_text_levels':1,
                }
```

```
block_1_params={
        'block_type':'type_1',
        'width':10.0,
        'height':10.0,
        'f1_params':S_params,
        'f2_params':P_params,
        'f3_params':V_params,
        'isopleth_values':[[3.0,'x',1.19]],
               }

main_params={
        'filename':'Type1-Isopleth.pdf',
        'paper_height':20.0,
        'paper_width':20.0,
        'block_params':[block_1_params],
        'transformations':[('rotate',0.01),('scale paper',),
             ('polygon',)],
        'title_x':11.0,
        'title_y':-1.0,
        'title_box_width': 10.0,
        'title_str':r'\Large $(S+0.64)^{0.58}(0.74V) = P$',
        'isopleth_params':[
                           {'color':'black',
                            'linewidth':'thick',
                            'linestyle':'dashed',
                            'circle_size':0.10,
                            'transparency':0.0,
                           },
                          ],
               }

Nomographer(main_params)
```



$$(S + 0.64)^{0.58}(0.74V) = P$$

8 Creating a Type 2 (*N* or *Z*) Nomogram

Nomograms categorized as Type 2 by PyNomo are also quite common. The form of this nomogram is $f_1(u_1) = f_2(u_2) \times f_3(u_3)$, although it is often expressed as a Division nomogram as in $f_3(u_3) = f_1(u_1)/f_2(u_2)$. It is called an *N* or *Z* nomogram because of its shape. We can use the same equation here as we used in the Type 1 nomogram for multiplication:

$$(S+0.64)^{0.58}(0.74V) = P \tag{1}$$

Again we assume that the engineering ranges we are interested in are $1.0 < S < 3.5$ and $1.0 < V < 2.0$ and from the equation calculate the range of P as $0.986 < P < 3.374$.

From the general form above, you can see that this can be drawn as a Type 2 nomogram if

$$\begin{aligned} f_1(P) &= P \\ f_2(V) &= 0.74V \\ f_3(S) &= (S+0.64)^{0.58} \end{aligned}$$

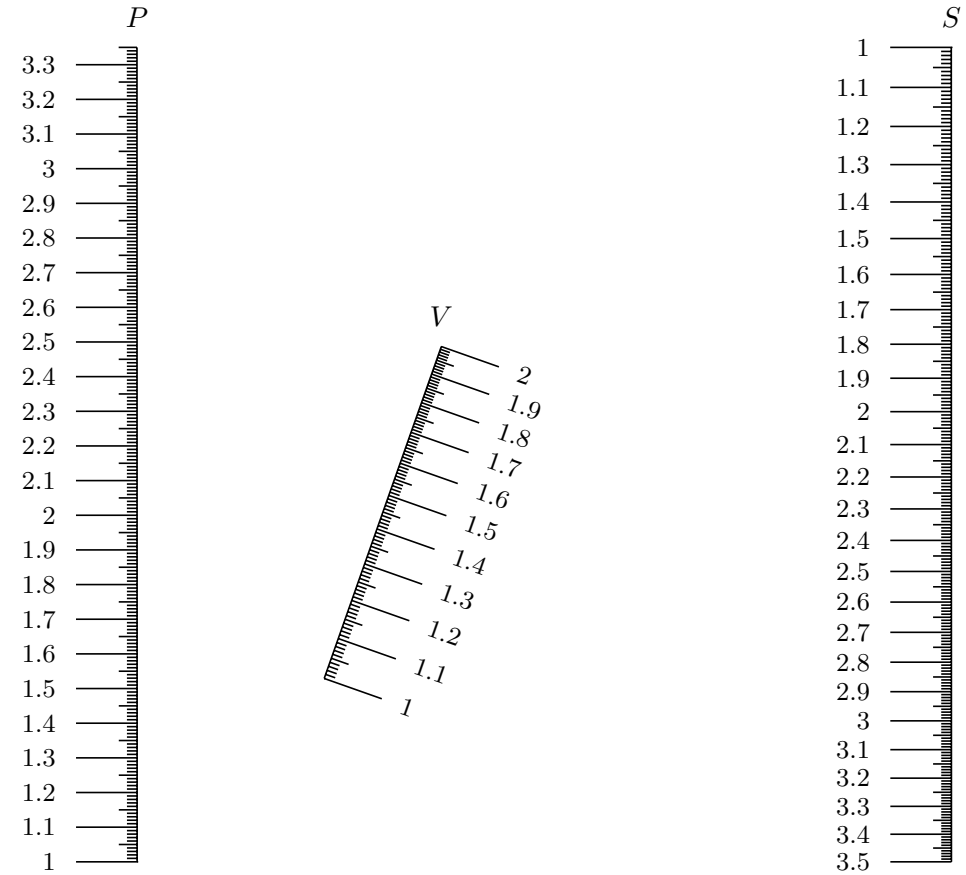
The *Type 2* link on the Software Documentation page of the PyNomo site[20] shows an example script for this type of nomogram for the equation $u_1 = u_2 \times u_3$. This is the case $f_1(u_1) = u_1$, $f_2(u_2) = u_2$ and $f_3(u_3) = u_3$, so again it's a simple matter to modify this example for what we need. To express the 0.58 as a power in f_3 we can write the expression as $(u+0.64)$**0.58 or we can use the Python math function `pow`$(u+0.64,0.58)$.

The PyNomo script and output PDF graphic are shown on the next page. Originally I had $f_2 = (S+0.64)^{0.58}$ and $f_3 = 0.74V$, but the S scale was very short as the diagonal so I switched f_2 and f_3. Trying variations to see which works best can be done very quickly with this software. Here the nomogram appears the same with and without the polygon transformation, since the outer scales are scaled to the paper size anyway. For this equation it appears to me that the earlier Type 1 version will provide greater practical accuracy for the user. The PyNomo script and output PDF graphic are available online for download.[21]

[20] *http://www.pynomo.org/wiki/index.php/Type_2*

[21] *http://www.myreckonings.com/pynomo11/Type2-Example.py* and *http://www.myreckonings.com/pynomo11/Type2-Example.pdf*

```
### Type2-Example.py ###

from pynomo.nomographer import *

P_params={
        'u_min':0.986,
        'u_max':3.374,
        'function':lambda u:u,
        'title':r'$P$',
        'tick_levels':3,
        'tick_text_levels':1,
        'tick_side':'left',
                }

V_params={
        'u_min':1.0,
        'u_max':2.0,
        'function':lambda u:0.74*u,
        'title':r'$V$',
        'tick_levels':3,
        'tick_text_levels':1,
                }

S_params={
        'u_min':1.0,
        'u_max':3.5,
        'function':lambda u:(u+0.64)**(0.58),
        'title':r'$S$',
        'tick_levels':3,
        'tick_text_levels':1,
        'tick_side':'left',
                }

block_1_params={
        'block_type':'type_2',
        'width':10.0,
        'height':10.0,
        'f1_params':P_params,
        'f2_params':V_params,
        'f3_params':S_params,
                }

main_params={
        'filename':'Type2-Example.pdf',
        'paper_height':10.0,
        'paper_width':10.0,
        'block_params':[block_1_params],
        'transformations':[('rotate',0.01),('scale paper',),
             ('polygon',)],
        'title_x':5.0,
        'title_y':-1.0,
        'title_box_width': 10.0,
        'title_str':r'$(S+0.64)^{0.58}(0.74V) = P$'
                }
Nomographer(main_params)
```



$$(S + 0.64)^{0.58}(0.74V) = P$$

9 Creating a Type 3 (Compound Parallel Scales) Nomogram

For situations where we have the sum of more than three functions of single variables, a nomogram with *pivot lines* can be drawn. The form of this equation is $f_1(u_1)+f_2(u_2)+\ldots+f_n(u_n)=0$ for any number n of variables. A pivot line is a blank scale that serves as a intermediate turning point for the straightedge. For example, if we have $f_1(u_1)+f_2(u_2)+f_3(u_3)+f_4(u_4)=0$, we can write this as a system of two equations

$$f_1(u_1)+f_2(u_2)+R=0$$
$$f_3(u_3)+f_4(u_4)-R=0$$

So we can create a nomogram block for the first equation consisting of three parallel lines with no scale drawn on the R-scale, then use the same R-scale to create a block for the second equation. The nomogram ends up as two concatenated blocks. To find u_4 the user lays a straightedge across the scale values for u_1 and u_2 and marks a point on the R-scale. Then the straightedge is placed across that mark and the scale value for u_3 to find u_4. Remember, you can find any unknown variable by working from the known variables.

PyNomo directly supports compound parallel lines for any number of variables, automatically sizing and placing the pivot lines appropriately. An example is provided on the Type 3 Software Documentation page.[22]

The example on the next page is drawn from electro-optics. It shows the relationship for a camera system of the target range, the number N of picture element cycles, the lens focal length, the target size, and the resolution achieved:

$$\text{Resolution}=\frac{N\times\text{Range}}{\text{Focal Length}\times\text{Target Size}}$$

or

$$\log(\text{Resolution})+\log(\text{Focal Length})+\log(\text{Target Size})-\log N-\log(\text{Range})=0$$

The new parameters that we introduce in this example are

- *title_y_shift* and *title_x_shift*: shifts in cm in a scale title for readability
- *text_format*: we changed the number display default of 2 digits to the right of the decimal point to 0 or 1 depending on the scale
- *scale_type*: The type of scaling (spacing) of the tick marks and labels of the scale. The options are:
  - *linear*: linear scaling (the default when no scale type is specified)
  - *log*: logarithmic scaling
  - *linear smart*: linear scaling with smart tick mark spacing
  - *log smart*: logarithmic scaling with smart tick mark spacing
  - *manual line*: manual selection of tick marks
  - *manual arrow*: manual selection using tick arrows
  - *manual point*: manual selection using tick points

  We will see the use of smart scale types later, but the following example shows the use of logarithmic scaling rather than the default linear scaling for all scales except the N scale, for which we will use manual scaling (see below)
- *reference_titles*: A list of titles to place over the pivot lines. For no titles at all, place no text between the single quotes. If this parameter is omitted, a default title of "R" is placed over each pivot line.

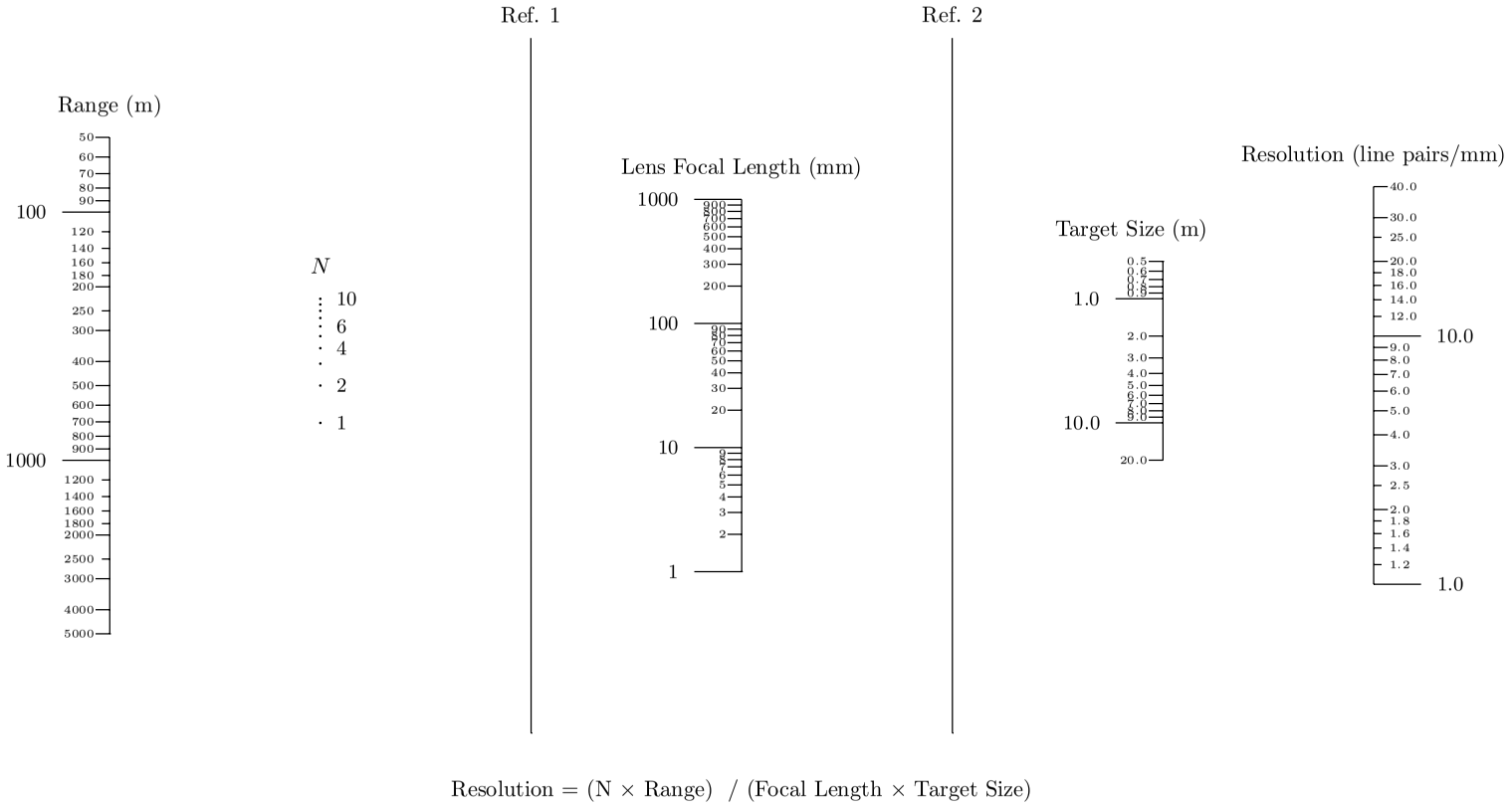
[22] *http://www.pynomo.org/wiki/index.php/Type_3*

It happens here that *N* is an integer from 1 to 10, so we only want to have ticks for those specific values. We can limit the tick marks on a scale to a predefined list of values through the *scale_type* and *manual_axis_data* parameters. The *scale_type* parameter can be set to *manual line*, *manual arrow* or *manual point* depending on whether we want a tick drawn, an arrow drawn or, as here, just the points drawn. In the *manual_axis_data* set we include a list of values to mark and a title text for each mark, which can be very useful for special labeling. Here we have no title for some of the points because they are so close the titles would collide. In this nomogram we also changed the overall paper size to get a nice horizontal layout of the scales.

This PyNomo script and output PDF graphic are available online for download.[23]

[23] *http://www.myreckonings.com/pynomo11/Type3-Example1.py* and *http://www.myreckonings.com/pynomo11/Type3-Example1.pdf*

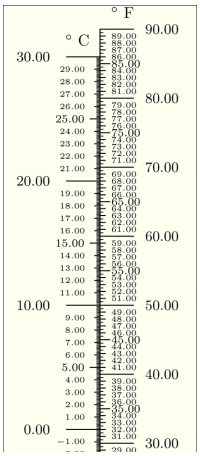
```
### Type3-Example1.py ###

from pynomo.nomographer import *

Range_meters_params={
        'u_min':5000.0,
        'u_max':50.0,
        'function':lambda u:-log10(u),
        'scale_type':'log',
        'title':r'Range (m)',
        'title_y_shift':0.4,
        'text_format':r"$%3.0f$",
        'tick_levels':4,
        'tick_text_levels':3,
                }

Number_of_Cycles_params={
        'u_min':1.0,
        'u_max':10.0,
        'function':lambda u:-log10(u),
        'scale_type':'log',
        'title':r'$N$',
        'title_y_shift':0.4,
        'text_format':r"$3.0f$",
        'tick_levels':2,
        'tick_text_levels':1,
        'scale_type':'manual point',
        'manual_axis_data': {1.0:'1',
                             2.0:'2',
                             3.0:'',
                             4.0:'4',
                             5.0:'',
                             6.0:'6',
                             7.0:'',
                             8.0:'',
                             9.0:'',
                             10.0:'10'
                             },
                }

Focal_Length_mm_params={
        'u_min':1.0,
        'u_max':1000.0,
        'function':lambda u:log10(u),
        'scale_type':'log',
        'title':r'Lens Focal Length (mm)',
        'title_y_shift':0.4,
        'tick_side':'left',
        'tick_levels':2,
        'text_format':r"$%3.0f$",
        'tick_text_levels':2,
        'scale_type':'log',
                }

Target_Size_meters_params={
        'u_min':0.5,
        'u_max':20.0,
        'function':lambda u:log10(u),
        'scale_type':'log',
        'title':r'Target Size (m)',
        'title_y_shift':0.4,
        'tick_side':'left',
        'title_x_shift':-0.5,
        'tick_levels':2,
        'text_format':r"$%3.1f$",
        'tick_text_levels':2,
        'scale_type':'log',
                }

Resolution_params={
        'u_min':1.0,
        'u_max':40.0,
        'function':lambda u:log10(u),
        'scale_type':'log',
        'title':r'Resolution (line pairs/mm)',
        'title_y_shift':0.4,
        'text_format':r"$%3.1f$",
        'tick_levels':3,
        'tick_text_levels':3,
                }

block_1_params={
        'block_type':'type_3',
        'width':10.0,
        'height':10.0,
        'reference_titles':['Ref. 1','Ref. 2'],
        'f_params':[Range_meters_params,Number_of_Cycles_params,
                    Focal_Length_mm_params,Target_Size_meters_params,
                    Resolution_params],
                }

main_params={
        'filename':'Type3-Example1.pdf',
        'paper_height':11.0,
        'paper_width':20.0,
        'block_params':[block_1_params],
        'transformations':[('rotate',0.01),('scale paper',)],
        'title_x':10.0,
        'title_y':-1.0,
        'title_box_width': 20.0,
        'title_str':r'Resolution = (N $\times$ Range) \
                      / (Focal Length $\times$ Target Size)',
                }

Nomographer(main_params)
```



Resolution = (N × Range) / (Focal Length × Target Size)

10 Creating a Type 8 (Single Scale) Nomogram

A Type 8 nomogram is a single, simple function scale. It provides a means of aligning (or overlaying) a scale right on top of another scale. This is useful if a scale in the nomogram is to be marked in two different units, with one set of tick marks on the left side and the other on the right. A single-scale example is provided on the Type 8 Software Documentation page.[24] The use of two aligned Type 8 scales for a Celsius/Fahrenheit temperature conversion scale is also provided as an example—a piece of it is shown on the right.[25]



In our previous electro-optics nomogram the Range scale was in meters. We will modify this script to align a Type 8 scale to this Range scale with tick marks and labels on the other side in units of feet. We will also add an inch scale to the mm scale for Lens Focal Length and a feet scale to the meter scale of the Target Size. This is shown on the following pages.

The new parameters that we introduce in this example are

- *tag*: scales with the same tag are aligned (overlaid) in the final nomogram
- *extra_titles*: additional titles to add to the scale beyond *title*. For each additional title, *dx* and *dy* specify the offsets in cm from the upper right corner of the scale, *text* is the title (remember to add the `r` between the colon and the string), and *width* is the title box width. We will see another parameter *pyx_extra_defs* in the next section to specify the color of the title. This last parameter also provides another way of specifying the text size rather than adding, say, \small in the text itself.
- *align_func*: the function used to align a Type 8 scale with another scale with the same tag, as described below
- *extra_texts*: additional titles to add to the overall nomogram. For each additional title, *x* and *y* specify the center of the title in cm from the lower left of the nomogram (*y* can be negative), *text* is the title (remember to add the `r` between the colon and the string), and *width* is the title box width. We will see another parameter *pyx_extra_defs* in the next section to specify the color of the title. This last parameter also provides another way of specifying the text size rather than adding, say, \large in the text itself.

We assign three tags to the scales, *range*, *focal* and *size*. Two scales are given each tag. The first is a scale from the previous example with the same title plus an extra title specifying the original units, shifted to left to lie over the side of the scale with those tick marks. The amounts *dx* and *dy* to shift the extra titles has to be done by experimenting. The second scale with the same tag is an aligned Type 8 scale with the tick marks and labels on the other side and its title (giving the other units) shifted right to lie over those tick marks. We make the font size of the units titles a bit smaller here by preceding the text with \small. The *Lens Focal Length* title is also broken into two titles to show the format of adding more than one extra title.

When we align a Type 8 scale to an existing scale, we have to specify the scale function and the minimum and maximum values that will provide a correct alignment. Let's look at the Range scale. The existing scale was in meters and had a minimum of 50.0m and a maximum of 5000.0m. Since there are 3.28 feet in a meter, the minimum in feet must be 50.0×3.28 and the maximum must be 5000.0×3.28. There is a new *align_func* parameter here that provides the scaling needed in the new units to produce the same scale length

[24] *http://www.pynomo.org/wiki/index.php/Type_8*
[25] *http://www.pynomo.org/wiki/index.php/Temperature_converter*

as the original one. In this example the align function should be $u/3.28$ so that the overall length of the scale is the same.

Instead of the single-line nomogram title in the previous section, this example shows the equation as a fraction, where the dividing line is simulated with a string of connected dashes. TeX assumes that two or three dashes are single longer dashes (endashes or emdashes), so to prevent an odd-looking extra dash at the end of the line, the number of dashes should not be 1 more than a multiple of 3. A backslash \ is used after "Resolution =" to indicate that the title text continues on the next script line.

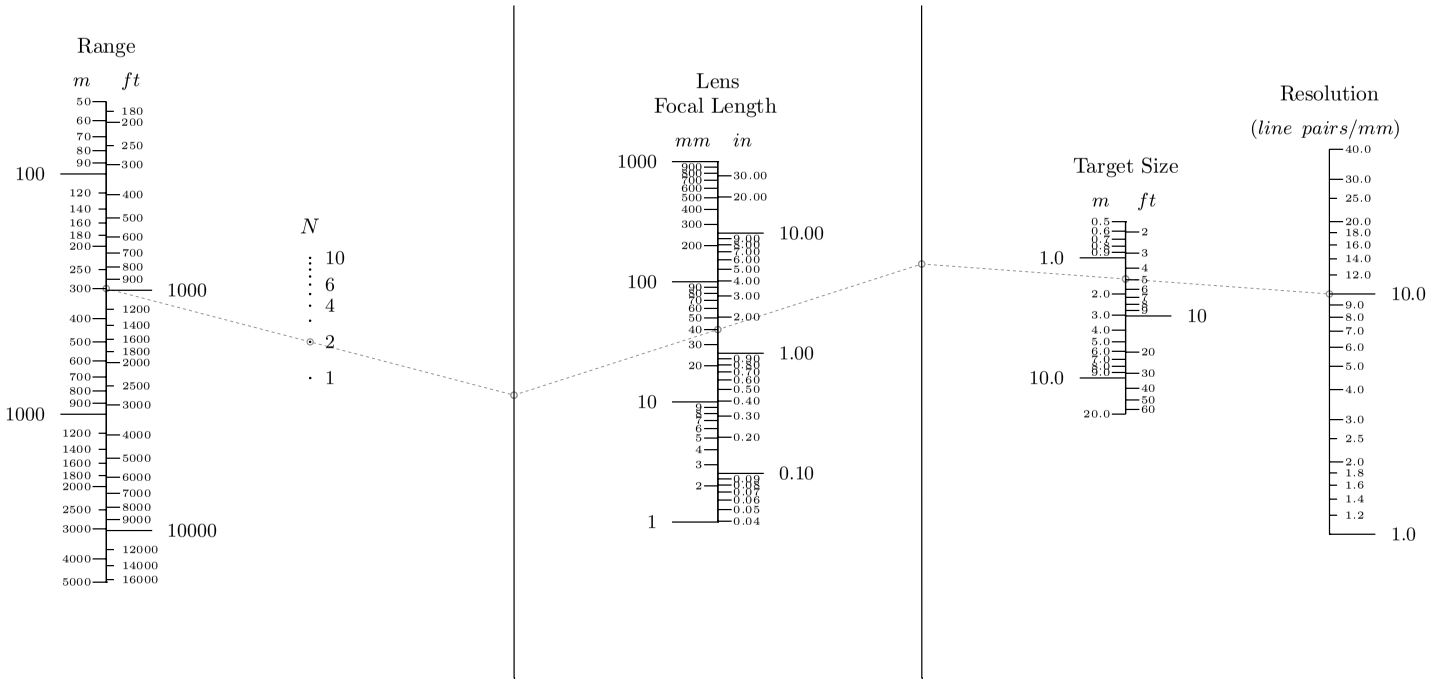
Here we also set the titles above the pivot lines to blanks in the *reference_titles* parameter in *block_1_params*. Notice that the *isopleth_values* parameter for the Type 3 block includes all the intersected scale values within a single set of square brackets, and that a calculated value (denoted by 'x') is required for each overlaid Type 8 scale.

This PyNomo script and output PDF graphic are available online for download.[26]

```
### Type3-DualScale.py ###

from pynomo.nomographer import *

Range_meters_params={
        'tag':'range',
        'u_min':5000.0,
        'u_max':50.0,
        'function':lambda u:-log10(u),
        'scale_type':'log',
        'title':r'Range',
        'title_y_shift':0.8,
        'text_format':r"$%3.0f$",
        'tick_levels':4,
        'tick_text_levels':3,
        'extra_titles':[
              {'dx':-1.25,
               'dy':0.25,
               'text':r'\small $m$',
               'width':5,
              }]
                }

Number_of_Cycles_params={
        'u_min':1.0,
        'u_max':10.0,
        'function':lambda u:-log10(u),
        'scale_type':'log',
        'title':r'$N$',
        'title_y_shift':0.4,
        'text_format':r"$3.0f$",
        'tick_levels':2,
        'tick_text_levels':1,
        'scale_type':'manual point',
        'manual_axis_data': {1.0:'1',
                             2.0:'2',
                             3.0:'',
                             4.0:'4',
                             5.0:'',
                             6.0:'6',
                             7.0:'',
                             8.0:'',
                             9.0:'',
                             10.0:'10'
                             },
                }

Focal_Length_mm_params={
        'tag':'focal',
        'u_min':1.0,
        'u_max':1000.0,
        'function':lambda u:log10(u),
        'scale_type':'log',
        'title':r'Lens',
        'title_y_shift':1.2,
        'tick_side':'left',
        'tick_levels':2,
        'text_format':r"$%3.0f$",
        'tick_text_levels':2,
        'extra_titles':[
              {'dx':-1.75,
               'dy':0.8,
               'text':'Focal Length',
               'width':5,
              },
              {'dx':-1.4,
               'dy':0.25,
               'text':r'\small $mm$',
               'width':5,
              }]
                }

Target_Size_meters_params={
        'tag':'size',
        'u_min':0.5,
        'u_max':20.0,
        'function':lambda u:log10(u),
        'scale_type':'log',
        'title':r'Target Size',
        'title_y_shift':0.8,
        'tick_side':'left',
        'tick_levels':2,
        'text_format':r"$%3.1f$",
        'tick_text_levels':2,
        'extra_titles':[
              {'dx':-1.25,
               'dy':0.25,
               'text':r'\small $m$',
               'width':5,
              }]
                }
```

[26] *http://www.myreckonings.com/pynomo11/Type3-DualScale.py* and *http://www.myreckonings.com/pynomo11/Type3-DualScale.pdf*

```
Resolution_params={
        'tag':'resolution',
        'u_min':1.0,
        'u_max':40.0,
        'function':lambda u:log10(u),
        'scale_type':'log',
        'title':r'Resolution',
        'title_y_shift':0.8,
        'text_format':r"$%3.1f$",
        'tick_levels':3,
        'tick_text_levels':3,
        'extra_titles':[
              {'dx':-2.0,
               'dy':0.25,
               'text':r'\small $(line \enspace pairs/mm)$',
               'width':5,
              }]
                }

block_1_params={
        'block_type':'type_3',
        'width':10.0,
        'height':10.0,
        'reference_titles':['',''],
        'f_params':[Range_meters_params,
                    Number_of_Cycles_params,
                    Focal_Length_mm_params,
                    Target_Size_meters_params,
                    Resolution_params],
        'isopleth_values':[[300,2,40,'x',10]],
                }

### TYPE 8 SINGLE-SCALE BLOCK ###
Range_feet_params={
        'tag':'range',
        'u_min':5000.0*3.28,
        'u_max':50.0*3.28,
        'function':lambda u:-log10(u),
        'scale_type':'log',
        'align_func':lambda u:u/3.28,
        'title':r'\small $ft$',
        'title_x_shift':0.4,
        'tick_side':'left',
        'text_format':r"$%3.0f$",
        'tick_levels':4,
        'tick_text_levels':3,
                }

block_2_params={
        'block_type':'type_8',
        'f_params':Range_feet_params,
        'isopleth_values':[['x']],
                 }

### TYPE 8 SINGLE-SCALE BLOCK ###
Focal_Length_inches_params={
        'tag':'focal',
        'u_min':1000.0/25.4,
        'u_max':1.0/25.4,
        'function':lambda u:log10(u),
        'scale_type':'log',
        'align_func':lambda u:u*25.4,
        'title':r'\small $in$',
        'title_x_shift':0.4,
        'tick_side':'left',
        'text_format':r"$%3.2f$",
        'tick_levels':2,
        'tick_text_levels':2,
                }

block_3_params={
        'block_type':'type_8',
        'f_params':Focal_Length_inches_params,
        'isopleth_values':[['x']],
                }

### TYPE 8 SINGLE-SCALE BLOCK ###
Target_Size_feet_params={
        'tag':'size',
        'u_min':0.5*3.281,
        'u_max':20.0*3.281,
        'function':lambda u:log10(u),
        'scale_type':'log',
        'align_func':lambda u:u/3.281,
        'title':r'\small $ft$',
        'title_x_shift':0.35,
        'tick_side':'right',
        'text_format':r"$%3.0f$",
        'tick_levels':2,
        'tick_text_levels':2,
                }

block_4_params={
        'block_type':'type_8',
        'f_params':Target_Size_feet_params,
        'isopleth_values':[['x']],
                }

main_params={
        'filename':'Type3-DualScale.pdf',
        'paper_height':11.0,
        'paper_width':20.0,
        'block_params':[block_1_params,block_2_params,
                        block_3_params,block_4_params],
        'transformations':[('rotate',0.01),('scale paper',)],
        'title_x':10.0,
        'title_y':-1.5,
        'title_box_width': 20.0,
        'title_str':r'\large Resolution =\
                     ------------------------------------',
        'extra_texts':[
              {'x':9.5,
               'y':-1.2,
               'text':r'\large $N$ $\times$ Range',
               'width':5,
              },
              {'x':8.1,
               'y':-1.85,
               'text':r'\large Focal Length $\times$ Target Size',
               'width':7,
              }],
        'isopleth_params':[
              {'color':'Gray',
               'linewidth':'thin',
               'linestyle':'dashed',
               'circle_size':0.05,
               'transparency':0.0,
              },
              ],
         }

Nomographer(main_params)
```



$$\text{Resolution} = \frac{N \times \text{Range}}{\text{Focal Length} \times \text{Target Size}}$$

It's also possible with Type 8 to create a curved set of scales for dial faces or for scales to be used in making a circular slide rule. Just for fun, let's create a set of circular scales that can be used as a dial face pasted behind a knob from which a clear plastic overlay is attached with a pointer line inscribed, or simply tacked in the center to a bulletin board with a thread tied to the pin of the tack.

The new parameters that we introduce in this example are

- *function_x*: a function that provides the x coordinate of a particular value of *u* along the scale
- *function_y*: a function that provides the y coordinate of a particular value of *u* along the scale
- *turn_relative*: if True, determine the left/right side of the scale for the *tick_side* parameter by the progression of values along the scale rather than the left/right side of the page. This is important only for curved scales as in this example, where if we did not set this value to True the side that the ticks is drawn on would change as the scale curved left or right. (False by default)
- *base_start* and *base_stop*: if present, use this minimum and maximum value rather than *u_min* and *u_max* to determine the settings of the *tick_levels*, useful for logarithmic scales. By setting these to a smaller range than provided for *u*, finer tick levels occur and setting *tick_level* to 1, for example, can produce finer spacing than would otherwise be possible.

The *function_x* and *function_y* parameters allow the scale curvature to be defined by functions of *u* in terms of x and y. For example, if we want a linear scale from 1.0 to 10.0 to be laid out in a circle of radius 7.0cm, we would write

```
'function_x':lambda u:7.0*cos(2*pi*u/(10.0-1.0),
'function_y':lambda u:7.0*sin(2*pi*u/(10.0-1.0),
```

Since $u/10.0-1.0$ is u divided by the overall range of u, then $2 \times \pi \times u/(10.0-1.0)$ will range from 0 to 2π, or a complete circle in radians. By using the same radius but with the ticks on the opposite side we can overlay two scales. The example on the next page has a scale for N for a radial pointer to be placed on. Then the other scales provide $\sqrt[3]{N}$, $\sqrt{N}$, $1/N$, N^2, N^3, the area and volume of a circle of radius N, and the radius of a circle with an area or volume N.

All the scales repeat the same basic set of parameters, which made it easy to write the script, but I had to test it a number of times to get the *x* and *y* offsets for the scale titles just right to fit in the available openings. The carriage return in the overall nomogram title in this example is forced by the width in cm of the *text_box_width* parameter rather than using \par or separating it into two texts with the *extra_texts* parameter as we did in the previous example.

The *N* scale ranges from 1 to 10 here, so if *N* lies outside this range the decimal point can be moved to place it in this range, the calculation can be completed, and then the decimal point can them be moved back appropriately. For example, the square root of 223 can be found by moving the decimal point two places to the left to get 2.23, the square root can be found as 1.49, and the decimal point can be moved right by one place to get 14.9 as the answer. The accuracy can be vastly improved by printing this on a large sheet of paper—zoom in to see what it would look like. Very complicated engineering formulas can be laid out on such a circular scale.

Many of the tick marks in this nomogram are a bit small to read very well. The size and orientation of the labels also dictate the minimum spacing between scales and therefore the overall size of the nomogram. We will see how to adjust all of these when we return to this circular dial in color form in the next section.

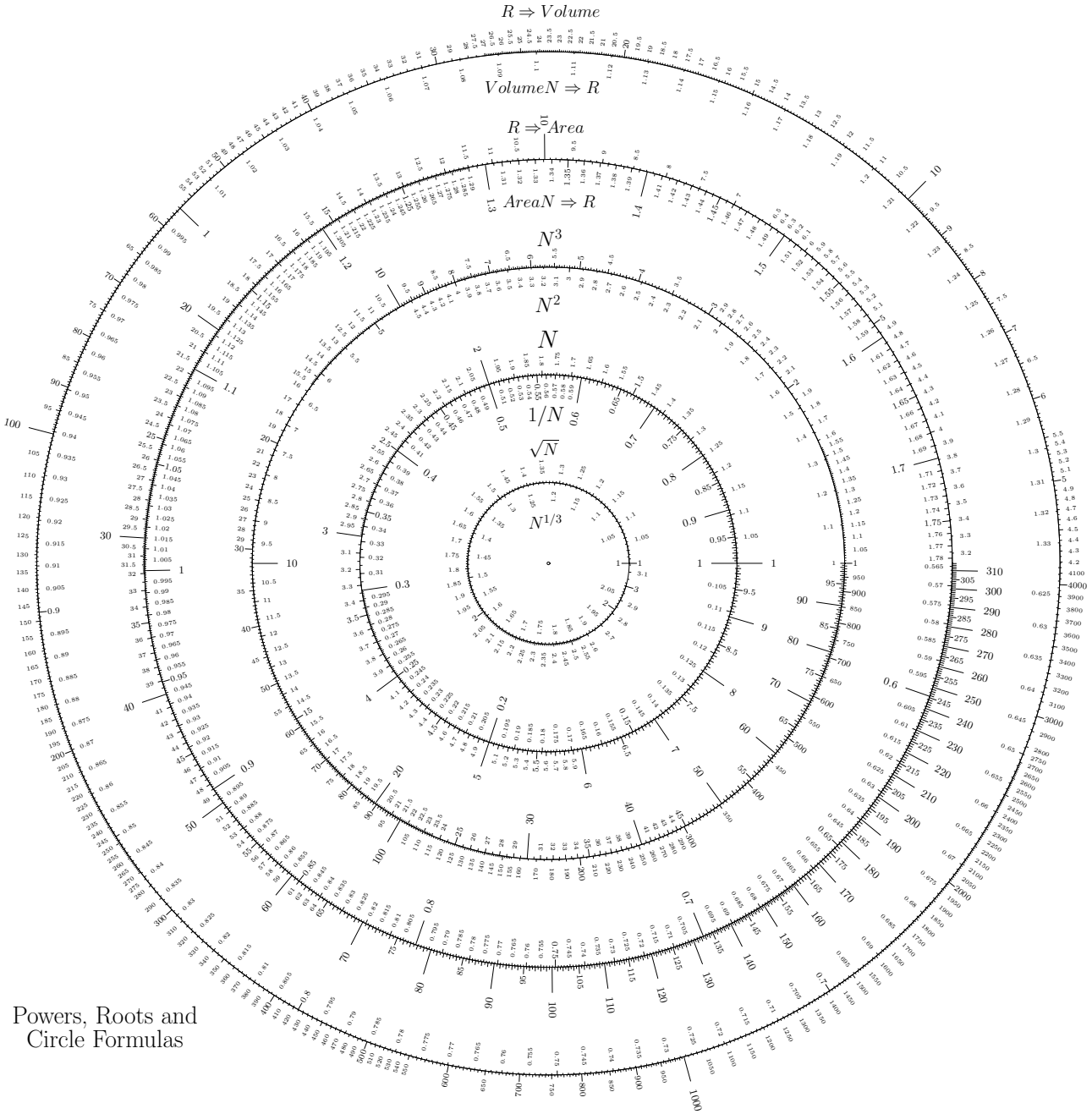
This PyNomo script and output PDF graphic are available online for download.[27]

```
### Type8-Dial-Smart.py ###

from pynomo.nomographer import *

CenterHole_params={
    'u_min':0.0,
    'u_max':1.0,
    'function_x':lambda u:0.05*cos(2*pi*u),
    'function_y':lambda u:0.05*sin(2*pi*u),
    'title':r'',
    'tick_levels':0,
    'tick_text_levels':0,
    }

CenterHole_block_params={
    'block_type':'type_8',
    'f_params':CenterHole_params,
    'width':15.0,
    'height':15.0,
    }

Cube_Root_params={
    'u_min':1.0,
    'u_max':2.09,
    'base_start':1.0,
    'base_stop':10.0,
    'function_x':lambda u:
          3.0*cos(2*pi*(log10(u)-log10(1.0))/
          (log10(2.1544)-log10(1.0))),
    'function_y':lambda u:
          3.0*sin(2*pi*(log10(u)-log10(1.0))/
          (log10(2.1544)-log10(1.0))),
    'title':r'\Large $N^{1/3}$',
    'title_x_shift':-0.05,
    'title_y_shift':-1.35,
    'tick_levels':5,
    'tick_text_levels':5,
    'scale_type':'log smart',
    'tick_side':'left',
    'turn_relative':True,
    }

Cube_Root_block_params={
    'block_type':'type_8',
    'f_params':Cube_Root_params,
    'width':15.0,
    'height':15.0,
    }

Square_Root_params={
    'u_min':1.0,
    'u_max':3.1623,
    'base_start':1.0,
    'base_stop':10.0,
    'function_x':lambda u:
          3.0*cos(2*pi*(log10(u)-log10(1.0))/
          (log10(3.1623)-log10(1.0))),
    'function_y':lambda u:
          3.0*sin(2*pi*(log10(u)-log10(1.0))/
          (log10(3.1623)-log10(1.0))),
    'title':r'\Large $\sqrt{N}$',
    'title_x_shift':-0.1,
    'title_y_shift':0.80,
    'tick_levels':5,
    'tick_text_levels':5,
    'scale_type':'log smart',
    'turn_relative':True,
    }

Square_Root_block_params={
    'block_type':'type_8',
    'f_params':Square_Root_params,
    'width':15.0,
    'height':15.0,
    }

Inverse_N_params={
    'u_min':0.101,
    'u_max':1.0,
    'base_start':0.1,
    'base_stop':1.0,
    'function_x':lambda u:
          7.0*cos(-2*pi*(log10(u)-log10(0.1))/
          (log10(1.0)-log10(0.1))),
    'function_y':lambda u:
          7.0*sin(-2*pi*(log10(u)-log10(0.1))/
          (log10(1.0)-log10(0.1))),
    'title':r'\LARGE $1/N$',
    'title_x_shift':-0.2,
    'title_y_shift':-1.35,
    'tick_levels':5,
    'tick_text_levels':5,
    'scale_type':'linear smart',
    'turn_relative':True,
    }

Inverse_N_block_params={
    'block_type':'type_8',
    'f_params':Inverse_N_params,
    'width':15.0,
    'height':15.0,
    }

N_params={
    'u_min':1.0,
    'u_max':9.9999,
    'base_start':1.0,
    'base_stop':9.9999,
    'function_x':lambda u:
          7.0*cos(2*pi*log10(u)/log10(10.0)),
    'function_y':lambda u:
          7.0*sin(2*pi*log10(u)/log10(10.0)),
    'title':r'\huge $N$',
    'title_x_shift':-0.15,
    'title_y_shift':0.8,
    'tick_levels':5,
    'tick_text_levels':5,
    'scale_type':'linear smart',
    'turn_relative':True,
    }

N_block_params={
    'block_type':'type_8',
    'f_params':N_params,
    'width':15.0,
    'height':15.0,
    }
```

[27] *http://www.myreckonings.com/pynomo11/Type8-Dial-Smart.py* and *http://www.myreckonings.com/pynomo11/Type8-Dial-Smart.pdf*

```
Square_params={
    'u_min':1.0,
    'u_max':99.999,
    'base_start':1.0,
    'base_stop':100.0,
    'function_x':lambda u:
          11.0*cos(2*pi*(log10(u)-log10(1.0))/
          (log10(100.0)-log10(1.0))),
    'function_y':lambda u:
          11.0*sin(2*pi*(log10(u)-log10(1.0))/
          (log10(100.0)-log10(1.0))),
    'title':r'\LARGE $N^2$',
    'title_x_shift':-0.5,
    'title_y_shift':-1.3,
    'tick_levels':5,
    'tick_text_levels':5,
    'scale_type':'linear smart',
    'tick_side':'left',
    'turn_relative':True,
    }

Square_block_params={
    'block_type':'type_8',
    'f_params':Square_params,
    'width':15.0,
    'height':15.0,
    }

Cube_params={
    'u_min':1.0,
    'u_max':999.999,
    'base_start':1.0,
    'base_stop':1000.0,
    'function_x':lambda u:
          11.0*cos(2*pi*(log10(u)-log10(1.0))/
          (log10(1000.0)-log10(1.0))),
    'function_y':lambda u:
          11.0*sin(2*pi*(log10(u)-log10(1.0))/
          (log10(1000.0)-log10(1.0))),
    'title':r'\LARGE $N^3$',
    'title_x_shift':2.2,
    'title_y_shift':0.9,
    'tick_levels':5,
    'tick_text_levels':5,
    'scale_type':'log smart',
    'turn_relative':True,
    }

Cube_block_params={
    'block_type':'type_8',
    'f_params':Cube_params,
    'width':15.0,
    'height':15.0,
    }

Inverse_Area_params={
    'u_min':0.5642,
    'u_max':1.7841,
    'base_start':0.5642,
    'base_stop':1.7841,
    'function_x':lambda u:
          15.0*cos(-2*pi*(log10(u)-log10(0.5642))/
          (log10(1.7841)-log10(0.5642))),
    'function_y':lambda u:
          15.0*sin(-2*pi*(log10(u)-log10(0.5642))/
          (log10(1.7841)-log10(0.5642))),
    'title':r'\Large $Area N \Rightarrow R$',
    'title_x_shift':0.0,
    'title_y_shift':-1.4,
    'tick_levels':5,
    'tick_text_levels':5,
    'scale_type':'linear smart',
    'turn_relative':True,
    }

Inverse_Area_block_params={
    'block_type':'type_8',
    'f_params':Inverse_Area_params,
     'width':15.0,
     'height':15.0,
     }

Area_params={
    'u_min':3.14159,
    'u_max':314.15,
    'base_start':3.14159,
    'base_stop':314.15,
    'function_x':lambda u:
          15.0*cos(2*pi*(log10(u)-log10(3.14159))/
          (log10(314.159)-log10(3.14159))),
    'function_y':lambda u:
          15.0*sin(2*pi*(log10(u)-log10(3.14159))/
          (log10(314.159)-log10(3.14159))),
    'title':r'\Large $R \Rightarrow Area$',
    'title_x_shift':-0.75,
    'title_y_shift':0.8,
    'tick_levels':5,
    'tick_text_levels':5,
    'scale_type':'linear smart',
    'turn_relative':True,
    }

Area_block_params={
    'block_type':'type_8',
    'f_params':Area_params,
    'width':15.0,
    'height':15.0,
    }

Inverse_Volume_params={
    'u_min':0.62035,
    'u_max':1.3365,
    'base_start':0.1,
    'base_stop':1.0,
    'function_x':lambda u:
          19.0*cos(-2*pi*(log10(u)-log10(0.62035))/
          (log10(1.3365)-log10(0.62035))),
    'function_y':lambda u:
          19.0*sin(-2*pi*(log10(u)-log10(0.62035))/
          (log10(1.3365)-log10(0.62035))),
    'title':r'\Large $Volume N \Rightarrow R$',
    'title_x_shift':-0.15,
    'title_y_shift':-1.2,
    'tick_levels':5,
    'tick_text_levels':5,
    'scale_type':'log smart',
    'turn_relative':True,
    }

Inverse_Volume_block_params={
    'block_type':'type_8',
    'f_params':Inverse_Volume_params,
    'width':15.0,
    'height':15.0,
    }

Volume_params={
    'u_min':4.1888,
    'u_max':4188.8,
    'base_start':1.0,
```

```
    'base_stop':1000.0,
    'function_x':lambda u:
          19.0*cos(2*pi*(log10(u)-log10(4.1888))/
          (log10(4188.8)-log10(4.1888))),
    'function_y':lambda u:
          19.0*sin(2*pi*(log10(u)-log10(4.1888))/
          (log10(4188.8)-log10(4.1888))),
    'title':r'\Large $R \Rightarrow Volume$',
    'title_x_shift':3.8,
    'title_y_shift':1.5,
    'tick_levels':5,
    'tick_text_levels':5,
    'scale_type':'log smart',
    'turn_relative':True,
    }

Volume_block_params={
    'block_type':'type_8',
    'f_params':Volume_params,
    'width':15.0,
    'height':15.0,
    }

main_params={
    'filename':'Type8-Dial-Example-Smart.pdf',
    'paper_height':30.0,
    'paper_width':30.0,
    'block_params':[CenterHole_block_params,
        Cube_Root_block_params,Square_Root_block_params,
        Inverse_N_block_params,N_block_params,
        Square_block_params,Cube_block_params,
        Inverse_Area_block_params,Area_block_params,
        Inverse_Volume_block_params,Volume_block_params,],
    'transformations':[('rotate',0.01),('scale paper',)],
    'title_x': 2,
    'title_y': 1.5,
    'title_box_width': 6,
    'title_str':r'\huge Powers, Roots and Circle Formulas',
    }

Nomographer(main_params)
```



Powers, Roots and Circle Formulas

11 Introducing Color

I think I have only ever seen a color nomogram once, and I made it.[28] But the most recent version of PyNomo supports color for scale axes, labels, titles, lines, pivot lines and arrows. In this age when people are used to splashy, colorful infographics, I think that adding color to nomograms is one way of lifting these graphical calculators from their dry, retro look to something that seems relevant and interesting today.

Scale parameters specifying color are:

- *axis_color*: the color of the axis (scale line) and tick marks in scale parameters
- *text_color*: the color of the numbers in scale parameters
- *title_color*: the color of the main scale title
- *pyx_extra_defs*: the text color in *extra_titles* in scale parameters
- *arrow_color*: the color of arrows in a *manual arrow* scale type
- *reference_color*: the color of the pivot lines and their titles in a Type 3 nomogram
- *u_line_color*: The color of the u-scale lines in a grid in a Type 9 Determinant nomogram
- *u_text_color*: The color of the u-scale labels of a grid in a Type 9 Determinant nomogram
- *v_line_color*: The color of the v-scale lines of a grid in a Type 9 Determinant nomogram
- *v_text_color*: The color of the v-scale labels of a grid in a Type 9 Determinant nomogram

Block parameters specifying color are:

- *reference_color*: the color of the diagonal reference line of a Type 4 Proportion nomogram
- *ladder_color*: the color of the ladder lines of a Type 6 Ladder nomogram
- *u_axis_color*: The color of the u-scale axis in a Type 5 Contour block
- *u_title_color*: The color of the title text of the u-scale in a Type 5 Contour block
- *u_text_color*: The color of the u-scale labels in a Type 5 Contour block
- *v_axis_color*: The color of the v-scale family of curves in a Type 5 Contour block
- *v_title_color*: The color of the title text of the v-scale in a Type 5 Contour block
- *v_text_color*: The color of the v-scale labels in a Type 5 Contour block
- *wd_axis_color*: The color of the wd-scale axis in a Type 5 Contour block
- *wd_title_color*: The color of the title text of the wd-scale in a Type 5 Contour block
- *wd_text_color*: The color of the wd-scale labels in a Type 5 Contour block

Main parameters specifying color are:

- *title_color*: the color of the main nomogram title
- *pyx_extra_defs*: the text color in *extra_texts* in the main parameters
- *color*: the color of an isopleth in the *isopleth_params* parameter, with a default of black (note that unlike the other parameters listed here, the name of the color is directly used, as in `Dandelion`.

[28] *http://myreckonings.com/wordpress/2008/02/24/a-zoomorphic-nomogram/*

- *circle_color*: the color of circles at the starting and ending points of line segments in the *line_params* parameter, with a default of black
- (the color of the line segments in the *line_params* parameter is specified as part of the *line_style* sub-parameter)

On the next page is the available color palette, where `color.` precedes the selected color, as for example `color.cmyk.BurntOrange` or `color.rgb.blue`.[29] It's possible to specify the fraction (0.0 to 1.0) of red, green and blue as in `color.rgb(0.9,0.0,0.3)` or `color.cmyk(0.5,0.0,0.8,0.1)`. This is how I specified the very light red for the titles in the next example. You can also specify a shade of gray, as in `color.gray(0.8)` for 80% white (20% black), which is done in the next example for the pivot lines. The default color for all items is black (`color.rgb.black`) unless specified differently, so all PyNomo scripts are backward-compatible. The one exception to this color naming convention is the *color* parameter within *isopleth_params*, which directly uses the color name with no prefix, as in `GreenYellow`.

An example of a nomogram with color is a version of the Type 3 nomogram shown on the pages following the color palette. The *scale_type* parameter has a new value *manual arrow* introduced here in order to show how the texts and colors are set for arrows. The scale parameter *extra_params* is used to place the arrows:

- *extra_params*: one or more additional sets of values to plot along a scale, which can specify new values for all scale parameters except *function* and *title*, useful to plot individual ranges, to increase tick levels in a particular range of a scale, or to add tick marks or arrows to specific values along the scale.

The nomogram doesn't exactly have a pleasing mix of colors (the blue and green are supposed to indicate metric vs. English scales), but it's just a demonstration of how different colors are applied. It's also a dry engineering formula in electro-optics, so it doesn't benefit much from a color design as, say, lighter nomograms on exercise time vs. calories or social statistics.

Also shown here is another way to size text in extra titles—rather than precede the text with, say, \small, you can specify the text size in the *pyx_extra_defs* parameter as *text.size.small* instead.

The PyNomo script and output PDF graphic for the color Type 3 nomogram are available online for download.[30]

[29] From the Pyx Reference Manual, page 69, at *http://pyx.sourceforge.net/manual.pdf*

[30] *http://www.myreckonings.com/pynomo11/Type3-Color.py*
and *http://www.myreckonings.com/pynomo11/Type3-Color.pdf*

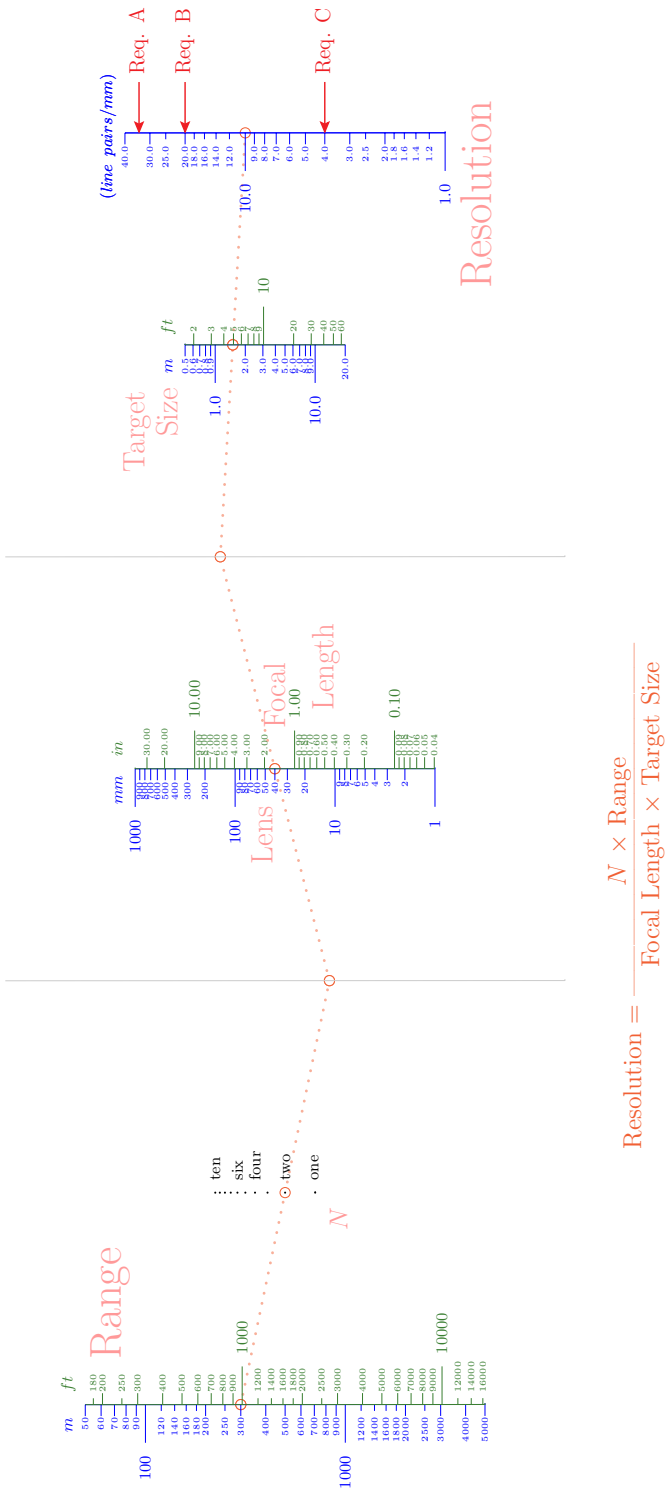
Named colors

```
### Type3-Color.py ###

from pynomo.nomographer import *

Range_meters_params={
        'tag':'range',
        'u_min':5000.0,
        'u_max':50.0,
        'function':lambda u:-log10(u),
        'scale_type':'log',
        'title':r'\Huge Range',
        'title_x_shift':2.0,
        'title_y_shift':-0.7,
        'text_format':r"$%3.0f$",
        'tick_levels':4,
        'tick_text_levels':3,
        'axis_color':color.rgb.blue,
        'text_color':color.rgb.blue,
        'title_color':color.rgb(1.0,0.6,0.6),
        'extra_titles':[
              {'dx':-1.25,
               'dy':0.25,
               'text':r'$m$',
               'width':5,
               'pyx_extra_defs':[color.rgb.blue,\
                                 text.size.small],
              }],
                }

Number_of_Cycles_params={
        'u_min':1.0,
        'u_max':10.0,
        'function':lambda u:-log10(u),
        'scale_type':'log',
        'title':r'\Large $N$',
        'title_x_shift':-0.5,
        'title_y_shift':-2.6,
        'text_format':r"$3.0f$",
        'tick_levels':2,
        'tick_text_levels':1,
        'title_color':color.rgb(1.0,0.6,0.6),
        'scale_type':'manual point',
        'manual_axis_data': {1.0:'one',
                             2.0:'two',
                             3.0:'',
                             4.0:'four',
                             5.0:'',
                             6.0:'six',
                             7.0:'',
                             8.0:'',
                             9.0:'',
                             10.0:'ten'
                             },
                }

Focal_Length_mm_params={
        'tag':'focal',
        'u_min':1.0,
        'u_max':1000.0,
        'function':lambda u:log10(u),
        'scale_type':'log',
        'title':r'\LARGE Lens',
        'title_x_shift':-1.3,
        'title_y_shift':-2.7,
        'tick_side':'left',
        'tick_levels':2,
        'text_format':r"$%3.0f$",
        'tick_text_levels':2,
        'axis_color':color.rgb.blue,
        'text_color':color.rgb.blue,
        'title_color':color.rgb(1.0,0.6,0.6),
        'extra_titles':[
              {'dx':0.1,
               'dy':-2.95,
               'text':r'Focal',
               'width':5,
               'pyx_extra_defs':[color.rgb(1.0,0.6,0.6),\
                                 text.size.LARGE],
              },
              {'dx':0.45,
               'dy':-3.9,
               'text':r'Length',
               'width':5,
               'pyx_extra_defs':[color.rgb(1.0,0.6,0.6),\
                                 text.size.LARGE],
              },
              {'dx':-1.4,
               'dy':0.25,
               'text':r'$mm$',
               'width':5,
               'pyx_extra_defs':[color.rgb.blue,\
                                 text.size.small],
              }],
                }

Target_Size_meters_params={
        'tag':'size',
        'u_min':0.5,
        'u_max':20.0,
        'function':lambda u:log10(u),
        'scale_type':'log',
        'title':r'\LARGE Target',
        'title_x_shift':-1.7,
        'title_y_shift':0.8,
        'tick_side':'left',
        'tick_levels':2,
        'text_format':r"$%3.1f$",
        'tick_text_levels':2,
        'axis_color':color.rgb.blue,
        'text_color':color.rgb.blue,
        'title_color':color.rgb(1.0,0.6,0.6),
        'extra_titles':[
              {'dx':-2.5,
               'dy':0.15,
               'text':r'Size',
               'width':5,
               'pyx_extra_defs':[color.rgb(1.0,0.6,0.6),\
                                 text.size.LARGE],
              },
              {'dx':-1.25,
               'dy':0.25,
               'text':r'$m$',
               'width':5,
               'pyx_extra_defs':[color.rgb.blue,\
                                 text.size.small],
              }],
                }

Resolution_params={
        'tag':'resolution',
        'u_min':1.0,
        'u_max':40.0,
        'function':lambda u:log10(u),
        'scale_type':'log',
        'title':r'\Huge Resolution',
        'title_x_shift':-1.2,
        'title_y_shift':-7.2,
        'tick_side':'left',
        'text_format':r"$%3.1f$",
        'tick_levels':3,
```

```
        'tick_text_levels':3,
        'axis_color':color.rgb.blue,
        'text_color':color.rgb.blue,
        'title_color':color.rgb(1.0,0.6,0.6),
        'extra_titles':[
              {'dx':-2.0,
               'dy':0.25,
               'text':r'$(line \enspace pairs/mm)$',
               'width':5,
               'pyx_extra_defs':[color.rgb.blue,\
                                 text.size.small],
              }],
        'extra_params':[{
               'tick_side':'right',
               'scale_type':'manual arrow',
               'manual_axis_data':{4.0:'\large Req. C',
                     20.0:'\large Req. B',
                     34.0:'\large Req. A'},
               'arrow_color':color.cmyk.Red,
               'text_color':color.cmyk.Red,
              }],
                }

block_1_params={
        'block_type':'type_3',
        'width':10.0,
        'height':10.0,
        'reference_titles':['',''],
        'reference_color':color.gray(0.8),
        'f_params':[Range_meters_params,
                    Number_of_Cycles_params,
                    Focal_Length_mm_params,
                    Target_Size_meters_params,
                    Resolution_params],
        'isopleth_values':[[300,2,40,'x',10]],
                }

### TYPE 8 SINGLE-SCALE BLOCK ###
Range_feet_params={
        'tag':'range',
        'u_min':5000.0*3.28,
        'u_max':50.0*3.28,
        'function':lambda u:-log10(u),
        'scale_type':'log',
        'align_func':lambda u:u/3.28,
        'title':r'\small $ft$',
        'title_x_shift':0.4,
        'tick_side':'left',
        'text_format':r"$%3.0f$",
        'tick_levels':4,
        'tick_text_levels':3,
        'axis_color':color.cmyk.OliveGreen,
        'text_color':color.cmyk.OliveGreen,
        'title_color':color.cmyk.OliveGreen,

block_2_params={
        'block_type':'type_8',
        'f_params':Range_feet_params,
        'isopleth_values':[['x']],
                }

### TYPE 8 SINGLE-SCALE BLOCK ###
Focal_Length_inches_params={
        'tag':'focal',
        'u_min':1000.0/25.4,
        'u_max':1.0/25.4,
        'function':lambda u:log10(u),
        'scale_type':'log',
        'align_func':lambda u:u*25.4,
        'title':r'\small $in$',
        'title_x_shift':0.4,
        'tick_side':'left',
        'text_format':r"$%3.2f$",
        'tick_levels':2,
        'tick_text_levels':2,
        'axis_color':color.cmyk.OliveGreen,
        'text_color':color.cmyk.OliveGreen,
        'title_color':color.cmyk.OliveGreen,
                }

block_3_params={
        'block_type':'type_8',
        'f_params':Focal_Length_inches_params,
        'isopleth_values':[['x']],
                }

### TYPE 8 SINGLE-SCALE BLOCK ###
Target_Size_feet_params={
        'tag':'size',
        'u_min':0.5*3.281,
        'u_max':20.0*3.281,
        'function':lambda u:log10(u),
        'scale_type':'log',
        'align_func':lambda u:u/3.281,
        'title':r'\small $ft$',
        'title_x_shift':0.35,
        'tick_side':'right',
        'text_format':r"$%3.0f$",
        'tick_levels':2,
        'tick_text_levels':2,
        'axis_color':color.cmyk.OliveGreen,
        'text_color':color.cmyk.OliveGreen,
        'title_color':color.cmyk.OliveGreen,
                }

block_4_params={
        'block_type':'type_8',
        'f_params':Target_Size_feet_params,
        'isopleth_values':[['x']],
                }

main_params={
        'filename':'Type3-Color.pdf',
        'paper_height':11.0,
        'paper_width':25.0,
        'block_params':[block_1_params,block_2_params,
                        block_3_params,block_4_params],
        'transformations':[('rotate',0.01),('scale paper',)],
        'title_x':10.0,
        'title_y':-1.5,
        'title_box_width': 20.0,
        'title_color':color.cmyk.RedOrange,
        'title_str':r'\Large Resolution =\
}                   ----------------------------------------',
        'extra_texts':[
              {'x':9.5,
               'y':-1.2,
               'text':r'\Large $N$ $\times$ Range',
               'width':5,
               'pyx_extra_defs':[color.cmyk.RedOrange],
               },
               {'x':7.9,
               'y':-1.85,
               'text':r'\Large Focal Length $\times$ Target Size',
               'width':7,
               'pyx_extra_defs':[color.cmyk.RedOrange],
              }],
        'isopleth_params':[
              {'color':'RedOrange',
               'linewidth':'THick',
```

```
 'linestyle':'dotted',
 'circle_size':0.10,
 'transparency':0.5,
},
```

```
                ],
                }

Nomographer(main_params)
```



Let's return to our previous Type 8 example of a dial face. First we want to add rainbow colors for the scales. To each Type 8 scale we add the lines

```
'axis_color':color.cmyk.Red,
'text_color':color.cmyk.Red,
'title_color':color.cmyk.Red,
```

where the color is different for the scales at different radii.

In the previous black-and-white version of this dial face on page 34, many of the tick marks are too short. Here we help the situation with the following new scale parameters that can be applied to **any** scale, curved or straight:

- *text_size_#*, where # can be 0, 1, 2, 3 or 4: the size of the scale labels for particular tick text levels, for example, `'text_size_0': text.size.small,` (defaults for #=0–4 are small, scriptsize, tiny, tiny and tiny, respectively)
- *text_distance_#*, where # can be 0, 1, 2, 3 or 4: the distance in cm of the scale label for a particular text level # to the scale, for example, `'text_distance_0':1.2,` (defaults for #=0–4 are 1.0, 0.25, 0.25, 0.25 and 0.25, respectively)
- *full_angle*: allow the scale labels to turn upside down as they run around a curved scale, rather than forcing them to stay more-or-less upright within a range of 0° to 180° as seen in the dial face on page 34 (False by default)
- *extra_angle*: an angle in degrees to tilt the scale labels clockwise from their default angle. For a vertical scale a value of 90 would align the labels along the direction of the scale, in which case you should also set *text_horizontal_align_center* below to True
- *text_horizontal_align_center*: if True, line up the left-right center of each scale label with its tick mark rather than the end of the label, useful when the labels are rotated 90° as seen in the next example (False by default)
- *grid_length_#*, where # can be 0, 1, 2, 3, 4 or 5: the length in cm of the tick mark of a particular tick level for example, `'grid_length_2':0.150,` (defaults for #=0–4 are 0.75, 0.225, 0.125, 0.075 and 0.075, respectively)

As before, we continue to set *turn_relative* to True so the tick side stays continuous by following the scale curvature rather than being based on the left/right side of the page. We also set the tick marks to be more uniform in length. We orient the labels along the scale (90°) and allow them to turn upside down (this is particularly useful if the dial is physically turned to line up with a fixed, vertical index line). We also define the label sizes. The result is a more readable dial face than the previous version on page 34.

Zooming in on the dial face shows how it would look when printed on large paper. The script and output PDF graphic are available online for download.[31]

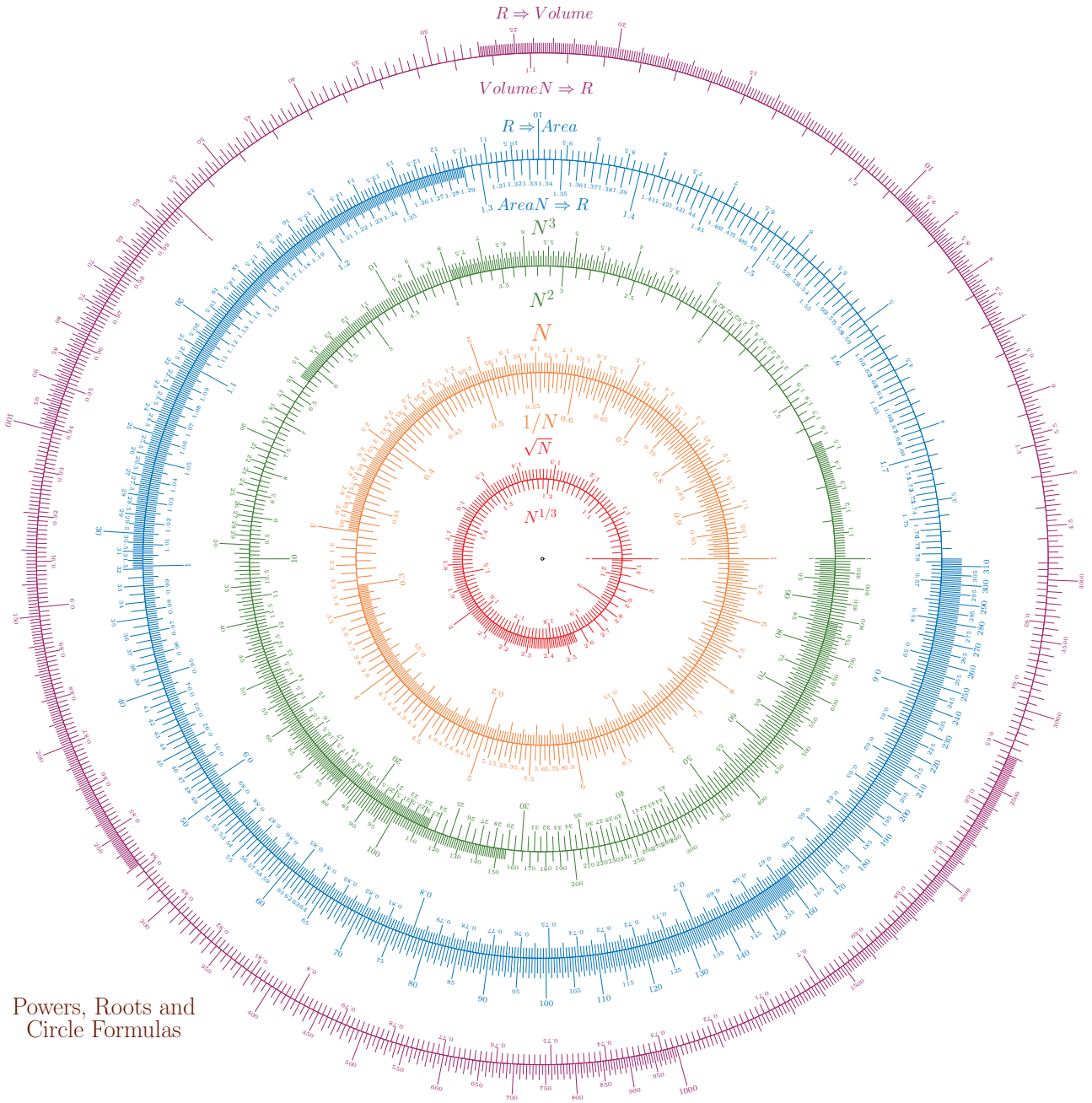
[31] *http://www.myreckonings.com/pynomo11/Type8-ColorDial-Smart.py* and *http://www.myreckonings.com/pynomo11/Type8-ColorDial-Smart.pdf*

```
### Type8-ColorDial-Smart.py ###

from pynomo.nomographer import *

CenterHole_params={
    'u_min':0.0,
    'u_max':1.0,
    'function_x':lambda u:0.05*cos(2*pi*u),
    'function_y':lambda u:0.05*sin(2*pi*u),
    'title':r'',
    'tick_levels':0,
    'tick_text_levels':0,
    }

CenterHole_block_params={
    'block_type':'type_8',
    'f_params':CenterHole_params,
    'width':15.0,
    'height':15.0,
    }

Cube_Root_params={
    'u_min':1.0,
    'u_max':2.15,
    'base_start':1.0,
    'base_stop':10.0,
    'function_x':lambda u:
          3.0*cos(2*pi*(log10(u)-log10(1.0))/
          (log10(2.1544)-log10(1.0))),
    'function_y':lambda u:
          3.0*sin(2*pi*(log10(u)-log10(1.0))/
          (log10(2.1544)-log10(1.0))),
    'title':r'\Large $N^{1/3}$',
    'title_x_shift':-0.05,
    'title_y_shift':-1.3,
    'tick_levels':5,
    'tick_text_levels':4,
    'scale_type':'log smart',
    'tick_side':'left',
    'turn_relative':True,
    # new script added below
    'grid_length_0':1.20,
    'grid_length_1':0.90,
    'grid_length_2':0.6,
    'grid_length_3':0.45,
    'grid_length_4':0.3,
    'grid_length_5':0.25,
    'text_distance_0':1.25,
    'text_distance_1':0.95,
    'text_distance_2':0.65,
    'text_distance_3':0.45,
    'text_distance_4':0.35,
    'text_distance_5':0.3,
    'text_size_0': text.size.scriptsize,
    'text_size_1': text.size.tiny,
    'extra_angle':90,
    'text_horizontal_align_center':True,
    'full_angle':True,
    'axis_color':color.cmyk.Red,
    'text_color':color.cmyk.Red,
    'title_color':color.cmyk.Red,
    }

Cube_Root_block_params={
    'block_type':'type_8',
    'f_params':Cube_Root_params,
    'width':15.0,
    'height':15.0,
    }

Square_Root_params={
    'u_min':1.0,
    'u_max':3.1623,
    'base_start':1.0,
    'base_stop':10.0,
    'function_x':lambda u:
          3.0*cos(2*pi*(log10(u)-log10(1.0))/
          (log10(3.1623)-log10(1.0))),
    'function_y':lambda u:
          3.0*sin(2*pi*(log10(u)-log10(1.0))/
          (log10(3.1623)-log10(1.0))),
    'title':r'\Large $\sqrt{N}$',
    'title_x_shift':-0.1,
    'title_y_shift':0.75,
    'tick_levels':5,
    'tick_text_levels':4,
    'scale_type':'log smart',
    'turn_relative':True,
    # new script added below
    'grid_length_0':1.20,
    'grid_length_1':0.90,
    'grid_length_2':0.6,
    'grid_length_3':0.45,
    'grid_length_4':0.3,
    'grid_length_5':0.25,
    'text_distance_0':1.25,
    'text_distance_1':0.95,
    'text_distance_2':0.65,
    'text_distance_3':0.45,
    'text_distance_4':0.35,
    'text_distance_5':0.3,
    'text_size_0': text.size.scriptsize,
    'text_size_1': text.size.tiny,
    'extra_angle':90,
    'text_horizontal_align_center':True,
    'full_angle':True,
    'axis_color':color.cmyk.Red,
    'text_color':color.cmyk.Red,
    'title_color':color.cmyk.Red,
    }

Square_Root_block_params={
    'block_type':'type_8',
    'f_params':Square_Root_params,
    'width':15.0,
    'height':15.0,
    }

Inverse_N_params={
    'u_min':0.101,
    'u_max':1.0,
    'base_start':0.1,
    'base_stop':1.0,
    'function_x':lambda u:
          7.0*cos(-2*pi*(log10(u)-log10(0.1))/
          (log10(1.0)-log10(0.1))),
    'function_y':lambda u:
          7.0*sin(-2*pi*(log10(u)-log10(0.1))/
          (log10(1.0)-log10(0.1))),
    'title':r'\LARGE $1/N$',
    'title_x_shift':-0.2,
    'title_y_shift':-1.65,
    'tick_levels':5,
    'tick_text_levels':2,
    'scale_type':'linear smart',
    'turn_relative':True,
    # new script added below
    'grid_length_0':1.20,
    'grid_length_1':0.90,
    'grid_length_2':0.6,
```

```
    'grid_length_3':0.45,
    'grid_length_4':0.3,
    'grid_length_5':0.25,
    'text_distance_0':1.25,
    'text_distance_1':0.95,
    'text_distance_2':0.65,
    'text_distance_3':0.45,
    'text_distance_4':0.35,
    'text_distance_5':0.3,
    'text_size_0': text.size.scriptsize,
    'text_size_1': text.size.tiny,
    'extra_angle':90,
    'text_horizontal_align_center':True,
    'full_angle':True,
    'axis_color':color.cmyk.Orange,
    'text_color':color.cmyk.Orange,
    'title_color':color.cmyk.Orange,
    }

Inverse_N_block_params={
    'block_type':'type_8',
    'f_params':Inverse_N_params,
    'width':15.0,
    'height':15.0,
    }

N_params={
    'u_min':1.0,
    'u_max':9.9999,
    'base_start':1.0,
    'base_stop':9.9999,
    'function_x':lambda u:
          7.0*cos(2*pi*log10(u)/log10(10.0)),
    'function_y':lambda u:
          7.0*sin(2*pi*log10(u)/log10(10.0)),
    'title':r'\huge $N$',
    'title_x_shift':-0.15,
    'title_y_shift':0.95,
    'tick_levels':5,
    'tick_text_levels':4,
    'scale_type':'linear smart',
    'turn_relative':True,
    # new script added below
    'grid_length_0':1.20,
    'grid_length_1':0.90,
    'grid_length_2':0.6,
    'grid_length_3':0.45,
    'grid_length_4':0.3,
    'grid_length_5':0.25,
    'text_distance_0':1.25,
    'text_distance_1':0.95,
    'text_distance_2':0.65,
    'text_distance_3':0.45,
    'text_distance_4':0.35,
    'text_distance_5':0.3,
    'text_size_0': text.size.scriptsize,
    'text_size_1': text.size.tiny,
    'extra_angle':90,
    'text_horizontal_align_center':True,
    'full_angle':True,
    'axis_color':color.cmyk.Orange,
    'text_color':color.cmyk.Orange,
    'title_color':color.cmyk.Orange,
    }

N_block_params={
    'block_type':'type_8',
    'f_params':N_params,
    'width':15.0,
    'height':15.0,
    }

Square_params={
    'u_min':1.0,
    'u_max':99.999,
    'base_start':1.0,
    'base_stop':100.0,
    'function_x':lambda u:
          11.0*cos(2*pi*(log10(u)-log10(1.0))/
          (log10(100.0)-log10(1.0))),
    'function_y':lambda u:
          11.0*sin(2*pi*(log10(u)-log10(1.0))/
          (log10(100.0)-log10(1.0))),
    'title':r'\LARGE $N^2$',
    'title_x_shift':-0.5,
    'title_y_shift':-1.2,
    'tick_levels':5,
    'tick_text_levels':4,
    'scale_type':'linear smart',
    'tick_side':'left',
    'turn_relative':True,
    # new script added below
    'grid_length_0':1.20,
    'grid_length_1':0.90,
    'grid_length_2':0.6,
    'grid_length_3':0.45,
    'grid_length_4':0.3,
    'grid_length_5':0.25,
    'text_distance_0':1.25,
    'text_distance_1':0.95,
    'text_distance_2':0.65,
    'text_distance_3':0.45,
    'text_distance_4':0.35,
    'text_distance_5':0.3,
    'text_size_0': text.size.scriptsize,
    'text_size_1': text.size.tiny,
    'extra_angle':90,
    'text_horizontal_align_center':True,
    'full_angle':True,
    'axis_color':color.cmyk.OliveGreen,
    'text_color':color.cmyk.OliveGreen,
    'title_color':color.cmyk.OliveGreen,
    }

Square_block_params={
    'block_type':'type_8',
    'f_params':Square_params,
    'width':15.0,
    'height':15.0,
    }

Cube_params={
    'u_min':1.0,
    'u_max':999.999,
    'base_start':1.0,
    'base_stop':1000.0,
    'function_x':lambda u:
          11.0*cos(2*pi*(log10(u)-log10(1.0))/
          (log10(1000.0)-log10(1.0))),
    'function_y':lambda u:
          11.0*sin(2*pi*(log10(u)-log10(1.0))/
          (log10(1000.0)-log10(1.0))),
    'title':r'\LARGE $N^3$',
    'title_x_shift':2.2,
    'title_y_shift':1.2,
    'tick_levels':5,
    'tick_text_levels':4,
    'scale_type':'log smart',
    'turn_relative':True,
    # new script added below
```

```
    'grid_length_0':1.20,
    'grid_length_1':0.90,
    'grid_length_2':0.6,
    'grid_length_3':0.45,
    'grid_length_4':0.3,
    'grid_length_5':0.25,
    'text_distance_0':1.25,
    'text_distance_1':0.95,
    'text_distance_2':0.65,
    'text_distance_3':0.45,
    'text_distance_4':0.35,
    'text_distance_5':0.3,
    'text_size_0': text.size.scriptsize,
    'text_size_1': text.size.tiny,
    'extra_angle':90,
    'text_horizontal_align_center':True,
    'full_angle':True,
    'axis_color':color.cmyk.OliveGreen,
    'text_color':color.cmyk.OliveGreen,
    'title_color':color.cmyk.OliveGreen,
    }

Cube_block_params={
    'block_type':'type_8',
    'f_params':Cube_params,
    'width':15.0,
    'height':15.0,
    }

Inverse_Area_params={
    'u_min':0.5642,
    'u_max':1.7841,
    'base_start':0.5642,
    'base_stop':1.7841,
    'function_x':lambda u:
          15.0*cos(-2*pi*(log10(u)-log10(0.5642))/
          (log10(1.7841)-log10(0.5642))),
    'function_y':lambda u:
          15.0*sin(-2*pi*(log10(u)-log10(0.5642))/
          (log10(1.7841)-log10(0.5642))),
    'title':r'\Large $Area N \Rightarrow R$',
    'title_x_shift':0.0,
    'title_y_shift':-1.5,
    'tick_levels':5,
    'tick_text_levels':3,
    'scale_type':'linear smart',
    'turn_relative':True,
    # new script added below
    'grid_length_0':1.20,
    'grid_length_1':0.90,
    'grid_length_2':0.6,
    'grid_length_3':0.45,
    'grid_length_4':0.3,
    'grid_length_5':0.25,
    'text_distance_0':1.25,
    'text_distance_1':0.95,
    'text_distance_2':0.65,
    'text_distance_3':0.45,
    'text_distance_4':0.35,
    'text_distance_5':0.3,
    'text_size_0': text.size.scriptsize,
    'text_size_1': text.size.tiny,
    'extra_angle':90,
    'text_horizontal_align_center':True,
    'full_angle':True,
    'axis_color':color.cmyk.RoyalBlue,
    'text_color':color.cmyk.RoyalBlue,
    'title_color':color.cmyk.RoyalBlue,
    }

Inverse_Area_block_params={
    'block_type':'type_8',
    'f_params':Inverse_Area_params,
     'width':15.0,
     'height':15.0,
     }

Area_params={
    'u_min':3.14159,
    'u_max':314.15,
    'base_start':3.14159,
    'base_stop':314.15,
    'function_x':lambda u:
          15.0*cos(2*pi*(log10(u)-log10(3.14159))/
          (log10(314.159)-log10(3.14159))),
    'function_y':lambda u:
          15.0*sin(2*pi*(log10(u)-log10(3.14159))/
          (log10(314.159)-log10(3.14159))),
    'title':r'\Large $R \Rightarrow Area$',
    'title_x_shift':-0.75,
    'title_y_shift':0.85,
    'tick_levels':5,
    'tick_text_levels':4,
    'scale_type':'linear smart',
    'turn_relative':True,
    # new script added below
    'grid_length_0':1.20,
    'grid_length_1':0.90,
    'grid_length_2':0.6,
    'grid_length_3':0.45,
    'grid_length_4':0.3,
    'grid_length_5':0.25,
    'text_distance_0':1.25,
    'text_distance_1':0.95,
    'text_distance_2':0.65,
    'text_distance_3':0.45,
    'text_distance_4':0.35,
    'text_distance_5':0.3,
    'text_size_0': text.size.scriptsize,
    'text_size_1': text.size.tiny,
    'extra_angle':90,
    'text_horizontal_align_center':True,
    'full_angle':True,
    'axis_color':color.cmyk.RoyalBlue,
    'text_color':color.cmyk.RoyalBlue,
    'title_color':color.cmyk.RoyalBlue,
    }

Area_block_params={
    'block_type':'type_8',
    'f_params':Area_params,
    'width':15.0,
    'height':15.0,
    }

Inverse_Volume_params={
    'u_min':0.62035,
    'u_max':1.3365,
    'base_start':0.1,
    'base_stop':1.0,
    'function_x':lambda u:
          19.0*cos(-2*pi*(log10(u)-log10(0.62035))/
          (log10(1.3365)-log10(0.62035))),
    'function_y':lambda u:
          19.0*sin(-2*pi*(log10(u)-log10(0.62035))/
          (log10(1.3365)-log10(0.62035))),
    'title':r'\Large $Volume N \Rightarrow R$',
    'title_x_shift':-0.15,
    'title_y_shift':-1.2,
    'tick_levels':5,
```

```
    'tick_text_levels':4,
    'scale_type':'log smart',
    'turn_relative':True,
    # new script added below
    'grid_length_0':1.20,
    'grid_length_1':0.90,
    'grid_length_2':0.6,
    'grid_length_3':0.45,
    'grid_length_4':0.3,
    'grid_length_5':0.25,
    'text_distance_0':1.25,
    'text_distance_1':0.95,
    'text_distance_2':0.65,
    'text_distance_3':0.45,
    'text_distance_4':0.35,
    'text_distance_5':0.3,
    'text_size_0': text.size.scriptsize,
    'text_size_1': text.size.tiny,
    'extra_angle':90,
    'text_horizontal_align_center':True,
    'full_angle':True,
    'axis_color':color.cmyk.RedViolet,
    'text_color':color.cmyk.RedViolet,
    'title_color':color.cmyk.RedViolet,
    }

Inverse_Volume_block_params={
    'block_type':'type_8',
    'f_params':Inverse_Volume_params,
    'width':15.0,
    'height':15.0,
    }

Volume_params={
    'u_min':4.1888,
    'u_max':4188.8,
    'base_start':1.0,
    'base_stop':1000.0,
    'function_x':lambda u:
          19.0*cos(2*pi*(log10(u)-log10(4.1888))/
          (log10(4188.8)-log10(4.1888))),
    'function_y':lambda u:
          19.0*sin(2*pi*(log10(u)-log10(4.1888))/
          (log10(4188.8)-log10(4.1888))),
    'title':r'\Large $R \Rightarrow Volume$',
    'title_x_shift':3.8,
    'title_y_shift':1.5,
    'tick_levels':5,
    'tick_text_levels':3,
    'scale_type':'log smart',
    'turn_relative':True,
    # new script added below
    'grid_length_0':1.20,
    'grid_length_1':0.90,
    'grid_length_2':0.6,
    'grid_length_3':0.45,
    'grid_length_4':0.3,
    'grid_length_5':0.25,
    'text_distance_0':1.25,
    'text_distance_1':0.95,
    'text_distance_2':0.65,
    'text_distance_3':0.45,
    'text_distance_4':0.35,
    'text_distance_5':0.3,
    'text_size_0': text.size.scriptsize,
    'text_size_1': text.size.tiny,
    'extra_angle':90,
    'text_horizontal_align_center':True,
    'full_angle':True,
    'axis_color':color.cmyk.RedViolet,
    'text_color':color.cmyk.RedViolet,
    'title_color':color.cmyk.RedViolet,
    }

Volume_block_params={
    'block_type':'type_8',
    'f_params':Volume_params,
    'width':15.0,
    'height':15.0,
    }

main_params={
    'filename':'Type8-ColorDial-Smart.pdf',
    'paper_height':30.0,
    'paper_width':30.0,
    'block_params':[CenterHole_block_params,
        Cube_Root_block_params,Square_Root_block_params,
        Inverse_N_block_params,N_block_params,
        Square_block_params,Cube_block_params,
        Inverse_Area_block_params,Area_block_params,
        Inverse_Volume_block_params,Volume_block_params,],
    'transformations':[('rotate',0.01),('scale paper',)],
    'title_x': 2,
    'title_y': 1.5,
    'title_box_width': 6,
    'title_color':color.cmyk.Sepia,
    'title_str':r'\huge Powers, Roots and Circle Formulas',
    }

Nomographer(main_params)
```



Powers, Roots and Circle Formulas

Notice that in this last color dial script we repeat a certain set of scale parameters in nearly every scale. Normally a nomogram would consist of fewer scales, so repeating the parameters is not an issue, but there is a way to define a set of default parameters that can be loaded into each scale. The scale can then override any default parameter by simpy re-defining it. This can save some lines in the script.

The default parameters are listed at the beginning of the script in a new set of parameters called *default_params*. Each scale that loads these defaults has a line of script that copies them in, and then the script uses an update command to define the rest of its parameters (or re-define any default parameters). Below is the first part of the previous color dial script that has been modified for default values. The CenterHole scale does not use these default parameters at all, so it does not use the copy and update mechanism. The Cube_Root scale, like all the other scales, does load the defaults, so `Cube_Root_params=copy.deepcopy(default_params)` is added prior to defining the scale parameters. The scale parameters name now has `.update` added at the end, which is not included when the scale is included in the block parameters, and the list of parameters in braces is now inserted between the parentheses of the update function. The parameters for this scale are now expressed as `Cube_Root_params.update({...})` rather than `Cube_Root_params={...}`, and this procedure is followed for all the other scales.

To reduce page count, the full script and PDF graphic output are not listed here, but they are available online for download.[32] You will see that the nomogram is identical to our previous color dial but the script size has been reduced from 519 to 360 lines.

```
### Type8-ColorDial-Smart-Defaults.py ###

from pynomo.nomographer import *

default_params={
    'grid_length_0':1.20,
    'grid_length_1':0.90,
    'grid_length_2':0.6,
    'grid_length_3':0.45,
    'grid_length_4':0.3,
    'grid_length_5':0.25,
    'text_distance_0':1.25,
    'text_distance_1':0.95,
    'text_distance_2':0.65,
    'text_distance_3':0.45,
    'text_distance_4':0.35,
    'text_distance_5':0.3,
    'text_size_0': text.size.scriptsize,
    'text_size_1': text.size.tiny,
    'extra_angle':90,
    'text_horizontal_align_center':True,
    'full_angle':True,
    'turn_relative':True,
    }

CenterHole_params={
    'u_min':0.0,
    'u_max':1.0,
    'function_x':lambda u:0.05*cos(2*pi*u),
    'function_y':lambda u:0.05*sin(2*pi*u),
    'title':r'',
    'tick_levels':0,
    'tick_text_levels':0,
    }

CenterHole_block_params={
    'block_type':'type_8',
    'f_params':CenterHole_params,
    'width':15.0,
    'height':15.0,
    }

Cube_Root_params=copy.deepcopy(default_params)
Cube_Root_params.update({
    'u_min':1.0,
    'u_max':2.15,
    'base_start':1.0,
    'base_stop':10.0,
    'function_x':lambda u:
          3.0*cos(2*pi*(log10(u)-log10(1.0))/
          (log10(2.1544)-log10(1.0))),
    'function_y':lambda u:
          3.0*sin(2*pi*(log10(u)-log10(1.0))/
          (log10(2.1544)-log10(1.0))),
    'title':r'\Large $N^{1/3}$',
    'title_x_shift':-0.05,
    'title_y_shift':-1.3,
    'tick_levels':5,
    'tick_text_levels':4,
    'scale_type':'log smart',
    'tick_side':'left',
    'axis_color':color.cmyk.Red,
    'text_color':color.cmyk.Red,
    'title_color':color.cmyk.Red,
    })

Cube_Root_block_params={
    'block_type':'type_8',
    'f_params':Cube_Root_params,
    'width':15.0,
    'height':15.0,
    }

... and so forth ...
```

[32] *http://www.myreckonings.com/pynomo11/Type8-ColorDial-Smart-Defaults.py* and *http://www.myreckonings.com/pynomo11/Type8-ColorDial-Smart-Defaults.pdf*

12 Compounding Type 1 (Three Parallel Scale) Nomograms

We've seen in earlier sections nomograms with pivot lines for an equation of the form $f_1(u_1) + f_2(u_2) + \ldots + f_n(u_n) = 0$. This is a Type 3 nomogram directly supported by PyNomo. However, there are times when the scale values on the pivot lines are important. For example, consider another relationship in electro-optics between the camera Photocathode Illuminance, Scene Exitance (the reflected light off the target) and the Lens F/No. when the lens transmission is 90% and the image magnification on the focal plane is approximately zero:

$$\text{Photocathode Illuminance} = \frac{\text{Scene Exitance} \times 90\%}{4 \times \text{Lens F/No.}^2}$$

But we also have a relationship between the Scene Exitance, Target Reflectivity, and Scene Illuminance:

$$\text{Scene Exitance} = \text{Scene Illuminance} \times \text{Target Reflectivity}$$

So we have two three-variable equations sharing one variable. If substitute the second equation into the first to get a single equation, we can use the Type 3 format with pivot lines to draw it (after converting it to a sum of logarithms) but we've lost the intermediate variable Scene Exitance which is an important parameter of the engineering problem.

Therefore, for each equation we need to create a separate Type 1 three-line nomogram block, and then tag the Scene Exitance scale in each of the two nomogram blocks with the same name so they align on top of each other. This is simply a procedure to connect two Type 1 nomograms, so there is no separate Type number in PyNomo for this combination.

The example for the equations above is shown on the next two pages. It defines two blocks of Type 1, with the tag *exitance* assigned to the Scene Exitance scale in each block. Only one of these blocks provides the title, and one has the tick marks on the left side and the other on the right. The *main* parameters list both blocks as part of the nomogram. The scale parameter *extra_params* is used to place additional arrows at specific scale values by specifying *manual arrow* as the *scale_type*. We also introduce a new parameter here:

- *arrow_length*: the length in cm of the arrows (default is 1.0)

Here the second line of the title is printed with the *extra_texts* parameter.

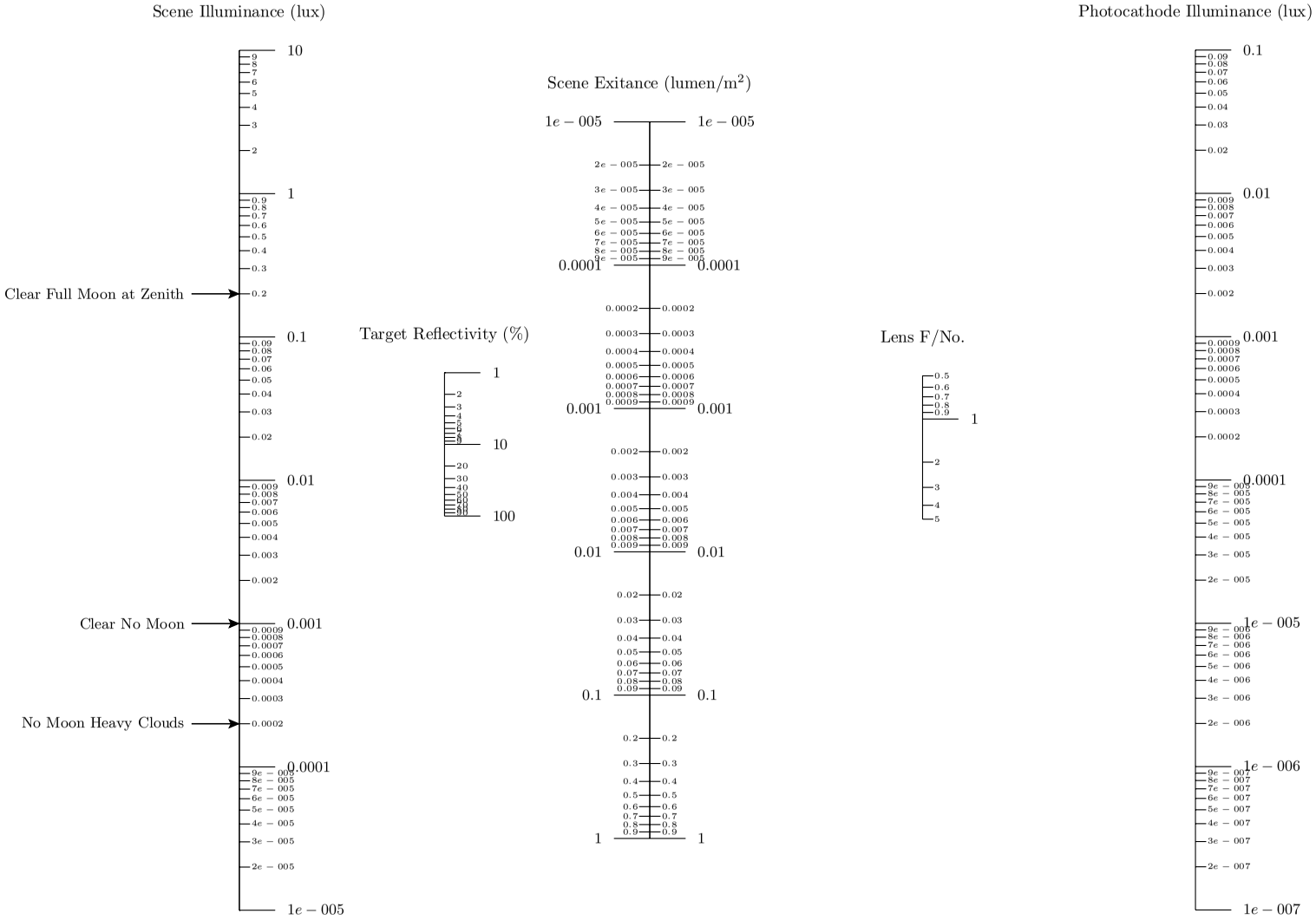
The tick marks and labels are printed on the Scene Exitance scale by both Type 1 blocks in order to make sure that they are identically aligned. Once we see that this is the case, we can comment out these lines in one of the blocks so we only have one set of tick marks. The result is shown in the second example on the third and fourth pages after this, where *tick_levels* and *tick_text_levels* are set to 0 for the Scene Exitance scale in the second block. The tick labels also seem excessively dense in the first example, so the *tick_text_levels* parameters are reduced to 1 instead of 2. As you can see, the second nomogram is much cleaner than the first one without any loss of precision.

These PyNomo scripts and output PDF graphics are available online for download.[33] [34]

[33] *http://www.myreckonings.com/pynomo11/CompoundType1-Example.py*
and *http://www.myreckonings.com/pynomo11/CompoundType1-Example.pdf*

[34] *http://www.myreckonings.com/pynomo11/CompoundType1-Final.py*
and *http://www.myreckonings.com/pynomo11/CompoundType1-Final.pdf*

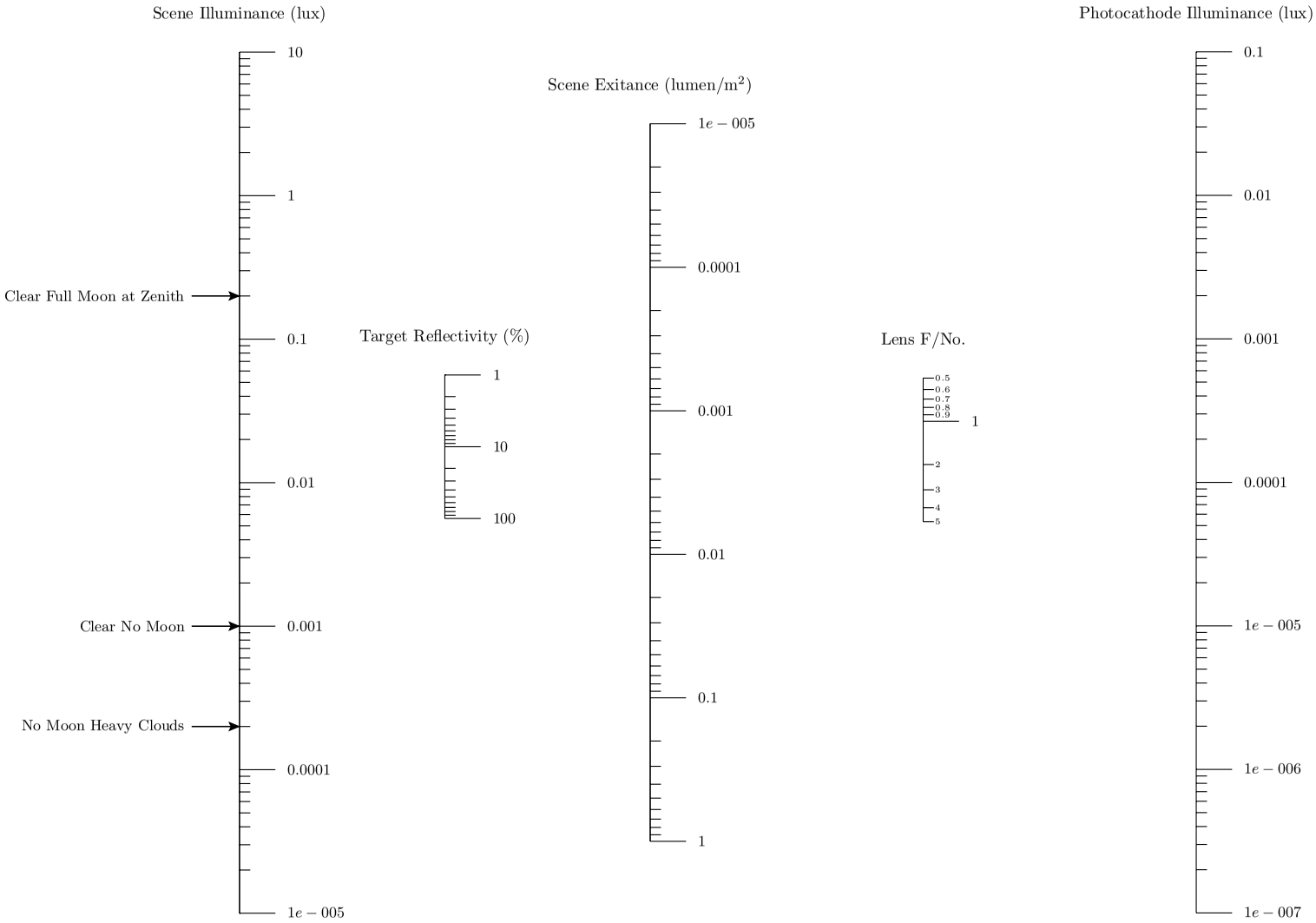
```
### CompoundType1-Example.py ###

from pynomo.nomographer import *

### TYPE 1 3-LINE BLOCK ###
Photocathode_Illuminance_params={
        'u_min':0.1,
        'u_max':0.0000001,
        'function':lambda u:log10(u),
        'scale_type':'log',
        'title':r'Photocathode Illuminance (lux)',
        'title_y_shift':0.7,
        'tick_side':'left',
        'tick_levels':2,
        'tick_text_levels':2,
                }

FNumber_params={
        'u_min':0.5,
        'u_max':5.0,
        'function':lambda u:2*log10(u),
        'scale_type':'log',
        'title':r'Lens F/No.',
        'title_y_shift':0.7,
        'tick_levels':2,
        'tick_text_levels':2,
                }

Scene_Exitance_1_params={
        'tag':'exitance',
        'u_min':0.00001,
        'u_max':1.0,
        'function':lambda u:-log10(0.225*u),
        'scale_type':'log',
        'title':r'Scene Exitance (lumen/m$^2$)',
        'title_y_shift':0.7,
        'tick_side':'right',
        'tick_levels':2,
        'tick_text_levels':2,
                }

block_11_params={
        'block_type':'type_1',
        'width':15.0,
        'height':15.0,
        'f1_params':Scene_Exitance_1_params,
        'f2_params':FNumber_params,
        'f3_params':Photocathode_Illuminance_params,
                }


### TYPE 1 3-LINE BLOCK ###
Scene_Illuminance_params={
        'u_min':0.00001,
        'u_max':10.0,
        'function':lambda u:-log10(u),
        'scale_type':'log',
        'title':r'Scene Illuminance (lux)',
        'title_y_shift':0.7,
        'tick_levels':2,
        'tick_text_levels':2,
        'extra_params':[
              {'tick_side':'left',
               'scale_type':'manual arrow',
               'arrow_length':1.0,
               'manual_axis_data':
                 {
                   0.2:'Clear Full Moon at Zenith',
                   0.001:'Clear No Moon',
                   0.0002:'No Moon Heavy Clouds',
                 },
             }],
               }

Target_Reflectivity_params={
        'u_min':1.0,
        'u_max':100.0,
        'function':lambda u:-log10(0.01*u),
        'scale_type':'log',
        'title':r'Target Reflectivity (\%)',
        'title_y_shift':0.7,
        'tick_levels':2,
        'tick_text_levels':2,
                }

Scene_Exitance_2_params={
        'tag':'exitance',
        'u_min':0.00001,
        'u_max':1.0,
        'function':lambda u:log10(u),
        'scale_type':'log',
        'title':r'',
        'title_y_shift':0.7,
        'tick_side':'left',          # Duplicate labels
        'tick_levels':2,             # Duplicate labels
        'tick_text_levels':2,        # Duplicate labels
                }

block_12_params={
        'block_type':'type_1',
        'width':15.0,
        'height':15.0,
        'f1_params':Scene_Exitance_2_params,
        'f2_params':Target_Reflectivity_params,
        'f3_params':Scene_Illuminance_params,
                }


### MAIN PARAMETERS ###
main_params={
        'filename':'CompoundType1-Example.pdf',
        'paper_height':18.0,
        'paper_width':20.0,
        'block_params':[block_11_params,block_12_params],
        'transformations':[('rotate',0.01),('scale paper',)],
        'title_x':8.5,
        'title_y':-1,
        'title_box_width':25.0,
        'title_str':r'\large Photocathode Illuminance = \
            (Scene Exitance $\times$ 90\% Lens Transmission)\
            / (4 $\times$ Lens F/No.$^2$ $\times$\
               (1 $+$ 0.0 Magnification))',\
        'extra_texts':[
              {'x':2.0,
               'y':-1.7,
               'text':r'\large where Scene Exitance = Scene\
                 Illuminance $\times$ Target Reflectivity',
               'width':15.0,
              }],
                }

Nomographer(main_params)
```



Photocathode Illuminance = (Scene Exitance × 90% Lens Transmission) / (4 × Lens F/No.^{2} × (1 + 0.0 Magnification))

where Scene Exitance = Scene Illuminance × Target Reflectivity

```
### CompoundType1-Final.py ###

from pynomo.nomographer import *

### TYPE 1 3-LINE BLOCK ###
Photocathode_Illuminance_params={
        'u_min':0.1,
        'u_max':0.0000001,
        'function':lambda u:log10(u),
        'scale_type':'log',
        'title':r'Photocathode Illuminance (lux)',
        'title_y_shift':0.7,
        'tick_side':'left',
        'tick_levels':2,
        'tick_text_levels':1,
                }

FNumber_params={
        'u_min':0.5,
        'u_max':5.0,
        'function':lambda u:2*log10(u),
        'scale_type':'log',
        'title':r'Lens F/No.',
        'title_y_shift':0.7,
        'tick_levels':2,
        'tick_text_levels':2,
                }

Scene_Exitance_1_params={
        'tag':'exitance',
        'u_min':0.00001,
        'u_max':1.0,
        'function':lambda u:-log10(0.225*u),
        'scale_type':'log',
        'title':r'Scene Exitance (lumen/m$^2$)',
        'title_y_shift':0.7,
        'tick_side':'right',
        'tick_levels':2,
        'tick_text_levels':1,
                }

block_11_params={
        'block_type':'type_1',
        'width':15.0,
        'height':15.0,
        'f1_params':Scene_Exitance_1_params,
        'f2_params':FNumber_params,
        'f3_params':Photocathode_Illuminance_params,
                }


### TYPE 1 3-LINE BLOCK ###
Scene_Illuminance_params={
        'u_min':0.00001,
        'u_max':10.0,
        'function':lambda u:-log10(u),
        'scale_type':'log',
        'title':r'Scene Illuminance (lux)',
        'title_y_shift':0.7,
        'tick_levels':2,
        'tick_text_levels':1,
        'extra_params':[
              {'tick_side':'left',
               'scale_type':'manual arrow',
               'arrow_length':1.0,
               'manual_axis_data':
                 {
                   0.2:'Clear Full Moon at Zenith',
                   0.001:'Clear No Moon',
                   0.0002:'No Moon Heavy Clouds',
                 },
             }],
               }

Target_Reflectivity_params={
        'u_min':1.0,
        'u_max':100.0,
        'function':lambda u:-log10(0.01*u),
        'scale_type':'log',
        'title':r'Target Reflectivity (\%)',
        'title_y_shift':0.7,
        'tick_levels':2,
        'tick_text_levels':1,
                }

Scene_Exitance_2_params={
        'tag':'exitance',
        'u_min':0.00001,
        'u_max':1.0,
        'function':lambda u:log10(u),
        'scale_type':'log',
        'title':r'',
        'title_y_shift':0.7,
        'tick_side':'left',
        'tick_levels':0,      # Don't draw duplicate ticks
        'tick_text_levels':0, # Don't draw duplicate labels
                }

block_12_params={
        'block_type':'type_1',
        'width':15.0,
        'height':15.0,
        'f1_params':Scene_Exitance_2_params,
        'f2_params':Target_Reflectivity_params,
        'f3_params':Scene_Illuminance_params,
                }


### MAIN PARAMETERS ###
main_params={
        'filename':'CompoundType1-Final.pdf',
        'paper_height':18.0,
        'paper_width':20.0,
        'block_params':[block_11_params,block_12_params],
        'transformations':[('rotate',0.01),('scale paper',)],
        'title_x':8.5,
        'title_y':-1,
        'title_box_width':25.0,
        'title_str':r'\large Photocathode Illuminance = \
            (Scene Exitance $\times$ 90\% Lens Transmission)\
            / (4 $\times$ Lens F/No.$^2$ $\times$\
               (1 $+$ 0.0 Magnification))',\
        'extra_texts':[
              {'x':2.0,
               'y':-1.7,
               'text':r'\large where Scene Exitance = Scene\
                 Illuminance $\times$ Target Reflectivity',
               'width':15.0,
              }],
                }

Nomographer(main_params)
```

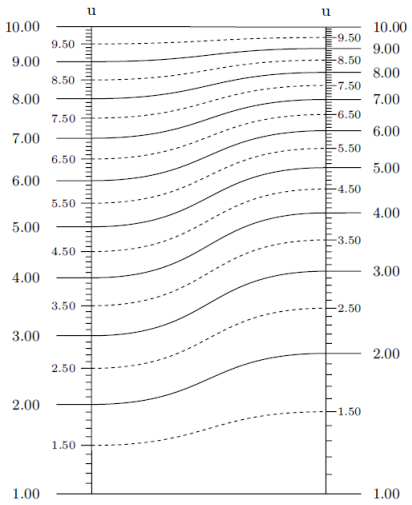


Photocathode Illuminance = (Scene Exitance × 90% Lens Transmission) / (4 × Lens F/No.^{2} × (1 + 0.0 Magnification))

where Scene Exitance = Scene Illuminance × Target Reflectivity

13 Creating a Type 6 (Ladder) Nomogram

A Type 6 Ladder nomogram is a simple connection of two function scales by lines running between matching values on the scales. It provides a transition between two nomograms that share a common variable, but which cannot be represented by the same function. This is normally avoided if possible by redesigning the nomogram, but there are occasions where it is needed. As Epstein says, "It is admittedly a crutch but is widely used." A ladder example is provided on the Type 6 Software Documentation page.[35], and we will simply modify the script on that page for our example here.



We will create a nomogram for the following equation from Epstein.[36]

$$R = S^T$$

where

$$Q = R + W$$

So we have two three-variable equations sharing one variable just as we plotted in the last section as two compounded Type 1 nomograms. As before, we don't want to substitute the first equation into the second to get a single equation because we want to see the intermediate variable R.

So again for each equation we need to create a separate nomogram block. The first equation represents a Type 2 N nomogram for $f_1(u_1) = f_2(u_2) \times f_3(u_3)$ after we take the logarithm of both sides and use the relation $\log a^b = b \log a$:

$$\log R = \log S^T$$
$$\log R = T \log S$$
$$\log S = {}^1/_T \log R$$

The second equation represents a Type 1 nomogram for simple addition $f_1(u_1) + f_2(u_2) + f_3(u_3) = 0$:

$$-R - W + Q = 0$$

So it turns out that it's impossible to share the same R-scale for the two blocks because R must be a logarithmic scale for the first equation and a linear scale for the second equation. The example for the equations above is shown next.

The new block parameters for this Type 6 nomogram (which are omitted in this example to accept their default values) are

- *curve_const*: sets the length of the angle of the Bezier curve, where a low value is a straighter line and a high value is a more curved line (default=0.5)—not set in this example
- *ladder_color*: the color of the ladder lines (default is *color.rgb.black*)—not set in this example

[35] *http://www.pynomo.org/wiki/index.php/Type_6*
[36] Epstein, L. Ivan. **Nomography**. Interscience Publishers, New York (1958), p. 66.

The order of blocks and the order of scales in each block are important for arranging the scales, and the tick side for the ladder determines which side the curves originate on, so you may have to build the nomogram up in pieces and experiment to get these correct. I found I had to set the *mirror_x* block parameter to True in the block parameters for the ladder. The T-scale was broken into two parts using the *extra_params* scale parameter in order to swap the label sides for better readability. Notice also that the T-scale, which has a $1/u$ function associated with it, is drawn as a log scale, which reduces the size of the labels other than 0.10, 1.00 and 10.00 so they don't collide with each other.

We have drawn isopleths on this nomogram, which always produces straight lines connecting the two sides of the ladder. If we had omitted the isopleth commands the connections would have been curved lines as in the example above (which I prefer, but I wanted to show how to add isopleths to a Type 6 nomogram). Notice that since we defined the isopleth intersection value at the ladder end of the first block, the ladder block must include an *isopleth_values* parameter in which the two values are the calculated (denoted by 'x').

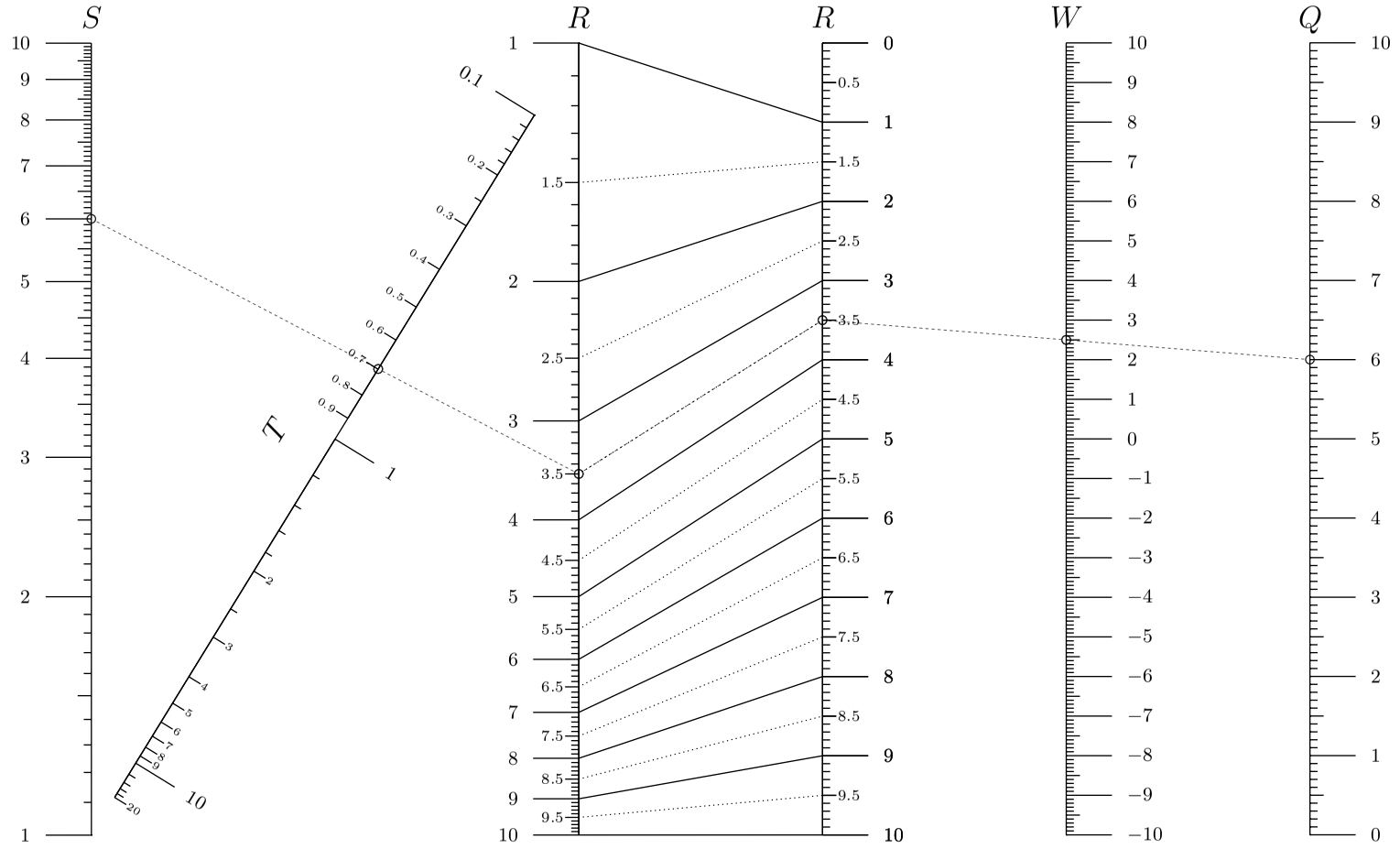
The PyNomo scripts and output PDF graphics are available online for download.[37]

```
### Type6-Isopleths.py ###

from pynomo.nomographer import *

### Type 2 Block ###
S_params={
        'u_min':1.0,
        'u_max':10.0,
        'function':lambda u:log(u),
        'title':r'\Large $S$',
        'tick_levels':3,
        'tick_text_levels':1,
        'tick_side':'left',
               }

T_params={
        'u_min':0.1,
        'u_max':20.0,
        'function':lambda u:1.0/u,
        'scale_type':'log',
        'title':r'\Large $T$',
        'tick_levels':0,
        'tick_text_levels':0,
        'title_distance_center':0.75,
        'title_draw_center':True,
        'extra_params':[
             {
             'u_min':0.1,
             'u_max':0.9,
             'tick_levels':3,
             'tick_text_levels':2,
             'tick_side':'left',
             },
             {'u_min':1.0,
              'u_max':20.0,
              'tick_levels':4,
              'tick_text_levels':2,
              'tick_side':'right'
              },
             ]
                }

R1_params={
        'tag':'r1-scale',
        'u_min':1.0,
        'u_max':10.0,
        'function':lambda u:log(u),
        'title':r'\Large $R$',
        'tick_levels':0,
        'tick_text_levels':0,
               }

Type2_params={
        'block_type':'type_2',
        'width':10.0,
        'height':10.0,
        'f1_params':S_params,
        'f2_params':T_params,
        'f3_params':R1_params,
        'isopleth_values':[[6,'x',3.5]],
              }

### Type 6 Ladder Block with Aligned Scales ###
R1_Ladder_params={
        'tag':'r1-scale',
        'u_min':1.0,
        'u_max':10.0,
        'function':lambda u:log(u),
        'title':r'',
        'tick_levels':3,
        'tick_text_levels':2,
        'tick_side':'left',
        }

R2_Ladder_params={
        'tag':'r2-scale',
        'u_min':0.0,
        'u_max':10.0,
        'function':lambda u:u,
        'title':r'',
        'tick_levels':3,
        'tick_text_levels':2,
        }

Ladder_params={
        'block_type':'type_6',
```

[37] *http://www.myreckonings.com/pynomo11/Type6-Isopleths.py* and *http://www.myreckonings.com/pynomo11/Type6-Isopleths.pdf*

```
        'f1_params':R1_Ladder_params,
        'f2_params':R2_Ladder_params,
        'width':5.0,
        'height':10.0,
        'mirror_x':True,
        'isopleth_values':[['x','x']],
        }

### Type 1 3-Scale Block ###
R2_params={
        'tag':'r2-scale',
        'u_min':0.0,
        'u_max':10.0,
        'function':lambda u:-u,
        'title':r'\Large $R$',
        'tick_levels':2,
        'tick_text_levels':1,
                }

W_params={
        'u_min':-10.0,
        'u_max':10.0,
        'function':lambda u:-u,
        'title':r'\Large $W$',
        'tick_levels':3,
        'tick_text_levels':1,
                }

Q_params={
        'u_min':0.0,
        'u_max':10.0,
        'function':lambda u:u,
        'title':r'\Large $Q$',
        'tick_levels':3,
        'tick_text_levels':1,
                }
```

```
Type3_params={
        'block_type':'type_1',
        'width':10.0,
        'height':10.0,
        'f1_params':R2_params,
        'f2_params':W_params,
        'f3_params':Q_params,
        'isopleth_values':[[3.5,'x',6]],
               }

main_params={
        'filename':'Type6-Isopleths.pdf',
        'paper_height':13.0,
        'paper_width':20.0,
        'block_params':[Type2_params,Ladder_params,Type3_params],
        'transformations':[('rotate',0.01),('scale paper',)],
        'title_x':10.0,
        'title_y':-1.5,
        'title_box_width': 5.0,
        'title_str':r'\LARGE $R = S^T$',
              'extra_texts':[
                     {'x':7.6,
                      'y':-2.7,
                      'text':'\LARGE $Q = R + W$',
                      'width':5,
                      }],
        'isopleth_params':[
              {'color':'black',
               'linewidth':'thin',
               'linestyle':'dashed',
               'circle_size':0.07,
               'transparency':0.25,
              },
              ],
               }

Nomographer(main_params)
```


S
R
R
W
Q
T
0.1
1
10
$R = S^T$
$Q = R + W$

14 Creating a Type 4 (Proportion) Nomogram

A Type 4 nomogram represents a proportional relationship between four functions of individual variables. The form of equation for this nomogram is

$$\frac{f_1(u_1)}{f_2(u_2)} = \frac{f_3(u_3)}{f_4(u_4)}$$

We will use this type of nomogram to graph the Law of Sines for a triangle:

$$\frac{A}{\sin\alpha} = \frac{B}{\sin\beta}$$

where A and B are two sides of a triangle and α and β are the vertex angles opposite A and B. All three sides have this relationship $A/\sin\alpha = B/\sin\beta = C/\sin\gamma$, so the complete triangle can be defined if two angles and a side are known, a common problem when doing triangulation as part of a survey.

Again, we can simply modify the Type 4 example given on the PyNomo site.[38] The following two pages show the script and nomogram. To use it, a straightedge is placed across the known values of, say, A and α in degrees, and a pencil mark is made on the diagonal pivot line. Then the straightedge is placed across the third known value (say B) and the mark, and the fourth value β is found by its intersection with that scale. Once β is found, we can find the third angle as $\gamma = 180^\circ - \alpha - \beta$, and then we can use that angle on either scale and the same mark on the diagonal to find the remaining side C.

We are using here the *title_draw_center* scale parameter that moves the title along the scale to its center, and the *title_distance_center* that places it a distance from the scale (by default 0.5cm but I felt the α and β symbols were too close to the scale with this default value). I also felt that the degree sign was too close to those symbols, so I added a thin space (\thinspace) between them. The *title_opposite_tick* parameter places the title on the opposite side of the scale from the ticks. The \ symbol preceding sin and cos in the title simply prints them out as non-italic text even within math mode (i.e., within $ signs).

As described in the section on color on page 35, we have also set the *reference_color* parameter in *block_1_params* to define the color of the diagonal reference line (to orange here).

There are two new block parameters for this Type 4 nomogram that are omitted in this example to accept their default values:

- *float_axis*: if set to *'F1 or F2'* scaling is done based on those two scales, otherwise it's based on the F3 and F4 scales (default is *'F1 or F2'*)—not set in this example
- *padding*: the length of the scales as a fraction of the width and height, providing some space between the upper ends of the scales (default = 0.9)—not set in this example

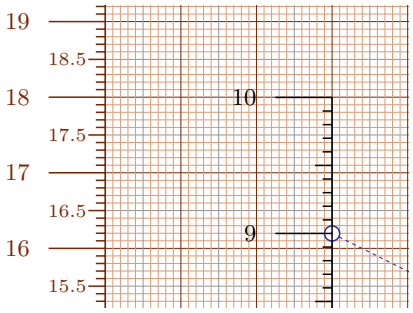
Earlier we saw how to add an isopleth to a nomogram as a guide for using it. This example shows how a pair of sample isopleths are added to a proportion nomogram.

Adding a Key

Another very common guide, of particular use when the isopleth or isopleths make the nomogram harder to read or use, is to provide a *key* in the nomogram. A key is a small graphic showing the scale lines and

[38] *http://www.pynomo.org/wiki/index.php/Type_4*

sample isopleths (and sometimes there is helpful text as well). In this example the *line_params* entries in the main parameters produce a key for the use of the nomogram. The key is drawn inside the nomogram because there's room for it there. There's no reason to include both the sample isopleths and the key on a nomogram, but they are both included here simply as a demonstration. This nomogram is shown with only the key included in the example on page 98.



The new parameters introduced here for creating lines for the key can be included in the main parameters of any nomogram type:

- *draw_lines*: if True, draw lines on the nomogram according to the *line_params* parameters (False by default)
- *line_params*: the characteristics of lines drawn on the nomogram (start/end coordinates, style, circles on endpoints) as described below
- *make_grid*: if True, overlay a grid in cm and mm over the nomogram so that start and end coordinates of lines can be easily determined by zooming in, as shown in the figure on this page (False by default)

The *line_params* parameter allows the following sub-parameters:

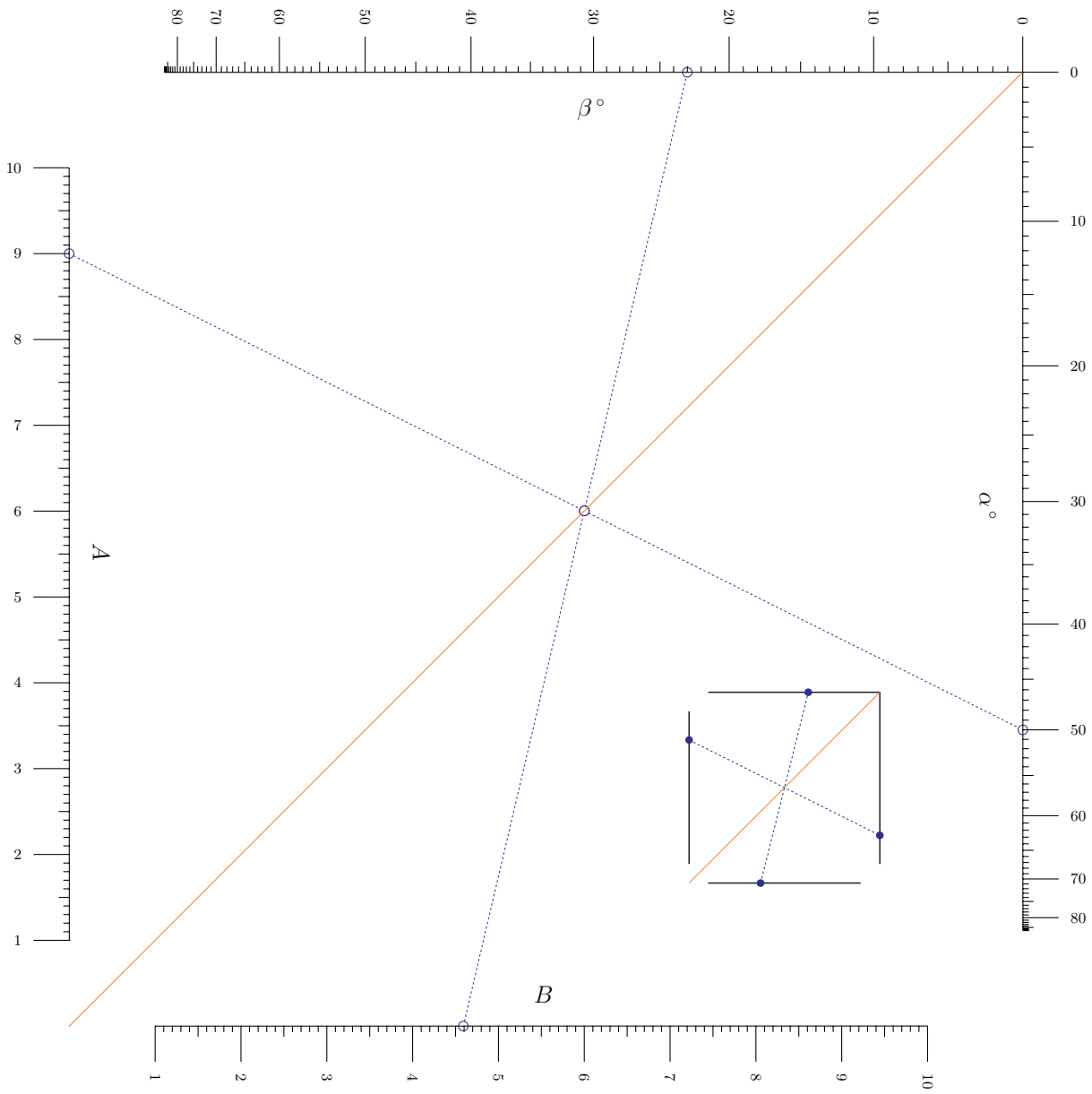
- *coords*: the x-y coordinates from the grid overlay of one or more line segments, the starting and ending coordinates specified in square brackets for each segment, separated by commas. Connected segments should include the ending coordinates of the previous segment as the starting coordinates of the new segment, as in `'coords':[[0,6.33,6.67,4.82],[6.67,4.82,13.33,7.0],[13.33,7.0,20.0,6.29]],`
- *line_style*: the style of the line segments, consisting of their color (e.g., `color.cmyk.Gray`), linewidth (e.g., `style.linewidth.thin`), and form (e.g., `style.linestyle.dashed`), with defaults of black, thick and dashed
- *circle_size*: the radius in cm of circles centered on the starting and ending points of the line segments, with a default of 0.0 for no circles
- *circle_color*: the color of circles at the starting and ending points of the line segments, with a default of black

There can be multiple *line_params* defined, each with different sub-parameters. The default values support standard black, dashed lines with no circles at their endpoints, in which case only *coords* needs to be defined. The format of the color sub-parameters are described in the earlier section devoted to color. The linewidth and linestyle names are provided in the figure on page 18. The example script on the next page shows values for all the sub-parameters to demonstrate their use.

This PyNomo script and output PDF graphic are available online for download.[39]

[39] *http://www.myreckonings.com/pynomo11/Type4-Isopleths.py* and *http://www.myreckonings.com/pynomo11/Type4-Isopleths.pdf*

```
### Type4-Isopleths.py ###

from pynomo.nomographer import *

A_params={
      'u_min':1.0,
      'u_max':10.0,
      'function':lambda u:u,
      'title':r'\Large $A$',
      'tick_levels':3,
      'tick_text_levels':1,
      'tick_side':'left',
      'title_draw_center':True,
      'title_opposite_tick':True,
             }
Alpha_params={
      'u_min':0.0,
      'u_max':89.0,
      'function':lambda u:sin(u*pi/180.0),
      'title':r'\Large $\alpha \thinspace ^\circ$',
      'tick_levels':3,
      'tick_text_levels':1,
      'tick_side':'right',
      'title_draw_center':True,
      'title_distance_center':0.9,
      'title_opposite_tick':True,
             }
B_params={
      'u_min':1.0,
      'u_max':10.0,
      'function':lambda u:u,
      'title':r'\Large $B$',
      'tick_levels':3,
      'tick_text_levels':1,
      'tick_side':'right',
      'title_draw_center':True,
      'title_opposite_tick':False,
             }
Beta_params={
      'u_min':0.0,
      'u_max':89.0,
      'function':lambda u:sin(u*pi/180.0),
      'title':r'\Large $\beta \thinspace ^\circ$',
      'tick_levels':3,
      'tick_text_levels':1,
      'tick_side':'left',
      'title_draw_center':True,
      'title_distance_center':0.9,
      'title_opposite_tick':False,
             }

block_1_params={
      'block_type':'type_4',
      'reference_color':color.cmyk.Orange,
      'f1_params':A_params,
      'f2_params':Alpha_params,
      'f3_params':B_params,
      'f4_params':Beta_params,
      'isopleth_values':[[9,50,'x',23]],
             }

main_params={
      'filename':'Type4-Isopleths.pdf',
      'paper_height':20.0,
      'paper_width':20.0,
      'block_params':[block_1_params],
      'transformations':[('rotate',0.01),('scale paper',)],
      'title_x':10,
      'title_y': -3.5,
      'title_box_width': 10,
      'title_str':r'\LARGE Law of Sines:  \
                    $A / \sin \alpha=B / \sin \beta$',
      'isopleth_params':[
           {'color':'Blue',
            'linewidth':'thin',
            'linestyle':'dashed',
            'circle_size':0.1,
            'transparency':0.0,
           },
           ],
      'make_grid':False,
      'draw_lines':True,
      'line_params': [
          {'coords':[[13,3.4,13,6.6],[17,3.4,17,7],
                     [13.4,3,16.6,3],[13.4,7,17,7]],
           'line_style':[color.cmyk.Black,
                         style.linewidth.thick,
                         style.linestyle.solid],
           'circle_size':0.0,
           'circle_color':color.cmyk.Black,
          },
          {'coords':[[13,3,17,7]],
           'line_style':[color.cmyk.Orange,
                         style.linewidth.thin,
                         style.linestyle.solid],
           'circle_size':0.0,
           'circle_color':color.cmyk.Orange,
          },
          {'coords':[[13,6,17,4],[14.5,3,15.5,7]],
           'line_style':[color.cmyk.Blue,
                         style.linewidth.thin,
                         style.linestyle.dashed],
           'circle_size':0.08,
           'circle_color':color.cmyk.Blue,
          },
          ],
              }

Nomographer(main_params)
```



Law of Sines: $A/\sin\alpha = B/\sin\beta$

15 Creating a Type 7 (Reciprocal or Angle) Nomogram

A Type 7 nomogram represents functions that are related by their reciprocals. The form of this nomogram is

$$\frac{1}{f_3(u_3)} = \frac{1}{f_1(u_1)} + \frac{1}{f_2(u_2)}$$

Here we will create a classic nomogram of this form for the equivalent resistance R of two resistors placed in parallel:

$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2}$$

This relationship also exists for capacitors in series. Another common occurrence is the equivalent focal length of lenses in series, and there are many others.

Again, we can simply modify the Type 7 example given on the PyNomo site.[40] To extend this example we will draw a nomogram for the equivalent resistance of either two or three resistors in parallel, where it can be seen from extending the above formula that $\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3}$. We simply create two Type 7 blocks with the u_3 scale of the first block aligned by a tag with the u_1 block of the second block.

The following two pages show the script and nomogram. If we decide to scale it to a larger paper size, the scales will likely not be the same length anymore and we would have to adjust the paper width or length if we want them the same length (although they would work fine otherwise). A straightedge across R_1 and R_2 crosses the middle scale $R_1 \parallel R_2$ at the value of the equivalent resistance of these two resistors. Then if there is a third resistor the straightedge is placed across that point on the $R_1 \parallel R_2$ scale and the R_3 scale, with the intersection on the $R_1 \parallel R_2 \parallel R_3$ scale (which happens to line up with the R_2 scale because we chose 60° angles) providing the equivalent resistance of all three resistors.

We are introducing the following new parameters specific to a Type 7 nomogram:

- *angle_u*: the angle in degrees between the u_1 scale and the center u_3 scale
- *angle_v*: the angle in degrees between the u_2 scale and the center u_3 scale

When both angles are set to 60° the scales are identical on all three axes of a block, the most common implementation of this type of nomogram. I was pleased to see that the two angles can be set independently, as I've spent quite a bit of time in the past trying to find out how to design a reciprocal nomogram with unequal angles.

The \parallel TeX command produces the symbol for parallel in the scale titles. You can also see that the x offset in the *extra_titles* parameter for the second row of the title is set to a negative number, as $x = 0$ for an extra title lines up with the center of the first title.

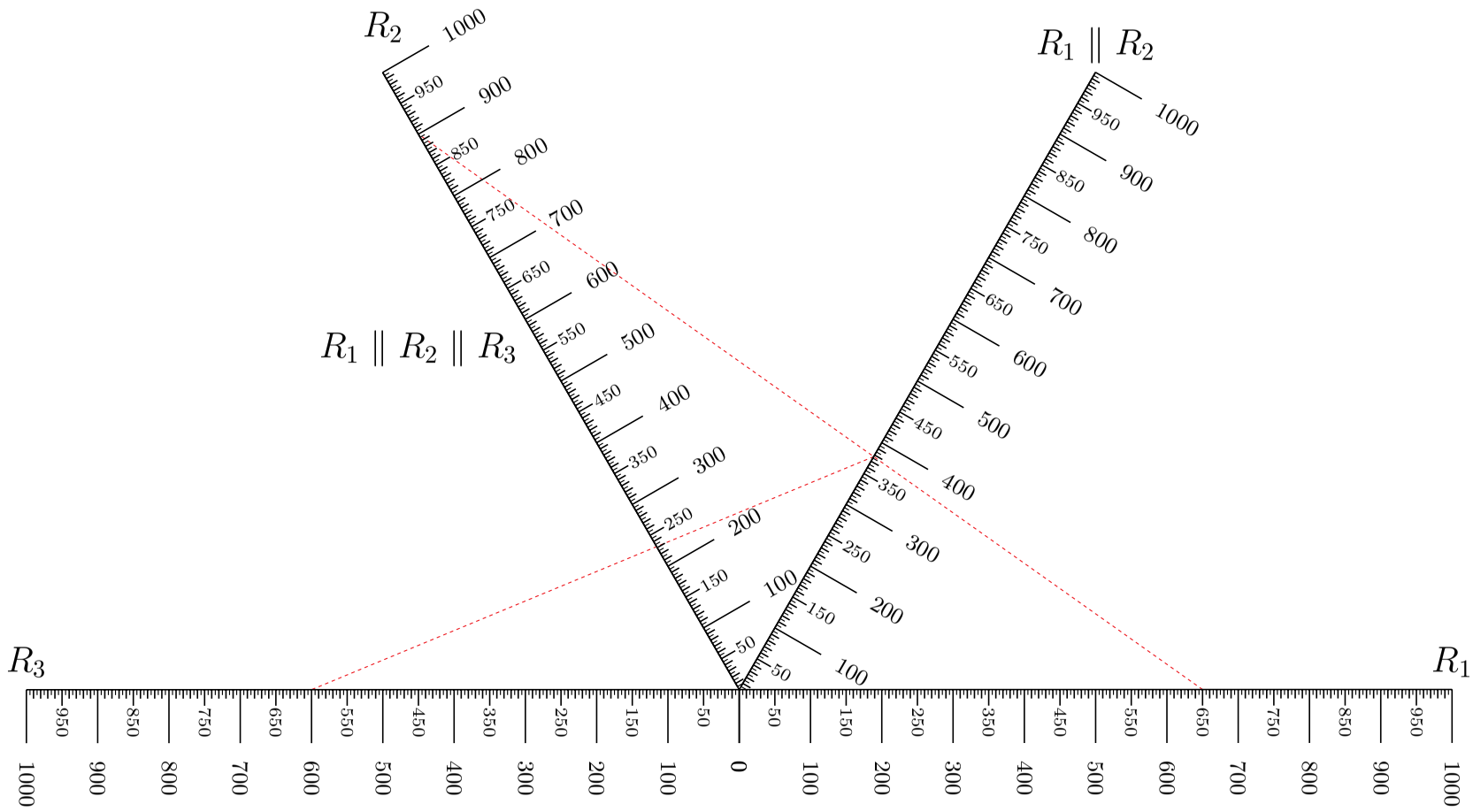
I'm rather tempted to add two more of these aligned blocks to get a full snowflake nomogram! Then we could continue going around the snowflake as many times as needed for any number of resistors to find the equivalent resistance.

This PyNomo script and output PDF graphic are available online for download.[41]

[40] *http://www.pynomo.org/wiki/index.php/Type_7*

[41] *http://www.myreckonings.com/pynomo11/Type7-Isopleths.py* and *http://www.myreckonings.com/pynomo11/Type7-Isopleths.pdf*

```
### Type7-Isopleths.py ###

from pynomo.nomographer import *

R1_params={
      'u_min':0.0,
      'u_max':1000.0,
      'function':lambda u:u,
      'title':r'\Large $R_1$',
      'text_format':r"$%4.0f$",
      'tick_levels':4,
      'tick_text_levels':2,
            }

R2_params={
      'u_min':5.0,
      'u_max':1000.0,
      'function':lambda u:u,
      'title':r'\Large $R_2$',
      'text_format':r"$%4.0f$",
      'title_y_shift':0.5,
      'tick_levels':4,
      'tick_text_levels':2,
            }

R1R2_params_1={
      'tag':'r1r2',
      'u_min':5.0,
      'u_max':1000.0,
      'function':lambda u:u,
      'title':r'\Large $R_1 \parallel R_2$',
      'text_format':r"$%4.0f$",
      'tick_levels':4,
      'tick_text_levels':2,
            }


block_1_params={
      'block_type':'type_7',
      'width':20.0,
      'height':20.0,
      'angle_u':60,
      'angle_v':60,
      'f1_params':R1_params,
      'f2_params':R2_params,
      'f3_params':R1R2_params_1,
      'isopleth_values':[[650,900,'x']],
            }

R1R2_params_2={
      'tag':'r1r2',
      'u_min':0.0,
      'u_max':1000.0,
      'function':lambda u:u,
      'title':r'',
      'text_format':r"$%4.0f$",
      'tick_levels':0,
      'tick_text_levels':0,
            }

R3_params={
      'u_min':0.0,
      'u_max':1000.0,
      'function':lambda u:u,
      'title':r'\Large $R_3$',
      'text_format':r"$%4.0f$",
      'title_x_shift':-10.0,
      'tick_levels':4,
      'tick_text_levels':2,
            }

R1R2R3_params={
      'u_min':0.0,
      'u_max':1000.0,
      'function':lambda u:u,
      'title':r'\Large $R_1 \parallel R_2 \parallel R_3$',
      'text_format':r"$%4.0f$",
      'title_x_shift':0.5,
      'title_y_shift':-4.0,
      'tick_levels':0,
      'tick_text_levels':0,
            }


block_2_params={
      'block_type':'type_7',
      'width':20.0,
      'height':20.0,
      'angle_u':60,
      'angle_v':60,
      'f1_params':R1R2_params_2,
      'f2_params':R3_params,
      'f3_params':R1R2R3_params,
      'isopleth_values':[['x',600,'x']],
            }

main_params={
      'filename':'Type7-Isopleths.pdf',
      'paper_height':20.0,
      'paper_width':20.0,
      'block_params':[block_1_params,block_2_params],
      'transformations':[('rotate',0.01)],
      'title_x': 0.0,
      'title_y': -3.0,
      'title_box_width': 12,
      'title_str':r'\Large Equivalent Resistance of 3\
                   Resistors in Parallel',
      'extra_texts':[
            {'x':-3.5,
             'y':-3.7,
             'text':r'\large $1/R = 1/R_1 + 1/R_2 + 1/R_3$',
             'width':12.0,
            }],
      'isopleth_params':[
           {'color':'Red',
            'linewidth':'thin',
            'linestyle':'dashed',
            'circle_size':0.0,
            'transparency':0.0,
           },
           ],
              }

Nomographer(main_params)
```



Equivalent Resistance of 3 Resistors in Parallel

$$1/R = 1/R_1 + 1/R_2 + 1/R_3$$

16 Creating a Type 10 (One Curved Line) Nomogram

A Type 10 nomogram consists of two straight scales and one curved scale. The form of this nomogram is $f_1(u) + f_2(v)f_3(w) + f_4(w) = 0$. We will use this type of nomogram to graph the equation

$$M = \frac{T - N}{T^2}$$

If we multiply both sides of this equation by T^2 and arrange the terms on the left side, we get

$$N + MT^2 - T = 0$$

From the general form above, you can see that this is a Type 10 nomogram where

$$\begin{aligned} u &= N \\ v &= M \\ w &= T \\ f_1(u) &= N \\ f_2(v) &= M \\ f_3(w) &= T^2 \\ f_4(w) &= -T \end{aligned}$$

Selecting the *Type 10* link on the Software Documentation page[42] shows an example script for this type of nomogram for the equation $u + vw + w = 0$. It's fairly simple to modify the example for what we need. However, the equation under consideration is not a friendly one, as the variable M can range from $-\infty$ to ∞, so the required ranges of the variables are very important in the design of the nomogram.

Let's assume a range of $-10 < N < 10$ and $-4 < T < 4$. Then M must range from $-\infty$ to ∞, so let's limit it to $-10 < M < 10$. The script and nomogram are found on the next page.

Now let's change our ranges to $0 < N < 10$ and $0 < T < 4$ and limit M to $-20 < M < 0$. This nomogram is shown on the second page following this one. Here PyNomo did much more for us than it first appears. As we will see in the next section when we plot this nomogram in determinant form, the T-scale would be expected to be roughly horizontal. But in addition to the scaling transformation to the paper size we've discussed earlier, PyNomo actually performed a shear on the T-scale and M-scale upward so the M-scale is aligned horizontally with the N-scale in order to maximize the size of the nomogram on the paper. (We also had to shift the T-scale title to place it correctly in the output.)

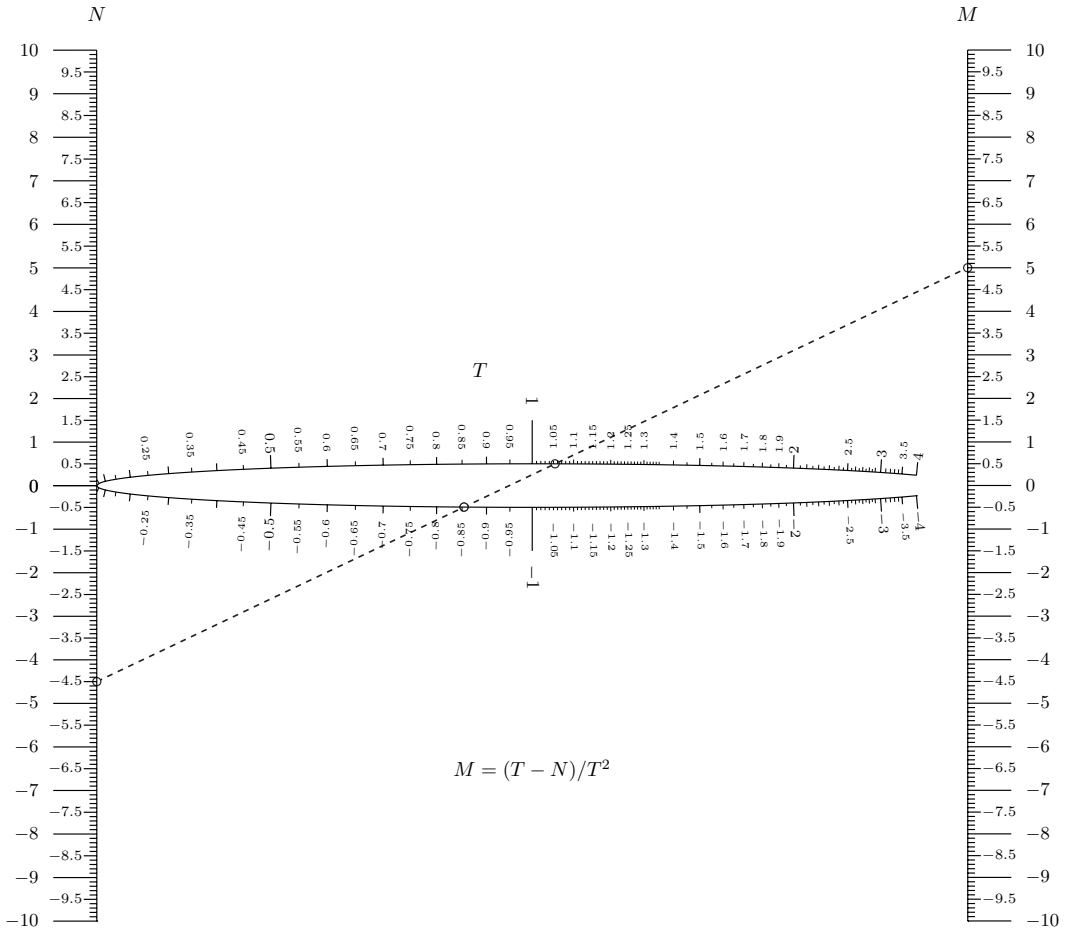
The PyNomo script and output PDF graphic for both nomograms are available online for download.[43][44]

[42] *http://pynomo.org/wiki/index.php?title=Type_10*

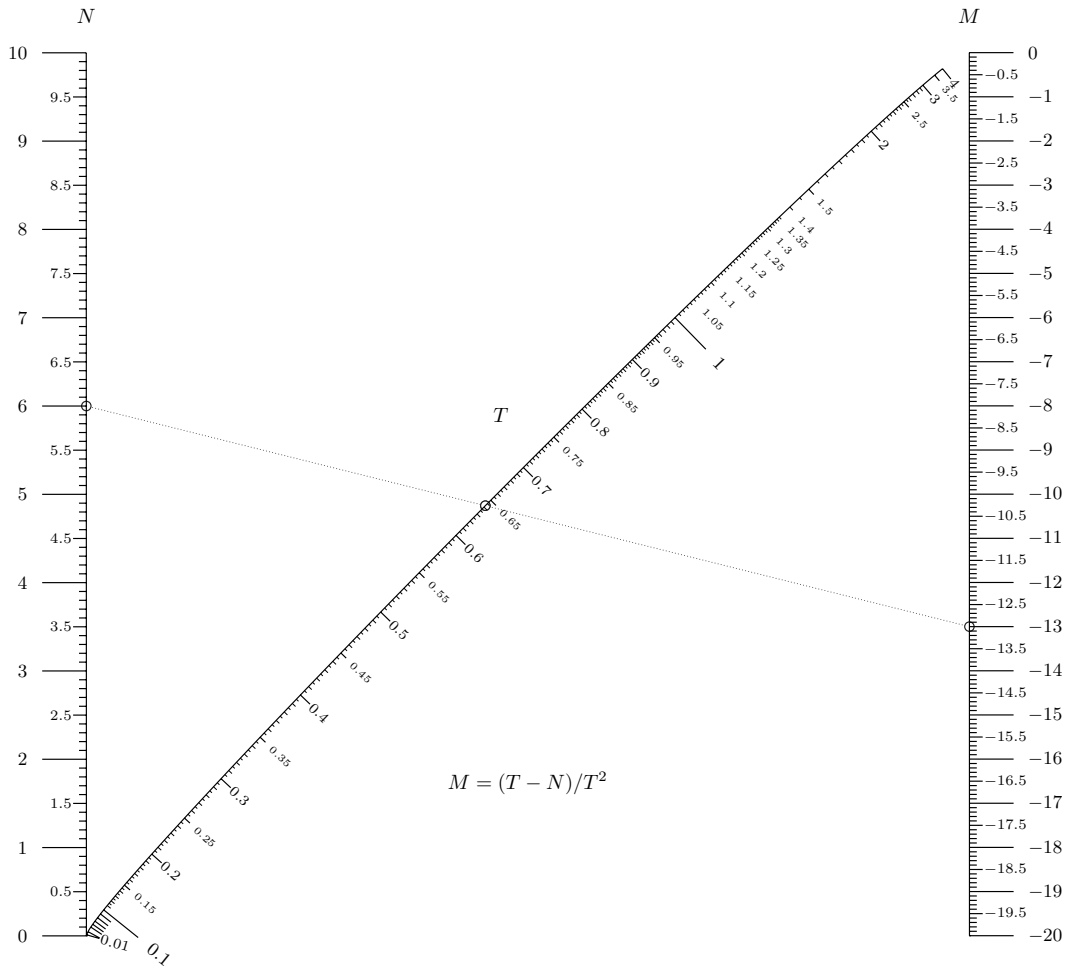
[43] *http://www.myreckonings.com/pynomo11/Type10-Range1.py* and *http://www.myreckonings.com/pynomo11/Type10-Range1.pdf*

[44] *http://www.myreckonings.com/pynomo11/Type10-Range2.py* and *http://www.myreckonings.com/pynomo11/Type10-Range2.pdf*

```
### Type10-Range1.py ###

from pynomo.nomographer import *

N_params={
        'u_min':-10.0,
        'u_max':10.0,
        'function':lambda u:u,
        'title':r'$N$',
        'title_y_shift':0.5,
        'tick_levels':3,
        'tick_text_levels':2,
        'tick_side':'left',
               }

M_params={
        'u_min':-10.0,
        'u_max':10.0,
        'function':lambda u:u,
        'title':r'$M$',
        'title_y_shift':0.5,
        'tick_levels':3,
        'tick_text_levels':2,
        'tick_side':'right',
               }

T_params={
        'u_min':-4.0,
        'u_max':4.0,
        'function_3':lambda u:u*u,
        'function_4':lambda u:-u,
        'scale_type':'log smart',
        'title':r'$T$',
        'title_x_shift':-1.0,
        'title_y_shift':1.5,
        'tick_levels':5,
        'tick_text_levels':5,
               }

block_1_params={
        'block_type':'type_10',
        'width':10.0,
        'height':10.0,
        'f1_params':N_params,
        'f2_params':M_params,
        'f3_params':T_params,
        'isopleth_values':[[-4.5,5.0,'x']],
             }

main_params={
         'filename':'Type10-Range1.pdf',
         'paper_height':15.0,
         'paper_width':15.0,
         'block_params':[block_1_params],
         'transformations':[('rotate',0.01),('scale paper',)],
         'title_y':2.5,
         'title_str':r'$M = (T - N) / T^2$',
         'isopleth_params':[
                     {'color':'black',
                      'linewidth':'thick',
                      'linestyle':'dashed',
                      'circle_size':0.07,
                      'transparency':0.0,
                     },
                    ],
               }
Nomographer(main_params)
```


N
M
T
M = (T − N)/T²

```
### Type10-Range2.py ###

from pynomo.nomographer import *

N_params={
        'u_min':0.0,
        'u_max':10.0,
        'function':lambda u:u,
        'title':r'$N$',
        'title_y_shift':0.5,
        'tick_levels':3,
        'tick_text_levels':2,
        'tick_side':'left',
                }

M_params={
        'u_min':-20.0,
        'u_max':0.0,
        'function':lambda u:u,
        'title':r'$M$',
        'title_y_shift':0.5,
        'tick_levels':3,
        'tick_text_levels':2,
        'tick_side':'right',
                }

T_params={
        'u_min':0.01,
        'u_max':4.0,
        'function_3':lambda u:u*u,
        'function_4':lambda u:-u,
        'scale_type':'log smart',
        'title':r'$T$',
        'title_x_shift':-7.5,
        'title_y_shift':-6.0,
        'tick_levels':5,
        'tick_text_levels':5,
                }

block_1_params={
        'block_type':'type_10',
        'width':10.0,
        'height':10.0,
        'f1_params':N_params,
        'f2_params':M_params,
        'f3_params':T_params,
        'isopleth_values':[[6,-13,'x']],
                }

main_params={
         'filename':'Type10-Range2.pdf',
         'paper_height':15.0,
         'paper_width':15.0,
         'block_params':[block_1_params],
         'transformations':[('rotate',0.01),('scale paper',)],
         'title_y':2.5,
         'title_str':r'$M = (T - N) / T^2$',
         'isopleth_params':[
                  {'color':'black',
                   'linewidth':'thin',
                   'linestyle':'dotted',
                   'circle_size':0.08,
                   'transparency':0.2,
                  },
                 ],
                }
Nomographer(main_params)
```

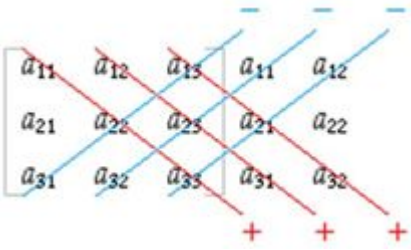

N
M
T
$M = (T - N)/T^2$

17 Creating a Type 9 (General Determinant) Nomogram

If an equation does not fall into one of the standard equation types supported directly by PyNomo, we can convert the equation into a general determinant form for PyNomo. This provides us with the ability to draw any nomogram, because an equation can be drawn as a classic nomogram (i.e, alignment chart) if and only if it can be put into determinant form.[45] As a benefit, there is generally no need to manipulate a determinant into Standard Nomographic Form for PyNomo as long as no variable appears in more than one row. PyNomo also supports equations with more than three variables in which case two variables share a row—this is called a grid nomogram. A grid nomogram is shown in the example of this type on the PyNomo site.[46]

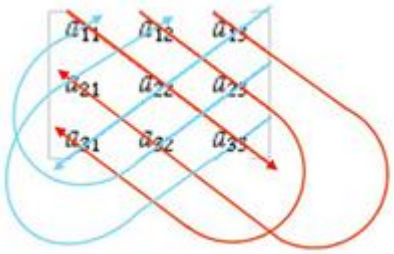
A determinant represents a particular operation on a matrix, and it is denoted by vertical bars on the sides of the matrix. The determinant of a 3x3 matrix is given by

$$a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - a_{13}a_{22}a_{31} - a_{11}a_{23}a_{32} - a_{12}a_{21}a_{33}$$



But there are visual ways of deriving this result. In the figure on the right the first two columns of the determinant are repeated to the right of the original, and then the products of all terms on diagonals from upper left to lower right are added and the products of all terms on diagonals from upper right to lower left are subtracted.

A convenient mental shortcut is to find these diagonal products by "wrapping around" to get the three components of each term. Here the first product we add is the main diagonal $a_{11}a_{22}a_{33}$, then the second is $a_{12}a_{23}a_{31}$ where we follow the curve around after we pick up a_{12} and a_{23} to pick up the a_{31}, then $a_{13}a_{32}a_{21}$ by starting at a_{13} and wrapping around to pick up the a_{32} and a_{21}. We do the same thing right-to-left for the subtracted terms. This is much easier to visualize than to describe. Determinants of larger matrices are not considered here.



Let's create the determinant form of the equation we discussed in the previous section, $N + MT^2 - T = 0$. By arranging terms in the determinant positions we can find this solution:

$$N + MT^2 - T = 0$$

$$\begin{vmatrix} T^2 & 1 & -T \\ 1 & 0 & -M \\ 0 & 1 & -N \end{vmatrix} = 0$$

If you multiply out this determinant this equation becomes $\{T^2 \times 0 \times (-N)\} + \{1 \times (-M) \times 0\} + \{-T \times 1 \times 1\} - \{-T \times 0 \times 0\} - \{T^2 \times 1 \times (-M)\} - \{1 \times 1 \times (-N)\} = 0$ which simplifies to our original equation, so this is a valid determinant to use for this equation.

Now historically a determinant such as this has to be manually transformed into Standard Nomographic Form in order to be plotted, in which the determinant has only 1's in the last column and no variable in more

[45] For more details, see *http://myreckonings.com/wordpress/2008/01/09/the-art-of-nomography-ii-designing-with-determinants/* or the webpages on the determinant forms of nomograms at *http://www.projectrho.com/nomogram/*

[46] *http://www.pynomo.org/wiki/index.php/Type_9*

than one row.[47] But PyNomo can handle a determinant like this that is not in standard form as long as no variable is in more than one row.

New scale parameters specific to a Type 9 Determinant nomogram are

- *grid*: set to False to indicate that this scale (row of the determinant) contains only one variable
- f: the function in u given by the first column in the determinant row corresponding to the scale
- g: the function in u given by the second column in the determinant row corresponding to the scale
- h: the function in u given by the third column in the determinant row corresponding to the scale

We will let $-10 < N < 10$, $-4 < T < 4$ and $-10 < M < 10$ as we did in the previous section. However, there is a problem for the case $T = 0$ in our range. In the original equation $M = \frac{T-N}{T^2}$ you can see that M is infinity when $T = 0$, a singularity. You can also see this in the way the equation is expressed above as $N + MT^2 - T = 0$, where a value $T = 0$ will result in $N = 0$ and M being any number at all.

This is an indication that we should perform an initial transformation on the scales. Another clue is when the PyNomo build does not end because it is encountering large numbers due to the singularity. Here PyNomo uses matrix multiplication to transform the nomogram such that four points on the paper end up at a different set of four points on the paper. The values of *u_min_trafo* and *u_max_trafo* indicate the points that will be transformed to the corners of a rectangle with sides along the x- and y-axes.

So here we specify this initial transformation by introducing the following new parameters:

- *u_min_trafo*: the mininum value for the transformation, which should be set to the value of *u_min* in each of the outer two scales
- *u_max_trafo*: the maximum value for the transformation, which should be set to the value of *u_max* in each of the outer two scales
- *transform_ini*: a block parameter to set to True to indicate that an initial transformation should be performed

The PyNomo script and output nomogram are shown on the following page. The nomogram that is produced matches the first Type 10 nomogram from the previous section. For an equation like this that can be expressed in one of the basic types that PyNomo recognizes, we have the great advantage that we do not need to derive the determinant form as we did here.

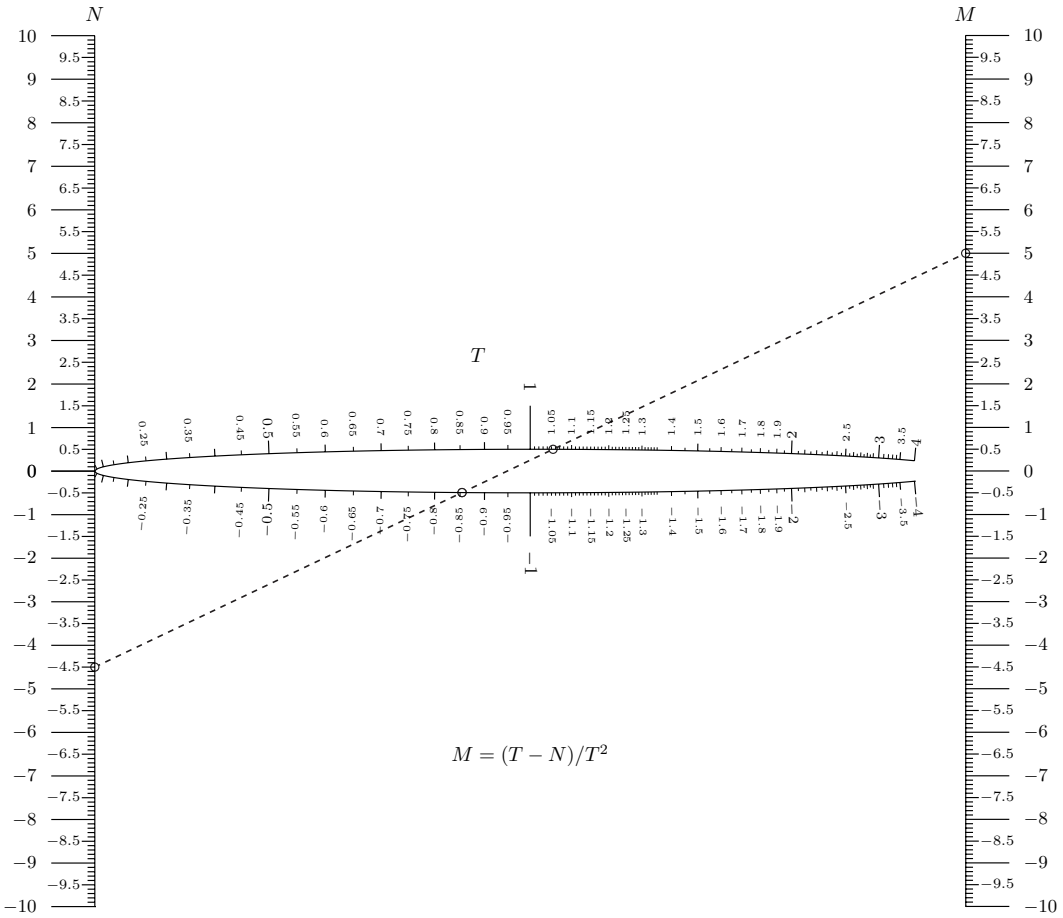
This PyNomo script and output PDF graphic are available online for download.[48]

[47] See *http://myreckonings.com/wordpress/2008/01/09/the-art-of-nomography-ii-designing-with-determinants/* for the rules for transforming a determinant to standard form. By adding the first column to the second column, dividing the first row by T^2+1, swapping the first and third columns, multiplying the second column by -1, and moving the third row to the top row, the determinant would end up as

$$\begin{vmatrix} 0 & N & 1 \\ \frac{T^2}{T^2+1} & \frac{T}{T^2+1} & 1 \\ 1 & M & 1 \end{vmatrix} = 0$$

[48] *http://www.myreckonings.com/pynomo11/Type9-Range1.py* and *http://www.myreckonings.com/pynomo11/Type9-Range1.pdf*

```
### Type9-Range1.py ###

from pynomo.nomographer import *

N_params={
            'u_min':-10.0,
            'u_max':10.0,
            'u_min_trafo':-10.0,
            'u_max_trafo':10.0,
            'f':lambda u:0,
            'g':lambda u:1,
            'h':lambda u:-u,
            'title':r'$N$',
            'scale_type':'linear',
            'tick_levels':3,
            'tick_text_levels':2,
            'tick_side':'left',
            'grid':False
       }

T_params={
            'u_min':-4.0,
            'u_max':4.0,
            'f':lambda u:u*u,
            'g':lambda u:1,
            'h':lambda u:-u,
            'title':r'$T$',
            'title_x_shift':-1.0,
            'title_y_shift':1.5,
            'scale_type':'log smart',
            'tick_levels':5,
            'tick_text_levels':5,
            'grid':False
           }

M_params={
            'u_min':-10.0,
            'u_max':10.0,
            'u_min_trafo':-10.0,
            'u_max_trafo':10.0,
            'f':lambda u:1,
            'g':lambda u:0,
            'h':lambda u:-u,
            'title':r'$M$',
            'scale_type':'linear',
            'tick_levels':3,
            'tick_text_levels':2,
            'grid':False
       }


block_params={
            'block_type':'type_9',
            'f1_params':N_params,
            'f2_params':T_params,
            'f3_params':M_params,
            'transform_ini':True,
            'isopleth_values':[[-4.5,'x',5.0]],
             }

main_params={
             'filename':'Type9-Range1.pdf',
             'paper_height':15.0,
             'paper_width':15.0,
             'block_params':[block_params],
             'transformations':[('rotate',0.01),('scale paper',)],
             'title_y':2.5,
             'title_str':r'$M = (T - N) / T^2$',
             'isopleth_params':[
                          {'color':'black',
                           'linewidth':'thick',
                           'linestyle':'dashed',
                           'circle_size':0.07,
                           'transparency':0.0,
                          },
                         ],
              }

Nomographer(main_params)
```


N
M
T
$M = (T - N)/T^2$

Now we'll change our ranges to $0 < N < 10$, $0 < T < 4$ and $-20 < M < 0$ as we did in the second Type 10 example of the previous section. This nomogram is shown on the following page. You can see that unlike our Type 10 plot there is no automatic shearing of the T-scale and M-scale upward so the M-scale is aligned horizontally with the N-scale. We would like to have this in order to maximize the size of the nomogram on the paper.

To adjust our determinant to do this involves looking up the most general form of the standard determinant for this equation, one that provides individual scale factors for M and N so they can be made the same length, and then applying a mathematical operation for a *shear transformation* to the determinant. The shear effectively slides the outer scales vertically, pushing the right scale up until the outer scales are aligned left-to-right, with the center scale rotating along with them as if the scales were linkages. It's not a lot of fun to do this, really.

But here's where the polygon transformation of PyNomo helps us so much. In the script on the following page, if we were to replace the transformations line from

```
'transformations':[('rotate',0.01),('scale paper',)],
```

to

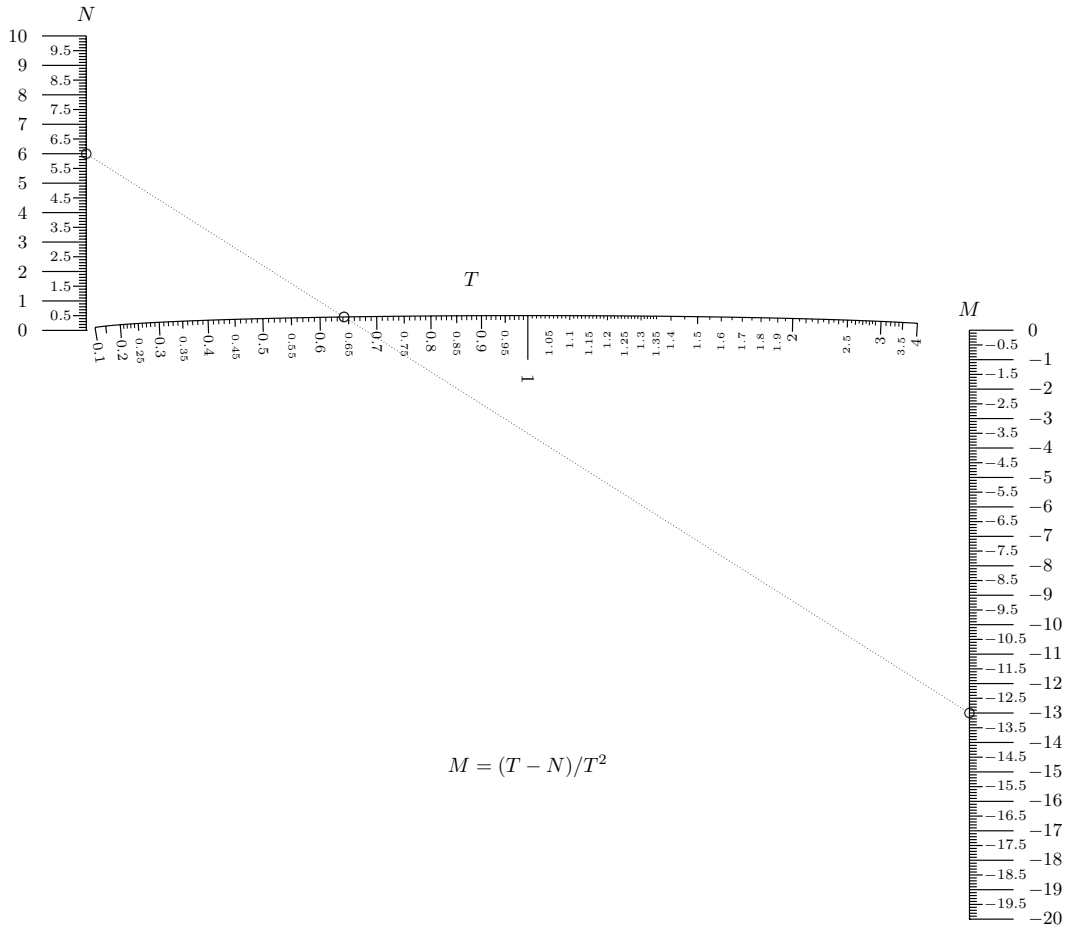
```
'transformations':[('rotate',0.01),('scale paper',),('polygon',)],
```

we would find that we get the same sheared result as we obtained in the previous section.

This PyNomo script and output PDF graphic are available online for download.[49]

[49] *http://www.myreckonings.com/pynomo11/Type9-Range2.py* and *http://www.myreckonings.com/pynomo11/Type9-Range2.pdf*

```
### Type9-Range2.py ###

from pynomo.nomographer import *

N_params={
            'u_min':0.0,
            'u_max':10.0,
            'u_min_trafo':-10.0,
            'u_max_trafo':10.0,
            'f':lambda u:0,
            'g':lambda u:1,
            'h':lambda u:-u,
            'title':r'$N$',
            'scale_type':'linear',
            'tick_levels':3,
            'tick_text_levels':2,
            'tick_side':'left',
            'grid':False
       }

T_params={
            'u_min':0.1,
            'u_max':4.0,
            'f':lambda u:u*u,
            'g':lambda u:1,
            'h':lambda u:-u,
            'title':r'$T$',
            'title_x_shift':-1.0,
            'title_y_shift':0.5,
            'scale_type':'log smart',
            'tick_levels':5,
            'tick_text_levels':5,
            'grid':False
           }

M_params={
            'u_min':-20.0,
            'u_max':0.0,
            'u_min_trafo':-10.0,
            'u_max_trafo':10.0,
            'f':lambda u:1,
            'g':lambda u:0,
            'h':lambda u:-u,
            'title':r'$M$',
            'scale_type':'linear',
            'tick_levels':3,
            'tick_text_levels':2,
            'grid':False
        }


block_params={
             'block_type':'type_9',
             'f1_params':N_params,
             'f2_params':T_params,
             'f3_params':M_params,
             'transform_ini':True,
             'isopleth_values':[[6,'x',-13]],
             }

main_params={
              'filename':'Type9-Range2.pdf',
              'paper_height':15.0,
              'paper_width':15.0,
              'block_params':[block_params],
              'transformations':[('rotate',0.01),('scale paper',)],
              'title_y':2.5,
              'title_str':r'$M = (T - N) / T^2$',
              'isopleth_params':[
                       {'color':'black',
                        'linewidth':'thin',
                        'linestyle':'dotted',
                        'circle_size':0.08,
                        'transparency':0.2,
                       },
                      ],
              }

Nomographer(main_params)
```



$$M = (T - N)/T^2$$

A Grid Nomogram

We can plot a nomogram for an equation of up to six variables using this Type 9 determinant form if we can find a way to express the determinant with no more than two variables per row and no variables used in more than one row. Each row having two variables is plotted as a grid rather than a curvilinear scale—the Type 9 example on the PyNomo site shows how the grid parameters are specified.[50] The celestial navigation example on the PyNomo site is also a grid nomogram created as a Type 9 determinant form.[51] This is a very useful capability—grid nomograms are striking in appearance but very tedious to draw by hand!

Let's draw a nomogram for the following relation based on its determinant form. Notice that no variables appear in more than one row, and the second row contains two variables. This second row will result in a grid where the effective scale point is the intersection of the curves for the given values of k and θ.

$$m = k\sin\theta + l\cos\theta$$

$$\begin{vmatrix} 0 & m & 1 \\ \frac{100\cos\theta}{1+\cos\theta} & \frac{k\sin\theta+100\cos\theta}{1+\cos\theta} & 1 \\ 100 & 100-l & 1 \end{vmatrix} = 0$$

This nomogram was designed by Liunian Li, and his derivation is provided in another essay,[52] but you can expand the determinant to verify that it produces the same equation given above it.

New scale parameters specific to a Type 9 Determinant nomogram with at least one grid are:

- *f_grid*: the function in u and v given by the first column in the determinant row corresponding to the grid scale
- *g_grid*: the function in u and v given by the second column in the determinant row corresponding to the grid scale
- *h_grid*: the function in u and v given by the third column in the determinant row corresponding to the grid scale
- *u_start*: the starting value of u in the grid scale
- *u_stop*: the ending value of u in the grid scale
- *v_start*: the starting value of v in the grid scale
- *v_stop*: the ending value of v in the grid scale
- *u_values*: values of u (if specified) in the grid scale
- *v_values*: values of v (if specified) in the grid scale
- *text_prefix_u*: text to add before every value of u in the grid scale
- *text_prefix_v*: text to add before every value of v in the grid scale
- *u_texts*: text to add before each value of u specified by *u_values* in the grid scale

[50] *http://www.pynomo.org/wiki/index.php/Type_9*

[51] *http://www.pynomo.org/wiki/index.php/Example:Star_navigation*

[52] *http://myreckonings.com/wordpress/2008/03/13/a-4-variable-nomogram-%e5%9b%9b%e5%8f%98%e9%87%8f%e8%af%ba%e6%a8%a1%e5%9b%be/* Also see the comments at the bottom of this post and the paper on applying shear transformations to the nomogram at *http://www.myreckonings.com/wordpress/wp-content/uploads/FourVariableNomogram/FourVariableNomogramShaping033008.pdf*

- *v_texts*: text to add before each value of *u* specified by *v_values* in the grid scale
- *u_texts_v_start*: set to True if the labels for *u* should lie at the starting values of the *v* curves in the grid scale
- *u_texts_v_stop*: set to True if the labels for *u* should lie at the ending values of the *v* curves in the grid scale
- *v_texts_u_start*: set to True if the labels for *v* should lie at the starting values of the *u* curves in the grid scale
- *v_texts_u_stop*: set to True if the labels for *v* should lie at the ending values of the *u* curves in the grid scale
- *text_distance*: the space between the label and the grid
- *u_line_color*: the color of the *u* curves
- *u_text_color*: The color of the u-scale labels
- *v_line_color*: the color of the *u* curves
- *v_text_color*: The color of the v-scale labels

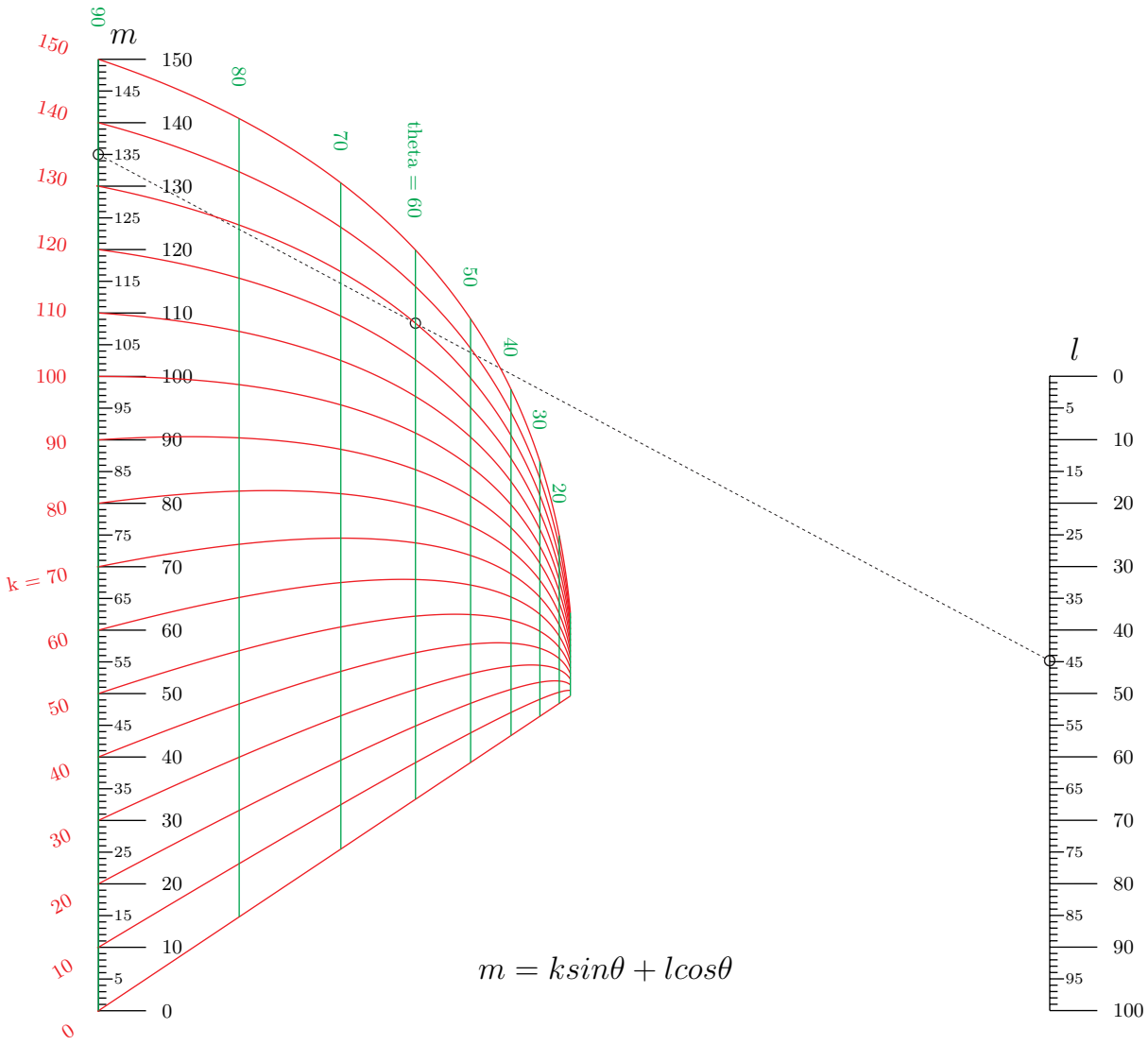
The PyNomo script and nomogram are shown on the following two pages. The *grid* parameter is set to *True* to indicate a grid scale and f, g and h parameters are replaced by *f_grid*, *g_grid* and *h_grid*. We will assign θ to u and k to v. The start and stop values for these are specified in *u_start*, *u_stop*, *v_start* and *v_stop*, while *u_min* and *u_max* are still needed in case we align (*tag*) another curve to the u-scale (which we are not doing). The *u_values* and *v_values* parameters are specified because we only want the grid curves to be constructed for those specific values of θ and k. I could have chosen a common text prefix to add to the labels using *text_prefix_u* and *text_prefix_v*, but I didn't want "$\theta =$" and "$k =$" for every value so I commented out these lines and specified the complete label for each value of u and v with *u_texts* and *v_texts*, where only one value has the prefix. (For some reason using *theta* produces θ in the text prefix parameters but not in the individual text values, so there I set it to the word *theta*.) We set the True/False values of *u_texts_v_start*, *u_texts_v_stop*, *v_texts_u_start* and *v_texts_u_start* to place the grid labels where we want them, as well as *text_distance*. Here we also set *u_line_color* and *v_line_color* to better distinguish the grid lines.

Notice that it happens that the scale values of m and k line up with each other at $\theta = 90$, so there really is no need to have both sets of labels. To clean up this nomogram we could set *u_texts* to all empty `''` strings in *theta_k_params* to remove the k-scale labels. Then we could change the title of the m-scale to *m*,*k* and add `'tick_side':'left',` in *m_params* to move its labels to the left side of the scale, resulting in a nomogram with better readability.

This PyNomo script and output PDF graphic are available online for download.[53]

[53] *http://www.myreckonings.com/pynomo11/Type9-Grid-Isopleth.py* and *http://www.myreckonings.com/pynomo11/Type9-Grid-Isopleth.pdf*

```
### Type9-Grid-Isopleth.py ###

from pynomo.nomographer import *

m_params={
      'u_min':0.0,
      'u_max':150.0,
      'f':lambda u:0,
      'g':lambda u:u,
      'h':lambda u:1.0,
      'title':r'\Large $m$',
      'title_x_shift':0.4,
      #'title_y_shift':0.0,
      'scale_type':'linear',
      'tick_levels':3,
      'tick_text_levels':2,
      'grid':False,
           }

theta_k_params={
      'u_min':10.0,
      'u_max':90.0,
      'f_grid':lambda u,v:100.0*cos(u*pi/180.0)/
                (1+cos(u*pi/180.0)),
      'g_grid':lambda u,v:(v*sin(u*pi/180.0)+
           100.0*cos(u*pi/180.0))/(1+cos(u*pi/180.0)),
      'h_grid':lambda u,v:1.0,
      'u_start':10.0,
      'u_stop':90.0,
      'v_start':0.0,
      'v_stop':150.0,
      'u_values':[10.0,20.0,30.0,40.0,50.0,60.0,70.0,
                      80.0,90.0],
      'v_values':[0.0,10.0,20.0,30.0,40.0,50.0,60.0,
                  70.0,80.0,90.0,100.0,110.0,120.0,
                  130.0,140.0,150.0],
      'u_texts':['','20','30','40','50','theta = 60',
                 '70','80','90'],
      'v_texts':['0','10','20','30','40','50','60',
                 'k = 70','80','90','100','110','120',
                 '130','140','150'],
      'v_texts_u_start':False,  # default value
      'v_texts_u_stop':True,    # default value
      'u_texts_v_start':False,  # default value
      'u_texts_v_stop':True,    # default value
      'text_distance':0.5,
      'u_line_color':color.cmyk.Green,
      'u_text_color':color.cmyk.Green,
      'v_line_color':color.cmyk.Red,
      'v_text_color':color.cmyk.Red,
      #'text_prefix_u':r'$\theta$=',
      #'text_prefix_v':r'$k$=',
        'grid':True,
            }

l_params={
      'u_min':0.0,
      'u_max':100.0,
      'f':lambda u:100.0,
      'g':lambda u:100.0-u,
      'h':lambda u:1.0,
      'title':r'\Large $l$',
      'title_x_shift':0.4,
      'scale_type':'linear',
      'tick_levels':3,
      'tick_text_levels':2,
      'grid':False,
           }

block_params={
      'block_type':'type_9',
      'f1_params':m_params,
      'f2_params':theta_k_params,
      'f3_params':l_params,
      'transform_ini':False,
      'isopleth_values':[[135,[60,130],'x']]
           }

main_params={
      'filename':'Type9-Grid-Isopleth.pdf',
      'paper_height':15.0,
      'paper_width':15.0,
      'block_params':[block_params],
      'transformations':[('rotate',0.01),('scale paper',)],
      'title_x': 8,
      'title_y': 0.5,
      'title_box_width': 6,
      'title_str':r'\Large $m = k sin \theta + l cos \theta$',
      'isopleth_params':[
           {'color':'black',
            'linewidth':'thin',
            'linestyle':'dashed',
            'circle_size':0.08,
            'transparency':0.0,
           },
           ],
           }

Nomographer(main_params)
```


m
l
k = 70
theta = 60
90
80
70
50
40
30
20
$m = ksin\theta + lcos\theta$

A Circular Nomogram

Let's do one more nomogram from the determinant form. We are going to create a circular nomogram, a form in which one scale on a circle provides two values when the straightedge crosses it. This nomogram provides the roots of the quadratic equation $q^2 - aq + b = 0$. The determinant appears overly complicated for this simple formula, but when expressed in this form the first scale becomes a circle. The derivation of this form of the determinant is provided elsewhere[54], but you can expand the determinant here if you want to verify that it does indeed produce our equation.

$$q^2 - aq + b = 0$$

$$\begin{vmatrix} \frac{q}{q^2+1} & \frac{1}{q^2+1} & 1 \\ \frac{1}{a} & 0 & 1 \\ 0 & \frac{-1}{b-1} & 1 \end{vmatrix} = 0$$

The script and nomogram are found on the following pages. The scale parameter *extra_params* is used to break the scales into different ranges to change from linear to logarithmic scaling and to avoid intervals where values go to infinity ($a = 0$ or $b = 1$). Finally, ranges with no tick marks or labels are provided for the q-scale and a-scale to extend their lines closer to the origin to bring the circle closer to closure (full closure would require the variables to go to infinity). The q-scale curves until it is upside down, which would normally switch the side of the ticks and labels on the bottom half of the circle to keep the labels mostly upright, so we also set the *turn_relative* parameter to True for that scale to allow them to follow around the circle.

The page size to scale to was originally set to 25×25cm, but this is a rough estimate of the final size of the nomogram, so in the end I tweaked the values to 25.5×30.4cm to get the q-scale to be perfectly circular (which I did by overlaying the nomogram with a grid by setting *make_grid* to True in the main parameters). Otherwise it would be an ellipse, which works perfectly well but doesn't look nearly as striking as a circle.

When a straightedge is placed across particular values of a and b, three things can happen. If it intersects the circle in two places, those two values of q are the two real roots of the equation (the values of q from the quadratic formula that solve the equation). If it grazes the edge of the circle as a tangent, that value is the one repeated real root (where the discriminant $B^2 - 4AC$ in the quadratic formula is zero). If it misses the circle entirely there are no real roots, or no real solutions to the equation ($B^2 - 4AC < 0$). Actually, the b-scale value turns out to always be the product of the two values on the q-scale.

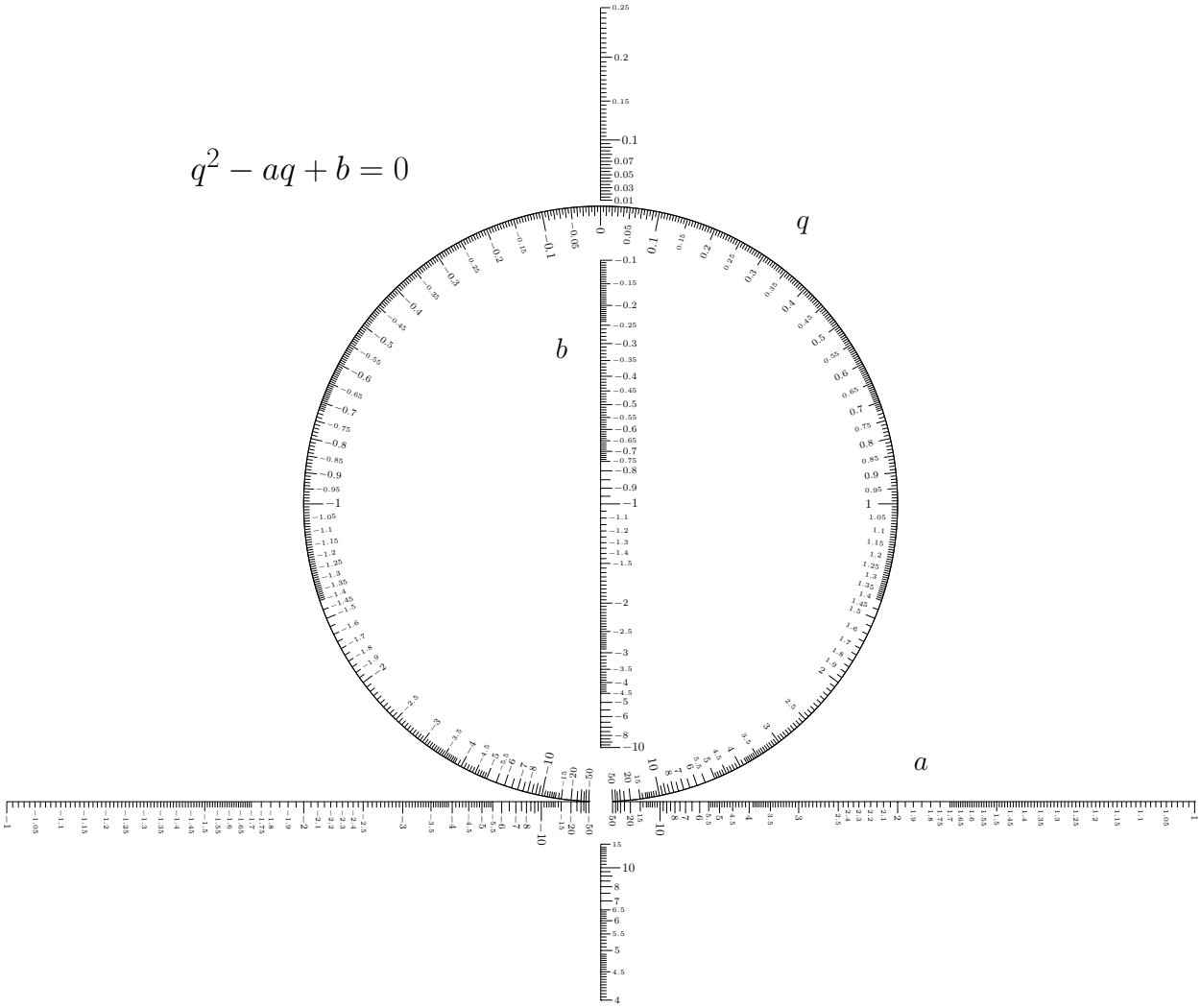
This PyNomo script and output PDF graphic are available online for download.[55]

[54] *http://myreckonings.com/wordpress/2008/01/09/the-art-of-nomography-iii-transformations/*

[55] *http://www.myreckonings.com/pynomo11/Type9-Circular.py*
and *http://www.myreckonings.com/pynomo11/Type9-Circular.pdf*

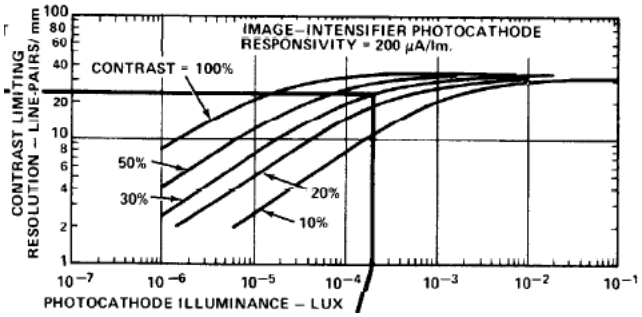
```
### Type9-Circular.py ###

from pynomo.nomographer import *

Q_params={
      'u_min':-50.0,
      'u_max':50.0,
      'f':lambda u:u/(u**2+1),
      'g':lambda u:1/(u**2+1),
      'h':lambda u:1,
      'title':r'\huge $q$',
      'title_x_shift':7.2,
      'title_y_shift':-0.3,
      'scale_type':'linear smart',
      'tick_levels':0,
      'tick_text_levels':0,
      'turn_relative':True,
      'grid_length_0':0.5,
      'text_distance_0':0.55,
      'grid_length_1':0.3,
      'text_distance_1':0.35,
      'grid_length_2':0.25,
      'text_distance_2':0.30,
      'grid_length_3':0.15,
      'text_distance_3':0.20,
      'grid_length_4':0.15,
      'text_distance_4':0.20,
      'extra_params':
            [
            {
               'u_min':-50.0,
               'u_max':50.0,
               'base_start':-50.0,
               'base_stop':50.0,
               'tick_levels':5,
               'tick_text_levels':5,
               'scale_type':'log smart',
               'text_format':r"$%3.3g$ ",
            }
            ],
      }

A_params={
      'u_min':1.0,
      'u_max':50.0,
      'f':lambda u:1/u,
      'g':lambda u:0,
      'h':lambda u:1,
      'title':r'\huge $a$',
      'title_x_shift':-7.0,
      'title_y_shift':0.8,
      'scale_type':'log smart',
      'tick_levels':5,
      'tick_text_levels':5,
      'grid_length_0':0.5,
      'text_distance_0':0.55,
      'grid_length_1':0.3,
      'text_distance_1':0.35,
      'grid_length_2':0.25,
      'text_distance_2':0.30,
      'grid_length_3':0.15,
      'text_distance_3':0.20,
      'grid_length_4':0.15,
      'text_distance_4':0.20,
      'extra_params':
            [{
               'u_min':-50.0,
               'u_max':-1.0,
               'tick_levels':5,
               'tick_text_levels':5,
               'scale_type':'log smart',
            }
            ]
      }

B_params={
      'u_min':-10.0,
      'u_max':-0.1,
      'f':lambda u:0,
      'g':lambda u:-1/(u-1),
      'h':lambda u:1,
      'title':r'\huge $b$',
      'title_x_shift':-1.0,
      'title_y_shift':-2.5,
      'scale_type':'log smart',
      'tick_levels':5,
      'tick_text_levels':5,
      'grid_length_0':0.5,
      'text_distance_0':0.55,
      'grid_length_1':0.3,
      'text_distance_1':0.35,
      'grid_length_2':0.25,
      'text_distance_2':0.30,
      'grid_length_3':0.15,
      'text_distance_3':0.20,
      'grid_length_4':0.15,
      'text_distance_4':0.20,
      'extra_params':
            [{
               'u_min':0.01,
               'u_max':0.25,
               'tick_levels':5,
               'tick_text_levels':5,
               'scale_type':'log smart',
            },
            {
               'u_min':4.0,
               'u_max':15.0,
               'tick_levels':5,
               'tick_text_levels':5,
               'scale_type':'log smart',
            }]
      }

block_params={
       'block_type':'type_9',
       'f1_params':Q_params,
       'f2_params':A_params,
       'f3_params':B_params,
       'transform_ini':False,
             }

main_params={
      'filename':'Type9-Circular.pdf',
      'paper_height':25.4,
      'paper_width':30.4,
      'block_params':[block_params],
      'transformations':[('rotate',0.01),('scale paper',)],
      'title_x':7.5,
      'title_y': 21.0,
      'title_box_width': 6,
      'title_str':r'\Huge $q^2-aq+b=0$',
      'make_grid':False,
             }

Nomographer(main_params)
```

$$q^2 - aq + b = 0$$



18 Creating a Type 5 (Contour) Nomogram

A Type 5 nomogram consists of an x-y plot of a family of curves, where the y-value is found from a particular x-value by the curve for a particular value of a third variable. This is often seen by itself as a graph with multiple curves as shown in the figure below, but it can also be part of a larger nomogram. There are a few examples involving contour blocks on the PyNomo site.[56]



A well-known electro-optic nomogram from the *RCA Electro-Optics Handbook* shown on the next page includes this contour block in the upper-right corner. This contour block provides the relationship between the Photocathode Illuminance, the Scene Contrast, and the Contrast-Limiting Resolution of a camera. If you're navigating the nomogram from below with a straightedge and encounter the x-axis, you simply move parallel to the y-axis until you hit the curve for the particular Contrast value, then horizontally to meet the y-axis, at which time you continue navigating the nomogram with the straightedge. You do the reverse if navigating from the left.

You may notice that we've already created the nomogram portion to the left of the contour block (see page 29) and the portion below the contour block (see page 52). After we create the contour block here we'll connect all three parts together (using tags to align the outer scales) to recreate this entire nomogram in vector form with PyNomo.

[56] *http://www.pynomo.org/wiki/index.php/Type_5*
and *http://www.pynomo.org/wiki/index.php/Example:Amortized_loan_calculator*
and *http://www.pynomo.org/wiki/index.php/Photography_exposure*
and *http://www.pynomo.org/wiki/index.php/Example:Radio-frequency_single_electron_transistor*

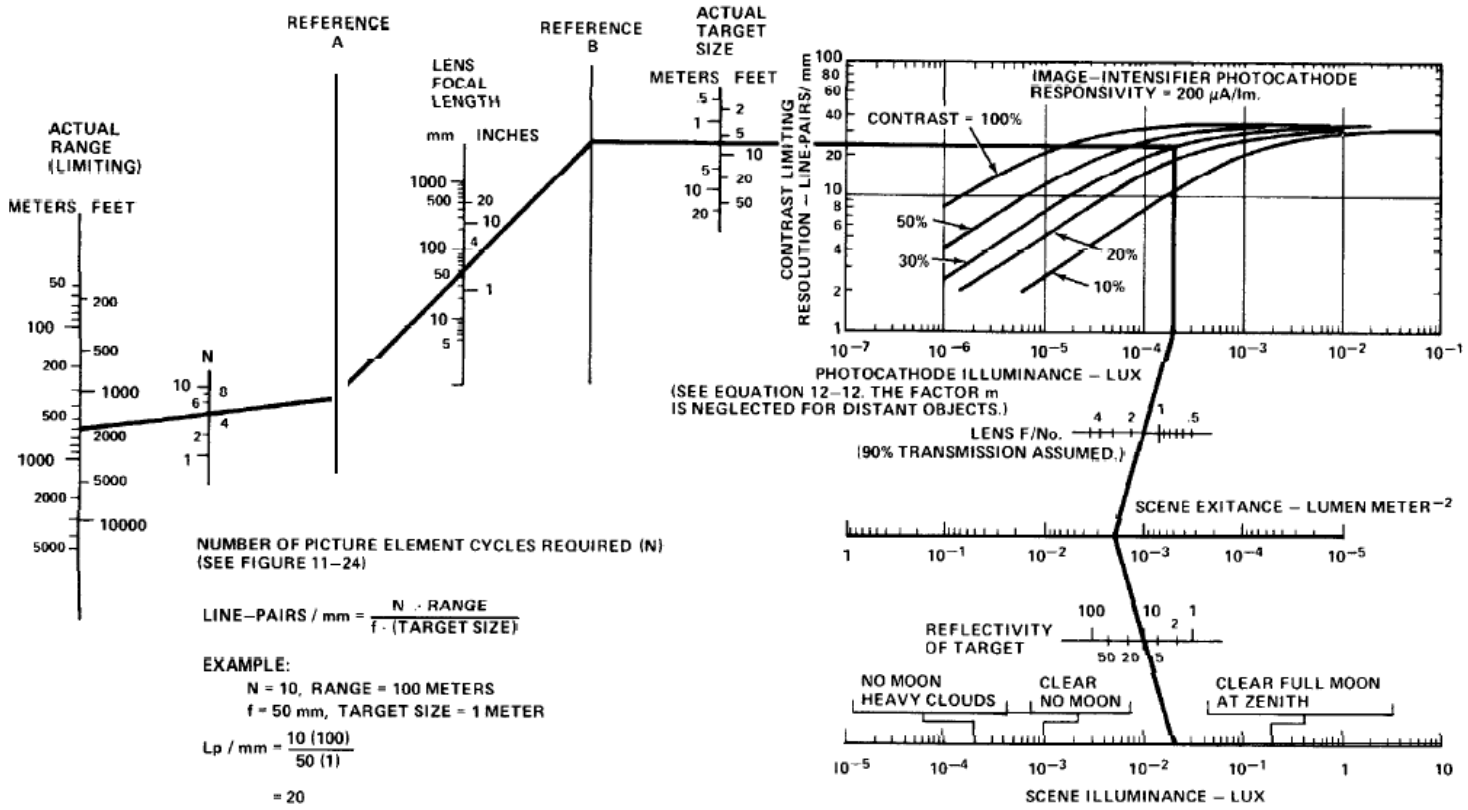


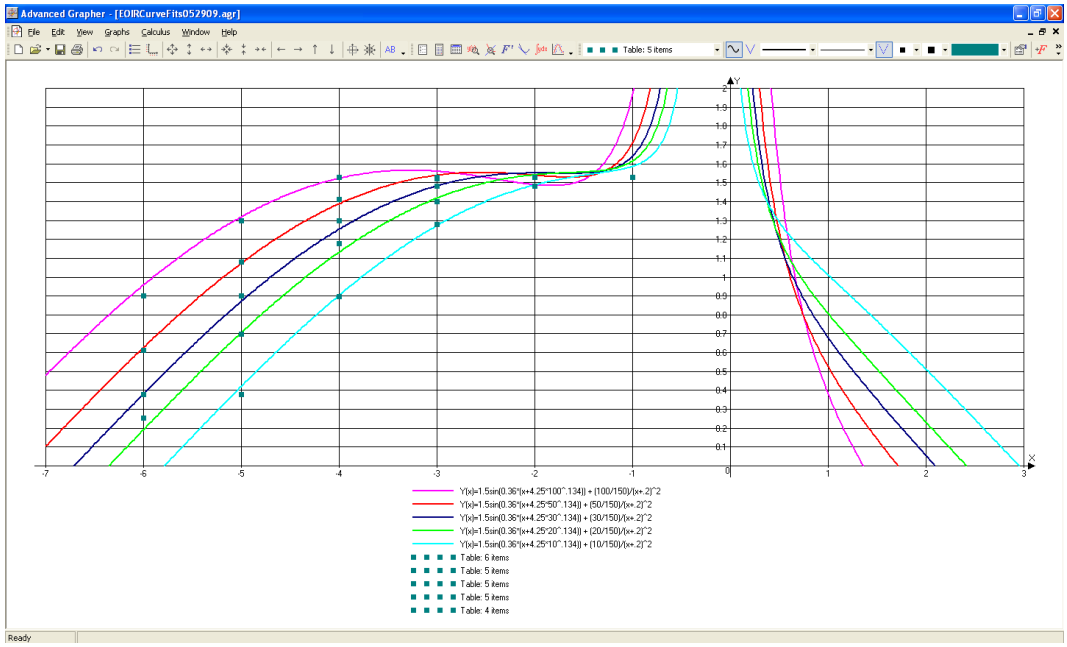
Fig. 11-25 Nomograph for determining performance parameters of a low-light-level image intensifier target detection or recognition system.

The construction of the nomogram for this contour block turned out to take some effort. First, I didn't have the equation for the family of curves since it was derived from experimental data. Therefore, I needed to do some curve fitting to find an equation that would provide the x-y curve for every value of the Contrast, denoted here by the variable v. There are tools for fitting a single curve as $y = f(x)$, but here we also have this third variable v, so I had to make educated guesses and make adjustments to find a good fit.

Each curve is made up of a relatively straight section followed by a flattened section. I figured that a sinusoidal curve might be the simplest type of function to fit, although I'd have to add something near the y-axis to prevent it from dropping back down again. With an easy-to-use shareware package called *Advanced Grapher* that I've happily used for at least 10 years,[57] I started experimenting by plotting equations and adjusting constants and terms until I had a close match (*Advanced Grapher* can do curve-fitting, but again only for one dependent variable). My initial results are shown in the screen capture below. We are interested in the curves to the left of the y-axis. The small squares are the actual values I read off of the contour block, and the colored curves are my best fit to them:

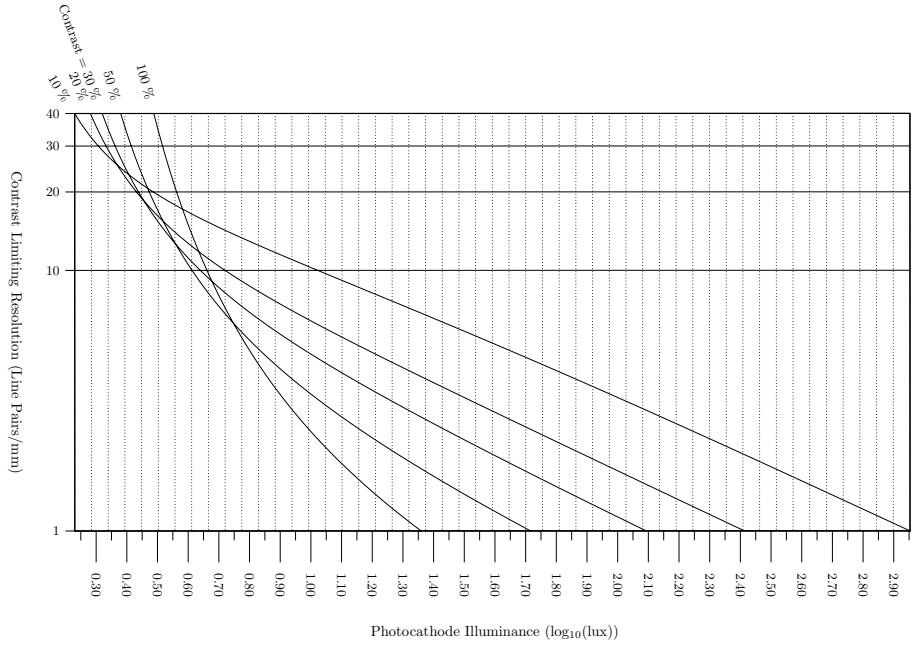
$$y = 1.5\sin(0.36(x + 4.25 \times 20^{0.134})) + \frac{v/150}{(x+0.2)^2}$$

where x is the exponent of 10 from the original contour block and v = 10, 20, 30, 50 and 100 is the Contrast variable. This equation took quite a while to come up with. Do you see how the second term gets large as x approaches zero to give a boost so the sine term doesn't drop off there?



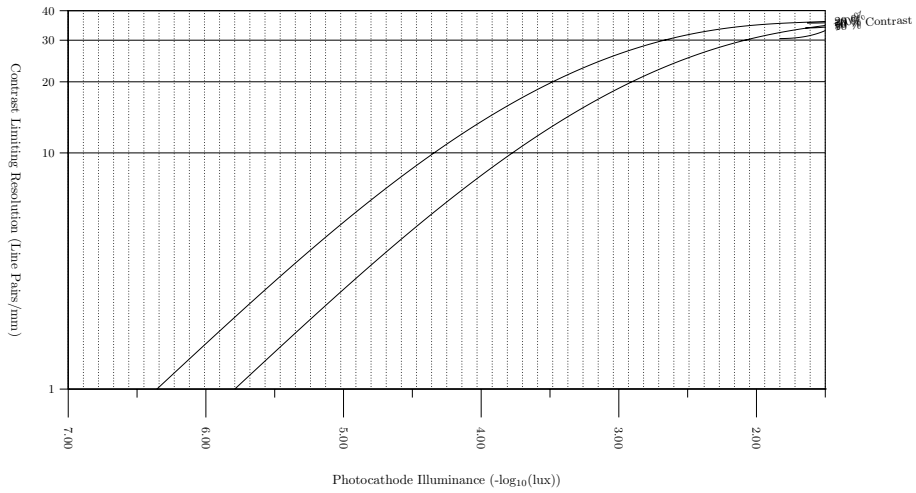
[57] *http://www.alentum.com/agrapher/*

The final PyNomo script will be seen shortly, but here's the result of using the equation above in it:



Contrast Limiting Resolution vs. Photocathode Illuminance for an Image-Intensifier Photocathode Responsivity of 200μA/lm

You can see that it plotted the curves from the $x > 0$ part of the curve, not the ones I wanted from $x < 0$. The way the Type 5 nomogram is constructed is to take the y-value and the v-value and find the x-value that produces it. Here there is more than one x, and the wrong set is chosen. So I changed x to $-x$ in the equation, realizing that the resulting x-axis would be the negative of the exponent in the original contour block. It worked for only two of the v-values:



Contrast Limiting Resolution vs. Photocathode Illuminance for an Image-Intensifier Photocathode Responsivity of 200μA/lm

It's plotting the correct set, but the curves stop where they curve upward, or in other words, where there

are two values of x for a given y. So I played around with my original curve-fitting equation until the curves were sufficiently flat in that area that they did not curve upward.

$$y = 1.5\sin(0.36(-x+4.25\times 20^{0.134})) + \frac{v/150}{(-x+0.15+v/200)^2}$$

Finally it worked. The script and output nomogram are shown on the following page. The lesson here is to use a simple, non-undulating function for the contour block. For example, polynomial functions will not work if a local minimum or maximum occurs in the range covered by the contour block.

You can see how constants and formulas can be defined at the top of the script for convenience. There are also a number of new parameters specific to this Type 5 nomogram. First, there is only one parameter set (a block parameter set) rather than separate ones for the block and individual scales. Here the variable *u* is the y-axis, and it's given the tag *resolution* so we can attach the nomogram on the left to it later. The variable *wd* is the x-axis, and it's given the tag *illum* so we can attach the nomogram below it. The variable *v* is the Contrast value. These variable names are defined in PyNomo and must be the ones used.

There are no scale parameters for a Type 5 Contour block, only block and main parameters. New block parameters specific to this are:

- *manual_x_scale*: if True, define the minimum and maximum values of x with *x_min* and *x_max* (this scale is normally *wd*)
- *x_min*: minimum value of x (this scale is normally *wd*)
- *x_max*: maximum value of x (this scale is normally *wd*)
- *u_scale_opposite*: place the u-scale on the opposite side of the block
- *u_func*: the function of the variable for the u-scale
- *u_values*: values to plot for the u-scale
- *u_manual_axis_data*: a list of custom labels to write for each *u_value*
- *draw_line*: if True, draw a line between the tick mark and the label for manual axis data
- *x_corr* and *y_corr*: x and y offsets in cm of the labels for manual axis data
- *u_title*: the title of the u-scale
- *scale_type_u*: type of scale (linear, log, manual line, manual arrow or manual point) for the u-scale
- *u_tick_side*: the side of the scale to print the tick marks and labels for the u-scale
- *u_tag*: scales with the same tag are aligned (overlaid) in the final nomogram
- *u_reference*: if True, axis is treated as reference line that is a turning point
- *u_tick_levels*: the level to which the tick marks are printed for the u-scale
- *u_tick_text_levels*: the level to which the tick labels are printed for the u-scale
- *u_title_opposite_tick*: if True, place the centered title on the opposite side from the tick marks for the u-scale
- *u_title_distance_center*: distance of the title from the u-scale center (0.5cm by default)
- *u_title_draw_center*: if True, locate the title at the center of the scale point of the u-scale
- *u_text_format*: the number of digits to the left and right of the decimal point in the tick labels (r"%3.2*f*" or nnn.nn is the default) for the u-scale

- *u_align_func*: the function used to align the u-scale with another scale with the same
- *horizontal_guides*: if True, draw equally-spaced horizontal grid lines (not related to values on the y-axis) as a visual aid (False by default)
- *vertical_guides*: if True, draw equally-spaced vertical grid lines (not related to values on the y-axis) as a visual aid (True by default)
- *u_axis_color*: The color of the u-scale axis (or family of curves if the v-scale is specified)
- *u_title_color*: The color of the title text of the u-scale
- *u_text_color*: The color of the u-scale labels

Also, for each parameter above that contains *u*, there are also analogous parameters for the *v* and *wd* scales.

Here we set *u_func* to log10(u) to make the y-axis a simple logarithmic scale, and we set *v_func* to the equation above in *x* and *v* (this may not seem to make sense but that's the way it works). We set *u_values* and *v_values* to specific values we want plotted for the Resolution and the Contrast.

In order to prevent the Contrast curve labels from overlapping, we use the *v_manual_axis_data* parameter to specify the specific label text for each curve, the x and y offset of the label, and the *draw_line* flag to draw a line between the end of the curve and the label. It works very nicely.

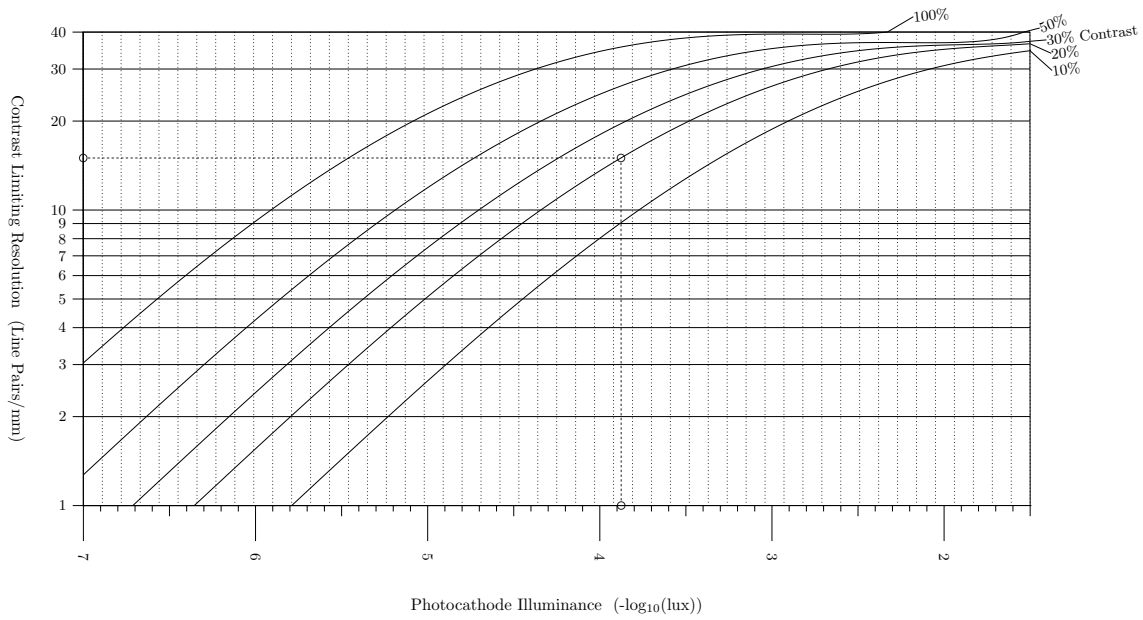
The wd-scale (the x-axis) has the same title, tick level and tick text levels we've seen for all scales before. I had to indicate in its title that the value is actually the negative of the exponent of the actual Photocathode Illuminance, or $-log(Illuminance)$. The minimum and maximum values of *wd*, however, are set by the *x_min* and *x_max* parameters. I left the wd-scale as a default linear scale, but I could have made it logarithmic by setting the parameter *scale_type_wd* to *log*.

I left the colors of the u/v/wd scales at their default color of black. To get the contour block oriented as I wanted required setting the *mirror_x* parameter to True, where *mirror_x* flips the wd-scale (if it were used, *mirror_y* would flip the u-scale). The *u_scale_opposite* parameter is also set to *True* to move the labels on the u-scale to the left side of the block rather than the right. The paper width and height in the main parameters give the contour block the size ratio I was looking for. The *isopleth_values* entries are for *u*, *v* and *wd*, respectively (where the 'x' entry for *wd* value means that it is computed from the other two values).

This PyNomo script and output PDF graphic are available online for download.[58]

[58] *http://www.myreckonings.com/pynomo11/Type5-Isopleth.py* and *http://www.myreckonings.com/pynomo11/Type5-Isopleth.pdf*

```
### Type5-Isopleth.py ###

from pynomo.nomographer import *

const_A = 1.5
const_B = 0.36
const_C = 4.25
const_D = .134
const_E = 150
const_F = 0.15

def f1(x,v):
  return const_A*sin(const_B*(-x+const_C*v**const_D))+ \
         (v/const_E)/(-x+const_F+(v/200))**2

block_20_params={
        'block_type':'type_5',
        'u_tag':'resolution',
        'wd_tag':'illum',
        'width':10.0,
        'height':13.0,
        'u_func':lambda u:log10(u),
        'u_values':[1.0,2.0,3.0,4.0,5.0,
                    6.0,7.0,8.0,9.0,
                    10.0,20.0,30.0,40.0],
        'u_text_format':r"$%3.0f$ ",
        'v_func':f1,
        'v_values':[10.0,20.0,30.0,50.0,100.0],
        'v_manual_axis_data':
               {
                   10.0:['$10 \%$',
                         {'x_corr':0.22,
                          'y_corr':-.45,
                          'draw_line':True}],
                   20.0:['$20 \%$',
                         {'x_corr':0.2,
                          'y_corr':-.22,
                          'draw_line':True}],
                   30.0:['$30 \%$',
                         {'x_corr':0.1,
                          'y_corr':0.0,
                          'draw_line':True}],
                   50.0:['$50 \%$',
                         {'x_corr':0.1,
                          'y_corr':0.0,
                          'draw_line':True}],
                  100.0:['$100 \%$',
                         {'x_corr':0.3,
                          'y_corr':.3,
                          'draw_line':True}],
                },
        'u_title':r'Contrast Limiting Resolution \
                   (Line Pairs/mm)',
        'v_title':r'Contrast',
        'wd_title':r'Photocathode Illuminance \
                    (-log$_{10}$(lux))',
        'wd_tick_side':'right',
        'wd_text_format':r"$%3.2f$",
        'wd_tick_levels':3,
        'wd_tick_text_levels':1,
        'wd_title_distance_center':2.2,
        'manual_x_scale':True,
        'x_min':1.5,
        'x_max':7.0,
        'mirror_x':True,
        'u_scale_opposite':True,
        'isopleth_values':[[15.0,20.0,'x']],
                }

main_params={
        'filename':'Type5-Isopleth.pdf',
        'paper_height':10.0,
        'paper_width':20.0,
        'block_params':[block_20_params],
        'transformations':[('rotate',0.01),('scale paper',)],
        'title_y': -4,
        'title_box_width':20,
        'title_str':r'Contrast Limiting Resolution vs.\
            Photocathode Illuminance for an Image-Intensifier\
            Photocathode Responsivity of 200$\mu$A/lm',
      'isopleth_params':[
           {'color':'black',
            'linewidth':'thin',
            'linestyle':'dashed',
            'circle_size':0.08,
            'transparency':0.0,
           },
           ],
                }

Nomographer(main_params)
```



Contrast Limiting Resolution vs. Photocathode Illuminance for an Image-Intensifier Photocathode Responsivity of 200μA/lm

An Electro-Optic Nomogram

Now we have the three nomograms needed to recreate the entire electro-optic nomogram on page 84: the nomogram on the left side found on page 29, the nomogram on the lower right found on page 52, and the contour block we just designed.

So we take the three scripts and put them together into one very large script. We add tags to the end scales of the earlier two nomograms so they will align with the x-axis and y-axis of the contour block. For those aligned scales we also remove the titles, tick marks and tick labels so we don't duplicate those on the contour block (after we verify that they match). Because the x-axis of the contour block is now a linear scale of the exponent of the actual value, we replace log(u) in that scale of the lower nomogram with -u over the same range 1.5 to 7.0 and plot it as a linear scale.

The order of the blocks listed in the main parameters turns out to be important in orienting the blocks in the overall nomogram. I suggest listing the contour block first in order to lock in the orientation of its axes. Then the other nomogram blocks will line up as we want relative to those axes. I found originally that the lower nomogram blocks proceeded vertically upward from the x-axis of the contour block rather than downward, but changing the order of those blocks in the main parameters caused them to proceed downward instead.

Beyond the change of units of the x-axis in the contour block the individual nomograms combine to make up the nomogram in the *Electro-Optics Handbook*. We've constructed a very complicated nomogram here by building smaller pieces and connecting them together. A good number of lines of the script are added here simply to place text onto the nomogram—this is a heavily annotated nomogram. Isopleths are also drawn as guidelines. It looks very nice when printed on large paper, as you can see if you zoom in on it.

The PyNomo script and output PDF graphic are shown below and are available online for download.[59]

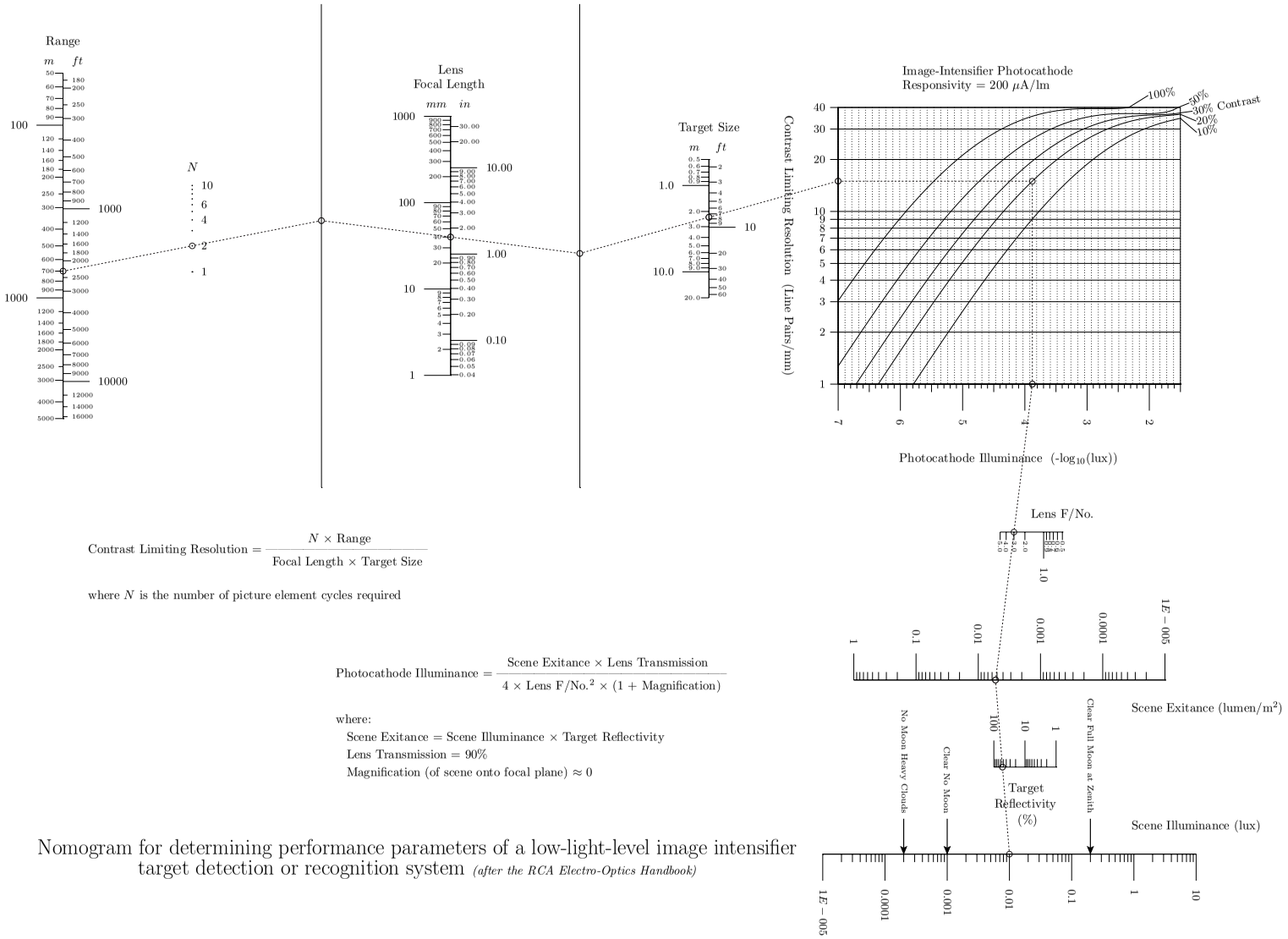
[59] *http://www.myreckonings.com/pynomo11/EO-Isopleths.py* and *http://www.myreckonings.com/pynomo11/EO-Isopleths.pdf*

```
### EO-Isopleths.py ###

from pynomo.nomographer import *

##########################################################
########## Previous Type 3 Example 2 Nomogram ##########
##########################################################

from pynomo.nomographer import *

Range_meters_params={
        'tag':'range',
        'u_min':5000.0,
        'u_max':50.0,
        'function':lambda u:-log10(u),
        'scale_type':'log',
        'title':r'Range',
        'title_y_shift':0.8,
        'text_format':r"$%3.0f$",
        'tick_levels':4,
        'tick_text_levels':3,
        'extra_titles':[
              {'dx':-1.25,
               'dy':0.25,
               'text':r'\small $m$',
               'width':5,
              }]
                }

Number_of_Cycles_params={
        'u_min':1.0,
        'u_max':10.0,
        'function':lambda u:-log10(u),
        'scale_type':'log',
        'title':r'$N$',
        'title_y_shift':0.4,
        'text_format':r"$3.0f$",
        'tick_levels':2,
        'tick_text_levels':1,
        'scale_type':'manual point',
        'manual_axis_data': {1.0:'1',
                             2.0:'2',
                             3.0:'',
                             4.0:'4',
                             5.0:'',
                             6.0:'6',
                             7.0:'',
                             8.0:'',
                             9.0:'',
                             10.0:'10'
                             },
                }

Focal_Length_mm_params={
        'tag':'focal',
        'u_min':1.0,
        'u_max':1000.0,
        'function':lambda u:log10(u),
        'scale_type':'log',
        'title':r'Lens',
        'title_y_shift':1.2,
        'tick_side':'left',
        'tick_levels':2,
        'text_format':r"$%3.0f$",
        'tick_text_levels':2,
        'extra_titles':[
              {'dx':-1.75,
               'dy':0.8,
               'text':r'Focal Length',
               'width':5,
              },
              {'dx':-1.4,
               'dy':0.25,
               'text':r'\small $mm$',
               'width':5,
              }]
                }

Target_Size_meters_params={
        'tag':'size',
        'u_min':0.5,
        'u_max':20.0,
        'function':lambda u:log10(u),
        'scale_type':'log',
        'title':r'Target Size',
        'title_y_shift':0.8,
        'tick_side':'left',
        'tick_levels':2,
        'text_format':r"$%3.1f$",
        'tick_text_levels':2,
        'extra_titles':[
              {'dx':-1.25,
               'dy':0.25,
               'text':r'\small $m$',
               'width':5,
              }]
                }

Resolution_params={
        'tag':'resolution',
        'u_min':1.0,
        'u_max':40.0,
        'function':lambda u:log10(u),
        'scale_type':'log',
        # Remove titles, ticks marks and tick labels as they
        # duplicate those on the y-axis of the Contour Block.
        #'title':r'Resolution',
        #'title_y_shift':0.8,
        #'text_format':r"$%3.1f$",
        #'tick_levels':3,
        #'tick_text_levels':3,
        'tick_levels':0,
        'tick_text_levels':0,
                }

block_1_params={
        'block_type':'type_3',
        'width':10.0,
        'height':10.0,
        'reference_titles':['',''],
        'f_params':[Range_meters_params,
                    Number_of_Cycles_params,
                    Focal_Length_mm_params,
                    Target_Size_meters_params,
                    Resolution_params],
        'isopleth_values':[[700,2,40,'x',15]],
                }

### TYPE 8 SINGLE-SCALE BLOCK ###
Range_feet_params={
        'tag':'range',
        'u_min':5000.0*3.28,
        'u_max':50.0*3.28,
        'function':lambda u:-log10(u),
        'scale_type':'log',
        'align_func':lambda u:u/3.28,
        'title':r'\small $ft$',
        'title_x_shift':0.4,
        'tick_side':'left',
        'text_format':r"$%3.0f$",
```

```
        'tick_levels':4,
        'tick_text_levels':3,
                }

block_2_params={
        'block_type':'type_8',
        'f_params':Range_feet_params,
        'isopleth_values':[['x']],
                 }

### TYPE 8 SINGLE-SCALE BLOCK ###
Focal_Length_inches_params={
        'tag':'focal',
        'u_min':1000.0/25.4,
        'u_max':1.0/25.4,
        'function':lambda u:log10(u),
        'scale_type':'log',
        'align_func':lambda u:u*25.4,
        'title':r'\small $in$',
        'title_x_shift':0.4,
        'tick_side':'left',
        'text_format':r"$%3.2f$",
        'tick_levels':2,
        'tick_text_levels':2,
                }

block_3_params={
        'block_type':'type_8',
        'f_params':Focal_Length_inches_params,
        'isopleth_values':[['x']],
                }

### TYPE 8 SINGLE-SCALE BLOCK ###
Target_Size_feet_params={
        'tag':'size',
        'u_min':0.5*3.281,
        'u_max':20.0*3.281,
        'function':lambda u:log10(u),
        'scale_type':'log',
        'align_func':lambda u:u/3.281,
        'title':r'\small $ft$',
        'title_x_shift':0.35,
        'tick_side':'right',
        'text_format':r"$%3.0f$",
        'tick_levels':2,
        'tick_text_levels':2,
                }

block_4_params={
        'block_type':'type_8',
        'f_params':Target_Size_feet_params,
        'isopleth_values':[['x']],
                }

##########################################################
########### Previous Type 5 Example Nomogram ############
##########################################################

const_A = 1.5
const_B = 0.36
const_C = 4.25
const_D = .134
const_E = 150
const_F = 0.15

def f1(x,v):
  return const_A*sin(const_B*(-x+const_C*v**const_D))+ \
         (v/const_E)/(-x+const_F+(v/200))**2

block_20_params={
        'block_type':'type_5',
        'u_tag':'resolution',
        'wd_tag':'illum',
        'width':10.0,
        'height':13.0,
        'u_func':lambda u:log10(u),
        'u_values':[1.0,2.0,3.0,4.0,5.0,
                    6.0,7.0,8.0,9.0,
                    10.0,20.0,30.0,40.0],
        'u_text_format':r"$%3.0f$ ",
        'v_func':f1,
        'v_values':[10.0,20.0,30.0,50.0,100.0],
        'v_manual_axis_data':
               {
                   10.0:['$10 \%$',
                        {'x_corr':0.22,
                         'y_corr':-.45,
                         'draw_line':True}],
                   20.0:['$20 \%$',
                        {'x_corr':0.2,
                         'y_corr':-.22,
                         'draw_line':True}],
                   30.0:['$30 \%$',
                        {'x_corr':0.1,
                         'y_corr':0.0,
                         'draw_line':True}],
                   50.0:['$50 \%$',
                        {'x_corr':0.1,
                         'y_corr':0.0,
                         'draw_line':True}],
                  100.0:['$100 \%$',
                        {'x_corr':0.3,
                         'y_corr':.3,
                         'draw_line':True}],
                },
        #'v_text_format':r"$%3.0f$ \%% ",
        'u_title':r'Contrast Limiting Resolution \
                   (Line Pairs/mm)',
        'v_title':r'Contrast',
        'wd_title':r'Photocathode Illuminance \
                    (-log$_{10}$(lux))',
        'wd_tick_side':'right',
        'wd_text_format':r"$%3.2f$",
        'wd_tick_levels':3,
        'wd_tick_text_levels':1,
        'wd_title_distance_center':2.2,
        'manual_x_scale':True,
        'x_min':1.5,
        'x_max':7.0,
        'mirror_x':True,
        'u_scale_opposite':True,
        'isopleth_values':[['x',20.0,'x']],
                }

##########################################################
###### Previous Compound Type 1 Example Nomogram #######
##########################################################

### TYPE 1 3-LINE BLOCK ###
Photocathode_Illuminance_params={
        # Replace log(u) with -u over range 1.5 to 7.0
        # and plot linearly to match the wd-scale on the
        # x-axis of the Contour Plot, and assign the
        # illum tag here as was assigned to that axis.
        'tag':'illum',
        'u_min':1.5,
        'u_max':7.0,
        'function':lambda u:u,
        'scale_type':'linear',
        # Remove title, ticks marks and tick labels, as
```

```
        # they duplicate those already drawn on the
        # y-axis of the Contour Block.
        #'title':r'Photocathode Illuminance (lux)',
        #'title_y_shift':0.7,
        #'tick_side':'left',
        #'text_format':r"$%2,5G$",
        #'tick_levels':2,
        #'tick_text_levels':1,
        'tick_levels':0,
        'tick_text_levels':0,
                }

FNumber_params={
        # Replace 2*log10(u) with -2*log10(u)
        'u_min':0.5,
        'u_max':5.0,
        'function':lambda u:-2*log10(u),
        'scale_type':'log',
        'title':r'Lens F/No.',
        'title_y_shift':0.4,
        'text_format':r"$%3.1f$",
        'tick_levels':2,
        'tick_text_levels':2,
                }

Scene_Exitance_1_params={
        # Replace -log10(0.225*u) with log10(0.225*u)
        # Replace 2*log10(u) with -2*log10(u)
        'tag':'exitance',
        'u_min':0.00001,
        'u_max':1.0,
        'function':lambda u:log10(0.225*u),
        'scale_type':'log',
        'title':r'Scene Exitance (lumen/m$^2$)',
        'title_x_shift':1.2,
        'title_y_shift':-0.9,
        'tick_side':'left',
        'text_format':r"$%2.5G$",
        'tick_levels':2,
        'tick_text_levels':1,
                }

block_11_params={
        'block_type':'type_1',
        'width':15.0,
        'height':15.0,
        'f1_params':Scene_Exitance_1_params,
        'f2_params':FNumber_params,
        'f3_params':Photocathode_Illuminance_params,
        'isopleth_values':[['x',3,'x']],
                }


### TYPE 1 3-LINE BLOCK ###
Scene_Illuminance_params={
        'u_min':0.00001,
        'u_max':10.0,
        'function':lambda u:log10(u),
        'scale_type':'log',
        'title':r'Scene Illuminance (lux)',
        'title_y_shift':0.7,
        'text_format':r"$%2.5G$",
        'tick_levels':2,
        'tick_text_levels':1,
        'extra_params':[
            {'tick_side':'left',
             'scale_type':'manual arrow',
             'arrow_length':1.0,
             'manual_axis_data':
               {
                0.2:'\scriptsize Clear Full Moon at Zenith',
                0.001:'\scriptsize Clear No Moon',
                0.0002:'\scriptsize No Moon Heavy Clouds',
              },
            }],
               }

Target_Reflectivity_params={
        'u_min':1.0,
        'u_max':100.0,
        'function':lambda u:log10(0.01*u),
        'scale_type':'log',
        'title':r'Target',
        'title_y_shift':-0.7,
        'title_x_shift':-0.85,
        'tick_side':'left',
        'text_format':r"$%3.0f$",
        'tick_levels':2,
        'tick_text_levels':1,
        'extra_titles':[
              {'dx':-2.45,
               'dy':-1.15,
               'text':r'Reflectivity',
               'width':5,
              },
              {'dx':-1.8,
               'dy':-1.6,
               'text':r'(\%)',
               'width':5,
              }]
                }

Scene_Exitance_2_params={
        'tag':'exitance',
        'u_min':0.00001,
        'u_max':1.0,
        'function':lambda u:-log10(u),
        'scale_type':'log',
        'title':r'',
        'text_format':r"$%2.5G$",
        'tick_levels':0,      # Don't draw duplicate ticks
        'tick_text_levels':0, # Don't draw duplicate labels
                }

block_12_params={
        'block_type':'type_1',
        'width':15.0,
        'height':15.0,
        'f1_params':Scene_Exitance_2_params,
        'f2_params':Target_Reflectivity_params,
        'f3_params':Scene_Illuminance_params,
        'isopleth_values':[['x','x',0.01]],
                }

#########################################################
######## Main Parameters of Overall Nomogram ###########
#########################################################

main_params={
    'filename':'EO-Isopleths.pdf',
    'paper_height':24.0,
    'paper_width':32.0,
    'block_params':[block_20_params,block_11_params,
           block_12_params,block_1_params,block_2_params,
           block_3_params,block_4_params],
    'transformations':[('rotate',0.01),('scale paper',)],
    'title_x': 10,
    'title_y': 0,
    'title_box_width': 22,
    'title_str':r'\LARGE Nomogram for determining\
```

```
        performance parameters of a low-light-level\
        image intensifier target detection or recognition\
        system \it (after the RCA Electro-Optics Handbook)',
        'extra_texts':[
              {
               'x':0.0,
               'y':8.5,
               'text':r'Contrast Limiting Resolution =\
                   ---------------------------------------',
               'width':15,
              },
              {
               'x':6.2,
               'y':8.8,
               'text':r'$N$ $\times$ Range',
               'width':5,
              },
              {
               'x':5.2,
               'y':8.15,
               'text':r'Focal Length $\times$ Target Size',
               'width':5,
              },
              {
               'x':0.0,
               'y':7.2,
               'text':r'where $N$ is the number of picture\
                       element cycles required',
               'width':15,
              },
              {
               'x':7.0,
               'y':5.0,
               'text':r'Photocathode Illuminance =\
-----------------------------------------------------',
               'width':15,
              },
              {
               'x':11.85,
               'y':5.3,
               'text':r'Scene Exitance $\times$\
                        Lens Transmission',
               'width':10,
              },
              {
               'x':11.7,
               'y':4.65,
               'text':r'4 $\times$ Lens F/No.$^2$\
                          $\times$ (1 + Magnification)',
               'width':10,
              },
              {
               'x':7.0,
               'y':3.7,
               'text':r'where:',
               'width':5,
              },
              {
               'x':7.3,
               'y':3.2,
               'text':r'Scene Exitance = Scene Illuminance\
                        $\times$ Target Reflectivity',
               'width':15,
              },
              {
               'x':7.3,
               'y':2.7,
               'text':r'Lens Transmission = 90\%',
               'width':15,
              },
              {
               'x':7.3,
               'y':2.2,
               'text':r'Magnification (of scene onto\
                        focal plane) $\approx$ 0',
               'width':15,
              },
              {
               'x':23.0,
               'y':22.0,
               'text':r'Image-Intensifier Photocathode',
               'width':15,
              },
              {
               'x':23.0,
               'y':21.6,
               'text':r'Responsivity = 200 $\mu$A/lm',
               'width':15,
               }],
         'isopleth_params':[
              {'color':'black',
               'linewidth':'thin',
               'linestyle':'dashed',
               'circle_size':0.08,
               'transparency':0.0,
              },
              ],
       'make_grid':False,
                 }

Nomographer(main_params)
```



$$\text{Contrast Limiting Resolution} = \frac{N \times \text{Range}}{\text{Focal Length} \times \text{Target Size}}$$

where N is the number of picture element cycles required

$$\text{Photocathode Illuminance} = \frac{\text{Scene Exitance} \times \text{Lens Transmission}}{4 \times \text{Lens F/No.}^2 \times (1 + \text{Magnification})}$$

where:
Scene Exitance = Scene Illuminance × Target Reflectivity
Lens Transmission = 90%
Magnification (of scene onto focal plane) ≈ 0

Nomogram for determining performance parameters of a low-light-level image intensifier target detection or recognition system *(after the RCA Electro-Optics Handbook)*

19 Editing Your PyNomo Nomogram

You may want to embellish the nomogram beyond what has been described here, such as adding another graphic showing the machine part or a geometrical figure to which the nomogram applies. One option is to use directly in PyNomo the drawing commands available in the PyX package installed as part of the PyNomo installation. Another option is to import your PyNomo nomogram into an image editor.

Editing with PyX

PyNomo offers two main parameters that provide access to the lower-level PyX drawing commands:

- *pre_func*: execute the PyX commands as defined in the named function *before* drawing the nomogram, using coordinates in cm defined in PyNomo for the nomogram
- *post_func*: execute the PyX commands as defined in the named function *after* drawing the nomogram, using coordinates in cm defined in PyNomo for the nomogram

The PyX website and Reference Manual describe a number of commands for editing your nomogram.[60] We can define a function before the main parameters that contains any PyX drawing commands to perform before PyNomo draws the nomogram, say, *pre(c)*. We can also define a function, say *post(c)*, containing PyX drawing commands to perform after PyNomo draws the nomogram. Setting *make_grid* to True overlays a grid on the nomogram, which is critical for determining the coordinates needed by the PyX commands while writing the script. Then we add the lines `'post_func':post,` and `'pre_func':pre,` to the main parameters.

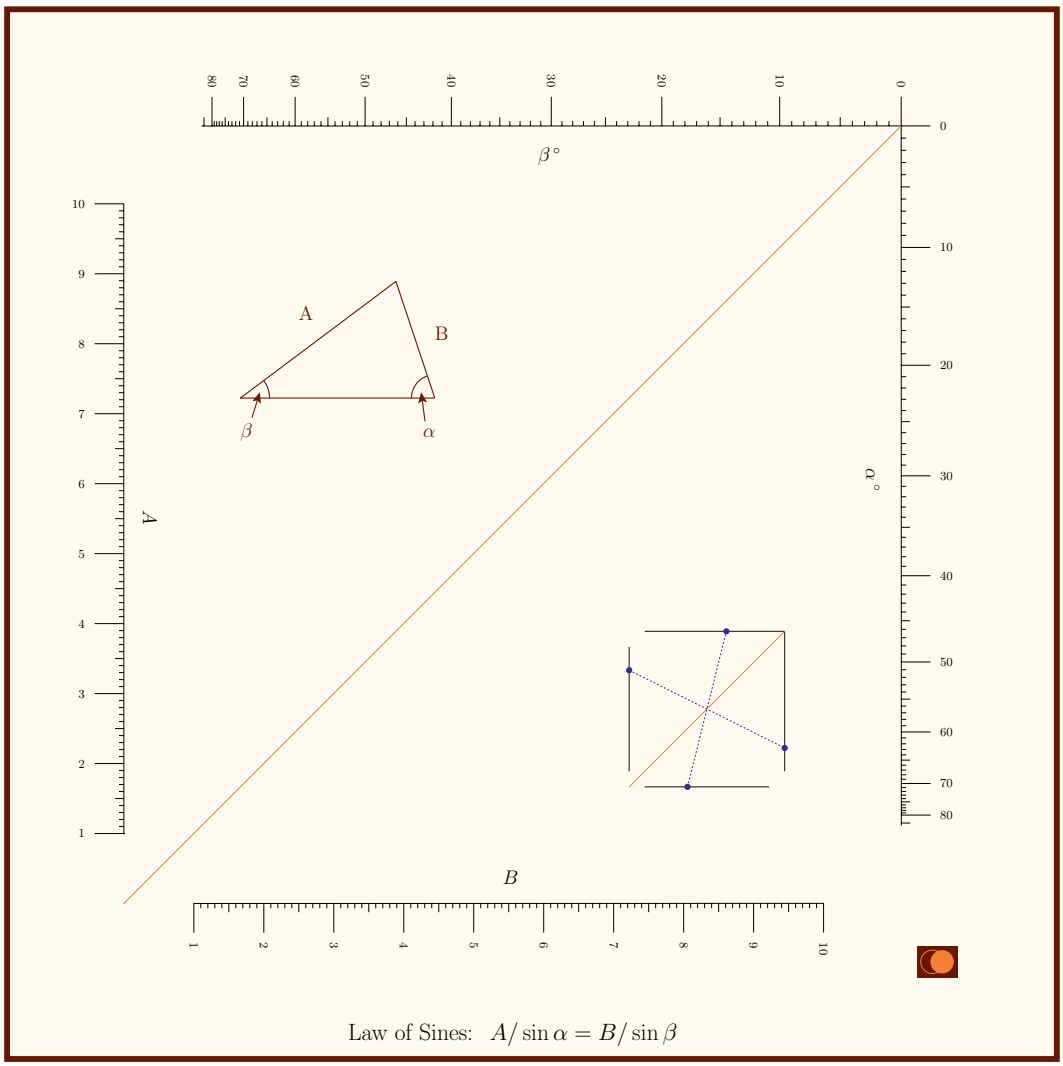
On the next page is an example of a nomogram, our earlier Type 4 Proportion nomogram, that has a variety of PyX drawing commands added to it. The logo in the lower right corner is drawn only to show some more drawing commands. More details are found in the PyX references.

- *rect*: a background rectangle in *pre(c)*, the only PyX drawing prior to the nomogram, with its starting x,y coordinates and its width and height, with a Sepia-colored line of width 0.15cm and filled with a light ivory color[61]
- *line*: a line segment with starting and ending x,y coordinates, here a Sepia-colored thick line (which could also be drawn in PyNomo by the *line_params* parameter)
- *text*: text with an x-y coordinate, here Sepia-colored (which could also be drawn in PyNomo by the *extra_texts* parameter)
- *arc*: an arc with the x,y coordinates of the center, the radius in cm, and the initial and final angle in degrees, specified here as Sepia-colored and thick (note the additional *path.path* operation required)
- *arrow*: a line with an arrow decoration, with its starting and ending x,y coordinates specified
- *rect*: another rectangle with its starting x,y coordinates and its width and height, filled because *c.fill* was used instead of *c.stroke*, and colored Sepia
- *circle*: a circle with its center x,y coordinates and its radius, in a filled version using *c.fill* and a hollow version using *c.stroke*.

The PyNomo script and output PDF graphic below and are available online for download.[62]

[60] *http://pyx.sourceforge.net/* and *http://pyx.sourceforge.net/manual.pdf*
[61] Floral White, with the rgb fractions calculated from *http://www.tayloredmktg.com/rgb/*
[62] *http://www.myreckonings.com/pynomo11/Type4-PyX.py* and *http://www.myreckonings.com/pynomo11/Type4-PyX.pdf*

```
### Type4-PyX.py ###

from pynomo.nomographer import *

A_params={
      'u_min':1.0,
      'u_max':10.0,
      'function':lambda u:u,
      'title':r'\Large $A$',
      'tick_levels':3,
      'tick_text_levels':1,
      'tick_side':'left',
      'title_draw_center':True,
      'title_opposite_tick':True,
             }
Alpha_params={
      'u_min':0.0,
      'u_max':89.0,
      'function':lambda u:sin(u*pi/180.0),
      'title':r'\Large $\alpha \thinspace ^\circ$',
      'tick_levels':3,
      'tick_text_levels':1,
      'tick_side':'right',
      'title_draw_center':True,
      'title_distance_center':0.9,
      'title_opposite_tick':True,
             }
B_params={
      'u_min':1.0,
      'u_max':10.0,
      'function':lambda u:u,
      'title':r'\Large $B$',
      'tick_levels':3,
      'tick_text_levels':1,
      'tick_side':'right',
      'title_draw_center':True,
      'title_opposite_tick':False,
             }
Beta_params={
      'u_min':0.0,
      'u_max':89.0,
      'function':lambda u:sin(u*pi/180.0),
      'title':r'\Large $\beta \thinspace ^\circ$',
      'tick_levels':3,
      'tick_text_levels':1,
      'tick_side':'left',
      'title_draw_center':True,
      'title_distance_center':0.9,
      'title_opposite_tick':False,
             }

block_1_params={
      'block_type':'type_4',
      'reference_color':color.cmyk.Orange,
      'f1_params':A_params,
      'f2_params':Alpha_params,
      'f3_params':B_params,
      'f4_params':Beta_params,
             }

def pre(c):
    c.stroke(path.rect(-3, -4, 27, 27),
        [style.linewidth(0.15),
        color.cmyk.Sepia,
        deco.filled([color.rgb(1.0,0.98,0.94)])])

def post(c):
    # Draw and triangle and labels
    c.stroke(path.line(3, 13, 8, 13) +
          path.line(8, 13, 7, 16) +
          path.line(7, 16, 3, 13),
          [style.linewidth.thick,color.cmyk.Sepia])
    c.text(4.5,15,r"\Large A",[color.cmyk.Sepia])
    c.text(8,14.5,r"\Large B",[color.cmyk.Sepia])
    c.text(3,12,r"\Large $\beta$",[color.cmyk.Sepia])
    c.text(7.7,12,r"\Large $\alpha$",[color.cmyk.Sepia])
    # Draw angle curves
    c.stroke(path.path(path.arc(3, 13, 0.75, 0, 36.9)),
          [style.linewidth.thick,color.cmyk.Sepia])
    c.stroke(path.path(path.arc(8, 13, 0.6, 108.4, 180)),
          [style.linewidth.thick,color.cmyk.Sepia])
    # Draw arrows from labels to angle curves
    c.stroke(path.line(3.3, 12.5, 3.5, 13.15),
        [style.linewidth.thick,
        color.cmyk.Sepia,deco.earrow.large])
    c.stroke(path.line(7.75, 12.4, 7.65, 13.15),
        [style.linewidth.thick,
        color.cmyk.Sepia,deco.earrow.large])
    # Draw logo
    c.fill(path.rect(20.4, -1.9, 1.05, 0.8),
          [color.cmyk.Sepia])
    c.fill(path.circle(21.05, -1.5, 0.3),
          [color.cmyk.Orange])
    c.stroke(path.circle(20.8, -1.5, 0.3),
          [style.linewidth.thin,color.cmyk.Orange])

main_params={
      'filename':'Type4-Pyx.pdf',
      'paper_height':20.0,
      'paper_width':20.0,
      'block_params':[block_1_params],
      'transformations':[('rotate',0.01),('scale paper',)],
      'title_x':10,
      'title_y': -3.5,
      'title_box_width': 10,
      'title_str':r'\LARGE Law of Sines:  \
                   $A / \sin \alpha=B / \sin \beta$',
      'make_grid':False,
      'draw_lines':True,
      'line_params': [
          {'coords':[[13,3.4,13,6.6],[17,3.4,17,7],
                     [13.4,3,16.6,3],[13.4,7,17,7]],
           'line_style':[color.cmyk.Black,
                         style.linewidth.thick,
                         style.linestyle.solid],
           'circle_size':0.0,
           'circle_color':color.cmyk.Black,
          },
          {'coords':[[13,3,17,7]],
           'line_style':[color.cmyk.Orange,
                         style.linewidth.thin,
                         style.linestyle.solid],
           'circle_size':0.0,
           'circle_color':color.cmyk.Orange,
          },
          {'coords':[[13,6,17,4],[14.5,3,15.5,7]],
           'line_style':[color.cmyk.Blue,
                         style.linewidth.thin,
                         style.linestyle.dashed],
           'circle_size':0.08,
           'circle_color':color.cmyk.Blue,
          },
          ],
          'pre_func':pre,
          'post_func':post,
              }

Nomographer(main_params)
```



Vector vs. Raster Images

The output of PyNomo is a *vector* image, or in other words the image is drawn on your printer or screen according to instructions within the PDF file. Therefore, the image can be zoomed in to any level and it remains sharp, as you can see by zooming in when viewing the PDF file. It can also be printed or plotted on any size of paper without loss of clarity or precision, which is essential for nomograms where you might want to produce large drawings for greatest accuracy. PyNomo can also produce output files in the EPS format (if the output file is specified as *yourfilename.eps*, and this is also a vector format that may be more universally accepted by image editors.

There are also *raster* images that you can create for specific sizes, such as .JPG, .PNG, .BMP, .GIF, .TIFF and so forth. If you zoom into these or print them with enlargement, they begin to show rough edges (or become fuzzy) because the image consists of discrete pixels. Raster image editors also produce much larger files.

Using a Raster Image Editor

So one way to edit your PyNomo nomogram is in a raster image editor. The big disadvantage in converting the PyNomo output to a raster image is the large file sizes that are needed to preserve high resolution, and for this reason I would not recommend using a raster image editor. If you do, Adobe Photoshop is one example of such an editor, but there are many others such as the powerful yet free GIMP software.[63] You can often open the PDF or EPS output file of PyNomo in these programs, but since these are vector images you will be asked to specify the size that you want your nomogram to converted to as a raster image. After that you are limited to that size for full resolution viewing and printing. If you can't directly input the PyNomo file, you can use an image converter to convert your nomogram to the desired format. In general, the PNG format is much better for the nomograms (and somewhat smaller in file size) than the JPG format because it tends to preserve edges such as the scales. A free online PDF image converter is also available.[64] (However, it appears to be under construction at the moment. There is a free 30-day trial of its PDF Converter on that site, though.) Once you input the nomogram as a raster image you can edit it as you like depending on the features of the editor.

Using a Vector Image Editor

It is almost essential, though, to edit the nomogram while retaining its vector nature. The file size will be much smaller and the printed resolution will be limited only by the resolution of the printer or plotter, which is very important given that the sharpness of a nomogram is critical for the precision of its calculations. Adobe Illustrator is one (very expensive) means of editing a vector image—a free but powerful alternative is Inkscape.[65] There are many other vector image editors of varying sophistication.[66] CAD drawing software such as Microsoft Visio or AutoCad can also edit vector images. Illustrator can directly open the PDF or EPS format produced by PyNomo, but missing fonts will be converted to Courier. Inkscape can open EPS files, and it can open PDF files if the free image converter pstoedit is installed.[67] You can also use pstoedit standalone to convert your PDF file to vector formats accepted by other vector editors, such as EMF, DXF for CAD packages, and SWF for Flash editors. Most vector editors (Illustrator, Inkscape and Visio included) accept SVG format. There are expensive PDF-to-SVG converters, and pstoedit can do this with a shareware plugin, but there is a free online converter from Texterity[68] that I've tested. Some warnings are generated about undefined fonts, and in fact the title strings are sometimes run together, but when I input the resulting SVG file into Visio I could simply delete them and add them back in as new text.

There is another free option, and that is to use the MiKTeX distribution of LaTeX that is already installed as part of the PyNomo installation. I would really only recommend this approach if you know LaTeX.[69] Within a LaTeX script file you can input the PDF file and use, or example, the TikZ drawing package to add a wide variety of graphics.[70]

---

[63] *http://www.gimp.org/*

[64] *http://www.coolutils.com/Online-PDF-Converter.php*

[65] *http://www.inkscape.org/*

[66] See a very nice comparison at *http://www.smashingmagazine.com/2008/12/05/20-vector-graphic-editors-reviewed/* or a simple list at *http://en.wikipedia.org/wiki/List_of_vector_graphics_editors*

[67] *http://www.pstoedit.net/*

[68] *http://www.texterity.com/artstech/freesvg/submit/*

[69] This paper you are reading is written in LaTeX.

[70] A gallery of TikZ examples can be found at *http://www.texample.net/tikz/examples/all/*. The outstanding TikZ users manual can be found at *http://www.ctan.org/tex-archive/graphics/pgf/base/doc/generic/pgf/pgfmanual.pdf* TikZ requires adding the *pgf* package to MiKTeX.

20 The Benefits of Using PyNomo to Draw Nomograms

As I've tried to point out in the examples here, PyNomo offers significant advantages in creating precision nomograms. PyNomo supports a number of common types of nomograms in equation form without having to derive the standard determinants, while offering the determinant form as another more general type. PyNomo can also provide transformations and optimization. And perhaps its most amazing capability is to create grid and compound nomograms for more than three variables. The examples of these on the PyNomo website are outstanding. PyNomo provides a fast way of testing and fine-tuning the layout and optimizations very, very quickly.[71] I relied on it extensively in preparing a variety of nomograms for a math journal article and a PowerPoint presentation. I continue to try things with PyNomo and I am surprised at what it can produce. Dr. Roschier has provided a software tool of real benefit to those of us with an interest in nomography.

[71] Joe Marasco provides an example of how he improved one of his nomogram designs by experimenting with PyNomo at *http://www.barbecuejoe.com/scan.htm*

Index