djboulet

Release 0.0.0.1

Daniel Boulet

Jan 26, 2022

CONTENTS

CHAPTER

ONE

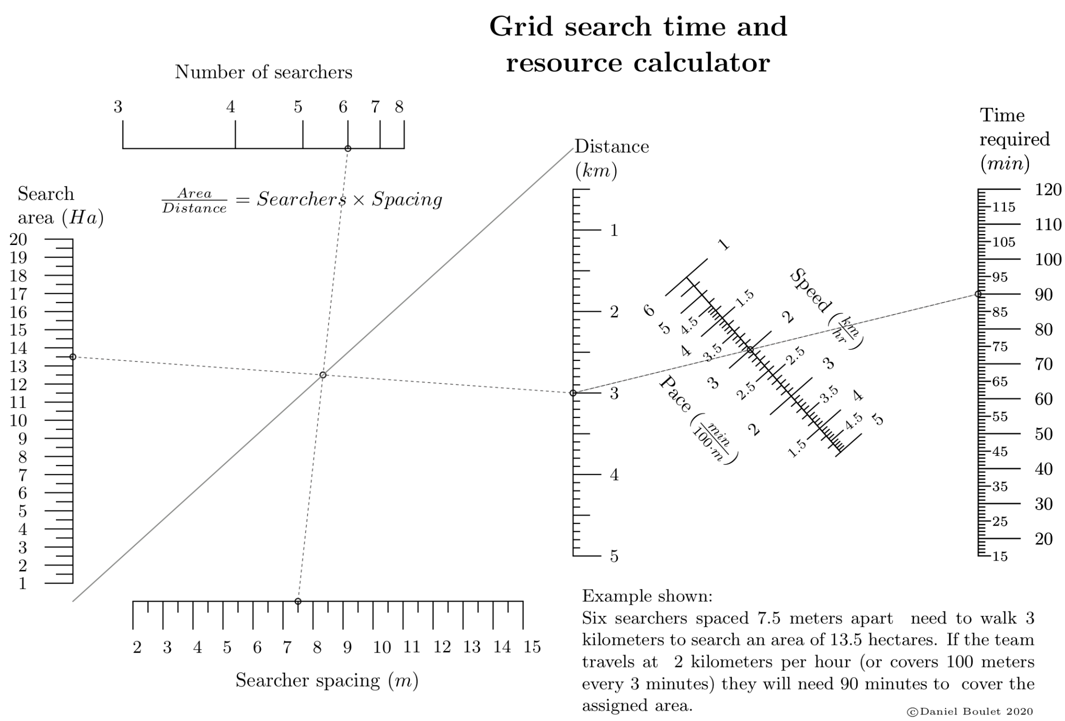
MYNOMOS

These nomographs were created with the PyNomo package. These examples help illustrate the range of problems that can be solved with nomographs. Some of the examples below are also included in the PyNomo documentation.

1.1 Search Planner

Grid searches are slow and labour intensive. Search planner will want to allocate scarce resources (searchers and time) appropriately. Knowing the length of time required to search a parcel of land can help assign resources effeciently.

1.1.1 Nomograph



1.1.2 Source code

```
"""
search_planner.py

Nomogram to calculate search effort for ground search activities.
along with this program.  If not, see < http: // www.gnu.org/licenses/>.
"""
from pynomo.nomographer import Nomographer
import sys
outputfile = sys.argv[0].split('.')[0]+'.pdf'

from pyx import *
text.set(text.LatexEngine)

searchers = {
    'ID': 'searchers',
    'u_min': 3.0,
    'u_max': 8.0,
    'function': lambda u: 1.0 / u,
    'scale_type': 'linear',
    'title': r'Number of searchers',
    'title_distance_center': -1.25,
    'tick_levels': 1,
    'tick_text_levels': 1,
    'tick_side': 'left',
    'grid_length_0': 0.5,
    'text_distance_0': 0.75,
    'title_draw_center': True,
    # 'title_opposite_tick': False,
    # 'text_horizontal_align': True,
    'extra_angle': 90.0

}


hectares = {
    'ID': 'hectares',
    'u_min': 1.0,
    'u_max': 20.0,
    'function': lambda u: u*10000.0,
    # 'title': r'Search area ($Ha$)',
    'extra_titles': [
        {
            'text': r'Search \par area ($Ha$)',
            'dx': -1.5,
            'dy': 0.7
        }
    ],

    'tick_levels': 3,
    'tick_text_levels': 1,
    'tick_side': 'left',
    'grid_length_0': 0.5,
    'grid_length_1': 0.3,
    'text_distance_0': 0.8,
    'scale_type': 'linear smart'
}

distance1 = {
    'ID': 'distance1',
    'tag': 'distance',
    'scale_type': 'linear smart',
    'u_min': 0.5,
    'u_max': 5.0,
    'function': lambda u: u*1000,
```

(continues on next page)

(continued from previous page)

```
    'align_function': lambda u: u * 1000,
    'extra_titles': [
        {
            'text': r'Distance \par ($km$)',
            'dx': -0.5,
            'dy': 0.65
        }
    ],
    'grid_length_0': 0.5,
    'grid_length_1': 0.3,
    'text_distance_0': 0.65,
    'tick_levels': 3,
    'tick_text_levels': 1,
    'tick_side': 'right',
}

spacing = {
    'ID': 'spacing',
    'u_min': 2.0,
    'u_max': 15.0,
    'function': lambda u: u,
    # 'scale_type': 'manual point',
    # 'manual_axis_data': {2.5: '2.5', 5.0: '5.0', 10: '10', 15: '15'},
    'title': r'Searcher spacing ($m$)',
    'title_draw_center': True,
    'title_opposite_tick': False,
    'tick_text_levels': 1,
    'tick_levels': 2,
    'grid_length_0': 0.5,
    'grid_length_1': 0.2,
    'text_distance_0': .8,
    'title_distance_center': -1.5,
    'text_horizontal_align': True,
    'extra_angle': 90.0,
    'turn_relative': True,

    # 'extra_params': [
    #     {
    #         'u_min': 2.5,
    #         'u_max': 15.0,
    #         'scale_type': 'linear',
    #         'tick_levels': 3,
    #         'tick_text_levels': 1,
    #         'tick_side': 'right',
    #     }
    # ]
}

area_block = {
    'ID': 'area_block',
    'block_type': 'type_4',
    'height': 8,
    'width':8,
    'f1_params': hectares,
    'f2_params': distance1,
    'f3_params': spacing,
    'f4_params': searchers,
    'reference_color': color.cmyk.Gray,
    'padding': .9,
    'isopleth_values': [['x', 3.0, 7.5, 6]],
}

distance2 = {
    # distance in meters (u3)
    'ID': 'distance2',
```

(continues on next page)

(continued from previous page)

```
    'tag': 'distance',
    'u_min': 0.5,
    'u_max': 5.0,
    'scale_type': 'manual_point',
    'function': lambda u: u*1000,
    'tick_levels': 0,
    'tick_text_levels': 0,
    # 'title_x_shift': 1.0,
}

speed_km_h = {
    # speed in km/hr
    'ID': 'speed_in_km_per_hr',
    'tag': 'speed',
    'u_min': 1.0,
    'u_max': 5.0,
    'function': lambda u: u*1000/3600,
    'title': r'Speed ($\frac{km}{hr}$)',
    'tick_levels': 3,
    'tick_text_levels': 2,
    'title_x_shift': 0.0,
    'title_draw_center': True,
    'title_distance_center': -1.4,
    'grid_length_0': 0.5,
    'grid_length_1': 0.3,
    'text_distance_0': 0.8,
    'text_distance_1':0.35

}


time_required = {
    # time in minutes
    'ID': 'time_required',
    'u_min': 15.0,
    'u_max': 120.0,
    'function': lambda u: u*60,
    # 'title': r'Time required ($min$)',
    'tick_levels': 3,
    'tick_text_levels': 2,
    'title_x_shift': 1.0,
    'extra_titles': [
        {
            'text': r'Time \par required \par ($min$)',
            'dx': -0.5,
            'dy': 1.2
        }
    ],

}

distance_speed_time_block = {
    'block_type': 'type_2',
    'f1_params': distance2,
    'f2_params': speed_km_h,
    'f3_params': time_required,
    'isopleth_values': [[3.0, 'x', 90]],
    'mirror_x': True
}

speed_min_100meter = {
    # speed in minutes / 100 meters
    'ID': 'speed_sec_meter',
    'u_min': 0.72*100/60,
    'u_max': 3.6*100/60,
```

(continues on next page)

(continued from previous page)

```
    'function': lambda u: u/100*60,
    'title': r'Pace ($\frac{min}{100\cdot m}$)',
    'tick_levels': 3,
    'tick_text_levels': 2,
    'title_draw_center': True,
    'scale_type': 'linear smart',
    'tick_side': 'left',
    'title_distance_center': -1.6,
    'grid_length_0': 0.5,
    'grid_length_1': 0.3,
    'text_distance_0': 0.8,
    'text_distance_1': 0.35

}


distance_speed_time_block2 = {
    'ID': 'distance_speed_time_block2',
    'block_type': 'type_2',
    'f1_params': time_required,
    'f2_params': speed_min_100meter,
    'f3_params': distance2,
    'isopleth_values': [[90, 'x', 3.0]],
    'width':10,
    'mirror_x': True,

}

main_params = {
    'filename': outputfile,
    'paper_height': 8,
    'paper_width': 16,
    'block_params': [area_block, distance_speed_time_block, distance_speed_time_block2],
    'transformations': [('rotate', 0.01), ('scale paper',),  ],
    'title_str': r'\Large \textbf{Grid search time and resource calculator}',
    'title_y': 10.0,
    'title_x':10,
    # 'make_grid': True,
    # 'debug': True,
    'isopleth_params': [
        # {'color': 'black',
        #  'linewidth': 'thin',
        #  'linestyle': 'solid',
        #  'circle_size': 0.08,
        #  'transparency': 0.0,
        #  },
        {'color': 'black',
         'linewidth': 'thin',
         'linestyle': 'dashed',
         'circle_size': 0.05,
         'transparency': 0.25,
         },
    ],
    'extra_texts': [
        {
            'x': 9.0,
            'y': 0.0,
            'text': r'\small \noindent Example shown: \par \noindent Six searchers spaced 7.5 meters apart \
                need to walk 3 kilometers to search an area of 13.5 hectares. If the team travels at \
                2 kilometers per hour (or covers 100 meters every 3 minutes) they will need 90 minutes to \
                cover the assigned area.',
            'width': 8,
        },
        {
            'x': 14.2,
```

(continues on next page)

(continued from previous page)

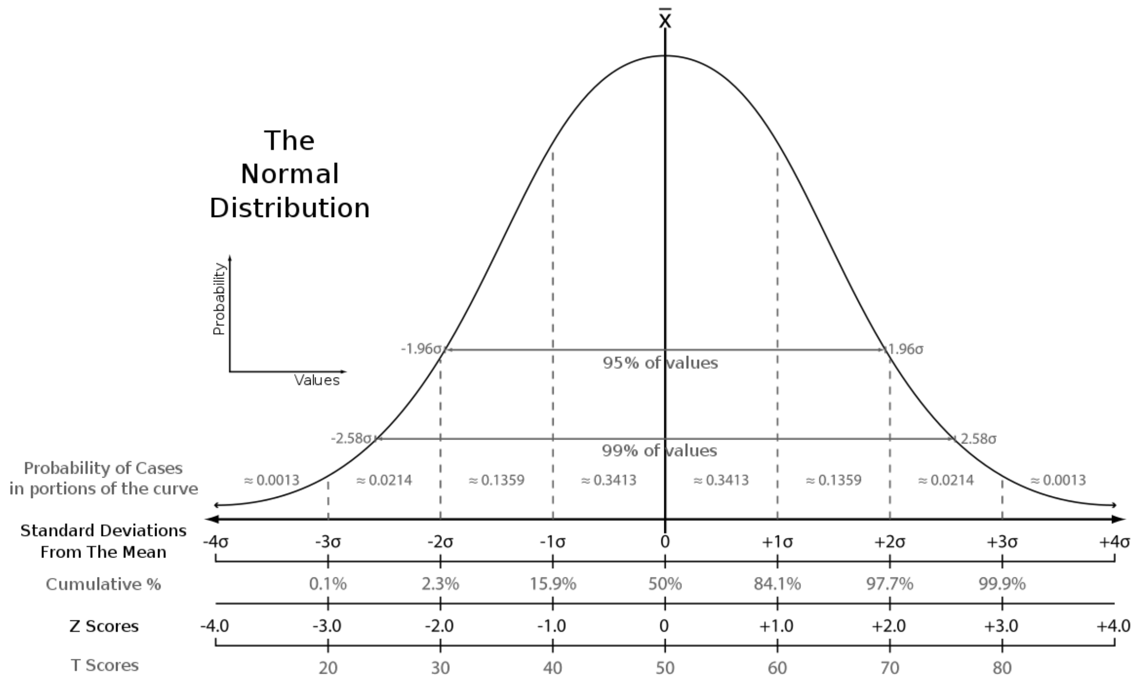
```
            'y': -2.0,
            'text': r'\tiny \copyright Daniel Boulet  2020',
            'width': 5.0,
        },
        {
            'x': 1,
            'y': 7.0,
            'text': r'$\frac{Area}{Distance}=Searchers \times Spacing$',
            'width': 10.0,

        },

    ],


}


Nomographer(main_params)
```

1.2 Z score calculator

1.2.1 Theory and background

This example extends Pynomo's versatility by using external libraries. Python's `scipy` library is the engine behind this nomograph which calculates the area under a normal distribution curve between two Z scores (one negative, the other positive).

To calculate the area between two Z scores (Z_{upper}, Z_{lower}) of a normal distribution one must compute the difference between the respective probability density functions $\mathrm{PDF}(Z_{upper})$ and $\mathrm{PDF}(Z_{lower})$. [wik07]



Recall that the functional relationship for a **Type 1** block is:

$F1(u_1) + F2(u_2) + F3(u_3) = 0$

and

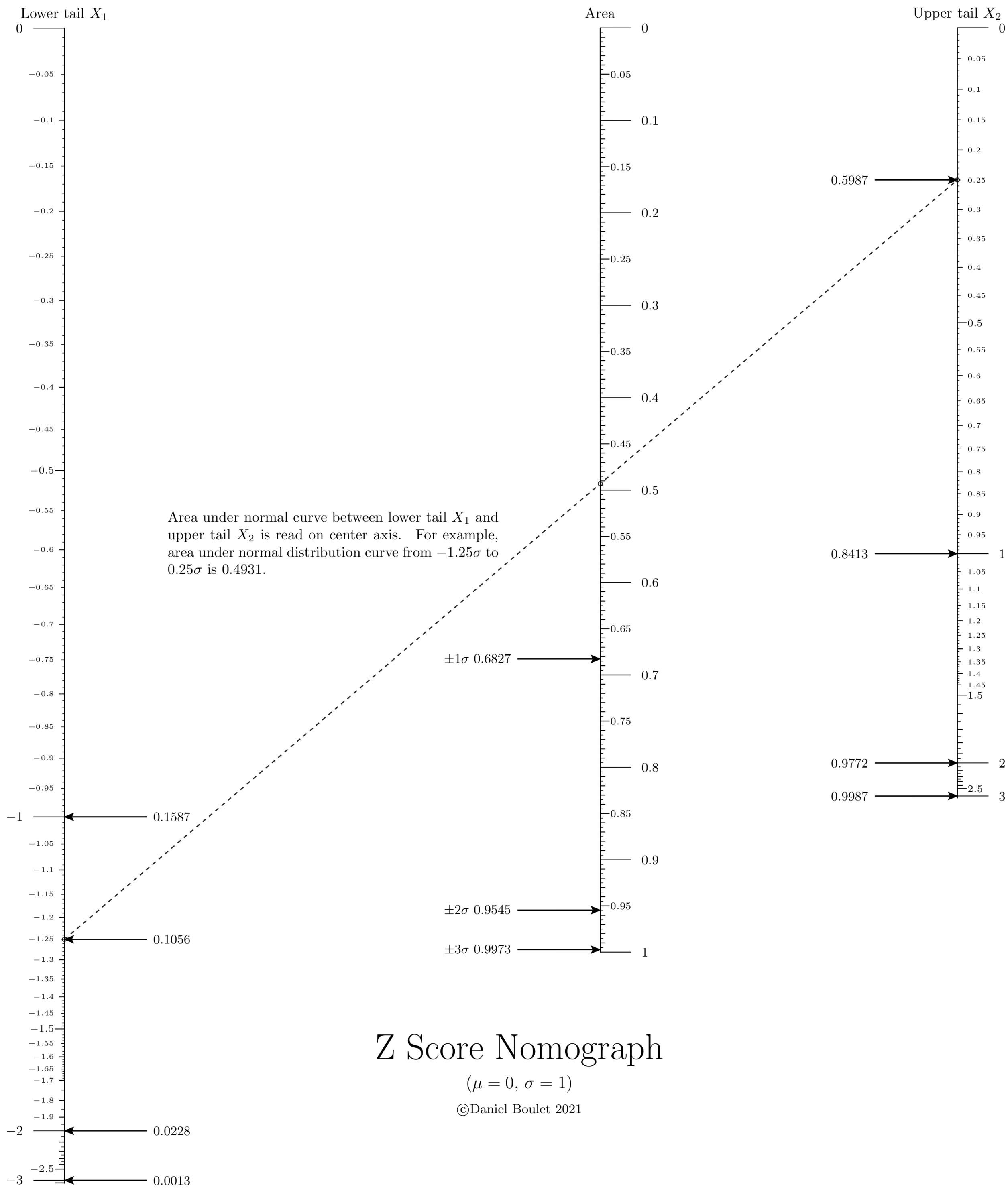
$\text{Area} = \text{PDF}(Z_{upper}) - \text{PDF}(Z_{lower})$

therefore

$\text{PDF}(Z_{upper}) - Area - \text{PDF}(Z_{lower}) = 0$.

Two **Type 8** axes are aligned with $\text{PDF}(Z_{upper})$ and $\text{PDF}(Z_{lower})$ to align a Z score with its associated PDF.

1.2.2 Nomograph



1.2.3 Source code

```
"""
    zscore.py

    Nomograph to calculate area under normal curve from z-score.
"""

import scipy.stats as stats
from pynomo.nomographer import *
import sys
outputfile = sys.argv[0].split('.')[0]+'.pdf'

sys.path.insert(0, "..")
# allows use of latex commands in PyX such as \frac{a}{b} and \par
from pyx import *

pyx.text.set(text.LatexEngine)


def cdf(u):
    return stats.norm.cdf(u)


def ppf(u):
    return stats.norm.ppf(u)


lmin = 0.0001
lmax = 0.9999

sd1 = cdf(1.0) - cdf(-1.0)
sd2 = cdf(2.0) - cdf(-2.0)
sd3 = cdf(3.0) - cdf(-3.0)


leftpdf = {
    "tag": "a",
    "u_min": lmin,
    "u_max": 0.5,
    "function": lambda u: (u),
    "scale_type": "manual point",
    # extra parameters
    "extra_params": [
        {
            "scale_type": "manual arrow",
            "manual_axis_data": {
                cdf(-1.0): r"%4.4f" % cdf(-1.0),
                cdf(-2.0): r"%4.4f" % cdf(-2.0),
                cdf(-3.0): r"%4.4f" % cdf(-3.0),
                cdf(-1.25): r"%4.4f" % cdf(-1.25),
            },
            "arrow_length": 2.0,
        },
    ],
}


leftz = {
    "tag": "a",
    "u_min": ppf(lmin),
    "u_max": ppf(0.5),
    "function": lambda u: cdf(u),
    "align_func": lambda u: cdf(u),
    "title": "Lower tail $X_1$",
    "tick_levels": 5,
```

(continues on next page)

(continued from previous page)

```
    "tick_text_levels": 4,
    "scale_type": "linear smart",
    "tick_side": "left",
}


rightpdf = {
    "tag": "c",
    "u_min": 0.5,
    "u_max": lmax,
    "function": lambda u: -(u),
    "scale_type": "manual point",
    # extra parameters
    "extra_params": [
        {
            "scale_type": "manual arrow",
            "tick_side": "left",
            "manual_axis_data": {
                cdf(3.0): r"%4.4f" % cdf(3.0),
                cdf(2.0): r"%4.4f" % cdf(2.0),
                cdf(1.0): r"%4.4f" % cdf(1.0),
                cdf(0.25): r"%4.4f" % cdf(0.25),
            },
            "arrow_length": 2.0,
        },
    ],
}


rightz = {
    "tag": "c",
    "u_min": ppf(0.5),
    "u_max": ppf(lmax),
    "function": lambda u: cdf(u),
    "align_func": lambda u: cdf(u),
    "title": "Upper tail $X_2$",
    "tick_levels": 5,
    "tick_text_levels": 4,
    "scale_type": "linear smart",
}

leftblock2 = {"block_type": "type_8", "f_params": leftz, "isopleth_values": [["x"]]}


rightblock2 = {"block_type": "type_8", "f_params": rightz, "isopleth_values": [["x"]]}


delta = {
    "u_min": 0.0,
    "u_max": 1.0,
    "function": lambda u: u,
    "scale_type": "linear smart",
    "title": "Area",
    "tick_levels": 5,
    "tick_text_levels": 4,
    "extra_params": [
        {
            "tick_side": "left",
            "scale_type": "manual arrow",
            "manual_axis_data": {
                sd1: r"$\pm 1 \sigma$ %4.4f" % sd1,
                sd2: r"$\pm 2 \sigma$ %4.4f" % sd2,
                sd3: r"$\pm 3 \sigma$ %4.4f" % sd3,
            },
            "arrow_length": 2.0,
```

(continues on next page)

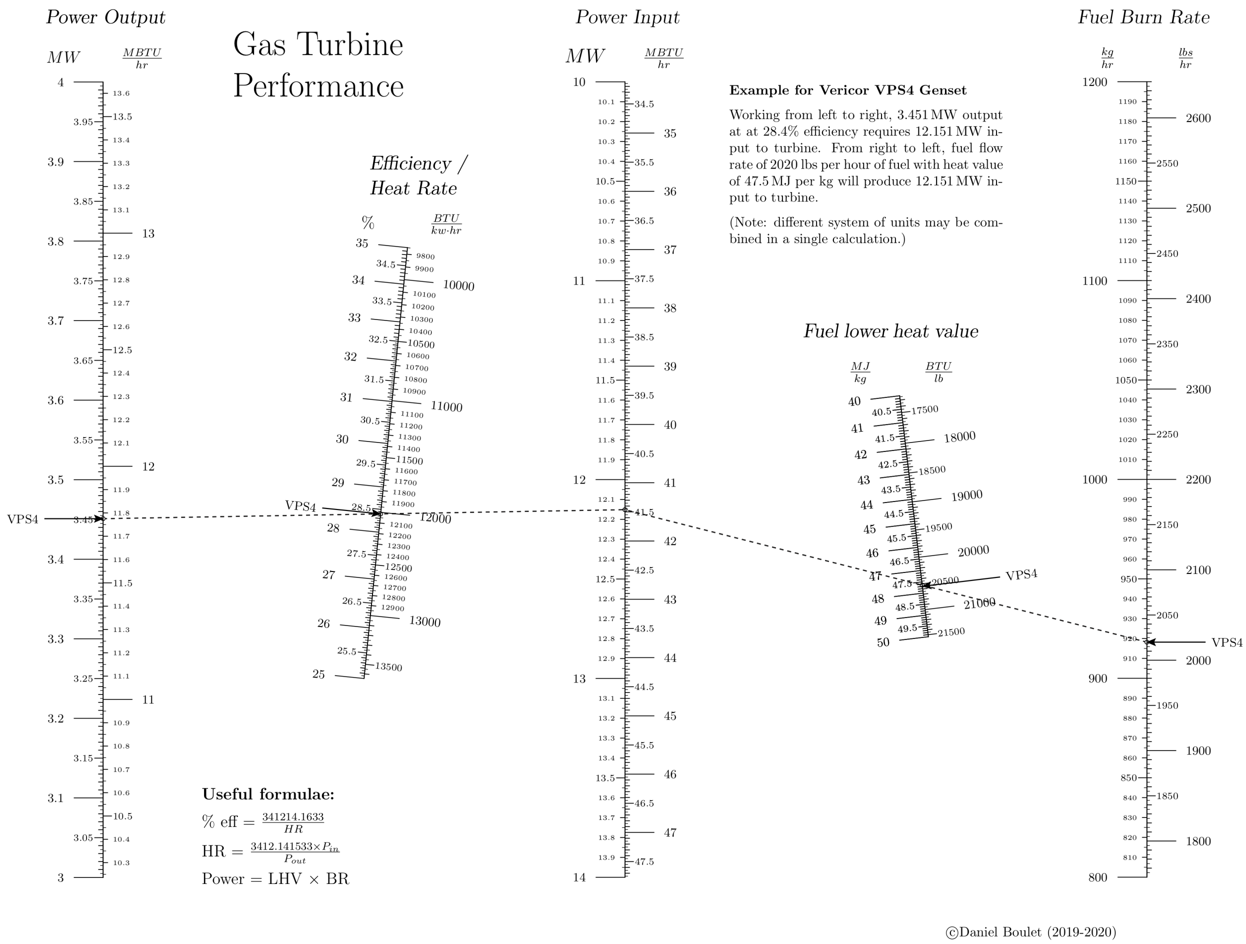
(continued from previous page)

```
        },
    ],
}

block_diff = {
    "block_type": "type_1",
    "f1_params": leftpdf,
    "f2_params": delta,
    "f3_params": rightpdf,
    "proportion": 1.5,
    "isopleth_values": [[cdf(-1.25), "x", cdf(0.25)]],
}


main_params = {
    "filename": outputfile,
    "paper_height": 11.0 * 2.54,
    "paper_width": 8.5 * 2.54,
    "block_params": [block_diff, leftblock2, rightblock2],
    "cdfations": [("rotate", 0.01), ("scale paper",), ("polygon",)],
    "title_x": 11.0,
    "title_y": 3.0,
    "title_box_width": 15.0,
    "title_str": r"\Huge Z Score Nomograph \par \medskip \large ($\mu = 0$, $\sigma = 1$) \par \medskip  \small \copyright Daniel Boulet  2021",
    # 'make_grid':True,
    "extra_texts": [
        {
            "x": 2.5,
            "y": 16.0,
            "text": r"\noindent Area under normal curve between lower tail $X_1$ and upper tail $X_2$ is read on center axis. \
                For example, area under normal distribution curve from $%g \sigma$ to $%g \sigma$ is $%4.4f$."
            % (-1.25, 0.25, cdf(0.25) - cdf(-1.25)),
            "width": 8.0,
        },
    ],
}

Nomographer(main_params)
```

1.3 VPS Generator Performance

This nomograph estimates power generation based on fuel consumption of a Vericor VPS4 Genset.

1.3.1 Nomograph



1.3.2 Source code

```python
"""
    vps4performance.py

    Gas turbine performance nomograph based on Vericor VPS4 genset.
"""
from pynomo.nomographer import *
import sys

sys.path.insert(0, "..")
outputfile = sys.argv[0].split(".")[0] + ".pdf"

# allows use of latex commands in PyX such as \frac{a}{b} and \par
from pyx import *

text.set(text.LatexEngine)


def watts2btu_hr(watts):
    return watts * 3.412141633


```

(continues on next page)

(continued from previous page)

```
def btu_hr2watts(btu_hr):
    return btu_hr / 3.412141633


def btu2joule(btu):
    return btu * 1055.06


def joule2btu(joule):
    return joule / 1055.06


"""
this part is the left handside of the total graph
"""

leftside_u1_left = {
    "tag": "left_1",
    # range of values (nominal 3451000.0 W )
    "u_min": 3.0,
    "u_max": 4.0,
    # functions
    "function": lambda u: u * 1000000.0,
    # scale type and detail
    "tick_levels": 4,
    "tick_text_levels": 3,
    "scale_type": "linear smart",
    # tick and title locations
    "title": r"\large $MW$",
    "title_x_shift": -1.0,
    "title_y_shift": 0.5,
    "tick_side": "left",
    # extra parameters
    "extra_params": [
        {
            "scale_type": "manual arrow",
            "manual_axis_data": {
                3.451: r"VPS4",
            },
            "arrow_length": 1.5,
        },
    ],
    # additional title information
    "extra_titles": [
        {
            "dx": -2.0,
            "dy": 1.5,
            "text": "\slshape \Large Power Output",
            # 'width': 5.0
        },
    ],
}

leftside_u1_right = {
    "tag": "left_1",
    # range of values (nominal 11778263.0 BTU/hr )
    "u_min": 3.0 * watts2btu_hr(1000000.0) / 1000000.0,
    "u_max": 4.0 * watts2btu_hr(1000000.0) / 1000000.0,
    # functions
    "function": lambda u: u * 1000000.0,
    "align_func": lambda u: u / watts2btu_hr(1000000.0) * 1000000.0,
    # scale type and detail
    "tick_levels": 4,
    "tick_text_levels": 3,
    "scale_type": "linear smart",
```

(continues on next page)

(continued from previous page)

```
    # tick and title locations
    "title": r"\large ${\frac{MBTU}{hr}}$",
    "title_x_shift": 1.0,
    "title_y_shift": 0.5,
    "tick_side": "right",
}

leftside_u2_left = {
    # range of values (nominal 28%)
    "u_min": 0.25 * 100.0,
    "u_max": 0.35 * 100.0,
    # functions
    "function": lambda u: u / 100.0,
    # scale type and detail
    "tick_levels": 4,
    "tick_text_levels": 3,
    "scale_type": "linear smart",
    # tick and title locations
    "title": r"\large $\%$",
    "title_x_shift": -1.0,
    "title_y_shift": 0.5,
    "tick_side": "left",
    # additional title information
    "extra_titles": [
        {
            "dx": -1.5,
            "dy": 2.0,
            "text": "\slshape \Large Efficiency / \par Heat Rate",
            # 'width': 5.0
        },
    ],
    # extra parameters
    "extra_params": [
        {
            "scale_type": "manual arrow",
            "manual_axis_data": {
                28.4: r"VPS4",
            },
            "arrow_length": 1.5,
        },
    ],
}

leftside_u2_right = {
    # range of values (nomninal 12025 BTU/kWh )
    "u_min": watts2btu_hr(100000.0) / 35.0,
    "u_max": watts2btu_hr(100000.0) / 25.0,
    # functions
    "function": lambda u: watts2btu_hr(1000.0) / (u),
    "text_format": r"$%4.0f$",
    # 'align_func': lambda u:341300.0/u,
    # scale type and detail
    "tick_levels": 4,
    "tick_text_levels": 3,
    "scale_type": "linear smart",
    # tick and title locations
    "title": r"\large ${\frac{BTU}{kw \cdot hr}}$",
    "title_x_shift": 1.0,
    "title_y_shift": 0.5,
    "tick_side": "right",
    "title_rotate_text": True,
}

leftside_u3_left = {
    "tag": "mid",
```

(continues on next page)

(continued from previous page)

```
    # range of values (nominal 12325000.0 W )
    "u_min": 10.0,
    "u_max": 14.0,
    # functions
    "function": lambda u: u * 1000000.0,
    # scale type and detail
    "tick_levels": 4,
    "tick_text_levels": 3,
    "scale_type": "linear smart",
    # tick and title locations
    "title": r"\Large $MW$",
    "title_x_shift": -1.0,
    "title_y_shift": 0.5,
    "tick_side": "left",
    # additional title information
    "extra_titles": [
        {
            "dx": -1.8,
            "dy": 1.5,
            "text": "\slshape \Large Power Input",
            # 'width': 5.0
        },
    ],
}

leftside_u3_right = {
    # range of values (nominal 42065225.0 BTU/hr )
    "u_min": 10.0 * watts2btu_hr(1000000.0) / 1000000.0,
    "u_max": 14.0 * watts2btu_hr(1000000.0) / 1000000.0,
    # functions
    "function": lambda u: u * 1000000.0,
    "align_func": lambda u: u / watts2btu_hr(1000000.0) * 1000000.0,
    # scale type and detail
    "tick_levels": 4,
    "tick_text_levels": 3,
    "scale_type": "linear smart",
    # tick and title locations
    "title": r"\large ${\frac{MBTU}{hr}}$",
    "title_x_shift": 1.0,
    "title_y_shift": 0.5,
    "tick_side": "right",
}

"""

this part is the right hand side of the total graph

"""

rightside_u3_left = {
    "tag": "mid",
    # range of values (nominal 12325000.0 W )
    "u_min": 10.0,
    "u_max": 14.0,
    # functions
    "function": lambda u: u * 1000000.0,
    # scale type and detail
    "scale_type": "manual point",
}

rightside_u3_right = {
    # range of values (nominal 42065225.0 BTU/hr )
    "u_min": 10.0 * watts2btu_hr(1000000.0) / 1000000.0,
    "u_max": 14.0 * watts2btu_hr(1000000.0) / 1000000.0,
    # functions
```

(continues on next page)

(continued from previous page)

```
    "function": lambda u: u * 1000000.0,
    "align_func": lambda u: u / watts2btu_hr(1000000.0) * 1000000.0,
    # scale type and detail
    "scale_type": "manual point",
}


rightside_u2_left = {
    # range of values (nominal 45000000 MJ / kg )
    "u_min": 40.0,
    "u_max": 50.0,
    # functions
    "function": lambda u: 1.0 / (u * 1000000.0),
    # scale type and detail
    "tick_levels": 4,
    "tick_text_levels": 3,
    "scale_type": "linear smart",
    # tick and title locations
    "title": r"\large ${\frac{MJ}{kg}}$",
    "title_x_shift": -1.0,
    "title_y_shift": 0.5,
    "tick_side": "left",
    # extra parameters
    "extra_params": [
        {},
    ],
    # additional title information
    "extra_titles": [
        {
            "dx": -3.0,
            "dy": 1.5,
            "text": r"\slshape \Large Fuel lower heat value",
            # 'width': 5.0
        },
    ],
}

rightside_u2_right = {
    # range of values (nominal 20000 BTU/lb)
    "u_min": 40000000.0 / 2321.13,
    "u_max": 50000000.0 / 2321.13,
    # functions
    "function": lambda u: 1.0 / (u),
    # 'align_func':lambda u: u*2326.0/1000000.0,
    # scale type and detail
    "tick_levels": 4,
    "tick_text_levels": 3,
    "scale_type": "linear smart",
    "text_format": r"$%4.0f$",
    # tick and title locations
    "title": r"\large ${\frac{BTU}{lb}}$",
    "title_x_shift": 1.0,
    "title_y_shift": 0.5,
    "tick_side": "right",
    # extra parameters
    "extra_params": [
        {
            "scale_type": "manual arrow",
            "manual_axis_data": {
                20548.0: r"VPS4",
            },
            "arrow_length": 2.0,
        },
    ],
}
```

(continues on next page)

(continued from previous page)

```


rightside_u1_left = {
    "tag": "right_1",
    # range of values (nominal 1000 kg/hr )
    "u_min": 800.0,
    "u_max": 1200.0,
    # functions
    "function": lambda u: u / 3600.0,
    # scale type and detail
    "tick_levels": 4,
    "tick_text_levels": 3,
    "scale_type": "linear smart",
    # tick and title locations
    "title": r"\large ${\frac{kg}{hr}}$",
    "title_x_shift": -1.0,
    "title_y_shift": 0.5,
    "tick_side": "left",
    # additional title information
    "extra_titles": [
        {
            "dx": -2.3,
            "dy": 1.5,
            "text": "\slshape \Large Fuel Burn Rate",
            # 'width': 5.0
        },
    ],
}

rightside_u1_right = {
    "tag": "right_1",
    # range of values (nominal 2020 lb/hr )
    "u_min": 800 * 2.2,
    "u_max": 1200.0 * 2.2,
    # functions
    "function": lambda u: u,
    "align_func": lambda u: u / 2.2,
    # scale type and detail
    "tick_levels": 4,
    "tick_text_levels": 3,
    "scale_type": "linear smart",
    # tick and title locations
    "title": r"\large ${\frac{lbs}{hr}}$",
    "title_x_shift": 1.0,
    "title_y_shift": 0.5,
    "tick_side": "right",
    # extra parameters
    "extra_params": [
        {
            "scale_type": "manual arrow",
            "manual_axis_data": {
                2020.0: r"VPS4",
            },
            "arrow_length": 1.5,
        },
    ],
}


leftside_SI_block = {
    "block_type": "type_2",
    "f1_params": leftside_u1_left,
    "f2_params": leftside_u2_left,
    "f3_params": leftside_u3_left,
    "isopleth_values": [[3.451, 28.4, "x"]],
```

(continues on next page)

(continued from previous page)

```
}

leftside_IMP_block = {
    "block_type": "type_2",
    "f1_params": leftside_u1_right,
    "f2_params": leftside_u2_right,
    "f3_params": leftside_u3_right,
    "isopleth_values": [[watts2btu_hr(3.451), "x", watts2btu_hr(3.451) / 0.284]],
}

rightside_SI_block = {
    "block_type": "type_2",
    "f1_params": rightside_u1_left,
    "f2_params": rightside_u2_left,
    "f3_params": rightside_u3_left,
    "isopleth_values": [[2020.0 / 2.2, "x", "x"]],
    "mirror_x": True,
}

rightside_IMP_block = {
    "block_type": "type_2",
    "f1_params": rightside_u1_right,
    "f2_params": rightside_u2_right,
    "f3_params": rightside_u3_right,
    "isopleth_values": [[2020.0, "x", watts2btu_hr(3.451) / 0.284]],
    "mirror_x": True,
}

main_params = {
    "filename": outputfile,
    "paper_height": 8.0 * 2.54,
    "paper_width": 10.5 * 2.54,
    "block_params": [
        leftside_SI_block,
        rightside_SI_block,
        leftside_IMP_block,
        rightside_IMP_block,
    ],
    "transformations": [("rotate", 0.01), ("scale paper",)],
    "title_str": r"\Huge Gas Turbine \par Performance",
    "title_x": 5.5,
    "title_y": 21.0,
    "extra_texts": [
        {
            "text": r"\large \textbf{Useful formulae:} \par \medskip \% eff = ${\frac{341214.1633}{HR}}$ \par \
↪medskip  HR = ${\frac{3412.141533 \times P_{in}}{P_{out}}}$  \par \medskip Power = LHV $\times$ BR",
            "x": 2.0,
            "y": 2.0,
            "width": 10.0,
        },
        {
            "text": r"\copyright Daniel Boulet (2019-2020)",
            "x": 21.0,
            "y": -1.5,
        },
        {
            "x": 16.0,
            "y": 20.0,
            "text": r"\noindent \textbf{Example for Vericor VPS4 Genset} \par \medskip \noindent \normalsize Working␣
↪from left to right, 3.451\,MW output at at 28.4\% efficiency requires 12.151\,MW input to turbine. From right to␣
↪left, fuel flow rate of 2020 lbs per hour of fuel with heat value of 47.5\,MJ per kg will produce 12.151\,MW␣
↪input to turbine. \par \medskip \noindent (Note: different system of units may be combined in a single␣
↪calculation.)",
            "width": 7.0,
        },
```

(continues on next page)

(continued from previous page)

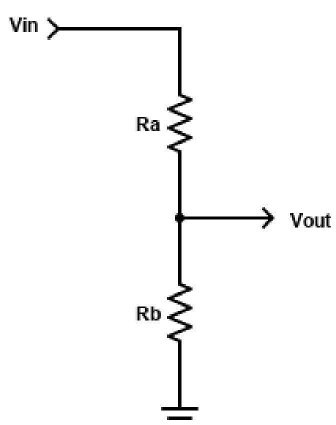
```
    ],
    # 'make_grid': True,
}
Nomographer(main_params)
```

1.4 Voltage Divider

1.4.1 Theory and background

In electronics, resistive voltage dividers are used for a variety of purposes. The formula for a resistive voltage divider is [Wik21c]:

$$\frac{V_{out}}{V_{in}} = \frac{R_b}{(R_a + R_b)}$$

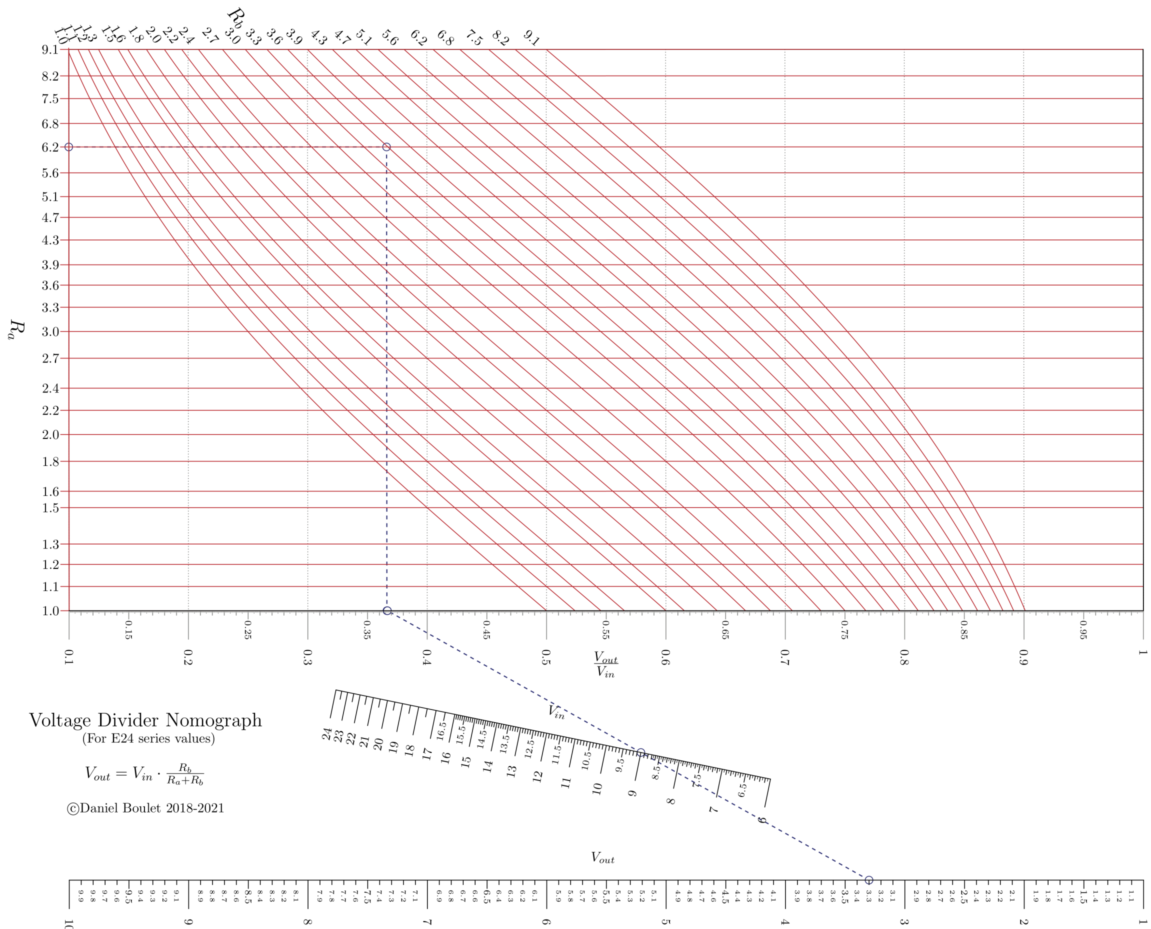


When designing voltage dividers, constraints may demand engineers choose resistors from a set of "preferred values" [Wik21b]. These values are discrete and engineers must select the best combination of resistors based on tolerance and the available preferred values. Similar voltage ratios can be obtained with different combinations of resistor values.

The voltage divider nomograph links input voltage (V_{in}), output voltage (V_{out}) and a pair of resistor values into a single nomograph. Pynomo's **Type 5** blocks are well suited for plotting relationships between pairs of discrete values. A vertical line dropped from the intersection of R_a values and R_b values reveals the V_{out} / V_{in} voltage ratio. Alignment with a **Type 2** block allows the engineer to determine V_{out} given V_{in} (or vice versa).

Of greater benefit is the ability to quickly determine the optimum pair of resistor values for a given application. For example, given an input voltage (9V) and desired output voltage (3.3V), the engineer draws a straight line from the V_{out} axis, through the V_{in} axis to the base of the voltage ratio graph. A perpendicular line is then drawn from the base to the top of the graph. The vertical line's nearest approach to the intersection of R_a and R_b values represents the best combination of resistor values. It can be quickly shown that one combination of values (R_a = 6.2 and R_b = 3.6) will produce an output voltage very close to the desired value (3.3061V).

1.4.2 Nomograph



1.4.3 Source code

```
"""
    voltdiv_E24_resistors.py

    Nomogram to calculate resistor values for simple voltage divider.  This
    along with this program.  If not, see <http://www.gnu.org/licenses/>.
"""

from pynomo.nomographer import *
import sys
sys.path.insert(0, "..")
outputfile = sys.argv[0].split('.')[0]+'.pdf'

from pyx import *
pyx.text.set(text.LatexEngine)

import numpy as np

resistors = [
        1.0,        1.1,        1.2,
        1.3,        1.5,        1.6,
        1.8,        2.0,        2.2,
```

(continues on next page)

(continued from previous page)

```
        2.4,        2.7,        3.0,
        3.3,        3.6,        3.9,
        4.3,        4.7,        5.1,
        5.6,        6.2,        6.8,
        7.5,        8.2,        9.1
]

# Type 5 contour
def f1(x, u):
    return np.log10(u * (1 - x) / x)


block_1_params = {
    'width': 12.0,
    'height': 25.0,
    'block_type': 'type_5',

    'u_func': lambda u: np.log10(u),
    'u_values': resistors,
    'u_axis_color': pyx.color.cmyk.Red,
    'u_title': r'\Large{$R_a$}',
    'u_text_format': r"\normalsize{$%3.1f$}",

    'v_func': f1,
    'v_values': resistors,
    'v_axis_color': pyx.color.cmyk.Red,
    'v_title': r'\Large{$R_b$}',
    'v_text_format': r"\normalsize{$%3.1f$}",

    'wd_tag': 'A',
    'wd_tick_side': 'right',
        'wd_title':r'\Large $\frac{V_{out}}{V_{in}}$',
    'wd_tick_levels': 5,
    'wd_tick_text_levels': 2,
    'wd_title_opposite_tick': True,
    'wd_axis_color': pyx.color.cmyk.Gray,
    'isopleth_values': [
        [6.2, 'x', 'x'],
    ],
    'vertical_guide_nr': 10,
        'manual_x_scale': True,                # trick to "decompress" Ra scale

}

# this is non-obvious trick to find bottom edge coordinates of the grid in order
# to align it with N nomogram
block1_dummy = Nomo_Block_Type_5(mirror_x=False)
block1_dummy.define_block(block_1_params)
block1_dummy.set_block()

# Let's define the N-nomogram
N_params_3 = {
    'u_min': block1_dummy.grid_box.params_wd['u_min'],
    'u_max': block1_dummy.grid_box.params_wd['u_max'],
    'function': lambda u: u,
    'title': '',
    'tag': 'A',
    'tick_side': 'right',
    'tick_levels': 2,
    'tick_text_levels': 2,
    'reference': False,
    'tick_levels': 0,
    'tick_text_levels': 0,
    'title_draw_center': True
}
```

(continues on next page)

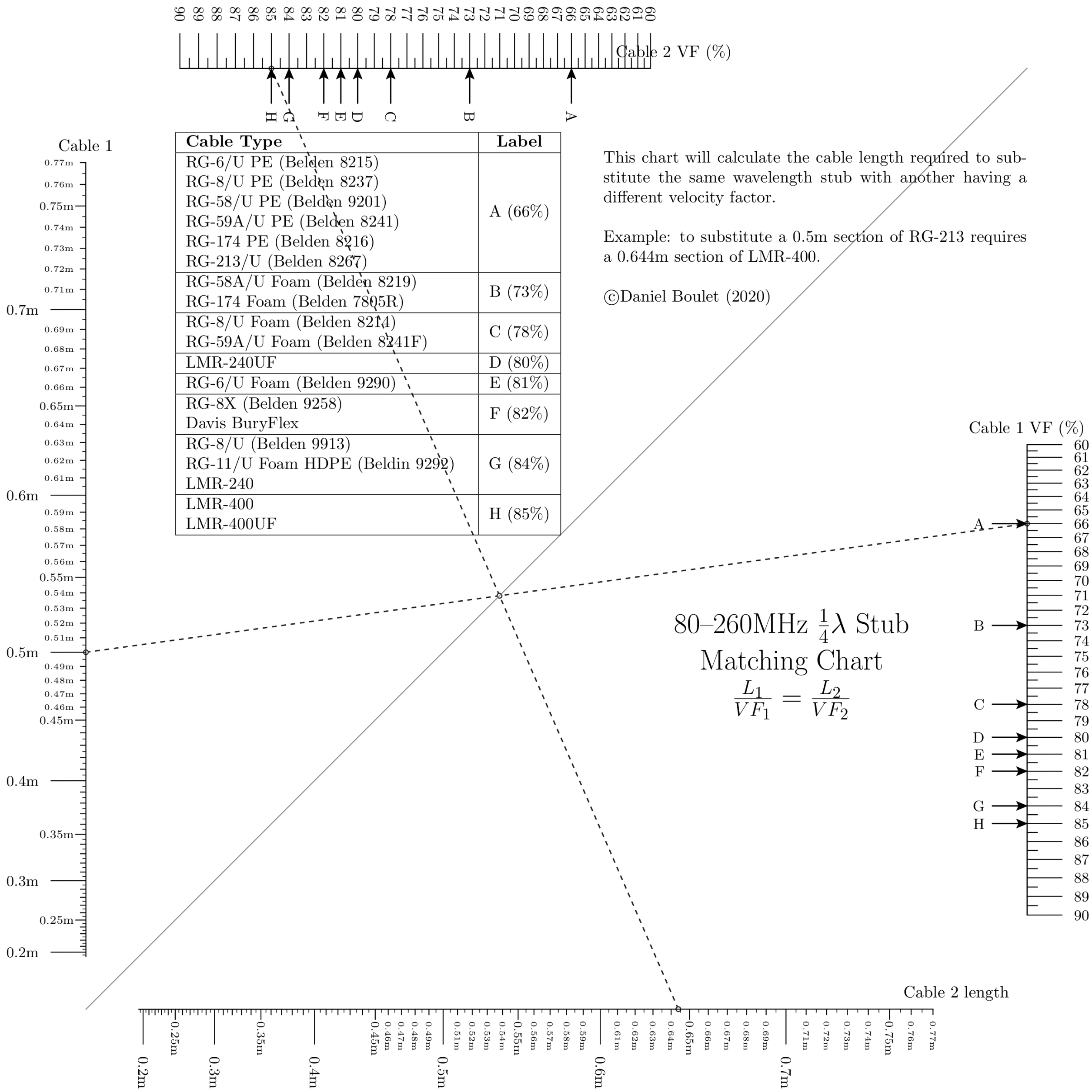
(continued from previous page)

```
N_params_2 = {
    'u_min': 6.0,
    'u_max': 24.0,
    'function': lambda u: u,
    'title': r'$V_{in}$',
    'tag': 'none',
    'tick_side': 'left',
    'tick_levels': 4,
    'tick_text_levels': 3,
    'title_draw_center': True,
    'scale_type': 'linear smart',
}
N_params_1 = {
    'u_min': 1.0,
    'u_max': 10.0,
    'function': lambda u: u,
    'title': r'$V_{out}$',
    'tag': 'none',
    'scale_type': 'linear smart',
    'tick_side': 'right',
    'tick_levels': 3,
    'tick_text_levels': 3,
    'title_draw_center': True
}

block_2_params = {
    'block_type': 'type_2',
    'f1_params': N_params_1,
    'f2_params': N_params_2,
    'f3_params': N_params_3,
    'isopleth_values': [
        # Vout, Vin, ratio
        [3.3, 9.0, 'x'],
    ]
}

main_params = {
        # 'make_grid': True,
    'filename': outputfile,
    'paper_height': 8.5*2.54,
    'paper_width': 11.0*2.54,
    'block_params': [block_1_params, block_2_params],
    'transformations': [('rotate', 0.01), ('scale paper',)],
    'title_str': r'\Large Voltage Divider Nomograph \par \
        \normalsize (For E24 series values) \par \bigskip \
        \large $V_{out}=V_{in} \cdot \frac{R_b}{R_a+R_b}$ \
        \par \bigskip   \normalsize \copyright    Daniel Boulet  2018-2021',
    'title_x': 2.0,
    'title_y': 4.0,
    'isopleth_params': [
        {
            'color': 'blue',
            'linewidth': 'thick',
            'linestyle': 'dashed',
            'circle_size': 0.10,
        },
    ],
}

Nomographer(main_params)
```

1.5 Velocity Factor Calculator

$\frac{1}{4}\lambda$ matching stubs can be created with different types of coax cable. However, the velocify factor of coax cables varies by type and manufacturer. This nomograph allows an engineer to substitute one type for another of different length and velocity factor.

1.5.1 Nomograph



Cable Type	Label
RG-6/U PE (Belden 8215) RG-8/U PE (Belden 8237) RG-58/U PE (Belden 9201) RG-59A/U PE (Belden 8241) RG-174 PE (Belden 8216) RG-213/U (Belden 8267)	A (66%)
RG-58A/U Foam (Belden 8219) RG-174 Foam (Belden 7805R)	B (73%)
RG-8/U Foam (Belden 8214) RG-59A/U Foam (Belden 8241F)	C (78%)
LMR-240UF	D (80%)
RG-6/U Foam (Belden 9290)	E (81%)
RG-8X (Belden 9258) Davis BuryFlex	F (82%)
RG-8/U (Belden 9913) RG-11/U Foam HDPE (Beldin 9292) LMR-240	G (84%)
LMR-400 LMR-400UF	H (85%)

1.5.2 Source code

```python
"""
    vf_calculator.py

    Stub matching nomogram for 80 to 260 MHz range.
"""

from pynomo.nomographer import Nomographer
import sys
from pyx import *
text.set(text.LatexEngine)
sys.path.insert(0, "..")
outputfile = sys.argv[0].split('.')[0]+'.pdf'

text.preamble(r"\usepackage{array}")

scalingFactor = 2
shortest_wavelength = 0.19250
longest_wavelength = 0.77001

stub_length_1 = {
    'u_min': shortest_wavelength,
    'u_max': longest_wavelength,
    'function': lambda u: u**scalingFactor,
    'title': r'Cable 1',
    'tick_levels': 5,
    'tick_text_levels': 4,
    'text_format': r'$%2.2g$m',
    'scale_type': 'linear smart',
    'tick_side': 'left',
}
vel_factor_1 = {
    'u_min': 60,
    'u_max': 90,
    'function': lambda u: (u/100)**scalingFactor,
    'title': r'Cable 1 VF (\%)',
    'tick_levels': 3,
    'text_format': r'$%2.2g$',
    'tick_text_levels': 1,
    'scale_type': 'linear smart',
    'tick_side': 'right',
    'extra_params': [
        {
            'scale_type': 'manual arrow',
            'arrow_length': 0.75,
            'manual_axis_data': {
                66: r'A',
                73: r'B',
                78: r'C',
                80: r'D',
                81: r'E',
                82: r'F',
                84: r'G',
                85: r'H',
            },
            'tick_side': 'left'
        }
    ]
}

stub_length_2 = {
    'u_min': shortest_wavelength,
    'u_max': longest_wavelength,
    'function': lambda u: u**scalingFactor,
    'title': r'Cable 2 length',
```

(continues on next page)

(continued from previous page)

```
    'tick_levels': 5,
    'tick_text_levels': 4,
    'text_format': r'$%2.2g$m',
    'scale_type': 'linear smart',
    'tick_side': 'right',
    'title_x_shift': 0.5,
    'title_rotate_text': True,
}
vel_factor_2 = {
    'u_min': 60,
    'u_max': 90,
    'function': lambda u: (u/100)**scalingFactor,
    'title': r'Cable 2 VF (\%)',
    'tick_levels': 5,
    'tick_text_levels': 4,
    'scale_type': 'linear smart',
    'tick_side': 'left',
    'title_rotate_text':True,
    'title_x_shift': 0.5,
    'extra_params': [
        {
            'scale_type': 'manual arrow',
            'arrow_length': 0.75,
            'manual_axis_data': {
                66: r'A',
                73: r'B',
                78: r'C',
                80: r'D',
                81: r'E',
                82: r'F',
                84: r'G',
                85: r'H',
            },
            'tick_side': 'right'
        }
    ]

}

block_1_params = {
    'block_type': 'type_4',
    'f1_params': stub_length_1,
    'f2_params': vel_factor_1,
    'f3_params': stub_length_2,
    'f4_params': vel_factor_2,
    'isopleth_values': [[0.5, 66, 'x', 85]],
    'reference_color': color.cmyk.Gray
}

main_params = {
    'filename': outputfile,
    'paper_height': 20.0,
    'paper_width': 20.0,
    'block_params': [block_1_params],
    'transformations': [('rotate', 0.01), ('scale paper',)],
    'title_box_width':7.0,
    'title_str': r'\LARGE 80--260MHz $\frac{1}{4}\lambda$ Stub Matching Chart \par $\frac{L_1}{VF_1}=\frac{L_2}{VF_2}
↪$',
    'title_x':15.0,
    'title_y': 8.0,
    # 'make_grid': True,
    'extra_texts': [
        {
            'x': 1,
            'y': 19,
```

(continues on next page)

(continued from previous page)

```
            'text': r'\begin{center} \begin{tabular}{| m{6cm} | c | } \hline \bf{Cable Type} & \bf{Label} \\ \hline RG-6/U PE (Belden 8215) \newline RG-8/U PE (Belden 8237) \newline RG-58/U PE (Belden 9201) \newline RG-59A/U PE (Belden 8241) \newline RG-174 PE (Belden 8216) \newline RG-213/U (Belden 8267) & A (66\%)  \\ \hline RG-58A/U Foam (Belden 8219) \newline RG-174 Foam (Belden 7805R) & B (73\%) \\ \hline RG-8/U Foam (Belden 8214) \newline RG-59A/U Foam (Belden 8241F) & C (78\%) \\ \hline LMR-240UF & D (80\%) \\ \hline RG-6/U Foam (Belden 9290) & E  (81\%) \\ \hline RG-8X (Belden 9258) \newline Davis BuryFlex & F (82\%)  \\ \hline RG-8/U (Belden 9913) \newline RG-11/U Foam HDPE (Beldin 9292) \newline LMR-240 & G (84\%) \\ \hline LMR-400 \newline LMR-400UF & H (85\%) \\ \hline \end{tabular} \end{center}',
            'width': 10.0,
        },
        {
            'x': 11.0,
            'y': 18,
            'text': r'\noindent This chart will calculate the cable length required to substitute the same wavelength stub with another having a different velocity factor. \newline \newline Example: to substitute a 0.5m section of RG-213 requires a 0.644m section of LMR-400. \newline \newline \copyright Daniel Boulet (2020)',
            'width': 9.0,
        },
    ],


}
Nomographer(main_params)
```

1.6 True VSWR Calculator

1.6.1 Introduction

Amateur radio operators frequently design and build their own antennae. An important performance characteristic of any antenna system is the **Voltage Standing Wave Ratio** (VSWR). The VSWR is a measure of the impedance match between the signal source (transmitter) and the load (antenna). Maximum power transfer from transmitter to antenna occurs when the source's output impedance matches the load's input impedance. A VSWR of 1:1 or simply "1" indicates a perfect match.

1.6.2 VSWR, Return Loss and Cable Attentuation

The VSWR is calculated from the forward and reflected power as follows [Wik21a]:

$$VSWR = \frac{1+\sqrt{P_r/P_f}}{1-\sqrt{P_r/P_f}}.$$

Return loss (RL) is also an indicator of antenna performance. It is the ratio of forward power to reflected power expressed in decibels (dB) [EN21]:

$$RL = 10\log_{10}\frac{P_\mathrm{f}}{P_\mathrm{r}}$$

where P_f and P_r are respectively forward and reflected power in Watts. By converting forward and reflected power readings to a decibel value referenced to 1W (0 dBW) the return loss can be expressed as:

$$RL = P_f - P_r \text{ (dB)}$$

Return loss and VSWR are related [EN21] by the following formulae:

$RL = -20 \times \log_{10} \frac{VSWR-1}{VSWR+1}$

and

$VSWR = \frac{10^{\frac{RL}{20}}+1}{10^{\frac{RL}{20}}-1}$.

A high return loss (>30dB) indicates a high ratio of forward power to reflected power therefore a good impedance match. A return loss approaching 0dB means the reflected power is nearly equal to the forward power indicating a mismatch.

1.6.3 Cable loss and impact on VSWR

VSWR is a useful indicator of power transfer but measured results can be misleading. The transmission line connecting the transmitter to the antenna will introduce some loss therefore the RF power reaching the antenna will be lower than the RF power delivered by the transmitter. Similarly, the power reflected by the antenna will also have been attenuated by the transmission line before it returns to the power meter. The combined effect of attenuated forward and reflected power will make it appear as though the VSWR is lower than it would be if measured at the antenna. It isn't practical to install a power meter at the antenna but the true VSWR can be calculated based on measured VSWR and known cable losses.

Cable attentuation is specified in dB per unit length. Cable attentuation for LMR®-195 is based on the following formula [Sys]:

$Loss = (0.356859) \times \sqrt{f} + (0.000470) \times f$

where $Loss$ is the attentuation per 100 feet and f is the operating frequency in megahertz. Given forward and reflected power readings in dB and cable length, we can easily calculate the true return loss at the antenna:

$RL_{true} = RL_{measured} - 2 * Loss$.

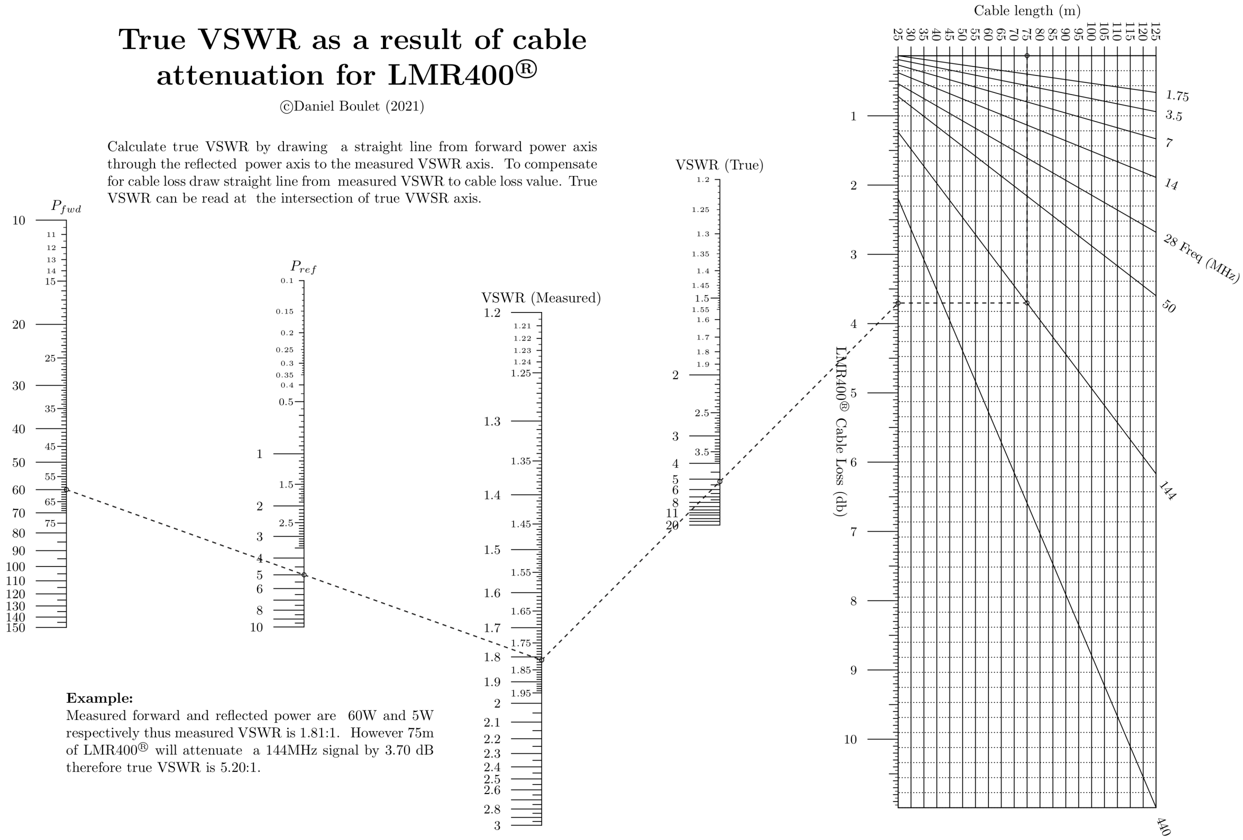
A return loss figure can then be converted to VSWR using the formula described earlier.

1.6.4 Nomograph construction

Both formulae for return loss can be easily put into a form acceptable for Type 1 block. A graph is created using a Type 5 nomograph to calculate cable losses given a length and operating frequency.

Since radio amateurs usually work in terms of power expressed in Watts and VSWR, Type 8 blocks are anchored to the Type 1 blocks to convert dbW and RL into Watts and VSWR respectively. The dB axes are hidden to minimize clutter.

1.6.5 Generated nomograph



1.6.6 Source code

```
"""
    full_vswr.py

    Nomogram to calculate VSWR at antenna given measured VSWR and feedline antenuation.
"""
import sys
import numpy as np
from pyx import *

outputfile = sys.argv[0].split(".")[0] + ".pdf"
sys.path.insert(0, "..")
text.set(text.LatexEngine)

from pynomo.nomographer import Nomographer

minimum_return_loss = 6.0
maximum_return_loss = 40.0


# various conversion functions
def vswr2rl(vswr):
    # vswr to return loss in db
    return -20 * np.log10((vswr - 1) / (vswr + 1))


```

(continues on next page)

(continued from previous page)

```
def rl2vswr(rl):
    # return loss to vswr
    return (1 + 10 ** (-rl / 20)) / (1 - 10 ** (-rl / 20))


def watts2dbw(watts):
    # watts to dbW
    return 10 * np.log10(watts)


def dbw2watts(dbw):
    # dbW to watts
    return 10 ** (dbw / 10.0)


def cableloss(freq):
    # cable loss in db per meter at freq
    # source formula is db per 100 feet
    return (np.sqrt(freq) * 0.122290 + freq * 0.000260) / 100.0 / 0.3048


# measured return loss axes and block section
axis1_forward_power_meas_watts = {
    "tag": "axis14",
    "u_min": 10.0,
    "u_max": 150.0,
    "title": r"$P_{fwd}$",
    "tick_levels": 5,
    "tick_text_levels": 4,
    "function": lambda u: watts2dbw(u),
    "align_func": lambda u: watts2dbw(u),
    "scale_type": "linear smart",
    "tick_side": "left",
}

block_forward_watts = {
    "block_type": "type_8",
    "f_params": axis1_forward_power_meas_watts,
    "isopleth_values": [["x"]],
}

axis2_reflected_power_meas_watts = {
    "tag": "axis25",
    "title": r"$P_{ref}$",
    "u_min": 0.1,
    "u_max": 10.0,
    "function": lambda u: watts2dbw(u),
    "align_func": lambda u: watts2dbw(u),
    "tick_levels": 5,
    "tick_text_levels": 4,
    "tick_side": "left",
    "scale_type": "linear smart",
}

block_reflected_watts = {
    "block_type": "type_8",
    "f_params": axis2_reflected_power_meas_watts,
    "isopleth_values": [["x"]],
}

axis3_return_loss_meas_vswr = {
    "tag": "axis36",
    "u_min": 1.2,
    "u_max": 3.0,
    "title": r"VSWR (Measured)",
```

(continues on next page)

(continued from previous page)

```
    "function": lambda u: vswr2rl(u),
    "align_func": lambda u: vswr2rl(u),
    "tick_levels": 5,
    "tick_text_levels": 4,
    "tick_side": "left",
    "scale_type": "linear smart",
}

block_return_loss_meas_vswr = {
    "block_type": "type_8",
    "f_params": axis3_return_loss_meas_vswr,
    "isopleth_values": [["x"]],
}

axis4_forward_power_meas_dbw = {
    "tag": "axis14",
    "u_min": watts2dbw(10.0),
    "u_max": watts2dbw(100.0),
    "function": lambda u: -u,
    "title_draw_center": True,
    "tick_side": "left",
    "tick_levels": 5,
    "title_distance_center": 1.7,
    "tick_text_levels": 4,
    "scale_type": "manual",
}

axis5_reflected_power_meas_dbw = {
    "tag": "axis25",
    "u_min": watts2dbw(0.1),
    "u_max": watts2dbw(7.0),
    "function": lambda u: u,
    "title_distance_center": -1.7,
    "title_draw_center": True,
    "tick_levels": 5,
    "tick_text_levels": 4,
    "scale_type": "manual",
}

axis6_return_loss_meas_dbw1 = {
    "tag": "axis36",
    "u_min": 7.0,
    "u_max": 30.0,
    "function": lambda u: u,
    "title_distance_center": 1.5,
    "title_draw_center": True,
    "tick_levels": 5,
    "tick_text_levels": 4,
    "scale_type": "manual",
}


block_measured_rl_dbw = {
    "block_type": "type_1",
    "f1_params": axis4_forward_power_meas_dbw,
    "f2_params": axis5_reflected_power_meas_dbw,
    "f3_params": axis6_return_loss_meas_dbw1,
    "height": 20.0,
    "width": 20.0,
    "isopleth_values": [[watts2dbw(60), watts2dbw(5), "x"]],
}


axis8_return_loss_true_dbw = {
    "tag": "axis8-10",
```

(continues on next page)

(continued from previous page)

```
    "u_min": 1.0,
    "u_max": 30.0,
    "function": lambda u: -u,
    "title_draw_center": True,
    "tick_levels": 5,
    "title_distance_center": 1.5,
    "tick_text_levels": 4,
    "scale_type": "manual",
}

axis10_return_loss_true_vswr = {
    "tag": "axis8-10",
    "u_min": 1.2,
    "u_max": 20.0,
    "title": r"VSWR (True)",
    "function": lambda u: vswr2rl(u),
    "align_func": lambda u: vswr2rl(u),
    "tick_side": "left",
    "tick_levels": 5,
    "tick_text_levels": 4,
    "scale_type": "linear smart",
}

block_return_loss_true_vswr = {
    "block_type": "type_8",
    "f_params": axis10_return_loss_true_vswr,
    "isopleth_values": [["x"]],
}


axis9_cable_loss = {
    "tag": "axis9",
    "u_min": 0.0,
    "u_max": 10.0,
    "function": lambda u: -2.0 * u,
    "title_distance_center": -1.5,
    "title_draw_center": True,
    "tick_levels": 5,
    "tick_text_levels": 4,
    "scale_type": "manual",
}

block_true_rl_dbw = {
    "block_type": "type_1",
    "f1_params": axis6_return_loss_meas_dbw1,
    "f2_params": axis8_return_loss_true_dbw,
    "f3_params": axis9_cable_loss,
    "isopleth_values": [["x", "x", "x"]],
}

# cable loss block using type 5
block_cable_loss = {
    "block_type": "type_5",
    "u_func": lambda u: u,
    "u_values": list(np.linspace(25.0, 125.0, 21)),
    "u_scale_type": "manual point",
    "u_title_distance_center": 1.0,
    "u_title": "Cable length (m)",
    "v_func": lambda x, v: x / cableloss(v),
    "v_values": [1.75, 3.5, 7.0, 14.0, 28.0, 50.0, 144.0, 440.0],
    "v_title": "Freq (MHz)",
    "wd_tick_levels": 4,
    "wd_tick_text_levels": 1,
    "wd_tick_side": "left",
    "wd_title": r"LMR400\textsuperscript{\textregistered} Cable Loss (db)",
```

(continues on next page)

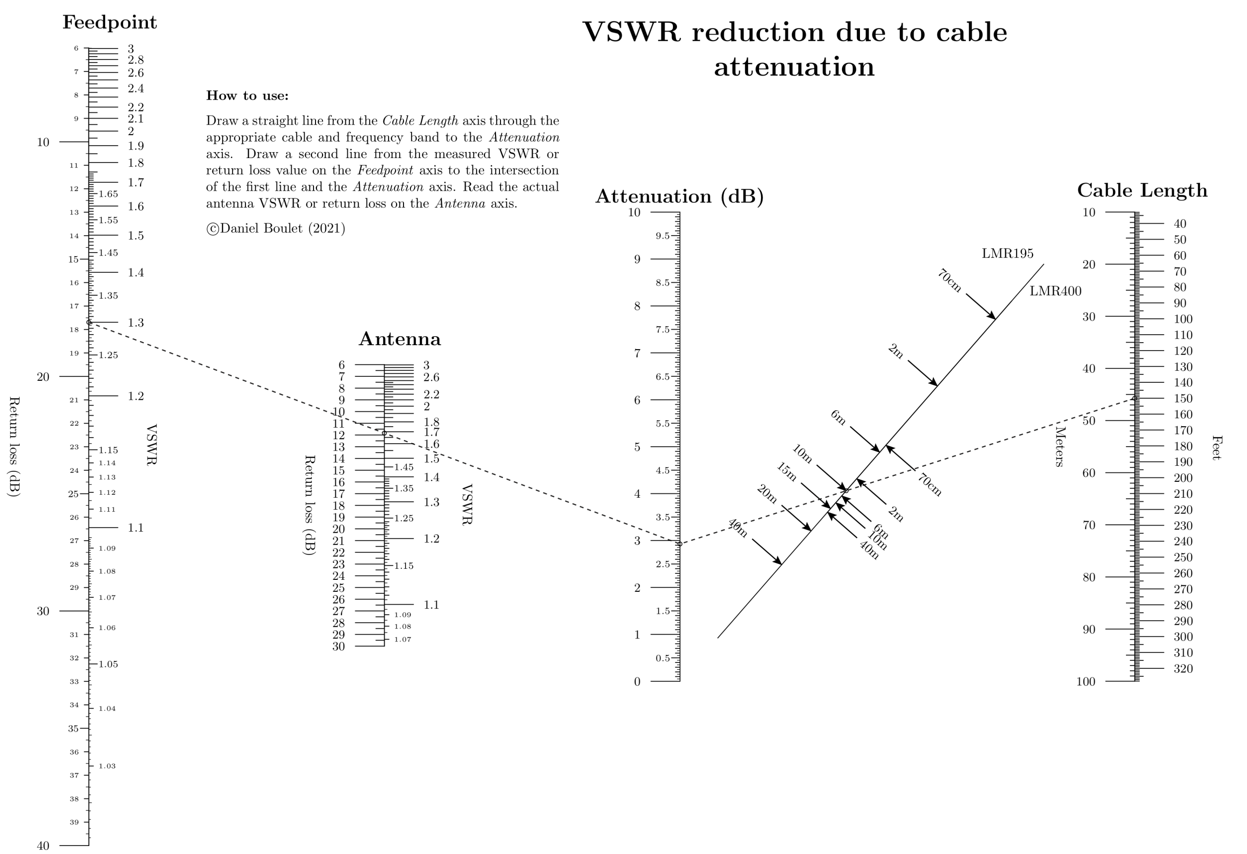
(continued from previous page)

```
    "wd_title_opposite_tick": True,
    "wd_tag": "axis9",
    "isopleth_values": [[75, 144, "x"]],
}

main_params = {
    "filename": outputfile,
    "paper_height": 8.0 * 2.54,
    "paper_width": 10.5 * 2.54,
    "block_params": [
        block_measured_rl_dbw,
        block_true_rl_dbw,
        block_reflected_watts,
        block_forward_watts,
        block_return_loss_meas_vswr,
        block_return_loss_true_vswr,
        block_cable_loss,
    ],
    "transformations": [("rotate", 0.01), ("scale paper",)],
    "title_str": r"\huge \textbf{True VSWR as a result of cable attenuation for LMR400\textsuperscript{\textregistered}} \
                \par\medskip \normalsize \copyright Daniel Boulet (2021)",
    "title_x": 7.0,
    "title_y": 19.0,
    "title_box_width": 12.0,
    "extra_texts": [
        {
            "x": 1.0,
            "y": 16.5,
            "text": r"\noindent Calculate true VSWR by drawing \
                    a straight line from forward power axis through the reflected \
                    power axis to the measured VSWR axis. \
                    To compensate for cable loss draw straight line from \
                    measured VSWR to cable loss value.  True VSWR can be read at \
                    the intersection of true VWSR axis.",
            "width": 12.0,
        },
        {
            "x": 0.0,
            "y": 3.0,
            "text": r"\noindent \textbf{Example:} \
                    \par \noindent Measured forward and reflected power are \
                    60W and 5W respectively thus measured VSWR is 1.81:1. \
                    However 75m of LMR400\textsuperscript{\textregistered} will attenuate \
                    a 144MHz signal by 3.70 dB therefore true VSWR is 5.20:1.",
            "width": 9.0,
        },
    ],
    # "make_grid": True,
}
Nomographer(main_params)
```

1.7 Return Loss Calculator

The Return Loss Calculator performs the same calculations as the True VSWR calculator but presents the results differently. The Return Loss Calculator allows for direct conversion of VSWR to Return Loss and vice versa. Two cable types are alos offered on the right-hand portion of the nomograph. The frequency is choosen by amateur band. Missing from this nomograph is the ability to calcuate VSWR from forward and reflected power readings.

1.7.1 Nomograph



1.7.2 Source code

```
"""
    return_loss.py

    Nomogram to calculate VSWR at antenna given measured VSWR and feedline antenuation.
"""
import sys
import numpy as np
from pyx import *

outputfile = sys.argv[0].split('.')[0]+'.pdf'
sys.path.insert(0, "..")
text.set(text.LatexEngine)
```

(continues on next page)

(continued from previous page)

```python

from pynomo.nomographer import Nomographer

minimum_return_loss = 6.0
maximum_return_loss = 40.0


# functions to convert return loss to VSWR and vice-versa


def vswr2rl(vswr):
    return -20 * np.log10((vswr - 1) / (vswr + 1))


def rl2vswr(rl):
    return (1 + 10 ** (-rl / 20)) / (1 - 10 ** (-rl / 20))


# main block items

measured_rl = {
    "tag": "measured",
    "u_min": minimum_return_loss,
    "u_max": maximum_return_loss,
    "function": lambda u: u,
    "scale_type": "linear smart",
    "title": r"Return loss (dB)",
    "title_relative_offset": (0, 1.5),
    "title_draw_center": True,
    "tick_side": "left",
    "tick_levels": 4,
    "tick_text_levels": 3,
    "extra_titles": [
        {
            "dx": -1.2,
            "dy": 0.5,
            "text": r"\Large \textbf{Feedpoint}",
            # 'width':5.0,
        }
    ],
}


antenna_rl = {
    "tag": "antenna",
    "u_min": 6,
    "u_max": 30,
    "scale_type": "linear smart",
    "function": lambda u: -u,
    "title": r"Return loss (dB)",
    "title_draw_center": True,
    "extra_titles": [
        {
            "dx": -1.2,
            "dy": 0.5,
            "text": r"\Large \textbf{Antenna}",
            # 'width':5.0,
        }
    ],
    "title_relative_offset": (0, 1.5),
    "tick_levels": 2,
    "tick_text_levels": 1,
    "tick_side": "left",
}

```

(continues on next page)

(continued from previous page)

```
cable_loss = {
    "tag": "cable_loss",
    "u_min": 0,
    "u_max": 10.0,
    "scale_type": "linear smart",
    "function": lambda u: -2 * u,
    "title": r"\Large \textbf{Attenuation (dB)}",
    "title_relative_offset": (0, -2.5),
    # "title_draw_center": True,
    "tick_side": "left",
    "tick_levels": 4,
    "tick_text_levels": 3,
    # "extra_params": [
    #     {
    #         "scale_type": "manual arrow",
    #                 'tick_side':'right',
    #         "manual_axis_data": {
    #             8.5: "100ft LMR195 70cm band",
    #         },
    #     }
    # ],
}

main_block = {
    "block_type": "type_1",
    "f1_params": measured_rl,
    "f3_params": cable_loss,
    "f2_params": antenna_rl,
    "isopleth_values": [["x", "x", 'x']],
    "mirror_y": True,
}

# measured vswr items
measured_vswr = {
    "tag": "measured",
    "u_min": rl2vswr(maximum_return_loss),
    "u_max": rl2vswr(minimum_return_loss),
    "function": lambda u: vswr2rl(u),
    "align_func": lambda u: vswr2rl(u),
    "title": r"VSWR",
    "title_draw_center": True,
    "title_relative_offset": (0, 1),
    "tick_levels": 6,
    "tick_text_levels": 3,
    "scale_type": "linear smart",
    "tick_side": "right",
}

measured_vswr_block = {
    "block_type": "type_8",
    "f_params": measured_vswr,
    "isopleth_values": [[1.3]],
}

# antenna vswr items

antenna_vswr = {
    "tag": "antenna",
    "u_min": rl2vswr(30),
    "u_max": rl2vswr(6),
    "function": lambda u: vswr2rl(u),
    "align_func": lambda u: vswr2rl(u),
    "title": r"VSWR",
    "title_relative_offset": (0, 1.5),
    "title_draw_center": True,
```

(continues on next page)

(continued from previous page)

```
    "tick_levels": 6,
    "tick_text_levels": 3,
    "scale_type": "linear smart",
    "tick_side": "right",
}

antenna_vswr_block = {
    "block_type": "type_8",
    "f_params": antenna_vswr,
    "isopleth_values": [["x"]],
}

# cable loss items (type 2 block)
cable_loss_arrows = {
    "tag": "cable_loss",
    "u_min": 0.0,
    "u_max": 10.0,
    "function": lambda u: u,
    # "title": r"cable loss",
    "tick_levels": 3,
    "tick_text_levels": 1,
    "scale_type": "manual",
}

loss_per_unit_length_lmr195 = {
    "tag": "cable_type",
    "u_min": 1.0,
    "u_max": 30.0,
    "function": lambda u: u,
    "align_function": lambda u: u,
    "title": r"LMR195",
    "title_rotate_text": True,
    "title_x_shift": -0.1,
    "title_y_shift": 1.1,
    # "title_draw_center": True,
    "tick_levels": 3,
    "tick_text_levels": 2,
    "tick_side": "left",
    "scale_type": "manual arrow",
    "manual_axis_data": {
        3.2: r"40m",
        4.5: r"20m",
        5.5: r"15m",
        6.4: r"10m",
        8.7: r"6m",
        14.5: r"2m",
        25.2: r"70cm",
    },
    # "title_relative_offset": (0, 0),
}

loss_per_unit_length_lmr400 = {
    "tag": "cable_type",
    "u_min": 1.0,
    "u_max": 30.0,
    "function": lambda u: u,
    "align_function": lambda u: u,
    "title": r"LMR400",
    "title_rotate_text": True,
    "title_x_shift": 0.3,
    "title_y_shift": -0.8,
    # "title_draw_center": True,
    "tick_levels": 3,
    "tick_text_levels": 2,
    "tick_side": "right",
```

(continues on next page)

(continued from previous page)

```
    "scale_type": "manual arrow",
    "manual_axis_data": {
        1.1: r"40m",
        # 1.5: r"20m",
        # 1.9: r"15m",
        2.2: r"10m",
        3.0: r"6m",
        5.0: r"2m",
        8.9: r"70cm",
    },
    # "title_relative_offset": (0, -2.5),
}

length_meters = {
    "tag": "clength",
    "u_min": 10.0,
    "u_max": 100.0,
    "function": lambda u: u / 100.0,
    "title": r"Meters",
    "tick_levels": 3,
    "tick_text_levels": 1,
    "title_draw_center": True,
    "tick_side": "left",
    "title_relative_offset": (0, 1.5),
    "extra_titles": [
        {
            "dx": -2.0,
            "dy": 0.4,
            "text": r"\Large \textbf{Cable Length}",
        }
    ],
}

length_feet = {
    "tag": "clength",
    "u_min": 10.0 / 0.3048,
    "u_max": 100.0 / 0.3048,
    "function": lambda u: u,
    "align_func": lambda u: u * 0.3048,
    "title": r"Feet",
    "title_draw_center": True,
    "tick_levels": 3,
    "tick_text_levels": 1,
    "title_relative_offset": (0, -2.5),
}
cable_parameters = {
    "block_type": "type_2",
    # "width": 10.0,
    # "height": 10.0,
    "f1_params": cable_loss_arrows,
    "f2_params": loss_per_unit_length_lmr195,
    "f3_params": length_meters,
    'isopleth_values': [['x', 6.4, 'x']],
}

cable_parameters2 = {
    "block_type": "type_8",
    "f_params": loss_per_unit_length_lmr400,
        'isopleth_values':[['x']],
}

cable_parameters3 = {
    "block_type": "type_8",
    "f_params": length_feet,
        'isopleth_values':[[150]],
```

(continues on next page)

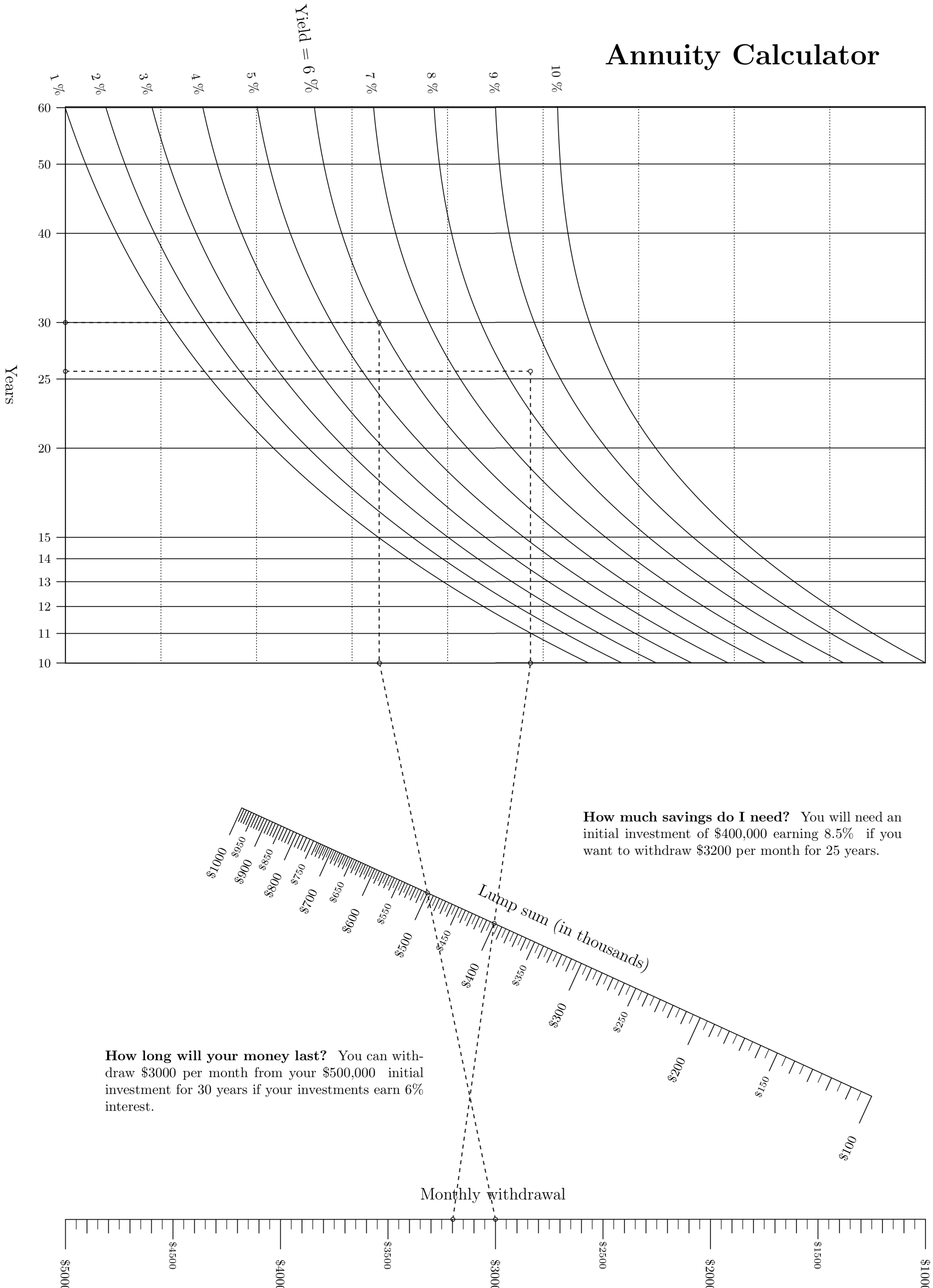
(continued from previous page)

```
}


main_params = {
    "filename": outputfile,
    "paper_height": 8.0 * 2.54,
    "paper_width": 10.5 * 2.54,
    # "block_params": [main_block],
    "block_params": [
        main_block,
        measured_vswr_block,
        antenna_vswr_block,
        cable_parameters,
        cable_parameters2,
        cable_parameters3,
    ],
    # "block_params": [main_block,measured_vswr_block],
    "transformations": [("rotate", 0.01), ("scale paper",)],
    "title_str": r"\huge \textbf{VSWR reduction due to cable attenuation}",
    "title_x": 18.0,
    "title_y": 20.5,
    "extra_texts": [
        {
            "x": 3.0,
            "y": 19.0,
            "text":r"\noindent \textbf{How to use:} \
                                \par \medskip \noindent Draw a straight line from the \textit{Cable Length} axis through the appropriate cable and frequency band  to the \textit{Attenuation} axis. Draw a second line from the measured VSWR or return loss value on the \textit{Feedpoint} axis to the intersection of the first line and the \textit{Attenuation} axis.  Read the actual antenna VSWR or return loss on the \textit{Antenna} axis. \
                                \par \medskip \noindent  \copyright Daniel Boulet (2021)",
            "width": 9.0,
        },
    ],
    "debug": False,
    # "make_grid": True,
}

Nomographer(main_params)
```

1.8 Simple Annuity

The following nomograph is related to the amortized loan calculator example in the PyNomo documentation. However, rather than calculate outflow of cash of a loan, the nomograph below calculates inflow of cash of a fixed annuity.

1.8.1 Nomograph



1.8.2 Source code

```
"""
    simple_annuity.py

    ---
    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.
"""

from pynomo.nomographer import *

import sys
import numpy as np
outputfile = sys.argv[0].split('.')[0]+'.pdf'

# allows use of latex commands in PyX such as \frac{a}{b} and \par
from pyx import *
pyx.text.set(text.LatexEngine)


def annuity(x, u):
    return np.log(
        np.log(-x/(u/1200.0-x))
        /
        np.log(u/1200.0+1)
    )


block_1_params = {
    'width': 10.0,
    'height': 5.0,
    'block_type': 'type_5',
    'u_func': lambda u: np.log(u*12.0),
    'v_func': annuity,
    'u_values': [10.0, 11.0, 12.0, 13.0, 14.0, 15.0, 20.0, 25.0, 30.0, 40.0, 50.0, 60.0],
    'v_values': [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0],
    'wd_tag': 'A',
    'u_title': r'\large Years',
    'v_title': r'\large Yield = ',
    'u_text_format': r"$%3.0f$ ",
    'v_text_format': r"$%3.0f$ \%% ",
    'vertical_guide_nr': 10,
    'horizontal_guides': False,
    'v_title_draw_center': True,
    'isopleth_values': [['x', 6, 'x'],['x',8.5,'x']], #['years','yield','ratio']
}

# this is non-obvious trick to find bottom edge coordinates of the grid in order
# to align it with N nomogram
block1_dummy = Nomo_Block_Type_5(mirror_x=False)
block1_dummy.define_block(block_1_params)
block1_dummy.set_block()

# Let's define the N-nomogram
N_params_3 = {
    'u_min': block1_dummy.grid_box.params_wd['u_min'],
    'u_max': block1_dummy.grid_box.params_wd['u_max'],
    'function': lambda u: u,
    'title': '',
    'tag': 'A',
    'tick_side': 'right',
    'tick_levels': 2,
    'tick_text_levels': 2,
    'reference': False,
    'tick_levels': 0,
```

(continues on next page)

(continued from previous page)

```
    'tick_text_levels': 0,
    # 'title_draw_center': True
}
N_params_2 = {
    'u_min': 100.0,
    'u_max': 1000.0,
    'function': lambda u: u,
    'title': r'\large Lump sum (in thousands)',
    'tag': 'none',
    'tick_side': 'left',
    'tick_levels': 4,
    'tick_text_levels': 2,
    'title_draw_center': True,
    'grid_length_1': 0.5,
    'grid_length_2': 0.35,
    'grid_length_3': 0.25,
    'text_distance_1': 0.55,
    'text_format': r"$\$%3.0f$ ",
    'scale_type': 'linear smart',
}
N_params_1 = {
    'u_min': 1000.0,
    'u_max': 5000.0,
    'function': lambda u: u/1000.0,
    'title': r'\large Monthly withdrawal',
    'tag': 'none',
    'scale_type': 'linear smart',
    'tick_side': 'right',
    'tick_levels': 4,
    'tick_text_levels': 2,
    'title_draw_center': True,
    'grid_length_1': 0.5,
    'grid_length_2': 0.35,
    'grid_length_3': 0.25,
    'text_distance_1': 0.55,
    'text_format': r"$\$%3.0f$ ",

}

block_2_params = {
    'block_type': 'type_2',
    'width': 10.0,
    'height': 20.0,
    'f1_params': N_params_1,    # withdrawal
    'f2_params': N_params_2,    # lump sum (in thousands)
    'f3_params': N_params_3,    # curve
    'isopleth_values': [[3000, 500, 'x'], [3198.25, 400, 'x']]
}

main_params = {
    'filename': outputfile,
    'paper_height': 2.54*11,
    'paper_width': 2.54*8.5,
    # 'make_grid': True,
    'block_params': [block_1_params, block_2_params],
    'transformations': [('rotate', 0.01), ('scale paper',)],
    'title_str': r'\textbf{\huge Annuity Calculator}',
    'title_x': 17,
    'title_y': 29,
    'title_box_width': 8,
    'extra_texts': [
        {
            'text': r'\copyright Daniel Boulet (2018-2020)',
            'x': 16.5,
            'y': -2.5,
```

(continues on next page)

(continued from previous page)

```
        },
        {
            'text': r'\noindent \textbf{How long will your money last?} \
                You can withdraw \$3000 per month from your \$500,000 \
                    initial investment for 30 years if your investments earn 6\% interest.',
            'x': 1,
            'y': 4,
            'width': 8,
        },
        {
            'text': r'\noindent \textbf{How much savings do I need?} \
                You will need an initial investment of \$400,000 earning 8.5\% \
                    if you want to withdraw \$3200 per month for 25 years.',
            'x': 13,
            'y': 10,
            'width': 8,
        },
    ],

}
Nomographer(main_params)
```

1.9 Gasoline Price Compare

1.9.1 Background

Comparing the unit price of a commodity with different units of measure and currencies is greatly simplified with a nomograph. In this example, cross-border travellers between the United States and Canada can easily compare the cost of gasoline (petrol) on both sides of the border. Moreover, by drawing an isopleth through the currency rate at par (1.0000) they can directly convert dollars per litre to dollars per USG. This nomograph implements the following formula:

$\frac{CAD}{L} = \frac{CAD}{USD} \times \frac{USD}{USGal} \div \frac{L}{USGal}$

This equation follows the form of a **Type 2** nomograph where:

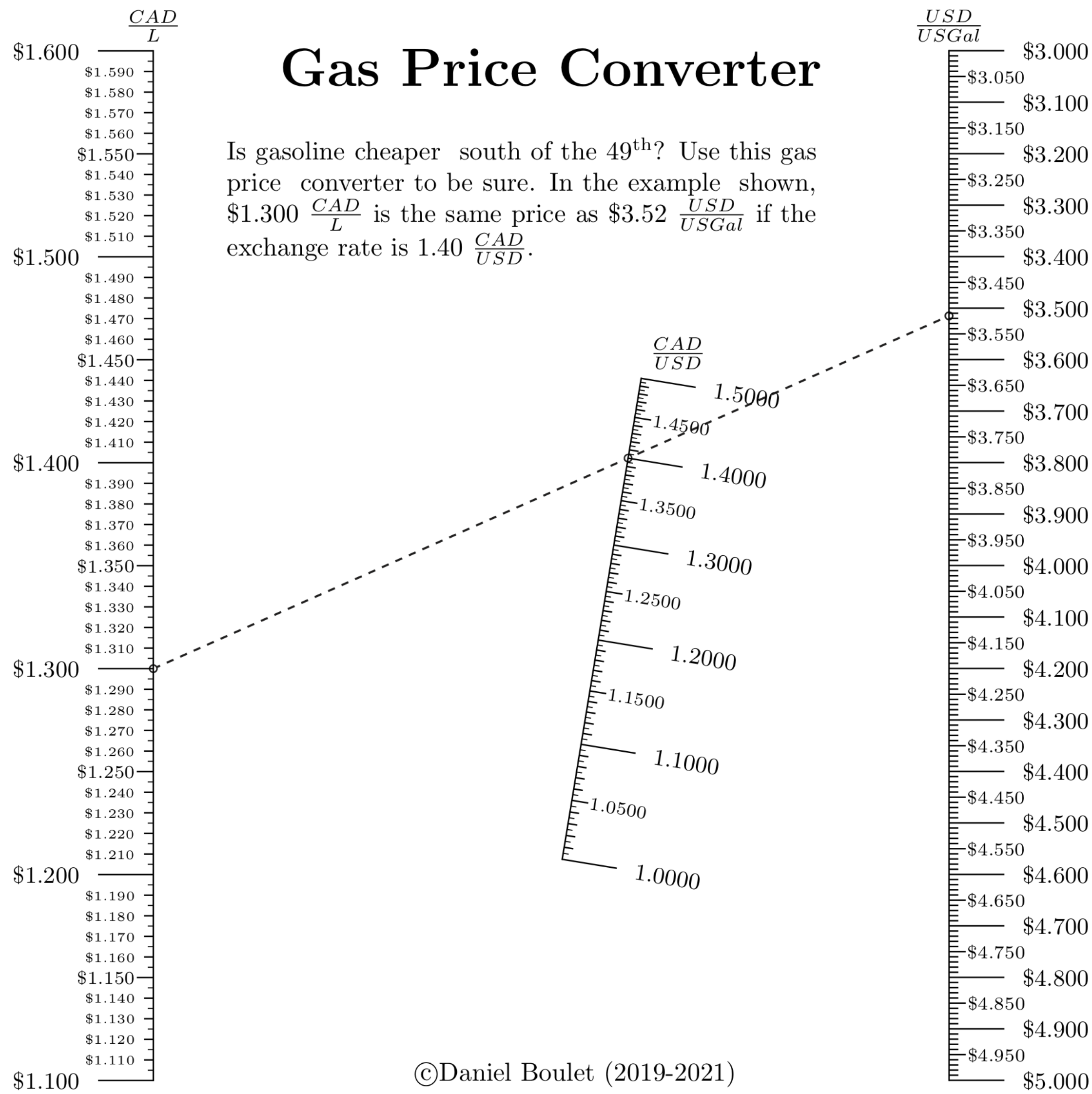
$F_1(u_1) = \frac{CAD}{L}$,

$F_2(u_2) = \frac{CAD}{USD}$

and

$F_3(u_3) = \frac{USD}{USGal} \div 3.78541 \frac{L}{USGal}$

1.9.2 Generated nomograph



1.9.3 Source code

```
"""
    gasolinepricesUSD_CAD.py

    Gasoline price converter
"""

from pynomo.nomographer import *
import sys
from pyx import *

sys.path.insert(0, "..")
outputfile = sys.argv[0].split('.')[0]+'.pdf'

# allows use of latex commands in PyX such as \frac{a}{b} and \par
```

(continues on next page)

(continued from previous page)

```
pyx.text.set(text.LatexEngine)

N_params_1 = {
    "u_min": 1.1,
    "u_max": 1.6,
    "function": lambda u: u,
    "title": r"$\frac{CAD}{L}$",
    "tick_levels": 4,
    "tick_text_levels": 3,
    "text_format": r"$\$%3.3f$",
    "scale_type": "linear smart",
    "tick_side": "left",
}

N_params_2 = {
    "u_min": 1.0,
    "u_max": 1.5,
    "function": lambda u: u,
    "title": r"$\frac{CAD}{USD}$",
    "tick_levels": 4,
    "tick_text_levels": 3,
    "text_format": r"$%3.4f$",
    "scale_type": "linear smart",
    "title_x_shift": 0.5,
    "title_rotate_text": True,
}

N_params_3 = {
    "u_min": 3.0,
    "u_max": 5.0,
    "function": lambda u: u / 3.78541,
    "title": r"$\frac{USD}{US Gal}$",
    "tick_levels": 4,
    "tick_text_levels": 2,
    "scale_type": "linear smart",
    "text_format": r"$\$%3.3f$",
    "scale_type": "linear smart",
}


block_1_params = {
    "block_type": "type_2",
    "f1_params": N_params_1,
    "f2_params": N_params_2,
    "f3_params": N_params_3,
    "isopleth_values": [[1.3, 1.4, "x"]],
}

main_params = {
    "filename": outputfile,
    "paper_height": 11.0 * 2.54 / 2.0,
    "paper_width": 8.5 * 2.54 / 2.0,
    "block_params": [block_1_params],
    "transformations": [("rotate", 0.01), ("scale paper",)],
    "title_str": r"\huge \textbf{Gas Price Converter}",
    "title_y": 13.50,
    "title_box_width": 15.0,
    "extra_texts": [
        {
            "x": 1.0,
            "y": 12.5,
            "text": r"\noindent Is gasoline cheaper \
                south of the 49\textsuperscript{th}? Use this gas price \
                converter to be sure. In the example \
                shown, \$1.300 $\frac{CAD}{L}$ is the same price as \$3.52 $\frac{USD}{US Gal}$ if the exchange rate␣
↪is 1.40  $\frac{CAD}{USD}$.",
```

(continues on next page)

(continued from previous page)

```
            "width": 8.0,
        },
        {
            "text": r"\copyright Daniel Boulet (2019-2021)",
            "x": 3.0,
            "y": -0.0,
        },
    ],
    # 'make_grid': True
}
Nomographer(main_params)
```

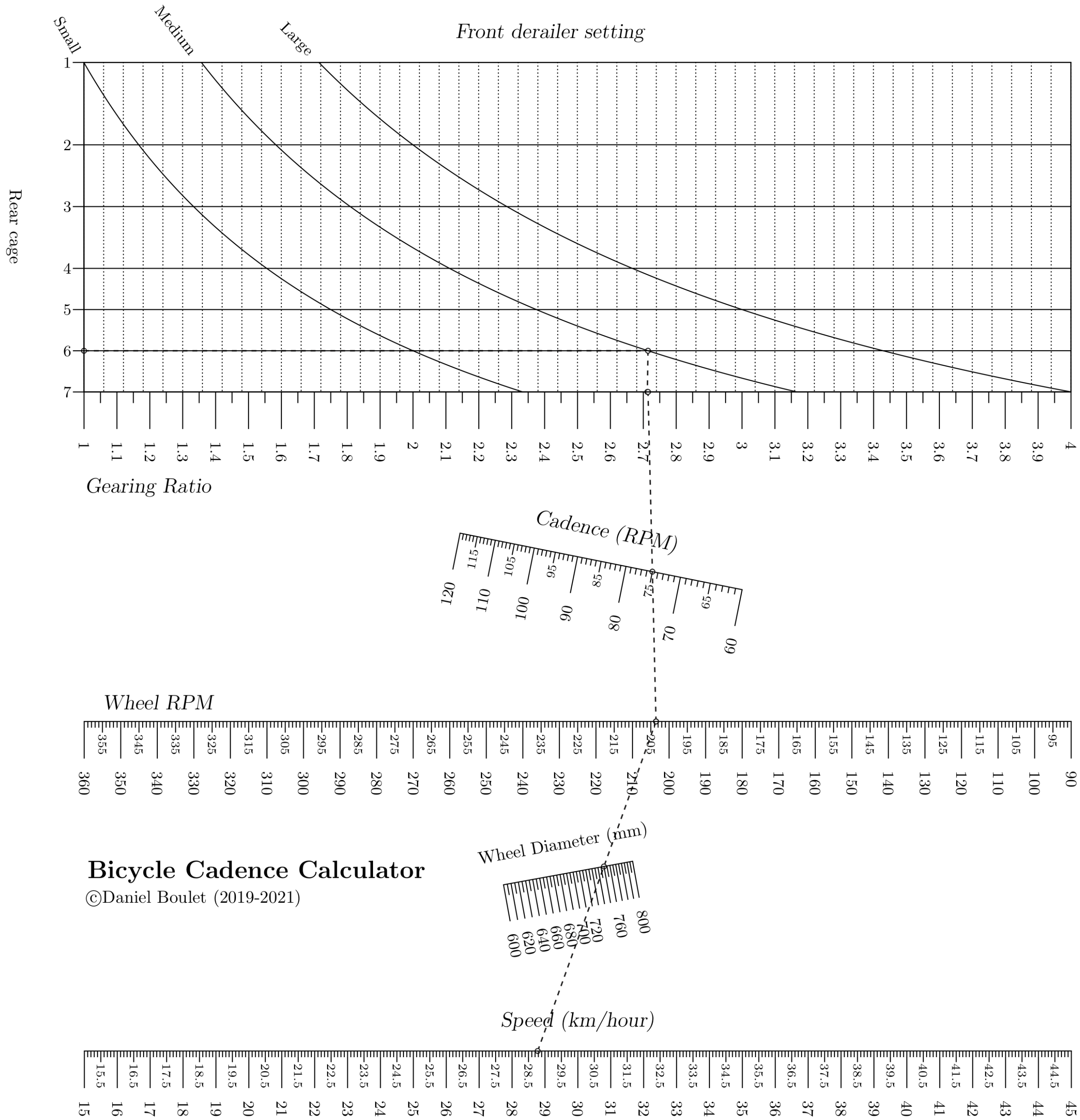
1.10 Bicycle Cadence

1.10.1 Theory and background

Choosing the correct gears on a bicycle allows a cyclist to maintain a comfortable cadence. A higher cadence helps reduce muscle fatigue [tra21] though it does put more stress on heart and lungs. However a lower cadence for the same power output puts more stress on the rider's knees, hips and back. [the21] Furthermore, cycling cadence will vary widely with beginning cyclists peddling more slowly (60-85 rpm) and professionals exceeding 100 rpm under certain conditions. Generally, a good cadence in cycling is between 80-100 rpm. [Hur21] The correct gear ratio can help the rider maximize their speed with a comfortable cadence.

A bicycle's speed is the product of the wheel diameter (e.g. 700mm), the wheel's rotation rate (in rpm) and π. The wheel's rotation rate is a function of the rider's cadence and the front to rear gear ratio. As in the previous example, pairs of discrete values such as the number of teeth on the front and rear sprockets are easily represented on a **Type 5** block. This nomograph combines a **Type 5** block (to calculate gearing ratio) with a pair of **Type 2** blocks to calculate the rider's speed given their cadence and gear settings.

1.10.2 Nomograph



1.10.3 Source code

```
"""
    bicycle_cadence.py

    Bicycle gearing cadence and speed calculator
"""
from pynomo.nomographer import *
from pyx import *
import sys
outputfile = sys.argv[0].split('.')[0]+'.pdf'

sys.path.insert(0, "..")
pyx.text.set(text.LatexEngine)


gearing = {
    'block_type': 'type_5',
    'wd_tag': 'ratio',

    'u_func': lambda u: u,
    'v_func': lambda x, v: v/x,

        # teeth on rear cage
    'u_values': [12.0, 14.0, 16.0, 18.0, 21.0, 24.0, 28.0],
    'u_scale_type': 'manual point',
    'u_manual_axis_data': {12.0: '7', 14.0: '6', 16.0: '5', 18.0: '4', 21.0: '3', 24.0: '2', 28.0: '1'},
    'u_title': 'Rear cage',

        # teeth on front derailer
    'v_values': [28.0, 38.0, 48.0],
    'v_scale_type': 'manual point',
    'v_manual_axis_data': {28.0: 'Small', 38.0: 'Medium', 48.0: 'Large'},

    'wd_tick_levels': 2,
    'wd_tick_text_levels': 1,
    'wd_tick_side': 'right',
    'wd_title_opposite_tick': True,
    'isopleth_values': [[14.0, 38.0, 'x']],

}


wheelrpm = {
    'tag': 'wheelrpm',
    'u_min': 90.0,
    'u_max': 360.0,
    'scale_type': 'manual point',
    'function': lambda u: u,
}

crankrpm = {
    'u_min': 60.0,
    'u_max': 120.0,
    'function': lambda u: u,
    'title': r'\large \slshape Cadence (RPM)',
    'tick_levels': 3,
    'tick_text_levels': 2,
    'scale_type': 'linear smart',
        'tick_side':'left',
    'title_draw_center': True,
    # 'title_distance_center': -0.5,
}

ratio = {
    'scale_type': 'manual point',
```

(continues on next page)

(continued from previous page)

```
    'tag': 'ratio',
    'u_min': 1.0,
    'u_max': 4.0,
    'function': lambda u: u,
    'tick_levels': 3,
    'tick_text_levels': 1,
}


rotation = {
    'block_type': 'type_2',
    'f1_params': wheelrpm,
    'f2_params': crankrpm,
    'f3_params': ratio,
    'isopleth_values': [['x', 75, 'x']],
}


speed = {
    'u_min': 15.0,
    'u_max': 45.0,
    'function': lambda u: u,
    'title': r'\large \slshape Speed (km/hour)',
    'tick_levels': 5,
    'tick_text_levels': 2,
    'scale_type': 'linear smart',
    'title_draw_center': True,
    'title_distance_center': -0.5,
}

diameter = {
    'u_min': 600.0,
    'u_max': 800.0,
    'function': lambda u: u*3.1415927*60.0/1000000.0,
    'title': r'Wheel Diameter (mm)',
    'tick_levels': 2,
    'tick_text_levels': 1,
    'scale_type': 'linear smart',
    'title_draw_center': True,
    'title_distance_center': -0.5,

}

wheelrpm2 = {
    'tag': 'wheelrpm',
    'u_min': 90.0,
    'u_max': 360.0,
    'function': lambda u: u,
    'scale_type': 'linear smart',

    'title': r'\large \slshape Wheel RPM',
    'title_x_shift': -18.5,
    'tick_levels': 5,
    'tick_text_levels': 3,
}


speedblock = {
    'block_type': 'type_2',
    'f1_params': speed,
    'f2_params': diameter,
    'f3_params': wheelrpm2,
    'mirror_x': True,
    'isopleth_values': [['x', 750.0, 'x']],
}
```

(continues on next page)

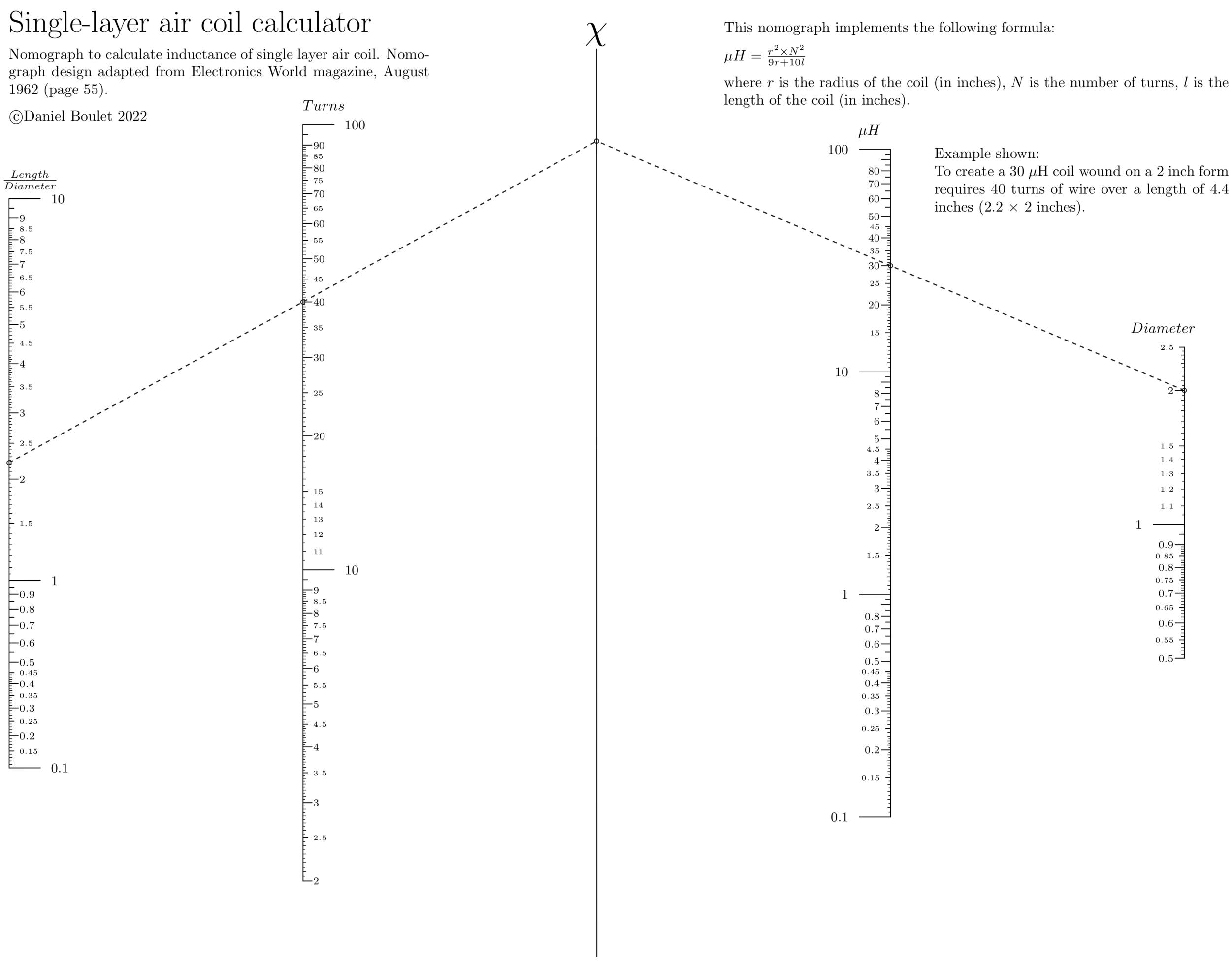
(continued from previous page)

```


main_params = {
    'filename': outputfile,
    'block_params': [gearing, rotation, speedblock],
    'transformations': [('rotate', 0.01), ('scale paper',)],
    'title_str': r'\Large \textbf{Bicycle Cadence Calculator}',
    'title_x': 3.5,
    'title_y': 3.5,

    'extra_texts': [
        {
            'x': -0.5,
            'y': 11.3,
            'text': r'\large \slshape{Gearing Ratio}',
        },
        {
            'x': 7.0,
            'y': 20.5,
            'text': r'\large \slshape{Front derailer setting}',
        },
        {
            'text': r'\copyright Daniel Boulet (2019-2021)',
            'x': -0.5,
            'y': 3.0,
        },


    ],
        # 'make_grid':True,

}

Nomographer(main_params)
```

1.11 Air Coil Design

Air coil design nomographs come in a range of shapes, sizes and methods of use. In this exmaple, PyNomo is used to reconstruct part of the single-layer air coil calculator nomograph published in Electronics World (1962).

1.11.1 Nomograph



1.11.2 Source code

```
"""
    air_core_coil.py

        Nomograph to calculate inductance of single layer air coil.  Design
"""
import sys
import numpy as np
from pyx import *

outputfile = sys.argv[0].split(".")[0] + ".pdf"
sys.path.insert(0, "..")
text.set(text.LatexEngine)

from pynomo.nomographer import Nomographer

length_over_diameter = {
    "u_min": 0.1,
    "u_max": 10.0,
    "function": lambda u: np.log(18.0 + 40.0 * u),
```

(continues on next page)

(continued from previous page)

```
    "title": r"$\frac{Length}{Diameter}$",
    "tick_levels": 5,
    "title_x_shift": 0.5,
    "title_y_shift": 0.35,
    "tick_text_levels": 4,
    "scale_type": "log smart",
}

number_of_turns = {
    "u_min": 2.0,
    "u_max": 100.0,
    "function": lambda u: -2.0 * np.log(u),
    "title": r"$Turns$",
    "tick_levels": 5,
    "tick_text_levels": 4,
    "title_x_shift": 0.5,
    "title_y_shift": 0.35,
    "scale_type": "log smart",
}

inductance = {
    "u_min": 0.1,
    "u_max": 100.0,
    "function": lambda u: np.log(u),
    "title": r"$\mu H$",
    "tick_levels": 5,
    "tick_text_levels": 4,
    "title_x_shift": -0.5,
    "tick_side": "left",
    "title_y_shift": 0.35,
    "scale_type": "log smart",
}


diameter = {
    "u_min": 0.5,
    "u_max": 2.5,
    "function": lambda u: -np.log(u),
    "title": r"$Diameter$",
    "tick_levels": 5,
    "tick_text_levels": 4,
    "title_x_shift": -0.5,
    "tick_side": "left",
    "title_y_shift": 0.35,
    "scale_type": "log smart",
}


block_1_params = {
    "block_type": "type_3",
    "width": 40.0,
    "height": 40.0,
    "f_params": [
        length_over_diameter,
        number_of_turns,
        inductance,
        diameter,
    ],
    "reference_padding": 0.1,
    "reference_titles": ["\Huge$\chi$"],
    "isopleth_values": [["x", 40, 30.0, 2.0]],
}

main_params = {
    "filename": "air_core_coil.pdf",
```

(continues on next page)

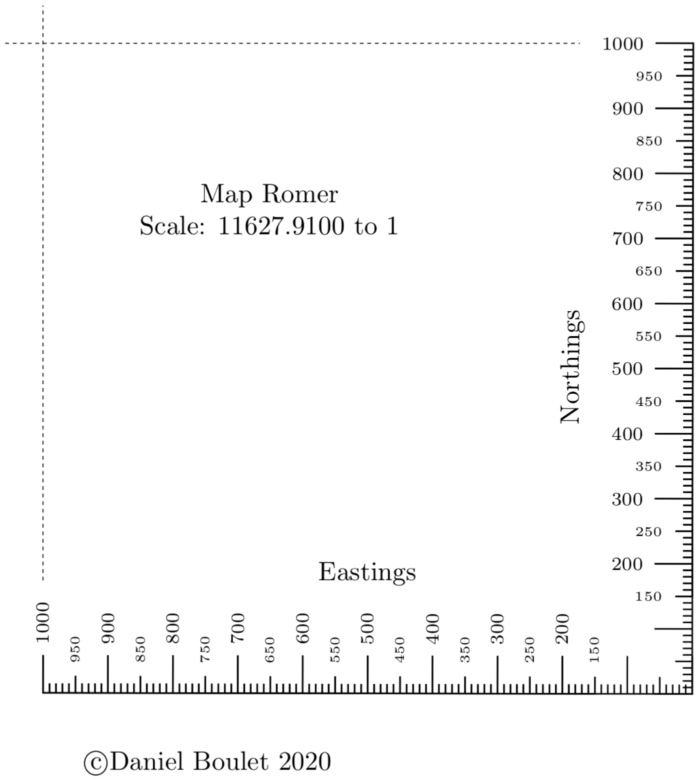
(continued from previous page)

```
    "paper_height": 8.5 * 2.54,
    "paper_width": 11.0 * 2.54,
    "block_params": [block_1_params],
    "transformations": [("rotate", 0.01), ("scale paper",)],
    # "title_str": r"\Huge Single-layer air coil calculator",
    # "make_grid": True,
    "title_y": 23.0,
    "title_x": 5.0,
    "extra_texts": [
        {
            "x": 0.0,
            "y": 22.0,
            "text": r"\noindent \huge Single-layer air coil calculator \par \normalsize \medskip \noindent Nomograph to calculate inductance of single layer air coil.  Nomograph design adapted from Electronics World magazine, August 1962 (page 55). \par \medskip   \noindent \copyright    Daniel Boulet  2022",
            "width": 10.0,
        },
        {
            "x": 17.0,
            "y": 22.0,
            "text": r"\noindent This nomograph implements the following formula: \par \medskip \noindent $\mu H = \frac{r^2 \times N^2}{9r + 10l}$ \medskip \par \noindent where $r$ is the radius of the coil (in inches), $N$ is the number of turns, $l$ is the length of the coil (in inches).",
            "width": 12.0,
        },
        {
            "x": 22.0,
            "y": 19.0,
            "text": r"\noindent Example shown: \par \noindent To create a 30 $\mu$H coil wound on a 2 inch form requires 40 turns of wire over a length of 4.4 inches (2.2 $\times$ 2 inches).",
            "width": 7.0,
        },
    ],
}
Nomographer(main_params)
```

1.12 Map Romer

Strictly speaking, this not a nomogrpah. However, this examples uses PyNomo's axis drawing functions to create a custom map romer to use on printed maps that don't have a standard scale (i.e. 1:25000). The only constraint is that the map's grid must have the same scale vertically and horizontally. Substitue values for actual and printed grids sizes to produce a useful map romer.

1.12.1 Nomograph



1.12.2 Source code

```
"""
    romer.py

    Nomogram to layout a custom romer for map reading. Scale values
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.
"""

from pynomo.nomographer import Nomographer
import sys
import numpy as np
from pyx import *
text.set(text.LatexEngine)

outputfile = sys.argv[0].split('.')[0]+'.pdf'

def deg2rad(deg):
    return np.pi * deg / 180.0


# actual grid size in meters
# actual_grid_size = float(os.environ['ACTUAL_GRID_SIZE'])
actual_grid_size = 1000.0

# printed grid size in centimeters
```

(continues on next page)

(continued from previous page)

```
# printed_grid_size = float(os.environ['PRINTED_GRID_SIZE'])
printed_grid_size = 21.5/2.5

map_scale = actual_grid_size / (printed_grid_size / 100.0)
romer_title = "Map Romer \par Scale: " + \
    "{0:.4f}".format(round(map_scale, 2)) + " to 1"

n_params = {
    'u_min': 0.0,
    'u_max': actual_grid_size,
    'function_x': lambda u: u * np.cos(deg2rad(45.0)),
    'function_y': lambda u: u * np.sin(deg2rad(45.0)),
    'tick_levels': 3,
    'tick_text_levels': 0,
    'tick_side': 'left',
    'title': r'Northings',
    'title_draw_center': True,''
    'title_distance_center': -1.5,
    'extra_params': [{
        'u_min': actual_grid_size/10.0 * 1.5,
        'u_max': actual_grid_size,
        'tick_text_levels': 2,
    }],
    'text_distance_0': 0.65,
    'text_distance_1': 0.4,
    'text_size_0': text.size.scriptsize,
    'text_size_1': text.size.tiny,
    'grid_length_0': 0.5,

}

e_params = {
    'u_min': 0.0,
    'u_max': actual_grid_size,
    'function_x': lambda u: u * np.cos(deg2rad(135.0)),
    'function_y': lambda u: u * np.sin(deg2rad(135.0)),
    'tick_levels': 3,
    'tick_text_levels': 0,
    'title': r'Eastings',
    'title_draw_center': True,
    'title_distance_center': 1.5,
    'extra_params': [{
        'u_min': actual_grid_size/10.0 * 1.5,
        'u_max': actual_grid_size,
        'tick_text_levels': 2,
    }],
    'text_distance_0': 0.65,
    'text_distance_1': 0.4,
    'grid_length_0': 0.5,
    'text_size_0': text.size.scriptsize,
    'text_size_1': text.size.tiny,
}

block_params_n = {
    'block_type': 'type_8',
    'f_params': n_params,
    # 'width': printed_grid_size * 100.0 * np.sqrt(2.0),
    # 'height': printed_grid_size * 100.0 * np.sqrt(2.0),
    'width': 5.0,
    'height': printed_grid_size / np.sqrt(2.0),
}

block_params_e = {
    'block_type': 'type_8',
    'f_params': e_params,
```

(continues on next page)

(continued from previous page)

```
    'width': 5.0,
    'height': printed_grid_size / np.sqrt(2.0),
}

main_params = {
    'filename': outputfile,
    # 'paper_height': printed_grid_size / np.sqrt(2.0),
    # 'paper_width': printed_grid_size * 2.0 / np.sqrt(2.0),
    'paper_height': printed_grid_size,
    'paper_width': printed_grid_size,
    'block_params': [block_params_n, block_params_e],
    'title_str': romer_title,
    'title_x': 3.0,
    'title_y': 6.5,
    'transformations': [('rotate', 44.9), ('scale paper',)],
    # 'make_grid': True,
    'draw_lines': True,
    'line_params': [
        {
            'coords': [
                [0, 1.5, 0, printed_grid_size + 0.5], [-0.5, printed_grid_size,
                                                       printed_grid_size - 1.5, printed_grid_size],
            ],
            'line_style': [color.cmyk.Black, style.linewidth.thin, style.linestyle.dashed],
        },
    ],
    'extra_texts': [
        {
            'x': 0.0,
            'y': -1.0,
            'text': r'\copyright Daniel Boulet  2020',
            'width': 10.0,
        },
    ],

}

Nomographer(main_params)
```

CHAPTER

TWO

LICENSE

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

BIBLIOGRAPHY

[wik07] English: The re-drawn chart comparing the various grading methods in a normal distribution. Includes: Standard deviations, cumulative percentages, percentile equivalents, Z-scores and T-scores. Inspired by Figure 4.3 on Page 74 of Ward, A. W., Murray-Ward, M. (1999). Assessment in the Classroom. Belmont, CA: Wadsworth. ISBN 0534527043. July 2007. URL: https://commons.wikimedia.org/wiki/File:The_Normal_Distribution.svg (visited on 2021-12-31).

[the21] Cycling cadence - what is it and why you should keep a close eye on yours? Mar 2021. URL: https://thebodymechanic.com.au/cycling-cadence-what-is-it-and-why-you-should-keep-a-close-eye-on-yours/.

[tra21] What to consider when finding your ideal cycling cadence. Dec 2021. URL: https://www.trainingpeaks.com/blog/what-to-consider-when-finding-your-ideal-cycling-cadence/.

[EN21] Electronic-Notes. Vswr to return loss conversion table & formulas. 2021. [Online; accessed 6-January-2022]. URL: https://www.electronics-notes.com/articles/antennas-propagation/vswr-return-loss/vswr-return-loss-conversion-table.php.

[Hur21] Sean Hurley. Cycling Cadence: What Is It, What's Most Efficient, and Cadence Drills To Make You Faster - TrainerRoad Blog. October 2021. URL: https://www.trainerroad.com/blog/whats-the-most-efficient-cycling-cadence-and-how-cadence-drills-can-make-you-faster/ (visited on 2021-12-31).

[Sys] Times Microwave Systems. Lmr®-195 flexible low loss communications coax - times ... [Online; accessed 6-January-2022]. URL: https://www.timesmicrowave.com/DataSheets/CableProducts/LMR-195.pdf.

[Wik21a] Wikipedia. Standing wave ratio — Wikipedia, the free encyclopedia. 2021. [Online; accessed 6-January-2022]. URL: https://en.wikipedia.org/w/index.php?title=Standing_wave_ratio&oldid=1061732394.

[Wik21b] Wikipedia. E series of preferred numbers — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=E%20series%20of%20preferred%20numbers&oldid=1059349411, 2021. [Online; accessed 31-December-2021].

[Wik21c] Wikipedia. Voltage divider — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Voltage%20divider&oldid=1047836365, 2021. [Online; accessed 31-December-2021].