Introduction

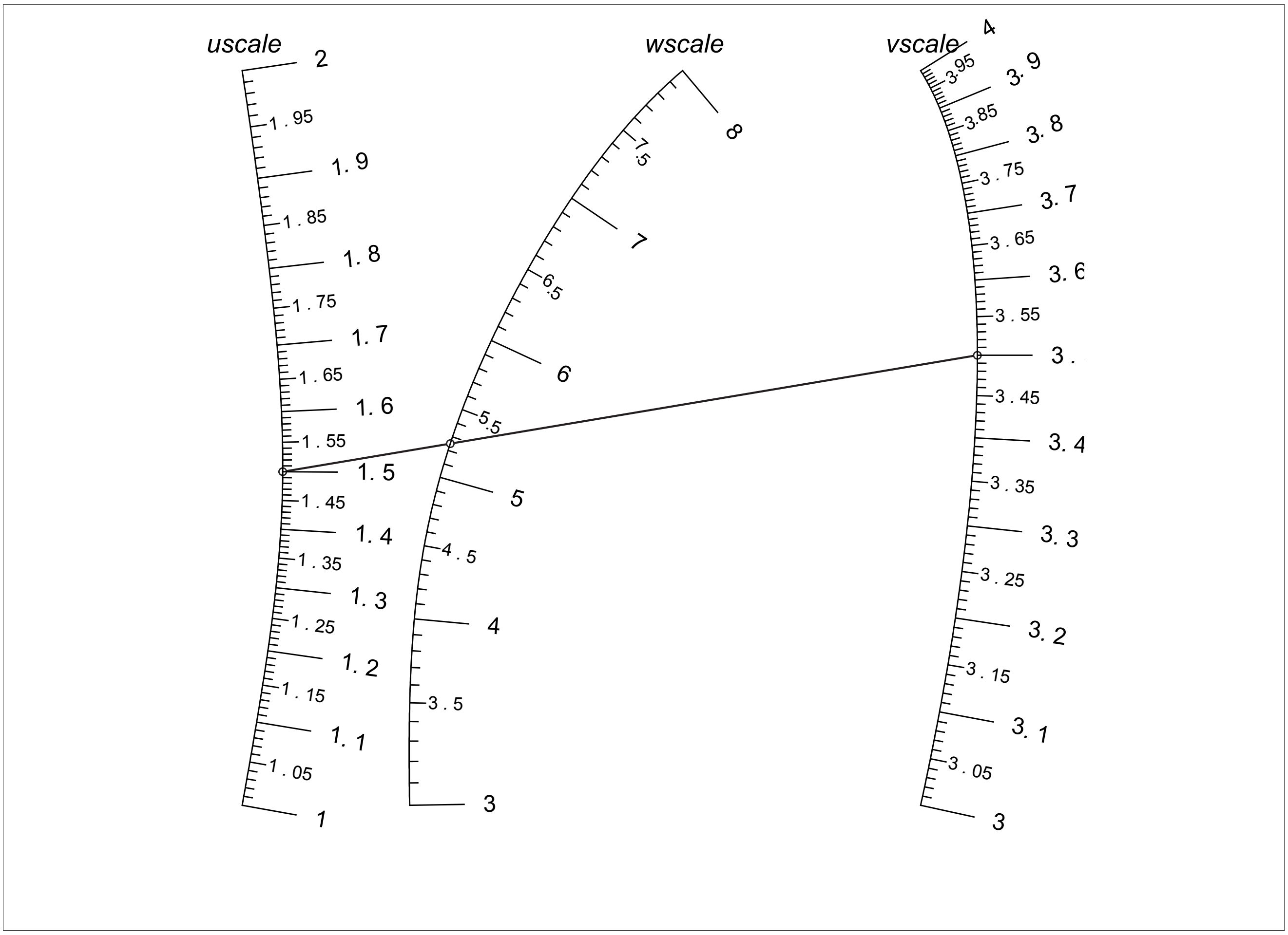
Nomgen makes a nomogram quick & easy to create by auto-generating the scale lines from a given function of 2 variables.

Enter formula, max & min values, generate a nomogram, possibly (probably) non linear.

- use Python to calculate nomogram numerically using SciPy libraries
- output using pynomo libraries

Note some formulas cannot have a nomogram, so need to be able to recognise this.

A nomogram with M lines, N of them curved is known as Class M, Genus N. Thus this project attempts to auto-generate class 3, genus 3 nomograms.



Using nomogen

Assume you have a python3 setup with pynomo, etc., as described in reference [2.].

Write a python function of 2 arguments, corresponding to the left and right hand scales, with the return value corresponding to the middle scale.

Implicit function? Can solve numerically, e.g. see colebrookf.py

So nomogen determines the equations of the scale line for you, but you still need to arrange the nomogram layout for things like the nomogram paper size, title, etc., as described in the pynomo documentation. Also, Ron Doerfler's excellent essay [2.] is definitely worth checking out for instructions on setting up the nomogram.

Nomogen uses your equation to derive and set up the determinant equations, so there's no need to do any of that.

The scale lines are represented by polynomials, so nomogen needs the polynomial degree to use. The more straight the scale lines, and the more evenly spaced the intervals, the lower the polynomial degree needed. Use trial and error, starting with a degree of about 7.

Smaller is faster, larger is more accurate.

A scale line can be configured for 'linear smart' if the scale is approximately evenly spaced. If the high values on the scale are bunched together tighter, the scale type might be better set to 'log' or 'log smart'. Nomogen plots the log of these scale variable, so the program might be faster and more accurate.

Note that some functions are not accurately representable as a nomogram, so reducing the range of one or more scales should improve accuracy.

When a nomogram is calculated, nomogen reports an estimate of the tolerance. If the tolerance is larger than 0.1mm, it is shown on the nomogram.

There's lots of reference examples included with nomogen.

Step by Step Example

Let's take example (xv) from Allcock & Jones, ref [3.], the relative load on a retaining wall.

Details are here:

https://babel.hathitrust.org/cgi/pt?id=mdp.39015000960115&view=1up&seq=128

The formula is:

$$(1+L)h^2 - Lh(1+p) - \frac{1}{3}(1-L)(1+2p) = 0$$

with

$$0.5 \leq L \leq 1$$

$$0.5 \leq p \leq 1$$

thus

$$0.75 \leq h \leq 1 \ .$$

The nomogram is to fit inside a 10 cm x 15 cm rectangle.

Use example file test1.py as a template, copy it to, say, myproj.py and edit that.

Write a new function **AJh(L, p)** replacing (or perhaps ignoring) the existing w(u,v). A beginners tutorial is given in ref [2.].

The function above cannot be rearranged to give a value for h (it's a so-called implicit function of h), but it is a function of the form

$Ah^2 + Bh + C = 0.$, from which h can be found using the quadratic formula.

Assigning values for the maximum and minimum of each of the variables gives a code fragment something like this:

```
def AJh(L,p):
    a = 1+L
    b = -L*(1+p)
    c = -(1-L)*(1+2*p)/3
    h = ( -b + math.sqrt(b**2 - 4*a*c) ) / (2*a)
    return h

Lmin = 0.5; Lmax = 1.0;
pmin = 0.5; pmax = 1.0;
hmin = AJh(Lmax, pmin);
hmax = AJh(Lmin, pmax);
NN = 3
```

The variable NN is the degree of the polynomials that define the scale lines. It can be tricky to get right, but 3 works well for this example.

Now define the scales, with **L** and **p** on the outer scales and **h** (the function value) in the middle. For each scale, set the minimum & maximum values, and (optionally) a title. Leave the other definitions as they are:

```
left_scale = {
    'u_min': Lmin,
    'u_max': Lmax,
    'title': r'$L$',
    'scale_type': 'linear smart',
    'tick_levels': 3,
    'tick_text_levels': 2
}

right_scale = {
    'u_min': pmin,
    'u_max': pmax,
    'title': r'$p$',
    'scale_type': 'linear smart',
    'tick_levels': 3,
    'tick_text_levels': 2 ,
}

middle_scale = {
    'u_min': hmin,
    'u_max': hmax,
    'title': r'$h$',
    'scale_type': 'linear smart',
    'tick_levels': 3,
    'tick_text_levels': 2
}
```

Note that the title text needs to go between the '$' characters, as in r'$myTitle$'. It is not always obvious how to add text, so it's worth checking ref [2.] beforehand.

In the main_params section, set the nomogram height and width, in cm, and add a title:

```
main_params = {
    'filename': __file__.endswith(".py") and __file__.replace(".py", ".pdf")
or "nomogen.pdf",
    'paper_height': 15, # units are cm
    'paper_width': 10,
    'title_x': 4.5,
    'title_y': 1.5,
    'title_str':r'$example$',
    'block_params': [block_params0],
    'transformations': [('scale paper',)],
    'pdegree': NN
}
```

The parameters **'title_x'** & **'title_y'** are the x and y positions of the title, measured in cm from the bottom left of the nomogram.

Finally, give nomogen the function defined above:

```
Nomogen(AJh, main_params);  # generate nomogram for AJh function
```

That's it! Now run the program, and a neatly generated nomogram will be in myproj.pdf.

The pynomo documentation and ref[2.] describe lots of other options for setting up and fine tuning the nomogram. See also the example Ajh.py.

Compare this with the calculations shown in the example in ref [3.], which needs several pages of algebra and geometric theory to derive the equations and fill a determinant to calculate the curves for the nomogram. There's no need for that with nomogen, just write the function and set the parameters.

Deriving a Nomogram

*[**Note:** It is not necessary to read the remainder of the document to use nomogen. It is for reference only. Since nomogen is experimental, some of the ideas here have been tested and abandoned, some others have yet to be tested, what's left should (hopefully) describe the internals of nomogen.]*

Use tried and tested **engineering principles**:

- use others' good work, just hook it all together
- use brute force. (given to us by Moore's law.)

Method outline:

- Approximate each scale line with a polynomial,
- estimate the cost of each approximation. The cost is a function of the error in the scale curves and how well the curves fit into the allocated area
- use python's SciPy numerical optimisation library to minimise this cost,

- use pynomo libraries to scale and plot the nomogram.

The nomogram is calculated to **fit inside a unit square**. This is achieved by one of

1. tie the ends of the outer scales to the corners of the unit square *(current scheme)*
2. use a cost function that rewards the nomogram as it gets larger up to the boundary of the unit square, and then penalises it if it extends outside the unit square. *(possible future scheme)*

The nomogram is generated on a unit square, which PyNomo scales to the required size when it creates the output.

The Polynomials

Each scale line is defined by a Chebyshev polynomial, not with a set of coefficients like $u_x = \sum_{k=0}^{N} \alpha_i T_i(u)$, but by interpolating its values at so-called Chebychev nodes.

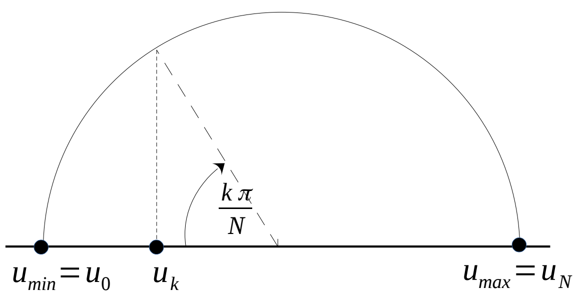
A major problem with using equally spaced nodes to approximate the scale lines (or any function in general) is that the approximation can deviate wildly near the end points. (This is intuitively plausible since there are no nodes beyond the end points to constrain the approximating function.)

Polynomials defined at the Chebyshev nodes counter this by bunching the nodes near the ends, so (very nearly) minimising the maximum error.

If there are N+1 Chebyshev nodes for the u curve, they occur at the points

$$u_k = u_{min} + (u_{max} - u_{min})(1 - \cos(\frac{k * \pi}{N}))/2, \text{ where k = 0,1, .. N}$$

These nodes can be interpreted as the projection from a semi-circle onto the u axis of equally spaced angles, so node u_k is the projection from an angle $k \frac{\pi}{N}$:



This scheme concentrates the nodes near the ends which keeps the approximation close to its correct value.

If we have values of the function at these points, there is a unique N-degree polynomial that fits these points.

We determine the coordinates of the u, v & w curves at the Chebyshev nodes, and from there generate the curves of the nomogram.

Given x_k & y_k, the x and y coordinates of each of these nodes, we can efficiently interpolate the x coordinates of any point w as follows:

(This code assumes N is odd)

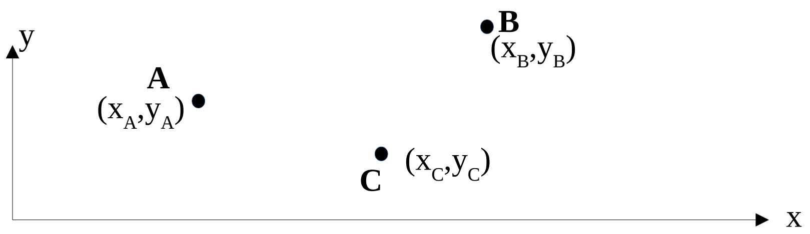
```
if w is one of w_k then
    use x_k & y_k, the known values for node k
else
    sumA = (x_0/(w-w_0) + x_N/(w-w_N))/2;
    sumB = (1/(w-w_0) + 1/(w-w_N))/2;
    for k in 0 .. N
        t = 1/(w-w_k);
        sumA = t*x_k - sumA;
        sumB = t - sumB;
    x(w) = sumA / sumB
```

The y coordinate, y(w) is very similar.

See ref[4.], which also describes how to calculate the first 2 derivatives of the function.

Basic Nomogram Calculations

Suppose we have 3 points, A, B & C in an x-y coordinate system:



The area inside the triangle ABC is

$$Area = \frac{1}{2}\left((x_A - x_B)(y_A - y_C) - (x_A - x_C)(y_A - y_B)\right)$$

(This is the vector cross product of the 2 line segments. It is calculated like a determinant, but technically, it is not a determinant.)

The points A,B,C lie on a straight line if the Area = 0, or

$$(x_A - x_B)(y_A - y_C) = (x_A - x_C)(y_A - y_B)$$

$$x_A y_B + x_B y_C + x_C y_A - x_A y_C - x_B y_A - x_C y_B = 0$$

The distance of the point C from the line AB is

$$d = \frac{2 * Area(ABC)}{base\ line\ AB}$$

$$d = \frac{(x_A - x_B)*(y_A - y_C) - (x_A - x_C)*(y_A - y_B)}{\sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}} \tag{1}$$

We wish to generate a nomogram for the formula

$w = w(u, v)$, where $u_{min} <= u <= u_{max}$, $v_{min} <= v <= v_{max}$

The scale lines are approximated by parametric curves with coordinates given by polynomial functions of u, v and w:

u = (x_u, y_u), where $x_u = x_u(u)$ and $y_u = y_u(u)$

and so on for the v & w scale lines.

This gives us functions $x_u(u)$, $y_u(u)$, etc., and since for any u & v such that w = w(u,v), the nomogram points must lie on a straight line, we can write

$$\frac{(u_x - v_x)}{(u_y - v_y)} = \frac{(u_x - w_x)}{(u_y - w_y)} \quad \text{or}$$

$$(u_x - v_x)(u_y - w_y) = (u_x - w_x)(u_y - v_y) \tag{2}$$

multiply throughout:

$$u_x v_y - u_x w_y - v_x u_y + v_x w_y + w_x u_y - w_x v_y = 0$$

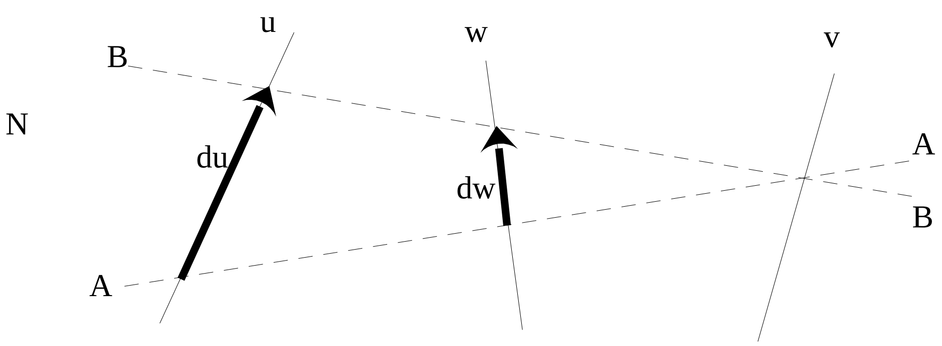
This is equivalent to the determinant:

$$\begin{vmatrix} u_x(u) & u_y(u) & 1 \\ v_x(v) & v_y(v) & 1 \\ w_x(w) & w_y(w) & 1 \end{vmatrix}$$

which can be used as input for a type 9 nomogram in pynomo (see refs [1.] and [2.]).

Derivatives of the scale lines

Consider a small segment of the u, v & w scale lines as follows:



The index line AA intersects the scale lines at (x_u,y_u), (x_v,y_v) & (x_w,y_w).

The index line BB is a small perturbation of AA by an amount du, keeping its intersection with the v scale line unchanged.

The scale lines are defined by the equations $x_u = x_u(u)$, etc., and the function w is defined as w = w(u,v)

We can write the following since the intersection points on the lines are co-linear:

$$(x_u - x_v)(y_w - y_v) = (x_w - x_v)(y_w - y_v) \tag{3}$$

$$(x_u + \frac{dx}{du}du - x_v)(y_w + \frac{dy}{dw}dw - y_v) = (x_w + \frac{dx}{dw}dw - x_v)(y_u + \frac{dy}{du}du - y_v) \tag{4}$$

Expanding (4):

$$\begin{aligned}&(x_u + \frac{dx}{du}du - x_v)\,y_w \;+\; (x_u + \frac{dx}{du}du - x_v)\frac{dy}{dw}dw \;-\; (x_u + \frac{dx}{du}du - x_v)\,y_v \\ = &(x_w + \frac{dx}{dw}dw - x_v)\,y_u \;+\; (x_w + \frac{dx}{dw}dw - x_v)\frac{dy}{du}du \;-\; (x_w + \frac{dx}{dw}dw - x_v)\,y_v\end{aligned}$$

Dropping second order terms & substituting (3):

$$y_w \frac{dx}{du}du + (x_u - x_v)\frac{dy}{dw}dw - y_v \frac{dx}{du}du = y_u \frac{dx}{dw}dw + (x_w - x_v)\frac{dy}{du}du - y_v \frac{dx}{dw}dw$$

$$(y_w - y_v)\frac{dx}{du}du + (x_u - x_v)\frac{dy}{dw}dw = (y_u - y_v)\frac{dx}{dw}dw + (x_w - x_v)\frac{dy}{du}du$$

We also know

$$dw = \frac{\partial w}{\partial u}du + \frac{\partial w}{\partial v}dv \quad ,$$

and since in this case dv = 0, we have:

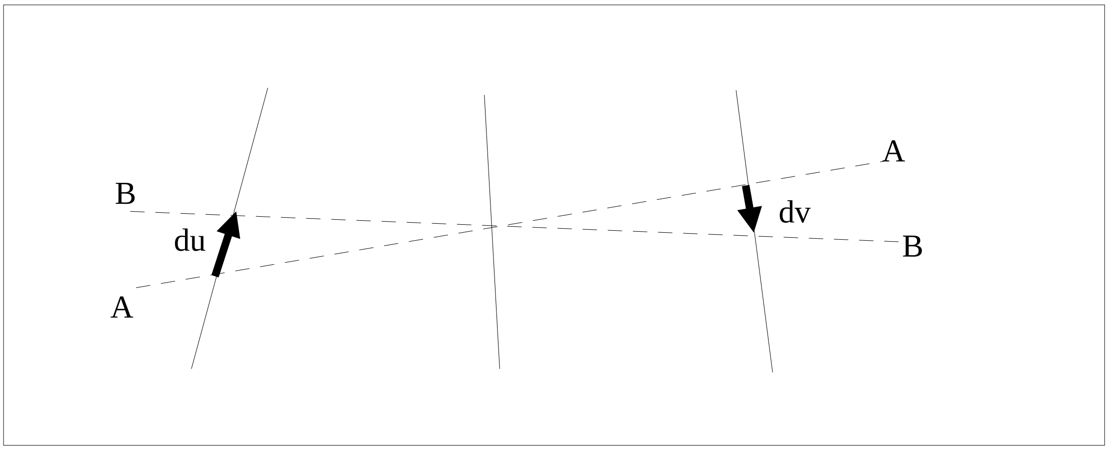
$dw = \frac{\partial w}{\partial u}du$, so substitute this and rearrange:

$$\frac{\partial w}{\partial u}((x_u - x_v)\frac{dy}{dw} - (y_u - y_v)\frac{dx}{dw}) = (x_w - x_v)\frac{dy}{du} - (y_w - y_v)\frac{dx}{du} \tag{5}$$

There is a similar equation for an index line perturbed along the v scale line while keeping the u value constant:

$$\frac{\partial w}{\partial v}((x_u - x_v)\frac{dy}{dw} - (y_u - y_v)\frac{dx}{dw}) = (x_u - x_w)\frac{dy}{dv} - (y_u - y_w)\frac{dx}{dv} \tag{6}$$

Now move an index line by a small amount, but keep the w point unchanged:



As before, we can write:

$$\frac{x_u - x_w}{y_u - y_w} = \frac{x_w - x_v}{y_w - y_v}$$, and

$$\frac{x_u + \frac{dx}{du}du - x_w}{y_u + \frac{dy}{du}du - y_w} = \frac{x_v + \frac{dx}{dv}dv - x_w}{y_v + \frac{dy}{dv}dv - y_w}$$

$$(x_u + \frac{dx}{du}du - x_w)(y_v + \frac{dy}{dv}dv - y_w) = (x_v + \frac{dx}{dv}dv - x_w)(y_u + \frac{dy}{du}du - y_w)$$

$$(y_v - y_w)\frac{dx}{du}du + (x_u - x_w)\frac{dy}{dv}dv = (y_u - y_w)\frac{dx}{dv}dv + (x_v - x_w)\frac{dy}{du}du$$

Now, from $w = w(u, v)$, we get

$$dw = \frac{\partial w}{\partial u}du + \frac{\partial w}{\partial v}dv$$

In this case, $dw = 0$, so

$$-\frac{\frac{\partial w}{\partial u}}{\frac{\partial w}{\partial v}}du = dv$$

Substitute into the above, then tidy up:

$$\frac{\partial w}{\partial v}((x_w - x_v)\frac{dy}{du} - (y_w - y_v)\frac{dx}{du}) = \frac{\partial w}{\partial u}((x_u - x_w)\frac{dy}{dv} - (y_u - y_w)\frac{dx}{dv}) \qquad (7)$$

NOTE: this is not independent of 5 & 6. This can be seen by dividing those 2 equations, then cancelling the common term and cross multiplying the denominators.

So, the coordinates of any index line must satisfy equations 3, 5 & 6. This information can be used to:

1. help determine an initial estimate
2. constrain the numerical solution for a smoother nomogram

TODO: this is just the first term of a Taylor's series about the index line. Is it helpful to go further?

Which way are the scales oriented?

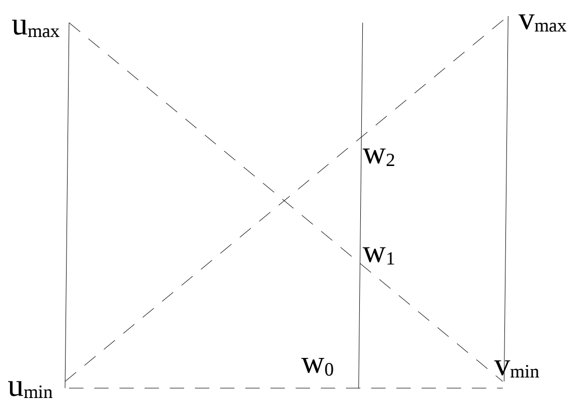
If we assume that the u scale increases (varies from min to max) as the y coordinate increases, and if we define

$$w_0 = w(u_{min}, v_{min})$$

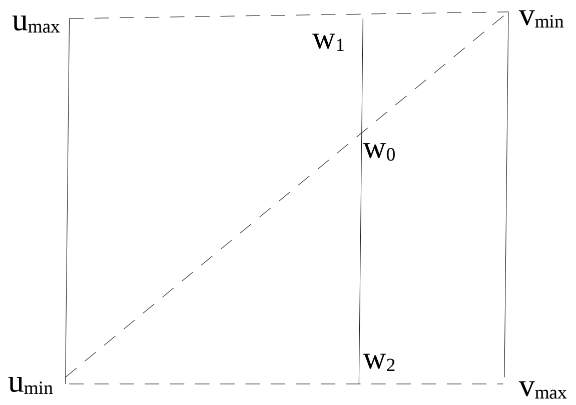
$$w_1 = w(u_{max}, v_{min})$$

$$w_2 = w(u_{min}, v_{max})$$

then the w scale increases upwards if $w_1 > w_0$ and the v scale increases upwards if $(w_1 > w_0) = (w_2 > w_0)$, i.e. w_1 and w_2 are both less than, or both greater than w_0, as can be seen in the diagrams below:



Here, the v scale increases upwards and both w_1 and w_2 must be greater than w_0 if the w scale increases upward, or they both must be less than w_0 if the w scale increases downwards.



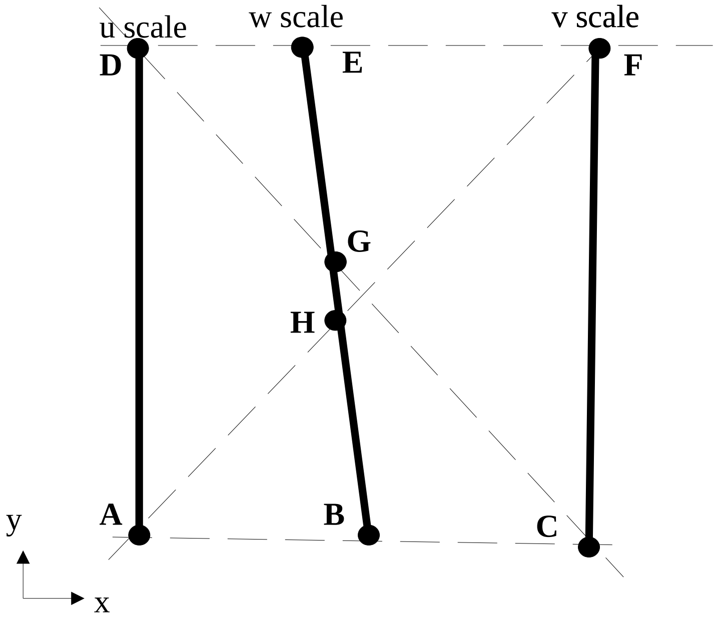
Here, the v scale increases downwards and w_0, must lie between w_1 and w_2. As above, w_1 is greater than w_0 if the w scale is increasing upwards, otherwise w_1 is less than w_0 if the w scale increases downwards

Initial Estimate

The optimisation algorithms need a good initial guess to converge on the optimal solution nomogram.

A Simple Linear Estimate

Try to approximate the true nomogram with a simple linear one as follows:



The u, v & w scale lines are respectively **AD**, **CF** & **BE**.

Point **A** is at (0,0), **C** is at (1,0), **D** is at (0,1), and **F** is at (1,1), u varies from u_{min} to u_{max} & v from v_{min} to v_{max}. Four index lines are shown, and their points of intersection with the scale lines are marked **A** to **H**. The x coordinate of u is given by the function $x_u = x_u(u)$, with similar functions for the y coordinate and the variables v & w.

We have for the equations of each of the scale lines:

$$u_x = 0$$

$$u_y = \frac{u - u_{min}}{u_{max} - u_{min}}$$

$$v_x = 1$$

$$w_x = c + \alpha_{wx}(w - w_B)$$

$$w_y = \alpha_{wy}(w - w_B) \quad ,$$

where c, α_{wx} and α_{wy} are constants yet to be determined.

If the v scale has v_{max} at the top, then define

$$v_{top} = v_{max}, v_{bottom} = v_{min}$$

Otherwise, the v scale is reversed so that v_{min} is at the top, then define:

$$v_{top} = v_{min}, v_{bottom} = v_{max}$$

We can now write

$$v_y = \frac{v - v_{bottom}}{v_{top} - v_{bottom}}$$

G is where the scale line for w intersects the index line for $u=u_{max}$, $v=v_{bottom}$, so $w_G=w(u_{max}, v_{bottom})$, and similarly, $w_H=w(u_{min},v_{top})$, $w_B=w(u_{min},v_{bottom})$ & $w_E=w(u_{max},v_{top})$.

Note that point **B** is at (c,0), **E** is at ($c+\alpha_{wx}(w_E - w_B),\ 1$), and **G** & **H** intersect the diagonal index lines, so

$$\alpha_{wy} = \frac{1}{w_E - w_B} \tag{8}$$

$$c + \alpha_{wx}(w_H - w_B) = \alpha_{wy}(w_H - w_B) = \frac{w_H - w_B}{w_E - w_B} \tag{9}$$

$$c + \alpha_{wx}(w_G - w_B) = 1 - \alpha_{wy}(w_G - w_B) = 1 - \frac{w_G - w_B}{w_E - w_B} = \frac{w_E - w_G}{w_E - w_B} \tag{10}$$

Equation (8) gives α_{wy} and (9) & (10) can be solved to give:

$$\alpha_{wx} = \frac{w_E - w_G - w_H + w_B}{(w_E - w_B)(w_G - w_H)}$$

$$c = \frac{w_H - w_B}{w_E - w_B} - \alpha_{wx}(w_H - w_B)$$

Problems that might occur:

1. w_G might equal w_H, leading to a division by zero in the equation above.
   A simple way to address this is to make the w scale line vertical, or use the derivative equations as described below.
2. this approximation can leave w_B & w_E outside the unit square.
   A simple way to address this is to clip the values so that $0 \le w_B, w_E \le 1$

A more sophisticated approach is to find quadratic and cubic approximations for the scale lines, using the derivative equations to find the required coefficients.

Using derivative equations when $W_G = W_H$

If W_G & W_H are co-incident, then the diagonal index lines meet at (x,y) = (0.5,0.5).

Use the derivative equations (5) & (6) above to determine the slope of the w scale line thruough this point.

Taking the index line from (0,0) to (1,1), we have $(x_u,y_u) = (0,0)$, $(x_w,y_w) = (0.5,0.5)$ and $(x_v,y_v) = (1,1)$ and

$$\frac{\partial w}{\partial u}\left((x_u - x_v)\frac{dy}{dw} - (y_u - y_v)\frac{dx}{dw}\right) = (x_w - x_v)\frac{dy}{du} - (y_w - y_v)\frac{dx}{du}$$

Given that for the initial estimate, all the scale lines are straight, so

$$\frac{dx}{du} = 0, \qquad \frac{dy}{du} = \frac{1}{u_{max} - u_{min}} \quad \text{and}$$

$$\frac{dy}{dw} = \alpha_{wy} = \frac{1}{w_E - w_B}$$

We know w=w(u,v), so we can find $\frac{\partial w}{\partial u}$ for a given (u,v), so the only unknown is $\frac{dx}{dw}$.

Substituting, and rearranging, we find

$$\frac{\partial w}{\partial u}\left((0-1)\alpha_{wy} - (0-1)\frac{dx}{dw}\right) = (0.5-1)\frac{1}{u_{max} - u_{min}} - 0$$

$$\frac{\partial w}{\partial u}\left(\alpha_{wy} - \frac{dx}{dw}\right) = 0.5\frac{1}{u_{max} - u_{min}}$$

$$\frac{dx}{dw} = \alpha_{wy} - \frac{1}{2\frac{\partial w}{\partial u}(u_{max} - u_{min})}$$

Finally, $\alpha_{wx} = \frac{dx}{dw}$ and the w scale can be constructed as shown above.

Note: there are similar estimates for α_{wx} using each of the 2 diagonal index lines, and the 2 derivative equations making 4 estimates in total. We can choose any one, or combine them into an average.

A Quadratic Estimate

Try to approximate the true nomogram with a quadratic one as follows:

TBD

Case 1b: Division by zero when $W_G = W_H$

When $w_G = w_H$, we can assume an initial approximation as follows, and where a, b, c, d, e, f, g & s are parameters to be determined:

$$x_u(u)=a(u-u_{min})(u-u_{max})$$

$$y_u(u)=(u-u_{min})(b(u-u_{max}) + \frac{1}{u_{max}-u_{min}})$$

$$x_v(v)=c(v-v_{bottom})(v-v_{top}) + 1$$

$$y_v(v)=(v-v_{bottom})(d(v-v_{top}) + \frac{1}{v_{top}-v_{bottom}})$$

Note that the above equations make

- the u and v scales quadratic, and
- the scale lines go through their respective corners of the unit square

On the other hand, choose $x_w(w)$ to be a cubic with 3 parameters, e, f, & g:

$$x_w(w)=e(w-w_G)^3 + f(w-w_G)^2 + g(w-w_G) + 0.5$$

Note that this satisfies the requirement $x_w(w_G) = 0.5$

Since $y_w(w_{bottom}) = 0$, $y_w(w_{top}) = 1$ and $y_w(w_G) =0.5$, $y_w(w)$ can be a cubic, which gives one degree of freedom, denoted by the parameter s:

$$y_w(w)=\alpha(w-w_{bottom})^3 + \beta(w-w_{bottom})^2 + s(w-w_{bottom})$$, where

$$\alpha = \frac{-\beta(w_{top}-w_{bottom})^2-s(w_{top}-w_{bottom})+1}{(w_{top}-w_{bottom})^3}$$, and

$$\beta = \frac{As+B}{(w_G-w_{top})(w_G-w_{bottom})^2(w_{top}-w_{bottom})^2}$$, where

$$A=w_G^3 w_{bottom} - w_G^3 w_{top} + 3w_G^2 w_{top} w_{bottom} - 3w_G^2 w_{bottom}^2 + w_G w_{top}^3 - 3w_G w_{top}^2 w_{bottom} + 2w_G w_{bottom}^3 - w_{top}^3 w_{bottom} + 3w_{top}^2 w_{bottom}^2 - 2w_{top} w_{bottom}^3$$

$$B=w_G^3-3w_G^2 w_{bottom}+3w_G w_{bottom}^2-0.5w_{top}^3+1.5w_{top}^2 w_{bottom}-1.5w_{top} w_{bottom}^2-0.5w_{bottom}^3$$

Now use the derivative equations 5 & 6 on the 4 known index lines ABC, DEF, AGF & DGC to give 8 linear equations with the 8 unknowns a, b, c, d, e, f, g & s. C***orrection:*** *there are a couple of non linear terms in there. TBC.*

Case 2: w scale outside unit square

Easy solution: just clip the w scale to remain inside the unit square.

(Work in progress ...)

Another way to address the problem of the line extending outside the unit square is to add a quadratic term to the formula for x_w:

$$x_w = c + \alpha_{wx}(w - w_B) + b(w - w_B)^2 \quad (11)$$

and choosing b, c and α_{wx} so that w_B & w_E lie inside the unit square.

Equations (9) & (10) above are modified to:

$$c + \alpha_{wx}(w_H - w_B) + b(w_H - w_B)^2 = \alpha_{wy}(w_H - w_B) = \frac{w_H - w_B}{w_E - w_B} \quad (12)$$

$$c + \alpha_{wx}(w_G - w_B) + b(w_G - w_B)^2 = 1 - \alpha_{wy}(w_G - w_B) = 1 - \frac{w_G - w_B}{w_E - w_B} \quad (13)$$

Solve by eliminating c, then rearrange so that c and α_{wx} are functions of b ….

$$\alpha_{wx}(w_G - w_H) + b(w_G - w_H)(w_G + w_H - 2w_B) = 1 - \frac{w_G + w_H - 2w_B}{w_E - w_B} \text{ , or}$$

$$\alpha_{wx} = \frac{w_E - w_G - w_H + w_B}{(w_G - w_H)(w_E - w_B)} - b(w_G + w_H - 2w_B) \quad (14)$$

$$c = \frac{w_H - w_B}{w_E - w_B} - \alpha_{wx}(w_H - w_B) - b(w_H - w_B)^2 \quad (15)$$

$\alpha_{wx} = R + bS$ and $c = P + Qb$, where P, Q, R & S are the constants

$$R = \frac{w_E - w_G - w_H + w_B}{(w_G - w_H)(w_E - w_B)}$$

$$S = 2w_B - w_G - w_H$$

$$P = \frac{w_H - w_B}{w_E - w_B} - R(w_H - w_B) \text{ and}$$

$$Q = (w_H - w_B)S + (w_H - w_B)^2$$

Requiring that $w = w_E$ and $w = w_B$ lie on the top and bottom edges of the unit square gives these inequalities:

$0 \le c \le 1$, and $0 \le c + \alpha_{wx}(w_E - w_B) + b(w_E - w_B)^2 \le 1$

$0 <= P + Qb <= 1$, or

$$b >= \frac{-P}{Q}$$

$$b <= \frac{1-P}{Q}$$

and

$$0 \le P+Qb+(R+Sb)(w_E-w_B)+b(w_E-w_B)^2 \le 1 \tag{16}$$

$$b >= -\frac{P + (w_E-w_B)R}{Q + (w_E-w_B)S + (w_E-w_B)^2}$$

$$b <= \frac{1 - P - R(w_E-w_B)}{Q + S(w_E-w_B) + (w_E-w_B)^2}$$

These inequalities give a feasible region for the value of b that makes the w scale line intersect the unit square at the top and bottom edges of the unit square.

The turning point of the parabola must not be in the shaded region shown:

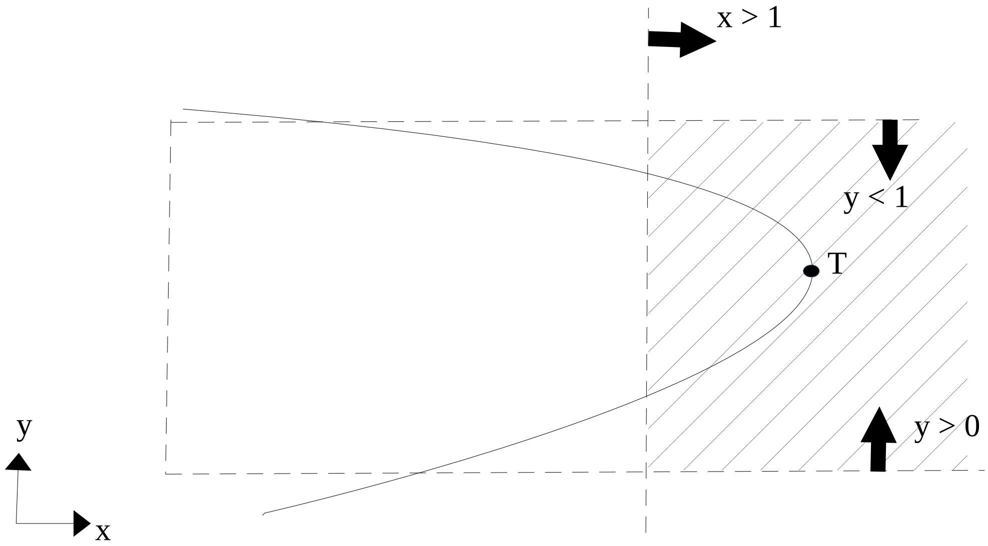


Diagram showing the w scale line with a vertex in a forbidden region

There is another forbidden region on the opposite side of the unit square.

The turning point, T, of the parabola occurs when

$$w'_x = \alpha_{wx} + 2b(w-w_B) = 0$$

giving

$$w_T=w_B - \frac{\alpha_{wx}}{2b}$$

From this, the x and y coordinates of T are:

$$x_T=c + \alpha_{wx}(w_T-w_B)+b(w_T-w_B)^2$$

$$y_T=\alpha_{wy}(w_T-w_B)$$

x_T and y_T can be expressed as functions of b:

… TBC …

The forbidden regions add more constraints on the forbidden values of b:

$x_T > 1$ and $0 < y_T < 1$ or $x_T < 0$ and $0 < y_T < 1$

Careful: combining **allowed** vs **forbidden** regions, **and** conditions vs **or** conditions

look at the extreme x pos of the w curve – it should be inside the unit square. This could add another constraint.

$w_x{}' = \alpha_{wx} + 2b(w - w_B)$

$w_{xB}{}' = \alpha_{wx}$ at B, and $w_{xE}{}' = \alpha_{wx} + 2b(w_E - w_B)$ at E

if these have opposite signs, then there is an extreme, (aka turning point), w_0, where

$\alpha_{wx} + 2b(w_0 - w_B) = 0$.

So may need to find some value of b in the allowed region st $0 <= w_x(w_0) <= 1$ or the derivatives have the same sign.

TBC

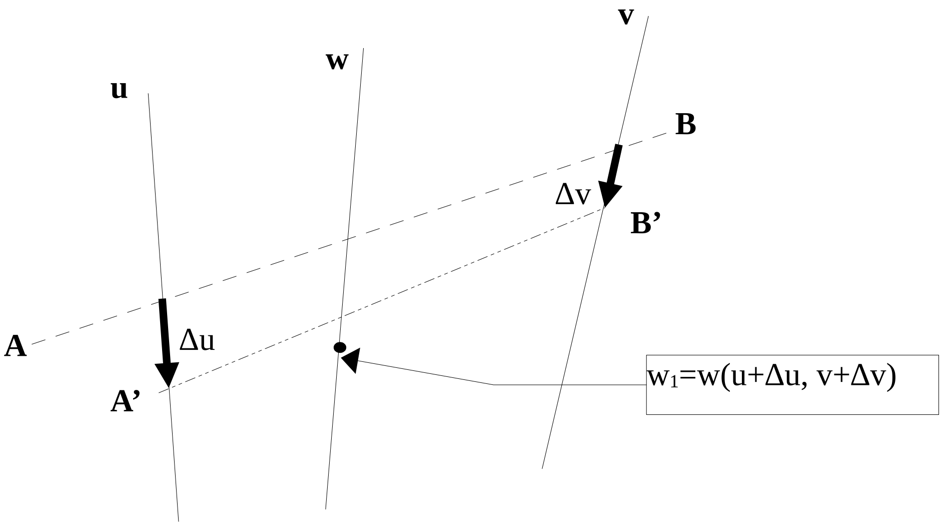
The approximation error

Given a candidate set of the u,v & w scales, determine an error function as $E=\sum_{i,j=0}^{N} e_{ij}^2$, where e_{ij} is the error of the line determined by u_i & v_j.

If the u,v & w scales are approximate, the point corersponding to $w_{ij}=w(u_i, v_j)$ will not lie on the index line connecting u_i & v_j. The distance of w_{ij} from the line is given by equation (1) on page 7, and the coordinates are found from the equations $u_x(u)$, etc.

We can extend this idea by demanding also that the slopes of the scale lines must have minimum error as well.

Consider a line AB intersecting the scale lines at points u & v:



Rewriting (1) above we have the error, e_0. For the line AB:

$$e_0 = \frac{x_u y_v + x_w y_u + x_v y_w - x_w y_v - x_u y_w - x_v y_u}{\sqrt{(x_u - x_v)^2 + (y_u - y_v)^2}}$$

For any nearby line, the intersection coordinates at A’B’ are given by

$$\left(x_u(u) + \frac{dx}{du}\Delta u \ , \ y_u(u) + \frac{dy}{du}\Delta u \right) \text{ and } \left(x_v(v) + \frac{dx}{dv}\Delta v \ , \ y_v(v) + \frac{dy}{dv}\Delta v \right)$$

where Δu and Δv are the small deviations along the u & v scale lines. The point w_1 is the point on the w scale line corresponding to w for the values of u+Δu & v+Δv, and has coordinates ($x_w(w(u+\Delta u , v+\Delta v))$, $y_w(w(u+\Delta u , v+\Delta v))$). If the scale lines are so far only approximate, w_1 will normally not lie on the line A’B’.

The error for the line A’B’ is the distance of the point w_1 from A’B’ and nearby lines, given by

$$E = e_0 + \frac{\partial e}{\partial u}\Delta u + \frac{\partial e}{\partial v}\Delta v$$

Differentiating the expression for e,

$$\frac{\partial e}{\partial u} = \frac{\partial}{\partial u}\left(\frac{x_u y_v + x_w(y_u - y_v) + (x_v - x_u) y_w - x_v y_u}{\sqrt{(x_u - x_v)^2 + (y_u - y_v)^2}}\right)$$

$$\frac{\partial e}{\partial u} = \frac{\frac{dx_u}{du}(y_v - y_w) + \frac{\partial w}{\partial u}\left(\frac{dx_w}{dw}(y_u - y_v) - (x_u - x_v)\frac{dy_w}{dw}\right) - (x_v - x_w)\frac{dy_u}{du}}{\sqrt{(x_u - x_v)^2 + (y_u - y_v)^2}}$$
$$- \frac{e_0\left((x_u - x_v)\frac{dx_u}{du} + (y_u - y_v)\frac{dy_u}{du}\right)}{(x_u - x_v)^2 + (y_u - y_v)^2}$$

where e_0 is the error defined above.

Similarly, for v:

$$\frac{\partial e}{\partial v} = \frac{\partial}{\partial v}\left(\frac{x_u y_v + x_w(y_u - y_v) + (x_v - x_u) y_w - x_v y_u}{\sqrt{(x_u - x_v)^2 + (y_u - y_v)^2}}\right)$$

$$\frac{\partial e}{\partial v} = \frac{\frac{dx_v}{dv}(y_w - y_u) + \frac{\partial w}{\partial v}\left(\frac{dx_w}{dw}(y_u - y_v) - (x_u - x_v)\frac{dy_w}{dw}\right) - (x_w - x_u)\frac{dy_v}{dv}}{\sqrt{(x_u - x_v)^2 + (y_u - y_v)^2}}$$
$$+ \frac{e_0\left((x_u - x_v)\frac{dx_v}{dv} + (y_u - y_v)\frac{dy_v}{dv}\right)}{(x_u - x_v)^2 + (y_u - y_v)^2}$$

When the errors e_0, $\frac{\partial e}{\partial u}$ and $\frac{\partial e}{\partial v}$ are reduced to zero, these equations match the equations found in the section Derivatives of the scale lines on page 7..

This gives an error funtion for the (u,v) pair:

$E_{ij} = e_0^2 + \mu_u\left(\frac{\partial e}{\partial u}\right)^2 + \mu_v\left(\frac{\partial e}{\partial v}\right)^2$, where μ_u and μ_v are some scaling constants.

- Create an error function by summing the squares of all errors over all nodes for u & v
- Use SciPy to minimise this error as a function of the node coordinates
- if the minimum error is acceptably small, then the solution is the nodes that define the nomogram, otherwise a solution cannot be found.

There are some issues that need to be resolved in the above scheme:

1. The number of error calculations is of the order $O(n^2)$, where n is the degree of the polynomial, but the number of coordinates to fix is of the order O(n). If n is large, pruning the number of error calculations can improve computation speed. *(This doesn't work.)*

2. If the distance between the tic marks of one of the scale lines is nonlinear, then the Chebyshev nodes are bunched together at some part in the line, and spread out at another. Then the error calculations are not accurate.

Instead of defining the chebyshev nodes on the values of u (or v or w), define them on some function $l_u(u)$ of the scale line, and similarly for $l_v(v)$ & $l_w(w)$

Assume $l_u(u)$ is a natural log function:

$$l_u \ = \ c \ln(au+b) \tag{17}$$

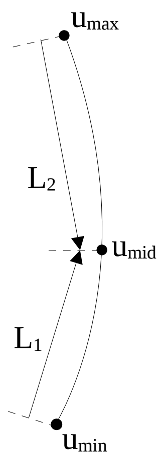
and that $l_u(u)$ ranges from 0 to 1 as u varies from u_{min} to u_{max}.

$$0 \ = \ c \ln(a u_{min}+b)$$
$$1 \ = \ c \ln(a u_{max}+b)$$

Taking c = 1, we have the solution

$$l_u(u)=\ln\left((e-1)\frac{(u-u_{min})}{(u_{max}-u_{min})} \ + \ 1\right) \tag{18}$$

Suppose we have an approximation of $u_x(u)$ & $u_y(u)$, so we know the coordinates of u_{mid}, where $u_{mid} = (u_{min} + u_{max})/2$. We then know L_1 & L_2 which are respectively the distances from u_{mid} to u_{min} and u_{max}.



Now we can write

$$L \ = \ \frac{L_1}{L_1+L_2} \ = \ c \ln(a u_{mid}+b)$$

If the L scale is perfectly linear, $L_1 == L_2$, so L is 0.5, and if the L scale is perfectly logarithmic,

$$\frac{(u_{mid}-u_{min})}{(u_{max}-u_{min})}=0.5, \quad \text{and}$$

$$L \;=\; c\ln\left((e-1)\frac{(u_{mid}-u_{min})}{(u_{max}-u_{min})} \;+\; 1\right) \;=\; 0.62$$

This gives us a method to decide if the scale should be linear or logarothmic – if L is closer to 0.62 than it is to 0.5 use a log scale, otherwise use a linear scale.

As an alternative to using the midpoint of the scale line, let us assume we know α_0 and α_1, where

$$\alpha_0 \;=\; \left.\frac{dS}{du}\right|_{u=u_{min}}$$, evaluated at $u=u_{min}$

$$\alpha_1 \;=\; \left.\frac{dS}{du}\right|_{u=u_{max}}$$, evaluated at $u=u_{max}$

and whereEdit

$$\frac{dS}{du} \;=\; \sqrt{\left(\frac{dx_u}{du}\right)^2 \;+\; \left(\frac{dy_u}{du}\right)^2}$$

is the rate of change of u along the u scale line. It differs from $l_u(u)$ by a constant scale factor.

Now differentiate (17) with respect to u:

$$\frac{dl_u(u)}{du} \;=\; \frac{ac}{au \;+\; b}$$

The ratio of this at u_{min} and u_{max} is the same as the ratio of α_0 and α_1

$$\frac{au_{max}+b}{ac}\frac{ac}{au_{min}+b} \;=\; \frac{\alpha_1}{\alpha_0}$$

Cancel terms and substitute equations (18) above:

$$\exp\left(\frac{1}{c}\right) \;=\; \frac{\alpha_1}{\alpha_0} \tag{19}$$

Substitute this into equations (18) and solve:

$$a \;=\; \frac{\alpha_1 \;-\; \alpha_0}{\alpha_0(u_{max}-u_{min})}$$

$$b \;=\; 1-a\,u_{min}$$

Recall that the positions of Chebychev nodes are:

$$l_u(u_k) \ = \ \frac{1}{2}(1 \ - \ \cos(\frac{k}{N}\pi)) \ = \ c \ \ln(au_k \ + \ b) \ \ ,$$

where N is the number of Chebychev nodes that define $x_u(u)$ (see page 5) and k = 0..N.

Rearranging:

$$u_k \ = \ \frac{\exp(\frac{1}{2c}(1 \ - \ \cos(\frac{k}{T}\pi))) \ - \ b}{a}$$

Tabulated data

Since the construction method already outlined uses a least-squares approach, it should be possible to construct a nomogram from tabulated data.

Instead of using u_i, v_j & w_{ij} derived from equations, use these values derived from the tabulated data.

(To be investigated)

Cost Functions

If we have a formula

$$w = w(u, v), where\ u_{min} \le u \le u_{max}, v_{min} \le v \le v_{max}$$

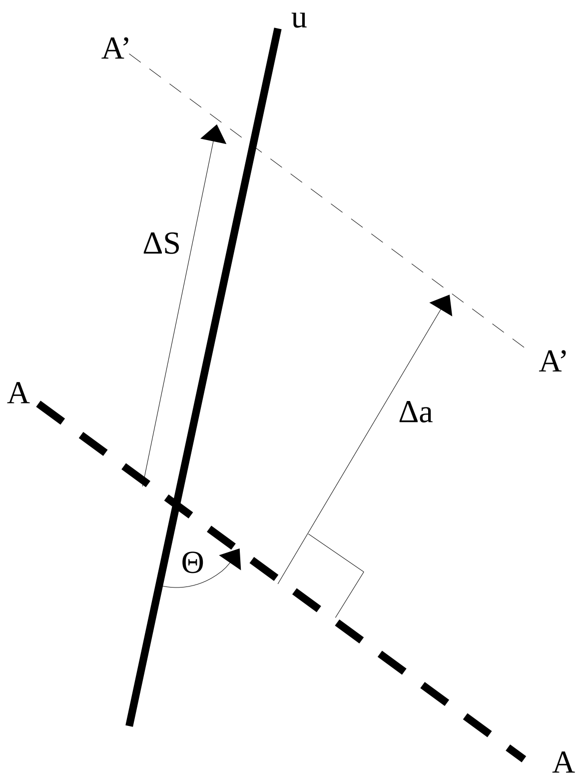
we can determine the position of the u, v & w curves at the Chebyshev nodes, and from there generate the curves of the nomogram.

To get best accuracy, we want the nomogram to be as large as much as possible, but lie inside the area available..
Available methods are:

1. Tie the ends of the outer scale line to the unit square.
   This is the simplest method, but doesn't take advantage of V-type nomograms.
2. Maximise area between scale lines. See section "Use Area Between the Scale Lines" below.
3. Minimise alignment error. If the scale lines are as tall as possible, the corresponding scale variable takes the maximun length. So an alignment error of, say, 0.1mm will correspond to the smallest possible error in the scale variable. Also, if the scale lines are separated as widely as possible, any parallax type errors will be minimised.

The alignment error

Even with a perfectly constructed nomogram, you cannot draw an index line with perfect accuracy. If we assume some tolerance in the position of the index line, we want the measurements of each of the scale variables to be as tolerant as possible to a given position error in that index line.
Here's a diagram of an index line intersecting the u scale line:



The measured line A'A' has a tolerance of ± Δa from the true line AA. The measured line intersects the u scale a distance of ± ΔS from the exact position. For a given Δa, we want the corresponding Δu to be as small as possible. Mathematically, we want to minimise $\frac{du}{da}$ subject to the nomogram fitting inside the unit square.

We can write:

$$\Delta a \;=\; \sin(\Theta)\Delta S$$

$$\Delta S = \sqrt{\frac{d x_u}{d u}^2 + \frac{d y_u}{d u}^2}\,\Delta u$$

For any given (u,v) pair, the line AA intersects the points (x_u(u), y_u(u)) and (x_v(v), y_v(v)), and at the intersection, the u scale line is aligned with the vector ($\frac{d x_u}{d u}$, $\frac{d y_u}{d u}$).
Now use the definition of the vector cross product to write:

$$\sin(\Theta)=\frac{\frac{dx_u}{du}(y_u(u)-y_v(v)) \ - \ \frac{dy_u}{du}(x_u(u)-x_v(v))}{\sqrt{(\frac{dx_u}{du})^2 \ + \ (\frac{dy_u}{du})^2}\sqrt{(x_v(v)-x_u(u))^2 \ + \ (y_v(v)-y_u(u))^2}}$$

Combine the above:

$$\Delta a \ = \ \frac{\frac{dx_u}{du}(y_u(u)-y_v(v)) \ - \ \frac{dy_u}{du}(x_u(u)-x_v(v))}{\sqrt{(\frac{dx_u}{du})^2 \ + \ (\frac{dy_u}{du})^2}\sqrt{(x_v(v)-x_u(u))^2 \ + \ (y_v(v)-y_u(u))^2}} \sqrt{\frac{dx_u}{du}^2 \ + \ \frac{dy_u}{du}^2}\,\Delta u$$

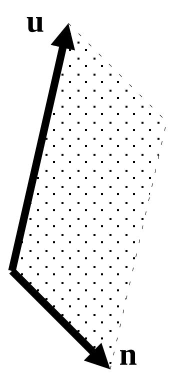
$$\Delta a \ = \ \frac{\frac{dx_u}{du}(y_u(u)-y_v(v)) \ - \ \frac{dy_u}{du}(x_u(u)-x_v(v))}{\sqrt{(x_v(v)-x_u(u))^2 \ + \ (y_v(v)-y_u(u))^2}}\Delta u$$

Now reaarrange and take the limit:

$$\frac{du}{da} \ = \ \frac{\sqrt{(x_v(v)-x_u(u))^2 \ + \ (y_v(v)-y_u(u))^2}}{\frac{dx_u}{du}(y_u(u)-y_v(v)) \ - \ \frac{dy_u}{du}(x_u(u)-x_v(v))} \qquad (20)$$

Note that the reciprocal of this is the cross product of two vectors, **u** and **n**, where **u** is $(\frac{dx_u}{du}, \ \frac{dy_u}{du})$ which is the direction of the u scale line. The vector **n** is $\frac{(y_u(u)-y_v(v), \ \ x_u(u)-x_v(v))}{\sqrt{(x_v(v)-x_u(u))^2 \ + \ (y_v(v)-y_u(u))^2}}$ which is a unit vector along the index line AA.

The magintude of this vector cross product is the area between **u** and **n** as follows:



So minimising equation (20) is equivalent to maximising this area (because it’s a reciprocal). This means we want to:

1. make **u** as long as possible, ie make the scale lines as tall as possible. Remember, **n** is a unit vector, so its length is fixed.

2. make the vectors as close to perpendicular as possible, ie make the scale lines as far apart as possible.

This matches our intuation on what makes a better nomogram, and now we have an expression which can be used to find an optimal nomogram.

Use Area Between the Scale Lines

Using a line integral to determine the cost of a nomogram

See for example, ref[5.]

The cost of a nomogram is given by the double integral

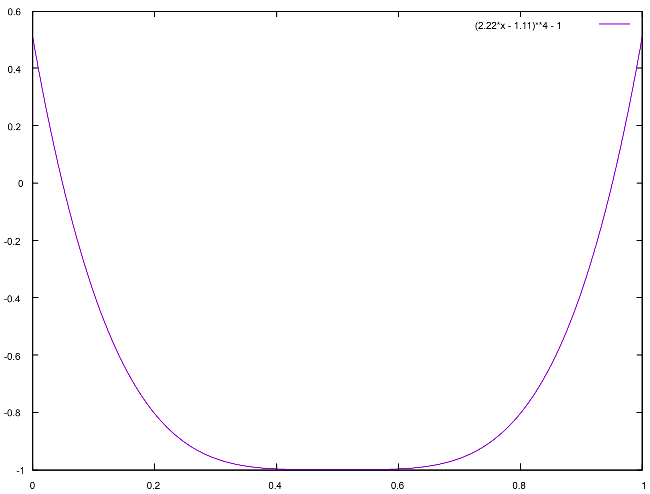
$$\iint_A c(x,y)\ \mathrm{dA} \quad ,$$

where c(x,y) is the cost function.

Consider this cost function:

$$c(x,y) = (20x/9-10/9)^4 + (20y/9-10/9)^4 - 1$$

It’s value is about -1 in the central region of the unit square, but near the edges it quickly grows positive as it approaches the boundaries at 0 or 1:



(TODO: investigate cost function $e^{-ax} + e^{a(x-1)}$ *)*

If we define the cost of the nomogram as

$$cost = \iint_A c(x,y)\,dxdy$$

then the optimisation algorithm can minimise the cost by of the nomogram by using as much area as possible in the middle and up to the edges. Near the edges, the cost savings get smaller and the costs start to increase rapidly beyond the boundary of the unit square.

This can be turned into a line integral if we use Green's theorem with a vector function, **F** whose curl is c(x,y), i.e.

$$c(x,y) = \nabla \boldsymbol{x} \boldsymbol{F} = \frac{\partial F_2}{\partial x} - \frac{\partial F_1}{\partial y}$$

Such a function is:

$$\boldsymbol{F} = \left(y - y(20x/9-10/9)^4\right)\boldsymbol{i} + \left(x(20y/9-10/9)^4\right)\boldsymbol{j}$$

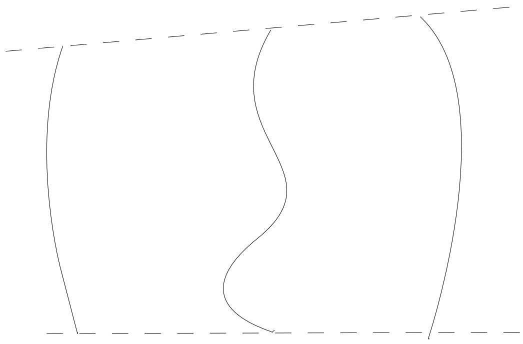
The line integral is

$$cost = \oint_C \left(y - y(20x/9-10/9)^4\right)dx + x(20y/9-10/9)^4\, dy$$

Along, say, the w scale line, we have $x_w = x_w(w)$, so $dx = x_w'(w)dw$ and similarly $dy = y_w'(w)dw$, and the cost function becomes

$$\oint_C \boldsymbol{F}.d\boldsymbol{s} = \int_a^b \boldsymbol{F}(\boldsymbol{c}(t)).\boldsymbol{c}'(t)dt$$

$$cost = \oint_C \left(y - y(20x/9-10/9)^4\right)x_w'(w) + \left(x(20y/9-10/9)^4\right)y_w'(w)dw$$



In the fictitious nomogram above, there are 3 areas –

1. s1 = the area between the 2 outer scales,
2. s2 = the area between the left and middle scales, and
3. s3 = the area between the right and middle scales

Ideally the nomogram cost should be least when the smaller 2 areas are roughly equal, i.e. $s2 \simeq s3$

If the cost is defined by combining the 3 areas as follows

$$\frac{1}{cost} = \frac{1}{s1} + \frac{1}{s2} + \frac{1}{s3}$$, or

$$cost = \frac{s1\ s2\ s3}{s1\ s2 + s2\ s3 + s3\ s1}$$

then the cost should me minimised when the smallest area is a large as possible.

Notes:

1. In the diagram above, s1 = s2 + s3. Could be slightly different if the min & max values of the middle scale do not lie on the dotted lines joining the 2 outer scales.
2. If the cost function is revised to
   $c(x,y) = (2x-1)^4 + (2y-1)^4 - 1$
   then the minimum cost of an area is -0.6, i.e. when an area exactly covers the unit square
3. By taking differentials, small changes in area affect the cost follows
   $\frac{\Delta(cost)}{cost^2} = \frac{\Delta s1}{s1^2} + \frac{\Delta s2}{s2^2} + \frac{\Delta s3}{s3^2}$

Cost function $\kappa(x^2 + y^2)$ can be implemented with vector field $\boldsymbol{F}(x,y) = (-\kappa x^2 y, \kappa x y^2)$

Boundary must be piecewise smooth, continuous & connected. accurate

TBD
finding w' - see ref [4.]
determine the circular path
detecting and handling crossovers

References

1. PyNomo documentation
2. http://www.myreckonings.com/pynomo/CreatingNomogramsWithPynomo.pdf
3. Allcock and Jones: The Theory and Practical Construction of Computation Charts
4. “Barycentric Lagrange Interpolation”, Jean-Paul Berrut & Lloyd N. Trefethen, SIAM REVIEW Vol. 46, No. 3, pp. 501–517
5. https://mathinsight.org/greens_theorem_find_area