Generating a Nomogram Numerically

This document describes the technical details of nomogen – a component of **pynomo** that generates a 3-line nomogram numerically from an equation of the form *w=w(x,y)*

Introduction

Audience

This document is intended for developers and others who wish to understand the techniques used by nomogen to generate nomograms – it is not necessary to read this document to use nomogen.

A basic level of undergraduate engineering mathematics is assumed, in particular of partial derivatives.

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.

Method Outline

- Approximate each scale line with a polynomial defined by a finite number of points
- 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.

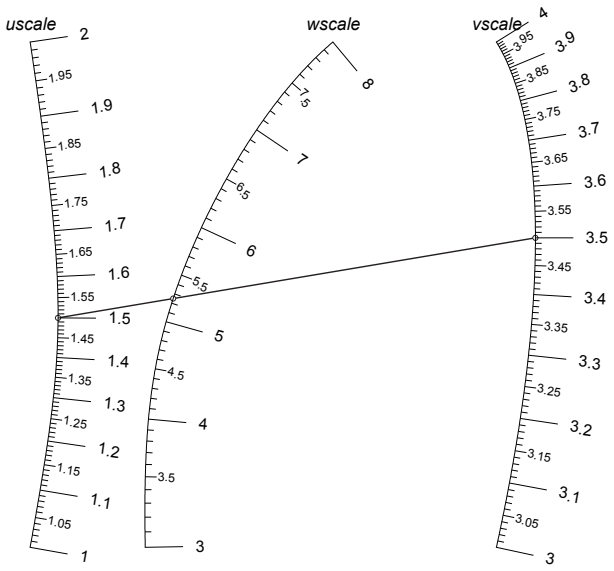
Use well known **engineering principles**:

- Use tried and tested components and techniques.
  Just hook together the good work of others.
- use brute force. (given to us by Moore's law.)

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 *(the default)*
2. optimise 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.

The following assumes we are to generate a nomogram for the function *w=w(u,v)*. The **u**, **v** & **w** scales are respectively on the left, right and middle of the nomogram.



The x and y coordinates of the value *u* on the **u** scale are given by the parametric functions $x_u(u)$ and $y_u(u)$, and so on for the other scales. Note the convention used in this document where **bold** type designates a scale line or axis, and *italic* type designates a variable or function on the axis.

Our goal is to use numerical techniques to find polynomials that closely approximate $x_u(u)$, etc.

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

Representing the Axes with Polynomials

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

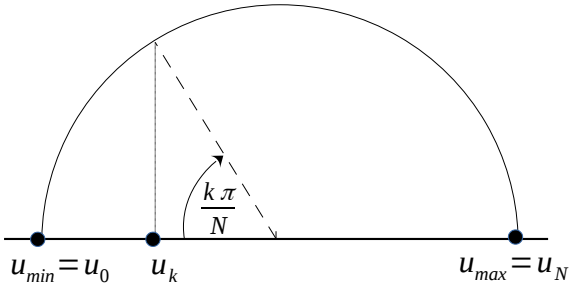
A major problem with using equally spaced nodes to approximate the scale lines (or any function in general) is that the approximation can deviate wildly near the end points – see reference [6]. (This is known as **Runge's phenomenon** and is intuitively plausible since there are no nodes beyond the end 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 between between the polynomial and the function being approximated.

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

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

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.

This scheme has several advantages (see reference[4]):

1. it is efficient
2. it is robust and stable, even for high order polynomials. See Appendix I. below.
3. it avoids Runge spikes, see chapter 13 of reference[4]
4. it is very close to the best polynomial that approximates the true function.

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

Taking the **u** scale as an example, we need to define the function $x_u(u)$. Given x_k, the x coordinate of the node u_k, we can efficiently and accurately interpolate the x coordinate of any point u as follows:

(This code assumes N is odd)

```
if u is one of uₖ then
    use xₖ & yₖ, the known values for node k
else
    sumA = (x₀/(u-u₀) + xₙ/(u-uₙ))/2
    sumB = (1/(u-u₀) + 1/(u-uₙ))/2
    for k in 0 .. N
        t = 1/(u-uₖ)
        sumA = t*xₖ - sumA
        sumB = t - sumB
    x(u) = sumA / sumB
```

The y coordinate, $y_u(u)$ is very similar, as are the functions for the other scale lines.

See references [4] and [5], which also describe how to calculate the derivatives of the function.

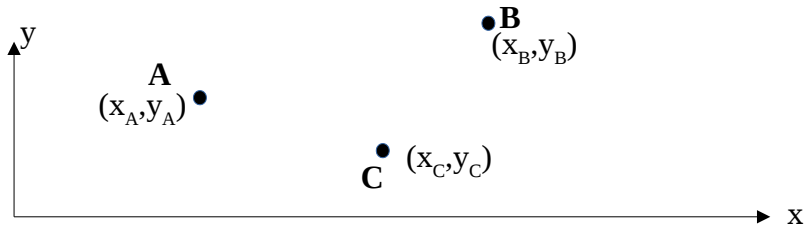
So now we can see a way to generate a nomogram:

1. Reduce the nomogram to a set of carefully chosen points along each axis. These are the Chebyshev points.

2. Arrange these points so the nomogram formula is correct for this set of points. This will involve determining the error of a given arrangement of the points, and using the Python libraries to minimise the error.
3. Interpolate the accurate set of points to fill in each axis.

Basic Nomogram Calculations

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



The area inside the triangle ABC is

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

(This is the vector cross product of the 2 line segments. It is calculated like a determinant.)

Alternatively, the area can be given by

$$Area = \frac{1}{2} |AB| |AC| \sin(\theta) ,$$

where |AB| and |AC| are the lengths of the line segments **AB** & **AC**, and θ is the angle between the two line segments.

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

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

$$x_A y_B + x_B y_C + x_C y_A - x_A y_C - x_B y_A - x_C y_B = 0 \quad (1)$$

The distance of the point C from the line AB is

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

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

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

$$\frac{(x_u - x_v)}{(y_u - y_v)} = \frac{(x_u - x_w)}{(y_u - y_w)} \quad \text{or rearranging}$$

$$x_u y_v - x_u y_w - x_v y_u + x_v y_w + x_w y_u - x_w y_v = 0$$

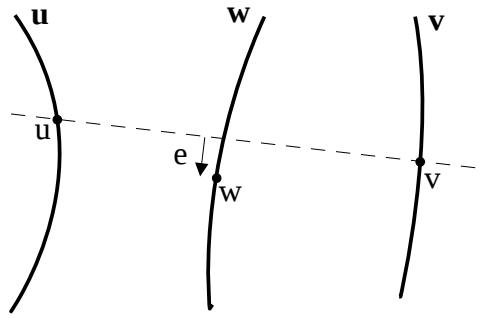
This is an alternative derivation of (1) above, and is equivalent to the determinant:

$$\begin{vmatrix} x_u(u) & y_u(u) & 1 \\ x_v(v) & y_v(v) & 1 \\ x_w(w) & y_w(w) & 1 \end{vmatrix}$$

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

Given the function $w=w(u,v)$, and assuming we have approximate functions for the scale lines as above, we have the x & y coordinates of points u, v and w. Since these functions are only approximate, the point w on the **w** scale does not lie exactly on the index line connecting u & v, as shown below.

This error is shown as e in the following diagram:



The strategy to generate a correct nomogram is to jiggle the equations of the scale lines until e converges to zero. (The python SciPy optimisation libraries do this for us.)

Derivatives of the scale lines

We have just seen that we want the error in the scale lines to be zero. To increase accuracy, we want to ensure the scale lines do not head off in a direction that makes the error larger.

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

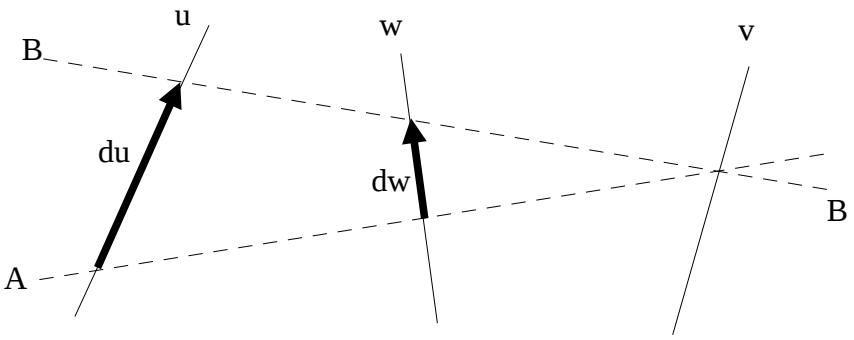


Figure 1:

In Figure 1 above, 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.

We assume that index line AA has no error, and we want to find the conditions that guarantee that line BB also has no error.

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:

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

$$\left(x_u + \frac{dx_u}{du}du - x_v\right)\left(y_w + \frac{dy_w}{dw}dw - y_v\right) = \left(x_w + \frac{dx_w}{dw}dw - x_v\right)\left(y_u + \frac{dy_u}{du}du - y_v\right) \tag{4}$$

Expanding (4):

$$\begin{aligned}&\left(x_u + \frac{dx_u}{du}du - x_v\right)y_w + \left(x_u + \frac{dx_u}{du}du - x_v\right)\frac{dy_w}{dw}dw - \left(x_u + \frac{dx_u}{du}du - x_v\right)y_v \\ &= \left(x_w + \frac{dx_w}{dw}dw - x_v\right)y_u + \left(x_w + \frac{dx_w}{dw}dw - x_v\right)\frac{dy_u}{du}du - \left(x_w + \frac{dx_w}{dw}dw - x_v\right)y_v\end{aligned}$$

Dropping second order terms & substituting (3):

$$y_w\frac{dx_u}{du}du + (x_u - x_v)\frac{dy_w}{dw}dw - y_v\frac{dx_u}{du}du = y_u\frac{dx_w}{dw}dw + (x_w - x_v)\frac{dy_u}{du}du - y_v\frac{dx_w}{dw}dw$$

$$(y_w - y_v)\frac{dx_u}{du}du + (x_u - x_v)\frac{dy_w}{dw}dw = (y_u - y_v)\frac{dx_w}{dw}dw + (x_w - x_v)\frac{dy_u}{du}du$$

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:

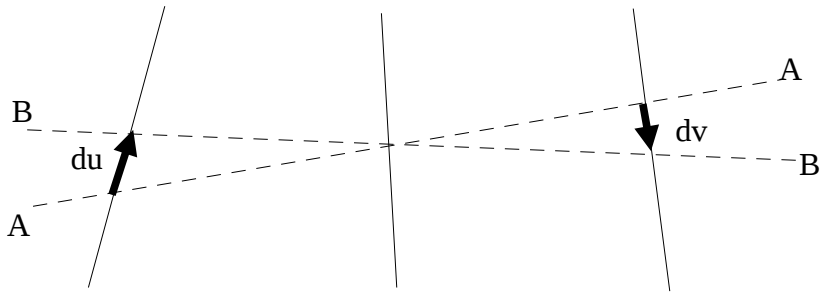
$$dw = \frac{\partial w}{\partial u} du \quad \text{, so substitute this and rearrange:}$$

$$\frac{\partial w}{\partial u}\left((x_u - x_v)\frac{dy_w}{dw} - (y_u - y_v)\frac{dx_w}{dw}\right) = (x_w - x_v)\frac{dy_u}{du} - (y_w - y_v)\frac{dx_u}{du} \tag{5}$$

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

$$\frac{\partial w}{\partial v}\left((x_u - x_v)\frac{dy_w}{dw} - (y_u - y_v)\frac{dx_w}{dw}\right) = (x_u - x_w)\frac{dy_v}{dv} - (y_u - y_w)\frac{dx_v}{dv} \tag{6}$$

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



As before, we can write:

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

$$\frac{x_u + \frac{dx_u}{du} du - x_w}{y_u + \frac{dy_u}{du} du - y_w} = \frac{x_v + \frac{dx_v}{dv} dv - x_w}{y_v + \frac{dy_v}{dv} dv - y_w}$$

$$\left(x_u + \frac{dx_u}{du} du - x_w\right)\left(y_v + \frac{dy_v}{dv} dv - y_w\right) = \left(x_v + \frac{dx_v}{dv} dv - x_w\right)\left(y_u + \frac{dy_u}{du} du - y_w\right)$$

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

$$(y_v - y_w)\frac{dx_u}{du}du + (x_u - x_w)\frac{dy_v}{dv}dv = (y_u - y_w)\frac{dx_v}{dv}dv + (x_v - x_w)\frac{dy_u}{du}du$$

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:

$$\frac{\partial w}{\partial v}((x_w - x_v)\frac{dy_u}{du} - (y_w - y_v)\frac{dx_u}{du}) = \frac{\partial w}{\partial u}((x_u - x_w)\frac{dy_v}{dv} - (y_u - y_w)\frac{dx_v}{dv}) \quad (7)$$

NOTES:

1. *Equation 7 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.*
2. *These equations are telling us that the slopes of the axes (ie $\frac{dx}{du}$, etc) must head off in a direction and rate that keeps the nomogram correctly aligned.*
3. *These equations are the derivatives of equation 3 with respect to u & v respectively. The above derivation is equivalent and provides some geometric insight.*

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?

As an aside, consider equation (7) above for the special case of a nomogram with 3 parallel lines. Without loss of generality, assume x is constant for each axis, so $\frac{dx_u}{du} = 0$ and $\frac{dx_v}{dv} = 0$.
This leaves:

$$\frac{\partial w}{\partial v}(x_w - x_v)\frac{dy_u}{du} = \frac{\partial w}{\partial u}(x_u - x_w)\frac{dy_v}{dv}$$

Now take the natural log of both sides:

$$\ln\frac{\partial w}{\partial v} + \ln\left(x_w - x_v\right) + \ln\left(\frac{dy_u}{du}\right) = \ln\frac{\partial w}{\partial u} + \ln\left(x_u - x_w\right) + \ln\left(\frac{dy_v}{dv}\right)$$

Differentiating with respect to u then v, and recognising that the x values are constant:

$$\frac{\partial^2 \ln\frac{\partial w}{\partial v}}{\partial u \ \partial v} = \frac{\partial^2 \ln\frac{\partial w}{\partial u}}{\partial u \ \partial v} \tag{8}$$

Note that it is possible to reverse this derivation and show that equation (7) is a valid solution to equation (8) above.

Thus we have derived Paul de Saint-Robert's criterion for the feasibility of a 3 parallel line nomogram – see reference [3].

Finding an Initial Estimate for the Nomogram

The optimisation algorithms need a good initial guess to converge on the optimal solution nomogram. Here we consider how to form a simple linear estimate of the nomogram.

Which way are the axes oriented?

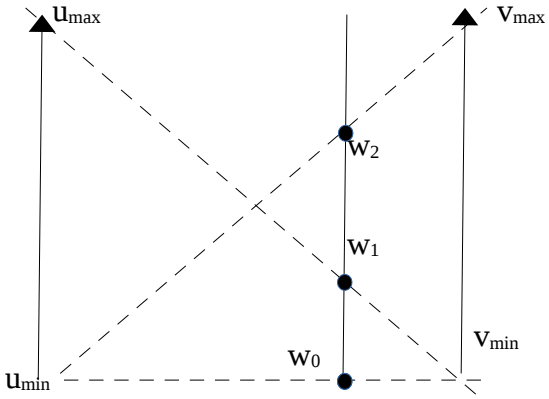
If we assume that the **u** scale increases (varies from u_{min} to u_{max}) as the y coordinate increases, and if we define

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

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

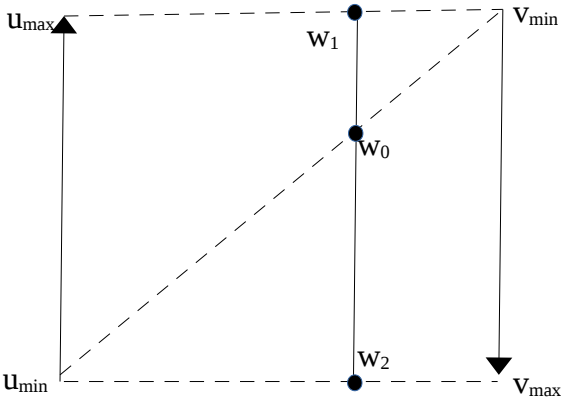
$$w_2 = w(u_{min}, v_{max})$$

then the **w** scale increases upwards if $w_1 > w_0$ and increases downwards if $w_1 < w_0$.
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.

Figure 2: Determining orientation of the axes



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

Figure 3: Determining axes orientation

Note:
The above assumes that the **w** *axis does not intersect the* **v** *axis at* v_{min}*, etc. because the* w_i *points must be distinct. The solution is simply to find another point on the* **v** *(or* **u)** *axis that is suitable.*

Finding Linear Equations for the Axes

Try to approximate the true nomogram with a simple linear one as shown in Figure 4 below.

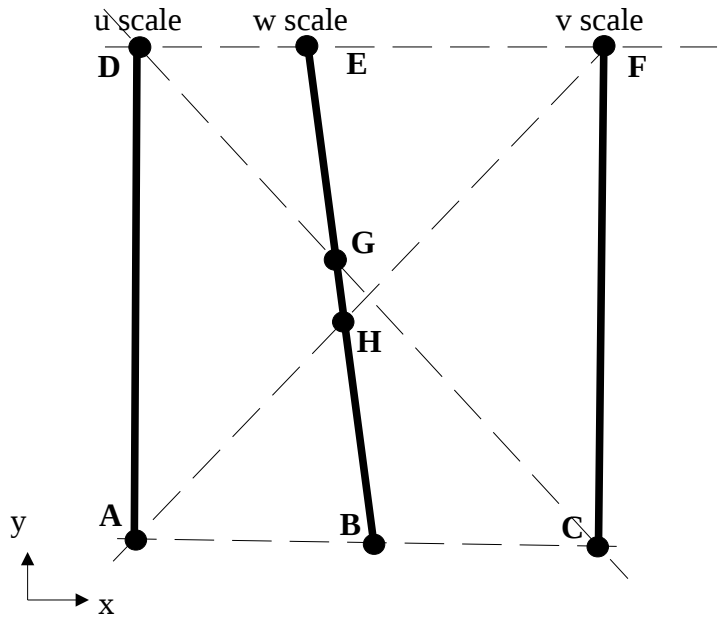


Figure 4: Construction to estimate initial w scale

The initial **u**, **v** & **w** scale lines are respectively the straight lines **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 as dashed lines, 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.*

For the **u** scale line:

$$u_x(u) = 0$$

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

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

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

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

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

We can now write

$$v_x(v) = 1$$

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

We have for the equations of the **w** scale line:

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

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

where c, α_{wx} and α_{wy} are constants that are not yet determined and w_B is the value of w at point **B**.

G is where the **w** scale line intersects the diagonal 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

$$\alpha_{wy} = \frac{1}{w_E - w_B} \tag{9}$$

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

$$c + \alpha_{wx}(w_G - w_B) = 1 - \alpha_{wy}(w_G - w_B) = 1 - \frac{w_G - w_B}{w_E - w_B} = \frac{w_E - w_G}{w_E - w_B} \tag{11}$$

Equation (9) gives α_{wy} and (10) & (11) can be solved to give:

$$\alpha_{wx} = \frac{w_E - w_G - w_H + w_B}{(w_E - w_B)(w_G - w_H)} \tag{12}$$

$$c = \frac{w_H - w_B}{w_E - w_B} - \alpha_{wx}(w_H - w_B) \tag{13}$$

Note:
A more sophisticated approach might be to find a quadratic or cubic approximation for the ***w*** *scale line, using the derivative equations described on page 5 above to find the required coefficients.*

Problems that might occur:

1. 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 $0 \le w_B, w_E \le 1$
2. 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 that are described on page 5 above. Note that the point $w_G = w_H$ must be at coordinates (0.5,0.5), so the functions $x_w(w)$ & $y_w(w)$ must pass through this point. If we

assume that the w axis is a vertical line then possible interpolation equations are:

$$x_w = 0.5$$

$$w = w_B + (4w_G - 3w_B - w_E)y_w + (2w_B - 4w_G + 2w_E)y_w^2$$

Note that the y equation here is the inverse, but the forward equation, ie *y=y(w)* could have a maximum or minimum making *y(w)* not monotonic.

To use the derivative equations, recall equation (7);

$$\frac{\partial w}{\partial v}\left((x_w - x_v)\frac{dy_u}{du} - (y_w - y_v)\frac{dx_u}{du}\right) = \frac{\partial w}{\partial u}\left((x_u - x_w)\frac{dy_v}{dv} - (y_u - y_w)\frac{dx_v}{dv}\right)$$

For the top and bottom isopleths, we know:

1. $x_u = 0$, and $x_v = 1$, so both derivatives are zero
2. all *y* values are identical since these isopleths are horizontal
3. $\frac{dy_u}{du} = \frac{1}{u_{max} - u_{min}}$ and $\frac{dy_v}{dv} = \frac{1}{v_{top} - v_{bottom}}$, since we are choosing linear axes
4. $\frac{\partial w}{\partial u}$ and $\frac{\partial w}{\partial v}$ can be found from the known function *w=w(u,v).*

Insert these known values:

$$\frac{\partial w}{\partial v}(x_w - 1)\frac{1}{u_{max} - u_{min}} - 0 = \frac{\partial w}{\partial u}(0 - x_w)\frac{1}{v_{top} - v_{bottom}} - 0$$

Solve for x_w:

$$x_w = \frac{\frac{\partial w}{\partial v}(v_{top} - v_{bottom})}{\frac{\partial w}{\partial v}(v_{top} - v_{bottom}) + \frac{\partial w}{\partial u}(u_{max} - u_{min})} \quad (14)$$

So now $x_w(w_B)$ *and* $x_w(w_E)$ are found by evaluating equation (14) for the bottom and top isopleths respectively.

TBD: check & explain why these terms must have the same sign, and therefore 0 <= x_w <= 1 here

The alignment 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 the points u_i & v_j, where u_i & v_j refer to the ith and jth Chebyshev nodes on the **u** & **v** scale lines.

If the **u**,**v** & **w** scales are approximate, the point corresponding to $w_{ij}=w(u_i, v_j)$ will not lie exactly on the index line connecting u_i & v_j. The distance of w_{ij} from the line is given by equation (2) on page 4, 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 figure 5 below where a line AB intersects the **u** & **v** scale lines at points (x_u, y_u) & (x_v, y_v)

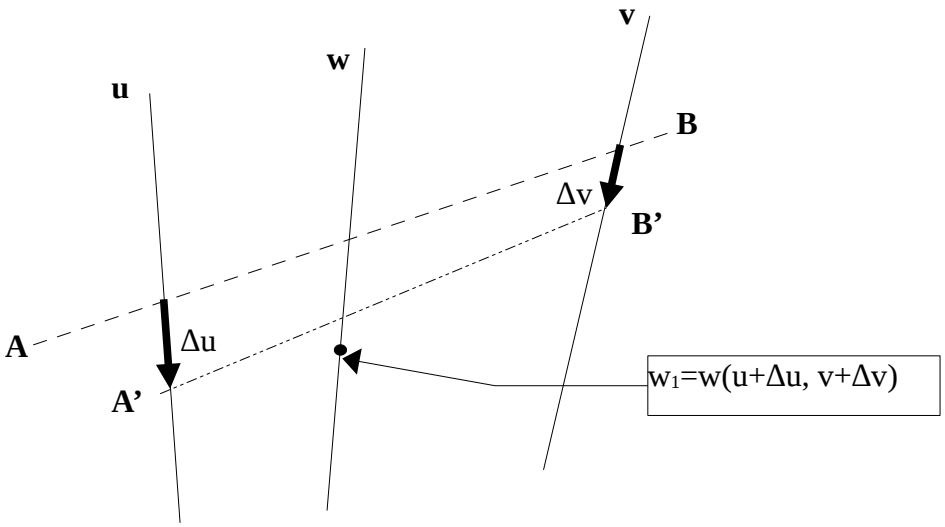


Figure 5: intersection line

Rewriting (2) 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 $w(u+\Delta u, v+\Delta v)$, 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

$$e_1 = e_0 + \frac{\partial e}{\partial u}\Delta u + \frac{\partial e}{\partial v}\Delta v$$

Differentiating the expression for e,

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

$$\frac{\partial e}{\partial u} = \frac{\frac{dx_u}{du}(y_v - y_w) + \frac{\partial w}{\partial u}\left(\frac{dx_w}{dw}(y_u - y_v) - (x_u - x_v)\frac{dy_w}{dw}\right) - (x_v - x_w)\frac{dy_u}{du}}{\sqrt{(x_u - x_v)^2 + (y_u - y_v)^2}} - \frac{e_0\left((x_u - x_v)\frac{dx_u}{du} + (y_u - y_v)\frac{dy_u}{du}\right)}{(x_u - x_v)^2 + (y_u - y_v)^2} \quad (15)$$

where e_0 is the error defined above.

Similarly, for v:

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

$$\frac{\partial e}{\partial v} = \frac{\frac{dx_v}{dv}(y_w - y_u) + \frac{\partial w}{\partial v}\left(\frac{dx_w}{dw}(y_u - y_v) - (x_u - x_v)\frac{dy_w}{dw}\right) - (x_w - x_u)\frac{dy_v}{dv}}{\sqrt{(x_u - x_v)^2 + (y_u - y_v)^2}} + \frac{e_0\left((x_u - x_v)\frac{dx_v}{dv} + (y_u - y_v)\frac{dy_v}{dv}\right)}{(x_u - x_v)^2 + (y_u - y_v)^2} \quad (16)$$

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 5.

This gives an error function for the *(ui,vj)* pair:

$$e_{ij} = e_0^2 + \mu_u\left(\frac{\partial e}{\partial u}\right)^2 + \mu_v\left(\frac{\partial e}{\partial v}\right)^2 \quad (17)$$

, where μ_u and μ_v are some scaling constants.

- Create an error function by summing the squares of all errors over all Chebyshev 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:

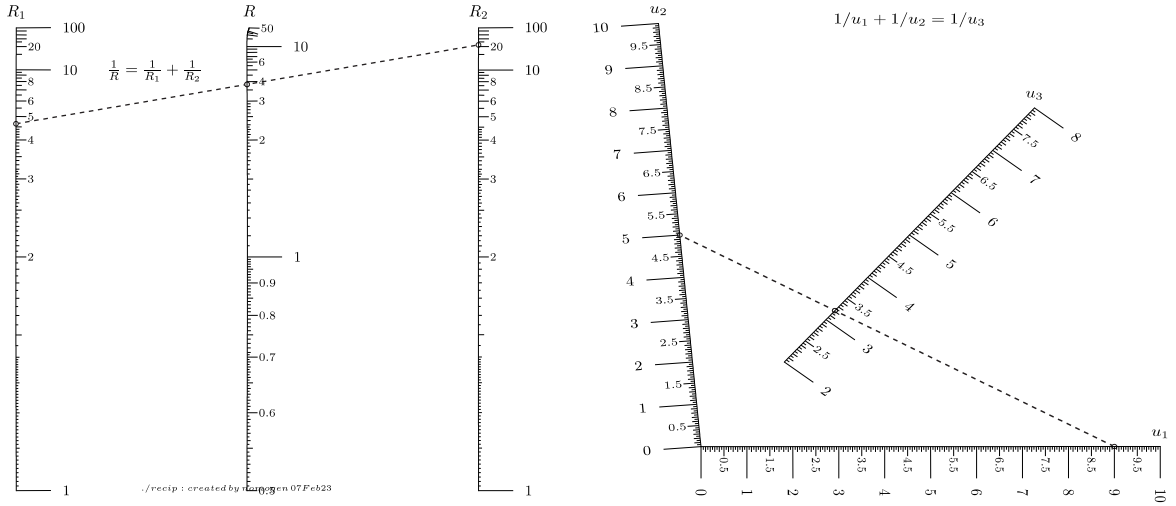
1. The number of error calculations is of the order $O(n^2)$, where n is the degree of the polynomial, but the number of coordinates to fix is of the order O(n). If n is large, pruning the number of error calculations can improve computation speed. *(This doesn't work.)*
2. If the distance between the tic marks of one of the scale lines is non-linear, then the Chebyshev nodes are bunched together at some part in the line, and spread out at another. This could reduce the accuracy of the error calculations. Using log scales helps if the scales are logarithmic.

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 $l_v(v)$ & $l_w(w)$

This is where anamorphosis comes in … TBC …

The measurement error

These 2 nomograms for the equation $1/w = 1/u + 1/v$ have different shapes. Is one better than the other?



Our intuition tells us a good nomogram must fill its area as much as possible to make readings accurate.

If we have a formula

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

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 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. Minimise measurement error. If the scale lines are as tall as possible, the corresponding scale variable takes the maximum 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.

Even with a perfectly constructed nomogram, a perfectly drawn index line is not possible in practice. If we assume some tolerance in the position of the index line, we want the measurement errors of each of the scale variables to be as small as possible.

In figure 6 below, index line AA intersects the **u** scale line:

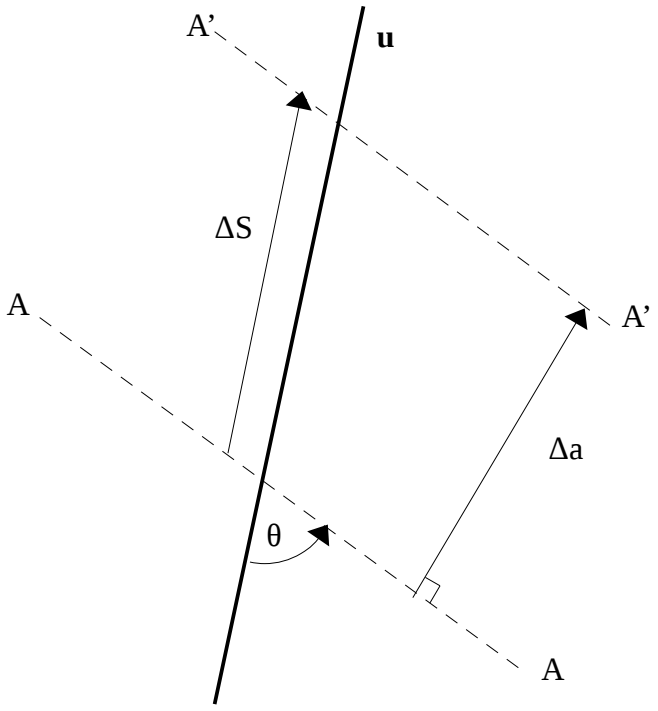


*Figure 6: index lines intersecting the **u** scale line*

The measured line A'A' has an offset error Δa from the true line AA and it 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 these 2 equations for ΔS:

$$\Delta S = \frac{\Delta a}{\sin(\theta)}$$

$$\Delta S = \sqrt{\frac{dx_u}{du}^2 + \frac{dy_u}{du}^2}\,\Delta u$$

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 $\left(\frac{dx_u}{du}, \frac{dy_u}{du}\right)$.
Now use the definition of the vector cross product to write:

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

Combine the above to eliminate ΔS and sin(θ):

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

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

Rearrange this and take the limit:

$$\frac{du}{da} = \frac{\sqrt{(x_v(v)-x_u(u))^2 + (y_v(v)-y_u(u))^2}}{\frac{dx_u}{du}(y_u(u)-y_v(v)) - \frac{dy_u}{du}(x_u(u)-x_v(v))} \qquad (18)$$

Note that the reciprocal of this is the cross product of two vectors, **x´** and **n**, where **x´** is $\left(\frac{dx_u}{du}, \frac{dy_u}{du}\right)$ which is the rate of change of distance in the direction of the **u** scale line. (It's a velocity vector.) 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.

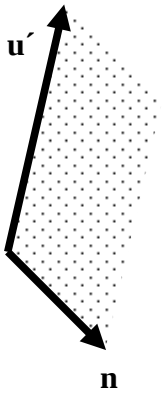


*Figure 7: The cross product of **u'** and **n***

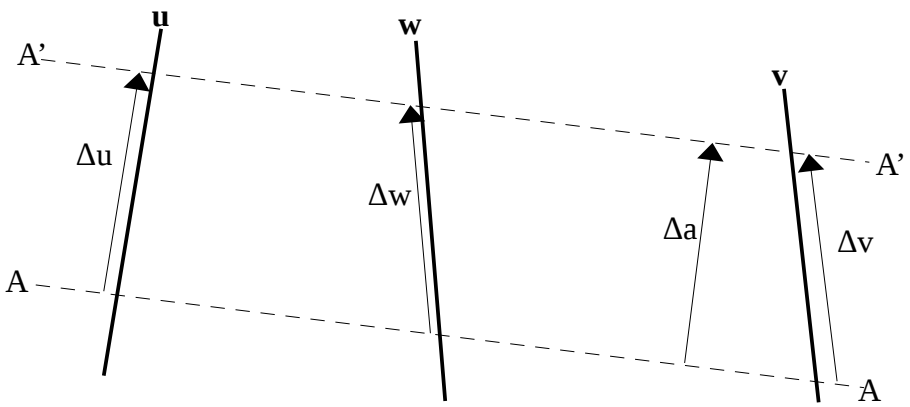
The magnitude of this vector is the area created by the cross product between **x´** and **n** as shown in figure 7 above.

So minimising equation (18) is equivalent to maximising this area (because it's a reciprocal).
This means we want to:

1. make **x´** as long as possible, i.e. stretch the scale lines as much as possible. Remember, **n** is a unit vector, so its length is fixed.
2. make the vectors as close to perpendicular as possible, i.e. make the scale lines as far apart as possible.

This matches our intuition on what makes a better nomogram.

Now consider an index line AA crossing 3 scale lines **u**, **v** & **w**:



A'A' is the measured line, and has an offset error Δa from the ideal line AA.
Δu, Δv & Δw are the measurement errors in the scale variables.

Recalling (18) above,

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

with a similar expressions for ξ_v.

We can write:

$$\Delta w \;=\; \frac{\partial w}{\partial u}\Delta u \;+\; \frac{\partial w}{\partial v}\Delta v$$

Now substitute Δu & Δv:

$$\Delta w \;=\; \Delta a\left(\frac{\partial w}{\partial u}\,\xi_u \;+\; \frac{\partial w}{\partial v}\,\xi_v\right)$$

Taking the limit as Δa approaches zero:

$$\frac{dw}{da} \;=\; \frac{\partial w}{\partial u}\,\xi_u \;+\; \frac{\partial w}{\partial v}\,\xi_v \tag{19}$$

We now we have expressions that can be optimised to generate the best nomogram for the given function.
This is not sufficient, however, because this would create a nomogram of infinite size.
We need a cost function to penalise nomograms that are larger than the specified paper area.

Consider this cost function:

$$c(t) \;=\; \exp(40*(-0.3-t)) \;+\; \exp(40*(t-1.3)) \tag{20}$$

Its value is very close to 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 as shown in 8 below.xxx

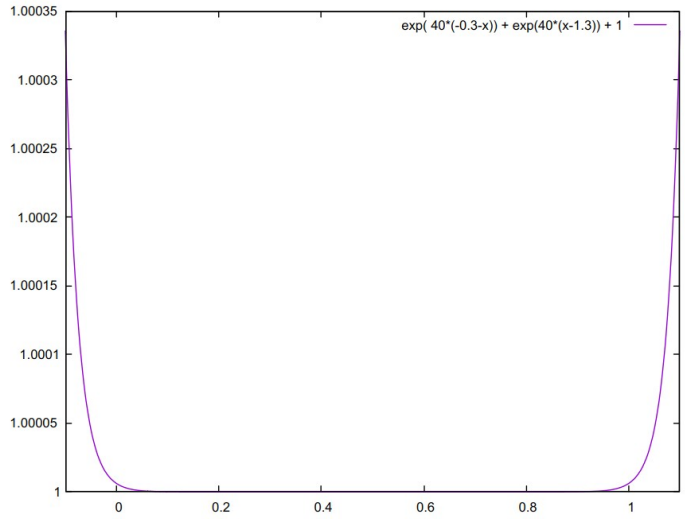


Figure 8: Cost vs position inside unit square

Define the cost of the nomogram due to its shape as

$$\frac{dw}{da}\left(c\left(x_u\right)+c\left(y_u\right)+c\left(x_v\right)+c\left(y_v\right)+1\right) \ ,$$

where $\frac{dw}{da}$ is defined in equation (19} above. The c(x) and c(y) terms are from (20) above and are the costs of the x & y coordinates of the Chebyshev nodes on the **u** & **v** scale lines. We can add this term to the overall cost function for the nomogram given by (17) above

Now we can minimise the shape cost of a nomogram by increasing the area up to the edges. Near the edges, the costs start to increase rapidly beyond the boundary of the unit square. Thus we can achieve a balance where the nomogram has the smallest measurement error for the specified nomogram size.

Further Investigations

Tabulated data

Since the construction method outlined here 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 stored in the tabulated data.

Scales Transformations

Eversham [3], p 203, “Džems-Levi shows that there are three, and only three, projectively distinct scales for x. They are:

$$f_1(x) = e^{ax}$$
$$f_1(x) = ax$$
$$f_1(x) = \tan(ax)$$

Appendix I. Accuracy of Polynomial Evaluation

Horner vs Barycentric interpolation

Here we compare their relative accuracy of polynomial evaluation using both Horner's method and barycentric interpolation.

Recall that Horner's method evaluates the polynomial

$ax^3 + bx^2 + cx + d$

as

$(((ax+b)x + c)x + d$

thus reducing the number of arithmetic operations.

Barycentric interpolation is evaluated with the following polynomial[1]:

$$p_n(x) = \frac{\sum_{j=0}^{n}{}' \frac{(-1)^j f_j}{x - x_j}}{\sum_{j=0}^{n}{}' \frac{(-1)^j}{x - x_j}}$$, where

x_j are the Chebyshev nodes and

f_i are the values of $p_n(x)$ at each of the x_j.

The primes on the summation signify that the first and last terms are halved. See reference [4] for more details.

Figures 9 and 10 show plots of 2 random polynomials, $p_n(x)$ of orders 4 and 11 (the solid lines). The red and blue dots show the accuracy for a sample of *x* values of the polynomials evaluated by barycentric interpolation and Horner's method respectively. The loss of accuracy scale is shown on the right.

The code that generates these plots is contained in the accompanying file `pdemo.py`, with log file `pdemo.log`.

1 Surprising as it seems, this is indeed a polynomial.

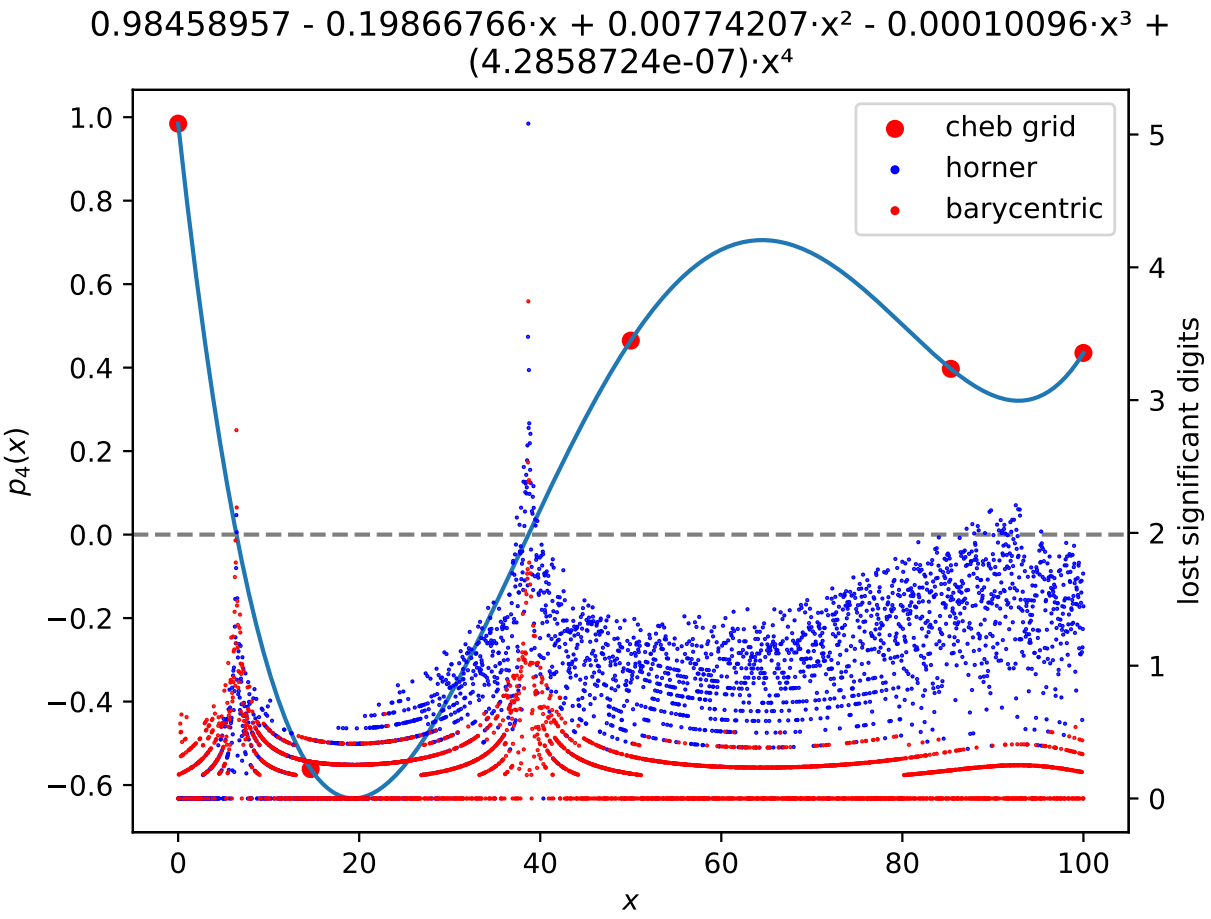


Figure 9: evaluating a 4th order polynomial

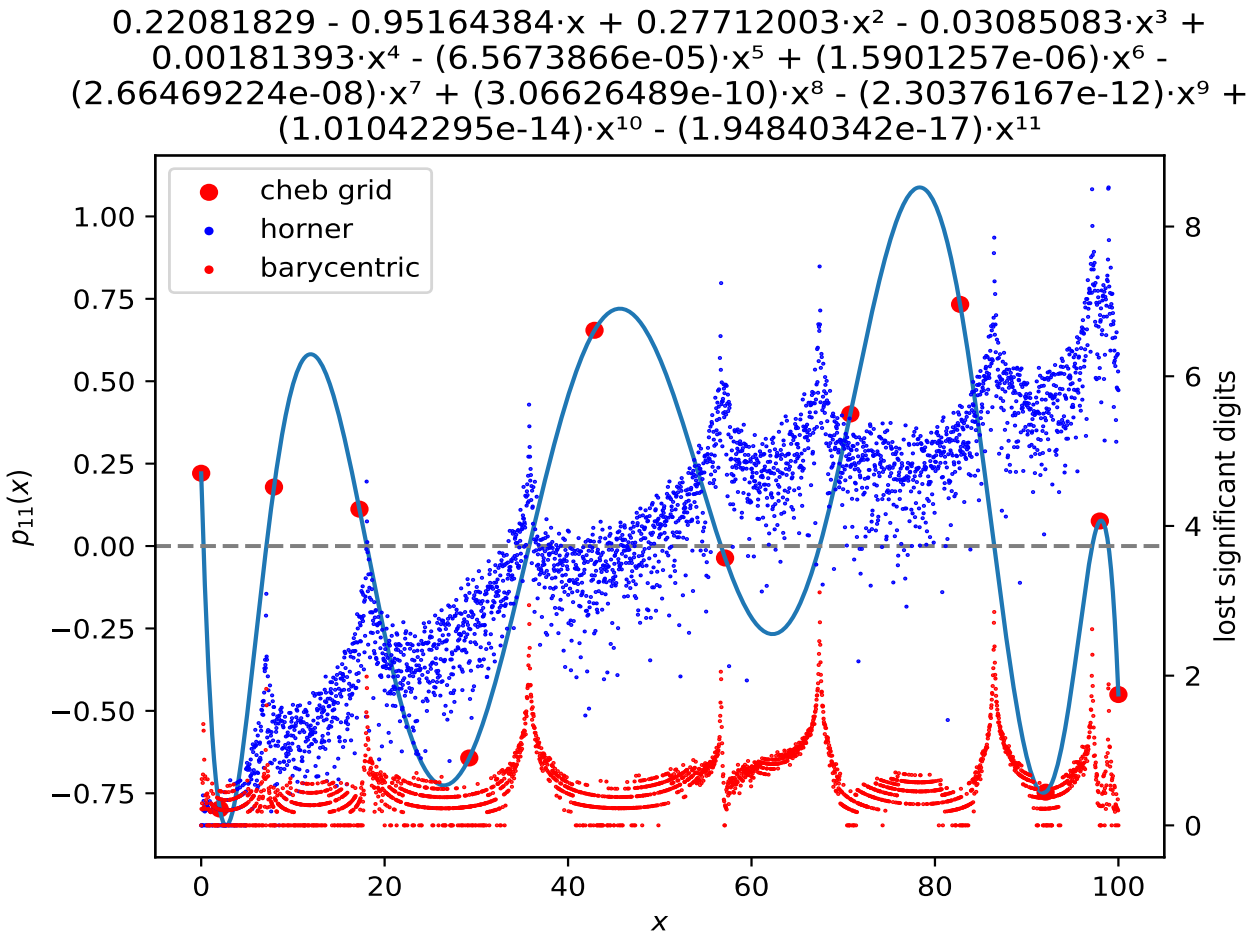


Figure 10: evaluating an 11th order polynomial

We see that for evaluating a polynomial *p(x)*, barycentric interpolation is more robust than Horner's method, especially for higher order polynomials and higher values of x.

This is because Horner's method multiplies intermediate values (with its associated error) by x, so if x is large and there are many multiplications needed, the error must increase.
On the other hand, barycentric interpolation is scale invariant, since the numerator and denominator of each term are of the same order so the error does not increase as *x* becomes large.

Note also that barycentric interpolation has better stability at the roots of *p(x)*.

Runge phenemenon

Runge's function, evenly spaced interpolation → Runge spikes,

Chebyshev approximation, Remez/optimal

plot error for a range of polynomial degrees

Show Chebyshev is very close to best approximation

TBC

The Runge Function, Polynomial Interpolation, and the Cauchy Residual Theorem ref:

https://www.youtube.com/watch?v=VSJqL_pkju8

Appendix II. The Jacobian

The optimisation libraries allow provision of a Jacobian function to improve speed and accuracy.

The Jacobian is a vector that says how much the error changes if any one of the Chebyshev nodes on the axes moves slightly in the x or y direction. It is the derivative of the error function and is a vector that points in the direction of maximum increase (uphill in other words).

It helps the cost optimisation algorithm determine how to steer the nomogram towards a minimum error.

The python libraries find this by adding a small amount to each of Chebyshev nodes in turn, then re-calculate a new error function. This is a time-consuming operation, and it turns out that we can save some time and increase accuracy by calculating the derivative of the error function analytically.

As seen in previous sections above, the error has alignment, slope and shape components.

Error is $\sum_{ij} E_{ij} = \sum_{ij} \left(e_0^2 + \mu_u \left(\frac{\partial e}{\partial u}\right)^2 + \mu_v \left(\frac{\partial e}{\partial v}\right)^2\right)$ evaluated for all combinations of u_i and v_j.

The alignment error

The alignment error, e_0^2 is given by Equation (2) for a particular u_i & v_j:

$$e_0^2 = \frac{\left(x_u y_v + x_w y_u + x_v y_w - x_w y_v - x_u y_w - x_v y_u\right)^2}{\left(x_u - x_v\right)^2 + \left(y_u - y_v\right)^2} = \frac{A^2}{D}$$

This needs to be summed for all combinations of u_i and v_j, but note that changing the position of u_i does not change the position of u_k, $k \neq i$, so

$\frac{\partial u_{xk}}{\partial u_{xi}} = 0$, so most of the terms of e_0^2 are zero.

for the u & v axes

$$\frac{\partial e_0^2}{\partial x_u} = \frac{\partial}{\partial x_u} \frac{\left(x_u y_v + x_w y_u + x_v y_w - x_w y_v - x_u y_w - x_v y_u\right)^2}{\left(x_u - x_v\right)^2 + \left(y_u - y_v\right)^2} = \frac{2A\left(\left(y_v - y_w\right)D - A\left(x_u - x_v\right)\right)}{D^2}$$

, summed over all the v_j nodes

$$\frac{\partial e_0^2}{\partial y_u} = \frac{\partial}{\partial y_u} \frac{\left(x_u y_v + x_w y_u + x_v y_w - x_w y_v - x_u y_w - x_v y_u\right)^2}{\left(x_u - x_v\right)^2 + \left(y_u - y_v\right)^2} = \frac{2A\left(\left(x_w - x_v\right)D - A\left(y_u - y_v\right)\right)}{D^2}$$

, summed over all the v_j nodes

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

, summed over all the u_i nodes

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

, summed over all the u_i nodes

Moving a node on any axis changes the whole curve, except at the positions of the other nodes. For the w axis the calculations generally do not occur at the node positions so we need to sum over all u_i and v_j.

Earlier, we saw the locus of the w-axis was given by the functions $x_w(w)$ & $y_w(w)$. Now the value of w is fixed, and the Chebyshev nodes are variables, so the functions become $x_w(x_1, x_2, \ldots x_n)$ and $y_w(y_1, y_2, \ldots y_n)$. We need to find $\frac{\partial x_w}{\partial x_k}$ and $\frac{\partial y_w}{\partial y_k}$, i.e. find the change in position of the axis when one of the nodes is moved by a small amount.

Firstly, we need to know how much the location of the value w changes when the location of node k changes.

Reference [4], equation (3.3) gives the first form of the barycentric interpolation formula[2]:

$$x_w(w) = l(w)\sum_{j=0}^{n}\frac{\alpha_j}{w - w_j}x_j \text{ , where}$$

$l(w)$ is the product $(w-w_1)(w-w_2)\ldots(w-w_n)$

w_k is the value of w at the k^{th} Chebyshev node,

x_j is the x coordinate of w_j

$$\alpha_j = \prod_{k \neq j}\left(\frac{1}{w_j - w_k}\right)$$

Taking the derivative of $x_w(w)$ wrt x_k, the x coordinate of the k^{th} Chebyshev node:

$$\frac{\partial x_w(w)}{\partial x_k} = l(w)\frac{\alpha_k}{w - w_k}$$ (note: the $(w-w_k)$ factor can be cancelled)

Similarly, for y_w we have

2 Also known as the modified Lagrange interpolation

$$\frac{\partial y_w(w)}{\partial y_k} = l(w)\frac{\alpha_k}{w-w_k} \tag{21}$$

Note that this depends only on the Chebyshev nodes, and is independent of the function $x_w(w)$, so it can be applied to other functions and axes with simple modifications.

Alternatively, reference [4], equation (4.2) gives the second form of the barycentric interpolation formula:

$$x_w(w) = \frac{\sum_{j=0}^{n}\frac{\alpha_j}{w-w_j}x_j}{\sum_{j=0}^{n}\frac{\alpha_j}{w-w_j}}\text{ , where}$$

$$\alpha_j = (-1)^j\delta_j, \quad \delta_j=\begin{cases}½, j=0 \text{ or } j=n\\ 1, \ otherwise\end{cases}$$

Taking the derivative of $x_w(w)$ wrt x_k leaves only the k[th] term on the numerator:

$$\frac{\partial x_w(w)}{\partial x_k} = \frac{\frac{\alpha_k}{w-w_k}}{\sum_{j=0}^{n}\frac{\alpha_j}{w-w_j}}$$

Similarly, we have for the y coordinate:

$$\frac{\partial y_w(w)}{\partial y_k} = \frac{\frac{\alpha_k}{w-w_k}}{\sum_{j=0}^{n}\frac{\alpha_j}{w-w_j}}$$

We can now derive the component of the jacobian for the alignment error for the **w** axis:

$$\frac{\partial e_0^2}{\partial x_k} = \frac{\partial}{\partial x_k}\frac{\left(x_u(y_v-y_w)+x_v(y_w-y_u)+x_w(y_u-y_v)\right)^2}{(x_u-x_v)^2+(y_u-y_v)^2}$$

$$= \frac{\partial}{\partial x_k}\frac{A^2}{D} \quad \text{, summed over the } u_i \text{ \& } v_j \text{ nodes}$$

$$= \frac{2A\frac{\partial x_w(w)}{\partial x_k}(y_u-y_v)}{D}$$

where

x_k is the x coordinate of the k[th] Chebyshev node on the **w** axis

x_u is the x coordinate for the value of u on the **u** axis, and so on for y_u, x_v & y_v

x_w & y_w are the coordinates of $w=w(u,v)$ on the **w** axis

And for the y coordinate:

$$\frac{\partial e_0^2}{\partial y_k} = \frac{2A\frac{\partial y_w(w)}{\partial y_k}(x_v - x_u)}{D}$$

The Slope Error

Equations (15) and (16) on page 15 show how the slope error is calculated for a particular index line. To find the component of the slope error in the jacobian, we need to differentiate this with respect to each of the x_u, y_u, x_v, y_v, x_w, and y_w variables.

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

Rewrite so we can break it into components:

$$E = \frac{A}{D} - \frac{B}{D^2}$$

Differentiate wrt x_k on the **u** axis:

$$\frac{\partial E^2}{\partial x_{uk}} = \frac{\partial}{\partial x_{uk}}\left(\frac{A}{D} - \frac{B}{D^2}\right)^2 = 2\left(\frac{A}{D} - \frac{B}{D^2}\right)\left(\frac{\partial}{\partial x_{uk}}\frac{A}{D} - \frac{\partial}{\partial x_{uk}}\frac{B}{D^2}\right)$$

from which the components are

$$\frac{\partial}{\partial x_{uk}}\frac{A}{D} = \frac{\frac{\partial A}{\partial x_{uk}}D - A\frac{\partial D}{\partial x_{uk}}}{D^2}$$

and

$$\frac{\partial}{\partial x_{uk}}\frac{B}{D^2} = \frac{\frac{\partial B}{\partial x_{uk}}D^2 - 2BD\frac{\partial D}{\partial x_{uk}}}{D^4} = \frac{\frac{\partial B}{\partial x_{uk}}D - 2B\frac{\partial D}{\partial x_{uk}}}{D^3}$$

Now expand and differentiate the expression for A:

$$\frac{\partial A}{\partial x_{uk}} = \frac{\partial}{\partial x_{uk}}\left(\frac{dx_u}{du}(y_v - y_w) + \frac{\partial w}{\partial u}\left(\frac{dx_w}{dw}(y_u - y_v) - (x_u - x_v)\frac{dy_w}{dw}\right) - (x_v - x_w)\frac{dy_u}{du}\right)$$

This equation must be applied to all combinations of the u & v nodes.

Here, we have the following:

- using equation (21),
  $$\frac{\partial}{\partial x_{uk}}\frac{dx_u}{du}(y_v - y_w) = l(u)\frac{\alpha_{uk}}{u - u_k}(y_v - y_w)$$
- $$\frac{\partial}{\partial x_{uk}}(x_{ui} - x_v)\frac{dy_w}{dw} = \begin{cases}\frac{dy_w}{dw}, & i = k \\ 0, & i \neq k\end{cases}$$
- the remaining terms in y_u, *v.* & *w* are constants wrt x_{uk}, so their derivatives are zero.

Now expand and differentiate the expression for B:

$$\begin{aligned}\frac{\partial B}{\partial x_{uk}} &= \frac{\partial}{\partial x_{uk}}\left(e_0\left((x_u - x_v)\frac{dx_u}{du} + (y_u - y_v)\frac{dy_u}{du}\right)\right) \\ &= \frac{\partial e_0}{\partial x_{uk}}\left(\left((x_u - x_v)\frac{dx_u}{du} + (y_u - y_v)\frac{dy_u}{du}\right)\right) + e_0\left(\frac{\partial}{\partial x_{uk}}\left((x_u - x_v)\frac{dx_u}{du}\right)\right)\end{aligned}$$

Recalling e_0 from equation 2

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

TBC

and now for the other equation ...

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

$$+ \frac{e_0\left((x_u - x_v)\frac{dx_v}{dv} + (y_u - y_v)\frac{dy_v}{dv}\right)}{(x_u - x_v)^2 + (y_u - y_v)^2}$$

References

1. PyNomo documentation
2. https://deadreckonings.files.wordpress.com/2009/07/creatingnomogramswithpynomo.pdf
3. The History and Development of Nomography, H. A. Eversham
4. “Barycentric Lagrange Interpolation”, Jean-Paul Berrut & Lloyd N. Trefethen, SIAM REVIEW Vol. 46, No. 3, pp. 501–51
5. Spectral Methods in Matlab, Lloyd N. Trefethen, http://www.comlab.ox.ac.uk/oucl/work/nick.trefethen
6. Introduction to approximating functions with Chebyshev polynomials
   https://www.youtube.com/watch?v=F_43oTnTXiw
   https://blogs.mathworks.com/cleve/2018/12/10/explore-runges-polynomial-interpolation-phenomenon/
   https://demonstrations.wolfram.com/RungesPhenomenon/

