Bygrave Position Line Slide Rule Scales using the PostScript Language

Robin G. Stuart
2026

The Bygrave Position Line Slide Rule

In celestial navigation the measured altitudes of celestial bodies, the Sun, Moon, Planets or stars are used to determine the observer's geographic position. This requires the solution of spherical triangles by means spherical trigonometric identities and is referred to as *solving the navigational triangle* to obtain a *line of position*. To be useful in navigational solutions are required to have an error of no more than about 1 minute of arc (1′) or 1 nautical mile on the Earth's surface. The complexity of the computations and their required accuracy makes them burdensome to perform manually. The labour can be eased by using tables such as Pub.229
https://msi.nga.mil/Publications/SRTMar
or Pub 249
https://msi.nga.mil/Publications/SRTAir
but with three independent input variables they each comprise several hefty volumes.

In 1920 Capt. Leonard Charles Bygrave of the Royal Air Force patented a slide rule design in which the scales formed helices wrapped around coaxial telescoping tubes. This allowed the scales to be made sufficiently long (nearly 8 meters) to obtain the required calculational accuracy. Bygrave's method for solving the navigational triangle involves splitting it into two right triangles and applying Napier's rules. The required operations can then be reduced to a series of multiplications and divisions of the trigonometric functions of the relevant angles. By suitable rearrangement these can all be written in terms of cosines and cotangents, or equivalently secants and tangents, The Bygrave slide rule scales are the logarithms of these functions. Versions of the Bygrave slide rule were produced in Germany and Japan both for marine and aerial use. In German versions, a mechanism that allowed the sliding scale tubes to be locked to together during calculations was introduced.

The physical characteristics of the Bygrave position line slide rules in their various incarnations have been described on detail by Ronald van Riet,
http://www.rechenschieber.org/PositionLineSlideRules.pdf
https://navlist.net/m2x.aspx?i=109429&y=200908
http://navlist.net/img/PositionLineSlideRules.pdf
Additional photographs and dimensions can be found at
https://navlist.net/m2x.aspx?i=109378&y=200908
https://navlist.net/m2x.aspx?i=109429&y=200908
https://navlist.net/m2x.aspx?i=111603
https://navlist.net/HR1-Sliderule-locking-mechanism-questions-Morris-feb-2013-g22308
https://navlist.net/Ruminatons-Bygrave-Hohenrechenschieber-scale-markings-Morris-feb-2013-g22419

A great deal more related discussions, references and information can be found by searching on Navlist or directly by
https://navlist.net/find/Bygrave-from-date-20100101-to-date-20261231.

The exact method of using the Bygrave to solve the navigational triangle will not be repeated here as it has been adequately described in the documents referenced above and in an original manual http://navlist.net/img/106329.bygrave%20manual.pdf.

In 2011 there was renewed interest in the construction and use of Bygrave position line slide rules seen prominently in members of the NavList discussion group https://navlist.net/. Modern versions of the Bygrave slide rule have been produced using versions of the program described here.
https://navlist.net/Bygrave-Rudzinski-apr-2011-g16101
https://navlist.net/MHR1-reproduction-Hasper-mar-2011-g16039
https://navlist.net/MHR1-reproduction-Morris-apr-2011-g16115
https://navlist.net/Working-Model-MHR1-Sliderule-Stuart-jul-2016-g36071
This document describes a PostScript language program that can be used to print precision scales for the construction of Bygrave position line slide rules of various dimensions and forms. The program provides the user with a wide range of input parameters that can used to configure the scales that are produced to their exact requirements.

PostScript Programming Language

PostScript is a powerful programming language ideal for printing graphics. For some time it was the native language of many laser printers which meant that files containing Postscript code could be sent directly to the printer. The files were then interpreted and processed by the printer which performed the necessary computations itself.

To use PostScript nowadays the program Ghostscript is can be obtained from https://ghostscript.com/. Ghostscript can be called from the command line by supplying the appropriate set of options but it is much easier in practice to access it through the program Ghostview a.k.a. GSview. Version 6 or higher can be downloaded from https://gsview.en.softonic.com/. It performs well for basic tasks but lacks much of the functionality and flexibility that was present in earlier versions and in some cases produces incorrect output.

GSView 5.0 can be downloaded from
https://www.ghostgum.com.au/software/gsview.htm. Unfortunately no further development is planned. It has been found that this version hangs when run on a PC with a high screen resolution. For some reason this prevents an initial popup window, saying that an unregistered version GSview is being used, from appearing. The solution is to lower the screen resolution to, for example 1920×1080, and things work fine. Select the required paper size from the Media tab. GSview can be instructed to print the PostScript output or save it in pdf format using File > Convert and selecting the Device: pdfwrite.

Although PostScript is very powerful it is structured like a low-level computer programming language. A gentle introduction to the PostScript language is available in the *PostScript Language Tutorial and Cookbook* from
https://archive.org/details/postscriptlangua0000unse
and other websites such as
http://13thmonkey.org/documentation/Postscript/PostscriptBlueBook.pdf.

The *PostScript Language Reference Manual*
https://www.adobe.com/jp/print/postscript/pdfs/PLRM.pdf,
provides a comprehensive guide.

Numbers to be operated on are placed on a stack, or registers in memory. Operations are performed by taking the required number of entries off the stack and placing the output if any back on the stack. This is very similar to the Reverse Polish Notation (RPN) familiar to HP calculator users. The sequence

```
3 1 sub
```

places 3 and 1 in the top registers in the stack. The operator sub takes the two top entries from the stack, subtracts them placing the result, 2, at the top of the stack for further processing. Assigning values to variables is done with the def operator. Thus

```
/pi 3.1415926535897932 def
```

associates the numerical value with the variable name pi.

In PostScript, dimensions on paper are represented in terms of printer's points. There are 72 points in one inch exactly. The program begins by defining certain useful dimensions, m, cm, mm and inch. The statement

```
/CotCylinderCircumference 174 mm def
```

sets the circumference of the cylinder of the Bygrave cotangent scale to 174mm. This is equivalent to

```
/CotCylinderCircumference 493.23 def
```

expressed using printer's points. Alternatively if the required diameter of the cylinder is known, the corresponding circumference could be set directly using

```
/CotCylinderCircumference 55.4 mm pi mul def
```

Scale Markings

The scale markings Bygrave and other line of position slide rules exist in a wide variety of forms. Moreover some historical slide rules exhibit some quite idiosyncratic features
https://navlist.net/Ruminatons-Bygrave-Hohenrechenschieber-scale-markings-Morris-feb-2013-g22419.
Rather than try to code for all possible scale types and forms the user must explicitly specify the characteristics of each and every tick mark on the scale either in an external file that the Postscript code reads or imbedded in the Postscript code file itself. The required information can be produced by a separate piece of computer code or manually. Each tick's characteristics are specified in order by nine numbers occupying a single line

Degree Minute TickWidth TickLength DotDiameter DotDistance LL LC LR

Here Degree and Minute give the angle that the scale tick represents. TickWidth gives the width of the tick mark in units of TickWidthUnit specified by the user in the Postscript code. Similarly TickLength gives the length of the tick mark in units of TickLengthUnit. Some scale tick marks have a ball or dot above them. DotDiameter gives the diameter of this dot in units of TickWidthUnit and DotDistance gives the distance of nearest edge of the dot from the centre of the scale base line in units of TickLengthUnit. If no dot is required then set the DotDiameter to a value of zero or negative. DotDistance will then be ignored but a value still needs to be provided. The last three numbers specify how the tick is to be labelled and can each be 0 or 1. If LL is 1 then the scale tick will be labelled with the angle's supplement at its left. No such labelling is performed if LL is 0. When LC is 1 the tick is labelled with its angle centred over the tick and a value of is 1 for LR places the label to the right. Typical values for LL LC LR are 1 0 1 which labels the tick with the angle's value to its right and the angle's supplement to its left, 0 1 0 which centres the angle's value over the tick and 0 0 0 leaves it unlabelled.

Tick data is provided for the cotangent and cosine scales separately. If the cotangent tick data is housed in the external file named CotScaleData.txt then the Postscript code should contain the statement

```
/CotScaleDataFile (CotScaleData.txt) (r) file def
```

or if the tick data is imbedded in the Postscript file itself it should contain the statement

```
/CotScaleDataFile currentfile def
```

Similarly for cosine scale tick data the Postscript code should contain the statement

```
/CosScaleDataFile (CosScaleData.txt) (r) file def
```

or

```
/CosScaleDataFile currentfile def
```

When tick data is embedded in the Postscript file it should immediately follow the
readCotScaleData
or
readCosScaleData
statements and end with a blank line. For example

```
readCotScaleData
0 20 3.0 0.1 0.0 0.0 1 0 1
0 21 1.0 0.04 0.0 0.0 0 0 0
0 22 1.0 0.04 0.0 0.0 0 0 0
.....
89 38 1.0 0.04 0.0 0.0 0 0 0
89 39 1.0 0.04 0.0 0.0 0 0 0
89 38 3.0 0.1 0.0 0.0 1 0 1

Some more PS code preceded by a blank line
```

In the application, drawing the scale begins at the point on paper given by CotScaleStartPoint or CosScaleStartPoint. The scale tick data is read in order and drawing of the scale proceeds from left to right and bottom to top. The scale Degree and Minute values should be monotonically increasing or decreasing and can have positive or negative values. This allows the scale directions to be reversed if needed. Minus signs are omitted when labelling the scales. Negative angles in the tick data should have signs on both the degree and minute entries. Thus -88 -30 is used to indicate that the value −88°30′. For the cotangent scale all Degree and Minute values should have the same sign as this scale cannot pass through zero. A scale tick data set of the form

```
-89 -40 3.0 0.1 0.0 0.0 1 0 1
-89 -39 1.0 0.04 0.0 0.0 0 0 0
-89 -38 1.0 0.04 0.0 0.0 0 0 0
.....
89 38 1.0 0.04 0.0 0.0 0 0 0
89 39 1.0 0.04 0.0 0.0 0 0 0
89 40 3.0 0.1 0.0 0.0 1 0 1
```

or

```
89 40 3.0 0.1 0.0 0.0 1 0 1
89 39 1.0 0.04 0.0 0.0 0 0 0
89 38 1.0 0.04 0.0 0.0 0 0 0
.....
-89 -38 1.0 0.04 0.0 0.0 0 0 0
-89 -39 1.0 0.04 0.0 0.0 0 0 0
-89 -40 3.0 0.1 0.0 0.0 1 0 1
```

will result in errors.

The cosine scale experiences no mathematical difficulties in passing through zero and if the signs of Degree and Minute in the first and last lines the scale tick data are different, such as in the examples above, the scale will pass through zero. The labelled scale values will then decrease from left to right approaching zero and increase beyond it. A bidirectional cosine scale of this type could be useful for a flat Bygrave when performing division. Division, or subtraction of logarithms, requires that a number on the cotangent scale be aligned with a number on the cosine scale and reading off the result on the cotangent scale. It may be found to be more convenient instead to align the zero pointer on the cosine scale with the point on the cotangent scale and then read the required result moving from right to left along the cosine scale.

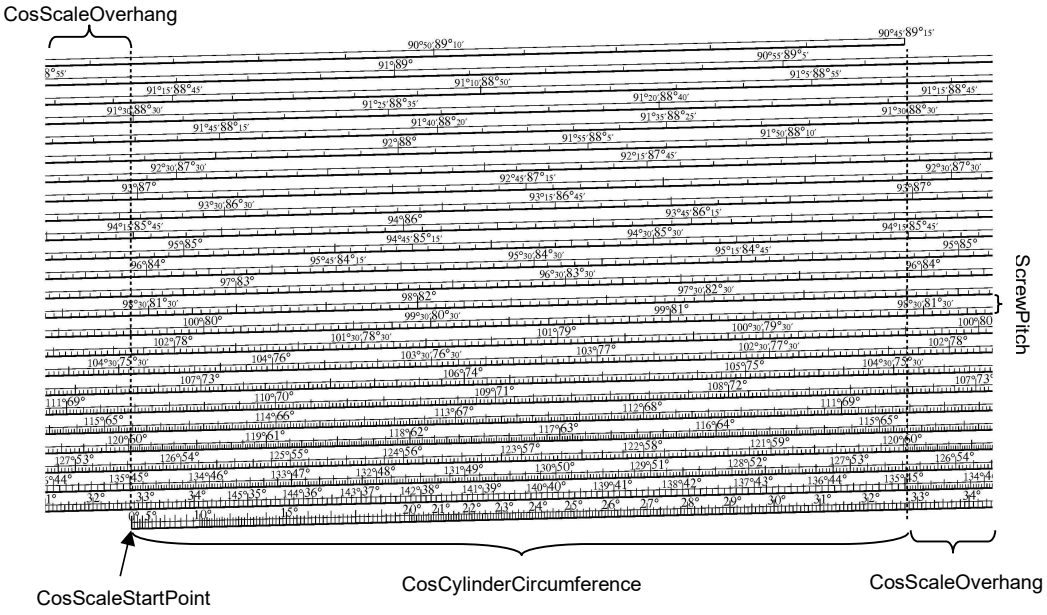


Figure 1. Bygrave Cosine Scale showing the meaning of some of the user parameters. The cotangent scale parameters use similar naming conventions. The scale is drawn on a slant to be wrapped around a cylinder and produce a helical scale. Cutting guide lines are shown. To obtain the required dimensions cuts should be made to the inside edge of these lines.

User Parameters

The following section describes the function of parameters in the code that can be set by the user to change the characteristics of the Bygrave scales that are produced. Changes are made by simply opening the .ps file in a text editor and making the required edits. The user parameters are set by a series of PostScript `def` commands.

TickLineLengthUnit

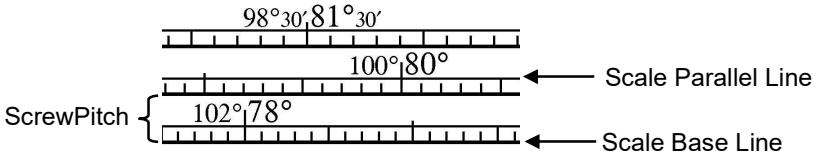
The unit of measure for the lengths of the scale tick marks. The quantities `TickLength` and `DotDistance` in the scale data input file (see *Scale Markings* section) are multiplied by this value to obtain the length on paper. Changing this value affects all scale tick lengths proportionally.

TickLineWidthUnit

The unit of measure for the widths of the scale tick marks. The quantities `TickWidth` and `DotDiameter` in the scale data input file (see *Scale Markings* section) are multiplied by this value to obtain their width on paper. Changing this value affects all scale tick widths proportionally.

DrawParallelScaleLine

If `true` a line is drawn parallel to the scale base line. For flat Bygraves it is usually better to set this to `false` to prevent the scales from becoming too crowded. The cosine scale in Fig. 2 is drawn with DrawParallelScaleLine set to `false`.



ParallelLineWidth
The width of the scale parallel line.

ParallelLineDistance
The centre to centre distance of the scale parallel line.

ScaleBaseLineWidth
The width of the scale base line.

ScaleLabelFontSize
The size of the font to be used to label tick marks on the scales. Marks labelled with the angle to the right and its supplementary angle to the left in a smaller size.

FontBaselineDistance
The distance of the bottom of the numbers that label the scale ticks from the centre of the scale base line. In the case of the cosine scale when the scale ticks are drawn downward this is the distance from the scale base line downwards to the top of the largest character in in the label.

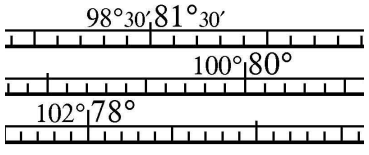
ScaleDigitFont
The name of the font to be used the numbers that label the tick marks on the scales. Possible choices include /Symbol, /Times-Roman, /Helvetica and /Courier.

ScaleSymbolFont
The name of the font to be used for the degree(°) and minute (′) symbols in the scale labels. Possible choices include /Symbol, /Times-Roman, /Helvetica and /Courier. The degree and minute symbols are not very easily visible for some fonts and so the option is provided to allow the use of a different font from ScaleDigitFont. /Symbol is usually a good choice.

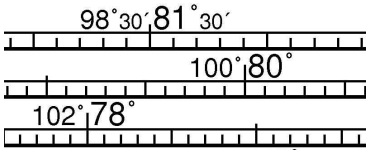
degreechar
The octal number corresponding to the characters used for the degree(°) symbol. This can vary depending on the font. For /Symbol it is (\260) but can be (\312) or (\353) for others.

minutechar
The octal number corresponding to the characters used for the degree(°) symbol. This can vary depending on the font. For /Symbol it is (\242) but can be (\242) or (\302) for others.

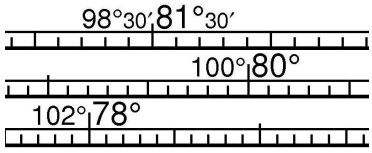
```
/ScaleDigitFont { /Symbol } def
/ScaleSymbolFont { /Symbol } def
/degreechar (\260) def
/minutechar (\242) def
```




```
/ScaleDigitFont { /Helvetica } def
/ScaleSymbolFont { /Helvetica } def
/degreechar (\312) def
/minutechar (\302) def
```



```
/ScaleDigitFont { /Helvetica } def
/ScaleSymbolFont { /Symbol } def
/degreechar (\260) def
/minutechar (\242) def
```



CotCylinderCircumference

The physical circumference of the cotangent scale cylinder, see Fig.1.

ScrewPitch

Specifies the vertical distance between rows of the scales or equivalently the vertical distance that the helical scale advances on one revolution of the cylinder, see Fig.1.

NCotTurns

Specifies the number of complete turns that the cotangent scale makes of the cylinder. This along with CotCylinderCircumference determines the overall physical length of the scale. Note that there is no corresponding input for the cosine scale as its dimensions are completely determined once CotCylinderCircumference and NCotTurns are given.

CotScaleStartPoint

The physical position of the bottom right hand corner of a box bounding the overall cotangent scale relative to the bottom left corner of the page, see Fig.1. Use this to adjust the overall position on the scale on the page. The two values are the *x* (horizontal) and *y* (vertical) coordinates respectively.

CotScaleOverhang

The amount of repeated scale printed at either edge of the cotangent scale. This can be useful when cutting the scales to size or when making a flat Bygrave.

CotScaleColour

The rgb colours used for drawing the cotangent scale, { 0 0 0 } gives black and { 1 0 0 } gives red.

CotScaleLandscape

If true the cotangent scale is printed on the page in landscape format and in portrait if false.

CotScaleDataFile

The name of the file containing the cotangent scale tick data. If it is set to currentfile the data is should be contained the Postscript file itself.

CosCylinderCircumference

The physical circumference of the cosine scale cylinder, see Fig.1. Generally this should be larger than CotCylinderCircumference however when FlatBygrave is true the input value is ignored and CosCylinderCircumference is set equal to CotCylinderCircumference.

CosScaleStartPoint
The physical position of the bottom right hand corner of a box bounding the overall cosine scale relative to the bottom left corner of the page, see Fig.1. Use this to adjust the overall position on the scale on the page. The two values are the *x* (horizontal) and *y* (vertical) coordinates respectively.

CosScaleOverhang
The amount of repeated scale printed at either edge of the cosine scale. This can be useful when cutting the scales to size.

CosScaleColour
The rgb colours used for drawing the cosine scale; {0 0 0} gives black and {1 0 0} gives red.

CosScaleZeroAdjust
When true the cosine scale's starting point will be adjusted so that the zero point falls on the left hand edge of the printed scale. When false the location of the zero point will be determined by the Degree Minute values of the first tick in the cosine tick scale data.

DrawCuttingGuides
If true two dotted vertical lines are drawn with a separation equal to the scale's specified cylinder circumference,
(CotCylinderCircumference or CosCylinderCircumference). Cut to the inside of the cutting guides to obtain the correct dimensions. Cutting guides can be seen in Fig.1.

CosScaleLandscape
If true the cotangent scale is printed on the page in landscape format and in portrait if false.

CosScaleDataFile
The name of the file containing the cosine scale tick data. If it is set to currentfile the data is should be contained the Postscript file itself.

MirrorImage
If true the scales are printed in mirror image. This feature could be used if the scales were to be printed on the back surface of transparent material.

- The following features are included to aid in the construction of flat Bygraves.

FlatBygrave
If false the scales are slanted to allow for wrapping around a cylinder.
If true the scales produced in a manner appropriate for flat Bygrave construction. The scales are printed parallel to the page edges and the cylinder circumferences are made the same by setting CosCylinderCircumference equal to CotCylinderCircumference.

CosScaleTicksDown
If true the tick marks on the cosine scale are printed pointing downward from and labelled below scale baseline. This feature might be helpful in the visual separation of cotangent and cosine scales when using a flat Bygrave, see Fig. 2.

DrawCosScaleZeroPointer

If true a triangular pointer is drawn indicating the location of the zero point on the cosine scale, see Fig. 2.

DrawFrame

If true a frame is drawn around both scales. This may improve the appearance of a flat Bygrave, see Fig. 2.

FrameWidth

Specifies the width of the frame border when DrawFrame is true.

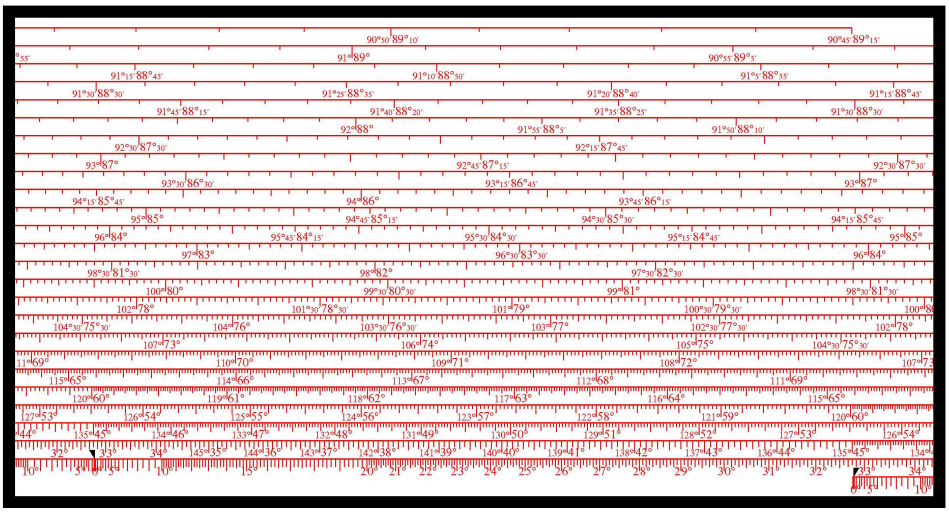


Figure 2. Flat Bygrave Cosine Scale with tick marks and labels drawn below the scale base line. Pointers indicate the location of the scale's zero point. The scale parallel line has been omitted for ease of use.

Examples

The following section lists files used to produce various versions of the Bygrave slide rule along with the output produced by them.

Example 1

These files generate the cotangent and cosine scales for a classical cylindrical Bygrave. The dimensions and scale lengths are chosen to closely match those of the Bygrave Mark II.

Associated files

BygraveMkIIAList.ps	Self-contained Postscript code containing all scale marking information
BygraveMkIIAFile.ps	Postscript code that reads scale marking information from external files
CotScaleMkIIA.txt	File containing the input data for the cotangent scale
CosScaleMkIIA.txt	File containing the input data for the cosine scale
BygraveMkIIA.pdf	Output of the Postscript files converted to pdf format

Example 2

These files generate the cotangent and cosine scales for a classical cylindrical Bygrave. The dimensions and scale lengths are chosen to closely match those of the German Höhenrechenschieber MHR1 made by Dennert & Pape.

Associated files

BygraveMHR1List.ps	Self-contained Postscript code containing all scale marking information
BygraveMHR1File.ps	Postscript code that reads scale marking information from external files
CotScaleMHR1.txt	File containing the input data for the cotangent scale
CosScaleMHR1.txt	File containing the input data for the cosine scale
BygraveMHR1.pdf	Output of the Postscript files converted to pdf format

Example 3

In the recent past Gary LaPook experimented with the construction cylindrical Bygraves using modern materials. This was successful but as a biproduct he also constructed a flat version of the Bygrave in which the scales are used in their planar form without wrapping them around a cylinder. It consists of a cotangent scale printed on paper and a cosine scale printed on transparent film. Calculations are performed by placing the cosine scale on directly top of the cotangent scale. In order to avoid situations where the values on the cosine scale fall beyond the edge of the cotangent scale, the latter is doubled. That is to say that all angles appear twice on the cotangent scale which is physically twice the width of the cosine scale. Full details can be found at the website
https://web.archive.org/web/20220911062426/https://sites.google.com/site/fredienoonan/other-flight-navigation-information/modern-bygrave-slide-rule.
The versions given there use a reduced frequency of scale marking and do not include supplementary angles in the scale labels. This helps to reduce the amount of visual confusion produced by laying the scales on top of one another. The cosine scale is printed in red to more readily distinguish from the cotangent. See also https://navlist.net/m2x.aspx?i=107414

Associated files

BygraveFlatList.ps	Self-contained Postscript code containing all scale marking information
BygraveFlatFile.ps	Postscript code that reads scale marking information from external files
CotScaleFlat.txt	File containing the input data for the cotangent scale
CosScaleFlat.txt	File containing the input data for the cosine scale
BygraveFlat.pdf	Output of the Postscript files converted to pdf format

Example 4

These files produce an alternative configuration for the flat Bygrave. Instead of doubling the cotangent scale, two zero markers are provided on the cosine scale. When one of the zero markers leads to a result falling outside of the cotangent scale, the required value is read from the other. In contrast to the flat Bygrave in example 3, the cotangent and cosine scales are of about the same physical width and can both occupy the full width of the printable area. The scales are longer overall which makes more efficient use of the available real estate on paper and improves the accuracy with which they can be read. A further modification that could be considered is printing the cosine scale with ticks and labels below the scale base line by setting CosScaleTicksDown to true. The cotangent and cosine scales can then be placed base line to base line without their scale tick marks interfering.

Associated files

BygraveAltFlatList.ps	Self-contained Postscript code containing all scale marking information
BygraveAltFlatFile.ps	Postscript code that reads scale marking information from external files
CotScaleAltFlat.txt	File containing the input data for the cotangent scale
CosScaleAltFlat.txt	File containing the input data for the cosine scale
BygraveAltFlat.pdf	Output of the Postscript files converted to pdf format

Example 5

These files produce the alternative configuration for the flat Bygrave similar to that in example 4 but in a format that is designed to be used with a square beam compass or dividers as described in detail in the document FlatBygraveSquareCompassInstructions.pdf. The rows in the scales are numbered. The scales have been configured to be printed in a large format on A3 paper to improve accuracy.

The backstory and original posting of this version of the flat Bygrave can be found at https://navlist.net/Evolution-Flat-Bygrave-Stuart-jul-2021-g50886

Associated files

FlatBygraveSquareCompassLost.ps	Self-contained Postscript code containing all scale marking information
FlatBygraveSquareCompassFile.ps	Postscript code that reads scale marking information from external files
CotScale FlatBygraveSquareCompassFile.txt	File containing the input data for the cotangent scale
CosScale FlatBygraveSquareCompassFile.txt	File containing the input data for the cosine scale
FlatBygraveSquareCompassFile.pdf	Output of the Postscript files converted to pdf
FlatBygraveSquareCompassInstructions.pdf	Instructions on how to use the scales with a square beam compass