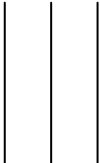
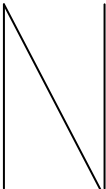


## THE FIRST STEP NOMOGRAMS FOR SUNDIALS

Type 1		<p><math>C = L + R</math></p> <p>works for multiplication if you use logs</p> <p><math>\log C = \log L + \log R</math></p> <p>i.e.</p> <p>logs enable</p> <p><math>C = L * R</math></p>
Type 2		<p><math>C = L / R</math></p> <p>with no logs</p>

## THREE PARALLEL LINES

(Type 1)

$$C = L + R$$

works for

$$\log C = \log L + \log R$$

and thus

if you use logs it works for

$$C = L * R$$

and

$$C = L / R$$

### **NOTE:**

**Chapter 31 of Illustrating Time's Shadow expands upon this topic.**

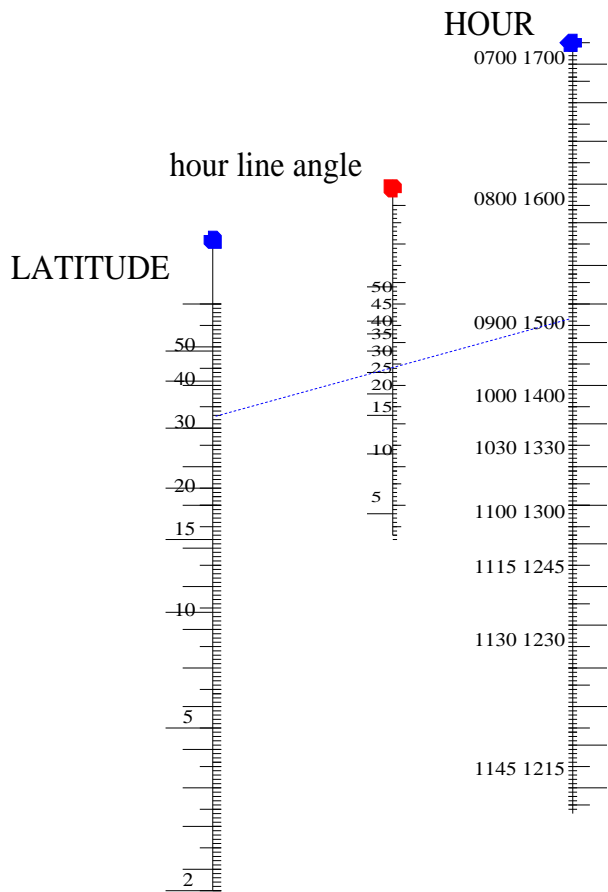
- 2 This document may be freely distributed provided the credit to the web site and author is retained above. Remember that updates may exist on the web site. Chapter 31 of the book Illustrating Time's Shadow expands on this topic.

### The nomogram and the sun dial

Nomograms are graphical solutions to mathematical problems, and in the case of the horizontal dial, we have two variables (latitude and solar hour angle) and one solution (hour line angle).

$$\text{hour line angle} = \text{atan} ( \sin(\text{lat}) * \tan(\text{hour angle}) )$$

The nomogram, popularized in 1880 by Philbert d'Ocagne, has many forms, the simplest is two variables represented by vertical lines left and right, with the solution represented by a vertical line mid way between the two. There are a number of web sites that discuss nomogram theory and design, and some earlier books are also available for online study. The JAVA alternative Python has software for drafting nomograms, and at least one online JAVA tool exists for nomograms as well. In the example below, the horizontal dial nomogram was derived from scratch. In essence the nomogram is a graphical adding machine. Because the horizontal dial formula uses multiplication, logarithms are used which allow multiplication by an additive process. This way adding still works, while solving a multiplication problem.



The left scale is the logarithm of the sine of the latitude, the right is the logarithm of the tangent of the hour from noon (times 15 of course), with the center vertical line being the logarithm of the tangent of the hour line angle on the dial plate compressed by a factor of two. The axes are labeled with latitude, hour, and hour line angle for simplicity, not their logarithmic equivalents. The end result is a nomogram whereby a line can be drawn from the latitude (left) to the hour from noon (right) and the hour line angle read in the middle.

This is not longitude corrected, you add the longitude correction before using the tool.

A sample line for latitude 32, for 0900 or 1500 is shown, and the hour angle in the middle is seen to be close to 27.92 degrees which it should be.

The Hour Line Angle and the Hours From Noon scales get smaller from the bottom as you move up, but then get bigger again. This is because initially the TAN is slowly increasing but the log decreases more rapidly, however later the TAN increases far more than the log decreases.

Illustrating Shadows provides a DeltaCAD macro to draw dial nomograms. The DeltaCAD macro for nomograms is:-

```
nomogram.bas
```

This macro is interesting since logarithms base 10 did not work as documented at the time, so the logs used were base 2.718 or Napierian or natural logs.

The first step in building nomogram is to look at the extremes of the three variables because they must be placed on the nomogram's vertical lines. Their magnitudes should be reasonable and should be selected so that extremes are omitted if the nomogram scales would be impractical.

input log sin (latitude) -1.76 to 0  
 input log tan (hour angle or time) -1.18 to +0.57  
 result log tan (hour line angle) -1.76 to +1.06 (actually can be twice this range)

The practical latitude range for a sundial is between 0.00 to -1.76 as shown by the spreadsheet to the right. By practical, what is meant is the range of the logarithm of the sine of the latitude should fit within a reasonable scale, excessive numbers would not be employed. This is the first input.

LATITUDE	log sin(lat)	= LOG10(SIN(RADIANS(lat)))
0	#NUM!	
1	-1.76	
2	-1.46	
5	-1.06	
10	-0.76	
15	-0.59	
20	-0.47	
25	-0.37	
30	-0.30	
35	-0.24	
40	-0.19	
45	-0.15	
50	-0.12	
55	-0.09	
60	-0.06	
65	-0.04	
70	-0.03	
75	-0.02	
80	-0.01	
85	0.00	
90	0.00	

HOURS FROM NOON	log tan(hr)	=LOG10(TAN(RADIANS(15*hrs)))
0.00	#NUM!	
0.25	-1.18	
0.50	-0.88	
0.50	-0.88	
0.75	-0.70	
1.00	-0.57	
1.25	-0.47	
1.50	-0.38	
1.75	-0.31	
2.00	-0.24	
2.50	-0.12	
3.00	0.00	
3.50	0.12	
4.00	0.24	
5.00	0.57	
6.00	16.21	
7.00	#NUM!	
8.00	#NUM!	

The practical hour range from noon for a sundial is -1.18 to 0 to 0.57 as shown by the spreadsheet to the left. By practical, what is meant is the range of the logarithm of the tangent of the hours from noon should fit within a reasonable scale, excessive numbers would not be employed. This is the second input.

The practical dial plate's hour line angle range for a sundial is -1.76 to 1.06 as shown by the spreadsheet to the right. By practical, what is meant is the range of the logarithm of the tangent of the hour line angle should fit within a reasonable scale, excessive numbers would not be employed. Note that ATAN is not used, but rather TAN is, see the formula development below.

hour LINE angle	log(tan) of resulting hour line angle
	#NUM!
1	-1.76
5	-1.06
10	-0.75
15	-0.57
20	-0.44
25	-0.33
30	-0.24
35	-0.15
40	-0.08
45	0.00
50	0.08
55	0.15
60	0.24
65	0.33
70	0.44
75	0.57
80	0.75
85	1.06
90	16.21

hla = atan( sin(lat) \* tan(hour angle) )  
 thus

tan(hla) = sin(lat) \* tan(hour angle)  
 thus

log tan(hla) = log sin(lat) + log tan(hour angle)

NOTE: This scale is compressed by a factor of two, see next page.

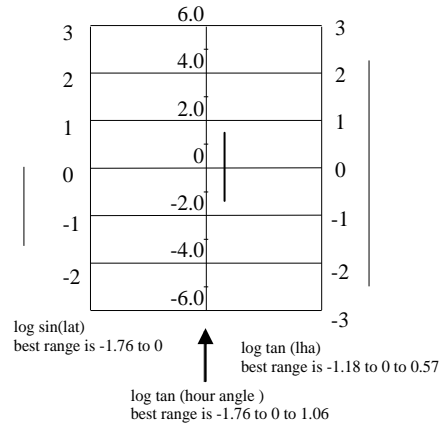
4 This document may be freely distributed provided the credit to the web site and author is retained above. Remember that updates may exist on the web site. Chapter 31 of the book *Illustrating Time's Shadow* expands on this topic.

The next step is to build a set of three parallel lines, equidistant for simplicity, with linear or equally spaced number scales that cover the ranges of the inputs and the output. These ranges once again are:-

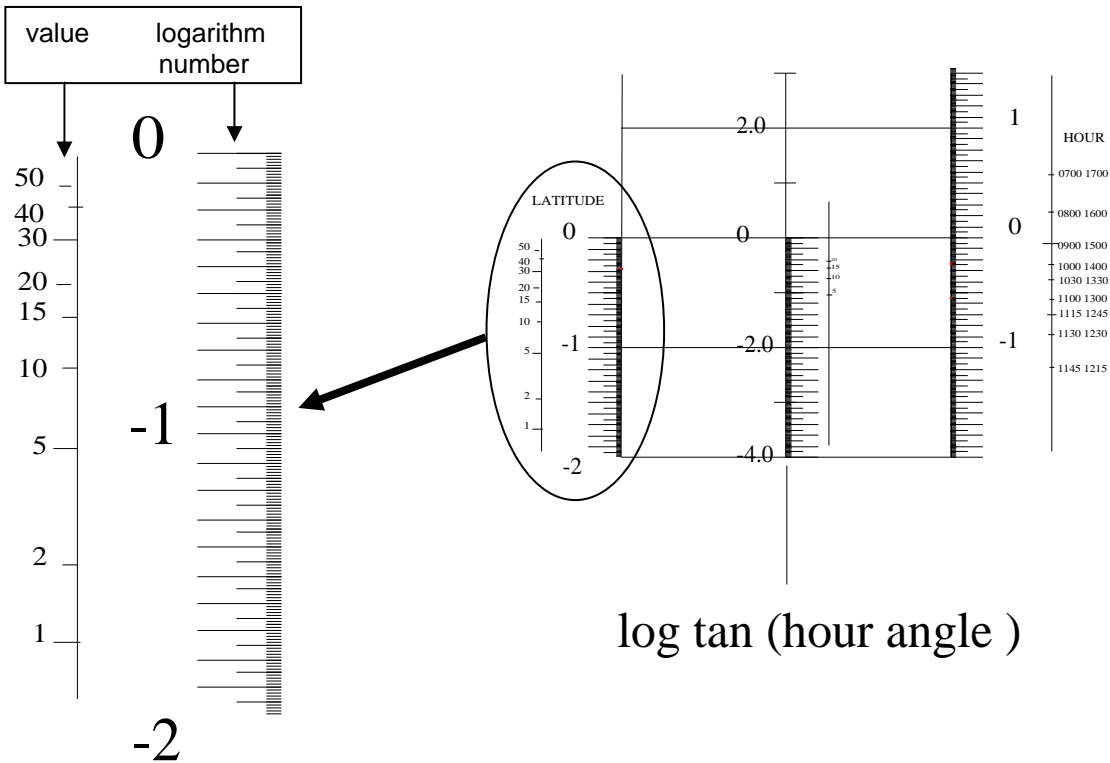
latitude range:	-1.76 to 0	$\log(\text{latitude})$
hour from noon range:	-1.18 to 0 to 0.57	$\log(\text{hour} \cdot 15)$
resulting hour line angle:	-1.76 to 0 to 1.06	$\log(\tan(\text{hour line angle}))$

Thus a nomogram with three lines ranging from -1.76 to 1.06 would be appropriate.

NOTE: The center scale, when equidistant between the other two lines, has its scale compressed in half. In other words the center solution scale can accommodate numbers twice as large as can the other scales.



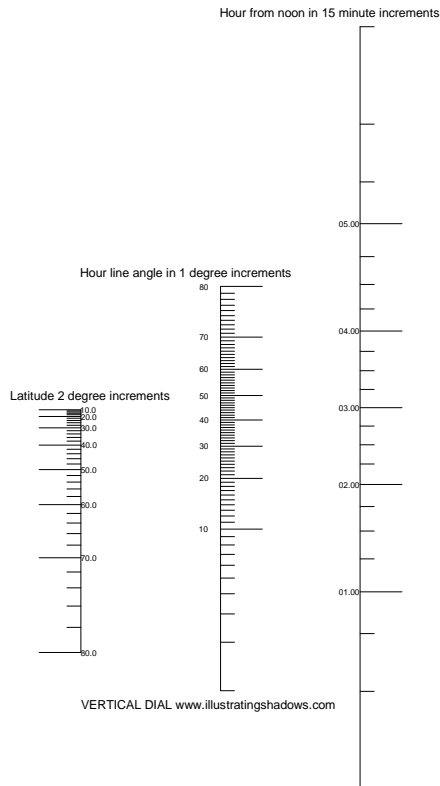
While the scale is linear, we will place the logarithm of our value at the right place, make a marker for that logarithm, and then label it with the original number (input) or final number (result), i.e. "30" for latitude 30, and not the logarithm of the sin(lat).



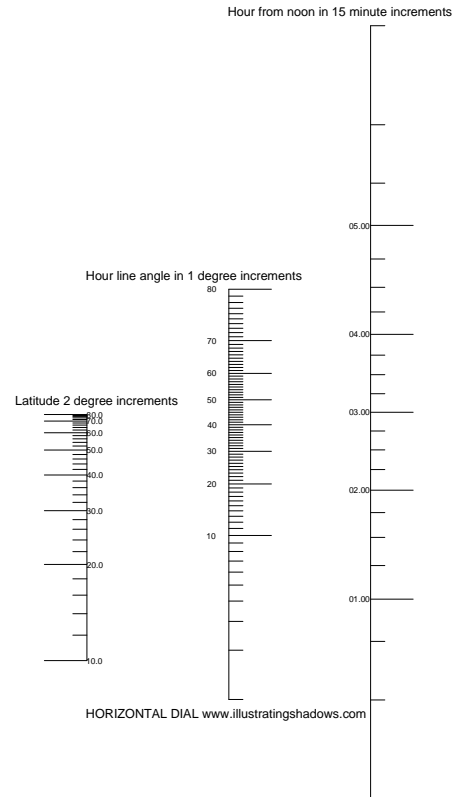
The final nomogram is shown at the start of this section, and several DeltaCAD automated nomograms are shown in the appendices.

DeltaCAD nomograms – nomogram.bas – is interesting since logarithms base 10 did not appear to work, so the logs that did were Napierian or natural logs, base 2.718, so they were used instead.

## DeltaCAD nomograms – vertical dial



## DeltaCAD nomograms – horizontal dial



Useful web sites:

JAVA online:

<http://www.ece.rochester.edu/~jones/NomoDevel/nomogram.htm>

The Art of The Nomogram - read this first

<http://myreckonings.com/wordpress/2008/01/09/the-art-of-nomography-i-geometric-design/>

Graphical and Mechanical Computation

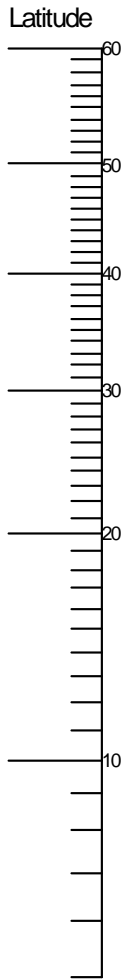
[http://www.myreckonings.com/wordpress/wp-content/uploads/Graphical\\_and\\_Mechanical\\_Computation.pdf](http://www.myreckonings.com/wordpress/wp-content/uploads/Graphical_and_Mechanical_Computation.pdf)

Creating nomograms with pynomo software

<http://www.myreckonings.com/wordpress/>

- 6 This document may be freely distributed provided the credit to the web site and author is retained above. Remember that updates may exist on the web site. Chapter 31 of the book *Illustrating Time's Shadow* expands on this topic.

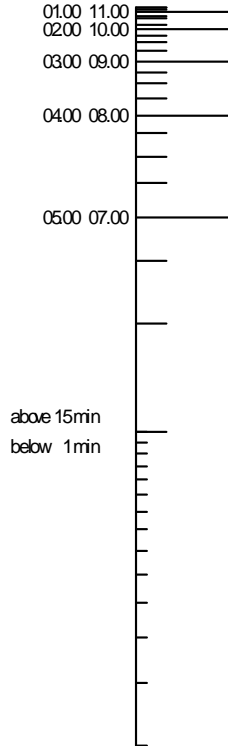
DeltaCAD nomograms – sunrise and sunset by latitude and declination



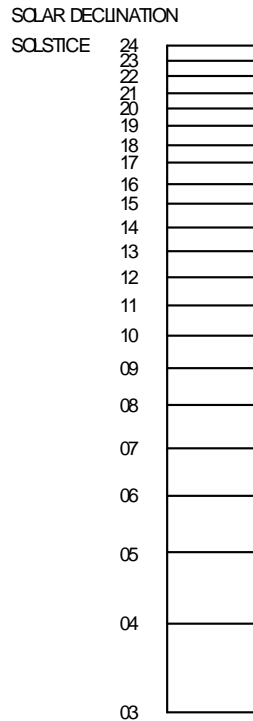
SUNRISE/SET nomogram www.illustratingshadows.com

1. The EOT must be added to correct the times
2. If west of meridian, add 4\*long diff  
If east of meridian, subtract 4\*long diff

Sunrise a.m. hours  
Summer Winter



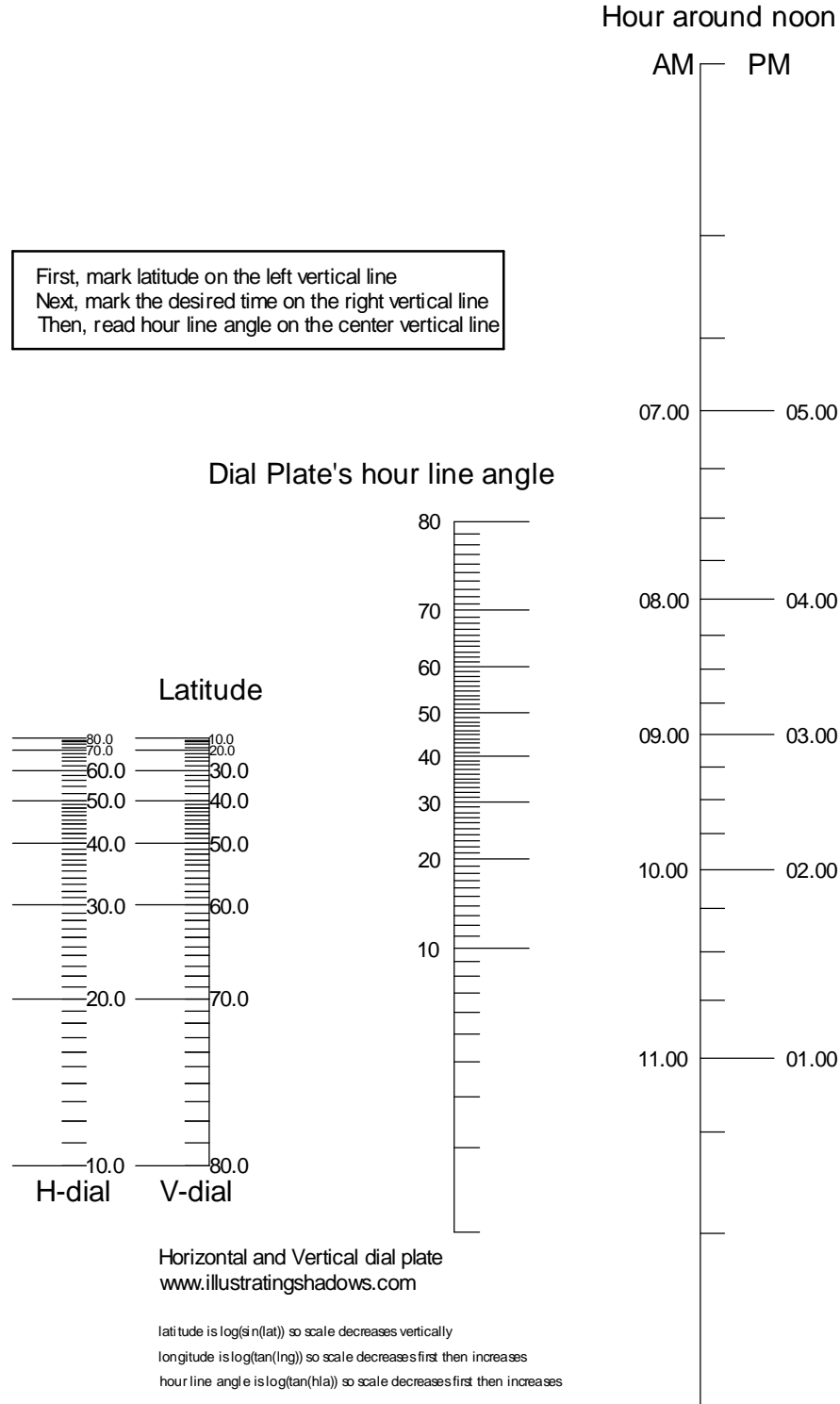
Winter Summer  
Sunset p.m. hours



	EOT mmm	DEC ddd
Jan	1 +3.2	-22.8
	11 +7.4	-21.6
	21 +10.8	-19.7
Feb	1 +13.1	-17.3
	11 +14.1	-14.4
	21 +13.9	-11.0
Mar	1 +12.5	-7.4
	11 +10.2	-3.6
	21 +7.3	0.4
Apr	1 +4.1	4.3
	11 +1.1	8.1
	21 -1.4	11.7
May	1 -3.1	15.0
	11 -3.8	17.8
	21 -3.6	20.1
Jun	1 -2.5	21.9
	11 -0.7	23.0
	21 +1.4	23.4
Jly	1 +3.6	23.2
	11 +5.3	22.4
	21 +6.4	20.9
Aug	1 +6.6	18.8
	11 +5.8	16.2
	21 +3.9	13.1
Sep	1 +1.1	9.6
	11 -2.3	5.9
	21 -6.1	2.0
Oct	1 -9.7	-2.0
	11 -12.9	-5.9
	21 -15.2	-9.6
Nov	1 -16.3	-13.1
	11 -16.1	-16.2
	21 -14.6	-18.8
Dec	1 -11.8	-20.9
	11 -8.1	-22.4
	21 -3.7	-23.2

DeltaCAD nomograms – horizontal and vertical dial

First, mark latitude on the left vertical line  
 Next, mark the desired time on the right vertical line  
 Then, read hour line angle on the center vertical line

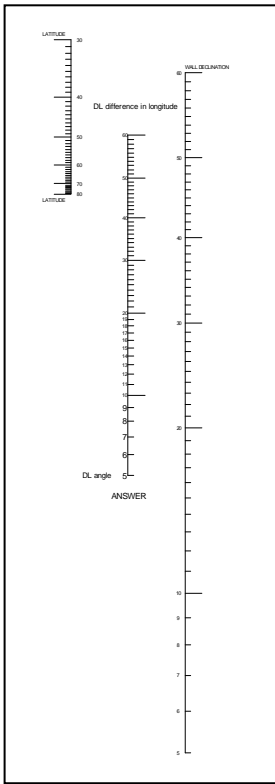


Horizontal and Vertical dial plate  
 www.illustratingshadows.com

latitude is  $\log(\sin(\text{lat}))$  so scale decreases vertically  
 longitude is  $\log(\tan(\text{lng}))$  so scale decreases first then increases  
 hour line angle is  $\log(\tan(\text{hla}))$  so scale decreases first then increases

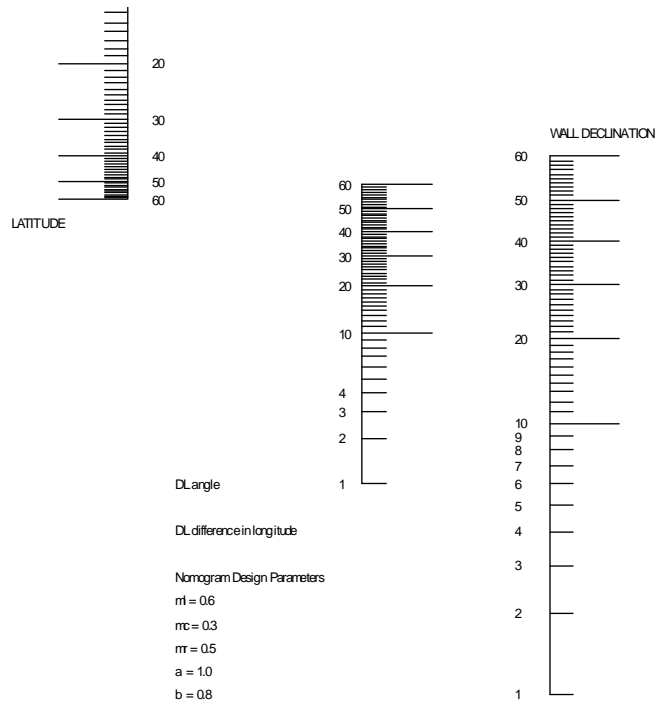
## THE NEXT STEP

### ALTERING NOMOGRAMS TO LOOK BETTER BY VARYING LINE SEPARATION

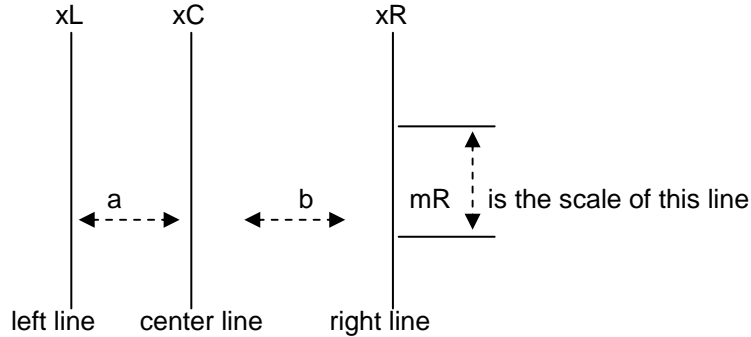


Which is the easier to read, the skinny nomogram on the left, or the better proportioned nomogram on the right.

The next page shows to to vary line spacing



The next step in advanced nomogram work is to change the line separation, and the scale of one of the lines. In other words what do you do if a nomogram looks just plain weird to make it more usable.



The “x” values are positions along the X scale, for example:-

$$\begin{array}{ll}
 x_l = -1 & \\
 x_c = 0 & \text{so } a = \text{Abs}(x_l) - \text{Abs}(x_c) \\
 x_r = 0.8 & \text{so } b = \text{Abs}(x_r) - \text{Abs}(x_c)
 \end{array}$$

and the “mR” is a scale (modulus) of the right line, and between the “a” and “b” separations, and the modulus of the scale, “mR” for example we can come up with a scale (modulus) for the other two lines. For example, assume :-

$$m_r = 0.5$$

in this case the right scale, we derive the modulus (scale) for the left and center lines:-

$$\begin{array}{ll}
 m_l = & \text{Abs}(m_r) * a / b \\
 m_c = & m_l * m_r / (m_l + m_r)
 \end{array}$$

The above logic allows the scales as well as line separation to be manipulated to achieve a more usable nomogram.

To the right are a set of real parameters both entered (xL, xC, and Xr, along with mR) and the derived parameters of mL and mC.

Nomogram Design Parameters (=> means derived)

- m<sub>l</sub> => 0.6
- m<sub>c</sub> => 0.3
- m<sub>r</sub> = 0.5
- a => 1.0
- b => 0.8
- x<sub>l</sub> = -1.0
- x<sub>c</sub> = 0.0
- x<sub>r</sub> = 0.8

THE NEXT STEP  
MOVING FROM THREE PARALLEL LINES  
TO  
AN N OR Z SHAPE  
Type 2  
 $C = L / R$

Sometimes, not much can be done to better align a nomogram that uses the three parallel lines.

To the right is a normal three line nomogram for a meridian or a polar dial. For any hour from local noon (polar) or local 6 am or 6 pm (meridian) it gives the distance from the sub style to the hour line, this is a simple tan of the hour (times 15).

However, for a declination, it also gives in the center line, the distance up the hour line for that declination point. This assumes a style linear height of 1.

Fully usable, but it takes up paper space and has longer solution lines that are desirable.

The three vertical lines works for formula:-

$$C = L + R \quad \text{and this works also for}$$

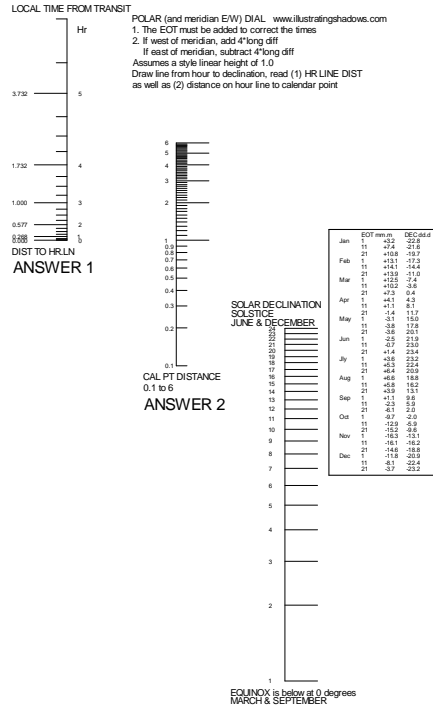
$$C = L * R \quad \text{if logs are used}$$

There is a nomogram chart for  $1/C = 1/L + 1/R$

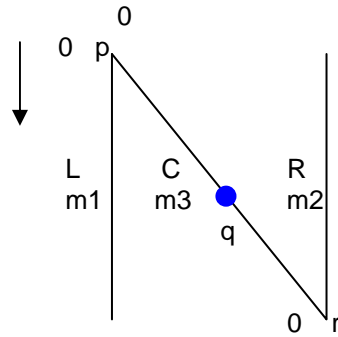
There is a nomogram chart for  $C = L / R$

There is a nomogram chart for  $a / b = c / d$

What may help here is the  $C = L / R$  nomogram, and that is called the N or Z nomogram.



**[ the N or Z chart ] C = L/R kind of nomogram**



pr = is total length of diagonal  
pq = length along the C scale

for equations like  $C = L / R$   
L and R scales are linear, the C scale is not

$$pq \text{ [along the diagonal]} = pr / ( (m2/m1) + v ) \quad \text{where "v" is the value of the data point}$$

**NOTE:** the m2 and m1 are the scale multipliers, not the range of numbers, nor the tan or cos of a value. So for  $dist = \tan(dec) / \cos(hr)$  with  $dec = 0$  to  $24$ ,  $hr = 0$  to  $6$ , neither the 24 nor 6 are used, nor the  $\tan(24)$  nor  $\cos(6 \text{ hours})$ , but rather the scale multipliers if any, in this DeltaCAD macro the left scale was multiplied by 4, the right scale multiplied by 2, so  $m2/m1$  would be  $2/4$  in this case.

**NOTE:** the left side scales have 0 at their vertex, and the diagonal connects the 0 of both vertical scales. The 0 is for the value used in the formula, so if R were  $\cos(\text{value})$  that 0 would be  $\cos(\text{value})$  (i.e. represent 90 degrees) and not "value", see next page.

This document may be freely distributed provided the credit to the web site and author is retained above. Remember that updates may exist on the web site. Chapter 31 of the book *Illustrating Time's Shadow* expands on this topic. 13

This chart is the N or Z nomogram, where the center sloped line "C" represents the left side "C" divided by the right side "R". It does division directly, whereas the last page used logarithms to make the division a subtract and thus use the three vertical line nomogram.

Same end result but better paper usage.

POLAR (and meridian E/W) DIAL www.illustratingshadows.com

1. The EOT must be added to correct the times

2. If west of meridian, add 4\*long diff

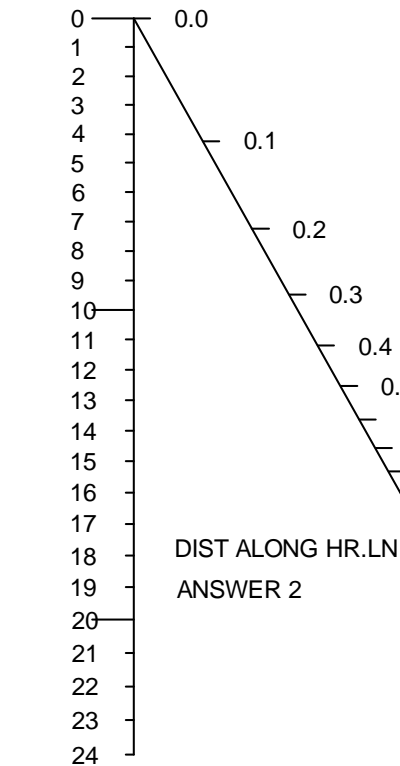
If east of meridian, subtract 4\*long diff

Assumes a style linear height of 1.0

Draw line from hour to declination, read (1) HR LINE DIST

as well as (2) distance on hour line to calendar point

MARCH & SEPTEMBER  
EQUINOX



JUNE & DECEMBER  
SOLSTICE  
SOLAR DECLINATION

		EOT mm.m	DEC dd.d
Jan	1	+3.2	-22.8
	11	+7.4	-21.6
	21	+10.8	-19.7
Feb	1	+13.1	-17.3
	11	+14.1	-14.4
	21	+13.9	-11.0
Mar	1	+12.5	-7.4
	11	+10.2	-3.6
	21	+7.3	0.4
Apr	1	+4.1	4.3
	11	+1.1	8.1
	21	-1.4	11.7
May	1	-3.1	15.0
	11	-3.8	17.8
	21	-3.6	20.1
Jun	1	-2.5	21.9
	11	-0.7	23.0
	21	+1.4	23.4
Jly	1	+3.6	23.2
	11	+5.3	22.4
	21	+6.4	20.9
Aug	1	+6.6	18.8
	11	+5.8	16.2
	21	+3.9	13.1
Sep	1	+1.1	9.6
	11	-2.3	5.9
	21	-6.1	2.0
Oct	1	-9.7	-2.0
	11	-12.9	-5.9
	21	-15.2	-9.6
Nov	1	-16.3	-13.1
	11	-16.1	-16.2
	21	-14.6	-18.8
Dec	1	-11.8	-20.9
	11	-8.1	-22.4
	21	-3.7	-23.2

## CIRCLES

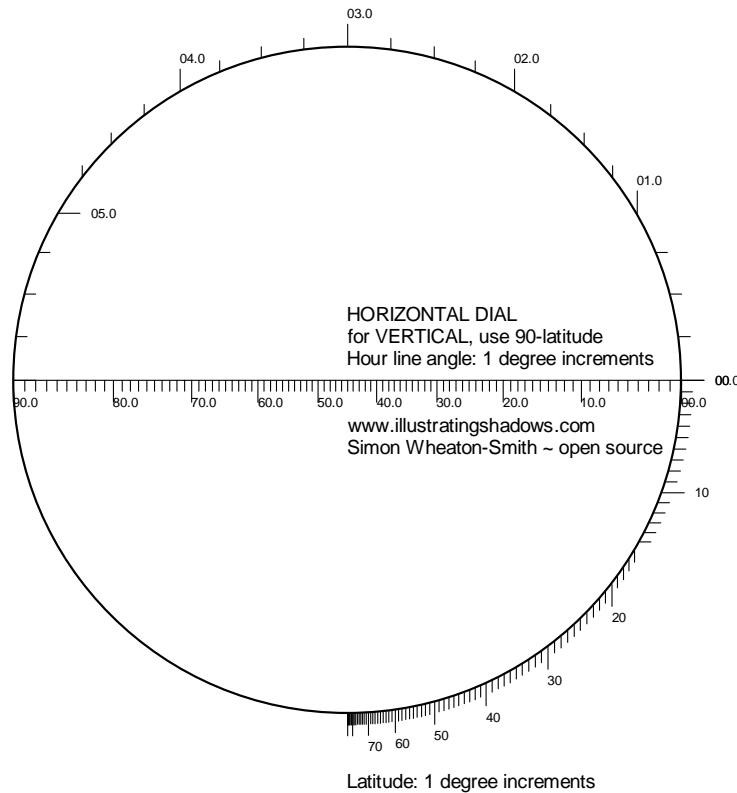
$$C = L * R$$

type 3

directly

see pages 209 and 210 of THE NOMOGRAM by Allcock and Jones

this works well with values that on the three parallel line nomogram would reach to infinity  
this also works well for formulae that are  $C = L * R$  and are performed without needing logs



## CIRCULAR NOMOGRAM FORMULA (using an h-dial as an example)

assuming:  $\tan(hla) = \sin(lat) * \tan(ha)$  then

from xL the x value for the top HOUR ANGLE (HA is 0 to 90, and 1 hour is 15 degrees) circle is:-

$$x = S / ( 1 + \tan^{**2}(ha) )$$

from xL the x value for the lower LATITUDE circle is:-

$$x = S / ( 1 + \sin^{**2}(lat) )$$

from xL the x value for the horizontal HOUR LINE ANGLE line is:-

$$x = S / ( 1 + \tan(hla) )$$

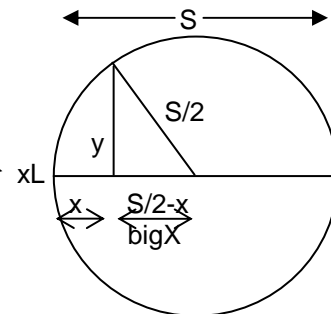
## MAKING A USABLE X AND Y FOR THE CIRCLE

Having an "x" value along the horizontal line is nice, but how do we get a "y" value where a vertical line extended at "x" meets the circle

by definition, the scale "S" is the semi-circle's diameter thus

$bigX = S/2 - x$  where X is the X from the semicircle center and  
where x is the value derived above

xL is the starting reference point for "x"  
for all variables



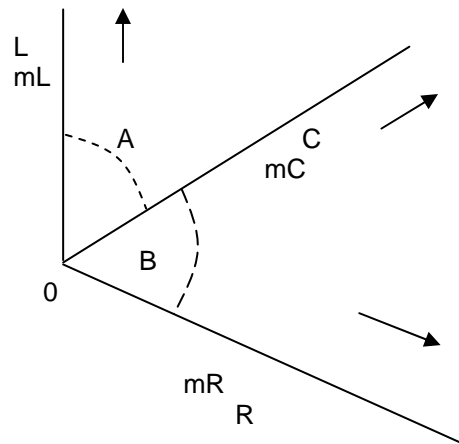
from the semi circle's center we have the x value "bigX", and we have the radius which is "S/2"  
thus by Pythagorus Theorum

$$\begin{aligned} (S/2)^2 &= bigX^2 + y^2 && \text{so} \\ y^2 &= (S/2)^2 - X^2 && \text{but } bigX = S/2 - x, \text{ so} \\ y &= \sqrt{S^2/4 - X^2} \\ y &= \sqrt{S^2/4 - (S/2 - x)^2} && \text{multiply the two parens} \\ y &= \sqrt{S^2/4 - (S^2/4 - S^2x/2 + x^2)} \\ y &= \sqrt{S^2/4 - (S^2/4 - 2*S^2x/2 + x^2)} \\ y &= \sqrt{S^2/4 - (S^2/4 - S^2x + x^2)} \\ y &= \sqrt{S^2x - x^2} \\ y &= \sqrt{S^2x - x^2} \end{aligned}$$

THE NEXT STEP  
OTHER FORMS FOR

$$1 / C = 1 / L + 1 / R$$

Type 4

**[ the concurrent scale chart ]  $1/C = 1/L + 1/R$  kind of nomogram**

for equations:  $1/L + 1/R = 1/C$ , linear examples would be parallel resistances

the zero of all scales meets at the junction of all lines

angle  $A =$  angle  $B$  and you select the angle

you select the scale of the center line  $m_3$ , then you derive  $m_1$  and  $m_2$

$$m_L = m_R = m_C / (2 * \cos A)$$

or you select the scale of either the left or right line  $m_3$ , then you derive the others

$$m_L = m_R$$

$$m_C = m_L * (2 * \cos A)$$

if  $A = B = 60$  then as  $\cos(60) = 0.5$  then  $2 * \cos(60) = 1$ , and all three scales are equal

## EXAMPLE PROGRAM CODE FOR

Type 1:     |||

Type 2:     N or Z

Type 3:     ⊖

Type 4:     V\_\_

Both DeltaCAD and Lazarus support all the nomogram types.

**Type 1 ~ THREE PARALLEL EQUIDISTANT LINE SKELETON CODE FOR DELTACAD**

```

If chc = -1 Then
' *****
' *** -1 is test three vertical lines nomogram ***
' *****

' *** Set X coordinates
' X = -1 latitude
' X = 0 resulting hour line angle
' X = +1 hour angle of the sun
'Dim xl, xr, xc As Single
'Dim x As Single
xl = 0.0
xc = 0.5
xr = 1.0

' set the text color, font, size, etc also
dcSetTextParms dcBLACK,"Ariel","Bold",0,6, 20,0,0
dcSetLineParms dcBLUE,dcSOLID,dcTHIN

' *** input #1 line is on the left
dcSetTextParms dcBLACK,"Ariel","Bold",0,6, 20,0,0
'Dim lat As Single

For lat = 0 to 8 step 1

' save y low and high values
y = 0.2 * lat ' use a scale factor of 0.2 to make sizes
reasonable
If lat = 0 Then
ylo = y
Else
yhi = y
End If

' draw a marker line
dcCreateText xl, y, 0, Format(lat, "00")
dcSetLineParms dcBLACK,dcSOLID,dcTHIN
dcCreateLine xl, y, xl-0.1, y

Next lat
dcCreateLine xl, ylo, xl, yhi
dcSetTextParms dcBLACK,"Ariel","Bold",0,6, 20,0,0
dcCreateText xl, yhi+0.1, 0, "IN #1"

' *** input #2 line is on the right
dcSetTextParms dcBLACK,"Ariel","Bold",0,6, 20,0,0
'Dim hr As Single

For hr = 0 to 8 step 1
' save y low and high values
y = 0.2 * hr ' use a scale factor of 0.2 to make sizes reasonable
If hr = 0 Then
ylo = y
Else
yhi = y
End If

' draw a marker line
' but only say number if 10 multiple
dcCreateText xr, y, 0, Format(hr, "00")

```

20 This document may be freely distributed provided the credit to the web site and author is retained above. Remember that updates may exist on the web site. Chapter 31 of the book *Illustrating Time's Shadow* expands on this topic.

```

    dcCreateLine xr, y, xr-0.1, y

Next hr
dcCreateLine xr, ylo, xr, yhi
dcSetTextParms dcBLACK,"Ariel","Bold",0,6, 20,0,0
dcCreateText xr , yhi+0.1, 0, "IN #2"

' *** sum or solution is in the center
dcSetTextParms dcBLACK,"Ariel","Bold",0,6, 20,0,0
'Dim hla As Single

For hla = 4 to 16 step 1
' save y low and high values
' but mid scale is 0.5 of actual
y = 0.5 * 0.2 * hla ' the 0.5 is also on top of use a scale factor
of 0.2 to make sizes reasonable
If hla = 4 Then
    ylo = y
Else
    yhi = y
End If

' draw a marker line
dcCreateText xc-0.15 , y, 0, Format(hla, "00")
dcSetLineParms dcBLACK,dcSOLID,dcTHIN
dcCreateLine xc, y, xc+0.1, y

Next hr
dcCreateLine xc, ylo, xc, yhi
dcSetTextParms dcBLACK,"Ariel","Bold",0,6, 20,0,0
dcCreateText xc , yhi+0.1, 0, "IN#1 + IN#2"
dcSetTextParms dcBLACK,"Ariel","Bold",0,6, 20,0,0
dcCreateText xc , yhi+0.3, 0, "Starts at 4 only to show how all"
dcCreateText xc , yhi+0.2, 0, "lines always have same base of 0"
End If

```

**Type 2 ~ N OR Z NOMOGRAM SKELETON CODE FOR DELTACAD**

```

If chc = 0 Then
' *****
' *** 0 is test N or Z nomogram ***
' *****

' distance up an hour line to calendar pt = tan (declination) / cos(time )

' *** Set X coordinates
' X = -1
' X = 0
' X = +1
xl = 0.00
xc = 0.5
xr = 1.0

' *** set size multiplier
'Dim s1, s2 As Single
s1 = 0.1
s2 = 0.3

' *** set modulus
'Dim m1, m2 As Single
m1 = 20
m2 = 10

' set the text color, font, size, etc also
dcSetTextParms dcBLACK,"Ariel","Bold",0,4, 20,0,0
dcSetLineParms dcBLACK,dcSOLID,dcTHIN

' *** SOLAR DECLINATION of the dial plate's hour lines on the left
dcSetTextParms dcBLACK,"Ariel","Bold",0,4, 20,0,0

' winter
For decl = 0 to m1 step 1
' save y low and high values
' make scale *** m2 = 2.5 ***
'y = - s1 * Tan(decl*2*3.1416/360)
y = - s1 * decl
If decl = 0 Then
ylo = y
Else
yhi = y
End If

' draw a marker line
' but only say number if 10 multiple
If (decl/10 - Int(decl/10)) = 0 Then
dcCreateText xl-0.15 , y, 0, Format( decl, "#0")
dcSetLineParms dcBLACK,dcSOLID,dcTHIN
dcCreateLine xl, y, xl-0.1, y
Else
dcCreateText xl-0.15 , y, 0, Format( decl, "#0")
dcCreateLine xl, y, xl-0.02, y
End If
Next hr
' *** SOLAR DECLINATION of the dial plate's hour lines on the right -
continued
dcSetTextParms dcBLACK,"Ariel","Bold",0,4, 20,0,0

```

22 This document may be freely distributed provided the credit to the web site and author is retained above. Remember that updates may exist on the web site. Chapter 31 of the book *Illustrating Time's Shadow* expands on this topic.

```

dcCreateText xl-0.3 , yhi-0.06 , 0, "20 range"
dcCreateText xl-0.3 , ylo+0.18 , 0, "0 range"
' *** SOLAR DECLINATION of the dial plate's hour lines on the right -
continued
dcSetTextParms dcBLACK,"Ariel","Bold",0,4, 20,0,0

dcCreateLine xl, ylo, xl, yhi
dcSetTextParms dcBLACK,"Ariel","Bold",0,4, 20,0,0

' save the line bottom for the N or Z line
'Dim xxxx,yyyy As Single
'Dim yyyyy As Single
xxxx = xl
yyyy = ylo
yyyyy=yhi

' *** a value line is on the right of "X" which becomes 1/X because
' the nomogram implies the 1/X
dcSetTextParms dcBLACK,"Ariel","Bold",0,4, 20,0,0
For hr = 0 to m2 step 0.25
' save y low and high values
y = yyyyy + s2 * hr
If hr = 0 Then
ylo = y
Else
yhi = y
End If

' draw a marker line
' but only say number if 10 multiple
If (hr - Int(hr)) = 0 Then
dcCreateText xr+0.1 , y, 0, Format(hr, "#0")
dcSetLineParms dcBLACK,dcSOLID,dcTHIN
dcCreateLine xr, y, xr+.1, y
Else
dcCreateLine xr, y, xr+.02, y
End If
Next hr
dcCreateLine xr, ylo, xr, yhi
dcSetTextParms dcBLACK,"Ariel","Bold",0,4, 20,0,0
dcSetTextParms dcBLACK,"Ariel","Bold",0,4, 20,0,0
dcCreateText xr+0.1 , yhi-0.1, 0, "0 to 10"

' save the line bottom for the N or Z line
'Dim xxx,yyy As Single
xxx = xr
yyy = ylo

' *** DISTANCE ON CENTER SCALE IS LEFT/RIGHT
'
'Dim pr As Single
'Dim pq As Single
' pr is the length of the diagonal
pr = Sqr( (xxxx-xxx)*(xxxx-xxx) + (yyyy-yyy)*(yyyy-yyy) )

dcSetTextParms dcBLUE,"Ariel","Bold",0,4, 20,0,0
dcSetLineParms dcBLUE,dcSOLID,dcTHIN

```

```

dcCreateLine xxx,yyy,xxxx,yyyy

'Dim decdist As Single
For decpt = 0 to 20 step 1
  'decpt = decdist/10
  ' get point along the diagonal ~ Z = L f3(w) / [(m2/m1) + f3(w)]
  pq = ( pr * decpt ) / ((s2/s1) + decpt)

  ' need an x and a y for that distance
  x = xxxx + ((xxx-xxxx) * (pq/pr))
  y = yyyy + ((yyy-yyyy) * (pq/pr))

  ' draw a marker line but only say number if integer
  If (decpt/10 - Int(decpt/10)) = 0 Then
    dcCreateText x+0.1 , y, 0, Format(decpt , "#0")
    dcCreateLine x, y, x+0.2, y
  ElseIf decpt <10 Then
    dcCreateLine x, y, x+0.1, y
    dcCreateText x+0.1 , y, 0, Format(decpt , "0.0")
  Else
    dcCreateLine x, y, x+0.1, y
  End If
Next decpt
dcSetTextParms dcBLACK,"Ariel","Bold",0,4, 20,0,0
dcSetLineParms dcBLACK,dcSOLID,dcTHIN
dcCreateText xc-0.4 , ylo+0.8, 0, "LEFT/RIGHT"

End If

```



```

'      *..+....C.....*          length "*" to "*" is S
'      <x>                    S/2 is radius and hypotenuse
'      < X >
'
' from the semi circle's center "C" we have the x value "X"
' and we have the radius which is "S/2"
'
' thus by Pythagorus Theorum
'
' (S/2)*(S/2) = X*X + Y*Y
'
' or
' Y*Y = (S/2)*(S/2) - X*X
'
' but X = S/2 - x
'
' Y = sqrt ( S*S/4 - X*X )
'
' = sqrt ( S*S/4 - (S/2 - x)*(S/2 - x) )    now multiply the two parenthesis
'
' = sqrt ( S*S/4 - (S*S/4 - S*x/2 - x*S/2 + x*x) )
'
' = sqrt ( S*S/4 - (S*S/4 - 2*S*x/2      + x*x) )
'
' = sqrt ( S*S/4 - (S*S/4 - S*x          + x*x) )
'
' = sqrt ( S*S/4 - S*S/4 + S*x          - x*x) )
'
' = sqrt (                S*x          - x*x) )
'
' = sqrt ( S*x - x*x )

' draw the two semi circles and the line
dcCreateCircle  xL+S/2 , y , S/2
dcCreateLine    xL      , 0 , xL+S, 0

' set the text color, font, size, etc also
dcSetTextParms dcBLACK,"Ariel","Bold",0,4, 20,0,0
dcSetLineParms dcBLUE,dcSOLID,dcTHIN

' *** L is on the bottom
'
' x = S / ( 1 + L * L )
'
' and
'
'      X*X means the square of the hypotenuse
' Y = sqrt ( S*x - x*x )
'
'      x here is S/2 - x
'
dcSetTextParms dcBLACK,"Ariel","Bold",0,4, 20,0,0
Dim L As Single

For L = 0 To 10 Step 1

' get x from xL distance
x = S / ( 1 + (L*L) )

' bigX is x from circle center and is used for Y calculation
bigX = S/2 - Abs(x)          ' x may have been L or R of center
Y = Sqr ( s*x - x*x )

' get x and y for screen location ~ i.e. adjust by yDisp and xL
x = xL + x
y = yDisp + y

' draw a marker line and number

```

```

    If Abs(y) < S/4 Then
        dcCreateText x+0.15 , -y , 0, Format(L, "00")
        dcSetLineParms dcBLACK,dcSOLID,dcTHIN
        dcCreateLine x , -y, x+0.1 , -y
    End If
    If Abs(y) >= S/4 Then
        dcCreateText x , -y-0.15 , 0, Format(L, "00")
        dcSetLineParms dcBLACK,dcSOLID,dcTHIN
        dcCreateLine x , -y, x , -y-0.1
    End If
Next lat
dcSetTextParms dcBLACK,"Ariel","Bold",0,6, 20,0,0
dcCreateText xL + S/2 , -S/2-0.3, 0, "L: bottom: 1 to 10"

' *** R is on the top
'
' x = S / ( 1 + R*R )
'
' and
'           X*X means the square of the hypotenuse
' Y = sqrt ( S*x - x*x )
'           x here is S/2 - x
'
dcSetTextParms dcBLACK,"Ariel","Bold",0,4, 20,0,0
Dim R As Single

For R = 0 To 5 Step 1

    ' get x from xL distance (ha here is 4 times hour thus ha*0.25)
    ' (and of course 1 hour is 15 degrees)
    x = S / ( 1 + (R*R))

    ' bigX is x from circle center and is used for Y calculation
    bigX = S/2 - Abs(x) ' x may have been L or R of center
    Y = Sqr ( s*x - x*x )

    ' get x and y for screen location ~ i.e. adjust by yDisp and xL
    x = xL + x
    y = yDisp + y

    ' draw a marker line but only say number if 10 multiple
    If Abs(y) < S/4 Then
        dcCreateText x+0.15 , y , 0, Format(R, "00")
        dcSetLineParms dcBLACK,dcSOLID,dcTHIN
        dcCreateLine x , y, x+0.1 , y
    End If
    If Abs(y) >= S/4 Then
        dcCreateText x , y+0.15, 0, Format(R, "00")
        dcSetLineParms dcBLACK,dcSOLID,dcTHIN
        dcCreateLine x , y, x , y+0.1
    End If
Next lat
dcSetTextParms dcBLACK,"Ariel","Bold",0,6, 20,0,0
dcCreateText xL + S/2 , S/2+0.3, 0, "R: top: 1 to 5"

' *** C is the answer in the center line
'
' x = S / ( 1 + C )
'
'           not tan*tan for the horizontal line
'
dcSetTextParms dcBLACK,"Ariel","Bold",0,4, 20,0,0
Dim C As Single

For C = 0 To 50 Step 1

    ' get x from xL distance
    x = S / ( 1 + C )

```

```
Y = 0

' get x and y for screen location ~ i.e. adjust by yDisp and xL
x = xL + x
y = yDisp + y

' draw a marker line
' but only say number if 10 multiple
If (hla/10 - Int(hla/10)) = 0 Then
    dcCreateText x , y-0.1, 0, Format(C, "00")
    dcSetLineParms dcBLACK,dcSOLID,dcTHIN
    dcCreateLine x , y , x , -y-0.1
Else
    dcCreateLine x , y , x , -y-0.05
End If

Next hla
dcSetTextParms dcBLACK,"Ariel","Bold",0,6, 20,0,0
dcCreateText xL + S/2 , 0+0.1, 0, "C=L*R: center: 1 to 50"

dcCreateText xL + S/2 , 0+0.3, 0, "CIRCULAR NOMOGRAM"
dcCreateText xL + S/2 , 0+0.2, 0, "for C = L * R without logs"
dcCreateText xL + S/2 , 0-0.2, 0, "www.illustratingshadows.com"
dcCreateText xL + S/2 , 0-0.3, 0, "Simon Wheaton-Smith ~ open source"

'End If
```

**Type 4 ~ ANGULAR TYPE 4 NOMOGRAM SKELETON CODE IN LAZARUS**

```

{ ***** }
{ ***      S T A R T      O F      T E S T      T Y P E      3      *** }
{ ***** }

procedure TForm1.testType3Click(Sender: TObject);
{
  KEY POINTS TO REMEMBER:
  0,0   top left
  y increases positively down

  DOCUMENTATION:
  http://delphi.about.com/library/bluc/text/uc052102a.htm   [ is page 1 ]
  http://delphi.about.com/library/bluc/text/uc052102b.htm   [ is page 2 ]
}

var { must declare FOR variables in the section in which they are used }

    { x and y working values }
    x,y,aspect,angle :single ;

begin

  { ***
    *** SET NORMAL PARAMETERS
    ***
  }
  { left, center, and right X placement for the lines }
  xl      :=    100+strtoint(bValue.text) ;

  { angle between the three lines }
  angle   :=    strtoint(vAngle.text) ;

  { the xybase (baseline for Y) is xysize }
  xybase  :=    xysize-100;

  { *** set size multiplier ~ and this is used in the center scale answer }
  sL :=15.0 * strtoint(mRvalue.text) / 10;    // left    <<<<< varies
  sR :=sL ;                                     // right   <<<<< fixed
  // sL = sR = sC / (2 cos Angle)    thus    sC = sL * (2 cos Angle)
  sC := sL * (2 * cos(degtoRad*angle));

  { advise about good values }
  recmRvalue.caption := '10';
  recDisplaceY.caption := '-150';
  recBvalue.caption := '150';

  { ***
    *** INITIAL CANVAS SCREEN SETUP
    ***
  }
begin
  { see FormPaint }
  { http://delphi.about.com/library/bluc/text/uc052102d.htm }

  { clear the graph area see: http://delphi.about.com/library/bluc/text/uc052102c.htm}
  Canvas.Rectangle( Bounds(xshift-15, yshift-15, 30+xySize, 30+xySize));

  { say what is being depicted }
  Canvas.TextOut ( xshift,      yshift-14      , 'TYPE 4 TEST');

  { move to aValue starting point if 0,0 (top left) scales shifted of course }
  Canvas.MoveTo ( xshift ,      yshift      );

  { draw the inner box box which lines will be bounded by }
  Canvas.Pen.Color := 100; { dark brown }
  Canvas.Pen.Width := 1;

```

```

Canvas.LineTo ( xshift+xySize, yshift );
Canvas.LineTo ( xshift+xySize, yshift+xySize );
Canvas.LineTo ( xshift , yshift+xySize );
Canvas.LineTo ( xshift , yshift );

{ move pen to top and center }
Canvas.Pen.Width := 1;
Canvas.Pen.Color := 255;
end ;

{ ***
*** F I R S T *** DRAW THE LEFT LINE \
***
}
begin
for hrI := 0 to 20 do
begin
hr := hrI ;
{ get an X }
// cos 60 = 0.5 and sin 60 = 0.866
x := xL +sL * hr * cos(degtorad*angle*2) ;

{ get a Y }
y := -sL * hr * sin (degtorad*angle*2)+ strtoint(yDisp.text);

{ draw a horizontal marker for this latitude }
Canvas.MoveTo ( xshift+ getGrXY(x), yshift+xybase+getGrXY(y) );
Canvas.LineTo ( xshift-10+getGrXY(x), yshift+xybase+getGrXY(y) );

{ label the horizontal marker line }
if (hr < 11) or (hr = 15) or (hr = 20) then
begin
Canvas.TextOut ( xshift-20+getGrXY(x), yshift+xybase-8+getGrXY(y),
FloatToStrF (hr,ffFixed,2,0) );
end ;
end ;

{ draw the actual nomogram line }
x := xL +sL * 0 * cos(degtorad*2*angle) ;
y := -sL * 0 * sin (degtorad*2*angle)+ strtoint(yDisp.text);
Canvas.MoveTo ( xshift+ getGrXY(x),yshift+xybase+getGrXY(y) );
// XX <---- the 0 and the 20 are the scale limits
x := xL +sL * 20 * cos(degtorad*2*angle) ;
y := -sL * 20 * sin (degtorad*2*angle)+ strtoint(yDisp.text);
Canvas.LineTo ( xshift+ getGrXY(x),yshift+xybase+getGrXY(y) );

{ give line a heading }
Canvas.TextOut ( xshift+ getGrXY(x),yshift+xybase+getGrXY(y)-30, 'LEFT 0:20');

end ;

{ ***
*** SECOND *** DRAW THE CENTER LINE /
***
}
begin
for hrI := 0 to 20 do
begin
hr := hrI ;
{ get an X }
x := xL +sC * hri * cos(degtorad*angle) ; { x is L line X }

{ get a Y }
y := -sC * hr * sin (degtorad*angle)+ strtoint(yDisp.text); { multiply by
modulus or scale }

{ draw a horizontal marker for this latitude }
Canvas.MoveTo ( xshift+ getGrXY(x), yshift+xybase+getGrXY(y) );
Canvas.LineTo ( xshift-10+getGrXY(x), yshift+xybase+getGrXY(y) );

{ label the horizontal marker line }

```

30 This document may be freely distributed provided the credit to the web site and author is retained above. Remember that updates may exist on the web site. Chapter 31 of the book *Illustrating Time's Shadow* expands on this topic.

```

    Canvas.TextOut ( xshift-20+getGrXY(x), yshift+xybase-8+getGrXY(y), FloatToStrF
(hr,ffFixed,2,0) );
    end ;

    { draw the actual nomogram line }
    x := xL +sC * 0 * cos(degtorad*angle) ;
    y := -sC * 0 * sin (degtorad*angle)+ strtoint(yDisp.text);
    Canvas.MoveTo ( xshift+ getGrXY(x),yshift+xybase+getGrXY(y) );
    // XX <---- the 0 and the 20 are the scale limits
    x := xL +sC * 20 * cos(degtorad*angle) ;
    y := -sC * 20 * sin (degtorad*angle)+ strtoint(yDisp.text);
    Canvas.LineTo ( xshift+ getGrXY(x),yshift+xybase+getGrXY(y) );

    { give line a heading }
    Canvas.TextOut ( xshift+ getGrXY(x),yshift+xybase+getGrXY(y)-30, 'CENTER
0:20');

    end ;

{ ***
*** THIRD *** DRAW THE RIGHT LINE \ /
***
_____

m1 = m2 = m3 / (2 cos A)

}
begin
for hrI := 0 to 20 do
begin
hr := hrI ;
{ get an X }
x := xL +sL * hri * cos(degtorad*0) ; { x is L line X }

{ get a Y }
y := -sL * hr * sin (degtorad*0)+ strtoint(yDisp.text); { multiply by
modulus or scale }

{ draw a horizontal marker for this latitude }
Canvas.MoveTo ( xshift+ getGrXY(x), yshift-5+xybase+getGrXY(y) );
Canvas.LineTo ( xshift+ getGrXY(x), yshift+5+xybase+getGrXY(y) );

{ label the horizontal marker line }
if (hr < 11) or (hr = 15) or (hr = 20) then
begin
Canvas.TextOut ( xshift+ getGrXY(x), yshift+10+xybase-8+getGrXY(y),
FloatToStrF (hr,ffFixed,2,0) );
end ;
end ;

{ draw the actual nomogram line }
x := xL +sL * 0 * cos(degtorad*0) ;
y := -sL * 0 * sin (degtorad*0)+ strtoint(yDisp.text);
Canvas.MoveTo ( xshift+ getGrXY(x),yshift+xybase+getGrXY(y) );
// XX <---- the 0 and the 20 are the scale limits
x := xL +sL * 20 * cos(degtorad*0) ;
y := -sL * 20 * sin (degtorad*0)+ strtoint(yDisp.text);
Canvas.LineTo ( xshift+ getGrXY(x),yshift+xybase+getGrXY(y) );

{ give line a heading }
Canvas.TextOut ( xshift+ getGrXY(x),yshift+xybase+getGrXY(y)-30, 'RIGHT 0:20');

end ;

end;

```