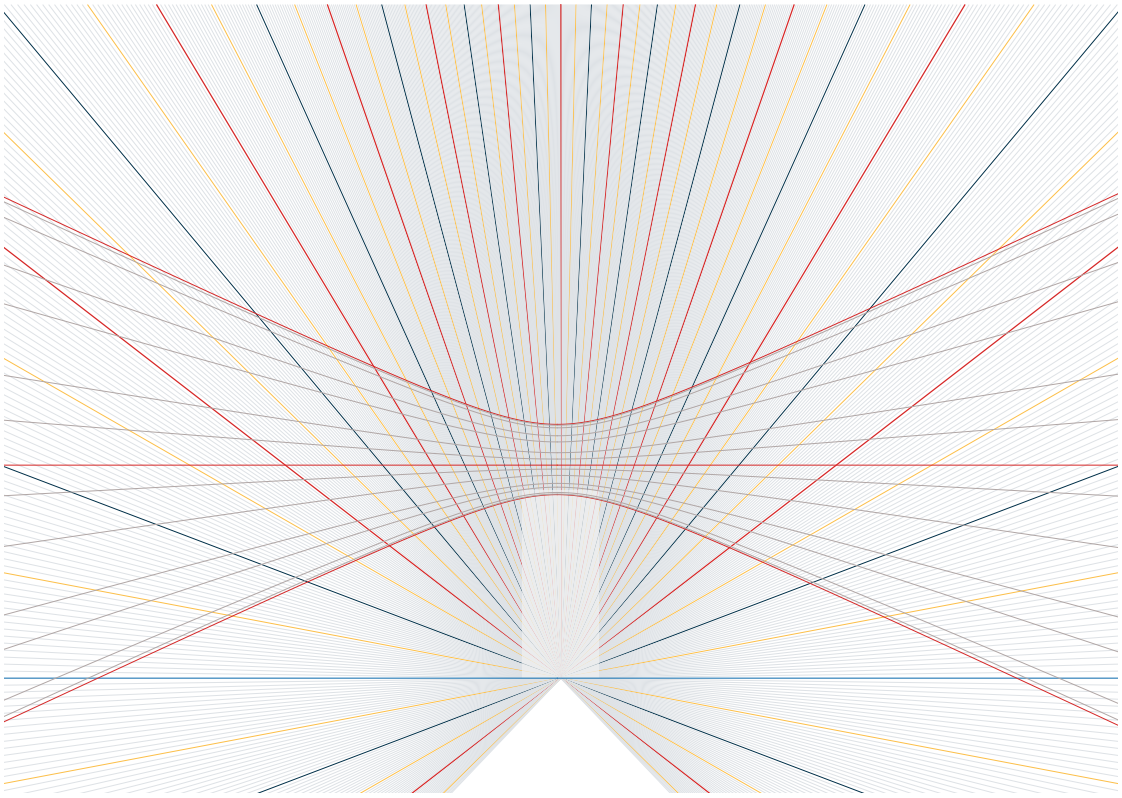


Diagram 1. Horizontal Sundial with Analemma. The Gnomon fits exactly on the rectangle.

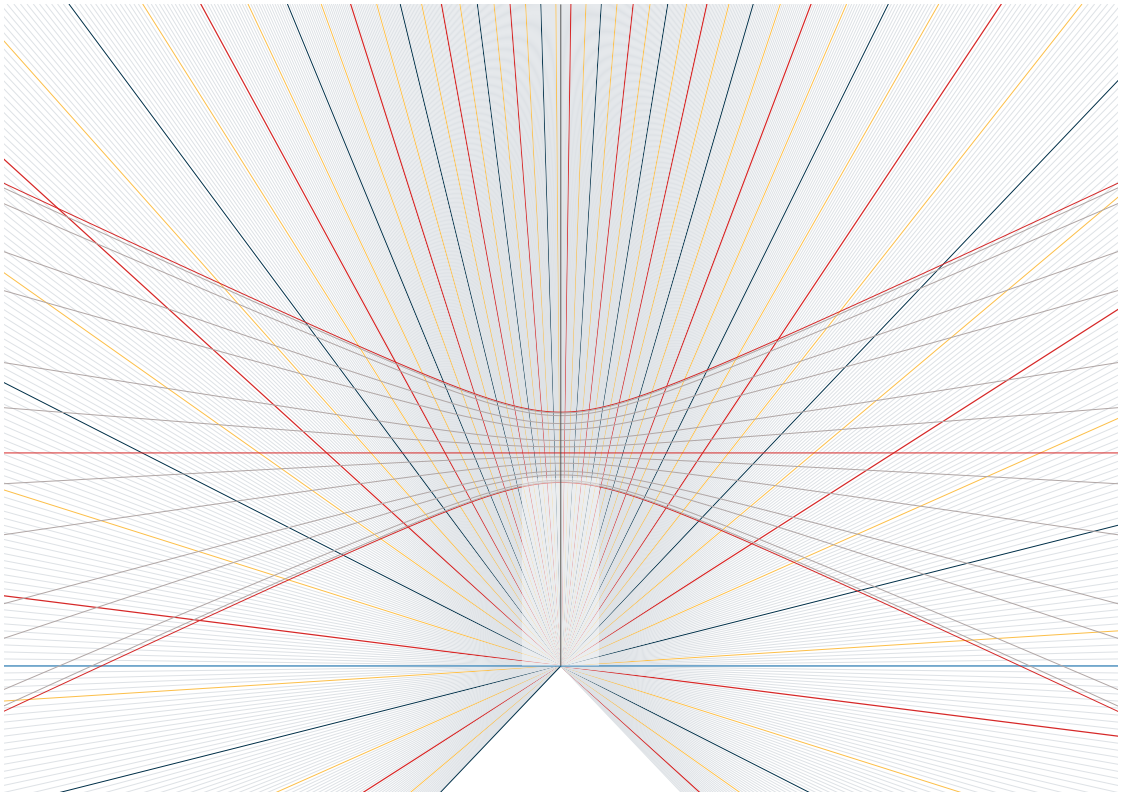


Diagram 2. The same Horizontal Sundial with Analemma, but colored differently to represent the Longitudinal Correction (LC). The difference between the Mauritian Mean Time and the Standard Gulf Mean Time (LC) was taken as exactly 10 minutes. So lines exactly 10 minutes before red Hour lines in Diagram 1 were coloured red instead.

The equation for the lines on a dial is given by[1]:

$$\tan D \;=\; \tan(\tau)\,[\sin(\phi)]$$

Our dial was produced using the graph plotter Matplotlib, a *module* for Python. Calling the point where all lines converge the *Origin*, Matplotlib generates coordinates $(0, 0)$ and (x, y) for each of the lines on the plot.
For a vertical line, the y-component is 1 and the x-component is 0.
As we slowly increase just the x-component, our vertical line becomes more and more slanted.

This leads to a problem.

Look, the x-component is $\tan\theta$, the angle to the vertical.
(Take the *adjacent* as the y-axis. Since the adjacent is unit length, the x-component is given by $\tan\theta \;=\; \frac{x\text{-component}}{1} \;=\; \frac{\text{opposite}}{\text{adjacent}}$)

But as θ increases to 90°, the x-component goes to infinity and cannot be plotted.

The problem was divided into 2 parts; part first increasing to 45° ignoring the infinity problem, and part second, keeping the x-component *at the value it had reached* so now the y-component decreases to 0 as we increase to 90°.

This took about 8 hours of coding and Misse, have some pity, give us marks! 🙏

Each line was produced in a colour based on the time it indicates.

———

The coordinates for the analemmatic hyperbolae are given by[2]:

$$\left(\frac{h\sin T}{\cos T\cos L - \tan D\sin L}\;,\;\frac{h\cos T}{\sin L\,(\cos T\cos L - \tan D\sin L)}\right)$$

By the way each hyperbola is specific to its Solar Declination on that day, and also the height of the Gnomon used.

Our final works had an additional *blue line* to indicate the path the Gnomon tip took on the day we tested our dials.

```
import matplotlib.pyplot as plt
import numpy as np

# tan D = tan(τ)[sin(φ)]

# tau's limits here arbitrary, so figure has comparable height along positive and width
# the latitude used is at Phoenix, 20°16'43.6"
# the corresponding longitude is 57°30'7"
tau = 0
minutesBetweenLines = 60
lineWidth = '0.1'
plt.axis('square')
# (15/60)° represents one minute away from Noon
while tau < 45:
    x = [0]
    y = [0, 1]

    xComponent = np.tan(np.deg2rad(tau)) * np.sin(np.deg2rad(20.27878))
    x.append(xComponent)
    if tau % (15) == 0:
        # print 'red'
        plt.plot(x, y, '#d62828', linewidth='0.33')
    elif tau % (7.5) == 0.0:
        # print 'blue'
        plt.plot(x, y, '#003049', linewidth=lineWidth)
    elif tau % (3.75) == 0.0:
        # print 'yellow'
        plt.plot(x, y, '#fcbf49', linewidth=lineWidth)
    else:
        # print 'background'. Formerly #b1a7a6.
        plt.plot(x, y, '#dee2e6', linewidth=lineWidth)
    tau = tau + (minutesBetweenLines * (15/60))


while tau <= 110:
    x = [0, xComponent]
    y = [0]

    yComponent = xComponent / (np.tan(np.deg2rad(tau)) * np.sin(np.deg2rad(20.27878)))
    y.append(yComponent)
    if tau % (15) == 0:
        # print 'red'
        plt.plot(x, y, '#d62828', linewidth='0.33')
    elif tau % (7.5) == 0.0:
        # print 'blue'
        plt.plot(x, y, '#003049', linewidth=lineWidth)
    elif tau % (3.75) == 0.0:
        # print 'yellow'
        plt.plot(x, y, '#fcbf49', linewidth=lineWidth)
    else:
        # print 'background'
        plt.plot(x, y, '#dee2e6', linewidth=lineWidth)
    tau = tau + (minutesBetweenLines * (15/60))
    print(xComponent)

plt.axis('off')
plt.savefig("test.svg")
plt.show()
```

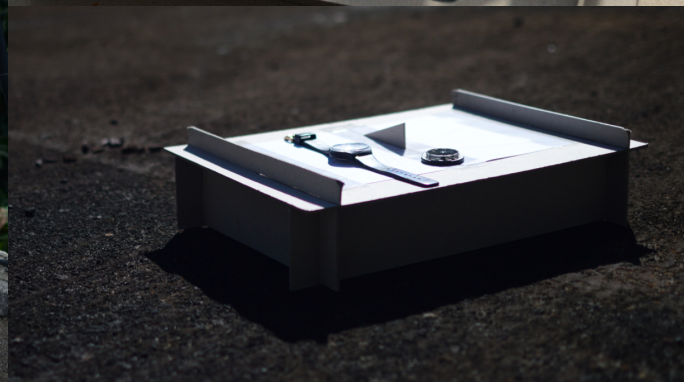
```python
# Angles in Astropy units

import numpy as np
import matplotlib.pyplot as plt
from astropy.coordinates import Angle as angle

# (        h.sinT                    h.cosT          )
# ( ---------------------, --------------------------- )
# ( cosT.cosL - tanD.sinL  sinL(cosT.cosL - tanD.sinL) )

def AccumulateAndPlot(what, colour, thickness):
    ## Numerical Computation for the Domain (Expanse) of Each Hyperbole,
    ## Based on the Width You Specify for Your Analemma
    # Initial Value of T
    T = np.deg2rad(angle('0d'))
    # 'stud' is a random number to begin the iteration
    stud = np.deg2rad(angle('30d'))
    # Final Value of T (use a symbol other than 'T')
    mark = np.arccos((h*np.sin(stud) + l*np.tan(what)*np.sin(L))/(l*np.cos(L)))

    while abs(mark - stud) > angle('0.1d'):
        stud = mark
        mark = np.arccos((h*np.sin(stud) + l*np.tan(what)*np.sin(L))/(l*np.cos(L)))

    # Accumulate Coordinates for Your Declination
    while T < mark:

        xCoord = h * np.sin(T) / ( np.cos(T)*np.cos(L) - np.tan(what)*np.sin(L) )
        x.append(xCoord)

        yCoord = h * np.cos(T) / (np.sin(L)*( np.cos(T)*np.cos(L) - np.tan(what)*np.sin(L) ))
        y.append(yCoord)

        T = T + np.deg2rad(angle('1d'))
    # Plot a Line Joining the Coordinates Calculated
    plt.plot(x, y, colour, linewidth=thickness)



# h, the height of Gnomon, in metres
# h = length of the base of Gnomon * np.tan(angle('20d16m43.6s'))
h = (0.077)*np.tan(angle('20d16m43.6s'))
# l, width hyperbolae extend to
l = 0.6
# Jayvin's declination; a hyperbole the day Jayvin will test his sundial
JayvinDeclination = [angle('21d01m')]

# Latitude at Phoenix
L = angle('20d16m43.6s')

# First of each month, January to December
# Declinations used are Average values, though errors do not exceed 9'
Declinations = [
    angle('-23d4m'), angle('-17d20m'), angle('-7d49m'),
    angle('4d18m'), angle('14d54m'), angle('21d58m'),
    angle('23d9m'), angle('18d10m'), angle('8d30m'),
    angle('-2d57m'), angle('-14d14m'), angle('-21d43m')
    ]

# Special declinations; Mar 21. Sep 23.
SpecialDeclinations = [
    0,
    np.deg2rad(23.5),
    np.deg2rad(-23.5),
    ]



# Horizontal Reference Line
x = [0, 0.5]
y = [0, 0]
plt.plot(x, y, linewidth='0.12')

# Determination of Coordinates
x = []
y = []

AccumulateAndPlot(JayvinDeclination[0], '#003049', '0.1')

d = 0
while d < 3:

    x = []
    y = []

    AccumulateAndPlot(SpecialDeclinations[d], '#d62828', '0.1')

    d = d + 1

d = 0
while d < 12:

    x = []
    y = []

    AccumulateAndPlot(Declinations[d], '#b1a7a6', '0.1')

    d = d + 1


# Output Distance of Equinox line from Horizontal through Origin
# (This is the yCoord when T = 0° and D = 0°).
Distance = h * np.cos(0) / (np.sin(L)*( np.cos(0)*np.cos(L) - np.tan(0)*np.sin(L) ))
print('The Distance of the Equinox line from Horizontal through Origin is '
    + str(Distance) + ' metres.')

# Plot
plt.axis('square')
plt.axis('off')
plt.savefig("test.svg")
plt.show()
```

```python
# Assumed Correction is Exactly 10mins.

import matplotlib.pyplot as plt
import numpy as np

# tan D = tan(τ)[sin(φ)]

# tau's limits here arbitrary, so figure has comparable height along positive and width
# the latitude used is at Phoenix, 20°16'43.6"
# the corresponding longitude is 57°30'7"
# since the plot is asymmetric in colour, we cannot plot just half of the dial and mirror it
# we start plot at -110° from the vertical, and end at 110°
tau = -110
minutesBetweenLines = 1
lineWidth = '0.1'
plt.axis('square')
# (15/60)° represents one minute away from Noon

# This value got from an iteration of the first code provided
# Its negative was taken since at tau = -110, xComponent is negative
xComponent = -0.20010283231263087

bothLoopsComplete = 0
# This while loop to skip #1, read chunk #2, then through #1 and #2
while bothLoopsComplete < 2:

    while abs(tau) < 45:
        x = [0]
        y = [0, 1]

        xComponent = np.tan(np.deg2rad(tau)) * np.sin(np.deg2rad(20.27878))
        x.append(xComponent)
        # These 'if-else' are simply 'checks' that give colour
        # to your already-obtained lines.
        if ( tau - 10 * (15/60) ) % (15) == 0:
            # print 'red'
            plt.plot(x, y, '#d62828', linewidth='0.33')
        elif ( tau - 10 * (15/60) ) % (7.5) == 0.0:
            # print 'blue'
            plt.plot(x, y, '#003049', linewidth=lineWidth)
        elif ( tau - 10 * (15/60) ) % (3.75) == 0.0:
            # print 'yellow'
            plt.plot(x, y, '#fcbf49', linewidth=lineWidth)
        else:
            # print 'background'. Formerly #b1a7a6.
            plt.plot(x, y, '#dee2e6', linewidth=lineWidth)
        tau = tau + (minutesBetweenLines * (15/60))

    while abs(tau) >= 45 and abs(tau) <= 110:
        x = [0, xComponent]
        y = [0]

        yComponent = xComponent / (np.tan(np.deg2rad(tau)) * np.sin(np.deg2rad(20.27878)))
        y.append(yComponent)
        if ( tau - 10 * (15/60) ) % (15) == 0:
            # print 'red'
            plt.plot(x, y, '#d62828', linewidth='0.33')
        elif ( tau - 10 * (15/60) ) % (7.5) == 0.0:
            # print 'blue'
            plt.plot(x, y, '#003049', linewidth=lineWidth)
        elif ( tau - 10 * (15/60) ) % (3.75) == 0.0:
            # print 'yellow'
            plt.plot(x, y, '#fcbf49', linewidth=lineWidth)
        else:
            # print 'background'
            plt.plot(x, y, '#dee2e6', linewidth=lineWidth)
        tau = tau + (minutesBetweenLines * (15/60))

    bothLoopsComplete = bothLoopsComplete + 1


plt.axis('off')
plt.savefig("test.svg")
plt.show()
```

Disclaimer

I—and Beepha and Gangaram—first got the Longitudinal Correction wrong—correcting counter-clockwise instead of clockwise. But this just means we have a systematic error of exactly 10 minutes in all our readings.

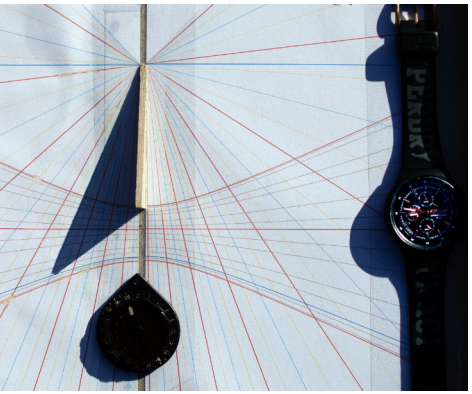
For May 29, our Sundial reads 17m11s $\simeq$ 17min less than our watch time ($\underset{Syst.Corr.}{10\text{min}}$ + $\underset{Long.Corr.}{10\text{min}}$ − $\underset{Eq.of Time}{2\text{m}49\text{s}}$[3]).

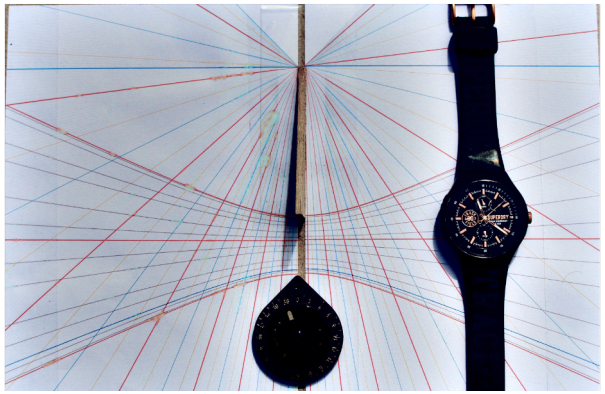


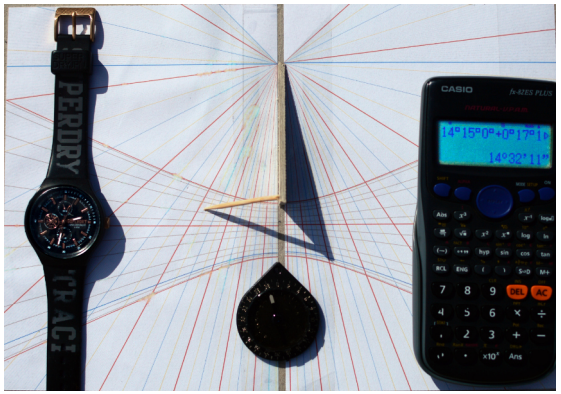
#1 The Gnomon was cut small from a larger triangle designed using trigonometry.

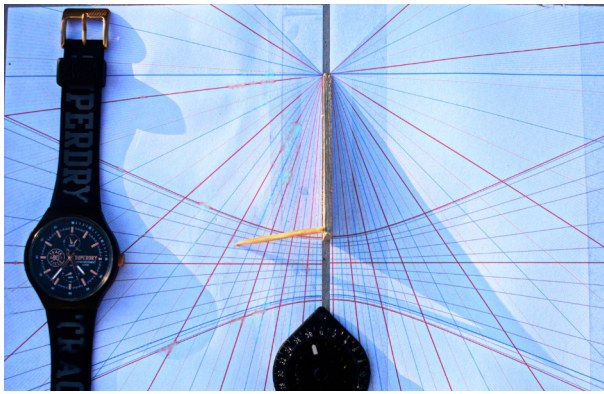
#2 Before noon the Sundial was oriented at 018.20°[4] to Magnetic North using compass.

#3 And *at* noon it was re-aligned using the shadow of a plumb line hanging at True Noon (12:07.11 on May 29).









In practice, the times shown on Sundial differs a little from our expected values (even after all corrections!) Because the fact our Gnomon was glued therefore not truly perpendicular comes into play. The explanation for a toothpick being used to hold it in place!

Sundial Times are nearly correct as do relate the tip of their shadows to the *blue analemma* of May 29.