Optimal Room Orientation to Minimize Solar Incidence

Manu Anish

August 2023

■

Contents

1 Introduction

1.1 Rationale

I live in Ahmedabad, a sunbaked tropical region in the far west of India. As one of the hottest regions in the country (Kalsi and Pareek p. 1), the scorching heat can be unbearable, surpassing 45 degrees Celsius (113 degrees Fahrenheit) during the peak summer months.

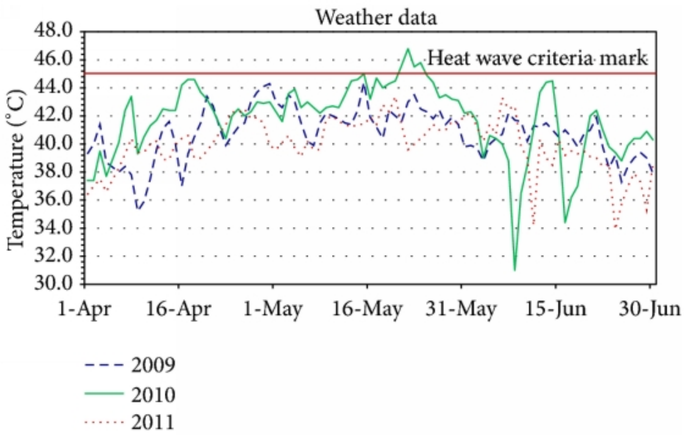

Figure 1 *— The extreme temperatures from the heat wave of 2010. Image source: (Kakkad, Khyati, et al. p. 4)*

These extreme temperatures have dire consequences, as evidenced by the devastating heat waves that struck Ahmedabad in 2010, resulting in over 100 deaths (Azhar et al. p. 4).

Furthermore, the relentless sunlight that permeates through our windows increasing the heat inside our homes. It not only raises the indoor temperature but also our the reliance on energy-intensive air conditioning systems, which further strain the power grid and contribute to environmental degradation.

In light of these issues, I am determined to explore and challenge my mathematical knowledge to address the heat problem. By investigating the optimal room orientation, I aim to find a viable solution that can reduce solar heat gain while maintaining a comfortable indoor environment.

2 Methodology

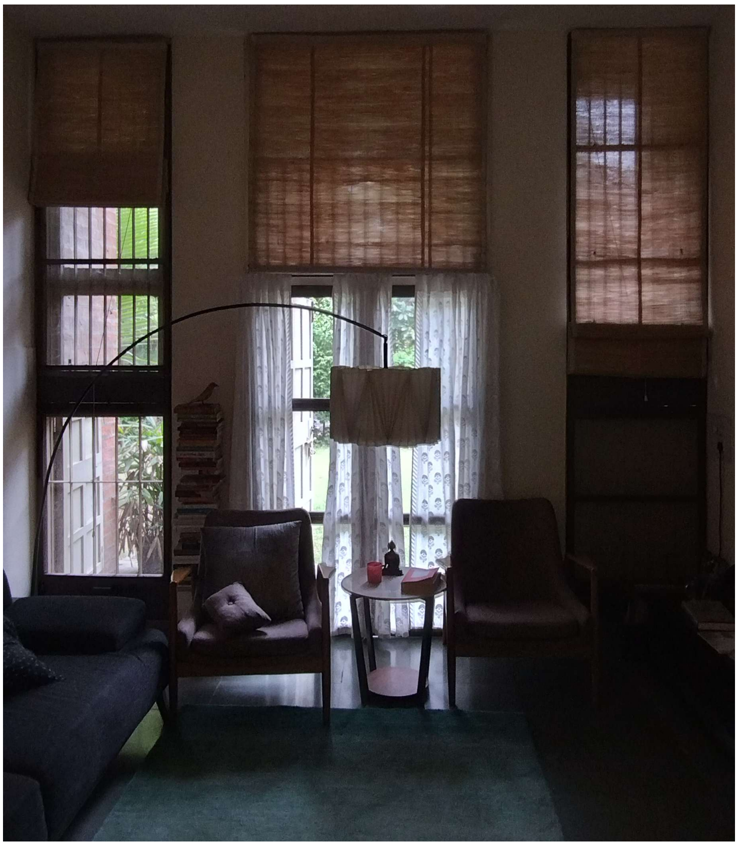

Figure 2 *— A photo of my living room.*

In this document I will specifically look at my living room and consider the orientations that it could be rotated to minimize the amount of sunlight that enters during the day.

2.1 The Model

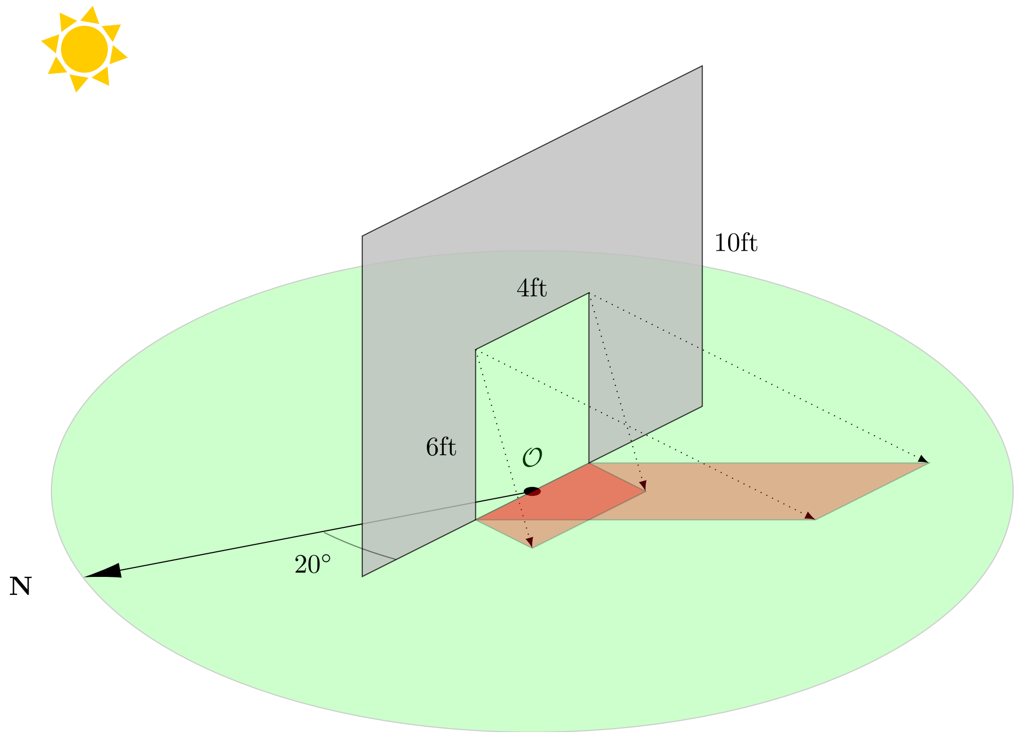

Figure 3 — *Rays of sunlight casted through the patio door. (Isometric)*

For the ease of calculation, I will consider only the patio door and since it is the major source of sunlight in my living room.

> **Remark:** Since the sun is negligibly distant from my house, I consider the sun a *directional light*; that is, parallel vectors.

2.2 Objective

My living currently room is oriented 20° North-West. I intend to consider different angles of orientation about $\mathcal{O}$ and find an "optimium" angle of orientation that minimizes the total sunlight entering my living room during the summer.

2.3 What is an "optimium" orientation?

For calculating the sunlight area the following factors are to be considered:

- My latitude $23.0225° N$ (constant)
- Hour of the day (variable)
- Day of the year (variable)
- Door dimensions 4ft × 6ft (constant)
- Room Length 10ft (constant)
- Room Orientation (variable)

By an optimal orientation I refer to one for which the total sunlight area entering my room is minimal for all 24 hours of day and 365 days of the year.

3 Modeling the Solar Vector

All of the equations in this section are cited from Wikipedia and are not my original derivations. They are restated and mildly modified to fit my circumstance.

3.1 Equitorial Coordinates

By convention, we use the **equitorial coordinate system** (Equatorial coordinate system , Wikipedia) to specify positions of celestial objects. The basic setup is given below:

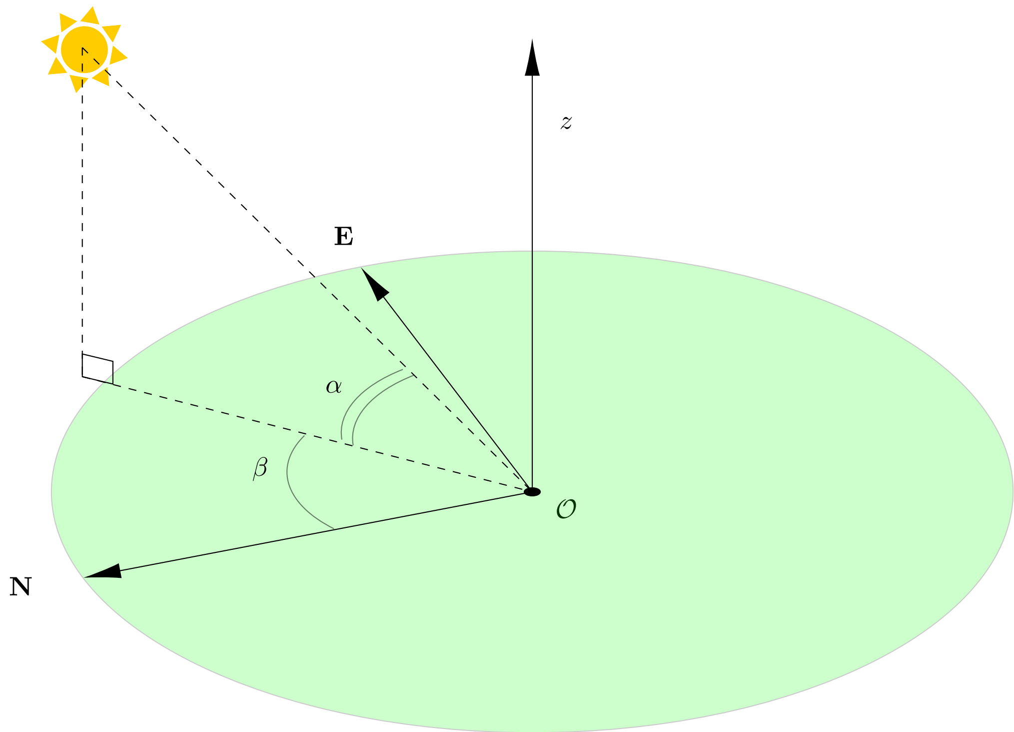

Figure 4 — *Position of the sun. (Isometric)*

The angle β denotes the **solar azimuth angle** (Solar azimuth angle , Wikipedia). The angle α denotes the **solar altitude**, the complement of the (Solar zenith angle , Wikipedia).

Remark: North (**N**) is given by 0° **Azimuth** (Azimuth , Wikipedia), so β is measured clockwise relative to **N** and $\mathcal{O}$. Similarly, α is measured clockwise relative to the **Zenith** (Zenith , Wikipedia) (z) and $\mathcal{O}$.

To calculate these angles we can use the equations cited from (Solar zenith angle , Wikipedia) and (Solar azimuth angle , Wikipedia). That is:

3.1.1 Solar Altitude α

Theorem 3.1: (Solar Altitude) The **Solar Altitude** α (Solar zenith angle) is given by:

$$\alpha = \sin^{-1}\left(\sin\Phi\sin\delta + \cos\Phi\cos\delta\cos h\right)$$

Where:

- Φ is the observer's latitude.
- δ is the **declination** of the sun.
- h is the **hour angle** in local solar time.

3.1.1.1 Solar Zenith θ

Remark: (Solar Zenith) The **Solar Zenith** θ (Solar zenith angle) is the complement of the solar altitude. Thus it is given simply by:

$$\theta = 90^\circ - \alpha$$

3.1.2 Solar Azimuth β

Theorem 3.2: (Solar Azimuth) The **Solar Azimuth** β (Solar azimuth angle) is given by:

$$\sin\beta = \frac{-\sin h\cos\delta}{\sin\theta}$$

Where:

- h is the **hour angle** in local solar time.
- δ is the **declination** of the sun.
- θ is the **Solar Zenith**.

Since my latitude Φ is going to be a constant, I have to define two more variables in order to compute the solar vector. That is, h the hour angle and δ the declination of the sun.

3.1.3 Declination δ

> **Theorem 3.3:** (Declination of the Sun) The **declination** of the sun is given by:
>
> $$\delta = -23.44^\circ \cdot \cos\left[\frac{360^\circ}{365} \cdot (N + 10)\right]$$
>
> Where N is the day of the year starting from $N = 0$.

3.1.4 Hour Angle h

> **Theorem 3.4:** (Hour Angle) The **hour angle** in local solar time is given by:
>
> $$h = 15^\circ(LST - 12)$$
>
> Where LST is the hour of the day (including fractional minutes).

3.2 Creating the Model

Using Python (Python 3), I converted the theorems above into functions. Using the library Matplotlib (Matplotlib) I was able to create a 3D vector pointing towards the sun given my latitude, time of day, and day of the year.

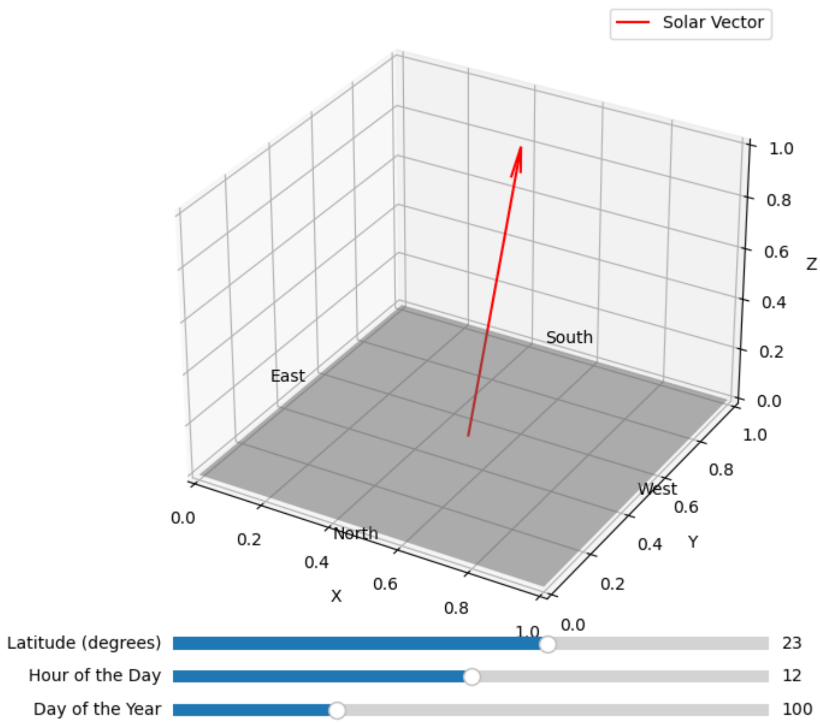

Figure 5 *— An interactive demo of the solar vector I created.*

3.2.1 Path of the Sun

I created the demo mostly for intuition and to understand the behavior of the Sun throughout the year. Messing around with it for a bit, I quickly realized that the sun generally rises from the East and sets towards the West. This is a crucial observation, as I can intuitively guess a minimal orientation would be at either 90° or 270° as this allows the patio door to be parallel to the Sun's movement. The room is therefore largely affected by overhead sunlight which is minimal as opposed to the directly lateral sunlight at 0° and 180°.

4 Calculating Solar Incidence

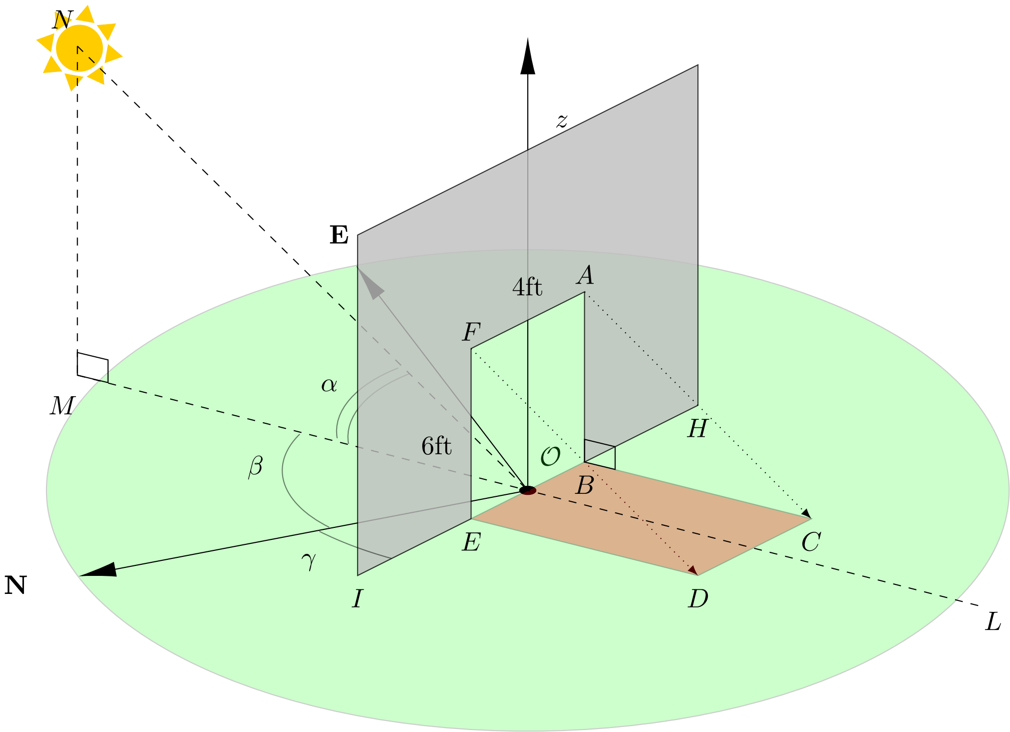

Figure 6 — *Calculating sunlight area with the solar vector.*

Suppose the given solar vector is $\vec{\boldsymbol{v}}$ with azimuth β and altitude α. Notice that all rays will be parallel since we assume the sun is a **directional light**.

It follows that $AC \parallel \mathcal{O}N$. Then we have $\angle AC\mathcal{O}$ is α, and since $AB \parallel MN$ we find that $\triangle ABC \sim \triangle MN\mathcal{O}$. Then, $\angle C = \alpha$ and $AB = 6\text{ft}$. Therefore, $\tan(\alpha) = \frac{6\text{ft}}{BC}$. Rearranging, we can solve for BC:

$$BC = \frac{6\text{ft}}{\tan(\alpha)}$$

Another observation is that $ED \parallel BC$ and since $FA \parallel EB \implies EB \parallel CD$. In other words, $\square BCDE$ is a rhombus.

4.1 Finding the area of $\square BCDE$

We know that $\angle I\mathcal{O}M = \beta + \gamma$ where γ is the orientation of my living room. Then since $\angle H\mathcal{O}L$ is an opposite angle it follows $\angle I\mathcal{O}M = \angle H\mathcal{O}L$. Then since $\mathcal{O}L \parallel BC$ we have $\angle HBC = \angle H\mathcal{O}L = \beta + \gamma$. Here is a top-down view for reference:

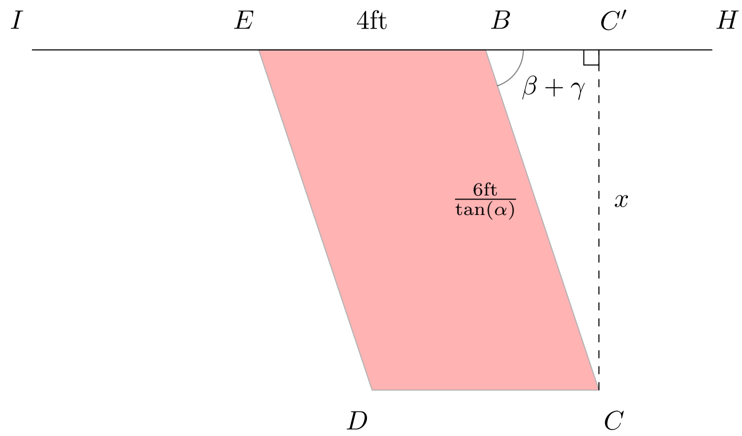

Figure 7 — *Top down view of the sunlight.*

The area of the sunlight is going to be BASE × HEIGHT, since we know $\square BCDE$ is a rhombus. We know the base $BE = 4\text{ft}$ so we just need to find the height CC'. Since, $\sin(B) = \frac{CC'}{BC}$ we find:

$$CC' = \sin(B) \cdot BC$$
$$CC' = \frac{\sin(\beta+\gamma) \cdot 6\text{ft}}{\tan(\alpha)}$$

Then the area of the sunlight is given by:

$$[\square BCDE] = CC' \times BC = \frac{\sin(\beta+\gamma) \cdot 6\text{ft}}{\tan(\alpha)} \times 4\text{ft} = \boxed{\frac{\sin(\beta+\gamma) \cdot 24\text{ft}}{\tan(\alpha)}}$$

4.2 Corrections

While the derived equation for sunlight area is largely accurate, there are a few edge cases to consider.

4.2.1 Sign Corrections

Notice I care about the area of sunlight "inside" my living room. In other words, I can use the sign of $\frac{\sin(\beta+\gamma)}{\tan(\alpha)}$ and define the area as a piecewise function:

$$[\square BCDE] = \begin{cases} \dfrac{\sin(\beta+\gamma)\cdot 24\text{ft}}{\tan(\alpha)} & \text{if } \frac{\sin(\beta+\gamma)}{\tan(\alpha)} > 0 \\ 0 & \text{if } \frac{\sin(\beta+\gamma)}{\tan(\alpha)} < 0 \end{cases}$$

Therefore, when the area is negative (on the opposite side) I set the area to be 0 so that it is not including in my calculations.

4.2.2 Area Bounds

Observe for smaller values of $\tan(\alpha)$ the sunlight area can exceed the size of my living room, thus skewing actual sunlight area to be much larger than it should be. (e.g when the sun is just a little above the horizon)

To deal with this, I bound the area of the sunlight to the length of my living room. As such, a maximum area of $4 \cdot 10 = 40\text{ft}$. Then our finalized area function looks like:

$$[\square BCDE] = \begin{cases} \min\left(\dfrac{\sin(\beta+\gamma)\cdot 24\text{ft}}{\tan(\alpha)},\ 40\text{ft}\right) & \text{if } \frac{\sin(\beta+\gamma)}{\tan(\alpha)} > 0 \\ 0 & \text{if } \frac{\sin(\beta+\gamma)}{\tan(\alpha)} < 0 \end{cases}$$

This ensures that the sun cannot produce unrealistically large areas.

4.3 Discrete Summation of Sunlight Area

For simplicity, I consider only discrete sums of integer hours and integer days. That is,

$$\sum_{d=0}^{365}\left(\sum_{h=0}^{24}[\square BCDE]\right), \qquad h, d \in \mathbb{Z}$$

I then aim to minimize this value with varying values of γ. As such,

$$\min\left(\{0 \le \gamma \le 360\ :\ \sum_{d=0}^{365}\left(\sum_{h=0}^{24}[\square BCDE]\right),\quad h, d \in \mathbb{Z}\}\right)$$

5 Minimizing Solar Incidence

To find an optimal room orientation γ, I will graph the the orientation vs the discrete summation of sunlight area and use the argmin [$\square BCDE$] (Arg max) to compute the minimal orientation:

5.1 Modeling Sunlight Area

I rewrote the discrete sums in python, and plotted the area against the orientation, here are the results:

> **Remark:** The code is omitted here for readability, you can find the full source code in section 7.2 of the appendix.

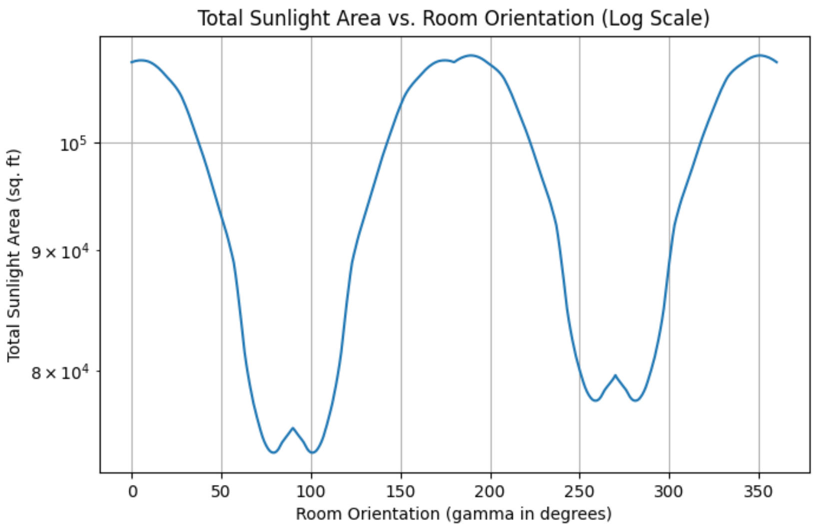

Figure 8 *— Log plot of discrete sunlight area vs room orientation.*

5.2 Optimal Room Orientation

To find the minima of [$\square BCDE$], I use the python library `numpy` specifically the `numpy.argmin` function. It uses an algorithm similar to Newton's Method to find local minima and compares it against existing ones to find a global minima.

Remark: The code is omitted here for readability, you can find the full source code in section 7.3 of the appendix.

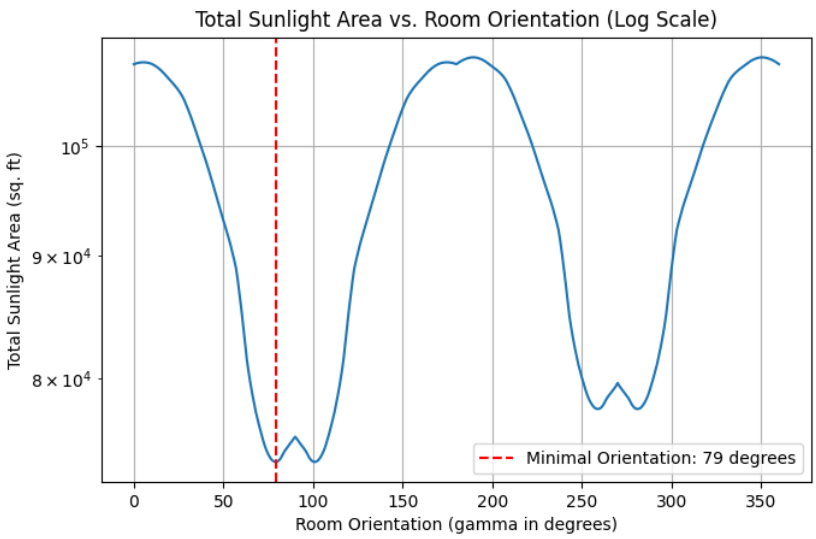

***Figure 9** — Optimal Room Orientation is 79° (Log Plot)*

6 Conclusion

So, using my knowledge of math I was able to find the optimal orientation of my room is 79° to minimize the amount of sunlight I have entering my room. While there are other factors that effect this value in practice, this gives us a rough estimate of how a room should be oriented to reduce solar incidence throughout the year.

6.1 Data Analysis (Making Sense of 79°)

79° aligns with my observations from section 3.2.1 as 79° Azimuth is close to the East-West $(90°, 270°)$ path. This can also been seen in the broad cyclicity observed in the graph every 180°. However, there is a slight distinction. Naturally we might ask why 73° is more optimal than say, $73° + 180° = 253°$? This difference can be attributed to varying declinations of the sun. Due to different latitudes, and the tilt of the Earth, the path of the sun is slightly skewed towards the North/South leading to an imbalance as seen in the graph.

6.2 Limitations of the Method

In reality the atmospheric conditions and diffusion of the light play a large role in determining the actual effect of heat from the sunlight entering the room (Diffuse reflection , Wikipedia). While the rhombic sunlight area provides a good approximation, in practice, we will likely have more scattered and reflected area of effect:

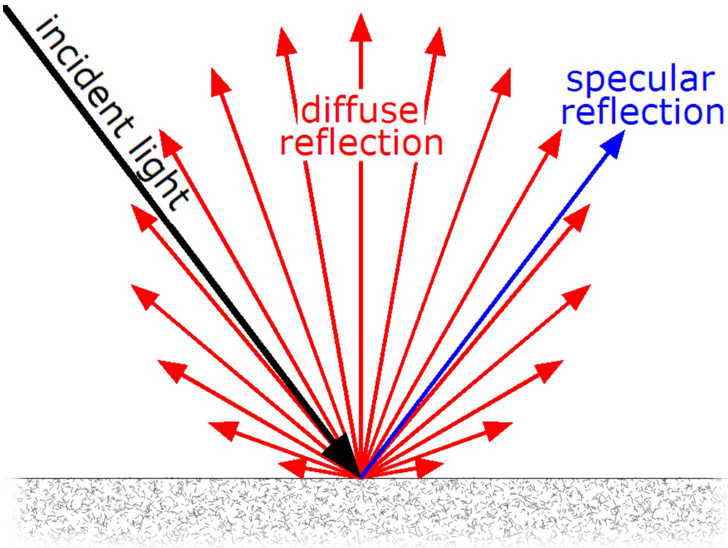

Figure 10 — *Diffused Light on a Surface. Image source Diffuse reflection*

Moreover, the dispersed light can contribute to localized heating on various surfaces, potentially increasing the overall heat gain.

6.3 Practical Applications

The idea of passive cooling is by no means novel. It is especially well-known as one of the most environmentally friendly means to combat climate change. For example, (Oropeza-Perez and Østergaard p. 2), considers a review of other such related passive cooling techniques.

The objective of this study is to provide findings that can complement and collaborate with existing research efforts. By doing so, reducing our reliance on environmentally detrimental active cooling systems.

Conventional cooling methods not only consume substantial amounts of energy but also contribute significantly to greenhouse gas emissions. Therefore, my results are intended to contribute to a broader strategy aimed at creating sustainable, environmentally conscious practices in the built environment.

6.4 Reflection

And there we have it! I thoroughly enjoyed this exploration, and the journey that brought me to the final result of 79°. Aligning this outcome with my initial intuition was particularly gratifying; this experience has unquestionably deepened my fascination with the subject.

The project is by no means complete. Sooner or later, I want to get back to this and enhance the accuracy of these results by accounting for the effects of light diffusion and atmospheric conditions. In the end, I hold the hope that my work will contribute to making the world a better, more sustainable place, one room at a time.

Works Cited

Arg max. *Wikipedia*, June 2023, Page Version ID: 1161677835. en.wikipedia.org/w/index.php?title=Arg_max&oldid=1161677835#Arg_min. Accessed 4 Sept. 2023.

Azhar, Gulrez Shah, et al. "Heat-Related Mortality in India: Excess All-Cause Mortality Associated with the 2010 Ahmedabad Heat Wave". *PLoS ONE*, vol. 9, no. 3, Mar. 2014, e91831. https://doi.org/10.1371/journal.pone.0091831.

Azimuth. *Wikipedia*, Aug. 2023, Page Version ID: 1170448401. en.wikipedia.org/w/index.php?title=Azimuth&oldid=1170448401. Accessed 1 Sept. 2023.

Diffuse reflection. *Wikipedia*, Nov. 2021, Page Version ID: 1055411516. en.wikipedia.org/w/index.php?title=Diffuse_reflection&oldid=1055411516. Accessed 6 Sept. 2023.

Equatorial coordinate system. *Wikipedia*, Apr. 2023, Page Version ID: 1149147383. en.wikipedia.org/w/index.php?title=Equatorial_coordinate_system&oldid=1149147383. Accessed 1 Sept. 2023.

Kakkad, Khyati, et al. "Neonates in Ahmedabad, India, during the 2010 Heat Wave: A Climate Change Adaptation Study". *Journal of Environmental and Public Health*, vol. 2014, Mar. 2014, Publisher: Hindawi, e946875. https://doi.org/10.1155/2014/946875.

Kalsi, S. R., and R. S. Pareek. "Hottest April of the 20th century over northwest and central India". *Current Science*, vol. 80, no. 7, 2001, Publisher: Temporary Publisher, pp. 867–73. *JSTOR*, www.jstor.org/stable/24105739. Accessed 21 Aug. 2023.

Matplotlib. matplotlib.org/stable/index.html#. Accessed 2 Sept. 2023.

Oropeza-Perez, Ivan, and Poul Alberg Østergaard. "Active and passive cooling methods for dwellings: A review". *Renewable and Sustainable Energy Reviews*, vol. 82, Feb. 2018, pp. 531–44. https://doi.org/10.1016/j.rser.2017.09.059.

Python 3. *Python.org*, Aug. 2023. www.python.org/. Accessed 2 Sept. 2023.

Solar azimuth angle. *Wikipedia*, Aug. 2023, Page Version ID: 1170286659. en.wikipedia.org/w/index.php?title=Solar_azimuth_angle&oldid=1170286659. Accessed 1 Sept. 2023.

Solar zenith angle. *Wikipedia*, June 2023, Page Version ID: 1159258763. en.wikipedia.org/w/index.php?title=Solar_zenith_angle&oldid=1159258763. Accessed 1 Sept. 2023.

Zenith. *Wikipedia*, Aug. 2023, Page Version ID: 1171273127. en.wikipedia.org/w/index.php?title=Zenith&oldid=1171273127. Accessed 1 Sept. 2023.

7 Appendix

7.1 Interactive Solar Vector

Here is the source code for the interactive demo of the solar vector I made. The code uses two files `library.py` containing the mathematical functions to create the simulation and `main.py` using matplotlib to create the view.

7.1.1 `library.py`

```
import math

# Solar Altitude
def calculate_solar_altitude(latitude, declination,
    hour_angle):
    rad_latitude = math.radians(latitude)
    rad_declination = math.radians(declination)
    rad_hour_angle = math.radians(hour_angle)

    solar_altitude = math.degrees(
        math.asin(math.sin(rad_latitude) *
        math.sin(rad_declination) +
        math.cos(rad_latitude) *
        math.cos(rad_declination) *
        math.cos(rad_hour_angle))
    )

    return solar_altitude

# Solar Zenith
def calculate_solar_zenith(solar_altitude):
    solar_zenith = 90 - solar_altitude
    return solar_zenith

# Solar Azimuth
def calculate_solar_azimuth(hour_angle, declination,
    solar_zenith):
    rad_hour_angle = math.radians(hour_angle)
    rad_declination = math.radians(declination)
    rad_solar_zenith = math.radians(solar_zenith)

```

```
    numerator = -math.sin(rad_hour_angle) * math.cos(rad_declination)
    denominator = math.sin(rad_solar_zenith)

    solar_azimuth = math.degrees(math.asin(numerator / denominator))

    return solar_azimuth

# Declination of the Sun
def calculate_declination(day_of_year):
    angle = 360 / 365 * (day_of_year + 10)
    declination = -23.44 * math.cos(math.radians(angle))

    return declination

# Hour Angle
def calculate_hour_angle(local_solar_time):
    hour_angle = 15 * (local_solar_time - 12)
    return hour_angle
```

7.1.2 main.py

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.widgets import Slider
from library import calculate_declination, calculate_hour_angle, calculate_solar_altitude, calculate_solar_azimuth

# Create a figure and a 3D axis
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Set the initial values for day of the year, hour of the day, and latitude
initial_day_of_year = 100
initial_hour_of_day = 12
initial_latitude = 23.0225  # Default latitude

# Create sliders for day of the year, hour of the day, and latitude
ax_day = plt.axes([0.2, 0.01, 0.65, 0.03])  # Adjusted position
ax_hour = plt.axes([0.2, 0.05, 0.65, 0.03])  # Adjusted position
ax_latitude = plt.axes([0.2, 0.09, 0.65, 0.03])  # Adjusted position

slider_day = Slider(ax_day, 'Day of the Year', 0, 365, valinit=initial_day_of_year)
slider_hour = Slider(ax_hour, 'Hour of the Day', 0, 24, valinit=initial_hour_of_day)
slider_latitude = Slider(ax_latitude, 'Latitude (degrees)', -90, 90, valinit=initial_latitude)

# Function to update the plot based on slider values
def update(val):
    day_of_year = slider_day.val
    hour_of_day = slider_hour.val
    latitude = slider_latitude.val

    # Calculate declination and hour angle
    declination = calculate_declination(day_of_year)
    hour_angle = calculate_hour_angle(hour_of_day)
```

```python

    # Calculate solar altitude and azimuth
    solar_altitude = calculate_solar_altitude(latitude, declination, hour_angle)
    solar_azimuth = calculate_solar_azimuth(hour_angle, declination, 90 - solar_altitude)

    # Convert to radians for 3D vector calculation
    solar_altitude_rad = np.radians(solar_altitude)
    solar_azimuth_rad = np.radians(solar_azimuth)

    # Calculate the 3D vector components
    x = np.sin(solar_azimuth_rad) * np.cos(solar_altitude_rad)
    y = np.cos(solar_azimuth_rad) * np.cos(solar_altitude_rad)
    z = np.sin(solar_altitude_rad)

    # Clear the previous plot
    ax.clear()

    # Add a 1x1x1 plane at the origin (0, 0, 0)
    ax.plot_surface(
        np.array([[0, 1], [0, 1]]),
        np.array([[0, 0], [1, 1]]),
        np.array([[0, 0], [0, 0]]),
        color='gray',
        alpha=0.5
    )

    # Position the solar vector in the middle of the plane
    ax.quiver(0.5, 0.5, 0, x, y, z, color='red', label='Solar Vector', arrow_length_ratio=0.1)

    # Add labels for North, South, East, and West
    ax.text(0.5, -0.05, 0, 'North', color='black', horizontalalignment='center')
    ax.text(0.5, 1.05, 0, 'South', color='black', horizontalalignment='center')
    ax.text(-0.05, 0.5, 0, 'East', color='black', horizontalalignment='center')
    ax.text(1.05, 0.5, 0, 'West', color='black', horizontalalignment='center')

    ax.set_xlim(0, 1)
    ax.set_ylim(0, 1)
    ax.set_zlim(0, 1)
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('Z')
    ax.legend()

    fig.canvas.draw()

# Attach the slider update function to the slider events
slider_day.on_changed(update)
slider_hour.on_changed(update)
slider_latitude.on_changed(update)

# Initial plot
update(None)

plt.show()
```

7.2 Discrete Sums Plot

Here is the source code for plot of the discrete sums vs room orienation. The code refers to the old `library.py` in section 7.1.1 of the appendix containing the mathematical functions for calculations and a new `plot.py` using matplotlib to create the view.

7.2.1 plot.py

```
import matplotlib.pyplot as plt
import numpy as np
from library import calculate_solar_altitude , calculate_solar_azimuth , calculate_declination , calculate_hour_angle

def calculate_sunlight_area(latitude , day_of_year , hour_of_day , room_orientation):
    declination = calculate_declination(day_of_year)
    hour_angle = calculate_hour_angle(hour_of_day)
    solar_altitude = calculate_solar_altitude(latitude , declination , hour_angle)
    solar_azimuth = calculate_solar_azimuth(hour_angle , declination , 90 - solar_altitude)

    beta = solar_azimuth
    alpha = solar_altitude

     # Calculate the area with sign correction
    area = (np.sin(np.radians(beta + room_orientation)) * 24) / np.tan(np.radians(alpha))

    # Check if it is negative
    if area < 0:
        area = 0

    # Cap the area at 40
    if area > 40:
        area = 40

    return area

def calculate_total_sunlight_area(latitude , room_orientation ):
    total_area = 0
    for day_of_year in range(366):  # Loop through all days of the year
        for hour_of_day in range(25):  # Loop through all hours of the day
            total_area += calculate_sunlight_area(latitude , day_of_year , hour_of_day , room_orientation)
    return total_area

latitude = 23.0225  # Your latitude
gamma_range = range(361)  # Room orientation from 0 to 360 degrees

total_areas = []
for gamma in gamma_range:
    total_area = calculate_total_sunlight_area(latitude , gamma)
    total_areas.append(total_area)

```

```
# Create a log plot
plt.semilogy(gamma_range, total_areas)
plt.xlabel('Room Orientation (gamma in degrees)')
plt.ylabel('Total Sunlight Area (sq. ft)')
plt.title('Total Sunlight Area vs. Room Orientation (Log
    Scale)')
plt.grid(True)
plt.show()
```

7.3 Optimal Room Orientation Plot

Here is the source code for plot of the discrete sums vs room orienation. The code refers to the old `library.py` in section 7.1.1 of the appendix containing the mathematical functions for calculations and a new `plot.py` using matplotlib to create the view.

7.3.1 optimize.py

```
import matplotlib.pyplot as plt
import numpy as np
from library import calculate_solar_altitude,
    calculate_solar_azimuth, calculate_declination,
    calculate_hour_angle

def calculate_sunlight_area(latitude, day_of_year,
    hour_of_day, room_orientation):
    declination = calculate_declination(day_of_year)
    hour_angle = calculate_hour_angle(hour_of_day)
    solar_altitude = calculate_solar_altitude(latitude,
        declination, hour_angle)
    solar_azimuth = calculate_solar_azimuth(hour_angle,
        declination, 90 - solar_altitude)

    beta = solar_azimuth
    alpha = solar_altitude

    # Calculate the area with sign correction
    area = (np.sin(np.radians(beta + room_orientation)) *
        24) / np.tan(np.radians(alpha))

    # Check if it is negative
    if area < 0:
        area = 0

    # Cap the area at 40
    if area > 40:
        area = 40

    return area

def calculate_total_sunlight_area(latitude, room_orientation
    ):
    total_area = 0
    for day_of_year in range(366):  # Loop through all days
        of the year
        for hour_of_day in range(25):  # Loop through all
            hours of the day
            total_area += calculate_sunlight_area(latitude,
                day_of_year, hour_of_day, room_orientation)
```

```
    return total_area

latitude = 23.0225  # Your latitude
gamma_range = range(361)  # Room orientation from 0 to 360
    degrees

total_areas = []
for gamma in gamma_range:
    total_area = calculate_total_sunlight_area(latitude,
        gamma)
    total_areas.append(total_area)

# Find the minimal point
min_index = np.argmin(total_areas)
min_orientation = gamma_range[min_index]
min_value = total_areas[min_index]

# Create a log plot
plt.semilogy(gamma_range, total_areas)
plt.xlabel('Room Orientation (gamma in degrees)')
plt.ylabel('Total Sunlight Area (sq. ft)')
plt.title('Total Sunlight Area vs. Room Orientation (Log
    Scale)')
plt.grid(True)

# Add a vertical line at the minimal point
plt.axvline(x=min_orientation, color='red', linestyle='--',
    label=f'Minimal Orientation: {min_orientation} degrees')
plt.legend()

plt.show()
```