Fermat's Principle, leading to the Laws of Reflection and Refraction

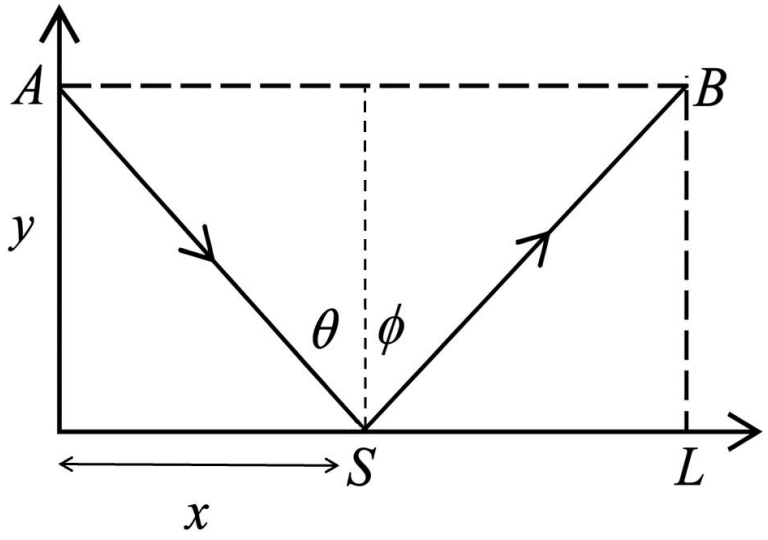
The study of light and optics is one of the oldest in Physics, and the two fundamental equations governing optics both come from Fermat's principle, or the principle of least time. As the latter suggests, it states that any ray of light passing from point A to B by any number of reflections and/or refractions will always take the path along which time is minimised. By creating equations in t and s, and solving when dt/ds = 0, one can derive Snell's law ($n_1 \sin\theta_1 = n_2 \sin\theta_2$) and the law of reflection ($\theta_i = \theta_r$). This paper intends to show this by mathematical rigor and also by using computer models created for the BPhO Computational Challenge 2025.

Before Fermat's principle, the velocity of light in various mediums was highly debated. Notably, Descartes believed that the velocity of light increased in denser mediums, as did Newton. The law of reflection was already known, ie that the angle of incidence always equals the angle of reflection for any given plane mirror, but the principle of least time helped explain why it was the case.

We can use computer modelling to show that Fermat's principle does indeed lead to the law of reflection, by creating formula from the distances travelled by light, and the time taken to get from point A to a plane mirror then to point B, and then solving for the minimum point. Doing so gives the following equation:

$$t = \frac{\sqrt{x^2 + y^2}}{c/n} + \frac{\sqrt{(L-x)^2 + y^2}}{c/n}$$

Taken from the following experimental set up:



Whereby the time taken for light to travel is calculated as the length of AS (Using Pythagoras) summed with the length of SB (Again using Pythagoras) all over the velocity of light in the medium (=c/n). Of course, for reflection the velocity in the medium does not matter, as long as the medium remains constant; n is inconsequential therefore, and as it remains constant, will not have any bearing on the final result of our calculation. The velocity of light (or indeed, wave) also does not matter: Fermat's theorem for reflection works with all waves, and notably he was one of the first to draw together the studies of light in reference to waves, and that of rays.

To start our computational proof, we define our constants (here n is stated as 1.5 initially, and later will take an input to show that the density of the medium does not matter for Fermat's principle to remain true.

Function computeT() will allow us to quickly and easily find the time taken for any given L and Y values.

```
let y = 2;
let L = 10;
let c = 299792458;
let n = 1.5;

let chart = null;

function computeT(x) {
    const denom = c / n;
    const d1 = Math.sqrt(x ** 2 + y ** 2);
    const d2 = Math.sqrt((L - x) ** 2 + y ** 2);
    return (d1 + d2) / denom;
}
```

To use Fermat's theorem to find the law of reflection, we must plot time against x, and find the minimum point of the graph (as The Principle of Least Time, evidently, requires light to take the path which requires the least time). If the value of x at the point which time is minimised is always L/2, we have our law of reflection: if the light ray meets the reflective surface at exactly half-way, it means that the angle of incidence always equals the angle of reflection (as Y remains constant, so if x = l-x, the light rays form two similar triangles with the same angles).

First, we plot the graph of t vs x, using two functions, one to compute data points in the range 0 to L, using a data array and.pushing each data point to it, before returning the whole array.

```
// Generate data points
function generateDataPoints(step = 0.01) {
    const data = [];
    for (let x = 0; x < L; x += step) {
        const t = computeT(x);
        data.push({ x: x, y: t });
    }
    // Ensure last point at x = L is included
    const tEnd = computeT(L);
    data.push({ x: L, y: tEnd });

    return data;
}
```

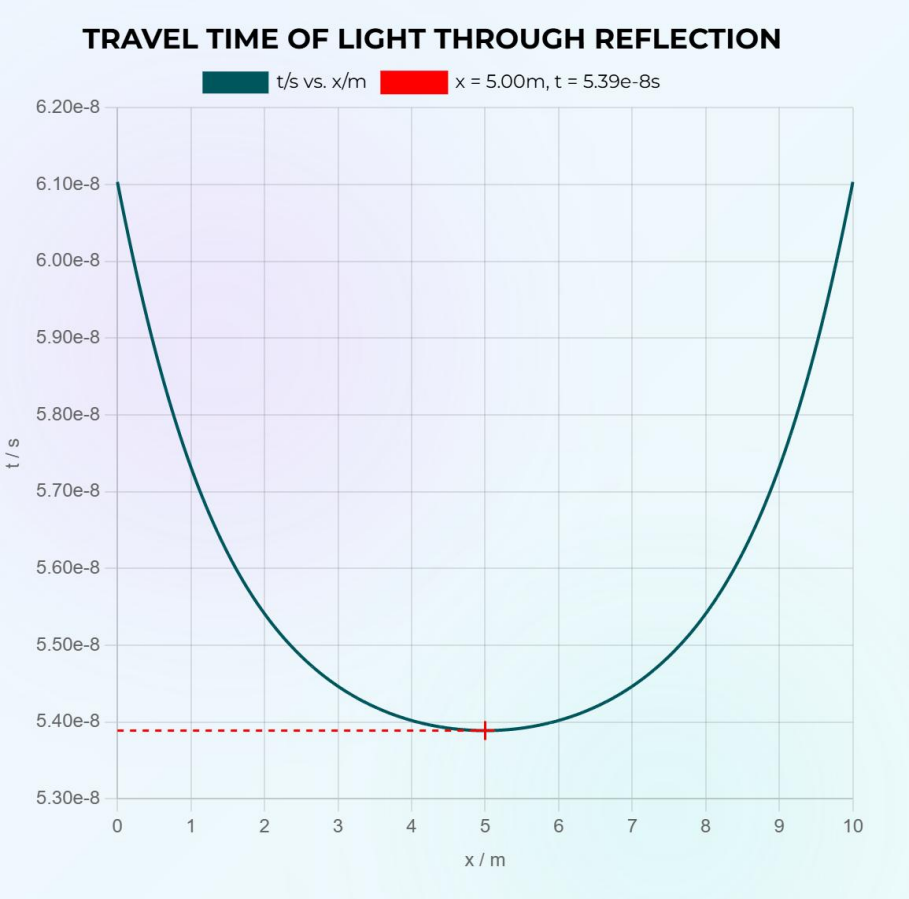
We also need a function to update the graph, including getting rid of old data points (from previous inputs), and plotting new ones:

```
function updateValuesAndChart() {
  // Update labels
  document.getElementById('yValue').textContent = y.toFixed(2);
  document.getElementById('LValue').textContent = L.toFixed(2);
  document.getElementById('nValue').textContent = n.toFixed(2);

  // Regenerate data
  const tData = generateDataPoints();
  const minPoint = tData.reduce((min, p) => (p.y < min.y ? p : min), tData[0]);

  // Update datasets
  chart.data.datasets[0].data = [{ x: minPoint.x, y: minPoint.y }];
  chart.data.datasets[0].label = `t = ${minPoint.y.toPrecision(3)} s`;

  chart.data.datasets[1].data = tData;

  chart.update();

    document.getElementById('minX').textContent = minPoint.x.toFixed(2);
    document.getElementById('minY').textContent = minPoint.y.toPrecision(3);
    document.getElementById('halfL').textContent = (L / 2).toFixed(2);
}
```

This will plot the necessary graph, but in order to fully show that Fermat's principle directly leads to the law of reflection, we need to find the point at which t is minimised. Mathematically, one would differentiate the equation for t, finding dt/dx, which would eventually lead us to the correct solution, but much easier to use a chart.js plugin, which finds the minimum point of the data array once the graph has been plotted, then marks it using a cross and dotted line so the user can see the exact value of x at the minimum time (an verify that it is always L/2).

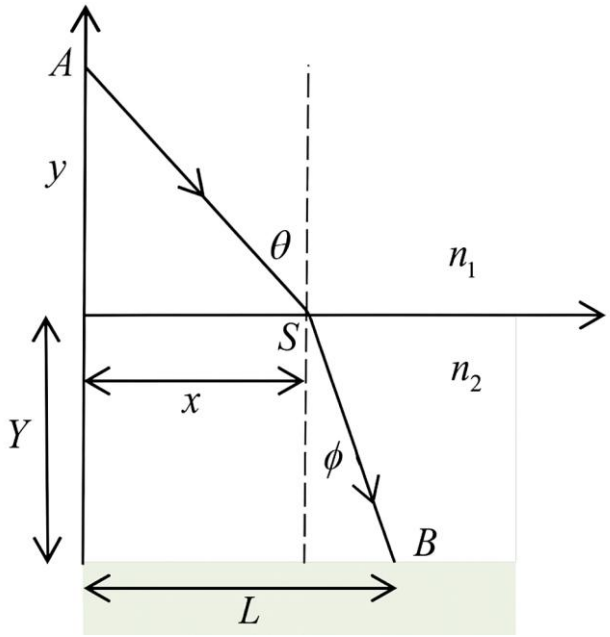
```
const highlightMinPlugin = {
  id: 'highlightMin',
  afterDraw(chart) {
    const crossDatasetIndex = chart.data.datasets.findIndex(ds => ds.isMinPoint);
    if (crossDatasetIndex === -1) return;

    const isVisible = chart.isDatasetVisible(crossDatasetIndex);

    // Set target opacity according to dataset visibility
    targetOpacity = isVisible ? 1 : 0;

    if (lastTimestamp === null) {
      lastTimestamp = performance.now();
      requestAnimationFrame(animateFade.bind(null, chart));
    }

    if (dashedLineOpacity > 0) {
      const minPoint = chart.data.datasets[crossDatasetIndex].data[0];
      const { ctx, scales: { x, y } } = chart;
      const px = x.getPixelForValue(minPoint.x);
      const py = y.getPixelForValue(minPoint.y);
      const x0 = x.left;

      ctx.save();
      ctx.globalAlpha = dashedLineOpacity;
      ctx.beginPath();
      ctx.moveTo(x0, py);
      ctx.lineTo(px, py);
      ctx.strokeStyle = 'rgb(255, 0, 0)';
      ctx.lineWidth = 1.5;
      ctx.setLineDash([4, 4]);
      ctx.stroke();
      ctx.restore();
    }
  }
};
```

With stylistic elements added to the chart.js function, which creates and plots the graph, as well as adding axis titles etc, the following graph is produced:



This graph is plotted at L=10, and clearly, the value of x at minimum time is 5. Having added various inputs (y, L and n), and varying these, one learns that no matter what combinations of these values, the minimum time is always taken at x = L/2. This means that regardless of distance, density of medium, and therefore velocity of the light, the law of reflection always holds true. Although this example is in 2D for the sake of simplicity of visualisation, of course this stays true in all real-world settings with plane mirrors, and Fermat provided a reason: light always takes the path that takes the least time.

One can also show that his property of light also implies Snell's law, ie that $n_1 \sin\theta_1 = n_2 \sin\theta_2$. We start again by proposing a common optics setup, where light travels from one medium of refractive index n_1 to another of refractive index n_2 at angle of incidence θ and refracts at angle φ:



From this, again we make an equation for time t taken for light to travel this distance. We use Pythagoras with the lengths x and y, and (L-x) and Y (or y_2) respectively to find the lengths of the paths travelled by the ray of light, then divide by the respective speed *in each medium*:

$$t = \frac{\sqrt{x^2 + {y_1}^2}}{c / n_1} + \frac{\sqrt{(L-x)^2 + {y_2}^2}}{c / n_2}$$

Coding this up is very similar to the reflection program, we first create a function to calculate values for t given inputs of refractive indices, c, x, y, and L:

```
let y1 = 2;
let y2 = 2;
let L = 10;
let c = 299792458;
let c1 = 200000000;
let c2 = 200000000;
let theta = 0;
let phi = 0;

let chart = null;

function computeT(x) {
    const n1 = c / c1;
    const n2 = c / c2;
    const d1 = Math.sqrt(x ** 2 + y1 ** 2);
    const d2 = Math.sqrt((L - x) ** 2 + y2 ** 2);
    const f1 = d1/(c / n1)
    const f2 = d2/(c / n2)
    return (f1 + f2);
}
```

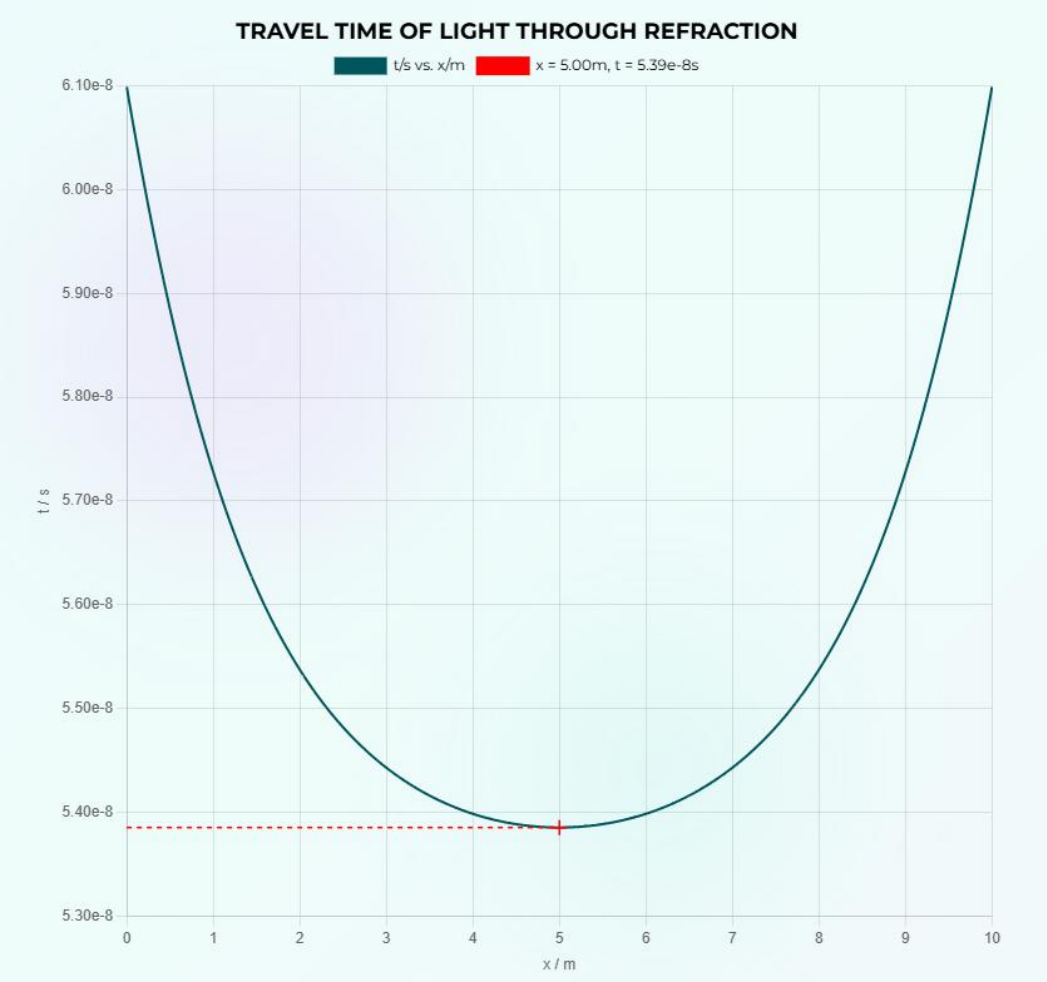
We then generate data points in the range of $0 \leq x \leq L$ (where L is set at the start of the program but then is dynamically adjusted by the user), by using a loop:

```
// Generate data points
function generateDataPoints(step = 0.01) {
    const data = [];
    for (let x = 0; x < L; x += step) {
        const t = computeT(x);
        data.push({ x: x, y: t });
    }
    // Ensure last point at x = L is included
    const tEnd = computeT(L);
    data.push({ x: L, y: tEnd });

    return data;
}
```

And returning data as an array. We plot this data, using chart.js, and create an update function which updates all the data, ranges of axes, etc.

```
function updateValuesAndChart() {
  // Update labels
  document.getElementById('y1Value').textContent = y1.toFixed(2);
  document.getElementById('y2Value').textContent = y2.toFixed(2);
  document.getElementById('LValue').textContent = L.toFixed(2);
  document.getElementById('c1Value').textContent = c1.toPrecision(3);
  document.getElementById('c2Value').textContent = c2.toPrecision(3);

  // Regenerate data
  const tData = generateDataPoints();
  const minPoint = tData.reduce((min, p) => (p.y < min.y ? p : min), tData[0]);

  // Update datasets
  chart.data.datasets[0].data = [{ x: minPoint.x, y: minPoint.y }];
  chart.data.datasets[0].label = `t = ${minPoint.y.toPrecision(3)} s`;

  chart.data.datasets[1].data = tData;

  chart.update();

  theta = Math.atan(minPoint.x / y1) * 180 / Math.PI;
  phi = Math.atan((L - minPoint.x) / y2) * 180 / Math.PI;

  document.getElementById('minX').textContent = minPoint.x.toFixed(2);
  document.getElementById('minY').textContent = minPoint.y.toPrecision(3);
  document.getElementById('theta').textContent = (theta).toFixed(2);
  document.getElementById('phi').textContent = (phi).toFixed(2);
  document.getElementById('sinThetaOverC1').textContent = (Math.sin(theta * Math.PI / 180)/c1).toPrecision(3);
  document.getElementById('sinPhiOverC2').textContent = (Math.sin(phi * Math.PI / 180)/c2).toPrecision(3);
}
```

We then get a chart like this:

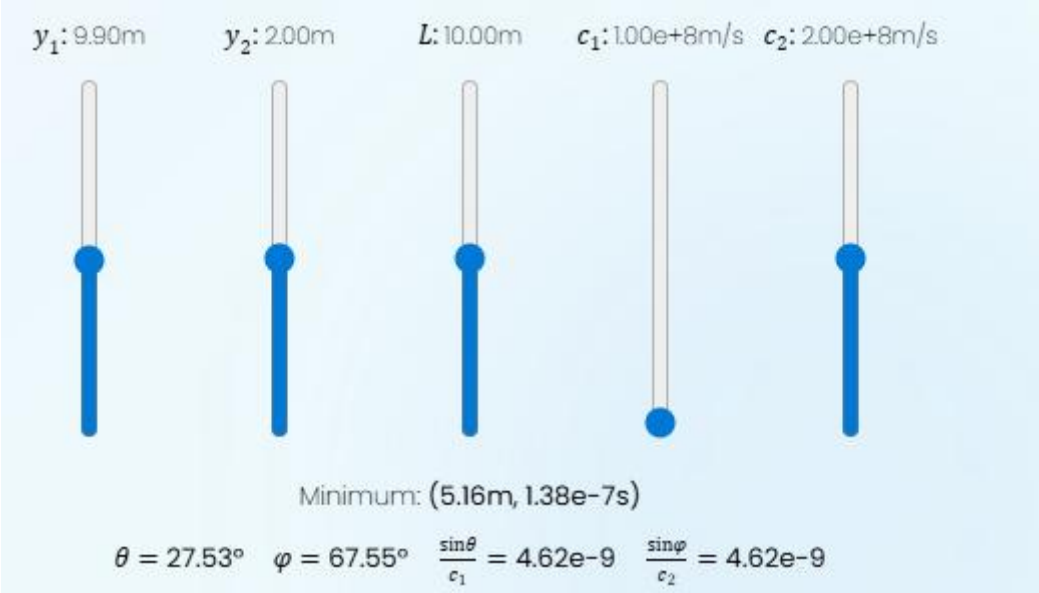


Having used the same HighlightMinPlugin to iterate until a minimum is found, then usefully highlight it using the red cross and dotted line, to indicate at what value of x the time is minimised. Then the input sliders are added, slightly more this time as not only the lengths are changeable but also both refractive indices (though the respective c(speed of light) values are in fact inputted). These are added in HMTL and linked up to the variables declared at the beginning of the Js script:

```
<div class="slider-container">
  <div class="slider-group">
    <div class="slider-label"><strong><math class="text-equation"><mi>y</mi></math>:</strong><span id="yValue" class="slider-value">2.00</span>m</div>
    <input type="range" id="sliderY" class="vertical" min="0.1" max="10" step="0.1" value="2" />
  </div>

  <div class="slider-group">
    <div class="slider-label"><strong><math class="text-equation"><mi>L</mi></math>:</strong><span id="LValue" class="slider-value">10.00</span>m</div>
    <input type="range" id="sliderL" class="vertical" min="1" max="20" step="0.1" value="10" />
  </div>

  <div class="slider-group">
    <div class="slider-label"><strong><math class="text-equation"><mi>n</mi></math>:</strong><span id="nValue" class="slider-value">1.50</span></div>
    <input type="range" id="sliderN" class="vertical" min="1" max="3" step="0.01" value="1.5" />
  </div>
</div>
```

```
//slider listeners
document.getElementById('sliderY').addEventListener('input', e => {
  y = parseFloat(e.target.value);
  updateValuesAndChart();
});

document.getElementById('sliderL').addEventListener('input', e => {
  L = parseFloat(e.target.value);
  updateValuesAndChart();
});

document.getElementById('sliderN').addEventListener('input', e => {
  n = parseFloat(e.target.value);
  updateValuesAndChart();
});
```

Having calculated which value of x t is minimised at, all that is left is to calculate θ and φ, their sines, and then the value of $\sin(\theta)/c_1$ and $\sin(\varphi)/c_2$, showing them to be always equal, regardless of y_1, y_2, L and both refractive indices:



As long as the minimum point of the graph is used to calculate these values. And so Snell's law is therefore discovered by working through the Maths of Fermat's principle, here computationally. This is because multiplying by 1/c is equivalent to multiplying by n. Fermat's principle also therefore shows that, if true, light travels faster in less dense mediums, with refractive index and speed being inversely proportional.

Therefore, Fermat's principle, the principle of least time, leads to both the law of reflection and the law of refraction, showing that Fermat has a good chance of being correct, about light travelling to choose the path to take the shortest possible time.

If you'd like to have a look at the models, they are task 3 and 4 on https://opticalcalculator.com.

All illustrations from either opticalcalculator.com or the 2025 BPhO Computational Challenge task sheets, which are available on https://opticalcalculator.com/about.html.