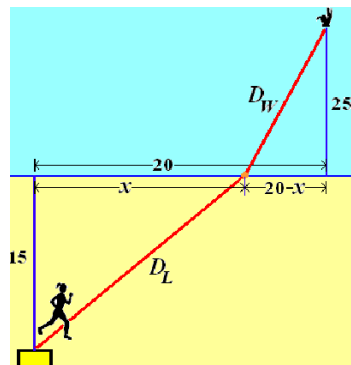


Multi-Layer Refraction Problem

Rafael Espericueta, Bakersfield College, November, 2006

Light travels at different speeds through different media, but refracts at layer boundaries in order to traverse the least-time path. In this paper we'll play light's game, finding the least-time path through a number of layers of media, where each media induces a different speed.

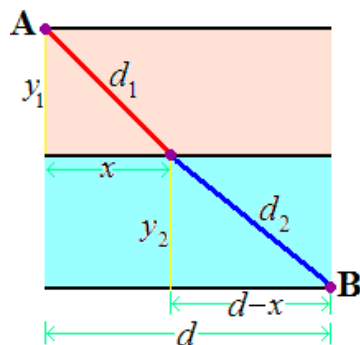
One example of such a problem is the lifeguard problem sometimes seen in calculus courses. The lifeguard can travel at a given speed running through sand and a different speed swimming, and the problem is to find the path that minimizes the time it takes to reach a drowning child.



2-Layer Problem

We start by solving the general two-layer case.

Given values y_1 , y_2 , and d , as shown in the figure, and also given the fixed speeds v_1 and v_2 through each layer, we seek the unique value of x resulting in the least time path from **A** to **B**.



As shown in the figure, let

$$d_1 = \sqrt{x^2 + y_1^2} \quad \text{and} \quad d_2 = \sqrt{(d-x)^2 + y_2^2}, \quad \text{and let}$$

$$t_1 = \frac{d_1}{v_1} = \frac{\sqrt{x^2 + y_1^2}}{v_1}, \quad \text{and} \quad t_2 = \frac{d_2}{v_2} = \frac{\sqrt{(d-x)^2 + y_2^2}}{v_2}$$

(These are the times it takes to traverse the two layers)

We wish to minimize the total travel time function f defined by

$$t_1 + t_2 = \frac{d_1}{v_1} + \frac{d_2}{v_2} = \frac{\sqrt{x^2 + y_1^2}}{v_1} + \frac{\sqrt{(d-x)^2 + y_2^2}}{v_2} = f(x),$$

i.e., we must find the minimum of the function

$$f(x) = \frac{1}{v_1} (x^2 + y_1^2)^{\frac{1}{2}} + \frac{1}{v_2} ((d-x)^2 + y_2^2)^{\frac{1}{2}}$$

Taking the derivative and setting it equal to zero, we obtain:

$$f'(x) = \frac{1}{2v_1} (x^2 + y_1^2)^{-\frac{1}{2}} \cdot 2x + \frac{1}{2v_2} ((d-x)^2 + y_2^2)^{-\frac{1}{2}} \cdot 2(d-x)(-1) = 0$$

$$\frac{x}{v_1 \sqrt{x^2 + y_1^2}} - \frac{d-x}{v_2 \sqrt{(d-x)^2 + y_2^2}} = 0$$

$$x \cdot v_2 \sqrt{(d-x)^2 + y_2^2} - (d-x) \cdot v_1 \sqrt{x^2 + y_1^2} = 0$$

$$x \cdot v_2 \sqrt{(d-x)^2 + y_2^2} = (d-x) \cdot v_1 \sqrt{x^2 + y_1^2}$$

$$x^2 v_2^2 ((d-x)^2 + y_2^2) = (d-x)^2 v_1^2 (x^2 + y_1^2)$$

...

$$(v_2^2 - v_1^2)x^4 + 2d(v_1^2 - v_2^2)x^3 + (v_2^2 d^2 + v_2^2 y_2^2 - v_1^2 d^2 - v_1^2 y_1^2)x^2 + 2v_1^2 d y_1^2 x - v_1^2 d^2 y_1^2 = 0$$

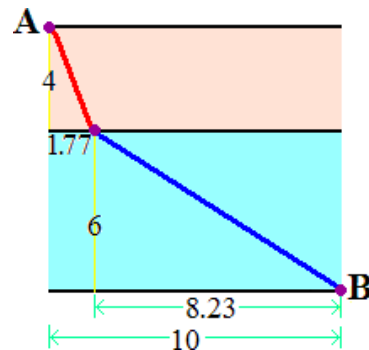
Being a fourth degree polynomial, it can be solved exactly, with the help of a computer algebra system. The solution is rather complicated, but numerical methods need not be invoked.

As a particular case, let $y_1 = 4$, $y_2 = 6$, $d = 10$, $v_1 = 1$, and $v_2 = 2$. Then Maple yields 4 exact solutions, of which two are real and two are complex. One of the real solutions for x is negative, and the other is the one we seek, which equals exactly

$$-\frac{2}{3} \frac{(529 + 10\sqrt{2867})^{2/3} - 19 - 11(529 + 10\sqrt{2867})^{1/3}}{(529 + 10\sqrt{2867})^{1/3}},$$

approximately equal to 1.7669994927838833928436718067469630557074373841252.

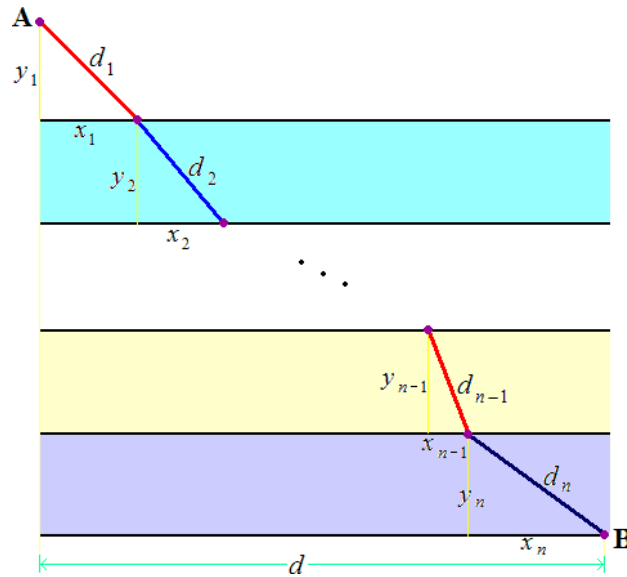
Notice how much longer the path is through the second layer, where the speed is doubled, in order to minimize the total time of the trip.



n-Layer Problem

Next we turn to the general, n -layer case. In the general case we must optimize a function of several variables. Closed form solutions can't be found; we must turn to numerical methods.

For fixed positive y_i , v_i , d , where v_i is the speed through the i^{th} layer, which has thickness y_i , as shown in the figure.



Notice that the distance from **A** to **B** is $\sqrt{d^2 + \left(\sum_{i=1}^n y_i\right)^2} = \sqrt{\left(\sum_{i=1}^n x_i\right)^2 + \left(\sum_{i=1}^n y_i\right)^2}$.

Let

$$d_i = \sqrt{x_i^2 + y_i^2}, \quad \text{for } i = 1, \dots, n.$$

(The lengths of the least-time path through each layer),

and let

$$t_i = \frac{d_i}{v_i}, \quad \text{for } i = 1, \dots, n.$$

(The times it takes to traverse the layers)

We wish to minimize

$$\sum_{i=1}^n t_i = \sum_{i=1}^n \frac{d_i}{v_i} = \sum_{i=1}^n \frac{\sqrt{x_i^2 + y_i^2}}{v_i} = f(x_1, \dots, x_n)$$

subject to the constraints $G(x_1, \dots, x_n) = \sum_{i=1}^n x_i = d$.

(Amazingly, each little photon solves this problem instantaneously!)

We have:

$$\begin{aligned}\nabla f &= \nabla \sum_{i=1}^n \frac{\sqrt{x_i^2 + y_i^2}}{v_i} = \sum_{i=1}^n \nabla \frac{\sqrt{x_i^2 + y_i^2}}{v_i} = \sum_{i=1}^n \frac{1}{v_i} \nabla (x_i^2 + y_i^2)^{\frac{1}{2}} \\ \nabla f &= \sum_{i=1}^n \left(\frac{1}{v_i} \cdot \frac{x_i}{(x_i^2 + y_i^2)^{\frac{1}{2}}} \hat{e}_i \right) = \sum_{i=1}^n \left(\frac{x_i}{v_i (x_i^2 + y_i^2)^{\frac{1}{2}}} \hat{e}_i \right) \\ \nabla G &= \nabla \sum_{i=1}^n x_i = \sum_{i=1}^n \nabla x_i = \sum_{i=1}^n \hat{e}_i = \langle 1, 1, \dots, 1 \rangle\end{aligned}$$

Invoking the method of Lagrange multipliers, we have $\nabla f = \lambda \nabla G$, hence:

$$\begin{aligned}\frac{x_i}{v_i (x_i^2 + y_i^2)^{\frac{1}{2}}} &= \lambda, \quad \text{for } i = 1, \dots, n \\ x_i &= \lambda v_i (x_i^2 + y_i^2)^{\frac{1}{2}} \\ x_i^2 &= \lambda^2 v_i^2 (x_i^2 + y_i^2) \\ (1 - \lambda^2 v_i^2) x_i^2 &= \lambda^2 v_i^2 y_i^2 \\ x_i^2 &= \frac{\lambda^2 v_i^2 y_i^2}{1 - \lambda^2 v_i^2} \Rightarrow x_i = \frac{\lambda v_i y_i}{\sqrt{1 - \lambda^2 v_i^2}} = \frac{y_i}{\sqrt{\lambda^{-2} v_i^{-2} - 1}} \\ \boxed{\Psi} \quad x_i &= y_i (\lambda^{-2} v_i^{-2} - 1)^{-\frac{1}{2}}\end{aligned}$$

This implies that $\lambda < \frac{1}{v_i}$, $\forall i$. λ must be less than the least of the reciprocals of the v_i (else the distances become complex numbers). Plugging these values into the constraint equation, we get:

$$\sum_{i=1}^n x_i = \sum_{i=1}^n y_i (\lambda^{-2} v_i^{-2} - 1)^{-\frac{1}{2}} = d$$

Let $\psi(x) = \sum_{i=1}^n y_i (v_i^{-2} x^{-2} - 1)^{-\frac{1}{2}} - d$. Then the Newton transform of ψ will have iterates that tend to converge to our desired λ . Since

$$\psi'(x) = \sum_{i=1}^n (v_i^{-2} x^{-2} - 1)^{-\frac{3}{2}} v_i^{-2} x^{-3},$$

the Newton transform is given by :


```

x := [5.1553156251724693223215255592392545300636527735250
15.093118415087759023504717094470732983405981120693
1.4297695018538734547432689693892438604672811030468
18.321796457885898199430488376900768626063085002731

```

Graph the optimal path through the layers:

```

> with(plots) : i := 'i':
Ysum := sum(y[i], i = 1 .. n) :
X[0] := 0 : Y[0] := Ysum :
for i from 1 to n
do
  Y[i] := Y[i-1]-y[i] : X[i] := X[i - 1] + x[i] :
od:

for i from 1 to n
do
  THICK := [op(THICK), 3] :
  if (i mod 2 > 0) then COL := [op(COL), 'red']
  else COL := [op(COL), 'blue'] :
  fi:
od:

COL := [seq('black', i = 0 .. n)] : THICK := [seq(2, i = 0 .. n)] :
for i from 1 to 2
do
  COL := [op(COL), 'yellow'] : THICK := [op(THICK), 1] :
od:

plot([seq([t, Y[i], t = 0 .. Dx], i = 0 .. n),
[Dx, t, t = 0 .. Ysum], [0, t, t = 0 .. Ysum],
seq([t*(X[i + 1]-X[i]) + X[i], t*(Y[i + 1]-Y[i])
+ Y[i], t = 0 .. 1], i = 0 .. (n - 1))],
color = COL, thickness = THICK, axes = none)

```

