Revista Brasileira de Ensino de Física
https://doi.org/10.1590/1806-9126-RBEF-2025-0133

Material Suplementar para "Algoritmo de Refração: Uma proposta didática para investigação do problema de minimização do tempo"

Anexo C - Código completo em Octave

```
clear, clc


% --- AJUSTE DE PARÂMETROS ---
% --- definição de n1 e n2 e o número de valores de x0 a ser
    variado

n1=1.00;       % altere aqui o índice de refração do meio 1
n2=1.33;      % altere aqui o índice de refração do meio 2
numero = 20; % altere aqui o número de valores de x0 a ser
    variado

printf("\n índice de refração do meio 1 - n1: %f ", n1);
printf("\n índice de refração do meio 2 – n2:  %f", n2);
printf("\n número de valores de x0 a ser variado:  %d",
    numero);

% verificando a consistência das informações de entrada
if numero != round(numero) || numero < 0
  error('O 'número de valores de x0 a ser variado' deve ser um
    inteiro positivo.');
endif
```

```
if  n1 < 1
  error('O valor de "n1" deve ser um inteiro positivo, maior
    que 1.');
endif

if n2 < 1
  error('O valor de "n1" deve ser um inteiro positivo, maior
    que 1.');
endif

printf("\n\n------- DADOS VALIDADOS! ------- \n");

% =====  coordenadas dos pontos cartesianos

p=[-1,1];
q=[1,-1];
c=299792458; % (m/s) velocidade da luz no vácuo



% =====  definindo o passo adotado para cada caminho calculado

passo=1/numero;
xi=[];
          % lista de abscissas, xi
tempo=[];   % lista de tempos de percurso
lista=[];   % matriz com os valores de xi, tempos de percurso e
    espaço percorrido
```

```
% ===== VALORES DE xi

printf("\n\n------- VALORES DE xi ------- \n");

for i = -numero: numero
xi=[xi;i*passo]; % inserindo valores no final de xi
o=[i*passo,0];
printf("xi = %d\n", i*passo);

% cálculo de v1 e v2
v1=c/n1;
v2=c/n2;

% cálculo do s1, s2 e s
s1=sqrt((p(1)-o(1))^2+(p(2)-o(2))^2);
s2=sqrt((q(1)-o(1))^2+(q(2)-o(2))^2);
s=s1+s2

% para tempo t=t1+t2=(s1/v1)+(s2/v2)
t1=s1/v1;
t2=s2/v2;
t=t1+t2;
tempo=[tempo;t]; % lista de tempos
lista=[lista;i*passo,t,s]; % linkando os tempos com os pontos
printf("\n---------------------- \n");
end
final=size(xi)(1)
```

```
% ===== construindo os caminhos da luz em um gráfico

printf("\n------- CONSTRUINDO OS CAMINHOS DA LUZ ------- \n");
figure (1);
clf;
 line ([-1.5 1.5], [0 0], "linestyle", "-.", "color", "k");
 line ([0 0], [-1.5 1.5], "linestyle", "-.", "color", "k");
 line ([-1.5 1.5], [1.5 1.5], "linestyle", "-", "color", "k");
    % linha fechar gráfico
 line ([1.5 1.5], [1.5 -1.5], "linestyle", "-", "color", "k");
    %linha fechar gráfico

% legendas e anotações: gráfico 1
text(1,0.5,"Meio 1","fontsize",12)
text(-1,-0.5,"Meio 2","fontsize",12)
text(-1.1,1.1,"P","fontsize",16)
text(1,-1.1,"Q","fontsize",16)
xlabel ("Posição em x(m)", "fontsize",12)
ylabel ("Posição em y(m)", "fontsize",12)
title ("Caminho da luz entre os pontos 'P' e
    'Q'","fontsize",12)

% ===== organizando os dados por ordem de menor tempo

ordem=sortrows(lista, 2);
cores=jet(final);
for j = 1:final
  line ([p(1) ordem(j,1)], [p(2) o(2)], "linestyle", "-",
```

```
    "color", cores(j,:));
  line ([ordem(j,1) q(1)], [o(2) q(2)], "linestyle", "-",
    "color", cores(j,:));
end
line ([p(1) ordem(1,1)], [p(2) o(2)], "linestyle", "-",
    "color", "k", "linewidth", 2);
line ([ordem(1,1) q(1)], [o(2) q(2)], "linestyle", "-",
    "color", "k", "linewidth", 2);
colormap (jet)
h = colorbar ();
 ytick = get (h, "ytick");
escala=[ordem(1,2),ordem(ceil(final/5),2),ordem(ceil(2*final/5)
    ,2),ordem(ceil(3*final/5),2),ordem(ceil(4*final/5),2),ordem
    (final,2)];
 set (h, "yticklabel", escala);
 ylim ([-1.5 1.5]);
 text(1.9,1.65,"tempo (s)","fontsize",12)
saveas (1,"figure1.png") % salvar imagem

% ===== plotando tempo (s) vs. Xi

printf("\n------- Plotando Tempo vs. Xi ------- \n");
figure (2);
clf;
 plot(xi, tempo, 'o-r');
 xlabel ("xi (m)", "fontsize",12);
 ylabel ("t (s)", "fontsize",12);
 title ("Tempo do percurso vs. Posição xi","fontsize",12)
```

```
 saveas (2,"figure2.png")

% ===== comparando o resultado de minimização de tempo com a
    lei de snell

printf("\n------- Comparação: resultado de minimização de tempo
    vs. lei de Snell ------- \n");

x0=ordem(1,1); % menor valor de tempo
 seno1=abs((p(1)-x0)/sqrt((p(1)-x0)^2+(p(2)-o(2))^2));
 seno2=abs((q(1)-x0)/sqrt((q(1)-x0)^2+(q(2)-o(2))^2));
printf("x0 = %d\n", x0);
printf("seno1/seno2 = %d\n", seno1/seno2);
printf("n2/n1 = %d\n", n2/n1);
```