Física estatística computacional

Equações de onda

Aluno
Leonardo Vaz Ferreira n° 13862330

São Carlos
2025

1 Tarefa 1

Na primeira parte do trabalho, foi proposta a criação de um programa em Fortran, com o intuito de analisar a propagação de uma onda de um pacote gaussiano, com velocidade c, em um meio não dissipativo de comprimento L, com as extremidades fixas, da forma:

$$Y(x,0) = Y_0(x) = exp[\frac{-(x-x_o)^2}{\sigma^2}] \tag{1}$$

com $x_0 = L/3$ e $\sigma = L/30$ e para aplicação nos testes $L = 1$ e $c = 300$.

Para isso, foi usado o método iterativo da seguinte forma para calcular a posição da onda em cada instante de tempo:

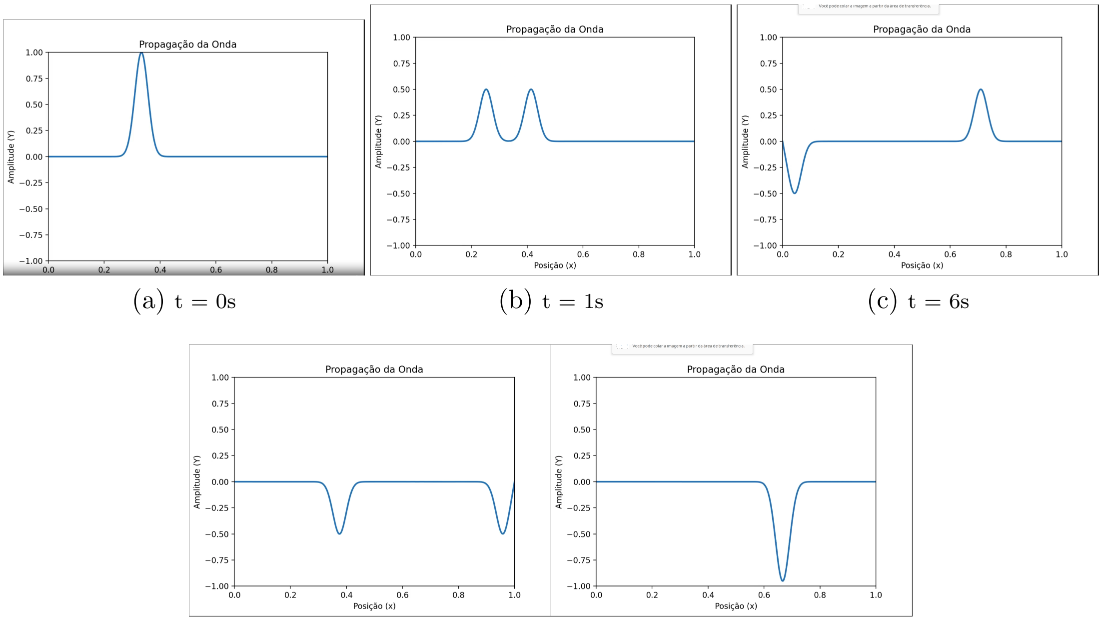
$$Y(i,n+1) = 2(1-r^2)Y(i,n) + r^2\left[Y(i+1,n) + Y(i-1,n)\right] - Y(i,n-1) \tag{2}$$

dessa forma, foi elaborado o seguinte código.

```fortran
program tarefa1
  implicit real*8 (a-h, o-z)
  integer, parameter :: max = 1000
  real*8 :: x(0:max), y_prev(0:max), y_curr(0:max), y_next(0:max)

  open(unit = 10, file="saida_1_13862330.out")

  L = 1.0d0
  c = 300.0d0
  dx = L / real(max)
  dt = dx / c
  r = 0.25e0 * c * dt / dx
  Nt = 1000

  do n_max = 1,1000
    do i = 0, max
      x(i) = i * dx
    end do

    do i = 0, max
      y_curr(i) = f(x(i), L)
    end do

    y_curr(0) = 0.0d0
    y_curr(max) = 0.0d0

    y_prev = y_curr

    do n = 1, Nt
      do i = 1, max-1
        y_next(i) = 2.0d0 * (1 - r**2) * y_curr(i) + r**2 * (y_curr(i+1) + y_curr(i-1)) - y_prev(i)
      end do

      y_next(0) = 0.0d0
      y_next(max) = 0.0d0

      y_prev = y_curr
      y_curr = y_next

      if (n == n_max) then
        do i = 0, max
          write(10, *) x(i), y_curr(i)
        end do
      end if
    end do
  end do

  close(10)
end program

function f(x, L) result(y)
  implicit real*8 (a-h, o-z)
  x_0 = L / 3.0d0
  s = L / 30.0d0
  y = exp(-(x - x_0)**2.0d0 / (s**2.0d0))
end function f
```

Figura 1: Código da primeira tarefa

Com o código da Fig.1 em mão, com os seguintes parâmetros definidos; valor infinitesimal de deslocamento $\Delta x = 0.001$, e o valor de equilíbrio $r = 1$. Podemos então obter como ocorre a propagação do pacote de onda ao longo do tempo.



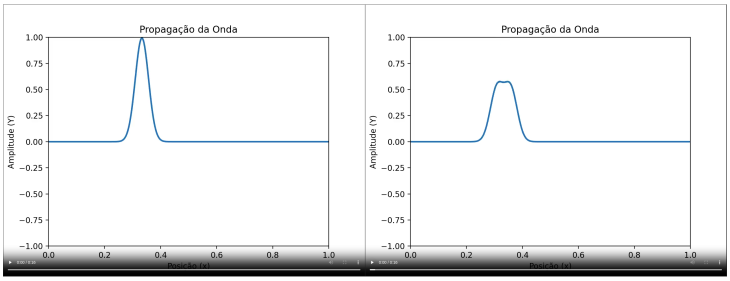
(a) t = 0s (b) t = 1s (c) t = 6s

(d) t = 11s (e) t = 16s

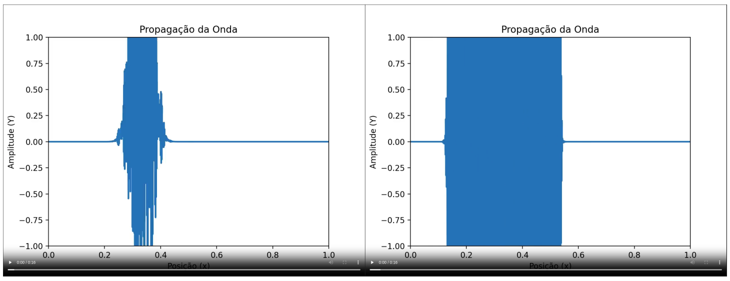
Figura 2: Evolução temporal do pacote gaussiano com $r = 1$

Como podemos observar na Fig.2, conseguimos notar que o pacote centrado em $L/3$ no começo da simulação permanece como uma gaussiana, logo após, ele se divide em dois, se propagando para os dois lados e, ao atingir uma das extremidades presas, o pulso de onda é espelhado de volta, de maneira que ele volte virado para baixo, dessa maneira, ao final da simulação as duas ondas se encontram, causando uma interferência construtiva e assim, a onda volta à sua amplitude inicial, porém espelhada verticalmente e horizontalmente. E caso fosse simulado por mais tempo, a onda retornaria ao formato inicial.

Podemos também analisar diferentes valores de r para analisar o comportamento da onda, primeiramente analisaremos para $r = 2$.



(a) t = 0s (b) t = 0.1s

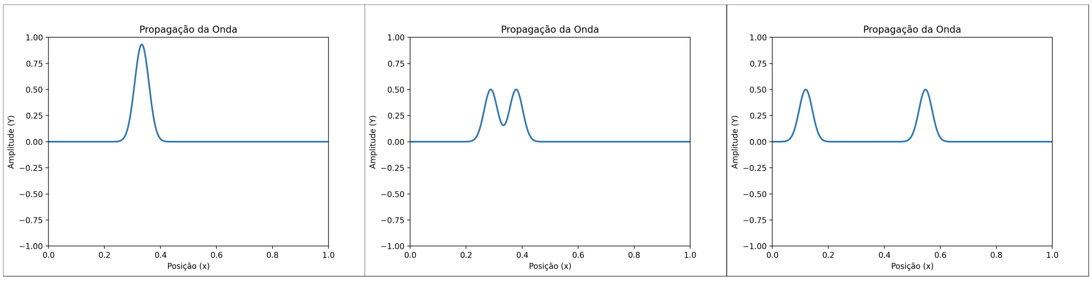


(c) t = 0.2s (d) t = 0.3s

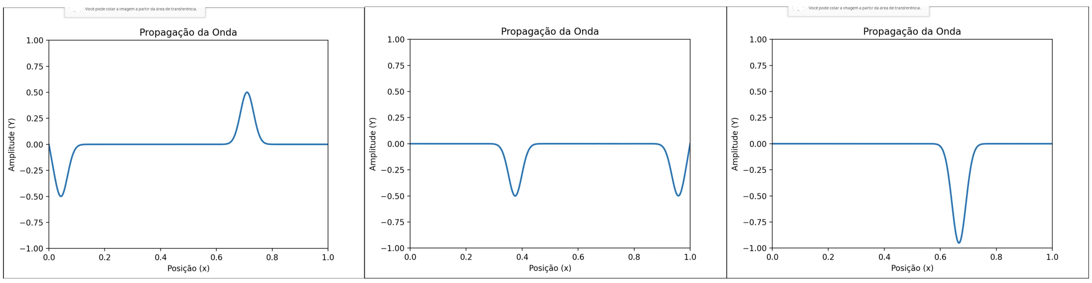
Figura 3: Evolução temporal do pacote gaussiano com $r = 2$

Podemos notar na Fig.3 que no começo da simulação o pacote de onda se mantém igual à simulação anterior, com a diferença que a movimentação ocorre de maneira mais rápida que a anterior. Observa-se também que, com o passar da simulação, a alteração do parâmetro r causa uma instabilidade da simulação numérica e, dessa maneira, impossibilitando a simulação.

Já quando alteramos o parâmetro para $r = 0.25$ obtemos os seguinte resultados:



(a) t = 0s (b) t = 3s (c) t = 15s



(d) t = 30s (e) t = 42s (f) t = 60s

Figura 4: Evolução temporal do pacote gaussiano com $r = 0.25$

Já quando alteramos o parâmetro para $r = 0.25$ segundo a Fig.13, podemos notar que a propagação se comporta de forma semelhante ao caso que $r = 1$, mas com a diferença na velocidade da propagação da onda, sendo que, quando $r = 0.25$ a simulação demora muito mais, visto que o passo temporal N_t precisa ser maior para simular o mesmo tempo.

2 Tarefa 2

Para a segunda parte do projeto, foi proposto ao invés de trabalharmos com uma onda gaussiana, deveríamos trabalhar com uma onda de violão da seguinte maneira:

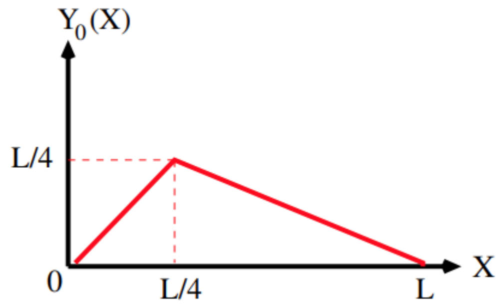


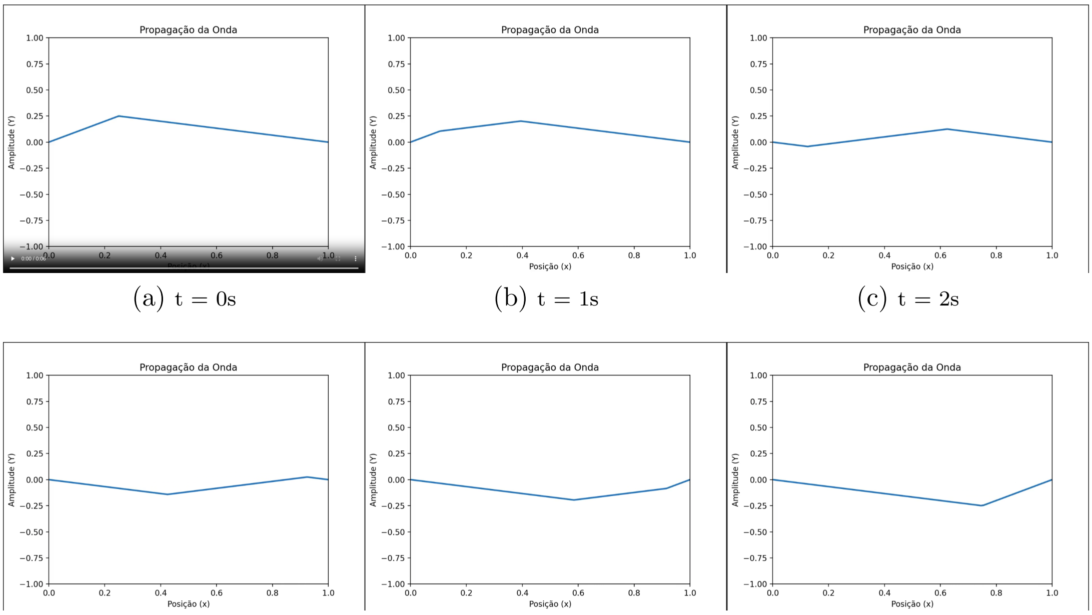
Figura 5: Onda de violão

Dessa maneira, foi elaborado o seguinte código:

```fortran
program tarefa2
    implicit real*8 (a-h, o-z)
    integer, parameter :: max = 1000
    real*8 :: x(0:max), y_prev(0:max), y_curr(0:max), y_next(0:max)

    open(unit = 10, file="saida_1_13862330.out")
    open(unit = 20, file="saida_2_13862330.out")

    L = 1.0d0
    c = 300.0d0
    dx = L / real(max)
    dt = dx / c
    r = c * dt / dx
    Nt = 1000

    do n_max = 1,1000
        do i = 0, max
            x(i) = i * dx
        end do

        do i = 0, max
            y_curr(i) = f(x(i), L)
        end do

        y_curr(0) = 0.0d0
        y_curr(max) = 0.0d0

        y_prev = y_curr

        do n = 1, Nt
            do i = 1, max-1
                y_next(i) = 2.0d0 * (1 - r**2) * y_curr(i) + r**2 * (y_curr(i+1) + y_curr(i-1)) - y_prev(i)
            end do

            y_next(0) = 0.0d0
            y_next(max) = 0.0d0

            y_prev = y_curr
            y_curr = y_next

            if (n.EQ.n_max) then
                do i = 0, max
                    if (int(x(i)*1000).EQ.250) then
                        write(20, *) n, y_curr(i)
                    end if
                    write(10, *) x(i), y_curr(i)
                end do
            end if
        end do
    end do

    close(10)
end program

function f(x, L) result(y)
    implicit real*8 (a-h, o-z)

    if (x >= 0.0d0 .and. x <= 0.25d0) then
        y = x
    else if (x > 0.25d0 .and. x <= 1.0d0) then
        y = -1.0d0 / 3.0d0 * x + 1.0d0 / 3.0d0
    end if
end function
```

Figura 6: Código da segunda tarefa

Assim, obtivemos os seguintes resultados:



(a) t = 0s (b) t = 1s (c) t = 2s

(d) t = 3s (e) t = 4s (f) t = 5s

Figura 7: Evolução temporal do da onda de violão

Semelhante à tarefa anterior, foi utilizado $r = 1$ para a realização dessa simulação, assim podemos ver o deslocamento da onda muito semelhante ao anterior, com apenas a diferença do formato, assim da mesma maneira ela segue sendo refletida nas extremidades por elas serem mantidas fixas, além disso e ela ainda mostra uma interferência construtiva no final da simulação quando as duas ondas espelhadas se encontram e da mesma maneira, caso fosse gerado o sobro do tempo da simulação a onda se encontraria da mesma maneira que iniciou.

3 Tarefa 3

Com os resultados das tarefas anteriores em mãos, podemos analisar o comportamento de diferentes maneiras. Uma delas pode ser analisar o espectro de potências através da Transformada de Fourier dos resultados obtidos de um ponto específico do pulso gaussiano em tempos diferentes.

$$P(f) = Y^s(f)^2 + Y^c(f)^2 \tag{3}$$

```fortran
program tarefa2
   implicit real*8 (a-h, o-z)
   integer, parameter :: max = 1000
   real*8 :: x(0:max), y_prev(0:max), y_curr(0:max), y_next(0:max)

   open(unit = 10, file="saida_k1_13862330.out")
   open(unit = 20, file="saida_2_k1_13862330.out")

   L = 1.0d0
   c = 300.0d0
   dx = L / real(max)
   dt = dx / c
   r = c * dt / dx
   Nt = 1000

   do n_max = 1,1000
      do i = 0, max
         x(i) = i * dx
      end do

      do i = 0, max
         y_curr(i) = f(x(i), L, 1)
      end do

      y_curr(0) = 0.0d0
      y_curr(max) = 0.0d0

      y_prev = y_curr

      do n = 1, Nt
         do i = 1, max-1
            y_next(i) = 2.0d0 * (1 - r**2) * y_curr(i) + r**2 * (y_curr(i+1) + y_curr(i-1)) - y_prev(i)
         end do

         y_next(0) = 0.0d0
         y_next(max) = 0.0d0

         df = 1.0d0 / (n * t)

         y_prev = y_curr
         y_curr = y_next

         if (n.EQ.n_max) then
            do i = 0, max
               if (int(x(i)*1000).EQ.250) then
                 write(20, *) i*df, y_curr(i)
               end if
               write(10, *) x(i), y_curr(i)
            end do
         end if
      end do
   end do

   close(10)
end program

function f(x, L, k) result(y)
   implicit real*8 (a-h, o-z)
   x_0 = L / 2.0d0
   s = L / 30.0d0

   y = 0.0d0

   do j = 1, k
      x0 = (2.0d0*j - 1.0d0) * L / (2.0d0 * k)
      y = y + (-1.0d0)**(j+1) * exp(-((x - x0)**2.0d0) / (s**2.0d0))
   end do
end function
```

Figura 8: Código da tarefa 3 (a-d)

A partir do código da Fig.13, foram coletados os dados de $x = L/4$ em diferentes tempos da simulação, dessa maneira, utilizando o código da Transformada de Fourier do primeiro projeto, foi possível obter o seguinte espectro de potência:

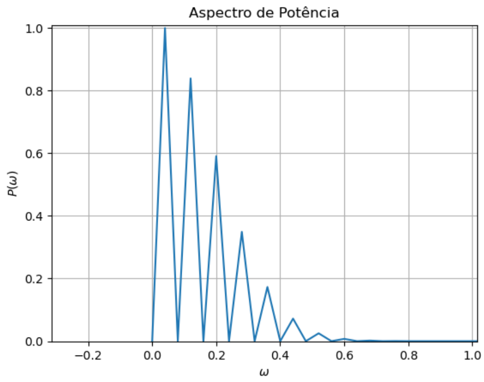
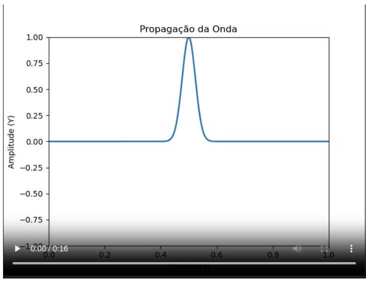


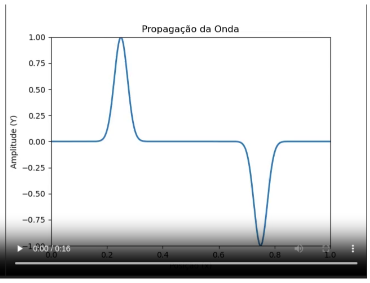
Figura 9: Série de potências de $x = L/4$ do pulso gaussiano

Podemos observar que o aspectro de potência está condizente com o pacote de onda que foi analisado, já que podemos ver a curva da gaussina no aspectro, e podemos notar que os primeiros são maiores, já que representam a parte central da gaussiana onde o pulso tem maior aplitude.

Podemos também fazer a mesma análise para diferentes harmônicos aplicados na corda, tais como:

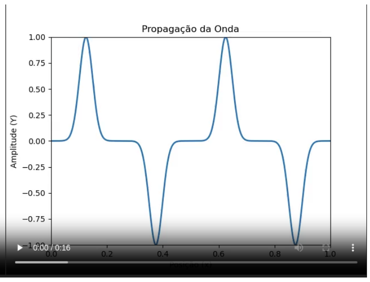


(a) λ_1



(b) λ_2

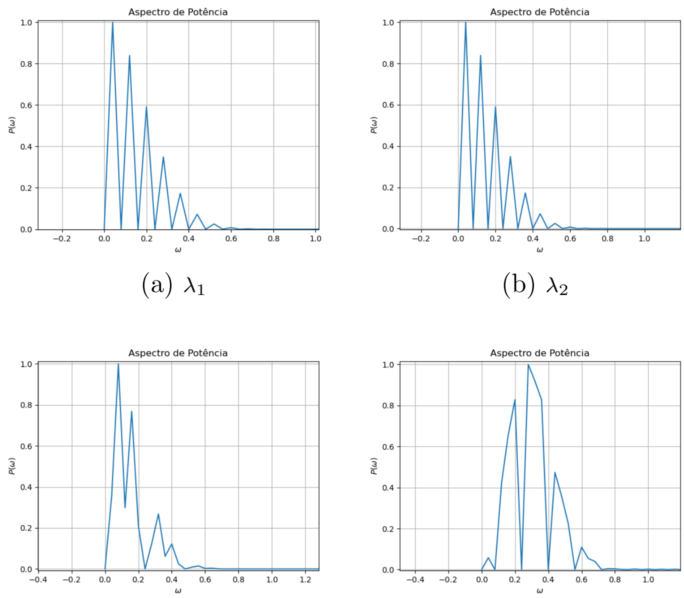
(c) λ_3



(d) λ_4

Figura 10: Diferentes harmônicos gaussianos

Assim, podemos obter os seguintes aspectros de potência:



(a) λ_1 (b) λ_2

(c) λ_3 (d) λ_4

Figura 11: Aspectro de potências em diferentes harmônicos gaussianos

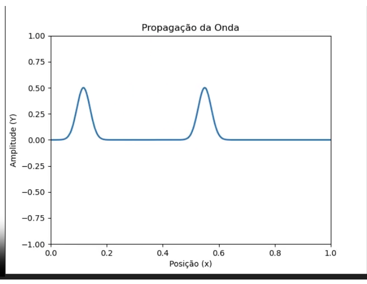
Podemos notar que os dois primeiros se mantém parecidos, mas a partir do terceiro algumas potências começam a sumir ao longo do gráfico, como o Fig.3.c, onde a frequência próximo a 0.2 foi cortada e na Fig.3.d o aspectro começa a ficar estranho por conta de muitas interferências das ondas ao longo da simulação.

Além disso, podemos alterar as condições de contorno da corda, para dessa maneira deixar uma de suas extremidades solta.

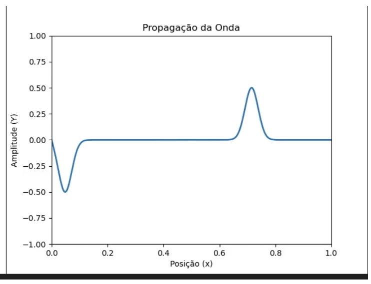
```fortran
program tarefa3e
    implicit real*8 (a-h, o-z)
    integer, parameter :: max = 1000
    real*8 :: x(0:max), y_prev(0:max), y_curr(0:max), y_next(0:max)

    open(unit = 10, file="saida_3e_13862330.out")

    L = 1.0d0
    c = 300.0d0
    dx = L / real(max)
    dt = dx / c
    r = c * dt / dx
    Nt = 1000

    do n_max = 1,1000
        do i = 0, max
            x(i) = i * dx
        end do

        do i = 0, max
            y_curr(i) = f(x(i), L)
        end do

        y_curr(0) = 0.0d0
        y_next(max) = y_next(max-1)

        y_prev = y_curr

        do n = 1, Nt
            do i = 1, max-1
                y_next(i) = 2.0d0 * (1 - r**2) * y_curr(i) + r**2 * (y_curr(i+1) + y_curr(i-1)) - y_prev(i)
            end do

            y_next(0) = 0.0d0
            y_next(max) = y_next(max-1)

            y_prev = y_curr
            y_curr = y_next

            if (n == n_max) then
                do i = 0, max
                    write(10, *) x(i), y_curr(i)
                end do
            end if
        end do
    end do

    close(10)
end program

function f(x, L) result(y)
    implicit real*8 (a-h, o-z)
    x_0 = L / 3.0d0
    s = L / 30.0d0
    y = exp(-(x - x_0)**2.0d0 / (s**2.0d0))
end function f
```

Figura 12: Código para propagação de onda gaussiana com a extremidade $x = L$ solta

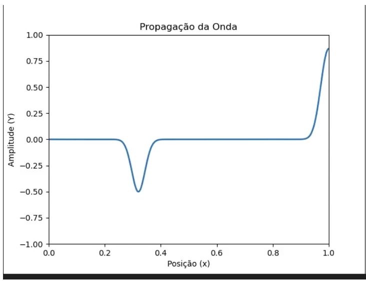
(a) $t = 0$



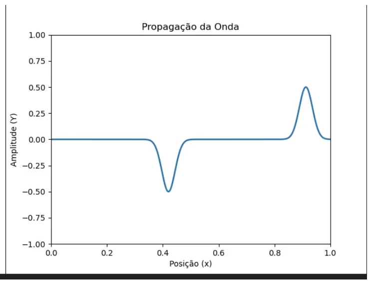
(b) $t = 3$



(c) $t = 6$



(d) $t = 11$



(e) $t = 13$

Figura 13: Propagação de onda gaussiana com a extremidade $x = L$ solta

Podemos observar que uma das extremidades soltar a onda continua sendo refletida na extremidade, porém não é mais espelhada e além disso uma espécie de efeito chicote no fianal da corda, fazendo com que a amplitude da corda aumente.