Caso de estudo: Transferência de calor 1D transiente

A equação de transferência de calor unidimensional transiente, considerando difusividade térmica constante, é dada a seguir.

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}$$

Interpretar qual seria a solução para o caso de regime permanente.

Vamos tentar resolver o problema numericamente.

$$\frac{u_k^{n+1} - u_k^n}{\Delta t} = \alpha \left[\frac{u_{k+1}^n - 2u_k^n + u_{k-1}^n}{(\Delta x)^2} \right]$$

Acima vemos que a parcela no tempo é desenvolvida como *Forward Euler*.

$$u_k^{n+1} - u_k^n = \frac{\alpha \Delta t}{(\Delta x)^2} [u_{k-1}^n - 2u_k^n + u_{k+1}^n]$$

Sendo o número de *Fourier*

$$Fo = \frac{\alpha \Delta t}{(\Delta x)^2}$$

Então

$$u_k^{n+1} = u_k^n + Fo[u_{k-1}^n - 2u_k^n + u_{k+1}^n]$$

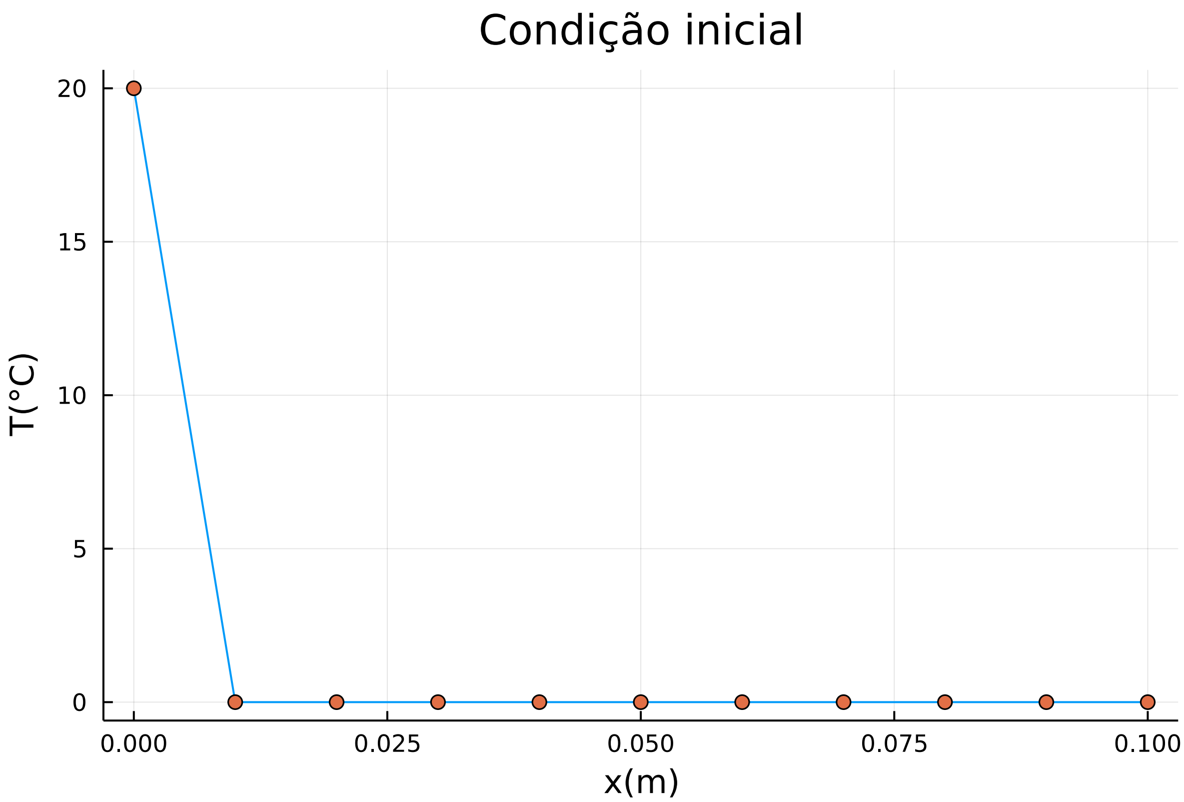
```
begin
    using LinearAlgebra
    using Plots
    using PlutoUI
end
```

```
fo (generic function with 1 method)
```

```
fo(α, Δt, Δx) = α*Δt/(Δx)^2
```

```
0.01
```

```julia
# Condições iniciais
begin
    u₀ = fill(0.0,11)
    u₀[1] = 20.0
    Δx = 0.01
end
```




```julia
begin
    plot(0:Δx:0.1, u₀, label=false)
    xlabel!("x(m)")
    ylabel!("T(°C)")
    title!("Condição inicial")
    scatter!(0:Δx:0.1, u₀, label=false)
end
```

Solução avançando no tempo

Uma forma de implementar é por meio de *loops* (tende a não ser eficiente dependendo da linguagem).

```
simloop (generic function with 1 method)
```

```julia
function simloop(u, Δt, Δx; steps, α=7.5e-7)
    Fo = fo(α, Δt, Δx)
    nn = length(u)
    for _ in 1:steps
        for k in 2:(nn-1)
            i = k-1
            w = k+1
            u[k] = u[k] + Fo*(u[i] - 2*u[k] + u[w])
        end
    end
    return u
end
```

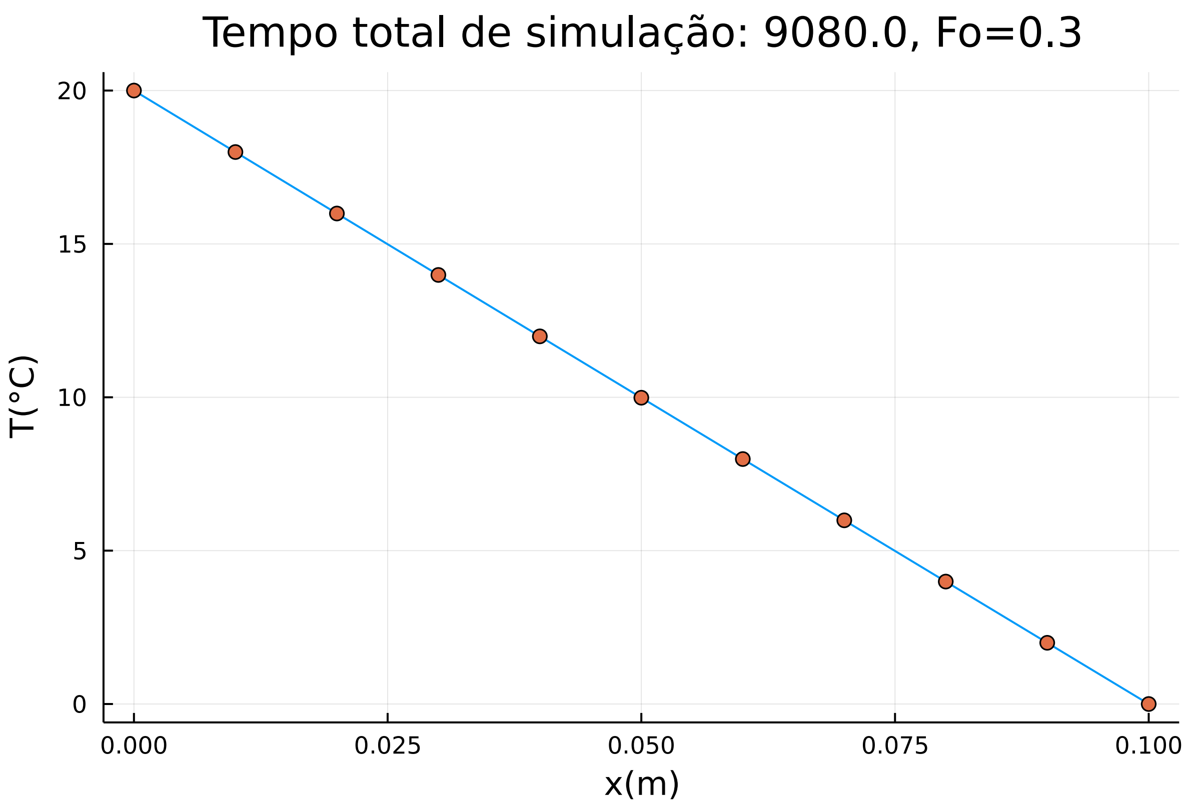
Também é possível escrever na forma matricial tomando cuidado com as condições de contorno, no caso mantidas à temperaturas fixas nas duas extremidades.

$$\begin{bmatrix} u_1^{n+1} \\ u_2^{n+1} \\ u_3^{n+1} \\ \vdots \\ u_{k-1}^{n+1} \end{bmatrix} = \begin{bmatrix} u_1^{n} \\ u_2^{n} \\ u_3^{n} \\ \vdots \\ u_{k-1}^{n} \end{bmatrix} + Fo \begin{bmatrix} -2 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 1 & -2 & -1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & -2 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 & -2 \end{bmatrix} \begin{bmatrix} u_1^{n} \\ u_2^{n} \\ u_3^{n} \\ \vdots \\ u_{k-1}^{n} \end{bmatrix} + Fo \begin{bmatrix} u_0 \\ 0 \\ 0 \\ \vdots \\ u_k \end{bmatrix}$$

```
sim (generic function with 1 method)
```

```julia
function sim(u, Δt, Δx; steps, α=7.5e-7)
    Fo = fo(α, Δt, Δx)
    nn = length(u)
    utemp = u[2:nn-1]
    A = Tridiagonal(fill(1,nn-3), fill(-2,nn-2), fill(1,nn-3))
    ubound = fill(0,nn-2)
    ubound[1] = u[1]
    ubound[end] = u[end]
    for _ in 1:steps
        utemp = utemp + Fo*A*utemp + Fo*ubound
    end
    pushfirst!(utemp, u[1])
    push!(utemp, u[end])
    return utemp
end
```

STEP




```julia
begin
    Δt = 40.0
    uinit = copy(u₀)
    if step==0
        u = u₀
    else
        u= sim(uinit, Δt, Δx, steps=step)
    end
    Fo = fo(7.5e-7, Δt, Δx)
    plot(0:Δx:0.1, u, label=false)
    xlabel!("x(m)")
    ylabel!("T(°C)")
    title!("Tempo total de simulação: "*string(Δt*step)*", Fo="*string(Fo))
    scatter!(0:Δx:0.1, u, label=false)
end
```

Estabilidade da solução numérica

Varie o número de Fourier no gráfico acima e veja o efeito na solução. Para isso é necessário rodar em Julia!