Método das diferenças finitas

O método se baseia na aproximação de derivadas por diferenças finitas (relembrando da 1ª aula). A aproximação pode ser feita a partir da expansão em série de Taylor:

$$f(x+h) = f(x) + hf'(x) + \frac{h^2 f''(x)}{2!} + \frac{h^3 f'''(x)}{3!} + \mathcal{O}(h^4)$$

Forward Euler

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} + \mathcal{O}(h)$$

Backward Euler

$$f'(x) \approx \frac{f(x) - f(x-h)}{h} + \mathcal{O}(h)$$

Diferença centrada

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} + \mathcal{O}(h^2)$$

Expressão para a derivada segunda

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + \mathcal{O}(h^2)$$

Aplicação em problema de contorno

$$-\frac{d^2 f}{dx^2} = 2$$

Caso fixo-fixo $\rightarrow$ condições de contorno: $f(0) = 0$ e $f(1) = 0$

Solução analítica aplicando as condições de contorno?

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

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

```
analiticoFF(x) = -x^2 + x
```

Dividindo a barra em 9 nós ($\Delta x = 0,125$, índices de 0 a 8), sendo o primeiro nó e o nono nó localizados nas extremidades e apresentando, pelas condições de contorno deslocamento nulo, pode-se escrever o problema no formato discreto:

$$\frac{-f_{i+1} + 2f_i - f_{i-1}}{(0,125)^2} = 2$$

ou no formato matricial:

$$\frac{1}{(0,125)^2} \begin{bmatrix} 2 & -1 & 0 & \cdots & 0 & 0 & 0 \\ -1 & 2 & -1 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 2 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_7 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 2 \\ \vdots \\ 2 \end{bmatrix}$$

```
7×7 Matrix{Float64}:
 128.0  -64.0    0.0    0.0    0.0    0.0    0.0
 -64.0  128.0  -64.0    0.0    0.0    0.0    0.0
   0.0  -64.0  128.0  -64.0    0.0    0.0    0.0
   0.0    0.0  -64.0  128.0  -64.0    0.0    0.0
   0.0    0.0    0.0  -64.0  128.0  -64.0    0.0
   0.0    0.0    0.0    0.0  -64.0  128.0  -64.0
   0.0    0.0    0.0    0.0    0.0  -64.0  128.0
```

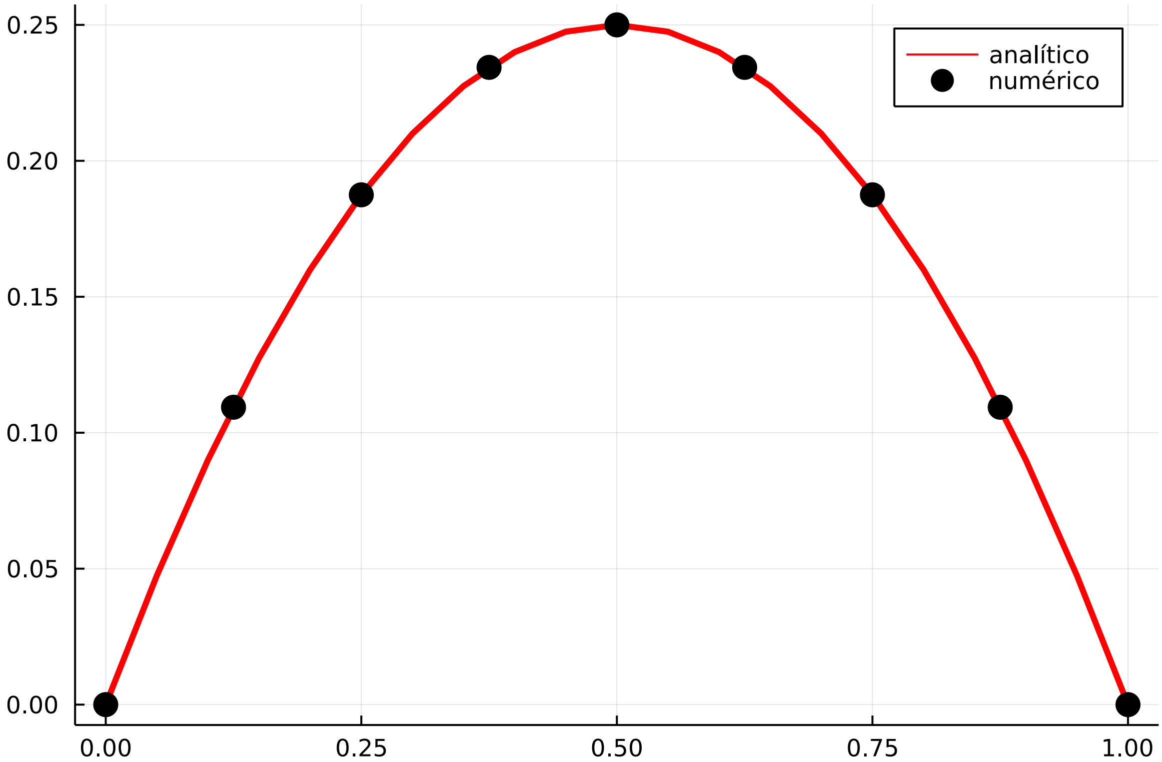
```
begin
    Δx = 0.125
    b = fill(2,7)
    K = (1/Δx^2).*SymTridiagonal(fill(2,7), fill(-1,6))
end
```

```
solFF = [0.109375, 0.1875, 0.234375, 0.25, 0.234375, 0.1875, 0.109375]
```

```
solFF = K \ b
```

```
Float64[
    1:  0.0
    2:  0.109375
    3:  0.1875
    4:  0.234375
    5:  0.25
    6:  0.234375
    7:  0.1875
    8:  0.109375
    9:  0.0
]
```

```
begin
    pushfirst!(solFF, 0)
    push!(solFF, 0)
end
```




```
# Plotando resultados
begin
    plot(0:0.05:1, analiticoFF.(0:0.05:1), linecolor="red", linewidth=3,
    label="analítico")
    scatter!(0:Δx:1, solFF, markercolor="black", markersize=7, label="numérico")
end
```

Caso livre-fixo $\longrightarrow$ condições de contorno: $\frac{df}{dx}(0) = 0$ e $f(1) = 0$

Solução analítica?

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

```
analiticoLF(x) = -x^2 + 1
```

Como a condição de contorno do primeiro nó é dada por sua derivada, a primeira linha da matriz simétrica utilizada no caso fixo-fixo precisa ser alterada.

Aplicando uma aproximação de primeira ordem para a derivada, temos

$$\frac{f_1 - f_0}{\Delta x} = 0$$

logo

$$f_1 = f_0$$

e na primeira linha...

$$\frac{-f_2 + 2f_1 - f_0}{\Delta x^2} = \frac{-f_2 + 2f_1 - f_1}{\Delta x^2} = \frac{-f_2 + f_1}{\Delta x^2} = 2$$

No formato matricial:

$$\frac{1}{(0,125)^2}\begin{bmatrix} 1 & -1 & 0 & \cdots & 0 & 0 & 0 \\ -1 & 2 & -1 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 2 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & -1 & 2 \end{bmatrix}\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_7 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 2 \\ \vdots \\ 2 \end{bmatrix}$$

```
7×7 Matrix{Float64}:
  64.0  -64.0    0.0    0.0    0.0    0.0    0.0
 -64.0  128.0  -64.0    0.0    0.0    0.0    0.0
   0.0  -64.0  128.0  -64.0    0.0    0.0    0.0
   0.0    0.0  -64.0  128.0  -64.0    0.0    0.0
   0.0    0.0    0.0  -64.0  128.0  -64.0    0.0
   0.0    0.0    0.0    0.0  -64.0  128.0  -64.0
   0.0    0.0    0.0    0.0    0.0  -64.0  128.0
```

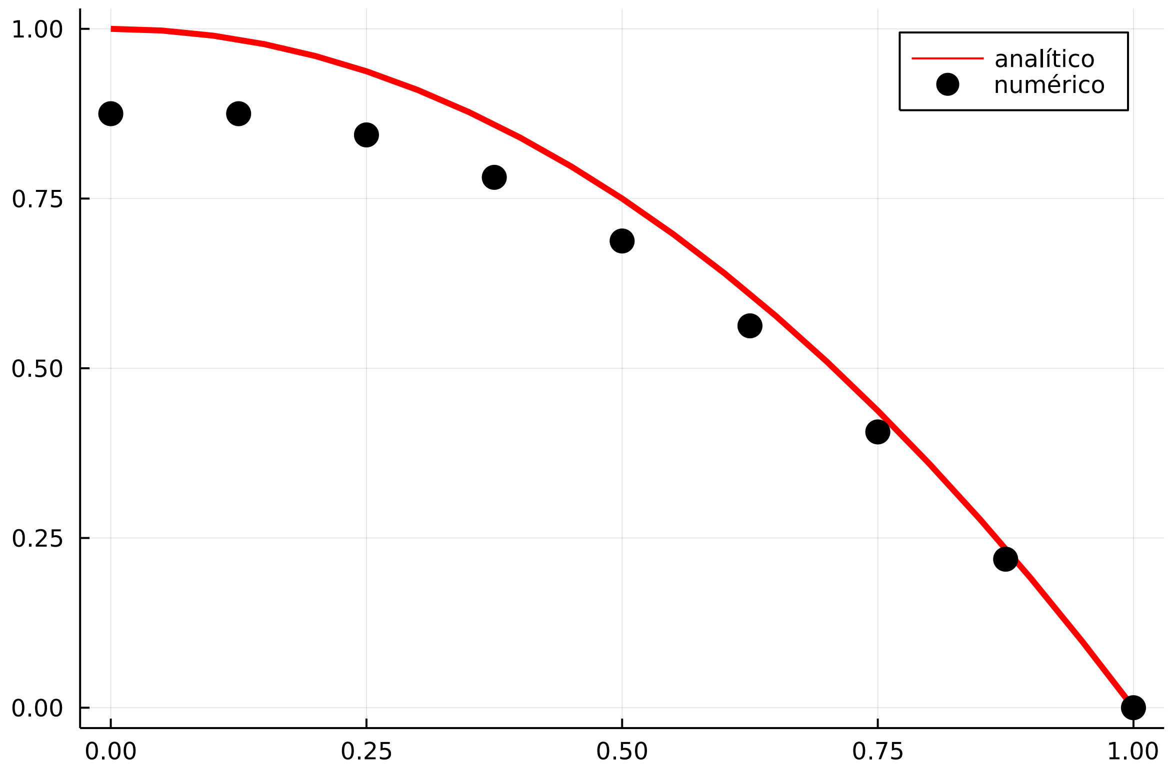
```
begin
    T = SymTridiagonal(fill(2,7), fill(-1,6))
    T[1,1] = 1
    T = (1/Δx^2).*T
end
```

```
solLF = [0.875, 0.84375, 0.78125, 0.6875, 0.5625, 0.40625, 0.21875]
```

```
solLF = T \ b
```

```
[0.875, 0.875, 0.84375, 0.78125, 0.6875, 0.5625, 0.40625, 0.21875, 0.0]
```

```
begin
    pushfirst!(solLF, solLF[1])
    push!(solLF, 0)
end
```




```
# Plotando resultados
begin
    plot(0:0.05:1, analiticoLF.(0:0.05:1), linecolor="red", linewidth=3,
    label="analítico")
    scatter!(0:Δx:1, solLF, markercolor="black", markersize=7, label="numérico")
end
```

Por que encontramos o erro na extremidade livre??

Aproximação de segunda ordem

$$\frac{f_1 - f_{-1}}{2\Delta x} = 0$$

Logo

$$f_1 = f_{-1}$$

A matriz terá, portanto, uma linha adicional.

$$\frac{-f_1 + 2f_0 - f_{-1}}{(\Delta x)^2} = \frac{-2f_1 + 2f_0}{(\Delta x)^2} = 2$$

Dividir por dois para manter a simetria.

$$\frac{-f_1 + f_0}{(\Delta x)^2} = 1$$

No formato matricial:

$$\frac{1}{(0,125)^2}\begin{bmatrix} 1 & -1 & 0 & \cdots & 0 & 0 & 0 \\ -1 & 2 & -1 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 2 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & -1 & 2 \end{bmatrix}\begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_7 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 2 \\ \vdots \\ 2 \end{bmatrix}$$

```
8×8 Matrix{Float64}:
  64.0  -64.0    0.0    0.0    0.0    0.0    0.0    0.0
 -64.0  128.0  -64.0    0.0    0.0    0.0    0.0    0.0
   0.0  -64.0  128.0  -64.0    0.0    0.0    0.0    0.0
   0.0    0.0  -64.0  128.0  -64.0    0.0    0.0    0.0
   0.0    0.0    0.0  -64.0  128.0  -64.0    0.0    0.0
   0.0    0.0    0.0    0.0  -64.0  128.0  -64.0    0.0
   0.0    0.0    0.0    0.0    0.0  -64.0  128.0  -64.0
   0.0    0.0    0.0    0.0    0.0    0.0  -64.0  128.0
```

```
begin
    Tb = SymTridiagonal(fill(2,8), fill(-1,7))
    Tb[1,1] = 1
    Tb = (1/Δx^2).*Tb
end
```

```
1
```

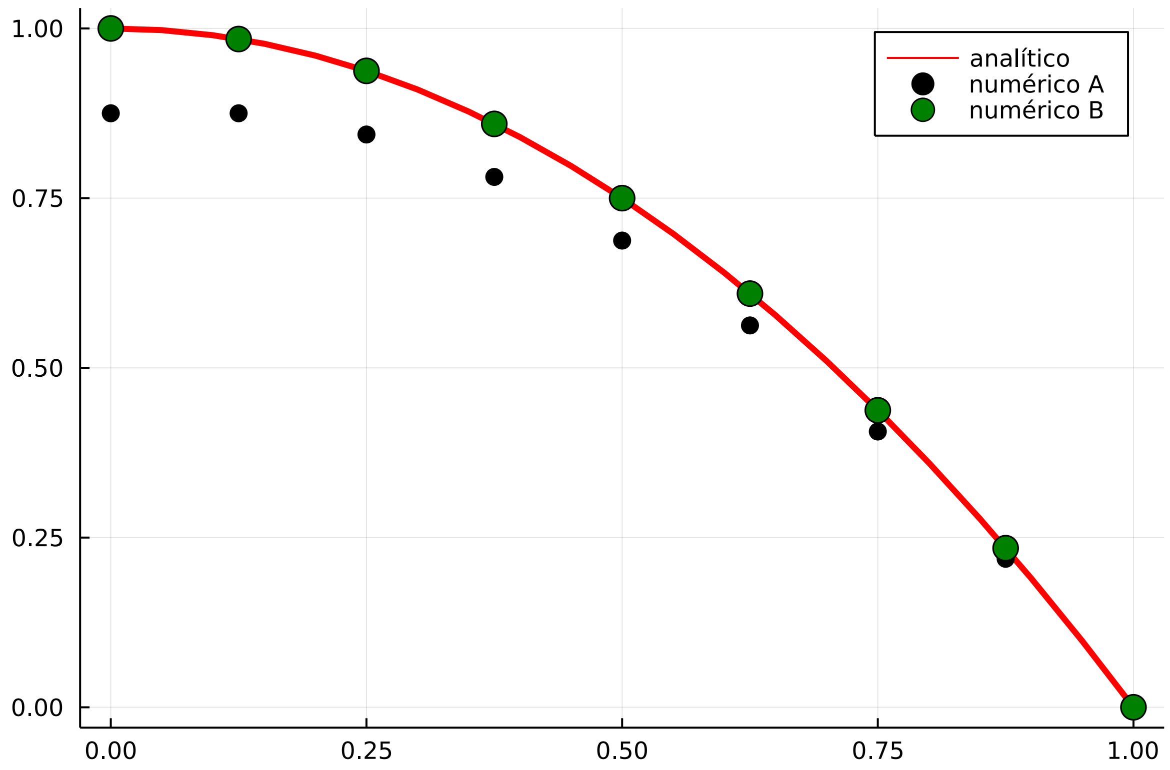
```
begin
    bb = fill(2,8)
    bb[1] = 1
end
```

```
solLFb = [1.0, 0.984375, 0.9375, 0.859375, 0.75, 0.609375, 0.4375, 0.234375]
```

```
solLFb = Tb \ bb
```

[1.0, 0.984375, 0.9375, 0.859375, 0.75, 0.609375, 0.4375, 0.234375, 0.0]

```
push!(solLFb, 0)
```




```
# Plotando resultados
begin
    plot(0:0.05:1, analiticoLF.(0:0.05:1), linecolor="red", linewidth=3,
    label="analítico")
    scatter!(0:Δx:1, solLF, markercolor="black", markersize=5, label="numérico A")
    scatter!(0:Δx:1, solLFb, markercolor="green", markersize=7, label="numérico B")
end
```

Aplicação em problema de valor inicial

Problema de oscilação de um sistema massa-mola de 1 GDL

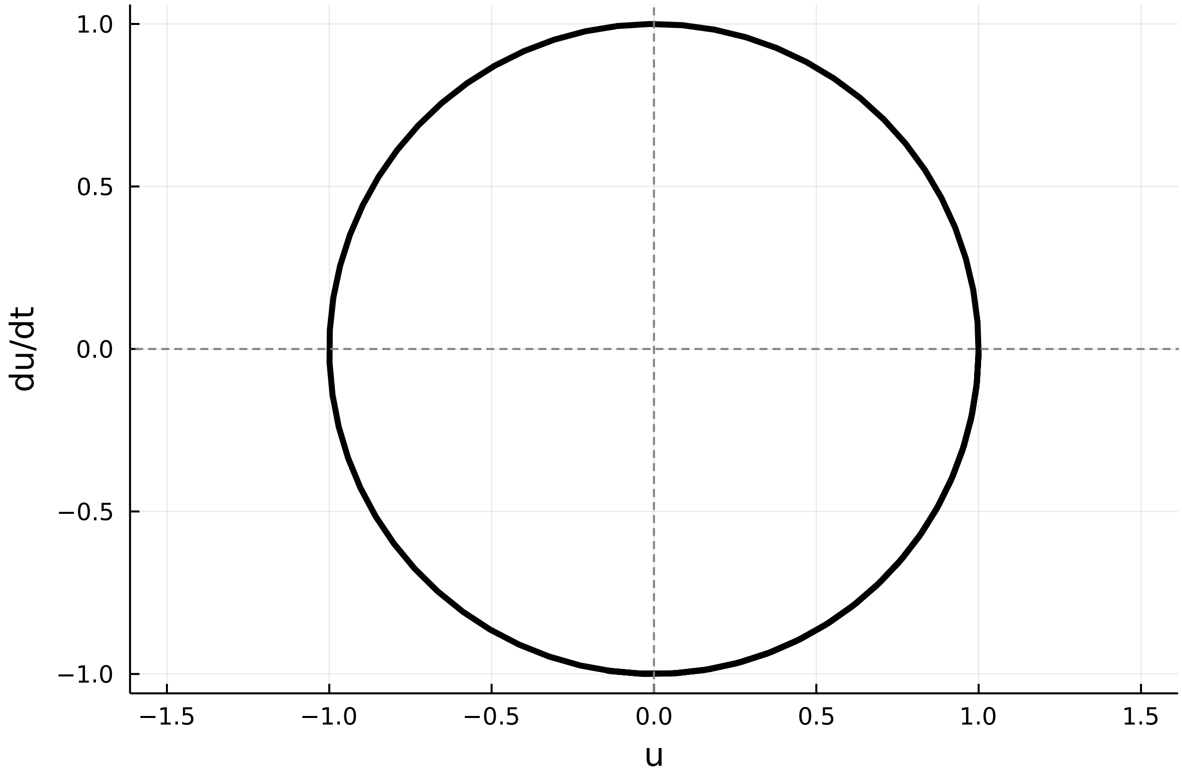
$$m\frac{d^2u}{dt^2} + ku = 0$$

Condições de contorno $\rightarrow u(0) = 1$ e $u'(0) = 0$

Solução exata

Considerando $m = k = 1$, qual a solução exata para o problema?

Diagrama de fase da solução analítica




```
begin
    plot(cos.(0:0.1:8), -sin.(0:0.1:8), linecolor="black", linewidth=3,
    label=false, ratio=1)
    vline!([0], linecolor="gray", linestyle=:dash, label=false)
    hline!([0], linecolor="gray", linestyle=:dash, label=false)
    xlabel!("u")
    ylabel!("du/dt")
end
```

Solução numérica utilizando o método das diferenças finitas

Sempre é possível escrever uma ODE de segunda ordem em um sistema de equações de primeira ordem.

$$\begin{cases} u' = w \\ w' = -u \end{cases}$$

Utilizando **Forward Euler**:

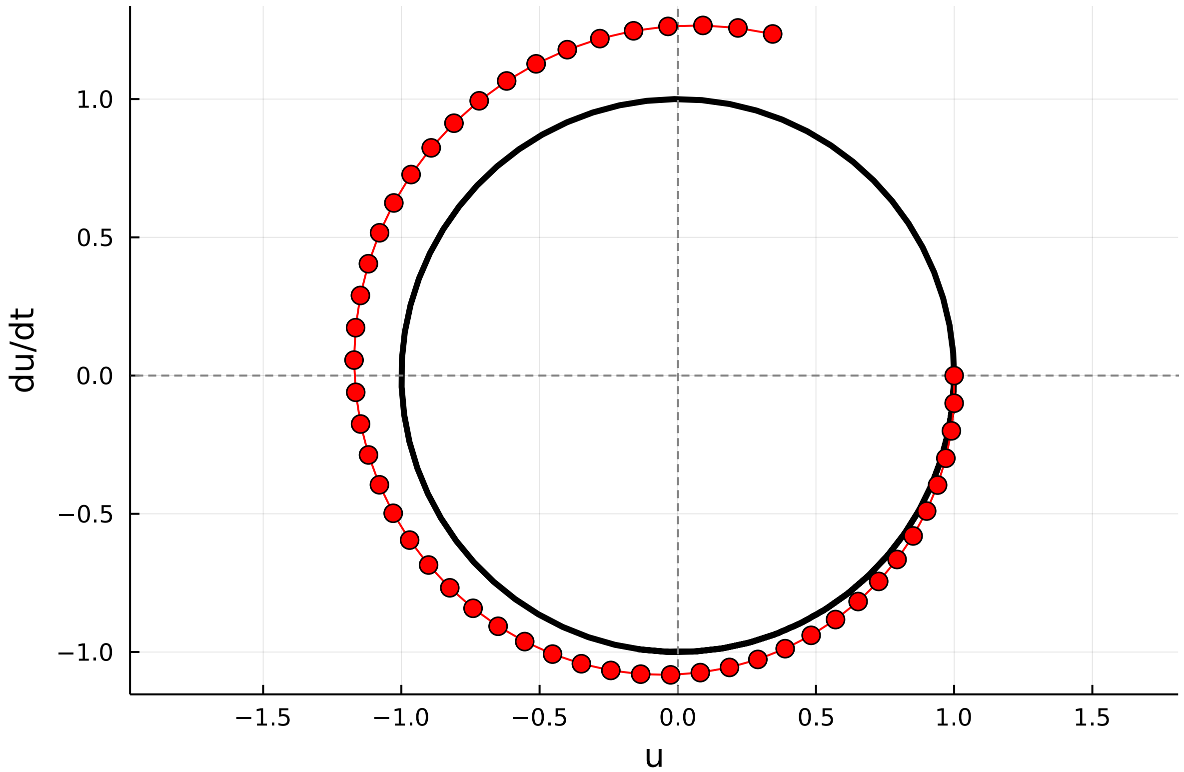
$$\begin{cases} u_{n+1} = u_n + \Delta t \; w_n \\ w_{n+1} = w_n - \Delta t \; u_n \end{cases}$$

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

```julia
function forwardEuler(dt,u₀,w₀,nsteps)
    u = [u₀]
    w = [w₀]
    steps = collect(1:nsteps)
    for i in steps
        uk = u[i] + dt*w[i]
        wk = w[i] - dt*u[i]
        push!(u, uk)
        push!(w, wk)
    end
    return [u, w]
end
```

```
df =
 [[1.0, 1.0, 0.99, 0.97, 0.9401, 0.9005, 0.851499, 0.793493, 0.726972,    more ,0.343355],
```

```julia
df = forwardEuler(0.1, 1.0, 0.0, 50)
```




```julia
# Plotando dados
begin
    plot(cos.(0:0.1:8), -sin.(0:0.1:8), linecolor="black", linewidth=3,
    label=false, ratio=1)
    vline!([0], linecolor="gray", linestyle=:dash, label=false)
    hline!([0], linecolor="gray", linestyle=:dash, label=false)
    xlabel!("u")
    ylabel!("du/dt")
    plot!(df[1], df[2], linecolor="red", label=false)
    scatter!(df[1], df[2], markercolor="red", markersize=5, label=false)
end
```

Utilizando **Backward Euler**:

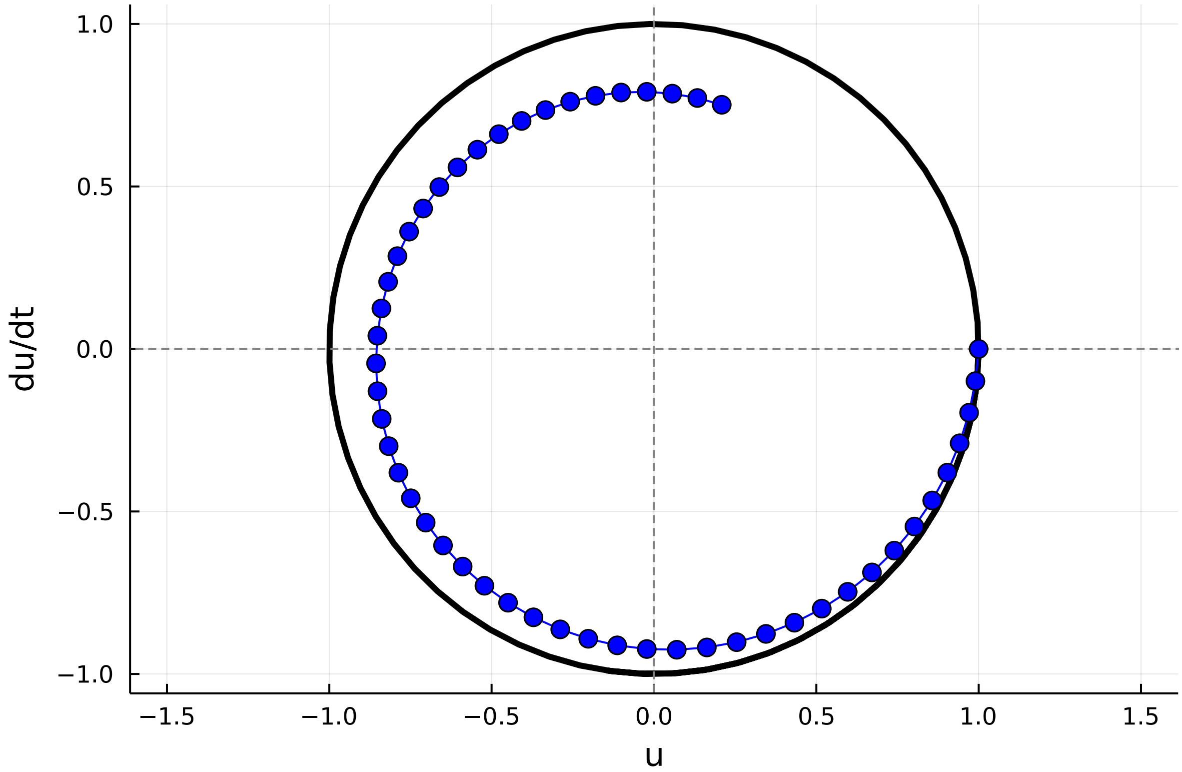
$$\begin{cases} u_{n+1} = u_n + \Delta t \; w_{n+1} \\ w_{n+1} = w_n - \Delta t \; u_{n+1} \end{cases}$$

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

```
function backwardEuler(dt,u₀,w₀,nsteps)
    u = [u₀]
    w = [w₀]
    steps = collect(1:nsteps)
    for i in steps
        uk = (u[i] + dt*w[i])/(1 + dt^2)
        wk = w[i] - dt*uk
        push!(u, uk)
        push!(w, wk)
    end
    return [u,w]
end
```

```
db =
[[1.0, 0.990099, 0.970493, 0.941472, 0.903418, 0.856795, 0.802151, 0.740105, 0.671346,
```

```
db = backwardEuler(0.1, 1.0, 0.0, 50)
```




```
# Plotando dados
begin
    plot(cos.(0:0.1:8), -sin.(0:0.1:8), linecolor="black", linewidth=3,
    label=false, ratio=1)
    vline!([0], linecolor="gray", linestyle=:dash, label=false)
    hline!([0], linecolor="gray", linestyle=:dash, label=false)
    xlabel!("u")
    ylabel!("du/dt")
    plot!(db[1], db[2], linecolor="blue", label=false)
    scatter!(db[1], db[2], markercolor="blue", markersize=5, label=false)
end
```

Utilizando **Diferenças centradas**:

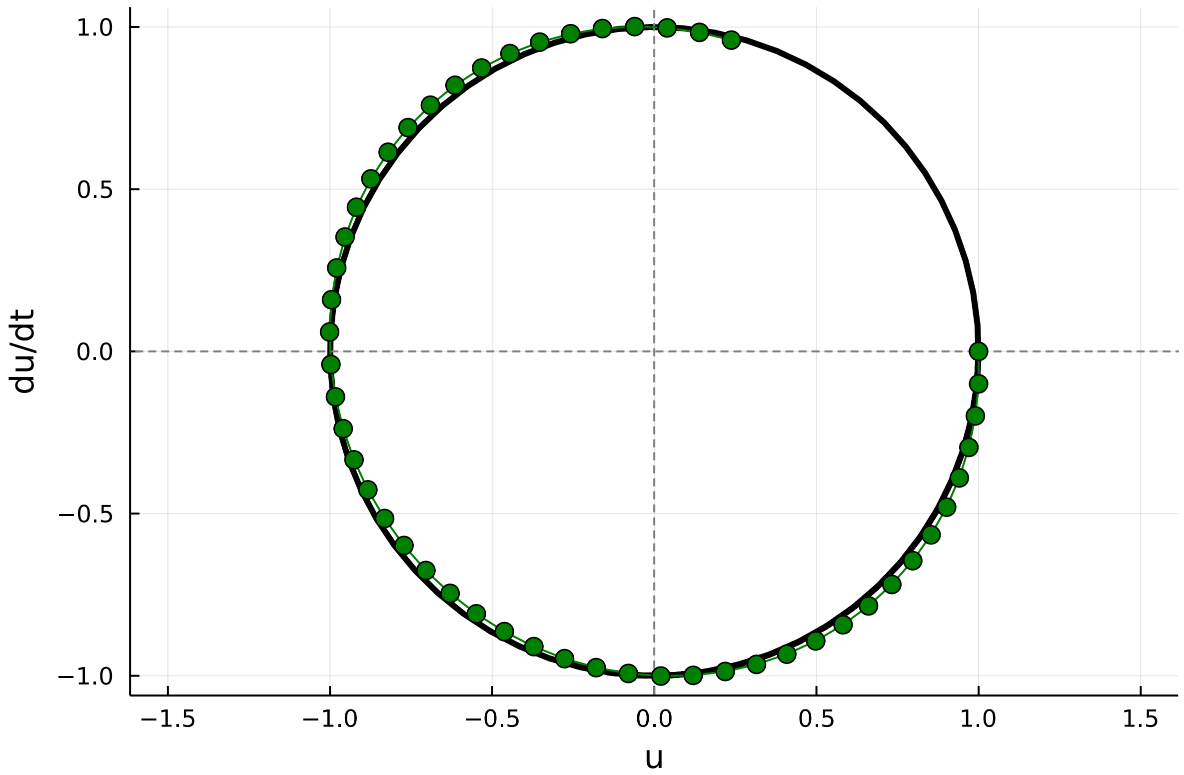
$$\begin{cases} u_{n+1} = u_n + \Delta t \ w_n \\ w_{n+1} = w_n - \Delta t \ u_{n+1} \end{cases}$$

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

```
function centeredDiff(dt,u₀,w₀,nsteps)
    u = [u₀]
    w = [w₀]
    steps = collect(1:nsteps)
    for i in steps
        uk = u[i] + dt*w[i]
        wk = w[i] - dt*uk
        push!(u, uk)
        push!(w, wk)
    end
    return [u, w]
end
```

```
dc =
 [[1.0, 1.0, 0.99, 0.9701, 0.940499, 0.901493, 0.853472, 0.796916, 0.732392,    more ,0.23
```

```
dc = centeredDiff(0.1, 1.0, 0.0, 50)
```




```
# Plotando dados
begin
    plot(cos.(0:0.1:8), -sin.(0:0.1:8), linecolor="black", linewidth=3,
    label=false, ratio=1)
    vline!([0], linecolor="gray", linestyle=:dash, label=false)
    hline!([0], linecolor="gray", linestyle=:dash, label=false)
    xlabel!("u")
    ylabel!("du/dt")
    plot!(dc[1], dc[2], linecolor="green", label=false)
    scatter!(dc[1], dc[2], markercolor="green", markersize=5, label=false)
end
```