HW #5 Computational

Kevin Corcoran

May 20, 2021

1 Including required packages

```
using Plots
using LaTeXStrings
using SparseArrays
theme(:mute)

using Pkg
Pkg.activate("DiffyQ")
include("code/DiffyQ.jl")
using .DiffyQ: ForwardEuler_sys, Trapezoid_sys, BTCS, s2_DIRK_sys, ForwardEuler_tsys
```

1.1 Setting up discretization for problems 3 and 4

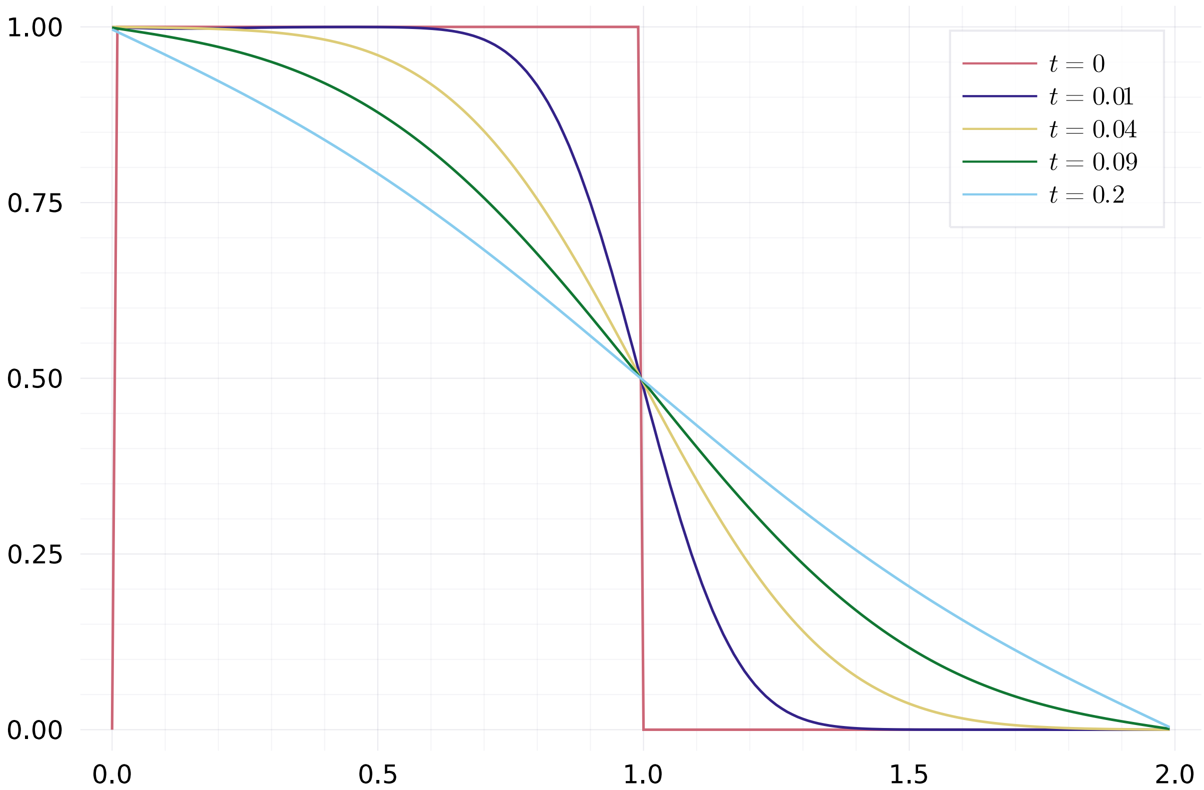
```
# Space
x0 = 0.0; x = 2.0;
Δx = 0.01; L = x-x0; N = Int(L/Δx);

A = 1/Δx^2 * spdiagm(-1=>ones(N-1),0=>-2.0*ones(N),1=>ones(N-1))

# Time
t0 = 0.0; t = 0.2;
T = t-t0;
```

2 Problem 3:

2.1 Stable threshold

```
# Δt = (Δx)^2/2 * 1/0.99; M = Int(T/Δt); # unstable
Δt = (Δx)^2/2 * 1/1.01; M = Int(T/Δt); # stable

# initial condition
f(x) = convert(Array{Float64}, (x .< 1) .& (x .> 0))

# boundary condition
b = zeros(N); b[1] = 1.0; b[end] = 0.0; b = 1/(Δx^2) * b;

xs = collect(0:N-1)*Δx
```

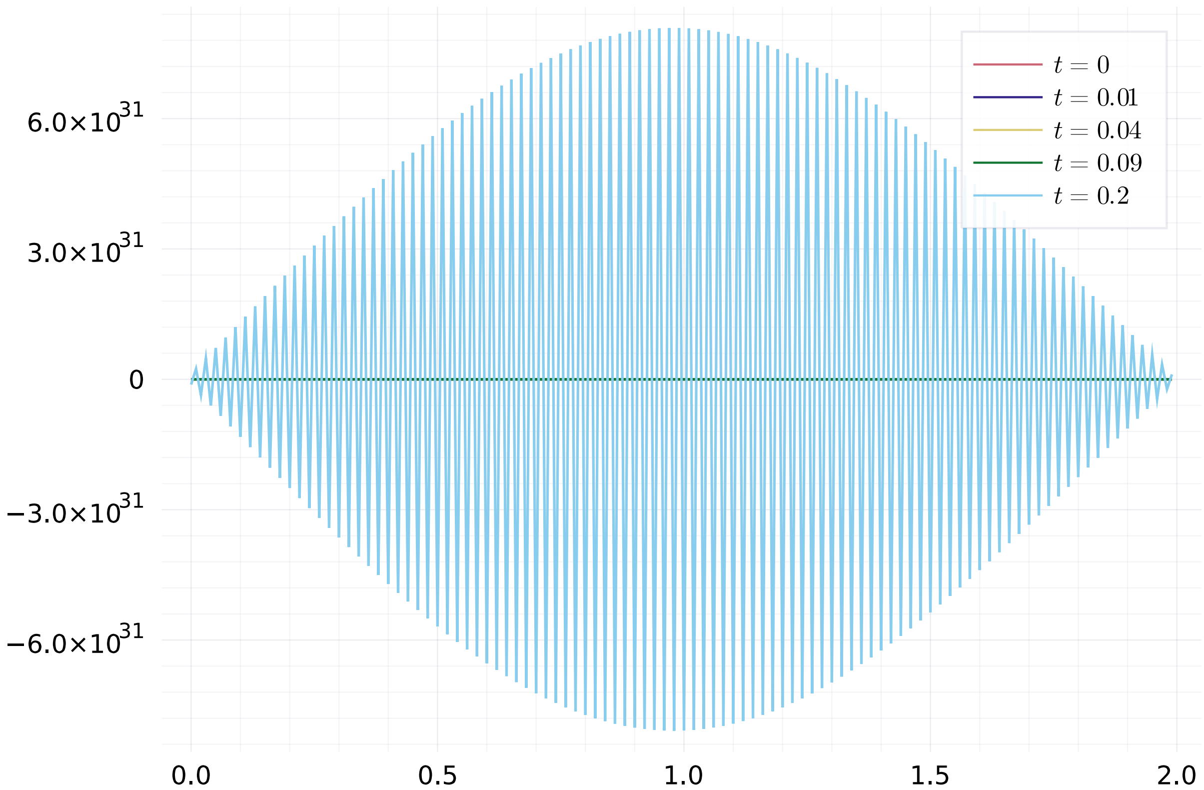
```
u0 = f(xs)

u = ForwardEuler_sys(M, T, u0, A, b)

xs = collect(0:N)*Δx

ts = (1:3) .^2 * 1e-2
ms = Int.(floor.((ts .- t0)/Δt))# indices for ts
plot(xs[1:N], u0[1:N], label = latexstring("t = ", 0 ))
j = 1
for i ∈ ms
    plot!(xs[1:N], u[1:N, i], label = latexstring("t = ",ts[j]))
    global j = j + 1
end

plot!(xs[1:N], u[1:N, end], label = latexstring("t = ", 0.2))
```



2.2 Unstable threshold

```
Δt = (Δx)^2/2 * 1/0.99; M = Int(T/Δt); # unstable

# initial condition
f(x) = convert(Array{Float64}, (x .< 1) .& (x .> 0))

# boundary condition
b = zeros(N); b[1] = 1.0; b[end] = 0.0; b = 1/(Δx^2) * b;

xs = collect(0:N-1)*Δx
u0 = f(xs)

u = ForwardEuler_sys(M, T, u0, A, b)
```

```
xs = collect(0:N)*Δx

ts = (1:3) .^2 * 1e-2
ms = Int.(floor.((ts .- t0)/Δt))# indices for ts
plot(xs[1:N], u0[1:N], label = latexstring("t = ", 0 ))
j = 1
for i ∈ ms
    plot!(xs[1:N], u[1:N, i], label = latexstring("t = ",ts[j]))
    global j = j + 1
end

plot!(xs[1:N], u[1:N, end], label = latexstring("t = ", 0.2))
```

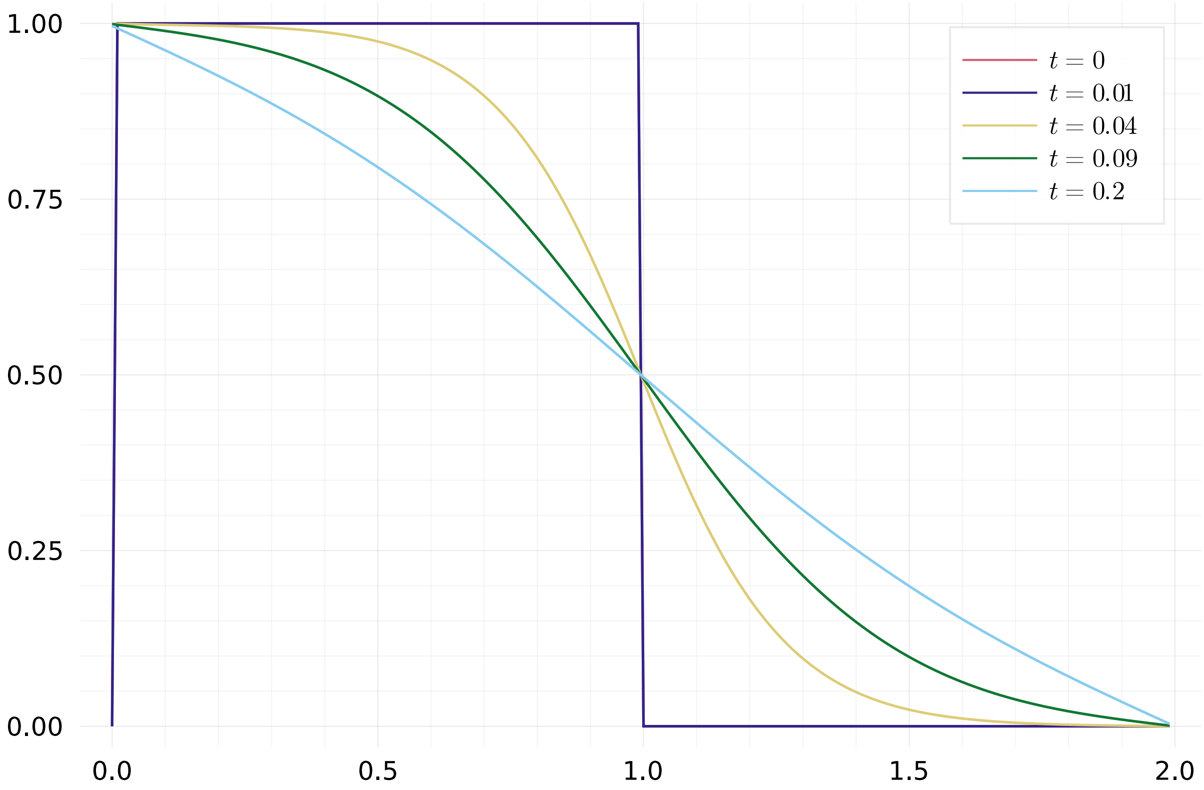


3 Problem 4:

3.1 BTCS (Backward Euler)

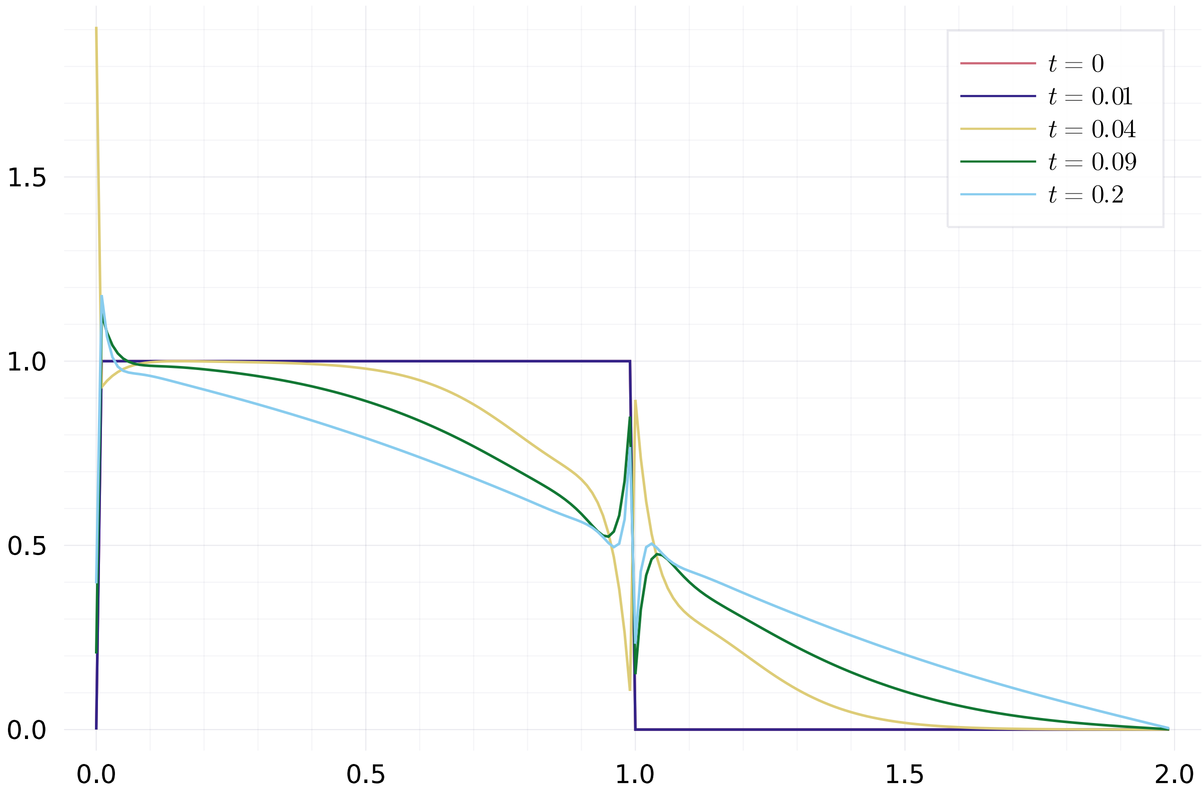
```
Δt = 0.01; M = Int(T/Δt);
# Backward Euler
u = BTCS(M, T, u0, A, b)

# Plotting routine
ts = (1:3) .^2 * 1e-2
ms = Int.(floor.((ts .- t0)/Δt))# indices for ts
plot(xs[1:N], u0[1:N], label = latexstring("t = ", 0 ))
j = 1
for i ∈ ms
    plot!(xs[1:N], u[1:N, i], label = latexstring("t = ",ts[j]))
```

```
    global j = j + 1
end

plot!(xs[1:N], u[1:N, end], label = latexstring("t = ", 0.2))
```



3.2 Crank-Nicolson (CTCS: Trapezoid)

```
u = Trapezoid_sys(M, T, u0, A, b)

# Plotting routine
ts = (1:3) .^2 * 1e-2
ms = Int.(floor.((ts .- t0)/Δt))# indices for ts
plot(xs[1:N], u0[1:N], label = latexstring("t = ", 0 ))
j = 1
for i ∈ ms
    plot!(xs[1:N], u[1:N, i], label = latexstring("t = ",ts[j]))
    global j = j + 1
end

plot!(xs[1:N], u[1:N, end], label = latexstring("t = ", 0.2))
```

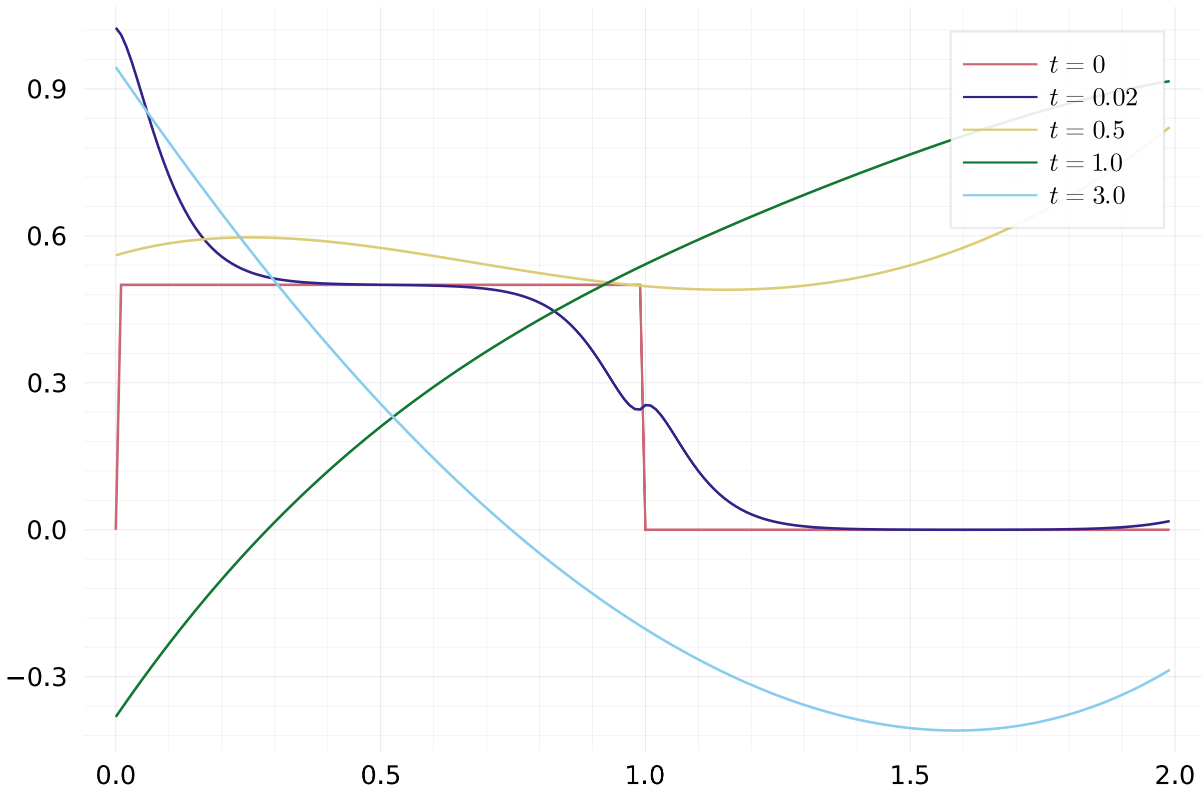


4 Problem 5:

4.1 Part 1

```
# Time
t0 = 0.0; t = 3.0;
T = t-t0;
Δt = 0.01; M = Int(T/Δt);

# boundary condition
function b_(t)
    b = zeros(N);
    b[1] = cos(2*t);
    b[end] = sin(2*t);
    return 1/(Δx^2) * b;
end

xs = collect(0:N-1)*Δx
u0 = 0.5*f(xs)

# 2s DIRK
α = 1 - 1/sqrt(2)
u = s2_DIRK_sys(M, T, u0, α, A, b_)

# Plotting routine
ts = [0.02,0.5,1]
ms = Int.(floor.((ts .- t0)/Δt))# indices for ts
plot(xs[1:N], u0[1:N], label = latexstring("t = ", 0 ))
j = 1
```

```
for i ∈ ms
    plot!(xs[1:N], u[1:N, i], label = latexstring("t = ",ts[j]))
    global j = j + 1
end

plot!(xs[1:N], u[1:N, end], label = latexstring("t = ", 3.0))
```



5 Problem 6:

```
# Space
x0 = 0.0; x = 2.0;
N = 200; L = x-x0; Δx = L/N;

# Time
t0 = 0.0; t = 3.0;
T = t-t0;
Δt = 4e-5; M = Int(T/Δt);

# initial condition
p(x) = (1 .-0.5*x).^2

# boundary condition
α = 0.4
q(t) = 2*sin(t)^2
A = spdiagm(-1=>ones(N-1),0=>-2.0*ones(N),1=>ones(N-1))
A[1,1] = A[1,1] + (2-α*Δx)/(2+α*Δx)
A = 1/(Δx^2)*A
b = zeros(N); b[end] = q(N*Δt); b = 1/(Δx^2) * b;

function b_(t)
```

```
    b = zeros(N); b[end] = 2*sin(t)^2;
    return 1/(Δx^2) * b;
end

xs = collect(0:N-1)*Δx
u0 = p(xs)

# FTCS
u = ForwardEuler_tsys(M,T,u0,A,b_)

# Plotting routine
ts = [0.02,0.5,1]
ms = Int.(floor.((ts .- t0)/Δt))# indices for ts
plot(xs[1:N], u0[1:N], label = latexstring("t = ", 0 ))
j = 1
for i ∈ ms
    plot!(xs[1:N], u[1:N, i], label = latexstring("t = ",ts[j]))
    global j = j + 1
end

plot!(xs[1:N], u[1:N, end], label = latexstring("t = ", 3.0))
```