HM4 – REPORT

Nonlinear hyperbolic problems

Finite Element in Fluids

Álvaro Rodríguez Luis

1. Problem statement

The problem to be solved is the 1D Burguers' equation with the vanishing viscosity approach:

$$\begin{cases} u_t + u u_x = \epsilon u_{xx} \\ u(x,0) = u_0(x) \end{cases}$$

where ϵ is the numerical diffusion parameter, the equation is solved in the spatial interval $[0,4]$, the time interval $[0,4]$ and the with the following initial condition.

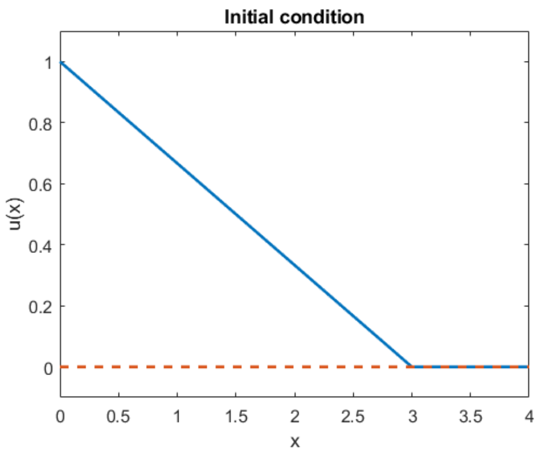


Figure 1. Initial condition.

The provided code is ready to solve the problem using FEM discretization that yields the following system:

$$\mathbf{M}\dot{\mathbf{U}} + \mathbf{C}(\mathbf{U})\mathbf{U} + \epsilon\mathbf{K}\mathbf{U} = 0$$

where the computation of the mass matrix $\mathbf{M}$, the convection matrix $\mathbf{C}(\mathbf{U})$, and the diffusion matrix $\mathbf{K}$ is already implemented. For the time integration scheme, the code includes an explicit forward Euler method and an implicit backward Euler method that uses Picard iteration to find the solution of the nonlinear equation.

In this report the work done to implement the Newton-Raphson iterative method to solve the nonlinear system is presented.

2. Code modifications

At each time step, the equation $\mathbf{f}(\mathbf{U}) = \mathbf{0}$ is solved, where:

$$\mathbf{f}(\mathbf{U}) = (\mathbf{M} + \Delta t \cdot \mathbf{C}(\mathbf{U}) + \epsilon \Delta t \cdot \mathbf{K}) \cdot \mathbf{U} - \mathbf{M} \cdot \mathbf{U}^n$$

In order to implement the well-known Newton-Raphson method, the Jacobian matrix $\mathbf{J} = \frac{d\mathbf{f}}{d\mathbf{U}}$ needs to be computed first. We have:

$$\mathbf{J} = \mathbf{M} + \Delta t \cdot \mathbf{C}(\mathbf{U}) + \epsilon \Delta t \cdot \mathbf{K}$$

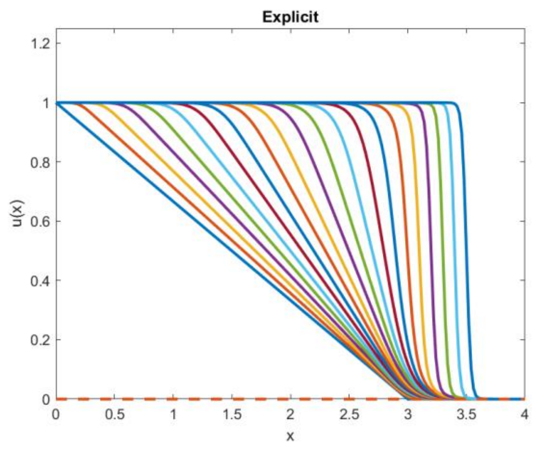
This was used to implement the `burgers_imNR.m` MATLAB function, which was missing in the provided code, and is presented in the Annex.

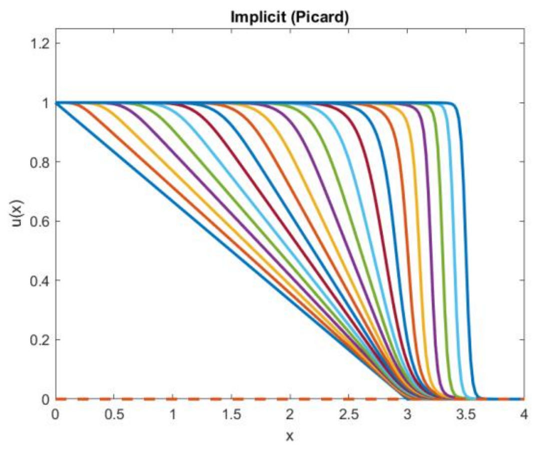
3. Results

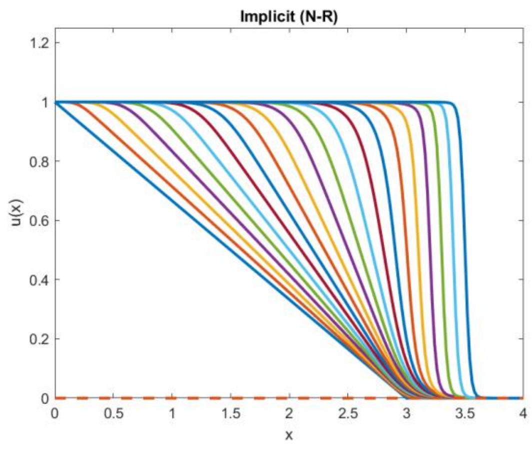
The implementation of the Newton-Raphson method was compared with the other available methods, obtaining satisfactory results. The test cases were the following:

Case 1

Mesh Size $\mathrm{h} = 0.02$, Time Step $\Delta\mathrm{t} = 0.005$, Numerical Diffusion $\epsilon = 0.01$

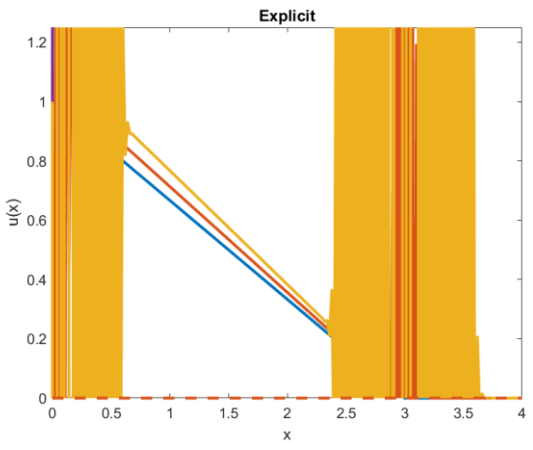


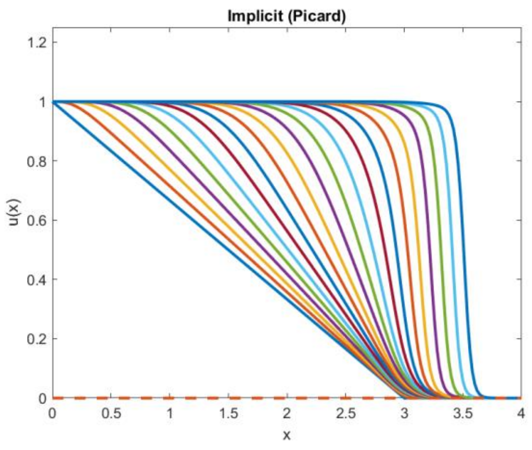


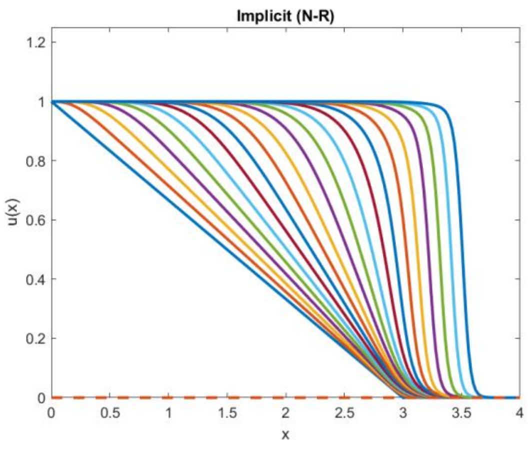


Case 2

Mesh Size $h = 0.02$, Time Step $\Delta t = 0.005$, Numerical Diffusion $\epsilon = 0.01$

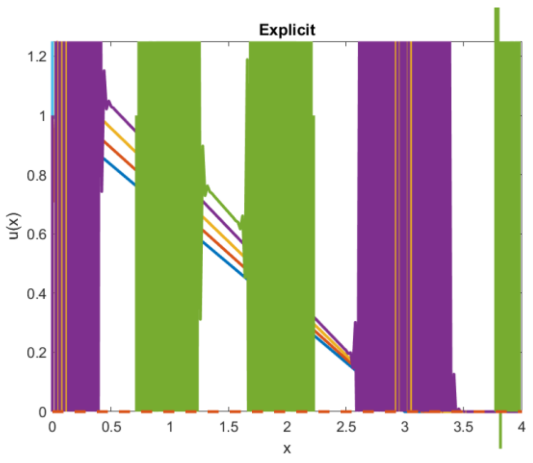


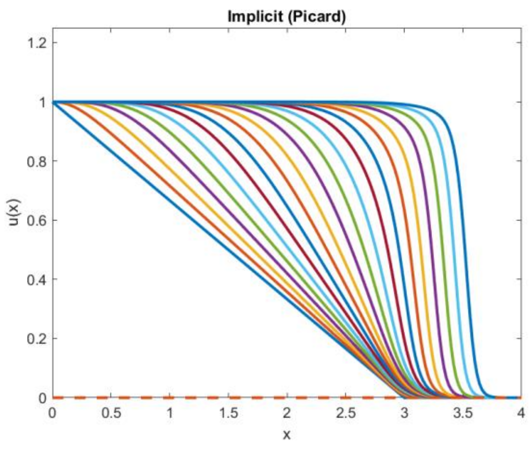


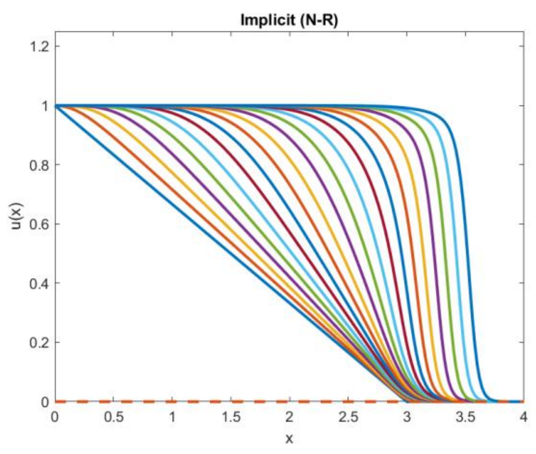


Case 3

Mesh Size $h = 0.02$, Time Step $\Delta t = 0.005$, Numerical Diffusion $\epsilon = 0.01$

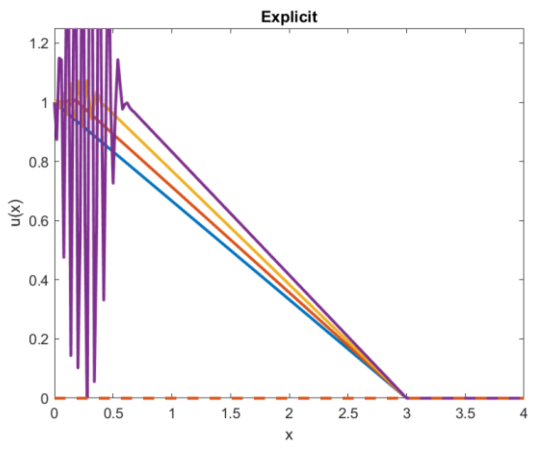


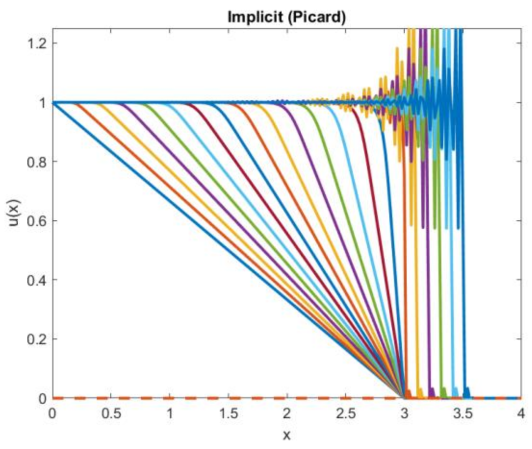


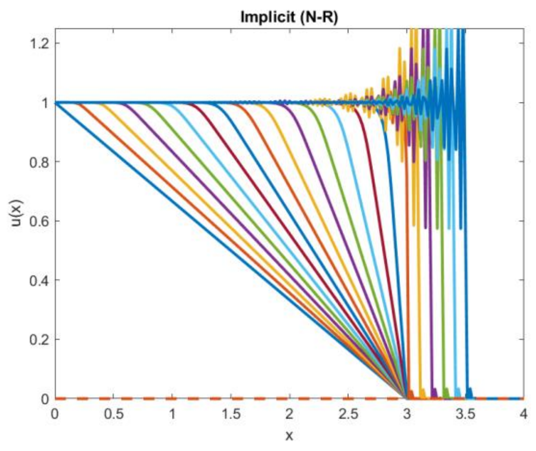


Case 4

Mesh Size $h = 0.02$, Time Step $\Delta t = 0.005$, Numerical Diffusion $\epsilon = 0.01$







Annex – Code modifications

burgers_imNR.m

```
function U = burgers_imNR(X,T,At,nTimeSteps,u0,uxa,uxb,E)
%Number of elements
m = length(X)-1;
% Allocate memory for the solution matrix
U = zeros(m+1,nTimeSteps+1);
% Initial condition
U(:,1) = u0;
% Boundary conditions
U(1,:) = uxa';
U(m+1,:) = uxb';
% Obtain the system constant matrices
M = computeMassMatrix(X,T);
K = computeDiffussionMatrix(X,T);
% Convergence criteria
nitermax = 20;
tol = 5e-6;
% Loop over all the time steps
for n = 1:nTimeSteps
    % Previous time step solution
    U0 = U(:,n);
    % Reset convergence parameters
    error_U = 1; k = 0;
    % Iterative NR method
    while (error_U > tol) && (k < nitermax)
        C = computeConvectionMatrix(X,T,U0);
        A = M + At*C + At*E*K;
        f = M*U(:,n);
        res = A*U0 - f;
        J = M + At*C + At*E*K;
        dU = -J\res;
        U1 = U0 + dU;
        % Boundary conditions
        U1(1,:) = uxa'; U1(m+1,:) = uxb';
        % Update iteration temporal data
        error_U = norm(U1-U0)/norm(U1);
        U0 = U1;k = k+1;
    end
    U(:,n+1) = U1;
    if k >= nitermax
        error('N-R iterative method could not find convergence')
    end
end
```