Màster en Mètodes Numèrics **Aitor Bazán Escoda**

Universitat Politècnica de Catalunya

Finite Elements in Fluids

Homework 4: Nonlinear hyperbolic problems

Problem statement

Case 1

```
u0 =  [zeros(size(X(X<1))); 0.5*ones(size(X(X>=1)))];
```

- For the conditions of Δt = 0.005 and E = 1e-3

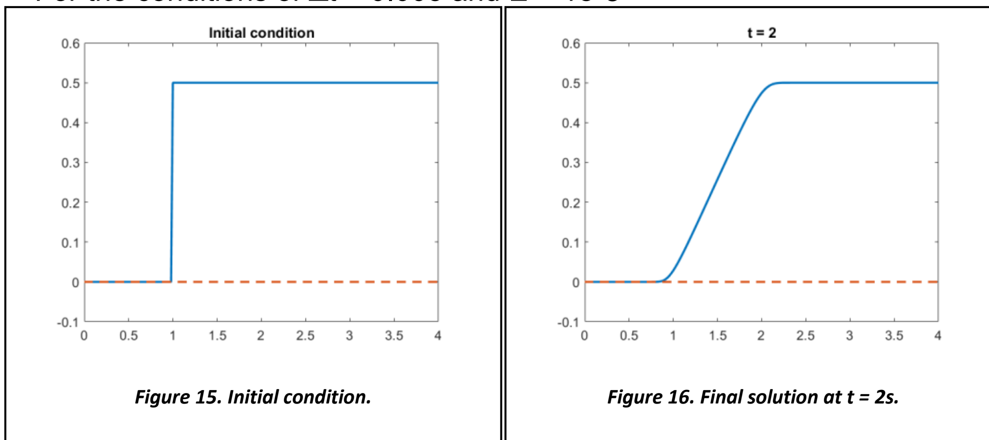

Figure 15. Initial condition.

Figure 16. Final solution at t = 2s.

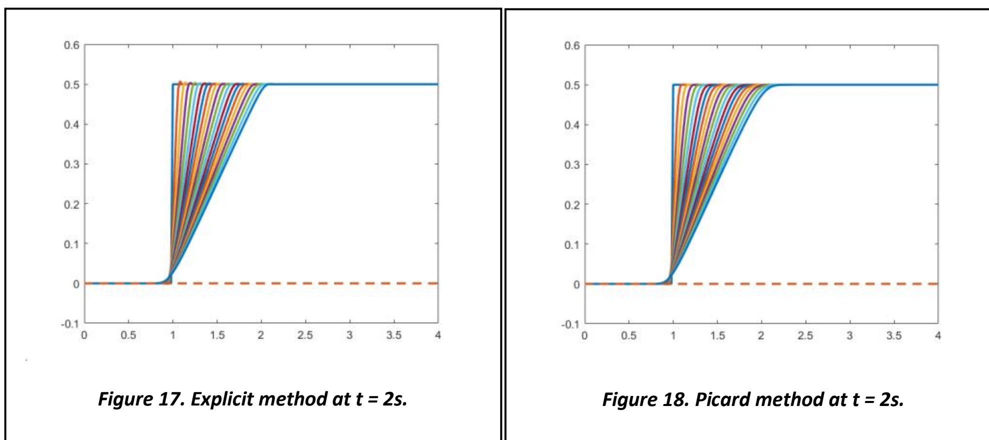

Figure 17. Explicit method at t = 2s.

Figure 18. Picard method at t = 2s.

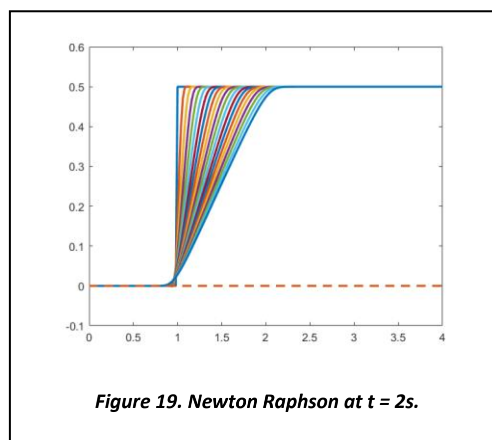

Figure 19. Newton Raphson at t = 2s.

- For the conditions of Δt = 0.005 and E = 0

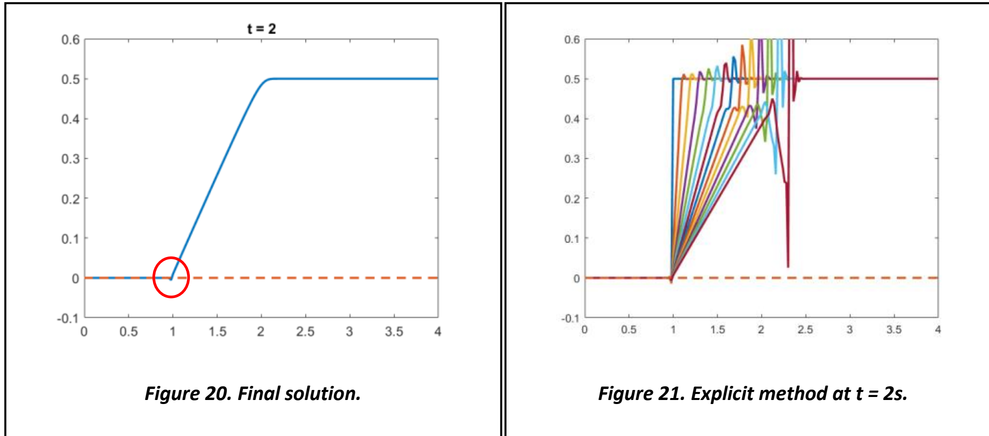

Figure 20. Final solution.

Figure 21. Explicit method at t = 2s.

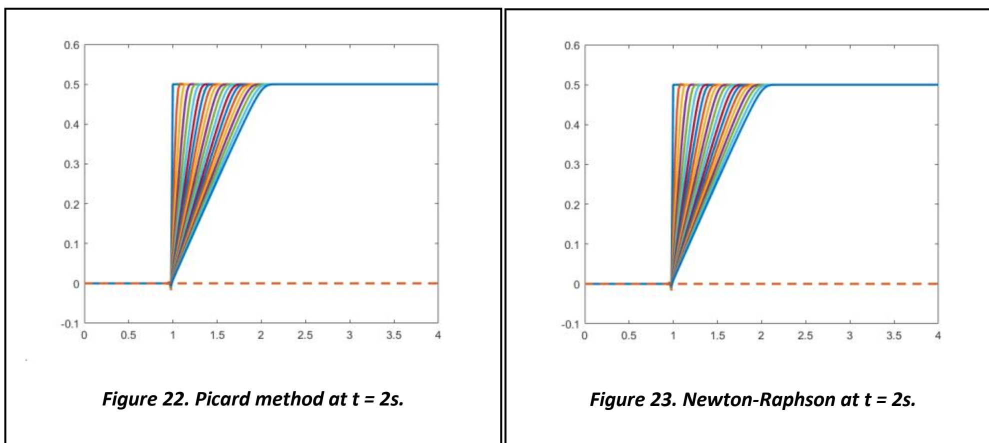

Figure 22. Picard method at t = 2s.

Figure 23. Newton-Raphson at t = 2s.

Comments on the results

The main remark here is to point out the effect of having low timesteps and diffusivity coefficients closer to 0. As it is seen in the case of Δt = 0.005 and E = 0, it appears a lack of accuracy in the solution in implicit methods whereas for explicit methods the diffusivity seems not to be playing a role.
With increasing initial conditions, the solution gets the shape an entropy-compliant solution for Burger's equation (particularly rarefaction wave).

<u>Case 2</u>

```
u0 = [1-X(X<3)/3; X(X>=3)*0];
```

- For the conditions of Δt = 0.005 and E = 1e-2

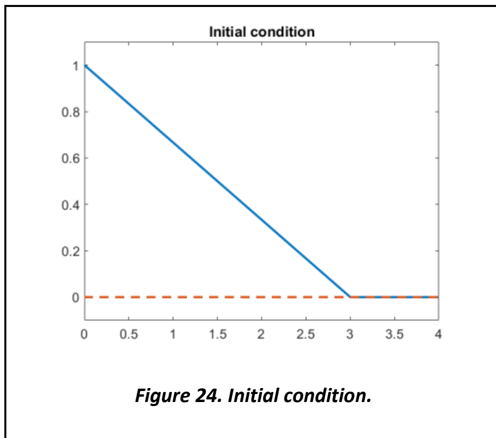

Figure 24. Initial condition.

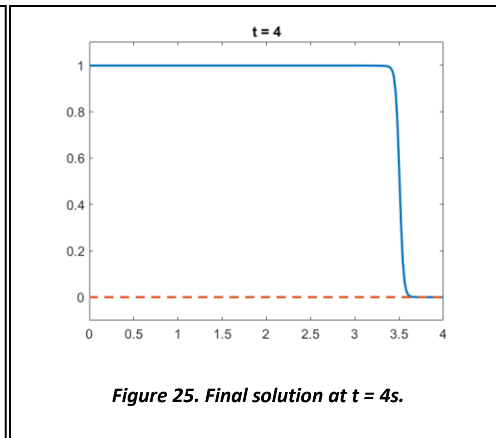

Figure 25. Final solution at t = 4s.

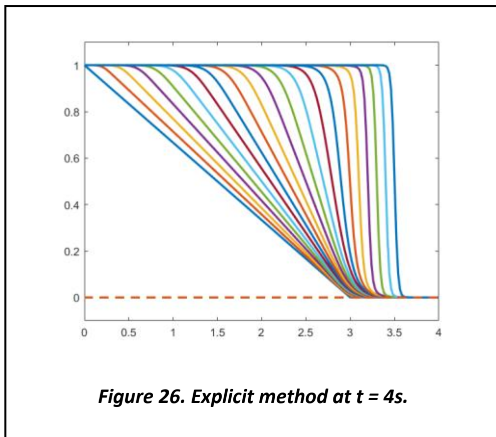

Figure 26. Explicit method at t = 4s.

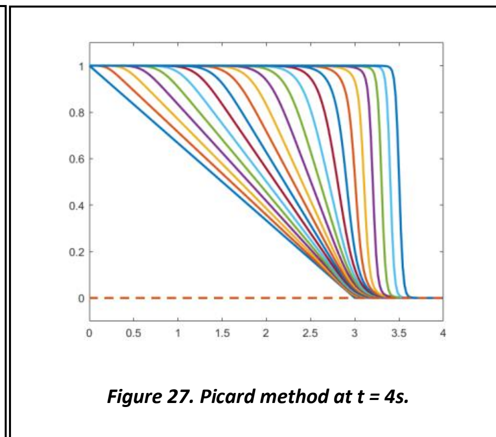

Figure 27. Picard method at t = 4s.

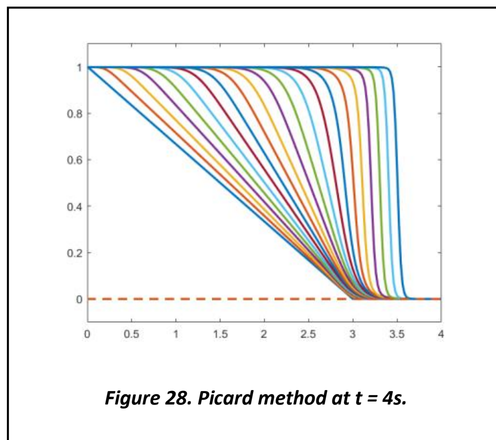

Figure 28. Picard method at t = 4s.

- For the conditions of $\Delta t = 0.05$ and E = 1e-2

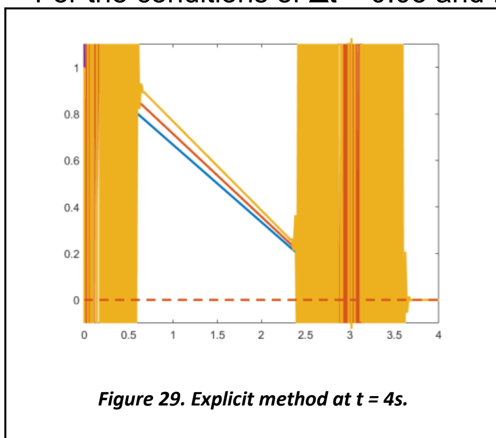

Figure 29. Explicit method at t = 4s.

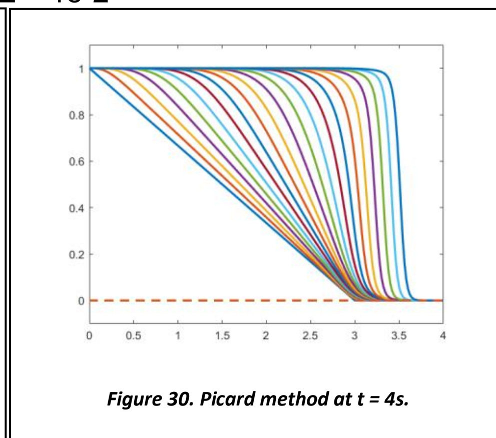

Figure 30. Picard method at t = 4s.

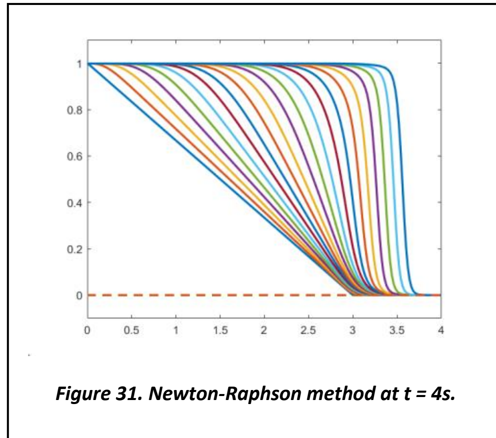

Figure 31. Newton-Raphson method at t = 4s.

- For the conditions of Δt = 0.1 and E = 1e-2

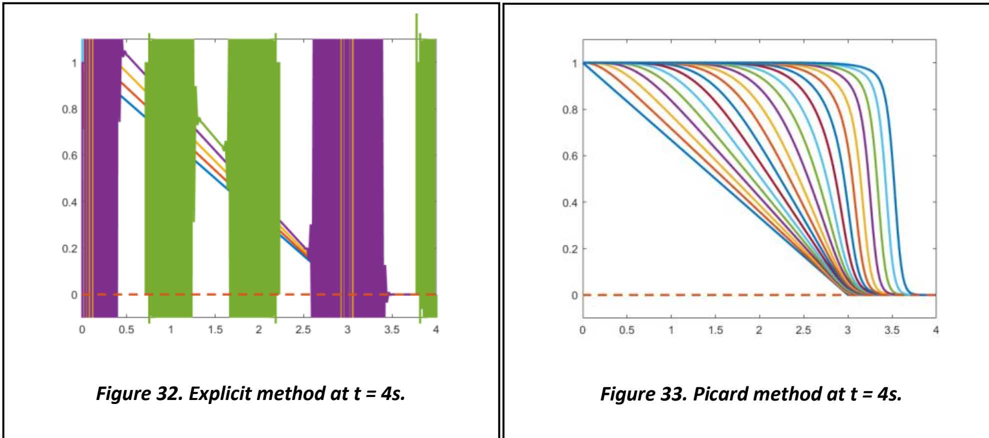

Figure 32. Explicit method at t = 4s.

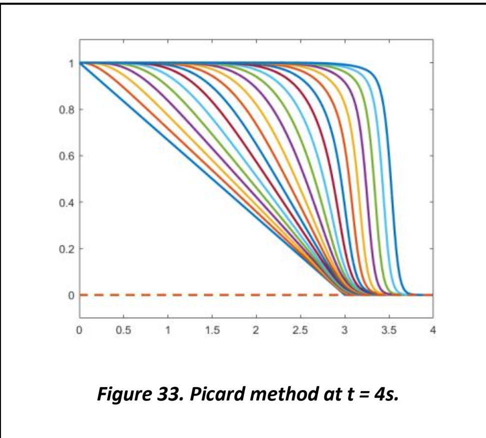

Figure 33. Picard method at t = 4s.

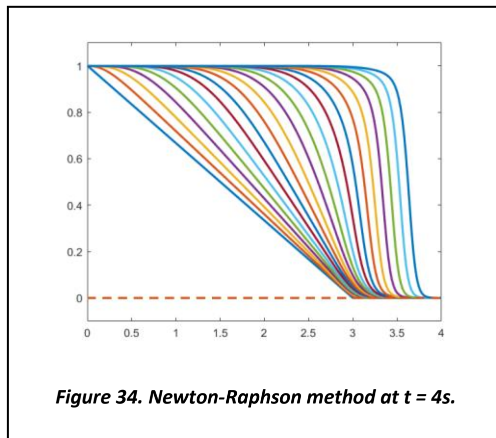

Figure 34. Newton-Raphson method at t = 4s.

- For the conditions of Δt = 0.005 and E = 1e-4

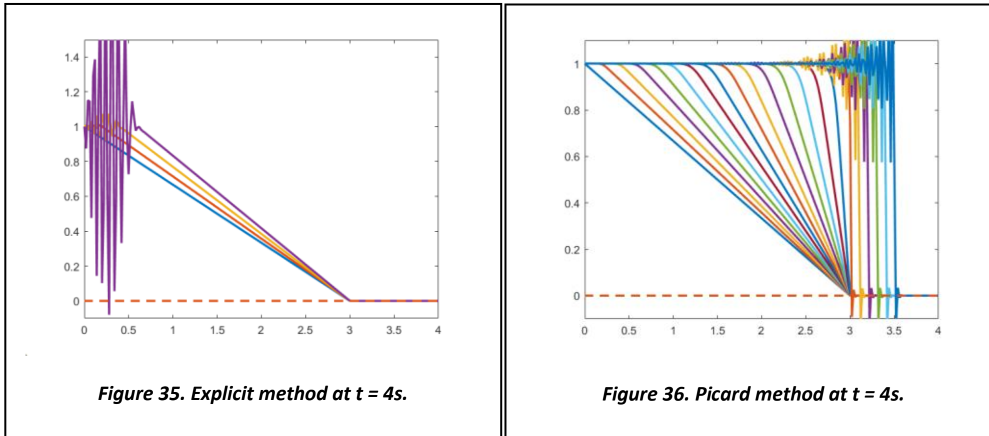

Figure 35. Explicit method at t = 4s.

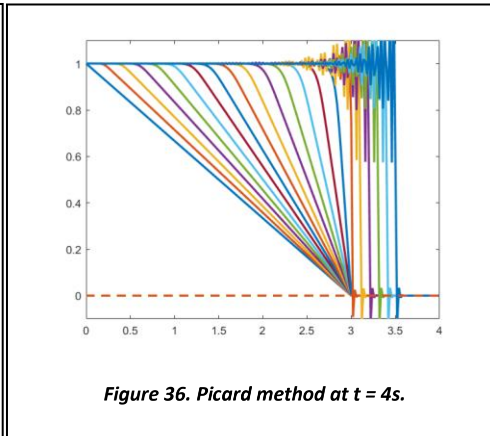

Figure 36. Picard method at t = 4s.

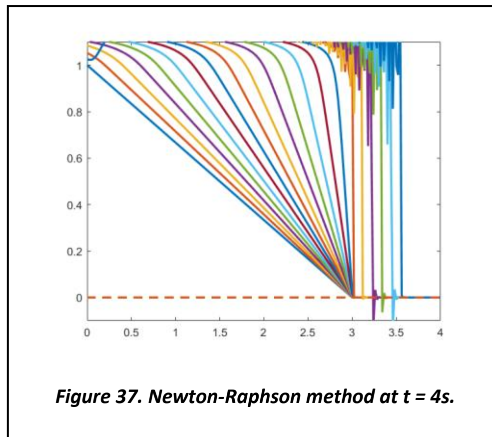

Figure 37. Newton-Raphson method at t = 4s.

Comments on the results

For the case where we have initial decreasing data, it is got that explicit method shuts down when the time step is increased and the diffusion is lowered as it happens between cases $\Delta t = 0.005$ and $E = 1e\text{-}4$ and $\Delta t = 0.005$ and $E = 1e\text{-}2$. Also note that diffusivity helps the implicit methods, for decreasing initial data case, not to have discontinuities in the solution. Time step increment, when diffusivity is large enough, keep the solution of implicit methods stable.

Comments on the code

Code is mainly implemented following the steps in the slides. However, difficulties were faced when differentiating term C(U).

```
for n = 1:nTimeSteps
    %fprintf('\nTime step %d\n', n);
    U0 = U(:,n);
    error_U = 1; k = 0;

    while (error_U > 0.5e-5) && k < 20
        [C] = computeConvectionMatrix(X,T,U0);
        L = (M + At*C + At*E*K)*U0-M*U(:,n);
        P = M + 2*At*C+At*E*K;
        U1 = U0-P\L;
        error_U = norm(U1-U0)/norm(U1);
        fprintf('\t Iteration %d, error_U=%e\n',k,error_U);
        U0 = U1; k = k+1;
    end
    U(:,n+1) = U1;
end
```

First, L is computed as M+At*C+At*E*K times the velocity vector U0 which in this case takes the values that U(:,n) has at every nTimeStep. So it is a vector that changes when the while condition is reached.

In the other hand, M is multiplying a matrix of U(:,n), being n the number of time steps. This is a matrix that gets actualized also when the while condition is reached.

Once the condition is obtained, the U0 vector, in the second iteration (p.e.) will take the value of last column solution of U. This means that last solution is used to compute the new one.

The jacobian in the Newton-Raphson method is the P value in the code. And the evaluation of the function F is L.

So the jacobian is implemented following the slides where the approach of the derivative of the convection matrix with respect to the solution is approximated by means of the residual method by 2 times dt * C.