# FINITE ELEMENTS IN FLUIDS

## Homework 3

### Unsteady convection and Burger's equation

Author: Cristina García Albela
MsC in Computational Mechanics

# 1D Unsteady convection problem

$$\begin{cases} u_t + au_x = 0 & x \in (0,1), t \in (0,0.6) \\ u(x,0) = u_0(x) & x \in (0,1) \\ u(0,t) = 1 & t \in (0,0.06] \end{cases}$$

$$u_0 = \begin{cases} 1 & if\ \ x \leq 0.2 \\ 0 & otherwise \end{cases}$$

$$a = 1 \ \ \Delta x = 2 \cdot 10^{-2} \quad \Delta t = 1.5 \cdot 10^{-2}$$

The propagation of a steep from is going to be studied under 1D problem conditions. The problem takes as the unit the convection speed $a$, and introduces a discontinuity that will be initially located at point $x = 0.2$. Moreover, the inlet condition at any time $t$ is impose, as well as the initial values for time and space steps discretization.

Starting with the imposed Courant number $\left(C = \frac{a \cdot \Delta t}{\Delta x} = \frac{1 \cdot 1.5 \cdot 10^{-2}}{2 \cdot 10^{-2}} = 0.75\right)$ the problem will be solved using three of the studied time discretization schemes, Crank-Nicholson, Lax-Wendroff (both with consistent and diagonal mass matrix) and third-order Taylor-Galerkin. From the obtained results and conclusions, some modifications are going to be introduced playing with the Courant number value, in all of them in order to understand their behaviour.

## MATLAB CODE NOTES

Before starting with the different cases, some remarks about the Matlab code are introduced.

1. Consistent vs. Diagonal Mass matrix. To different ways to define de mass matrix for an element are defined in the code in order to satisfy the different cases.

```
Me = Me + w_ig*(N_ig'*N_ig);
```

*Figure 1. Consistent M matrix code*

```
Me = Me + w_ig*(N_ig'*N_ig);
Me1 = sum(Me,2);
Me = [Me1(1,1) 0; 0 Me1(2,1)];
```

*Figure 2. Diagonal M matrix code*

2. In order to implement the methods, the following matrices are defined for each element
   Mass matrix → `Me = Me + w_ig*(N_ig'*N_ig)`
   Convective matrix → `Ce = Ce + w_ig*(N_ig'*Nx_ig)`
   Stiffnes matrix → `Ke = Ke + w_ig*(Nx_ig'*Nx_ig)`

3. The variables A and B represent l.h.s. and r.h.s of the equation respectively

### 1) Crank-Nicholson scheme (consistent and diagonal) in time and Galerkin scheme in space

To begin, the Crank-Nicholson scheme is implemented, being the only one of the $\theta$-methods with second order accuracy. Starting from the general formulation of $\theta$-methods, with $\theta = {}^1/_2$, taking into account that non-source term $s$ is involved and introducing the weighted residual method for space discretization, the method can be written as

$$\left(w, \frac{\Delta u}{\Delta t}\right) + \left(w, \frac{1}{2}(a \cdot \nabla)\Delta u\right) = -(w, a \cdot \nabla u^n)$$

Substituting in the equation the previous defined matrices, $M, C$ *and* $K$, the method is implemented and solved in Matlab.

```
% Crank-Nicolson + Galerkin
A = M + 1/2*a*dt*C;
B = -a*dt*C;
methodName = 'CN';
```

*Figure 3. Crank-Nicholson + Galerkin Matlab code*

## Results for $C = 0.75$ & $C = 0.20$

The first calculations are made following the data of the problem. Then, the time step size is modified in order to decrease heavily the Courant number for future comparations.

$$Case\ 1 \rightarrow \Delta x = 2 \cdot 10^{-2} \ - \ \Delta t = 1.5 \cdot 10^{-2}$$

$$Case\ 2 \rightarrow \Delta x = 2 \cdot 10^{-2} \ - \ \Delta t = 4 \cdot 10^{-3}$$

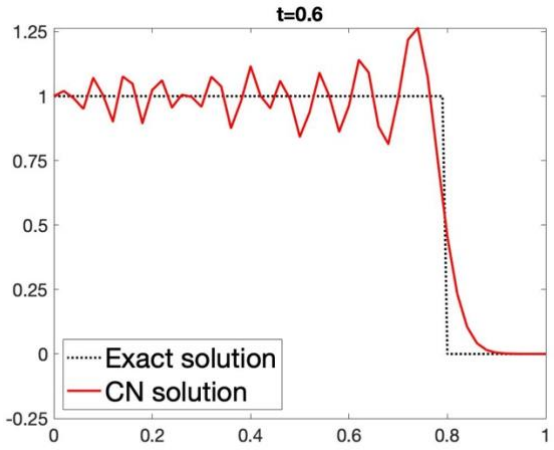


*Figure 4. Consistent M matrix – C = 0.75*

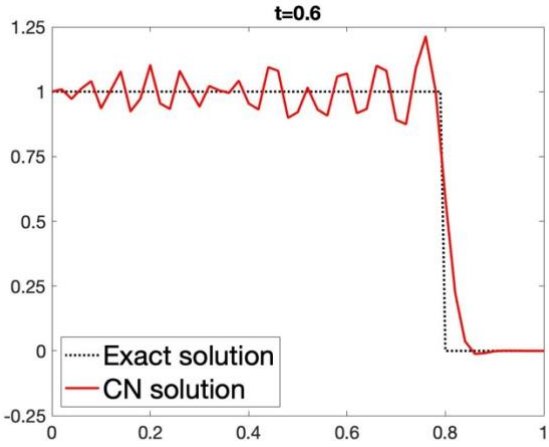


*Figure 5. Consistent M matrix – C = 0.2*

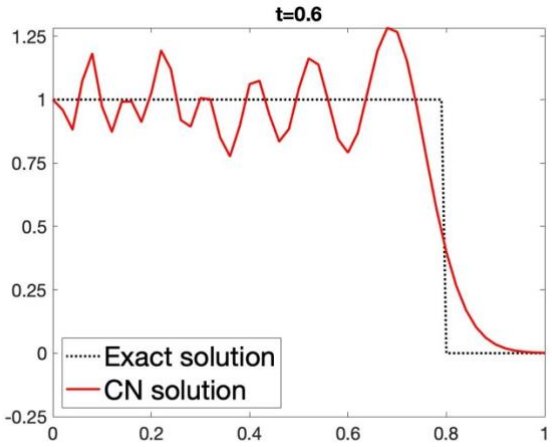


*Figure 6. Diagonal M matrix – C = 0.75*

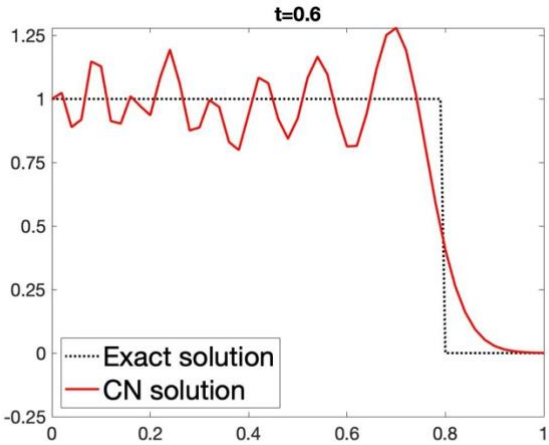


*Figure 7. Diagonal M matrix – C = 0.2*

Both from consistent and diagonal cases, oscillations appear for any Courant number. This is due to the fact that for this kind of problem (steep from), Galerkin formulation introduce oscillations over the domain that remains at the front.

However, different behaviours for consistent and diagonal mass matrix can be observed as Courant number is changed. As the results for consistent CN turn better as Courant number decreases, the opposite occurs when the diagonal M matrix is computed. The consistent mass matrix reduces the stability range to $C^2 \leq 1/3$, whereas the diagonal matrix presents a range of $C \leq 1$, achieving better results as $C > 0.5$.

## 2) Lax-Wendroff method (consistent and diagonal) in time and Galerkin scheme in space.

Obtained from Taylor expansion, this explicit second-order accuracy method is implemented. Taking into account that non-source term $s$ is involved and introducing the weighted residual method for space discretization, the method can be written as

$$\left(w, \frac{\Delta u}{\Delta t}\right) = -(w, a \cdot \nabla u^n) + \left(w, \frac{\Delta t}{2}(a \cdot \nabla)^2 u^n\right)$$

After integrating by parts, substituting by the previous defined matrices, $M, C \text{ and } K$, the method is implemented and solved in Matlab.

```
% Lax-Wendroff + Galerkin
  A = M;
  B = -a*dt*C - dt^2/2*a^2*K;
  methodName = 'LW';
```

*Figure 8. Lax-Wendroff + Galerkin Matlab code*

### Results for $C = 0.75$ & $C = 0.20$

Two now calculations are made using the same values as in the previous method for space and time discretization.

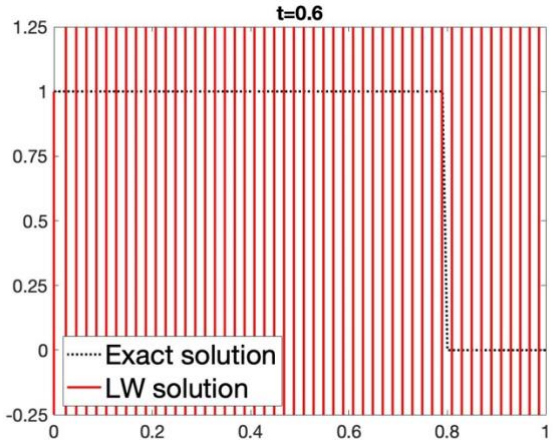


*Figure 9. Consistent M matrix – C = 0.75*

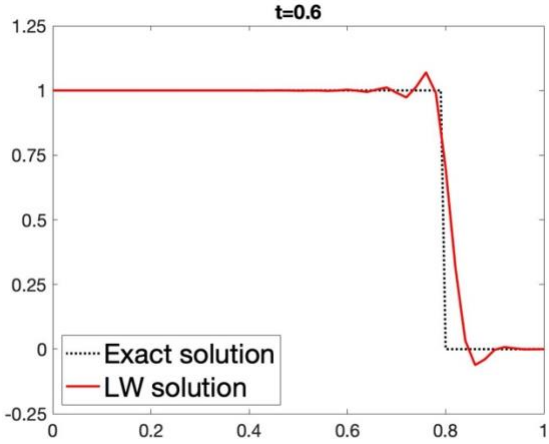


*Figure 10. Consistent M matrix C = 0.2*

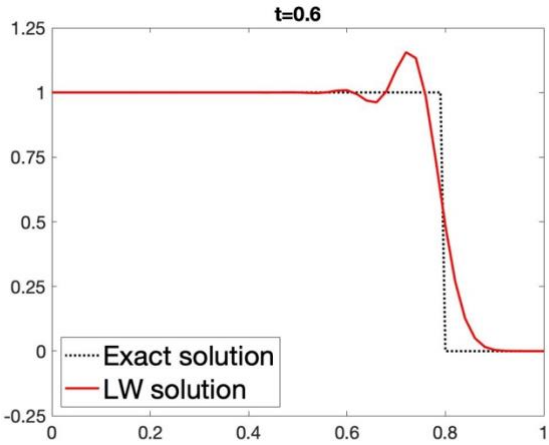


*Figure 11. Diagonal M matrix – C = 0.75*

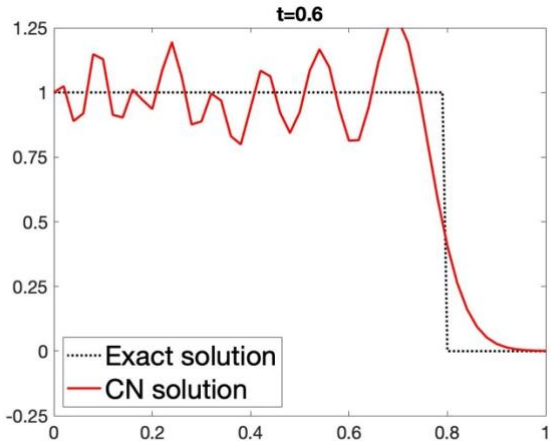


*Figure 12. Diagonal M matrix – C = 0.2*

As it was expected the solution obtained for the consistent LW case with Courant number of 0.75 is completely unstable. The stability parameter for this method is established as $C^2 \leq 1/3$, much smaller than the actual value of the problem. Two different changes can be introducing in order to obtain better results, the diagonal M matrix or changing the time step size.

Decreasing the C number ($C = 0.2$) until it meets the stability condition it is obtained such a better result but taking into account the increment of computational cost as the time step gets smaller.

On the other hand, as it is known that the stability range once the diagonal M matrix is introduced turns to $C \leq 1$ with better results for $C > 0.5$. Solving with the new M matrix the problem again ($C = 0.75$) it changes completely the scene too, achieving stable and accurate results. Remark the fact that, if $C = 0.2 < 0.5$ is computed for the diagonal case, the results turn worse, following the expected behaviour.

### 3) Third order Taylor – Galerkin method

Second-order time schemes don't allow the needed accuracy, thus higher-order schemes are developed to allow a better account of the propagation of information along the characteristics. The explicit 3rd-order Taylor-Galerkin method is obtained from the Taylor expansion up to the third order. Taking into account that non-source term $s$ is involved and introducing the weighted residual method for space discretization, the method can be written as

$$\left(w, \left[1 - \frac{\Delta t^2}{6}(a \cdot \nabla)^2\right] \frac{u^{n+1} - u^n}{\Delta t}\right) = -(w, (a \cdot \nabla)u^n) + \left(w, \frac{\Delta t}{2}(a \cdot \nabla)^2 u^n\right)$$

After integrating by parts and substituting by the previous defined matrices, $M, C$ *and* $K$, the method is implemented and solved in Matlab.

```
%Taylor-Garlekin 3rd order
 A = M + dt^2/6*a^2*K;
 B = -a*dt*C - dt^2/2*a^2*K;
 methodName = 'TG3';
```

*Figure 13. TG3 Matlab code*

**Results for $C = 0.75$ & $C = 0.20$**

The method is implemented twice using the same values as in the previous methods for space and time discretization.

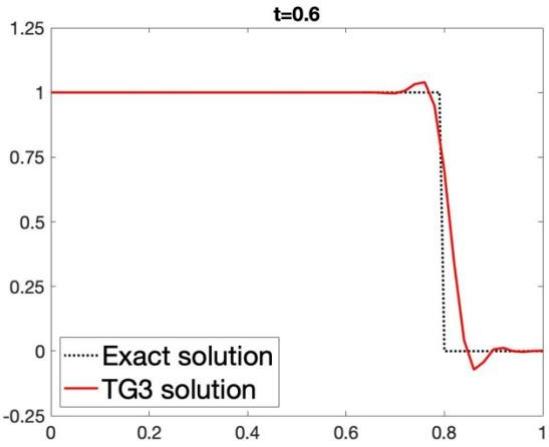


*Figure 14. TG3 – C = 0.75*

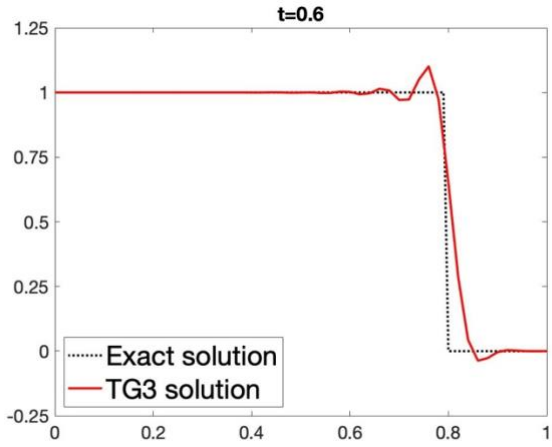


*Figure 15. TG3 – C = 0.2*

Once again, some oscillations along the domain characterized the implemented Galerkin formulation, however the best behaviour of the method is proved if the results are compared with the corresponding ones of TG2 and CN.

Highlighting that the stability range for TG3 is $C^2 \leq 1$, it is observed that the method show better results for a higher Courant number in contrast to the previous two methods with consistent M matrix.

# Burger's Equation

$$\begin{cases} u_t + uu_x = 0 & x \in (0,4), t \in [0,4] \\ u(x,0) = u_0(x) & at\ t = 0 \end{cases}$$

$$\text{Initial condition: } u_0 \begin{cases} \frac{1-x}{3} & if\ x < 3 \\ 0 & if\ x \geq 3 \end{cases}$$

Spatial and time steps sizes: $\Delta x = 0.02 \quad \Delta t = 0.005$

Burger's equation is one of the classical examples of non-linear hyperbolic equations for which the convective velocity is the solution $u$ itself. To work with this kind of equations, the concept of "weak solutions" that admit discontinuities should be taking into account, together with jump and entropy conditions. Applying all these concepts, the physical correct weak solution for Burger's equation corresponds with Burger's equation for the inviscid case

$$u_t^\epsilon + u^\epsilon u_t^\epsilon = \epsilon\, u_{xx}^\epsilon$$

For this case, the initial data decreases from left to right side

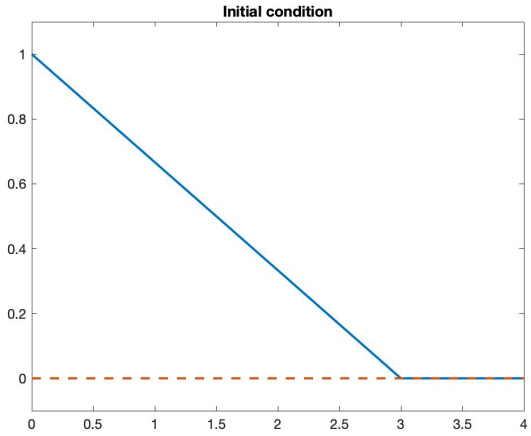


*Figura 1. Initial condition graphic*

Different methods are going to be used in Matlab in order to solve the problem. The explicit method and the implicit Picard method are already written, whereas the implicit Newton-Raphson is going to be implemented. Then, solutions will be compared and discussed, obtaining the final conclusions.

## 1. Implicit Newton – Raphson method. Matlab implementation

For the three method, the problem matrices of mass M, convection C and stiffness K are equally defined. The main differences appear when the iteration process for time discretization is introduced. Taking as start point the FEM discretization of the problem and after some algebra, the solution of a non-linear system of equations applying the N-R method can be briefly described.

Being $f$ the system linear of equations

$$f(U) = (M + \Delta t C(U) + \Delta t E K)U - MU^n$$

solve at each time step $f(U^{n+1}) = 0$. For being able to do it, at each time step the initial data is going to be defined from the previous one, so that

$$^0U^{n+1} = U^n$$

Then, at each time step, the solution is going to be computed using an iterative process (k) until the convergence criterium fixed by the user is satisfied. Here, the main differences with respect to Picard method appears, being the N-R final result computed as

$$^{k+1}U^{n+1} = {}^kU^{n+1} - J\left({}^kU^{n+1}\right) f\left({}^kU^{n+1}\right)$$

The main remarkable fact is the apparition of the Jacobian, defined as the derivative of $f$ with respect $U$, that must be implemented inside the iterative process to take part in the final solution equations.

$$J = \frac{df}{dU} = M + 2\Delta t C(U) + \Delta t E K$$

## Newton-Raphson Matlab code

Taking as basis the Picard method, a new function ("burgers_imNR.m") is created to introduce the N-R method. Moreover, a small modification is made in the C matrix function, to create a vector "u_k" in which the $^kU^{n+1}$ values will be saved at each iterative k step.

```
function [C,u_k] = ComputeConvectionMatrix(X,T,Un)

. . .

C = spalloc(npt,npt,3*npt);
u_k = zeros(numel+1,1);
for ielem = 1:numel
    Te = T(ielem,:);
    Xe = X(Te,:);
    u_e = Un(Te);

    Ce = zeros(nen);
    for ig=1:ngaus
        N_ig   = N(ig,:);
        Nxi_ig = Nxi(ig,:);
        %Jacobia
        J = Nxi_ig*Xe;
        % Derivadas de las funciones de forma
respecto a (x,y)
        Nx_ig = J\Nxi_ig;
        % diferencial de volum
        dvolu = wgp(ig)*det(J);
        u_ig = N_ig*u_e;
        Ce = Ce + N_ig'*(u_ig*Nx_ig)*dvolu;
    end
    C(Te,Te) = C(Te,Te) + Ce;
    u_k(ielem,1) = u_e(1,1);
    if ielem == numel
        u_k(ielem+1,1) = u_e(2,1);
    end
```

*Figura 2. Code modifications at C matrix function*

```
function U = burgers_imNR(X,T,At,nTimeSteps,u0,uxa,uxb,E)

. . .

for n = 1:nTimeSteps
    %fprintf('\nTime step %d\n', n);
    bccd = [uxa; uxb];
    U0 = U(:,n);
    error_U = 1; k = 0;
    while (error_U > 0.5e-5) && k < 20
        [C,u_k] = computeConvectionMatrix(X,T,U0);
        A = M + At*C + At*E*K;
        f = A*u_k - M*U(:,n);
        J = M + 2*At*C + At*E*K;
        Z = J\f;
        sol = u_k - Z;
        U1 = sol(1:m+1);
        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
```

*Figura 3. N-R code main points*

## 2. Results and conclusions

Once the three methods are implemented, the problem is solve applying the mentioned variables and discretization values. Them, different values of time discretization and $E$ are going to be implement comparing and discussing the results.

### $\Delta t = 0.005$ and $E = 0.01$

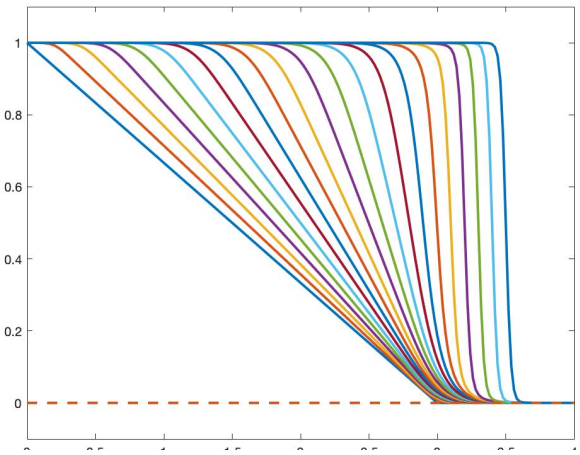


*Figura 4. Explicit method*

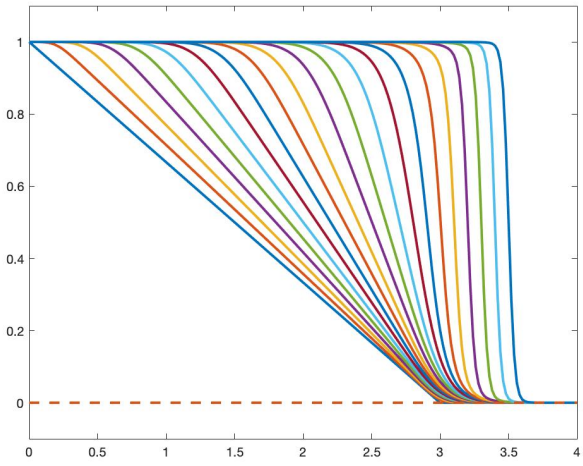


*Figura 5. PICARD implitic method*

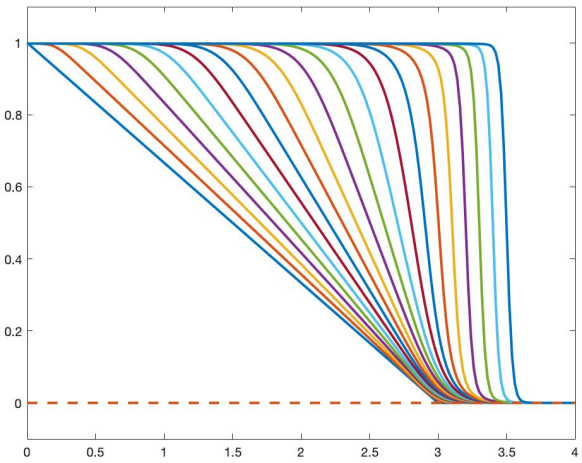


*Figura 6. N-R implicit method*

For the initial case, all the methods achieve an accurate solution. If the Pe number is computed, being the highest value for the convective velocity $a$ around 1, it's checked that $Pe \leq 1$, so that the effects of convective dominant problems are not going to appear. Moreover, with a Courant number $C \leq 0.25$ stability is guaranteed for the three methods, explicit and implicit ones.

**$\Delta t = 0.005$ and $E = 0.0001$**

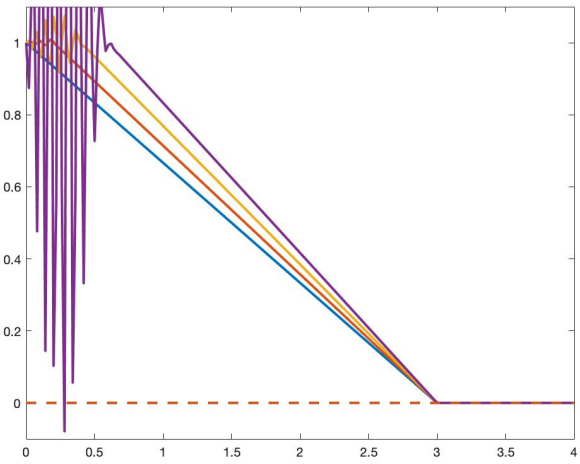


*Figura 7. Explicit method*

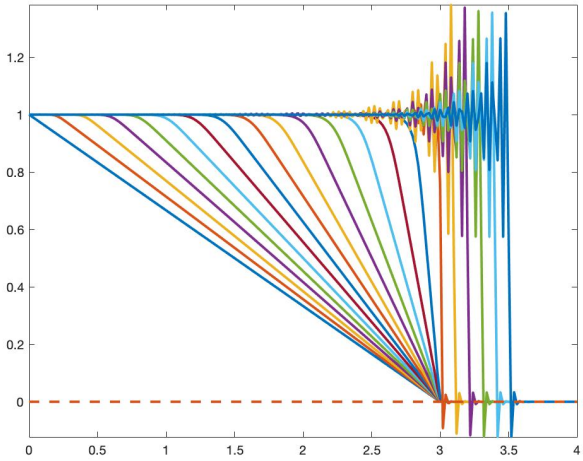


*Figura 8. PICARD implicit method*

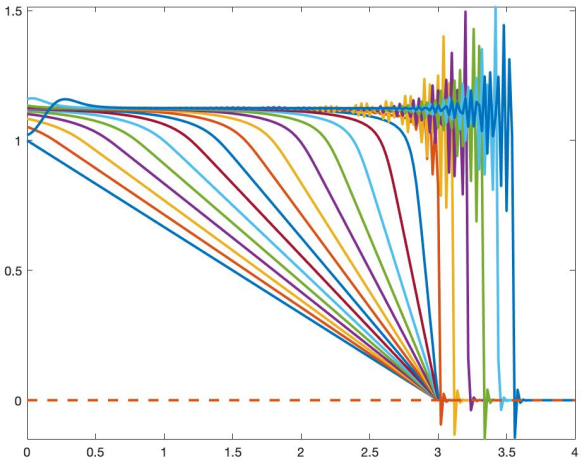


*Figura 9. N-R implicit method*

A change on the viscosity value has been introduced and with such a small viscosity value, oscillations appear. Decreasing the viscosity parameter, the Pe number tunrs to be $Pe \leq 100$, turning not possible for Galarkin formulations to reproduce the solution without oscillations, that appears around the singularity.
The solution for implicit methods is much closer to the previous one while the explicit one is completely wrong. This shows the stronger stability conditions that the implicit methods present, at the expence of a higher computational cost.

**$\Delta t = 0.1$ and $E = 0.01$**

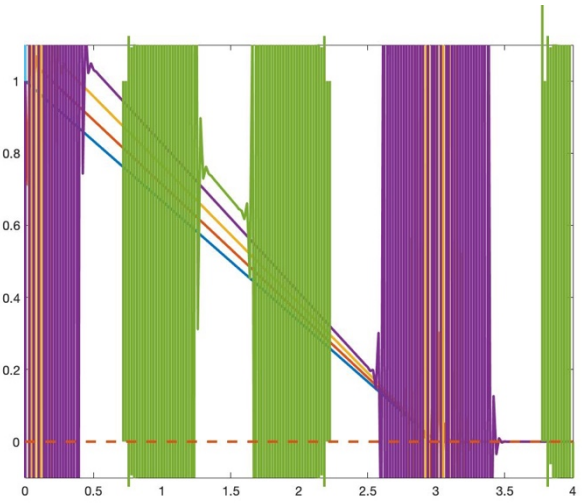


*Figura 10. Explicit method*

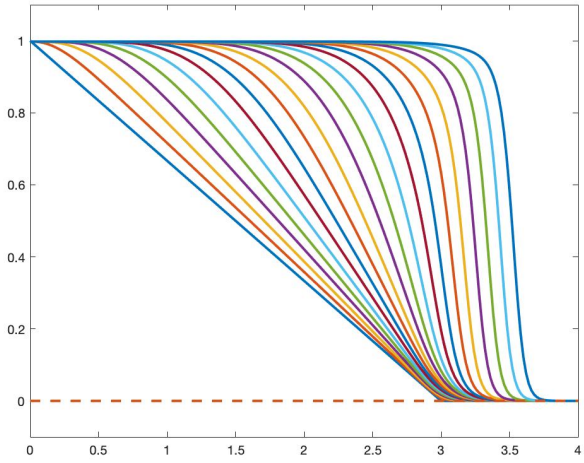


*Figura 11. PICARD implicit method*

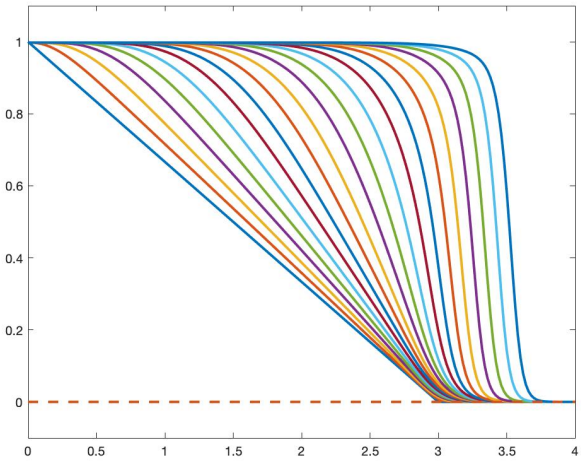


*Figura 12. N-R implicit method*

For this last case, recovering the initial viscosity $Pe \leq 1$, eliminating the problems of convective dominant problems; and Courant number turns to $C \leq 5$ due to the time step increase. Once again implicit method better stability conditions can be proved, with no oscillations in both cases, while at the explicit case the solution is a completely mess.

After studying different cases, it is concluded that implicit methods offer better results for Burger's equation than the explicit one, but for all of the them the accuracy of the results will be conditioned by Péclet and Courant numbers. If the point is at choosing between the two implicit options, probably the key point to take into account will be the quadratic convergence that N-R will offer in front of Picard method.