FEF Lab Assignment 2

The report is completed in two parts. In the first part, the author compared different temporal schemes for solving unsteady transport problems – Crank-Nicholson, Crank-Nicholson with lumped mass matrix, Lax-Wendroff, Lax-Wendroff with lumped mass matrix, third-order explicit Taylor- Galerkin method. In the section part, a non-linear compressible flow problem is studied using Newton-Raphson scheme and the results from different schemes for solving nonlinear systems are compared.
At the end of the file, a more detailed version of maths derivation of the discrete system can be found.

Part One: Transient convection problem

The strong form of the pure convection problem is written as

$$\begin{cases} u_t + \boldsymbol{a} \cdot \boldsymbol{\nabla} u = 0 \quad \mathbf{x} \in \Omega \\ u(\boldsymbol{x}, 0) = u_0 \quad \boldsymbol{x} \in \Gamma_D \end{cases} \tag{1}$$

1 Galerkin formulation of different methods

1.1 Lax-Wendroff

In the following derivation of weak forms, we neglect the source term as it is 0, so the expression is much simplified.
The Lax-Wendroff(TG2) method gives the following time discretization form

$$\frac{\triangle u}{\triangle t} = -\boldsymbol{a} \cdot \boldsymbol{\nabla} u^n + \frac{\triangle t}{2} (\boldsymbol{a} \cdot \boldsymbol{\nabla})^2 u^n \tag{2}$$

The weak form associated with this model problem is

$$\big(w, \frac{\triangle u}{\triangle t}\big)_\Omega = -\big(w, \boldsymbol{a} \cdot \boldsymbol{\nabla} u^n - \frac{\triangle t}{2} (\boldsymbol{a} \cdot \boldsymbol{\nabla})^2 u^n\big)_\Omega \tag{3}$$

After integration by parts, it is given by

$$\big(w, \frac{\triangle u}{\triangle t}\big)_\Omega = \big(\boldsymbol{a} \cdot \boldsymbol{\nabla} w, u^n - \frac{\triangle t}{2} (\boldsymbol{a} \cdot \boldsymbol{\nabla}) u^n\big)_\Omega - \big\langle (\boldsymbol{a} \cdot \mathbf{n}) w, u^n - \frac{\triangle t}{2} (\boldsymbol{a} \cdot \boldsymbol{\nabla}) u^n \big\rangle_{\partial\Omega} \tag{4}$$

For the 1D scalar case, the weak form will be discretised with shape functions. At last we obtain a linear system

$$\mathbf{M} \triangle \boldsymbol{u} = (-a \triangle t \mathbf{C} - \frac{\triangle t^2}{2} a^2 \mathbf{K} + \mathbf{B}_{out}) \mathbf{u}^n + \mathbf{f} \tag{5}$$

where the matrices are defined at the elemental level by

$$\mathbf{M}_{ij}^e = \int_{\Omega_e} N_i N_j d\Omega_e \tag{6a}$$

$$\mathbf{K}_{ij}^e = \int_{\Omega_e} N_{xi} N_{xj} d\Omega_e \tag{6b}$$

$$\mathbf{C}_{ij}^e = \int_{\Omega_e} N_i N_{xj} d\Omega_e \tag{6c}$$

In the given examples, $\mathbf{B}_{out} = \mathbf{0}$.
The lumped mass matrix $\mathbf{M}^e$ is defined at the elemental level by

$$\mathbf{M}_{ij}^e = \int_{\Omega_e} \begin{bmatrix} N_1 & 0 \\ 0 & N_2 \end{bmatrix} d\Omega_e \tag{7}$$

In the code '$FEM\_matrices$.m' we add

```
Me = Me + w_ig*(N_ig'*N_ig);
Mle = diag(sum(Me,2));
```

which calculated the elemental lumped matrix.
In the function 'System'.m, we change the LW method to:

```
A = M;
B = -a*dt*C-dt*dt/2*a^2*K;
```

If it is Lax-Wendroff with lumped mass matrix, we apply lumped matrix $\mathbf{M}$.

1.2 Crank-Nicholson

The Crank-Nicholson method gives the following time discretization form

$$\frac{\triangle u}{\triangle t} + \frac{1}{2}(\boldsymbol{a}\cdot\boldsymbol{\nabla}) \triangle u = -\boldsymbol{a}\cdot\boldsymbol{\nabla} u^n \tag{8}$$

Similarly the Galerkin weak form is

$$\begin{aligned}\big(w, \frac{\triangle u}{\triangle t}\big)_\Omega - \frac{1}{2}\big(\boldsymbol{\nabla} w, \boldsymbol{a} \triangle u\big)_\Omega + \frac{1}{2}\big\langle (\boldsymbol{a}\cdot\mathbf{n}) w, \triangle u\big\rangle_{\partial\Omega} = \\ \big(\boldsymbol{a}\cdot\boldsymbol{\nabla} w, u^n\big)_\Omega - \big\langle (\boldsymbol{a}\cdot\mathbf{n}) w, u^n\big\rangle_{\partial\Omega}\end{aligned} \tag{9}$$

For the 1D scalar case, the weak form will be discretised with shape functions. At last we obtain a linear system

$$(\mathbf{M} + \frac{a\triangle t}{2}\mathbf{C} + \frac{a\triangle t}{2}\mathbf{B}_{out})\triangle\boldsymbol{u} = (-a\triangle t\mathbf{C} - \mathbf{B}_{out})\mathbf{u}^n + \mathbf{f} \tag{10}$$

$\mathbf{B}_{out}$ can be neglected in this example.

```
A = M + 1/2*a*dt*C;
B = -a*dt*C;
```

If it is Crank-Nicolson with lumped mass matrix, we apply lumped matrix $\mathbf{M}$.

1.3 Third order Taylor-Galerkin

The TG3 method gives the following time discretization form

$$[1-\frac{\triangle t^2}{6}(\boldsymbol{a}\cdot\boldsymbol{\nabla})^2]\frac{\triangle u}{\triangle t}=-\boldsymbol{a}\cdot\boldsymbol{\nabla}u^n+\frac{\triangle t}{2}(\boldsymbol{a}\cdot\boldsymbol{\nabla})^2u^n \tag{11}$$

Follow the same procedure as we did before, the linear system is

$$(\mathbf{M}+\frac{a^2\triangle t^2}{6}\mathbf{K})\triangle\boldsymbol{u}=(-a\triangle t\mathbf{C}-\frac{\triangle t^2a^2}{2}\mathbf{K}+\mathbf{B}_{out})\mathbf{u}^n+\mathbf{f} \tag{12}$$

In the given examples, $\mathbf{B}_{out}=\mathbf{0}$. The implementation of the code is:

```
A = (M+dt^2/6*a^2*K);
B = -a*dt*C-dt*a*dt/2*a*K;
```

2 Comparison between different methods

The following results are calculated from the example of the propagation of a step front.

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

$$u_0(x)=\begin{cases}1 & \text{if } x\le 0.2, \\ 0 & \text{otherwise}\end{cases}$$

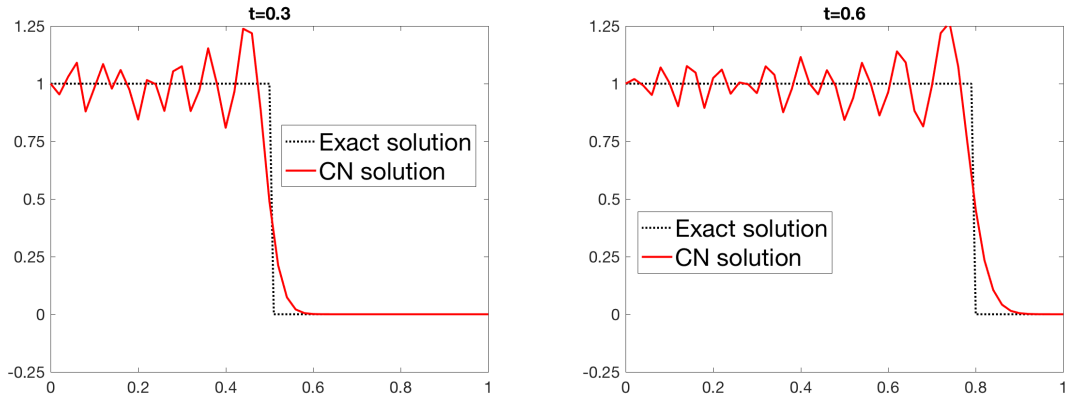
$$a=1,\Delta x=2\cdot10^{-2},\Delta t=1.5\cdot10^{-2}$$

(1)Compute the Courant number.

The Courant number is

$$C=\frac{a\triangle t}{h}=0.75 \tag{13}$$

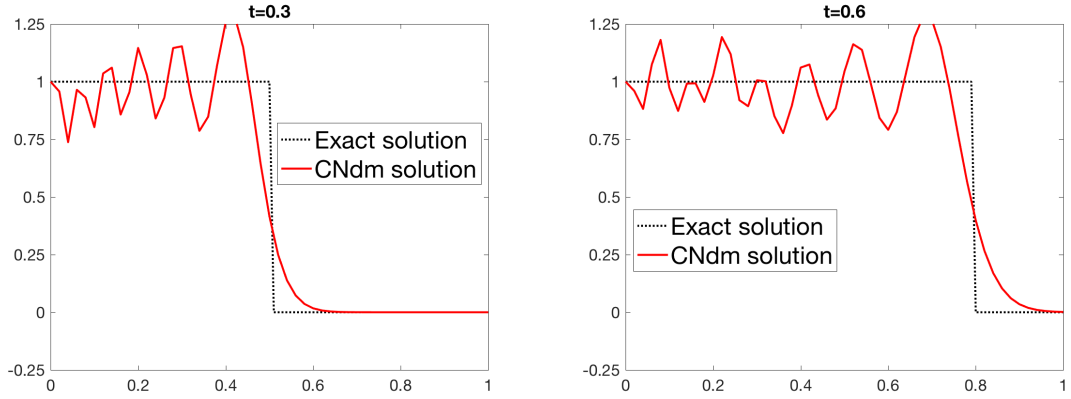
(2)Solve the problem using the Crank-Nicholson scheme in time and linear finite element for the Galerkin scheme in space. Is the solution accurate?
As we can see from Figure.1, the solution from the Crank-Nicholson scheme is stable but not very accurate.



(a) t=0.3 (b) t=0.6

Figure 1: Crank-Nicholson

If we solve it with lumped mass matrix, the solution is also stable but less accurate.

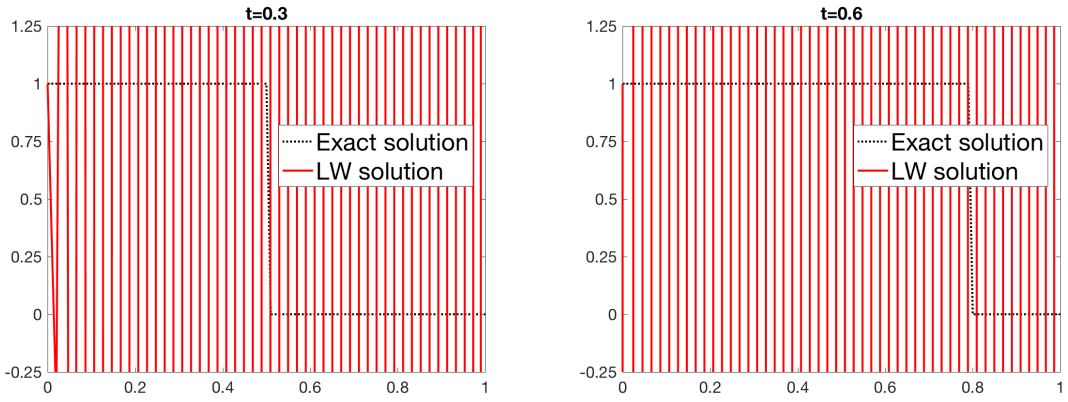


(a) t=0.3 (b) t=0.6

Figure 2: Crank-Nicholson with lumped mass matrix

(3)Solve the problem using the second-order Lax-Wendroff method. Can we expect the solution to be accurate? If not, what changes are necessary? Comment the results.

Figure3 illustrates that the Lax-Wendroff method is stable when the Courant number is too large. In this case $C = 0.75$ is greater than the threshold $C = \frac{\sqrt{3}}{3}$, so the results are not accurate at all.

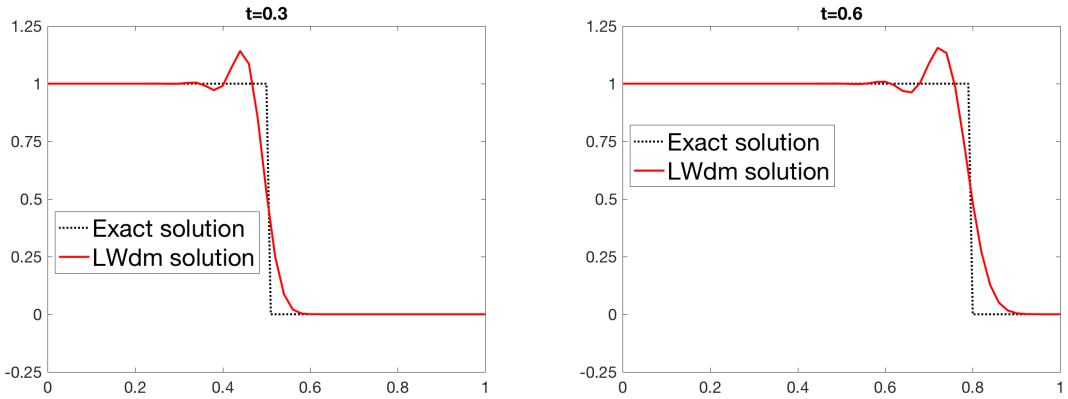


(a) t=0.3 (b) t=0.6

Figure 3: Lax-Wendroff

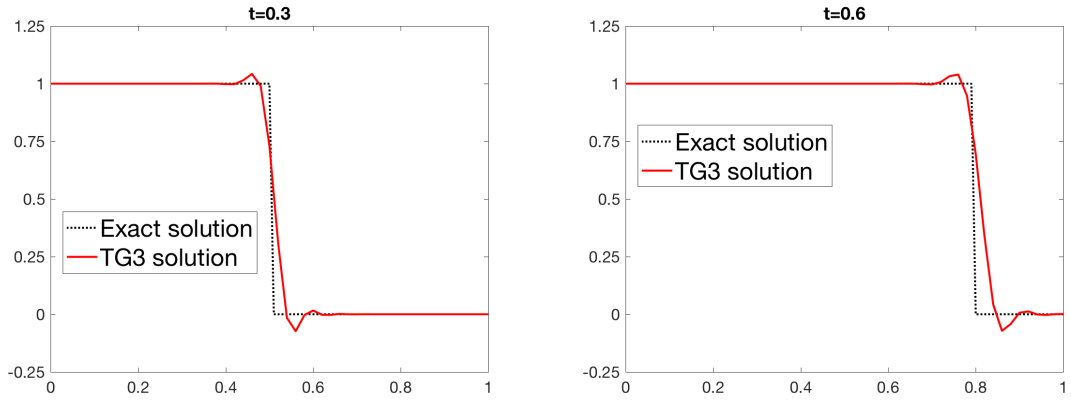
We need C-N method combined with a diagonal mass to stabilize the results, which is demonstrated in Figure.4.

(4)Solve the problem using the third-order explicit Taylor- Galerkin method. Comment the results.
Figure.5 shows that TG3 gives stable and more accurate results than L-W and C-N.



(a) t=0.3 (b) t=0.6

Figure 4: Lax-Wendroff with lumped mass matrix



(a) t=0.3 (b) t=0.6

Figure 5: TG3

Part Two: Solution of the non-linear system

In this part, we implement Newton-Raphson method to solve the non-linear Burger's equation.

$$\begin{aligned} u_t + uu_x &= \epsilon u_{xx} \\ u(x,0) &= u_0(x) \end{aligned} \tag{14}$$

3 Newton-Raphson method

After FEM discretization, we obtain the linear system

$$\mathbf{M}\dot{\mathbf{U}} + \mathbf{C}(\mathbf{U})\mathbf{U} + \epsilon\mathbf{K}\mathbf{U} = \mathbf{0} \tag{15}$$

At each time step, we have to solve

$$\boldsymbol{f}(\mathbf{U}^{n+1}) = \mathbf{0} \tag{16}$$

with $\boldsymbol{f} = (\mathbf{M}+\triangle t\mathbf{C}(\mathbf{U})+\triangle t\epsilon\mathbf{K})\mathbf{U}-\mathbf{M}\mathbf{U}^n = \mathbf{A}(\mathbf{U})\mathbf{U}-\mathbf{M}\mathbf{U}^n$, where $\mathbf{A}(\mathbf{U}) = \mathbf{M}+\triangle t\mathbf{C}(\mathbf{U})+\triangle t\epsilon\mathbf{K}$
The iteration part is

$$\mathbf{U}_{k+1}^{n+1} = \mathbf{U}_k^{n+1} - \mathbf{J}^{-1}(\mathbf{U}_k^{n+1})\boldsymbol{f}(\mathbf{U}_k^{n+1}) \tag{17}$$

where $\mathbf{J} = \frac{\partial \boldsymbol{f}}{\partial \mathbf{U}} = \mathbf{A}(\mathbf{U}) + \triangle t\frac{\partial \mathbf{C}(\mathbf{U})}{\partial \mathbf{U}}\mathbf{U}$. For the burger's equation, we have

$$\begin{aligned} \frac{\partial \mathbf{C}^e}{\partial \mathbf{U}}\mathbf{U} &= \int_{\Omega_e} \mathbf{N}^T(\mathbf{N}_x \cdot \mathbf{U}_e)\mathbf{N}dx \\ \mathbf{C}^e &= \int_{\Omega_e} \mathbf{N}^T(\mathbf{N} \cdot \mathbf{U}_e)\mathbf{N}_x dx \end{aligned} \tag{18}$$

In the code, we change the Newton-Raphson part as

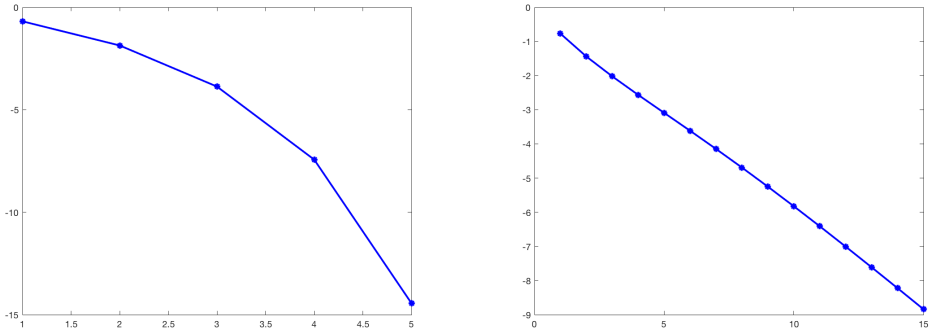
```
for n = 1:nTimeSteps
    %fprintf('\nTime step %d\n', n);
    bccd = [uxa; uxb];
    U0 = U(:,n);
    error_U = 1; k = 0;
    while (error_U > 0.5e-5) && k < 20
        C = computeConvectionMatrix(X,T,U0);
        G = dCdU_U(X,T,U0);
        A = M + At*C + At*E*K;
        J = A+At*G;
        J_T = [J,Accd'; Accd,zeros(size(Accd,1))];
        A_T = [A,Accd'; Accd,zeros(size(Accd,1))];
        f = M*U(:,n);
        f_T = [f;bccd];
        R_T = f_T-A_T*[U0;bccd];
        du = J_T\R_T;
        U1 = U0+du(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;
        res = [res error_U];
    end
    U(:,n+1) = U1;
end
```

$\frac{\partial \mathbf{C(U)}}{\partial \mathbf{U}}\mathbf{U}$ is implemented as

```
function  G= dCdU_U(X,T,Un)
ngaus = 2;
zgp = [-sqrt(3)/3; sqrt(3)/3];
wgp= [1  1];
N = [(1-zgp)/2 (1+zgp)/2];
Nxi = [-1/2  1/2; -1/2  1/2];

npt = size(X,1);
[numel,nen] = size(T);
G = spalloc(npt,npt,3*npt);
for ielem = 1:numel
    Te = T(ielem,:);
    Xe = X(Te,:);
    u_e = Un(Te);

    Ge = 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
```



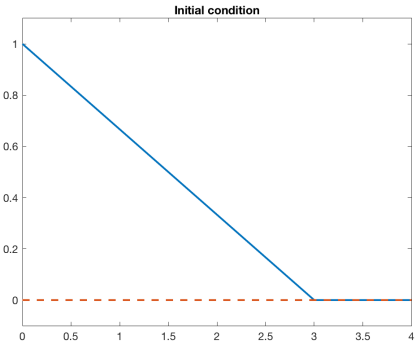
(a) Newton Raphson (b) Picard method

Figure 6: Convergence rate of both implicit methods with $\triangle t = 0.8$, $\epsilon = 0.01$

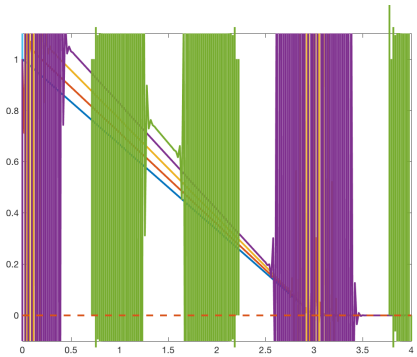
```
25          dvolu = wgp(ig)*det(J);
26
27          Ge = Ge + N_ig'*(Nx_ig*u_e)*N_ig*dvolu;
28      end
29      G(Te,Te) = G(Te,Te) + Ge;
30  end
```

4 Numerical results

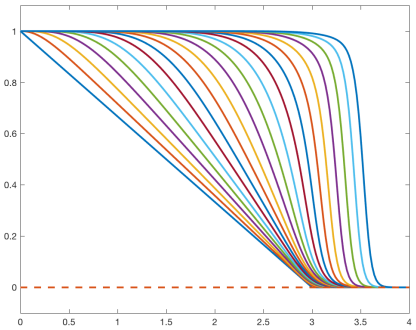
The simulation is done with both increasing and decreasing initial conditions. From Figure.7, it can be observed that explicit method fails when the time step is 0.1 while the implicit methods can give us accurate results. The stability of the explicit method can be achieved by decreasing the time step as shown in Figure.8. However, it is unefficient and computationally expensive. Therefore, implicit methods are preferable for the transient non-linear system, as they are unconditionally stable and computationally cost less.
If we look at the convergence rate of both implicit methods in just one time step – Figure 6, it can be observed that the Newton Raphson method converges quadratically while the Picard method is of linear convergence rate.



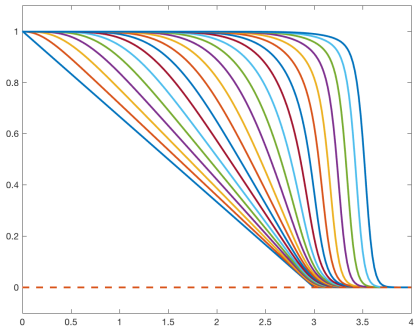
(a) Initial condition decrease



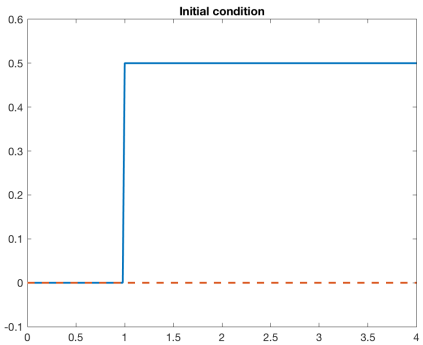
(b) Explicit method



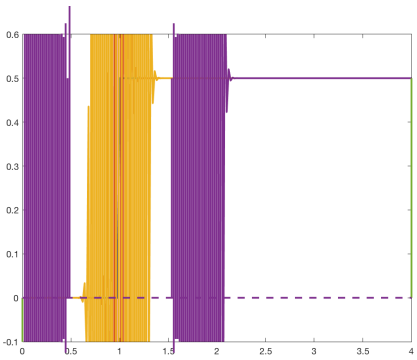
(c) Implicit method (Picard)



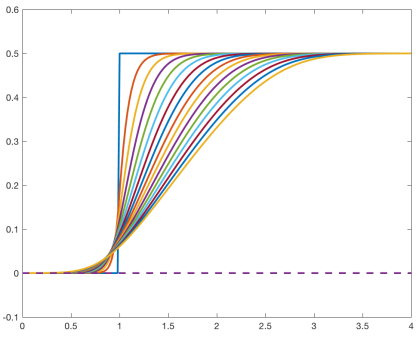
(d) Implicit method (N-R)



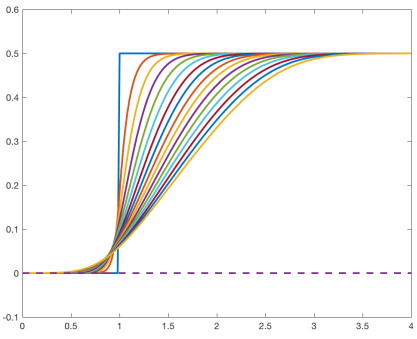
(e) Initial condition increase



(f) Explicit method



(g) Implicit method (Picard)



(h) Implicit method (N-R)

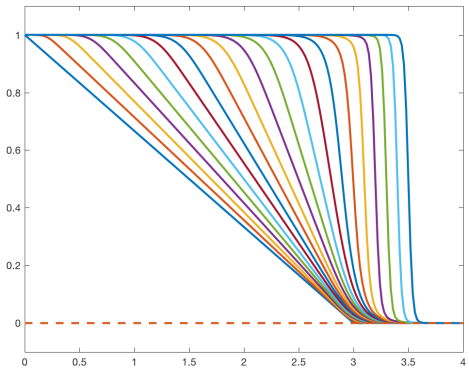


Figure 8: Explicit method with $\triangle t = 0.005$, $\epsilon = 0.01$