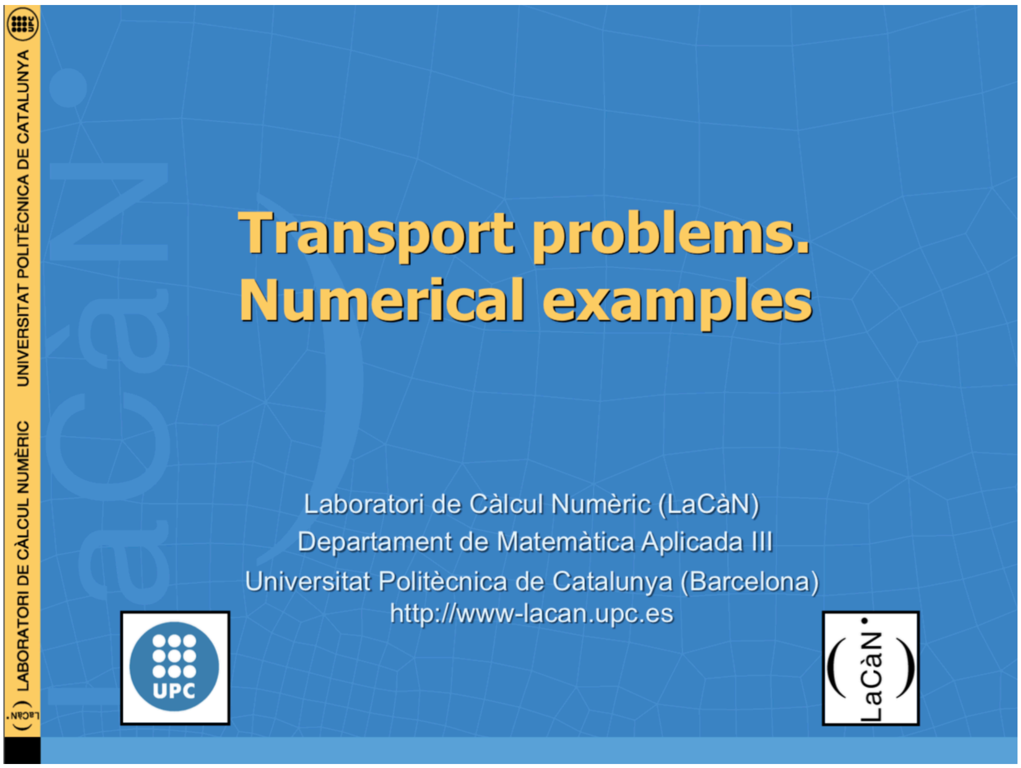


Transport problems.

NUMERICAL EXAMPLES

Iberico Leonardo, Juan Diego | Finite Elements in Fluids | 02/04/2018

Different problems available

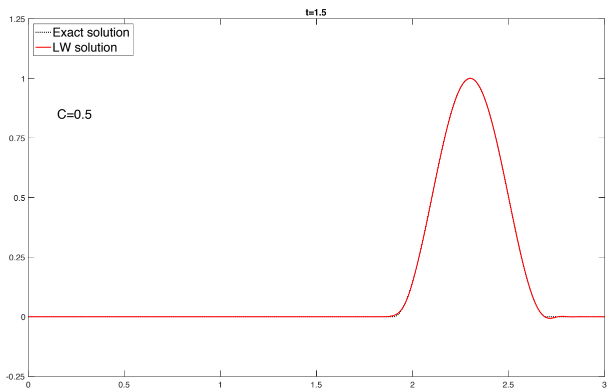
The following report it is focus on unsteady pure convection problem. The aim of the document it is to show the different behavior of the available schemes (time discretization) to solve a pure transport problem. All the schemes are discretized in the space with the Galerkin formulation.

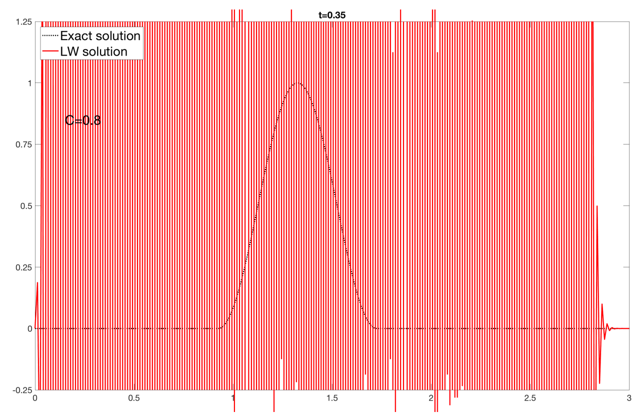
The equation to be solved is: $u_t + a\, u_x = 0$

<u>Remark:</u> the boundary conditions on the outflow boundary are not considered, because the front does not reach it.

LAX-WENDROFF

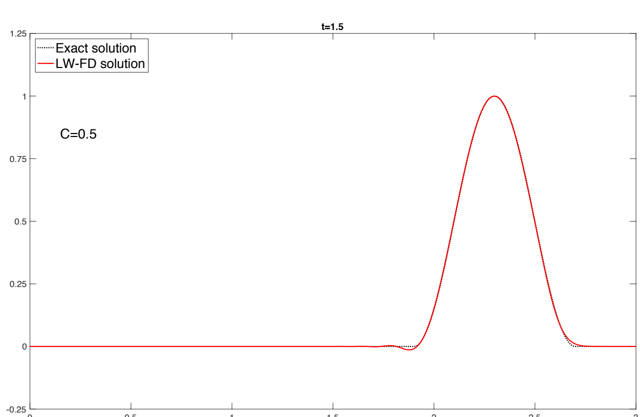
This method is its explicit and second order accuracy, being conditionally stable ($C^2 < 1/3$)

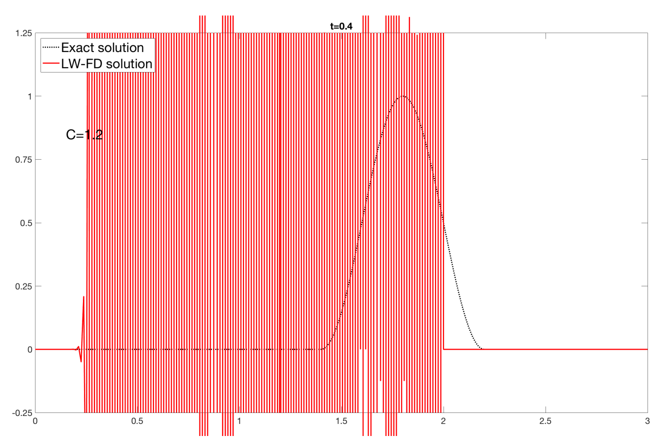




As it can be appreciated on both figures, the behavior of the scheme Lax-Wendorff it is conditionally stable. The figure with courant number 0.5 it is inside the margin ($C^2 < 1/3$), which reproduce almost the exact transient solution with an accuracy of second order.

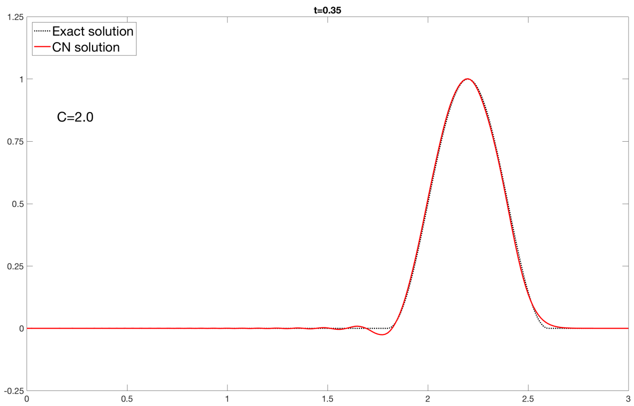
LAX-WENDROFF (MASS MATRIX)

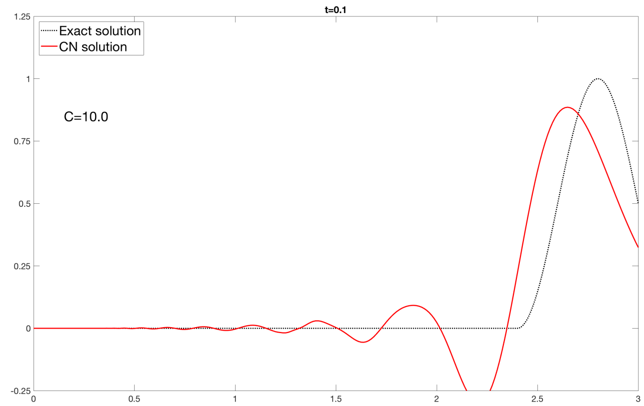




Adding the Lumped mass matrix (diagonalization of the current mass matrix), it is possible to work with higher values of courant number ($C^2 < 1/3$) to ($C^2 < 1$),but introducing numerical error as it can be seen.

CRANK- NICOLSON





In that example, using the theta family for time discretization, we find unconditional stable Crank-Nicolson scheme. For any value of the courant number the solution it is stable, but the introduction of numerical error it is increasing, leading to higher order time discretization schemes as it is presented on the following lines.

EXERCISE

Write down the Galerkin formulation and modify your code to be able to use:

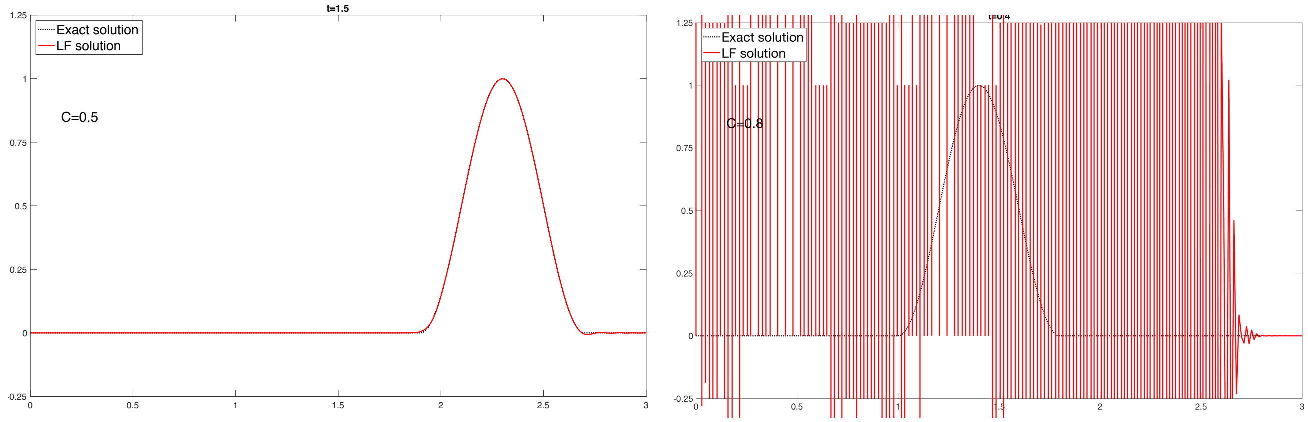
- The leap frog method.
  This method it was implemented in a new specific function, adding new variables as H and J for the first step (calling the need to use the information from two previous points).

```
function [A,B,H,J,methodName] = System_5(method,M,K,C,a,dt)


switch method

    case 5 % Two Step Leap-Frog + Galerkin
        A = M;
        B = -2*dt*a*C;
        % defining the variable for 1st step of LG (uses LW)
        H = M;
        J = -a*dt*C- 0.5*a^2*dt^2*K;
        methodName = 'LF';

    otherwise
        error('not available method')
end
```

  Solving the system, it was modified, adding the first iteration to be done by Lax-Wendorff and posteriori continue the remaining steps with leap frog.

```
% Diego Implementation ""LEAP FROG"" method 5
if method == 5
    % First iteration Lax-Wendroff
    H = H(ind_unk,ind_unk);
    J = J(ind_unk,ind_unk);
    Du_n0 = H\(J*u(ind_unk,1) + f);
    u(ind_unk,2) = u(ind_unk,1) + Du_n0;
    % Second iteration
    for n = 2:nStep
    u(ind_unk,n+1)  = u(ind_unk,n-1) + A\(B*u(ind_unk,n) + f);
    end
end
```

The leap frog scheme is using the information from the two previous points u_{n-1} and u_n for compute the new the step u_{n+1} . Related to the stability, the method it keeps the courant number range as Lax-Wendorff being that ($C^2 < 1/3$) and explicit method of second order accuracy.



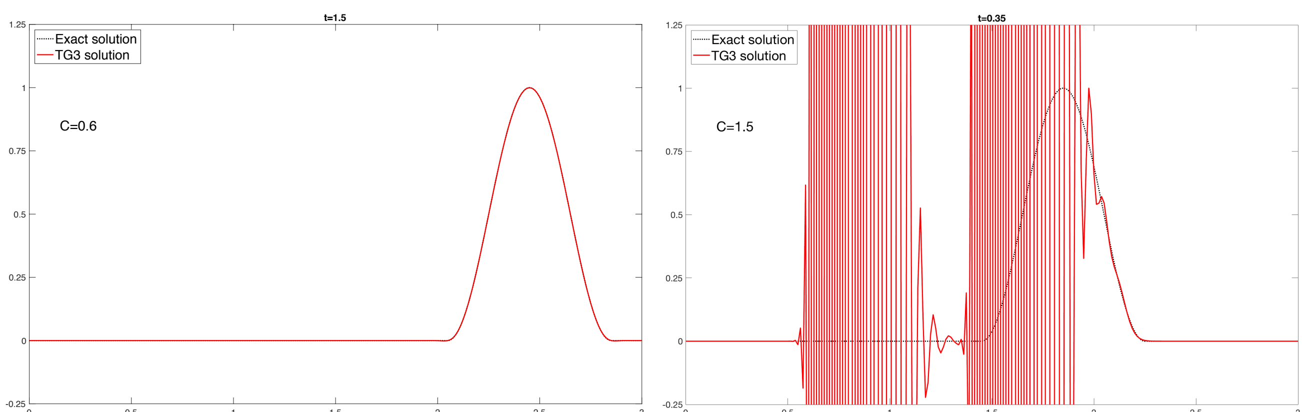
As it can be seen, the two courant numbers reflect the stability range for the Leap frog scheme with the first iteration done by Lax-Wendorff.

- The third order and two-step third-order Taylor-Galerkin method (use $\alpha = 1/9$ to reproduce the phase-speed characteristics of the TG scheme).

The TG3 scheme is implemented on the general function system as it follows:

```
case 6 % Taylor-Garlerkin TG3
    A = M + (1/6)*dt^2*a^2*K;
    B = -a*dt*C -a^2*0.5*dt^2*K;
    methodName = 'TG3';
```

TG3 is conditional stable for ($C^2 < 1$), reaching a third order accuracy and being explicit method.



On the other side we find the scheme TG3-2S, that basically present improvements compared with TG3. The benefits of the multistage for the third order Taylor Galerkin are easier implementation for nonlinear hyperbolic equations as well as wider range of stability on multidimensional case. In our case, the stability in 1D is ($C^2 < 3/4$).

The implementation of the TG3-2S was done in a specific function, where it was introduced two new variable Y, Z defining the left and right side of the equation for the step u_{n+1} .

```matlab
function [A,B,Y,Z,methodName] = System_7(method,M,K,C,a,dt)

switch method

    case 7 % Taylor-Garlerkin TG3-2S
        Alpha = 1/9;
        % For step 1
        A = M;
        B = -a*(1/3)*dt*C -Alpha*a^2*dt^2*K;
        % For step 2
        Y =  -dt*a*C; % Times u(n)
        Z = -0.5*dt^2*a^2*K; % Times u_hat(n)

        methodName = 'TG3-2S';

    otherwise
        error('not available method')
end
```

Meanwhile, the implementation of the solution had to be modified in order to adapt the two steps for the TG3-2S scheme.

```matlab
% Diego Implementation method 7
if method == 7
    Y = Y(ind_unk,ind_unk); % Times u(n)
    Z = Z(ind_unk,ind_unk); % Times u_hat
    u_hat = zeros(nPt,nStep-1); % Improving the velocty
    for n = 1:nStep
        % Computing step 1
        Du_hat = A\(B*u(ind_unk,n) + f);
        u_hat(ind_unk,n) = u(ind_unk,n) + Du_hat;
        % Computing step 2
        Du = A\(Y*u(ind_unk,n) + Z*u_hat(ind_unk,n) + f);
        u(ind_unk,n+1) = u(ind_unk,n) + Du;
    end
end
```