FINITE ELEMENTS IN FLUIDS

Assignment 2: Unsteady Convective Transport

1. Leap-Frog method implementation

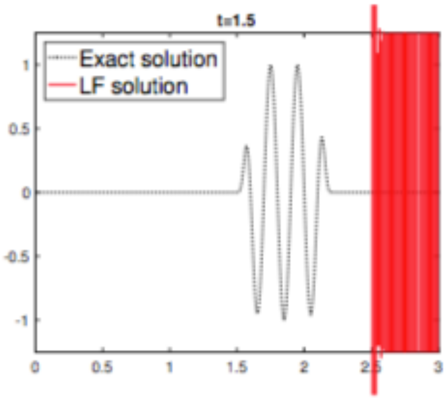
After applying weight functions and integrating by parts we get that
A = M and B = -2*a*dt*C.
We can now integrate the method in the code like:

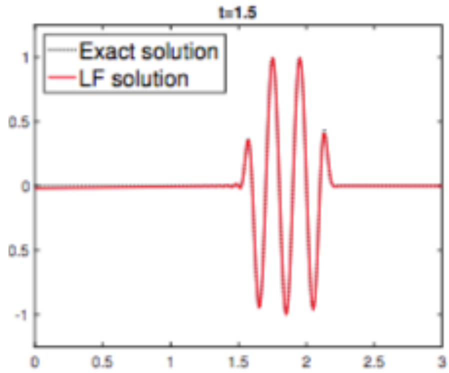
```
case 5 % Leap-Frog method
    A = M;
    B = -2*a*dt*C;
    methodName = 'LF';
```

We need to modify few things for implementing LF in main.m:

```
if method == 5
    for n = 1:nStep
        % Primer incialitzem amb Lax Wendroff
        if n == 1
            [A,B,methodName] = System(1,M,K,C,a,dt);
            DELTA_U = A\(B*u(1:nPt,n);
            u(1:nPt,n+1) = u(1:nPt,n) + DELTA_U;
            clear A,B;
        else % A partir del segon step ja es fa Leap-Frog
            A,B,methodName] = System(5,M,K,C,a,dt);
            DELTA_U = A\(B*u(1:nPt,n)); %afegim el delta_u
            %afegint U(n-1) ens torna al step que volem de LF
            u(1:nPt,n+1) = u(1:nPt,n-1) + DELTA_U;
        end
    end
```

As we can see in the solution, a large Courant number (2) will ahve very low accuracy. But if we take C = 0.0125 LF fits the solution almost perfectly.



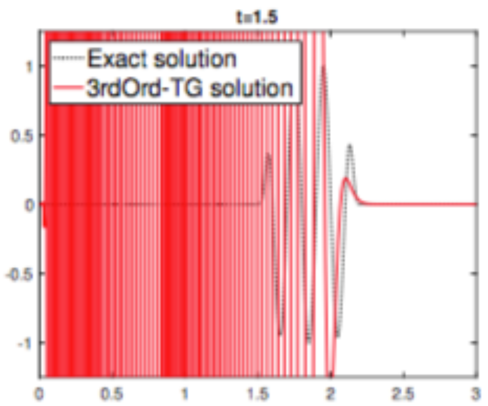


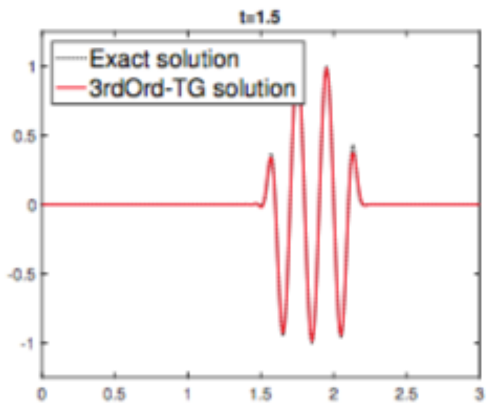
2. Implementation of Taylor Galerkin 3rd order

In this case we are going to implement TG3. For this we will need to add the following matrices:

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

We won't need to modify anything in main.m to run TG3. The results we get are the following with variating the Courant Number:





As we can see, for a Courant number = 2 we get a very unstable solution, however, using 0.0125 we aget a very accurate solution close to the exact one.

3. Implementation of TG3 2 steps

For this implementation, we will need to add the following code in System.m, both steps separetly:

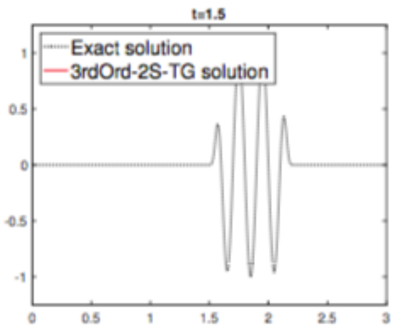
```
case 7 % TG3 2steps 1st step
    A = M;
    B = -(1/3)*a*dt*C - (1/9)*dt^2*a^2*K);
    methodName = '1st step';
case 8 % TG3 2steps 2nd step
    A = M;
    B = -a*dt*C - (1/2)*a^2*dt^2;
    methodName = '2nd step';
```

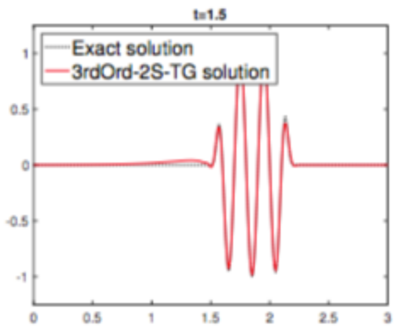
In this case we will also need to modify the code for main.m such as:

```
if method == 7
for n = 1:nStep
    % Primer incialitzem amb el primer pas de TG3 2 steps
    if n == 1
        [A,B,methodName] = System(7,M,K,C,a,dt);
        DELTA_U = A\(B*u(1:nPt,n));
        UBAR =     u(1:nPt,n) + DELTA_U;
        clear A,B;
    %Ara implementem el segon pas amb un fals 8 mètode ficat a System

       [A,B,methodName] = System(8,M,K,C,a,dt);
        DELTA_U = A\(B*u(1:nPt,n) - (1/2)*a^2*dt^2*K*UBAR); %afegim el delta_u
        %Ara ja podem buscar el valor a la iteració u(n+1)
        u(1:nPt,n+1) = u(1:nPt,n) + DELTA_U;
    end
end
```

Now that we have already implemented this in our code, we can plot the results and their behaviour:





As we can see, the same happens as in TG3, with Courant number = 2, on the left, the solution doesn't even appear on the plot, in the right case, Courant number = 0.125 we canse e that the solution is pretty accurate and close to the exact.