Universitat Politècnica de Catalunya
Master of Science in Computational Mechanics
Finite Element in Fluids
FEF
Spring Semester 2017/2018

MatLab Session 2
Usteady Advection Equation

Luan Malikoski Vieira

February 26, 2018

1 Introduction

This report will cover the 1D *usteady state advection* equation, seen in (1), solution using integration in in time domain by means of cartesian coordinates (not by means of charactheristcs lines) followed by a spatial domain integration via *FEM* Galerkin formulation. Three time domain integration methods will be implemented as follows: Leap-Frog (LF), Third-order Taylor-Galerkin (TG3) and Two-steps Third-order Taylor-Galerkin (TG3-2S). Change in code to enable the use of 1D quadratic elements will be shown. Next section will briefly explain the Matlab code implementation for each method. Following, some results will be discussed.

$$u_t + au_x = 0 \tag{1}$$

1.1 Implementation for 1D Quadratic elements

In order to enable the use of quadratic elements in the formulation of the problem some small changes were made in the code provided. In the MatLab file named as *main.m* the definition of the connectivity matrix **T** and number of points nPt (based on number of elements input) for a mesh with 3-noded elements was implemented using an *if* logic operator as shown.

```
if p==1
nPt = nElem + 1;
h = (dom(2) - dom(1))/nElem;
X = (dom(1):h:dom(2))';
T = [1:nPt-1; 2:nPt]';
elseif p ==2
nPt = 2*nElem + 1;
h = (dom(2) - dom(1))/(2*nElem);
X = (dom(1):h:dom(2))';
```

```
T = [1:2:nPt-2; 2:2:nPt-1; 3:2:nPt]';
end
```

This enabled the use of quadratic elements ($p = 2$ in the code).

1.2 Implementation of Leap-Frog (LF)

The Galerkin weak formulation of the LF method for equation 1, written in compact and scalar form without Neumann boundary conditions, reads:

$$(w, \Delta u) = -2a\Delta t(w, u_x^n) \tag{2}$$

Where $\Delta u = u^{n+1} - u^{n-1}$. After spatial discretization the general system of equations, which will also be used for other methods is written as:

$$\mathbf{A\Delta u} = \mathbf{Bu^n} \tag{3}$$

Thus, for the implementation of this method, the matrices A and B in the Matlab code file named as *System.m* were defined as follows.

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

As the LF method uses a centered approximation for the first derivative in time, u^{n-1} is needed, which is not available in the very first step (u^n is the initial condition). Thus, a different integration method needs to be employed for this first integration step. Here, the LW (Law-Wendroff) were employed, due to its lower relative phase error for high values of the dimensionless wave number (ξ) when compared with others 2^{nd} order methods. Thus, the loop integration in the Matlab code file named as *main.m* had the following part add (using *if* logical operator).

```
if method == 5
% FIRST STEP INTEGRATION LW
[A1,B1] = System(1,M,K,C,a,dt);
A1 = M;
B1 = -a*dt*C- 0.5*a^2*dt^2*K;
A1 = A1(ind_unk,ind_unk);
B1 = B1(ind_unk,ind_unk);
Du1 = A1\(B1*u(ind_unk,1) + f);
u(ind_unk,2) = u(ind_unk,1) + Du1;
% LEAP-FROG
for n = 2:nStep
Du = A\(B*u(ind_unk,n) + f);
u(ind_unk,n+1) = u(ind_unk,n-1) + Du;
end
```

1.3 Implementation of Third-order Taylor-Galerkin (TG3)

The Galerkin weak formulation of the TG3 method for equation 1, written in compact and scalar form without Neumann boundary conditions, reads:

$$(w, \Delta u) + \frac{a^2\Delta t^2}{6}(w_x, \Delta u_x) = -a\Delta t(w, u_x^n) - \frac{a^2\Delta t^2}{2}(w_x, u_x^n) \tag{4}$$

Where $\Delta u = u^{n+1} - u^n$. Thus, for the implementation of this method, the matrices A and B in the Matlab code file named as *System.m* were defined as follows.

```
case 6 % Third order Taylor-Galerkin + Galerkin
A = M + 1/6*a^2*dt^2*K;
B = -a*dt*C-1/2*a^2*dt^2*K;
methodName = 'TG3';
```

1.4 Implementation of Two-steps Third-order Taylor-Galerkin (TG3-2S)

The Galerkin weak formulation of the TG3-2S method for equation 1, written in compact and scalar form without Neumann boundary conditions, is composed of two equations (each step) as follows:

$$(w, \tilde{u}^n) = (w, u^n) - \frac{a\Delta t}{3}(w, u_x^n) - \alpha a^2 \Delta t^2 \, (w_x, u_x^n) \tag{5}$$

$$(w, \Delta u) = -a\Delta t(w, u_x^n) - \frac{a^2 \Delta t^2}{2} \, (w_x, \tilde{u}_x^n) \tag{6}$$

Where $\Delta u = u^{n+1} - u^n$. Also, $\tilde{u}$ represents the solution for a second order Taylor expansion when the step size $\tilde{\Delta t} = \frac{\Delta t}{3}$, which is the case when $\alpha = 1/9$ in equation 5, as will be assumed in this work.
Thus, for the implementation of this method, the matrices A and B in the Matlab code file named as *System.m* were defined as follows.

```
case 7 % Third order Taylor-Galerkin + Galerkin (2 steps)
A = M ;
B = -a*dt*C;
methodName = 'TG3-2S';
```

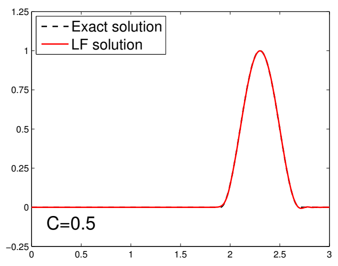
As this is a two steps method, the integration loop also had to be modified (as did for LF method). Thus, the loop integration in the Matlab code file named as *main.m* had the following part add (using *elseif* logical operator).

```
elseif method == 7
alpha = 1/9;
u1 = zeros(nPt,nStep+1);
M = M(ind_unk,ind_unk);
K = K(ind_unk,ind_unk);
C = C(ind_unk,ind_unk);
for n = 1:nStep
%FIRST STEP
u1(ind_unk,n+1) = M\((M-a/3*dt*C-alpha*a^2*dt^2*K)*u(ind_unk,n));
D = -a^2/2*dt^2*K;
%SECOND STEP
Du = A\(B*u(ind_unk,n)+D*u1(ind_unk,n+1)+ f );
u(ind_unk,n+1) = u(ind_unk,n) + Du;
end
```

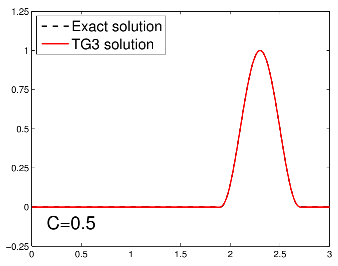
2 Results

2.1 Problem I: LF, TG3 and TG3-2S algorithm test

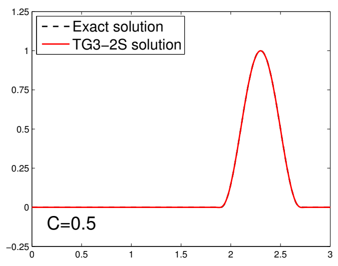
The first example were run using Galerkin Method combined with the three time discretization methods in order to check stability limit. Three different Courant numbers (C) were tested, $C = 0.5, C = 0.6$ and $C = 0.9$. Figure (1) shows the result for $C = 0.5$.



(a)



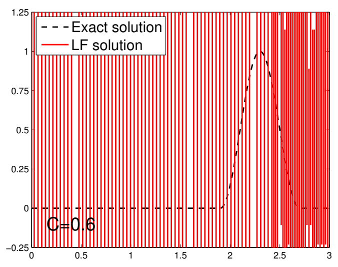
(b)



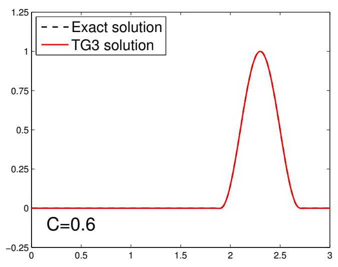
(c)

Figure 1: Response for C = 0.5 (Problem 1).

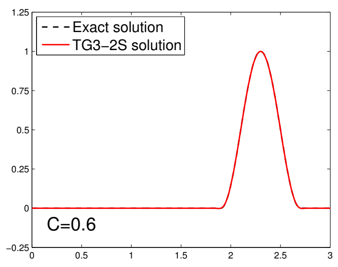
For $C = 0.6$, the stability limit for the LF method ($C \approx 0.57$) is surpassed, as it can be seen in Figure (2).



(a)



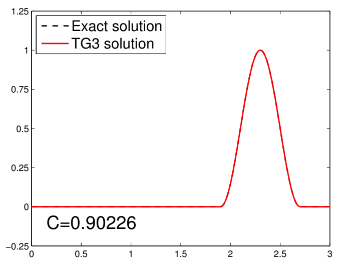
(b)



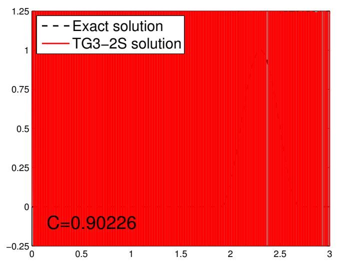
(c)

Figure 2: Response for C = 0.6 (Problem 1).

For $C = 0.9$, the stability limit for the TG3-2S method ($C \approx 0.87$) is surpassed, as it can be seen in Figure (3).



(a)

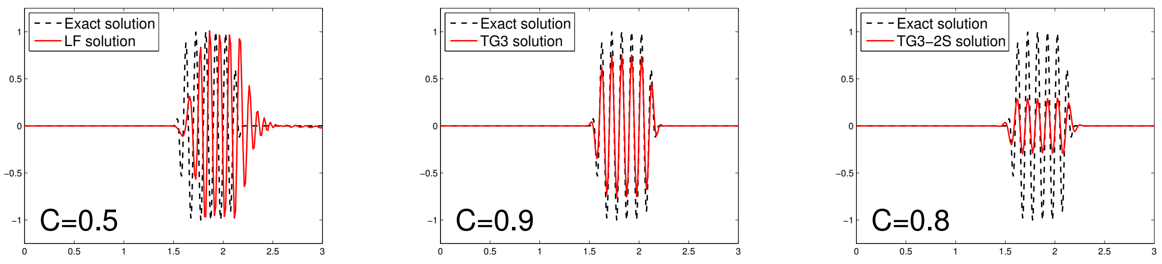


(b)

Figure 3: Response for C = 0.9 (Problem 1).

2.2 Problem III: LF, TG3 and TG3-2S phase error analysis

To evaluate phase response, the problem two were run for each method at 90% ($C = 0.52$,$C = 0.9$ and $C = 0.78$ for LF, TG3 and TG3-2S respectively).of the stability limit, and a non-dimensional wave number ($\xi = \pi/4$), which is considered a limit for accuracy. As it can be seen in Figure (4), LF method as a worst behavior in terms of phase error near of stability limit, as expected for 2^{nd} order methods, which has the phase deteriorated as δt increases. The two steps Taylor-Galerkin (TG3-2S) has a worst performance, in terms of amplitude than its one step counterpart (TG3) near each stability limit, as it can be seen, however, as expected both methods has similar performance when it comes to the relative phase error.

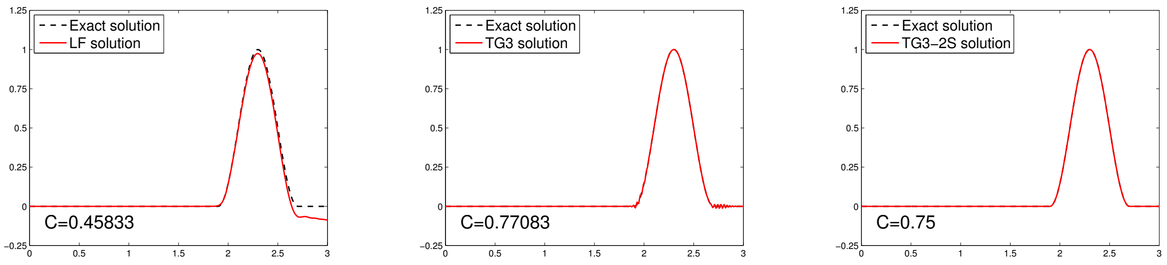


(a) (b) (c)

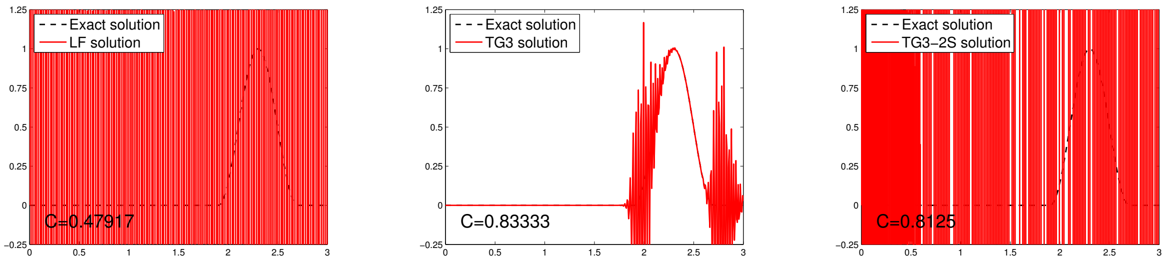
Figure 4: Response for Courant at 90% of stability limit (Problem 3).

2.3 Problem I: LF, TG3 and TG3-2S algorithm test for 1D quadratic elements

Problem I were run again for all three methods this time using 1D quadratic elements. The general behavior was a reduction in the stability limit for all three methods, as it can be seen below.



(a)LF Stable (b) TG3 Stable (c) TG3-2S stable



(a)LF unstable (b) TG3 unstable (c) TG3-2S unstable

Figure 5: Response with 1D quadratic elements.

As it can be seen, now the stability limit lies around $C = 0.46$,$C = 0.80$ and $C = 0.78$ for LF, TG3 and TG3-2S respectively. Thus, TG3 and LF had their stability limit reduced in a major extent when compared with LG3-2S.