FINITE ELEMENTS IN FLUIDS

ASSIGNMENT2- 1D UNSTEADY TRANSPORT PROBLEM

-By Anurag Bhattacharjee

LEAP FROG METHOD

The galerkin formulation for the leap-frog method is presented below-

$$\frac{u^{n+1} - u^{n-1}}{2\Delta t} = u_t^n \quad , \quad u_t = -a \cdot \nabla u$$

$$\Rightarrow \quad \frac{u^{n+1} - u^{n-1}}{2\Delta t} = (-a \cdot \nabla u^n)$$

$$\Rightarrow \quad u^{n+1} - u^{n-1} = -2a\Delta t \, \nabla u^n$$

$\Rightarrow$ Using galerkin formulation, we have

$$(\omega, u^{n+1}) - (\omega, u^{n-1}) = -2a\Delta t(\omega, \nabla u^n)$$

$$\Rightarrow \left[\omega, (u^{n+1} - u^{n-1})\right] = -2a\Delta t\,(\omega, \nabla u^n)$$

Changes in the code (System.m) made to accommodate this code is given below-

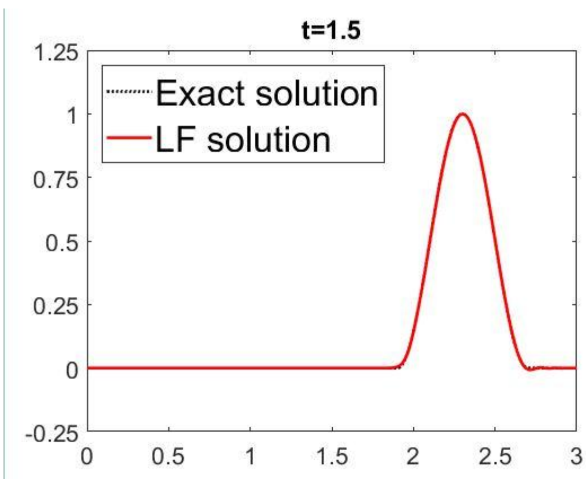
```
        case 5 % Leap-Frog
            A = M;
            B = -2*a*dt*C;
            methodName = 'LF';
```

The following changes were made to the main.m

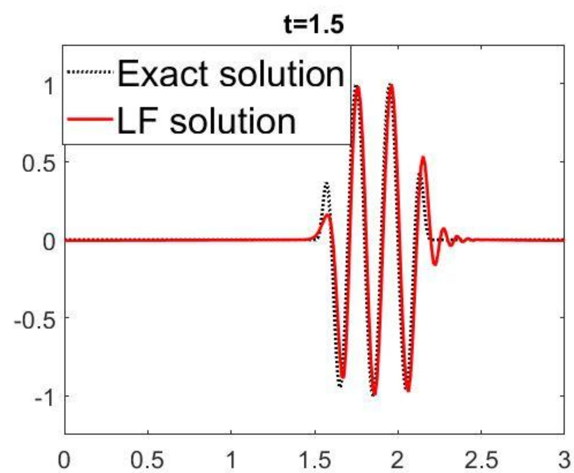
```
if method==5
    for n= 1:nStep
     if n==1
    [A,B,methodName]= System(1,M,K,C,a,dt);
    Du = A\(B*u(1:nPt,n) + f);
     u(1:nPt,n+1) = u(1:nPt,n) + Du;
     clear A,B;
     else
        [A,B,methodName]= System(5,M,K,C,a,dt);
    Du = A\(B*u(1:nPt,n) + f);
     u(1:nPt,n+1) = u(1:nPt,n-1) + Du;
     end
    end
```

It should be noted that as for n=1, u(n-1) doesn't exist, Leap frog formulation is ineffective. So we need to implement another method to initialize the Leap-Frog then from n=2, Leap frog takes over further computation. Here I have used Lax-Wendroff + Galerkin formulation to initialize the Leap-Frog formulation.

Results obtained-

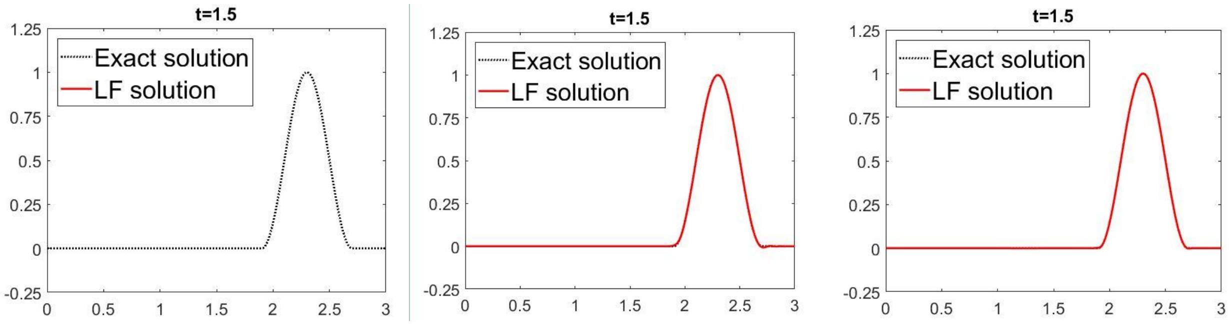


Problem-1



Problem-2

The simulation has been done for 2 problems C=0.5 and the code seems to be working. To check the stability of the method the following results were obtained by varying the courant number for problem 1 which seems to be consistent with the theory. Leap frog method is stable for courant number C^2<=(3/4), which is consistent with the results. It is unstable for C=1 but stable for values less than 0.8.



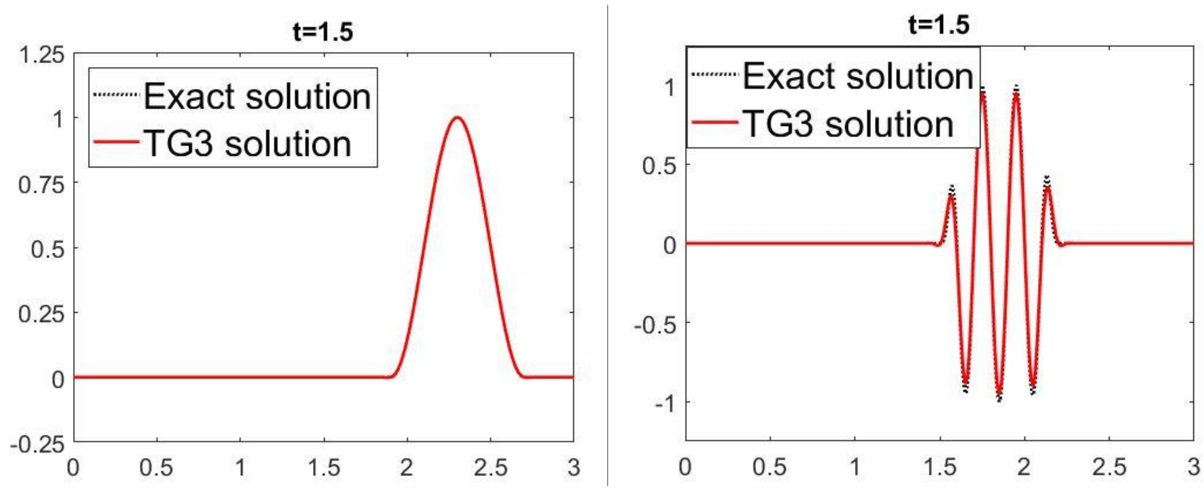
C=1 C=0.5 C=0.25

THIRD ORDER TAYLOR-GALERKIN

As per the theory given in the slides, the following changes were made to the system.m file-

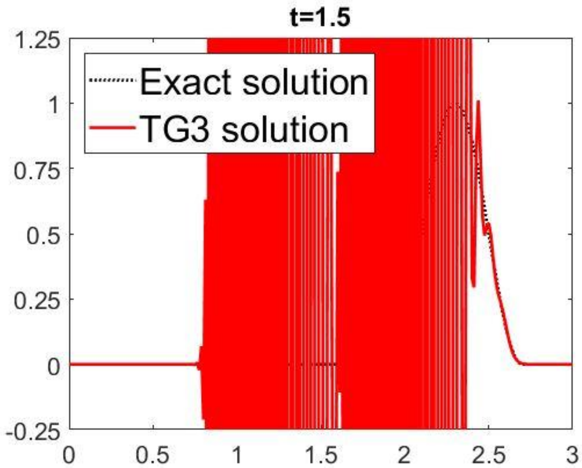
```
        case 6 % TG3
            A = M + 0.16667*dt^2*a^2*K;
            B = -a*dt*C- 0.5*a^2*dt^2*K;
            methodName = 'TG3';
```

Results obtained-

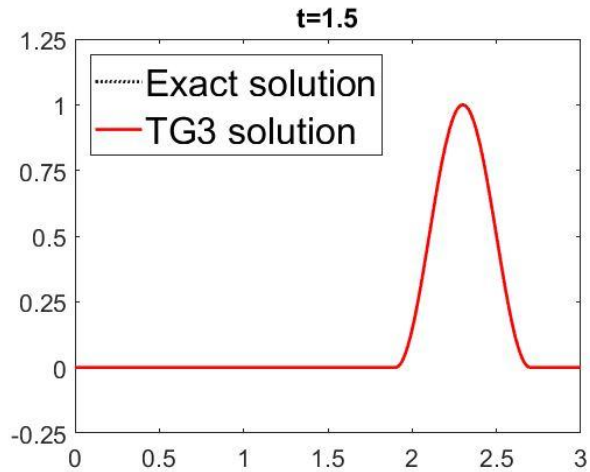


Problem-1 Problem-2

The simulation has been done for 2 problems C=0.5 and the code seems to be working. To check the stability of the method the following results were obtained by varying the courant number for problem 1 which seems to be consistent with the theory. TG3 method is stable for courant number C^2<=1, which is consistent with the results. It is unstable for C>1 but stable for values less than equal to 1.



C=1.33



C=1

THIRD ORDER TAYLOR-GALERKIN 2-STEP

The galerkin formulation for the TG3-2S method is presented below-

First step

$$\tilde{u}^n = u^n + \frac{1}{3}\Delta t\, u_t^n + \alpha\, \Delta t^2 u_{tt}^n$$

$$\Rightarrow \tilde{u}^n = u^n + \frac{1}{3}\Delta t\, u_t^n + \frac{1}{9}\Delta t^2 u_{tt}^n \quad [\alpha = 1/9]$$

$$\Rightarrow \tilde{u}^n = u^n + \frac{1}{3}\Delta t(-a.\nabla u^n) + \frac{1}{9}\Delta t^2 (-a.\nabla)^2 u^n$$

$$\Rightarrow \tilde{u}^n = u^n + \left(-\frac{a\Delta t}{3}\right)\nabla u^n + \frac{\Delta t^2}{9} a^2 \nabla^2 u^n$$

Using Galerkin formulation,

$$(\omega, \tilde{u}^n) = (\omega, u^n) - \frac{\Delta t}{3} a(\omega, \nabla u^n) + \frac{a^2\Delta t^2}{9}(\omega, \nabla^2 u^n)$$

$$[\omega, \nabla^2 u^n]_\Omega = \cancelto{0}{[\omega, \nabla u^n]_\Gamma} - [\nabla\omega, \nabla u^n]_\Omega \quad \text{[Boundary flux in this case is neglected]}$$

$$\therefore (\omega, \tilde{u}^n) = (\omega, u^n) - \frac{a\Delta t}{3}(\omega, \nabla u^n) - \frac{a^2\Delta t^2}{9}(\nabla\omega, \nabla u^n)$$

From here we can get the value for U^n(bar) and substitute it in the second step of formulation which is presented below-

This can be clearly seen from the two step code in (main.m) given below-

<u>Second step</u>

$$u^{n+1} = u^n + \Delta t\, u_t^n + \tfrac{1}{2}\Delta t^2\, \tilde{u}_{tt}^n$$

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

$$\Rightarrow u^{n+1} = u^n - a\Delta t\, \nabla u^n + \frac{\Delta t^2 a^2}{2}\nabla^2 \tilde{u}^n$$

Using galerkin formulation,

$$(\omega, u^{n+1}) = (\omega, u^n) - a\Delta t(\omega, \nabla u^n) + \frac{a^2\Delta t^2}{2}(\omega, \nabla^2 \tilde{u}^n)$$

$$\left[\omega, \nabla^2 \tilde{u}^n\right]_\Omega = \left[\omega, \cancelto{0}{\nabla \tilde{u}^n}\right]_\Gamma - \left[\nabla\omega, \nabla\tilde{u}^n\right]$$

$$\Rightarrow \left[\omega, \underbrace{(u^{n+1} - u^n)}_{\Delta u}\right] = -a\Delta t(\omega, \nabla u^n) + \frac{a^2\Delta t^2}{2}\left(-\nabla\omega, \nabla\tilde{u}^n\right)$$

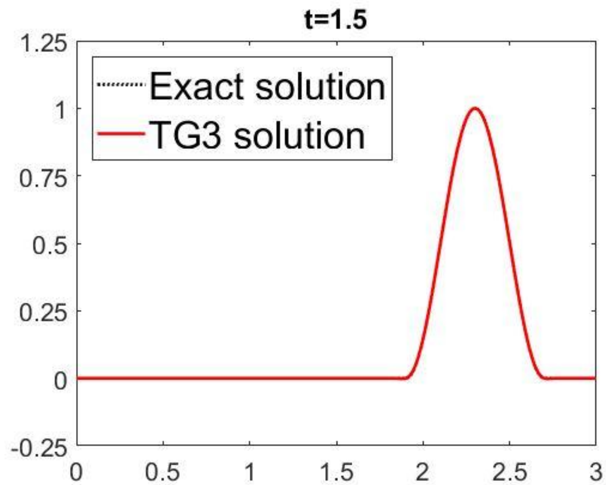
Changes in the code (System.m) made to accommodate this code is given below- TG3-I and TG3-II represent the two steps-

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

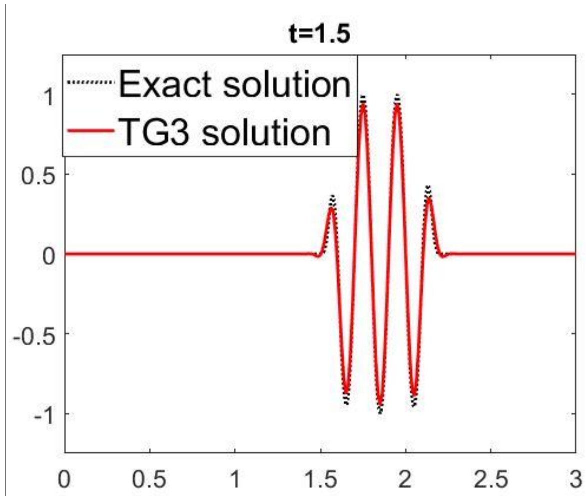
The following changes were made to the main.m

```
    else if method==7
            for n= 1:nStep
            [A,B,methodName]= System(7,M,K,C,a,dt);
            Du = A\(B*u(1:nPt,n) + f);
              u_bar= u(1:nPt,n) + Du;
            clear A,B;
            [A,B,methodName]= System(8,M,K,C,a,dt);
            Du = A\(B*u(1:nPt,n)- .5*a^2*dt^2*K*u_bar + f);
            u(1:nPt,n+1) = u(1:nPt,n) + Du;
            end
        else
```

Results obtained-

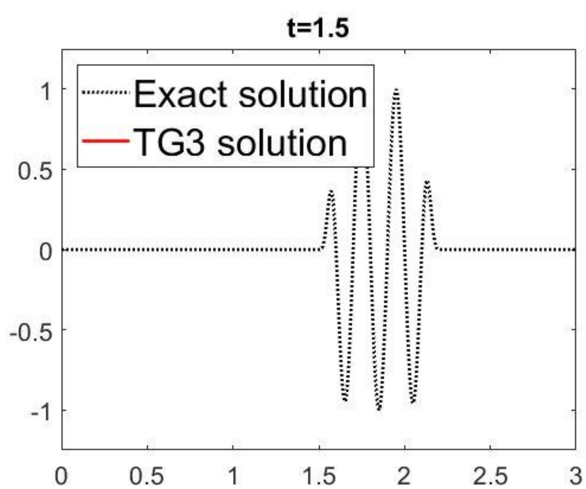


Problem-1

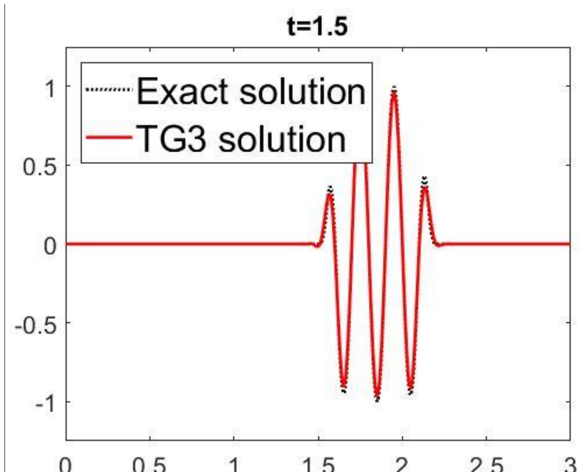


Problem-2

The simulation has been done for 2 problems for C=0.5 and the code seems to be working. To check the stability of the method the following results were obtained by varying the courant number for problem 2 which seems to be consistent with the theory. TG3-2S method is stable for courant number $C^2 <= 1$, which is consistent with the results. It is unstable for C>1 but stable for values less than equal to 1.



C=1.33



C=1

For one dimensional computations both TG3 and TG3-2S are expected to give similarresults. However because of the higher dissipative nature of TG3-2S, it is expected to perform better for two-dimensional computations.