POLYTECHNIC UNIVERSITY OF CATALONIA

MASTER OF SCIENCE IN COMPUTATIONAL MECHANICS

Finite Element in Fluids
2D convection-diffusion equation
Assignment

Student: Chiluba Isaiah Nsofu

Lecture : Prof. Pablo Sáez

Date: 2nd April, 2018

Contents

1 2D convection-diffusion equation

1.1 The Galerkin Least Square (GLS) implementation

The Galerkin Least Square (GLS) method was implemented into the Matlab code as shown in the figures (1.1) and (1.2) below. This implementation involves the introduction of the 2D derivatives for the mapping of the 2nd order derivatives. This mapping are a prerequisite for the GLS method. Implementation of these derivatives was quiet a task for this part of the assignment. The results obtained using this method are presented in the sections to follows together with other formulations

```
for ig = 1:ngaus
    N_ig = N(ig,:);  Nxi_ig = Nxi(ig,:);   Neta_ig = Neta(ig,:);
    N2xi_ig = N2xi(ig,:);  N2eta_ig = N2eta(ig,:);
    N2xieta_ig = N2xieta(ig,:);

    dxdxi  = Nxi_ig*(Xe(:,1)); dydxi  = Nxi_ig*(Xe(:,2));
    dxdeta = Neta_ig*(Xe(:,1)); dydeta = Neta_ig*(Xe(:,2));
    d2xdxi2   = N2xi_ig*(Xe(:,1)); d2xdeta2  = N2eta_ig*(Xe(:,1));
    d2ydxi2   = N2xi_ig*(Xe(:,2)); d2ydeta2  = N2eta_ig*(Xe(:,2));
    d2xdxieta = N2xieta_ig*(Xe(:,1)); d2ydxieta = N2xieta_ig*(Xe(:,2));

    % mapping of 1st order derivarives
    Jacob = [dxdxi  dydxi; dxdeta   dydeta]; dvolu = wgp(ig)*det(Jacob);
    res = Jacob\[Nxi_ig;Neta_ig];
    Nx = res(1,:);  Ny = res(2,:);
    % mapping of 2nd order derivatives
    mappingMatrix =  [dxdxi^2        dydxi^2        2*dxdxi*dydxi;
                      dxdeta^2       dydeta^2       2*dxdeta*dydeta;
                      dxdxi*dxdeta   dydxi*dydeta   dxdxi*dydeta+dxdeta*dydxi];
    referenceVector = [N2xi_ig - Nx*d2xdxi2 - Ny*d2ydxi2 ;
                       N2eta_ig - Nx*d2xdeta2 - Ny*d2ydeta2;
                       N2xieta_ig - Nx*d2xdxieta - Ny*d2ydxieta];
    physicalVector = mappingMatrix\referenceVector;
    N2x = physicalVector(1,:);
    N2y = physicalVector(2,:);
    Nxy = physicalVector(3,:);
```

Figure 1.1: Implementation of both the 1st and 2nd order derivatives required for the 2D case

```
elseif method == 3
    % GLS
    Ke = Ke + (N_ig'*(ax*Nx+ay*Ny) + nu*(Nx'*Nx+Ny'*Ny) + N_ig'*sigma*N_ig + ...
        tau*((ax*Nx+ay*Ny) - nu*(N2x+N2y) + sigma*N_ig)'*...
        ((ax*Nx+ay*Ny) - nu*(N2x+N2y) + sigma*N_ig))*dvolu;
    aux = N_ig*Xe;
    f_ig = SourceTerm(aux);
    fe = fe + (N_ig + tau*((ax*Nx+ay*Ny) - nu*(N2x+N2y) + sigma*N_ig ))'*(f_ig*dvolu);
end
```

Figure 1.2: Implementation of both the 1st and 2nd order derivatives required for the 2D case

Galerkin, Artificial diffusion, SUPG and GLS

A convection-reaction dominated case with **Neumann boundary conditions** at the outlet

$$\|a\| = 1 \ , \ \nu = 0.0004$$

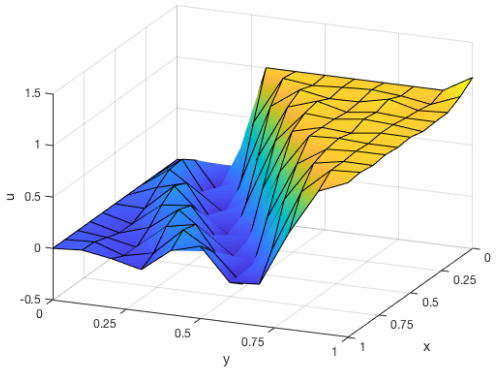

Figure 1.3: Galerkin Neumann- BC

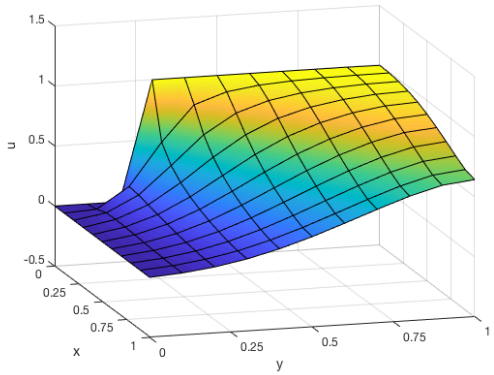

Figure 1.4: Artificial-Neumann

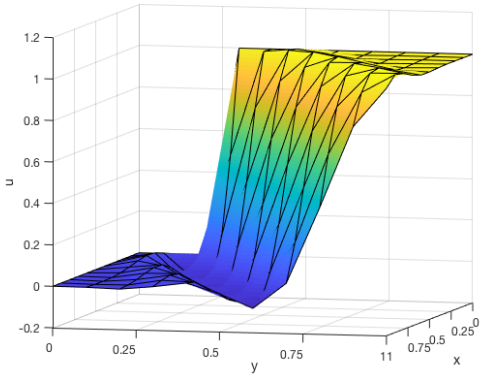

Figure 1.5: SUPG with N- BC, Linear Elements

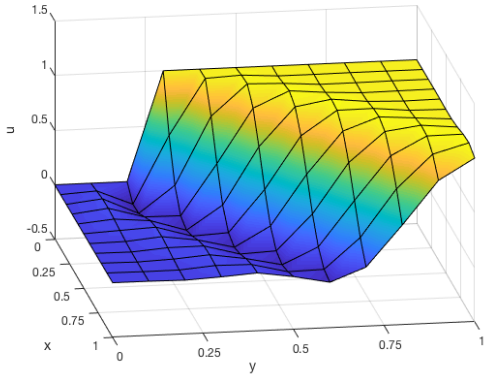

Figure 1.6: GLS N-BC, Quad Elements

The results for the case where Neumann Boundary conditions are applied on the outlet are as shown in figures (1.3) (1.4) (1.5) (1.6). As it can be seen in (1.3) the Galerkin is not able to give very good results since its not able to resolve the discontinuity and hence it produces spurious oscillations. The artificial diffusion , SUPG and GLS methods produces better results though both SUPG and GLS introduces less crosswind diffusion

A convection-reaction dominated case with **Dirichlet boundary conditions** at the outlet

$$\|a\| = 1 \ , \ \nu = 0.0004$$

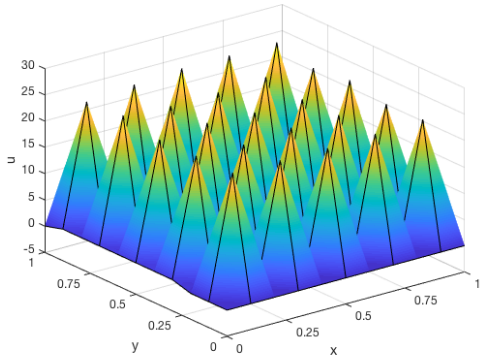

Figure 1.7: Galerkin with D- BC, Linear Elements

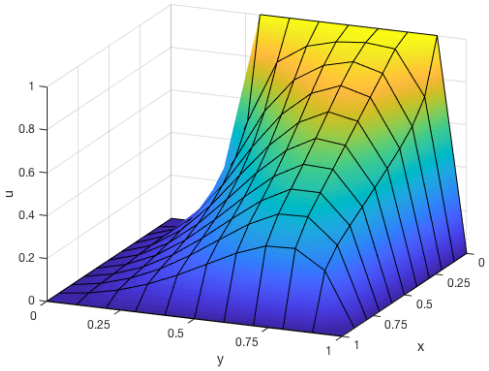

Figure 1.8: Artificial-Dif D-BC, Quad Elements

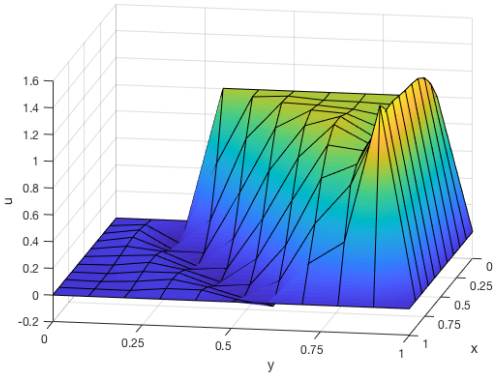

Figure 1.9: SUPG with D- BC, Linear Elements

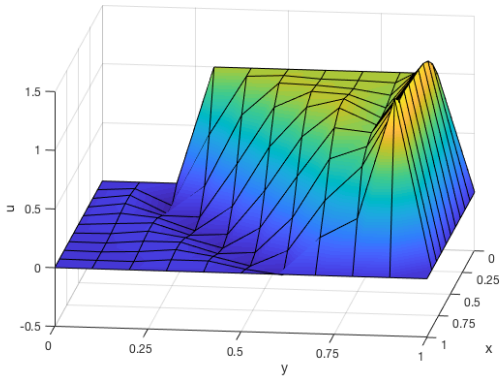

Figure 1.10: GLS D-BC, Quad Elements

Figures (1.7) (1.8) (1.9) (1.10) represents the results once the the Dirichlet BC also applied at the outlet boundary. This can easily be verified by the presence of a thin boundary layer at the outlet. The results for the Garlekin method (1.7) shows that this method produces results with alot of oscillations and hence it does not represent the exact solution. Both SUPG and GLS produces better results which shows that the stabilization terms are having an impact of the results. The results produced by the artificial diffusion method (1.8) shows that the method introduces too much diffusion

A convection-reaction dominated case with **Dirichlet boundary conditions** at the outlet

$$\|a\| = 1/2 \ , \ \nu = 0.0004 \ \textit{and} \ \sigma = 1$$

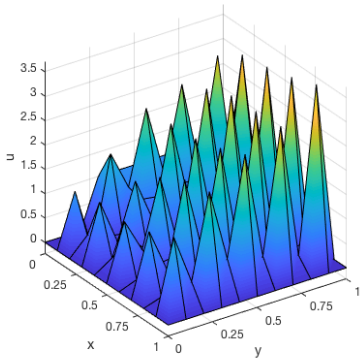

Figure 1.11: Galerkin with D- BC, Linear Elements

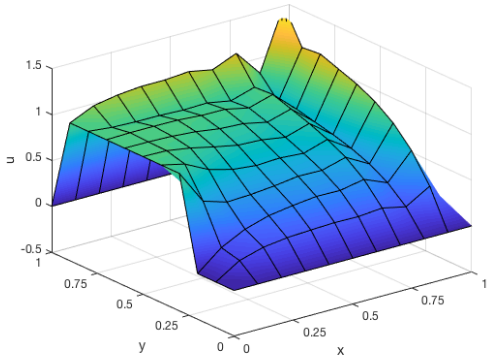

Figure 1.12: Artificial diffusion D-BC, Linear Elements

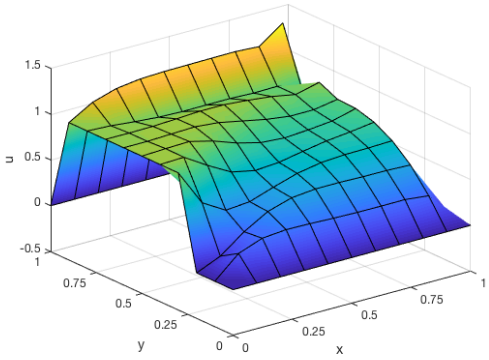

Figure 1.13: SUPG with D-BC, Linear Elements

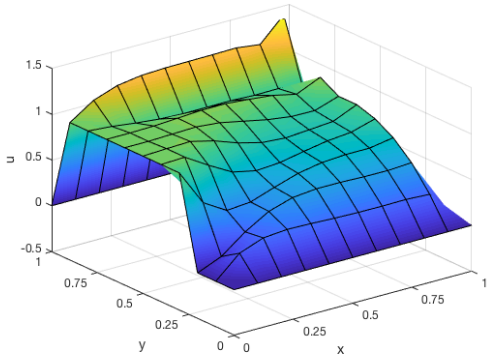

Figure 1.14: GLS with D-BC, Linear Elements

Introducing the reaction term into the convection diffusion equation brings about a change in the results. This change can be seen by the results presented in figures (1.11) , (1.12), (1.13) and (1.14). It should be noted that the Dirichlet BC is used at the outlet boundary. Therefore, comparing the results in these figures with those in figures (1.7) (1.8) (1.9) (1.9) we can see a very big difference. To elaborate more on the new results we see that the results produced by the Galerkin method in (1.11) are not accurate as compared with the results from the stabilized methods in (1.12), (1.13) and (1.14). Note that the results produced by the artificial diffusion formulation is slightly different from that of (1.13) and (1.14).

A convection-reaction dominated case with **Dirichlet boundary conditions** at the outlet

$$\|a\| = 0.001 \ , \ \nu = 0.0004 \ \textit{and} \ \sigma = 1$$

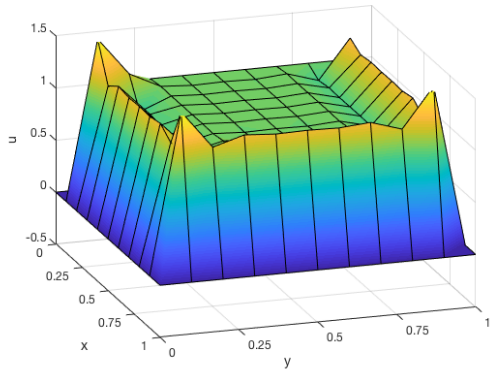

Figure 1.15: Galerkin- D-BC, Lin Elements

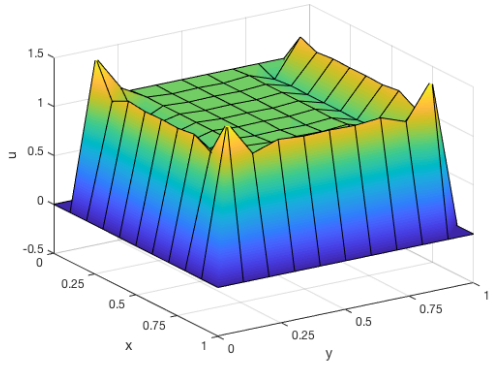

Figure 1.16: Atificial diff- D-BC, Lin Elements

A convection-reaction dominated case with **Dirichlet boundary conditions** at the outlet

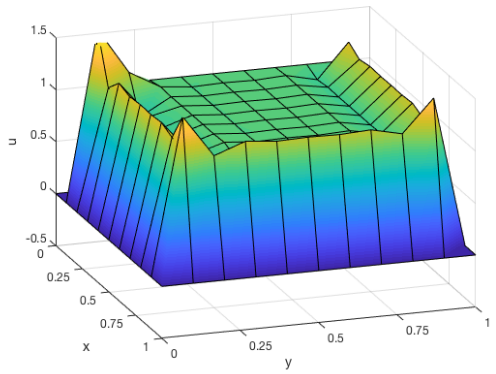

Figure 1.17: SUPG - D- BC, Linear Elements

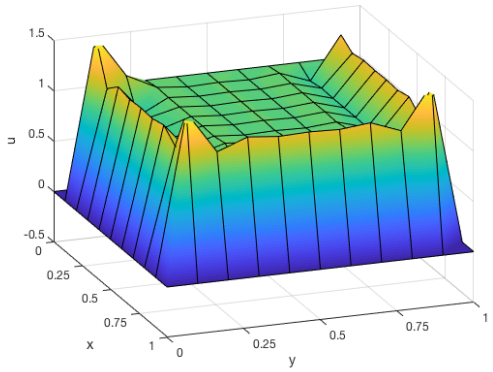

Figure 1.18: GLS D- BC, Linear Elements

Figures (1.15) , (1.16) ,(1.17) and (1.18) show the results of the convection diffusion reaction problem with a change in the value of the velocity. From these results we can see that the results produced by all the formulations ae the very similar. However we can notice that there thin boundary layer that has been added at the ends of the boundaries. Hence we can see there is a very big change in these results in comparison with the results produced in (1.11) , (1.12), (1.13) and (1.14). This change is caused by the new velocity value.

2 Rotating Cone problem

2D homogeneous convection equation with initial condition

2.1 Lax-Wendroff + Galerkin

In order to apply the Lax-Wendroff + Galerkin method to this equation we start with the Taylor series expansion.

$$u(t^{n+1}) = u(t^n) + \Delta t u(t^n) + \frac{1}{2}\Delta t^2 u_{tt}(t^n)... \tag{1}$$

Using

$$u_t + \mathbf{a} \cdot \nabla u = s \tag{2}$$

hence

$$u_t^n = s_n - \mathbf{a} \cdot \nabla u_n \tag{3}$$
$$u_{tt}^n = s_t^n - \mathbf{a} \cdot \nabla s^n + (\mathbf{a} \cdot \nabla)^2 u^n \tag{4}$$

we know that $(u^{n+1} - u^n)/\Delta t = u_t^n + u_{tt}^n/2$ therefore

$$\frac{\Delta u}{\Delta t} = -\mathbf{a} \cdot \nabla u_n + \frac{\Delta t}{2}(\mathbf{a} \cdot \nabla)^2 u^n + s^n + \frac{\Delta t}{2}(s_t^n - \mathbf{a} \cdot \nabla s^n) \tag{5}$$

Before integrating by the parts for some of the terms in (5) we need to set all the terms containing the **source term to be zero** as the equation we are dealing with has no source term. Therefore the equation reduces to:

$$\frac{\Delta u}{\Delta t} = -\mathbf{a} \cdot \nabla u_n + \frac{\Delta t}{2}(\mathbf{a} \cdot \nabla)^2 u^n \tag{6}$$

Introducing the test function we obtain the following variational form;

$$\left(w, \frac{\Delta u}{\Delta t}\right) = -\left(w, \mathbf{a} \cdot \nabla u_n - \frac{\Delta t}{2}(\mathbf{a} \cdot \nabla)^2 u^n\right) \tag{7}$$

After integration by parts we get

$$\left(w, \frac{\Delta u}{\Delta t}\right) = -\left(\mathbf{a} \cdot \nabla w, u^n + \frac{\Delta t}{2}(-\mathbf{a} \cdot \nabla) u^n\right) + \left((\mathbf{a} \cdot \mathbf{n}) w, u^n + \frac{\Delta t}{2}(-\mathbf{a} \cdot \nabla) u^n\right)_{\Gamma^{out}} \tag{8}$$

Comparison with the code
Using the code we will get the following terms M, C and K from th terms that are not in the boundary outlet.

$$M = CreMassMat(X, T, pospg, wpg, N, Nxi, Neta);$$
$$C = CreConvMat(X, T, Conv, pospg, wpg, N, Nxi, Neta);$$
$$K = CreStiffMat(X, T, Conv, pospg, wpg, N, Nxi, Neta);$$

Hence using (8) we see that from the left hand side of the equation

$$M = (w, \Delta u)\,; \;\; where\ \Delta u = u^{n+1} - u^n \tag{9}$$
$$C = (\nabla w, u^n)\,; \tag{10}$$
$$K = (\nabla w, \nabla u^n)\,; \tag{11}$$

From the outlet we can get the following terms using the function in the code as

$$Mo = CreOutMat1(X, T, Conv, elemy0_out, [1, 2]);$$
$$Co = CreOutMat2(X, T, Conv, elemy0_out, [1, 2]);$$
$$vo = CreOutVect(X, T, Conv, elemy0_out, [1, 2]);$$

Therefore using the outlet terms in (8) we get the following

$$Mo = (w, u^n)_{\Gamma^{out}} \tag{12}$$
$$Co = (w, \nabla u^n)_{\Gamma^{out}}\ ; \tag{13}$$

Using all the above equations we can see that equation(8) is written in the code as

$$A = M; \tag{14}$$
$$B = dt * (C - (dt/2) * K - Mo + (dt/2) * Co); \tag{15}$$
$$f = dt * (v1 + (dt/2) * (v2 - vo)); \tag{16}$$

2.2 Lax-Wendroff + Galerkin with lumped mass matrix

The derivation of this fornulation is the same as that of Lax-Wendroff + Galerkin formulation, the only difference in this formaultion is that instead of taking all the terms in the resulting mass matrices we only take the diagonal terms which is of course computationally cheap. Therefore, just as stated in equations (17),(18) and (19) we see that the form of Lax-Wendroff + Galerkin with lumped mass matrix is as follows;

$$A = Md; \tag{17}$$
$$B = dt * (C - (dt/2) * K - Mod + (dt/2) * Co); \tag{18}$$
$$f = dt * (v1 + (dt/2) * (v2 - vo)); \tag{19}$$

2.3 Crank-Nicolson + Galerkin formulation

To develop this formulation the Galerkin formulation for the θ method is used. Using the θ time discretization method we know that the solution of u^{n+1} is given using the formulation below

$$\frac{\Delta u}{\Delta t} - \theta \Delta u_t = u_t^n \tag{20}$$

Introducing the test function w and multiplying it with 20 we have

$$(w, \frac{\Delta u}{\Delta t}) - \theta(w, \Delta u_t) = (w, u_t^n) \tag{21}$$

For the problem at hand we replace u_t with $\mathbf{a}\ \cdot \boldsymbol{\nabla} u$ which lead to the equation below once integration by parts has been performed

$$(w, \frac{\Delta u}{\Delta t}) - \theta(\boldsymbol{\nabla} w, \mathbf{a}\ \Delta u) + \theta\left((\mathbf{a}\ \cdot \boldsymbol{n}) w, \Delta u\right)_{\Gamma^{out}} = (\boldsymbol{\nabla} w, \mathbf{a}\ u_n) - ((\mathbf{a}\ \cdot \boldsymbol{n}) w, u_n)_{\Gamma^{out}} \tag{22}$$

To introduce the Crank-Nicolson formulation we set $\theta = \frac{1}{2}$ in equation 22. This leads to the following formulation

$$\left(w, \frac{\Delta u}{\Delta t}\right) - \frac{1}{2}\left(\boldsymbol{\nabla} w, \mathbf{a}\Delta u\right) + \frac{1}{2}\left((\mathbf{a}\cdot\mathbf{n})w, \Delta u\right)_{\Gamma^{out}} = \left(\boldsymbol{\nabla} w, \mathbf{a}u^n\right) - \left((\mathbf{a}\cdot\mathbf{n})w, u^n\right)_{\Gamma^{out}} \quad (23)$$

Multiplying every term with Δt we get the following

$$(w, \Delta u) - \frac{\Delta t}{2}\left(\boldsymbol{\nabla} w, \mathbf{a}\Delta u\right) + \frac{\Delta t}{2}\left((\mathbf{a}\cdot\mathbf{n})w, \Delta u\right)_{\Gamma^{out}} = \Delta t\left(\boldsymbol{\nabla} w, \mathbf{a}u^n\right) - \Delta t\left((\mathbf{a}\cdot\mathbf{n})w, u^n\right)_{\Gamma^{out}} \quad (24)$$

Code representation of the Crank-Nicolson Formulation
Based on the above equation we can identify the parts that correspond to the Crank-Nicolson + Galerkin formulation in the code.

Firstly considering the **left hand side** (L.H.S) of the above (24) .
1. The first part on the equation we can find the mass matrix as $M = N_i N_j$ from $(w, \Delta u)$ bearing in mind that $\Delta u = u^{n+1} - u^n$
2. From the second term we see that discretization process leads to $C = \boldsymbol{\nabla} w, u^n$
3. The third terms leads to the term $Mo = (w, u)$ on the boundary

Using the above terms and replacing the in (24) we that the term on L.H.S corresponds to the line in the code as follows:

$$A = M - (dt/2) * C + (dt/2) * Mo \quad (25)$$

Secondly considering the **right hand side** (R.H.S) of the above (24) .

1.The first term leads to $C = \boldsymbol{\nabla} w, u^n$
2. The second term leads to $Mo = (w, u)$ on the boundary
Therefore using these terms and replacing the in (24) we that the term on R.H.S corresponds to the line in the code as follows:

$$B = dt/2 * C + -dt/2 * Mo \quad (26)$$

Therefore, using (25) and (26) together with the term representing the velocity on the source term, the Crank-Nicolson formulation is given in the code as

$$A = M - (dt/2) * C + (dt/2) * Mo \quad (27)$$
$$B = dt/2 * C + -dt/2 * Mo \quad (28)$$
$$f = dt * v1 \quad (29)$$

2.4 Crank-Nicolson + Galerkin with lumped mass matrix

The steps leading to the derivation of this formulation are the same as that of Crank-Nicolson + Galerkin formulation, the only difference in this formulation is that instead of taking all the terms in the resulting mass matrices we only take the diagonal terms which is of course computationally cheap. Therefore, just as stated in equations (27) and (28) we see that the form of Crank-Nicolson + Galerkin with lumped mass matrix is as follows;

$$A = Md - (dt/2) * C + (dt/2) * Mod \quad (30)$$
$$B = dt/2 * C + -dt/2 * Mod \quad (31)$$
$$f = dt * v1 \quad (32)$$

Where in the code Md and Mod represents the diagonal mass matrices of the main matrix M and the mass matrix on the boundary term Mo.

2.5 TG3 formulation

To understand the implementation of TG3 we begin with the derivation of the time discretization then after that we will introduce the spatial discretization. The Taylor series is used to derive the time discretization as follows

$$\frac{u(t^{n+1}) - u(t^n)}{\Delta t} = u_t(t^n) + \frac{1}{2}\Delta t u_{tt}(t^n) + \frac{1}{6}\Delta t^2 u_{ttt}(t^n)... \tag{33}$$

since in our problem at hand s=0, h= 0 we have

$$u_t = -\mathbf{a} \cdot \boldsymbol{\nabla} u \tag{34}$$
$$u_{tt} = (\mathbf{a} \cdot \boldsymbol{\nabla})^2 u \tag{35}$$
$$u_{ttt} = (\mathbf{a} \cdot \boldsymbol{\nabla})^2 u_t \tag{36}$$

by inserting u_t in (36) as $(u^{n+1} - u^n)/\Delta t$ the (33) becomes

$$\left[1 - \frac{\Delta t^2}{6}(\mathbf{a} \cdot \boldsymbol{\nabla})^2\right] \frac{u^{n+1} - u^n}{\Delta t} = -(\mathbf{a} \cdot \boldsymbol{\nabla})u^n + \frac{\Delta t}{2}(\mathbf{a} \cdot \boldsymbol{\nabla})^2 u^n \tag{37}$$

Introducing the test function and integrating by part to get TG3 the above equation becomes

$$\begin{aligned}\left(w, \frac{\Delta u}{\Delta t}\right) + \frac{\Delta t^2}{6}\left(\mathbf{a} \cdot \boldsymbol{\nabla} w, \mathbf{a} \cdot \boldsymbol{\nabla}\frac{\Delta u}{\Delta t}\right) - \frac{\Delta t^2}{6}\left((\mathbf{a} \cdot \boldsymbol{n})w, \mathbf{a} \cdot \boldsymbol{\nabla}\frac{\Delta u}{\Delta t}\right)_{\Gamma^{out}} \\ = \left(\mathbf{a} \cdot \boldsymbol{\nabla} w, u^n - \frac{\Delta t}{2}(\mathbf{a} \cdot \boldsymbol{n})u^n\right) - \left((\mathbf{a} \cdot \boldsymbol{n})w, u^n - \frac{\Delta t}{2}(\mathbf{a} \cdot \boldsymbol{n})u^n)\right)_{\Gamma^{out}}\end{aligned} \tag{38}$$

Taking Δt to the right hand sine in th eabove equation we obtain

$$\begin{aligned}(w, \Delta u) + \frac{\Delta t^2}{6}\left(\mathbf{a} \cdot \boldsymbol{\nabla} w, \mathbf{a} \cdot \boldsymbol{\nabla}\frac{\Delta u}{\Delta t}\right) - \frac{\Delta t^2}{6}\left((\mathbf{a} \cdot \boldsymbol{n})w, \mathbf{a} \cdot \boldsymbol{\nabla}\frac{\Delta u}{\Delta t}\right)_{\Gamma^{out}} \\ = \Delta t\left(\mathbf{a} \cdot \boldsymbol{\nabla} w, u^n - \frac{\Delta t}{2}(\mathbf{a} \cdot \boldsymbol{\nabla} u^n)\right) - \Delta t\left((\mathbf{a} \cdot \boldsymbol{n})w, u^n - \frac{\Delta t}{2}(\mathbf{a} \cdot \boldsymbol{\nabla} u^n)\right)_{\Gamma^{out}}\end{aligned} \tag{39}$$

Using (39) we can identify the terms correspond to the TG3 formulation in the code. This process is carried out as follows.

Firstly we consider the **L.H.S** of equation (39)
1. The first term leads to M $=(w, \Delta u)$ where of course $\Delta u = u^{n+1} - u^n$
2. The seconds terms shows that K $= (\boldsymbol{\nabla} w, \boldsymbol{\nabla}\Delta u)$
3. The third term leads to Co $= (w, \boldsymbol{\nabla}\Delta u)$ on the boundary

Using these terms and replacing them in (39) we can show the terms of the L.H.S correspond to:

$$A = M + (dt^2/6) * (K - Co) \tag{40}$$

Secondly we consider the **R.H.S** of equation (39)
1. The first term leads to the discretization of C = $(\boldsymbol{\nabla} w, u^n)$ and K= $(\boldsymbol{\nabla} w, \boldsymbol{\nabla} u^n)$
2. The second term leads to the discretization of Mo = (w, u^n) and Co = $(w, \boldsymbol{\nabla} u^n)$ on the boundary

These terms can replaced in (39) and grouped together as to represent B as follows;

$$B = dt * (C - (dt/2) * K - Mo + (dt/2) * Co \tag{41}$$

Therefore, using (40) and (41) together with the corresponding terms representing the velocities, TG3 formulation is given in the code as

$$A = M + (dt^2/6) * (K - Co) \tag{42}$$
$$B = dt * (C - (dt/2) * K - Mo + (dt/2) * Co \tag{43}$$
$$f = dt * ((dt/2)) * (v2 - vo) + v1) \tag{44}$$

2.6 Two step third order Taylor Galerkin formulation (TG3-2S)

The derivation of this formulation follows closely with that of the one step TG3 method. Here we begin with the two steps shown below

$$\bar{u}^n = u^n + \frac{1}{3}\Delta t u_t^n + \alpha \Delta t^2 u_{tt}^n \tag{45}$$
$$u^{n+1} = u^n + \Delta t u_t^n + \frac{1}{2}\Delta t^2 \bar{u}_{tt}^n \tag{46}$$

Step 1
We start by introducing the test function to the (45) and also we denoting g $u_t = -\mathbf{a} \cdot \boldsymbol{\nabla} u$ after that we will then perform integration by parts.

$$(w, \bar{u}^n - u^n) = -(w, \frac{\Delta t}{3}(\mathbf{a} \cdot \boldsymbol{\nabla} u^n)) + \alpha \Delta t^2 (w, (\mathbf{a} \cdot \boldsymbol{\nabla})^2 u^n) \tag{47}$$

After integration by parts

$$\begin{aligned}(w, \bar{u}^n - u^n) = \frac{\Delta t}{3}(\mathbf{a} \cdot \boldsymbol{\nabla} w, u^n) - \frac{\Delta t}{3}((\mathbf{a} \cdot \boldsymbol{n}) w, u^n)_{\Gamma^{out}} \\ - \alpha(\mathbf{a} \cdot \boldsymbol{\nabla} w, (\mathbf{a} \cdot \boldsymbol{\nabla}) u^n) + \alpha(\mathbf{a} \cdot \boldsymbol{n} w, (\mathbf{a} \cdot \boldsymbol{\nabla}) u^n)_{\Gamma^{out}}\end{aligned} \tag{48}$$

Using (48) we can write the various terms that can be grouped together when performing the discretization process.
1. The first term on the LHS we see that M= $(w, \bar{u}^n - u^n)$
2. From the first trm in the RHS C = $(\boldsymbol{\nabla} w, u^n)$
3. From the second term on the RHS Mo = $(w, u^n)_{\Gamma^{out}}$
4. From the third term K = $(\boldsymbol{\nabla} w, \boldsymbol{\nabla} u^n)$
5. from the fourth term we see that Co = $(w, \boldsymbol{\nabla} u^n)$

Using these new terms we can write the the terms representing the 1st step in the code as

$$A_1 = M \tag{49}$$
$$B1 = (dt/3) * (C - Mo) - \alpha * dt^2 * (K - Co) \tag{50}$$
$$f1 = (dt/3) * v1 + \alpha * dt^2 * (v2 - vo); \tag{51}$$

Step 2

We begin by introducing the test function into (46) and and also we let $u_t^n = -\mathbf{a} \cdot \boldsymbol{\nabla} u$

$$(w, u^{n+1} - u^n) = -\Delta t(w, \mathbf{a} \cdot \boldsymbol{\nabla} u^n) + \frac{1}{2}\Delta t^2 (w, (\mathbf{a} \cdot \boldsymbol{\nabla})^2 u^n)_{\Gamma^{out}} \tag{52}$$

Carrying out integration by parts on the convective terms

$$\begin{aligned}(w, u^{n+1} - u^n) = \Delta t(\mathbf{a} \cdot \boldsymbol{\nabla} w, u^n) - \Delta t((\mathbf{a} \cdot \boldsymbol{n}) w, u^n)_{\Gamma^{out}} \\ + \frac{1}{2}\Delta t^2 (\mathbf{a} \cdot \boldsymbol{\nabla} w, (\mathbf{a} \cdot \boldsymbol{\nabla} u^n)) - \frac{1}{2}\Delta t^2 ((\mathbf{a} \cdot \boldsymbol{n}) w, (\mathbf{a} \cdot \boldsymbol{\nabla} u^n))_{\Gamma^{out}}\end{aligned} \tag{53}$$

By observing equation (53) we can show that the corresponding parts of the code are as follows;

1. From the LHS we see that we have M= $(w, \bar{u}^n - u^n)$
2. From the first term on the RHS of the equation C = $(\boldsymbol{\nabla} w, u^n)$
3. From the second term on the RHS Mo=$(w, u^n)_{\Gamma^{out}}$
4. From the third term on the RHS K = $(\boldsymbol{\nabla} w, \boldsymbol{\nabla} u^n)$
5. From the fourth term on the RHS Co =$(w, \boldsymbol{\nabla} u^n)_{\Gamma^{out}}$

Therefore using the above terms and replacing them in (53)we can see that

$$A2 = M \tag{54}$$
$$B2 = dt * (C - Mo) \tag{55}$$
$$C2 = -(dt^2/2) * (K - Co) \tag{56}$$
$$f2 = dt * v1 - (dt^2/2) * (v2 - vo) \tag{57}$$

The final version of the Two-step TG3 is a combination of the resulting terms from both the first step and the second step. This final form is given as follows.

$$A_1 = M \tag{58}$$
$$B1 = (dt/3) * (C - Mo) - \alpha * dt^2 * (K - Co) \tag{59}$$
$$f1 = (dt/3) * v1 + \alpha * dt^2 * (v2 - vo) \tag{60}$$
$$A2 = M \tag{61}$$
$$B2 = dt * (C - Mo) \tag{62}$$
$$C2 = -(dt^2/2) * (K - Co) \tag{63}$$
$$f2 = dt * v1 - (dt^2/2) * (v2 - vo) \tag{64}$$

It should be noted the above formulation is slightly different from the one in given in the code. The original lines of code are commented on as shown in the Figure 2.1 below.

```
%     B1 = -(dt/3)*C'- alpha*dt^2*(K - Co);
      B1 = (dt/3)*(C-Mo)- alpha*dt^2*(K - Co);
      f1 = (dt/3)*v1 + alpha*dt^2*(v2 - vo);
      A2 = M;
%     B2 = -dt*C';
      B2 = dt*(C-Mo);
      C2 = - (dt^2/2)*(K-Co);
      f2 = dt*v1 - (dt^2/2)*(v2 - vo);
```

Figure 2.1: 2-Step TG3 Code implementation

2.7 Galerkin formulation + fourth order two-step Taylor-Galerkin method

This formulation is given as follows;

$$\bar{u}^n = u^n + \frac{1}{3}\Delta t u_t^n + \frac{1}{12}\Delta t^2 u_{tt}^n \tag{65}$$

$$u^{n+1} = u^n + \Delta t u_t^n + \frac{1}{2}\Delta t^2 \bar{u}_{tt}^n \tag{66}$$

The only difference between this formulation and the previously derived two-step TG3 formulation is the value of α been used. In the two-step TG3 method the value of α is considered to be 1/9 while in the fourth order two-step TG4 formulation α is taken to as 1/12. Therefore instead of go through the same derivation steps as that of TG3 we will just take the final derivation of the two-step TG3 as shown in (58) to (64) which shows that the fourth order two-step TG4 method is derived as;

$$A_1 = M \tag{67}$$

$$B1 = (dt/3) * (C - Mo) - 1/12 * dt^2 * (K - Co) \tag{68}$$

$$f1 = (dt/3) * v1 + 1/12 * dt^2 * (v2 - vo) \tag{69}$$

$$A2 = M \tag{70}$$

$$B2 = dt * (C - Mo) \tag{71}$$

$$C2 = -(dt^2/2) * (K - Co) \tag{72}$$

$$f2 = dt * v1 - (dt^2/2) * (v2 - vo) \tag{73}$$

2.8 Comparison of the explicit methods

TG2: Maximum = 0.978438 and Minimum = -0.011149

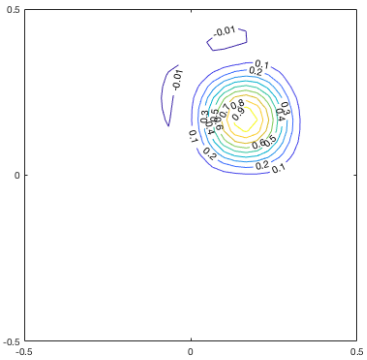

Figure 2.2: TG2 contour plot

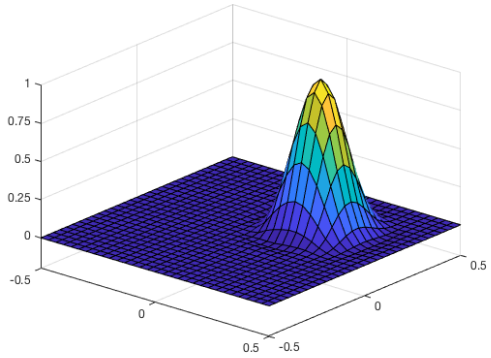

Figure 2.3: TG2 revolution plot

TG2 / diagonal mass matrix: Maximum = 0.818575 and Minimum = -0.177432

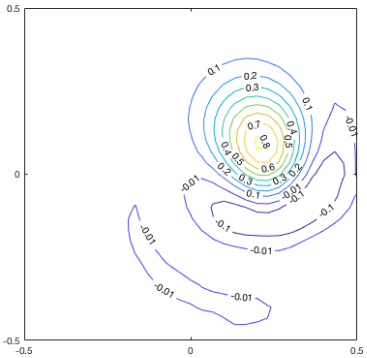

Figure 2.4: TG2/Diag-M contour plot

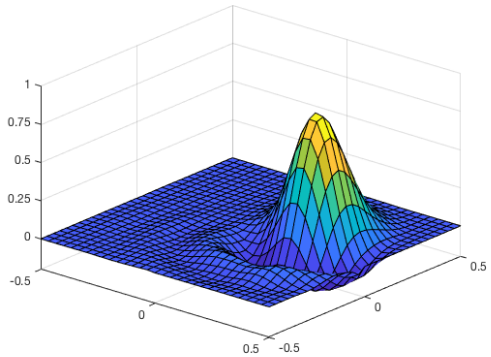

Figure 2.5: TG2/Diag-M revolution plot

TG3 Maximum = 0.983465 Minimum = -0.014839

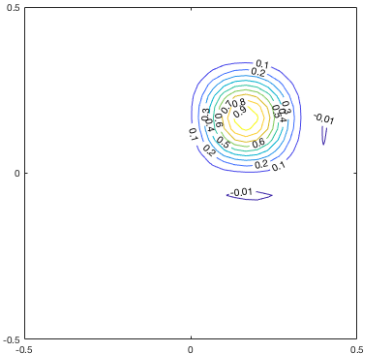

Figure 2.6: TG3 contour plot

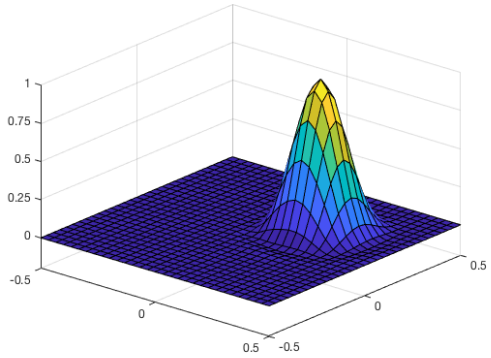

Figure 2.7: TG3 revolution plot

Comparing these implicit methods we see that its clear that the formulations with a consistent mass as shown in figures (2.2) (2.2) and (2.4) (2.5) are more accurate than the TG2 method with the diagonal mass matrix as can bee seen in (2.4) and (2.5). At the same time we have noticed that using a formulation with a diagonal mass matrix is cheaper compared with the formulation with a consistent mass matrix.

2.9 Comparison of the Implicit methods

Crank Nicolson formulation with Maximum = 0.994966 Minimum = -0.022970

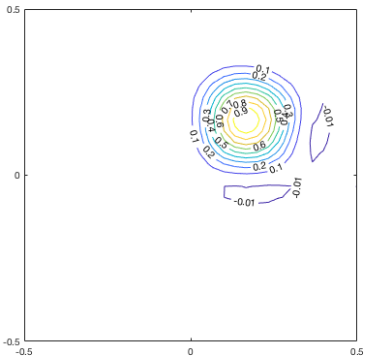

Figure 2.8: CN contour plot

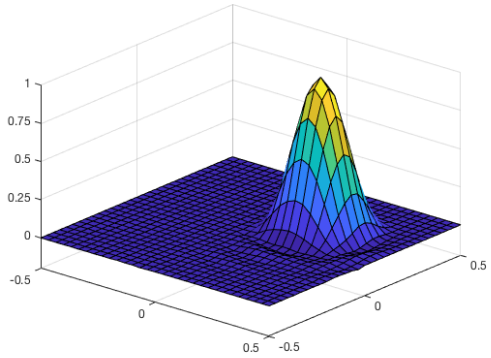

Figure 2.9: CN revolution plot

CN-FD : Maximum = 0.823508 Minimum = -0.210957

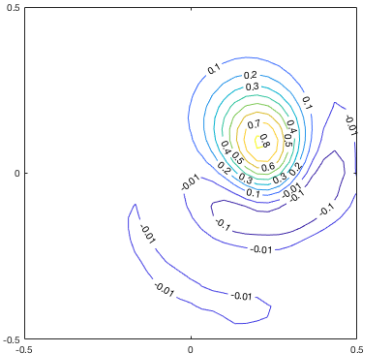

Figure 2.10: CN-FD contour plot

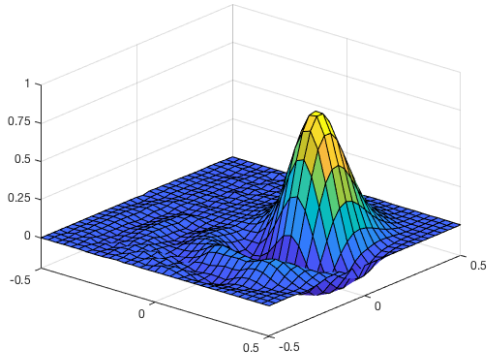

Figure 2.11: CN-FD revolution plot

TG4 : Maximum = 0.992350 Minimum = -0.017285

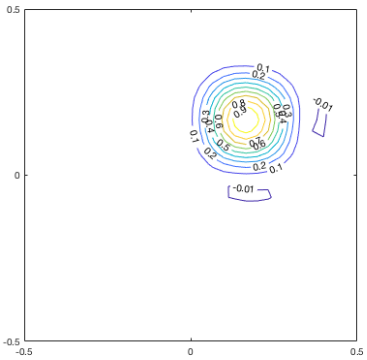

Figure 2.12: TG4 contour plot

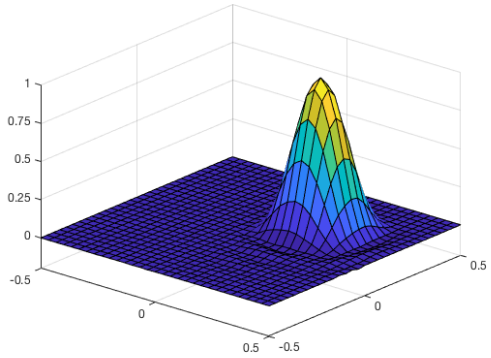

Figure 2.13: TG4 revolution plot

Once again we can notice that from Figures (2.8) (2.9) and (2.12) (2.13), the consistent mass matrix has a large effect on the accuracy of the method when we those shown in Figures (2.10) and (2.11) . This change is vital to understand even when the methods been used are implicit in nature