Name	Ahmed Saeed Mohamed Sherif
Course	Finite Elements in Fluids
Master	MSc Computational Mechanics
Report no.	3
Topic	2D steady and unsteady Transport Equation

Contents

1 2D steady transport equation

1.1 Developing discrete form of the Galerkin/Least-Square (GLS) for steady convection-diffusion-reaction equation

The steady convection-diffusion-reaction equation is given by:

$$\boldsymbol{a}\cdot\boldsymbol{\nabla}u-\boldsymbol{\nabla}\cdot(\nu\boldsymbol{\nabla}u)+\sigma u=s \tag{1}$$

which is alternatively written as:

$$\mathcal{R}(u)=\mathcal{L}(u)-s=0 \tag{2}$$

The Galerkin weak form of equation (1) is written as:

$$\big(w,\boldsymbol{a}\cdot\boldsymbol{\nabla}u\big)_\Omega+\big(\boldsymbol{\nabla}w,\nu\boldsymbol{\nabla}u\big)_\Omega+\big(w,\sigma u\big)_\Omega=\big(w,s\big)_\Omega+\big(w,h\big)_{\Gamma_N} \tag{3}$$

The general form of a consistent stabilization technique is:

$$\big(w,\boldsymbol{a}\cdot\boldsymbol{\nabla}u\big)_\Omega+\big(\boldsymbol{\nabla}w,\nu\boldsymbol{\nabla}u\big)_\Omega+\big(w,\sigma u\big)_\Omega+\sum_e\int_{\Omega_e}\mathcal{P}(w)\tau\mathcal{R}(u)\;\mathrm{d}\Omega=\big(w,s\big)_\Omega+\big(w,h\big)_{\Gamma_N} \tag{4}$$

For Galerkin/Least Square (GLS), the operator $\mathcal{P}$ applied to the test function w is defined as:

$$\mathcal{P}(w)=\mathcal{L}(w)=\boldsymbol{a}\cdot\boldsymbol{\nabla}w-\boldsymbol{\nabla}\cdot(\nu\boldsymbol{\nabla}w)+\sigma w \tag{5}$$

Therefore, the weak form for the GLS method is written as:

$$\begin{aligned}&\big(w,\boldsymbol{a}\cdot\boldsymbol{\nabla}u\big)_\Omega+\big(\boldsymbol{\nabla}w,\nu\boldsymbol{\nabla}u\big)_\Omega+\big(w,\sigma u\big)_\Omega\\&\qquad+\sum_e\int_{\Omega_e}\big[\boldsymbol{a}\cdot\boldsymbol{\nabla}w-\boldsymbol{\nabla}\cdot(\nu\boldsymbol{\nabla}w)+\sigma w\big]\tau\big[\boldsymbol{a}\cdot\boldsymbol{\nabla}u-\boldsymbol{\nabla}\cdot(\nu\boldsymbol{\nabla}u)+\sigma u\big]\;\mathrm{d}\Omega\\&\qquad=\big(w,s\big)_\Omega+\big(w,h\big)_{\Gamma_N}+\sum_e\int_{\Omega_e}\big[\boldsymbol{a}\cdot\boldsymbol{\nabla}w-\boldsymbol{\nabla}\cdot(\nu\boldsymbol{\nabla}w)+\sigma w\big]\tau s\;\mathrm{d}\Omega\end{aligned} \tag{6}$$

and the corresponding part of the Matlab code is:

```
% 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);
```

1.2 2D steady convection-diffusion

1.2.1 Downwind homogeneous natural boundary conditions

Considering a homogeneous convection-diffusion problem on a square domain $[0,1]\times[0,1]$ shown in Figure 1. Discontinuous Dirchlet boundary conditions are imposed on the inlet boundary. On the outlet boundary (drawn in green), homogeneous (zero) Neumann boundary (natural) condition is applied. The convection velocity is of magnitude $\|a\| = 1$ and is skew to the mesh with an angle 30°, and the diffusion coefficient is $\nu = 10^{-4}$. The mesh is composed of 10 bilinear quadrilateral elements in each direction. The solutions obtained using different spatial discretization techniques are shown in Figure 2. It is noticed that Galerkin formulation doesn't satisfactorily resolve the discontinuity and produces spurious oscillations (nearly pure convection). The artificial diffusion method introduces too much crosswind diffusion. The SUPG and GLS produce much better results, where less crosswind diffusion is added and the oscillations of the Galerkin formulation are reduced.

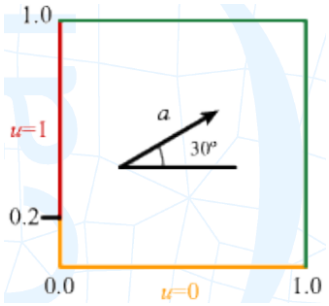
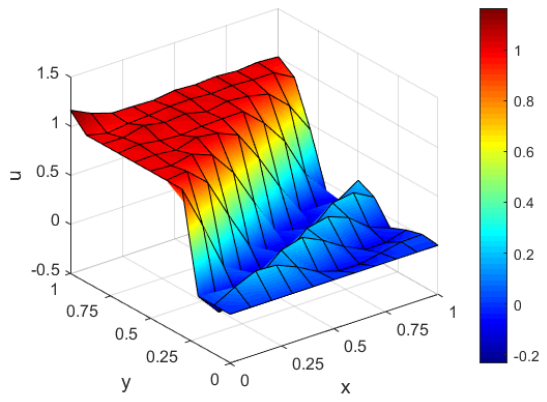
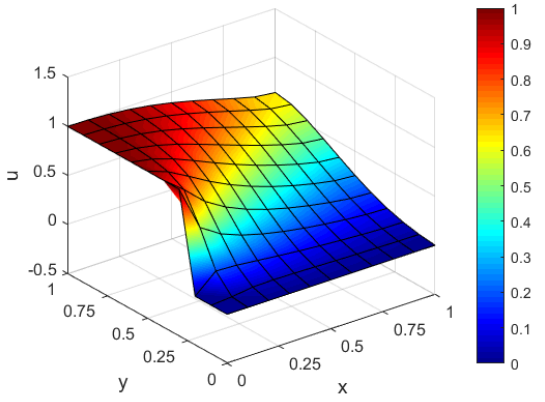


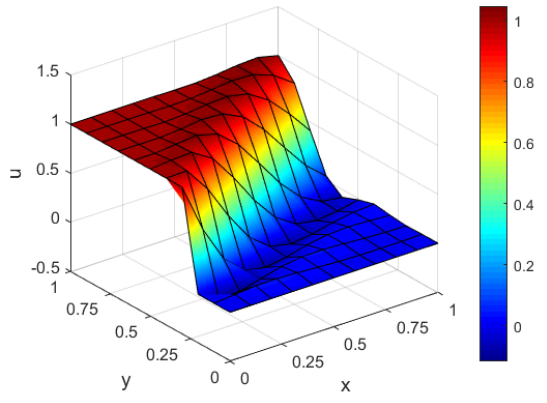
Figure 1: Problem setup



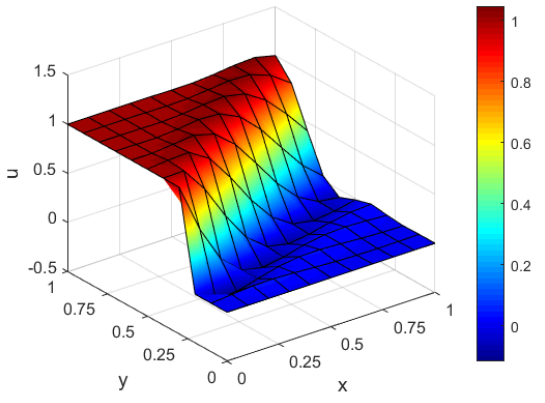
(a) Galerkin



(b) Artificial diffusion



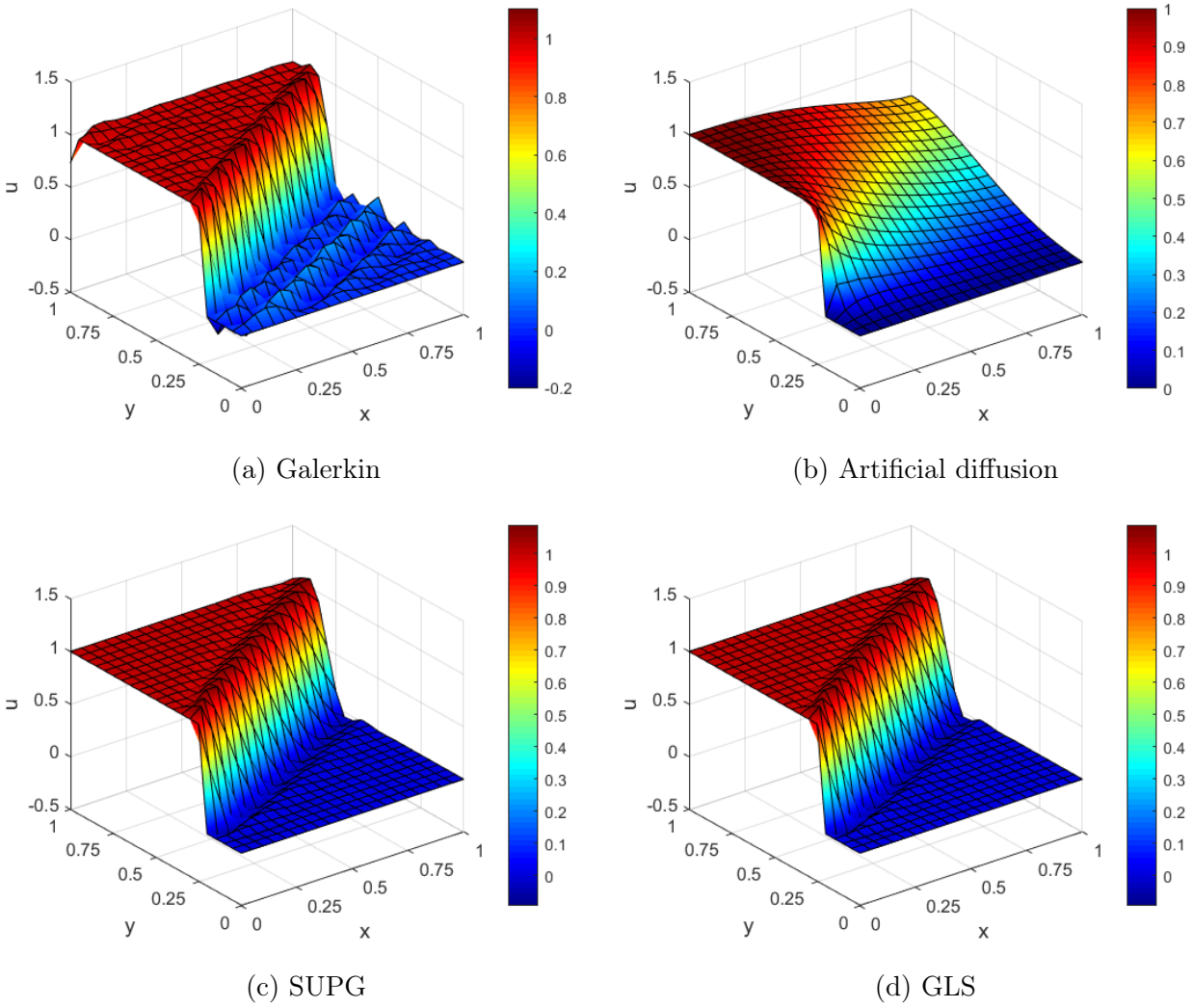
(c) SUPG



(d) GLS

Figure 2: Galerkin, artificial diffusion, SUPG and GLS solution for the 2D steady convection-diffusion problem with downwind natural conditions, using bi-linear quadrilateral elements. (Peclet number = 500)

Keeping all the parameters the same but changing the order of elements from $p = 1$ to $p = 2$, i.e. using bi-quadratic quadrilaterals, the solutions obtained are shown in Figure 3. Better representation of the discontinuity is obtained. However, the Galerkin and artificial diffusion methods still don't provide acceptable results. The results prove that the modifications made to the code, to account for quadratic elements, are correct. The details of the modifications and additions to the code are shown in appendix A.1.



(a) Galerkin

(b) Artificial diffusion

(c) SUPG

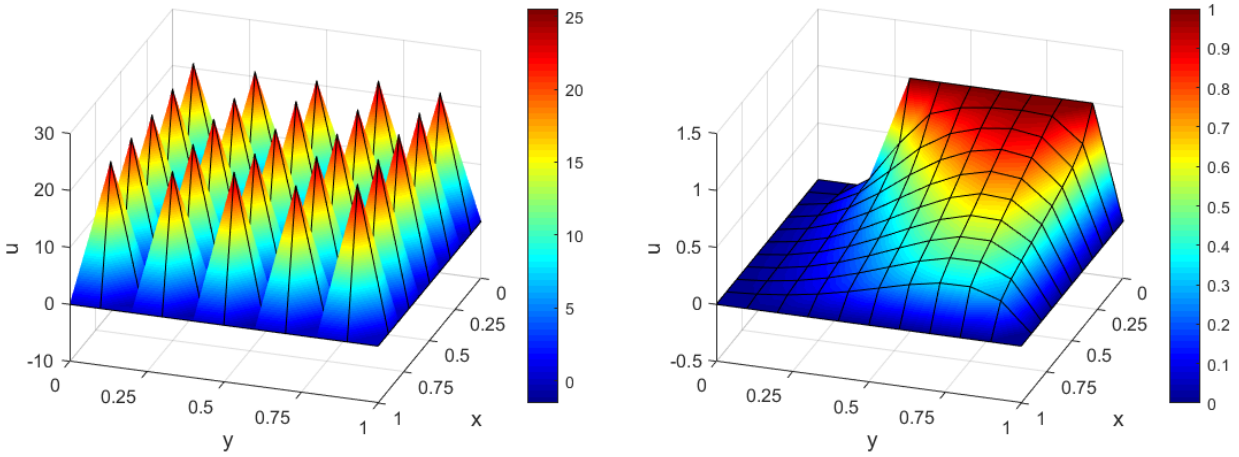
(d) GLS

Figure 3: Galerkin, artificial diffusion, SUPG and GLS solution for the 2D steady convection-diffusion problem with downwind natural conditions, using bi-quadratic quadrilateral elements. (Peclet number = 500)

1.2.2 Downwind homogeneous essential boundary conditions

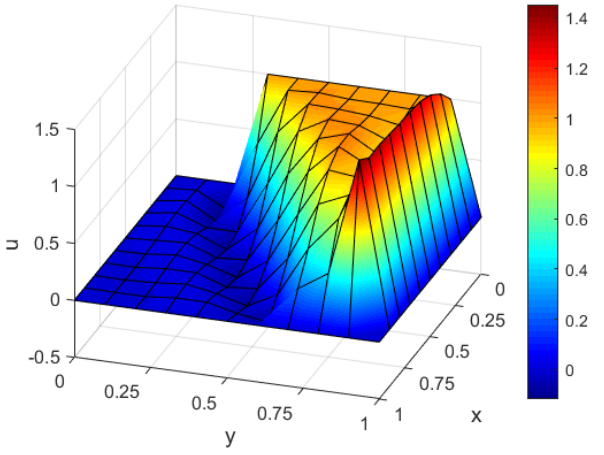
Next, the problem is solved with homogeneous essential boundary conditions (zero Dirchlet) on the outlet boundary. The solution now involves a thin boundary layer at the outlet as seen in Figure 4. However, the boundary layer is not well captured because of the coarse mesh employed. It is observed that the Galerkin formulation yields completely unstable solution. Again, the artificial diffusion method introduces excessive numerical diffusion. Better results are obtained using SUPG and GLS.

The modifications to the code, to account for zero Dirchlet boundary conditions on the outlet boundary, are shown in appendix A.1.1 (lines 43 to 46 and 80 to 87).

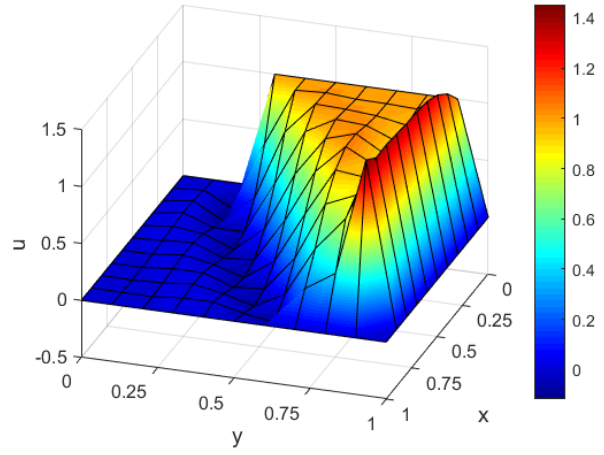


(a) Galerkin

(b) Artificial diffusion



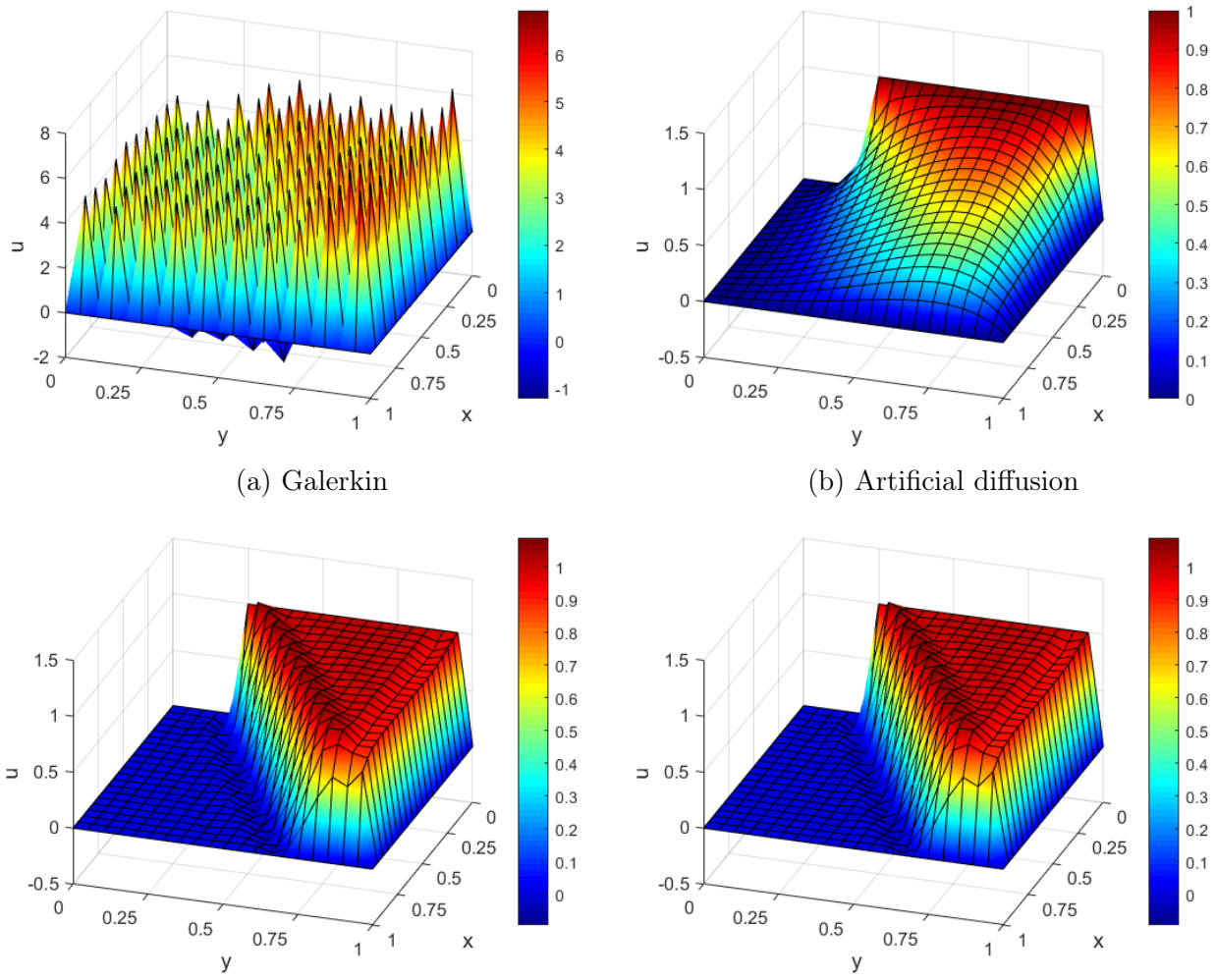
(c) SUPG



(d) GLS

Figure 4: Galerkin, artificial diffusion, SUPG and GLS solution for the 2D steady convection-diffusion problem with downwind essential boundary conditions, using bi-linear quadrilateral elements. (Peclet number = 500)

The solution for the problem with zero Dirchlet condition on the outlet boundary using bi-quadratic quadrilaterals is shown in Figure 5. Similar conclusions as in the case of bi-linear elements are obtained. However, it is clearly seen the more accurate representation of the boundary layer at the outlet in the stable solutions.



(a) Galerkin

(b) Artificial diffusion

(c) SUPG

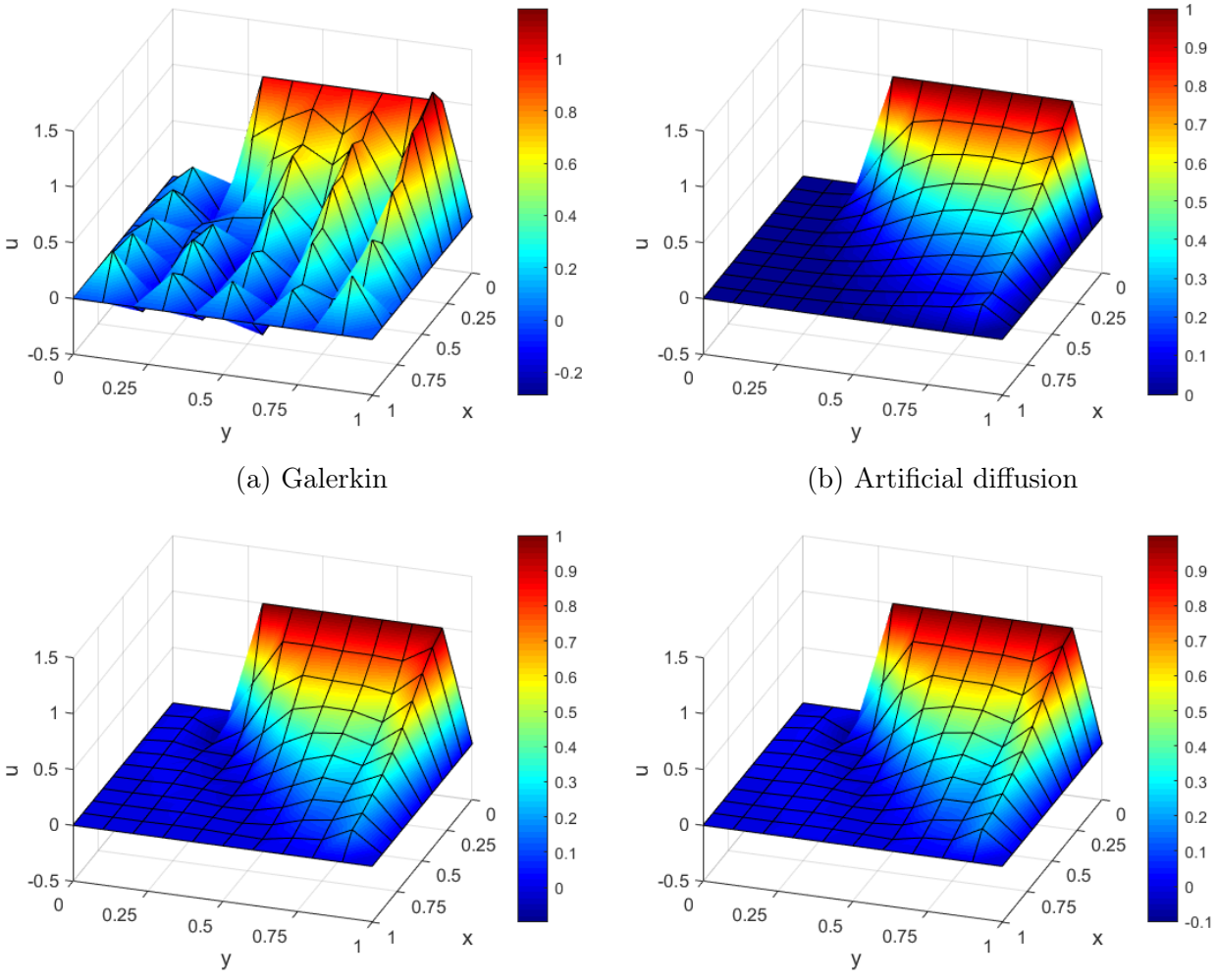
(d) GLS

Figure 5: Galerkin, artificial diffusion, SUPG and GLS solution for the 2D steady convection-diffusion problem with downwind essential boundary conditions, using bi-quadratic quadrilateral elements. (Peclet number = 500)

1.3 2D steady convection-diffusion-reaction

1.3.1 Convection-reaction-dominated case

Setting the following parameters: $\|a\| = 1/2$, $\nu = 10^{-4}$, $\sigma = 1$, $s = 0$, downwind essential boundary conditions, bi-linear quadrilateral elements. The solution is shown in Figure 6. It is seen that the Galerkin formulation yields unstable solution because the Peclet number is 250 (more than 1) as the problem is convection dominated. It is also noted that the stabilized methods yields satisfactory solutions.



(a) Galerkin

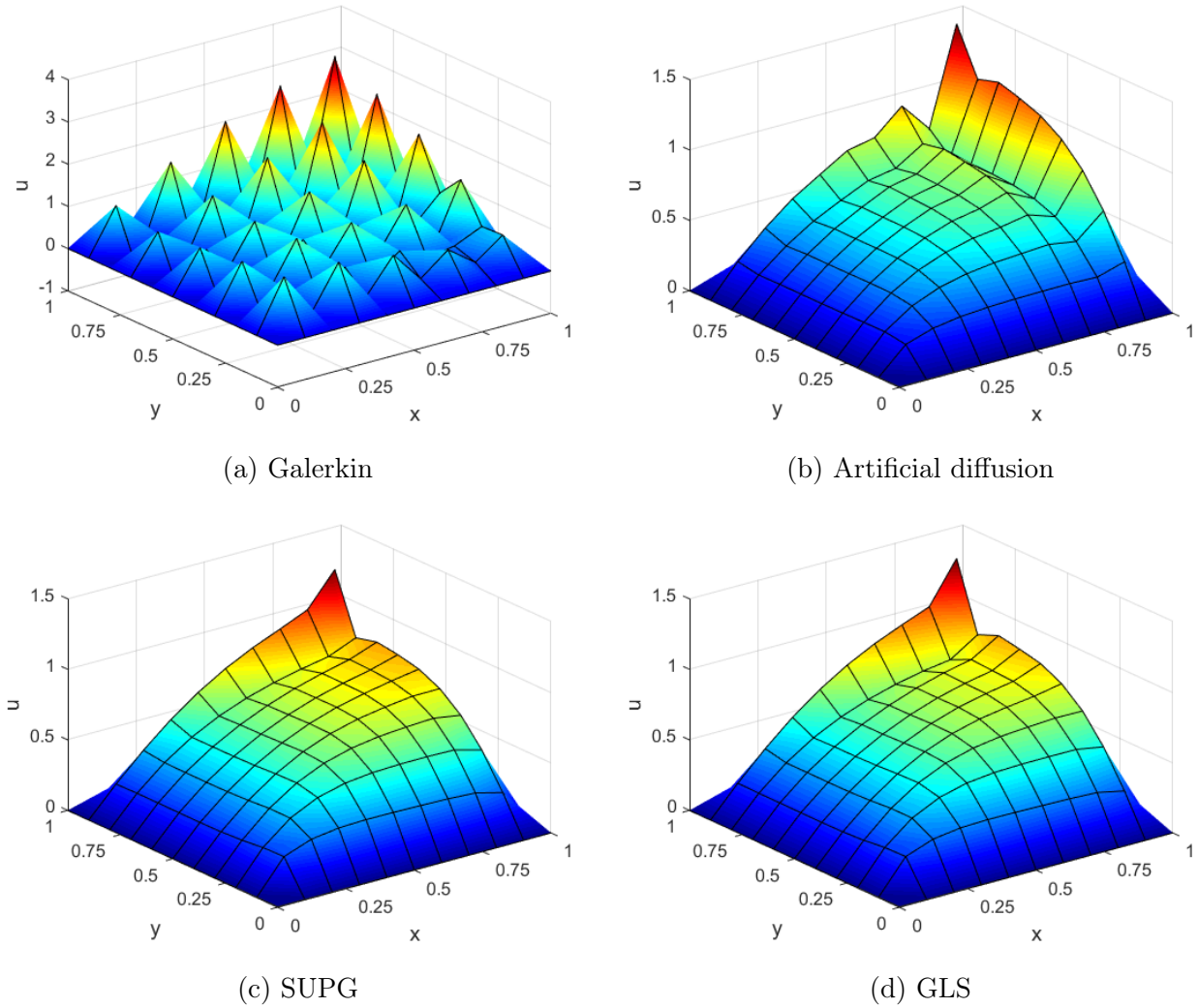
(b) Artificial diffusion

(c) SUPG

(d) GLS

Figure 6: Galerkin, artificial diffusion, SUPG and GLS solution for the 2D steady convection-reaction dominated problem with downwind essential boundary conditions and no source term, using bi-linear quadrilateral elements. (Peclet number = 250)

To duplicate the results obtained in the book at page 77 (Fig. 2.21), the following parameters are used: $\|a\| = 1/2$, $\nu = 10^{-4}$, $\sigma = 1$, $s = 1$ homogeneous essential boundary conditions everywhere, bi-linear quadrilateral elements. The variation here is that the source term is not zero and that all boundaries are homogeneous Dirchlet. The solution is shown in Figure 7. Again, the Galerkin formulation gives unstable solution (Peclet $= 250 > 1$). The SUPG and GLS formulations gives better solutions that the Artificial diffusion formulation.



(a) Galerkin

(b) Artificial diffusion

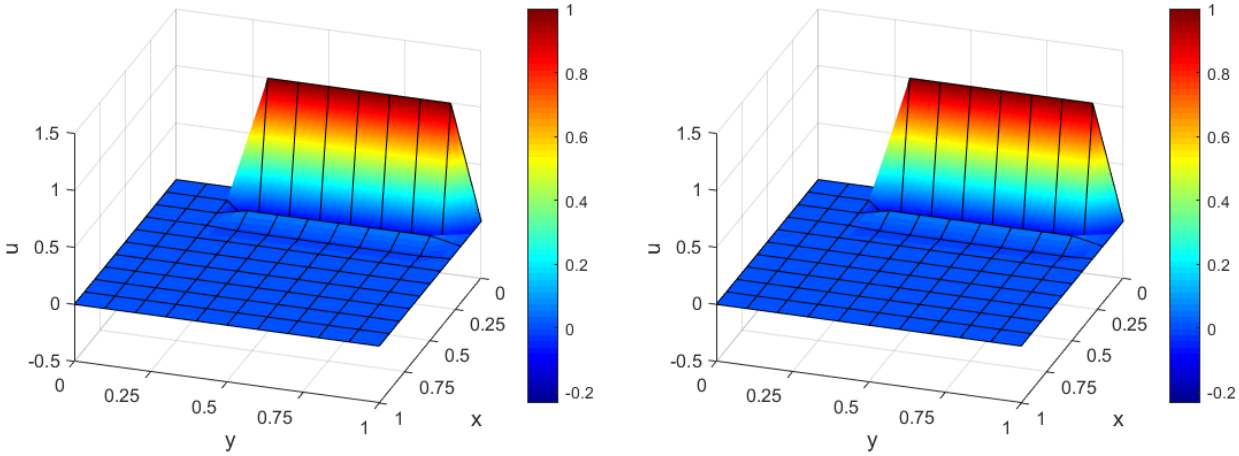
(c) SUPG

(d) GLS

Figure 7: Galerkin, artificial diffusion, SUPG and GLS solution for the 2D steady convection-reaction dominated problem with homogeneous essential boundary conditions everywhere and a source term $s = 1$, using bi-linear quadrilateral elements. (Peclet number $= 250$)

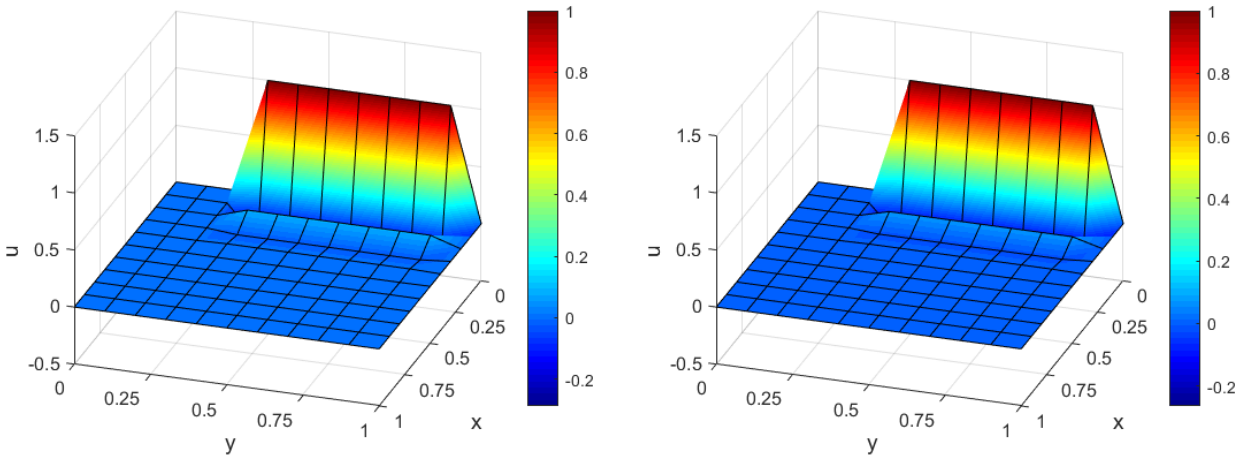
1.3.2 Reaction-dominated case

Setting the following parameters: $\|a\| = 10^{-3}$, $\nu = 10^{-4}$, $\sigma = 1$, $s = 0$, downwind essential boundary conditions, bi-linear quadrilateral elements. The solution is shown in Figure 8. It is seen that all formulations yields very similar solutions. In this case, the Peclet number is 0.5 which is less than the stability limit (Pe=1), therefore the Galerkin formulation is able to yield stable solution.



(a) Galerkin

(b) Artificial diffusion

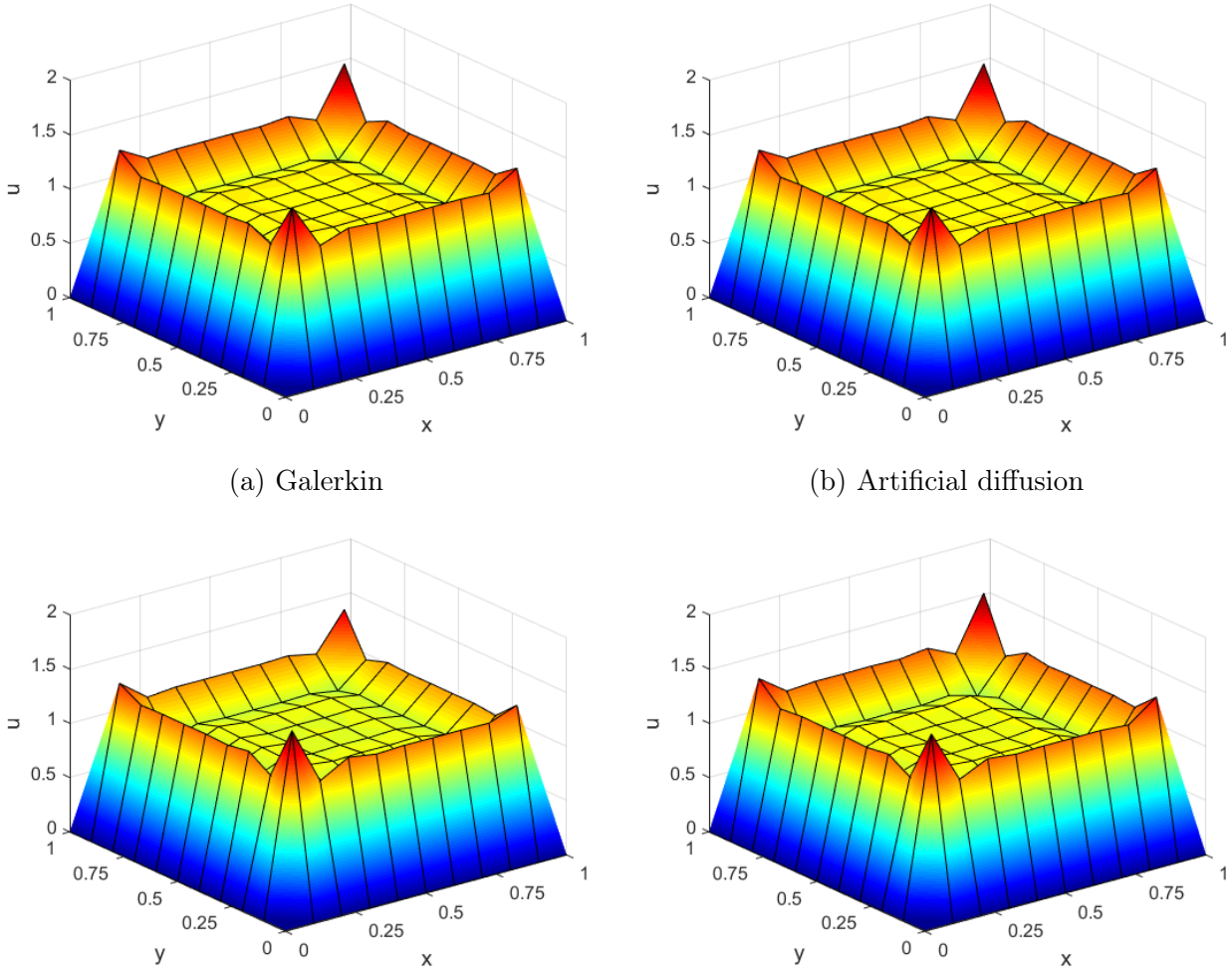


(c) SUPG

(d) GLS

Figure 8: Galerkin, artificial diffusion, SUPG and GLS solution for the 2D steady reaction dominated problem with downwind essential boundary conditions and no source term, using bi-linear quadrilateral elements. (Peclet number = 0.5)

To duplicate the results obtained in the book at page 77 (Fig. 2.22), the following parameters are used: $\|a\| = 10^{-3}$, $\nu = 10^{-4}$, $\sigma = 1$, $s = 1$, homogeneous essential boundary conditions everywhere, bi-linear quadrilateral elements. The variation here is that the source term is not zero and that all boundaries are homogeneous Dirchlet. The solution is shown in Figure 9.



(a) Galerkin

(b) Artificial diffusion

(c) SUPG

(d) GLS

Figure 9: Galerkin, artificial diffusion, SUPG and GLS solution for the 2D steady reaction dominated problem with homogeneous essential boundary conditions everywhere and a source term $s = 1$, using bi-linear quadrilateral elements. (Peclet number = 0.5)

2 2D unsteady transport equation

In this section, the 2D unsteady pure convection equation is considered

$$u_t + \boldsymbol{a} \cdot \boldsymbol{\nabla} u = s \tag{7}$$

2.1 Lax-Wendroff method + Galerkin formulation

The time discretization is given by:

$$\frac{\Delta u}{\Delta t} = -\boldsymbol{a} \cdot \boldsymbol{\nabla} u^n + \frac{\Delta t}{2}(\boldsymbol{a} \cdot \boldsymbol{\nabla})^2 u^n + s^n + \frac{\Delta t}{2}(s_t^n - \boldsymbol{a} \cdot \boldsymbol{\nabla} s^n) \tag{8}$$

The Galerkin weak form is:

$$\big(w, \frac{\Delta u}{\Delta t}\big)_\Omega = -\big(w, \boldsymbol{a} \cdot \boldsymbol{\nabla} u^n\big)_\Omega + \frac{\Delta t}{2}\big(w, (\boldsymbol{a} \cdot \boldsymbol{\nabla})^2 u^n\big)_\Omega + \big(w, s^n\big)_\Omega + \frac{\Delta t}{2}\big(w, s_t^n\big)_\Omega - \frac{\Delta t}{2}\big(w, \boldsymbol{a} \cdot \boldsymbol{\nabla} s^n\big)_\Omega \tag{9}$$

Integration by parts for the first two terms and the last term on the RHS yields:

$$\begin{aligned}\big(w, \frac{\Delta u}{\Delta t}\big)_\Omega = \big(\boldsymbol{a} \cdot \boldsymbol{\nabla} w, u^n\big)_\Omega - \frac{\Delta t}{2}\big(\boldsymbol{a} \cdot \boldsymbol{\nabla} w, \boldsymbol{a} \cdot \boldsymbol{\nabla} u^n\big)_\Omega + \frac{\Delta t}{2}\big(\boldsymbol{a} \cdot \boldsymbol{\nabla} w, s^n\big)_\Omega \\ -\big((\boldsymbol{a} \cdot \boldsymbol{n}) w, u^n\big)_\Gamma + \frac{\Delta t}{2}\big((\boldsymbol{a} \cdot \boldsymbol{n}) w, \boldsymbol{a} \cdot \boldsymbol{\nabla} u^n\big)_\Gamma - \frac{\Delta t}{2}\big((\boldsymbol{a} \cdot \boldsymbol{n}) w, s^n\big)_\Gamma \\ +\big(w, s^n\big)_\Omega + \frac{\Delta t}{2}\big(w, s_t^n\big)_\Omega\end{aligned} \tag{10}$$

Given that the boundary $\Gamma = \Gamma^{in} \cup \Gamma^{out}$, and recalling that for hyperbolic problems the boundary conditions are applied only to inflow boundaries. Furthermore, the inflow boundary could be a Neumann part and a Dirchlet part, i.e. $\Gamma^{in} = \Gamma^{in}_N \cup \Gamma^{in}_D$. Since $w = 0$ on the Dirchlet boundary, therefore the weak form is written as:

$$\begin{aligned}\big(w, \frac{\Delta u}{\Delta t}\big)_\Omega = \big(\boldsymbol{a} \cdot \boldsymbol{\nabla} w, u^n\big)_\Omega - \frac{\Delta t}{2}\big(\boldsymbol{a} \cdot \boldsymbol{\nabla} w, \boldsymbol{a} \cdot \boldsymbol{\nabla} u^n\big)_\Omega + \frac{\Delta t}{2}\big(\boldsymbol{a} \cdot \boldsymbol{\nabla} w, s^n\big)_\Omega \\ -\big((\boldsymbol{a} \cdot \boldsymbol{n}) w, u^n\big)_{\Gamma^{out}} + \frac{\Delta t}{2}\big((\boldsymbol{a} \cdot \boldsymbol{n}) w, \boldsymbol{a} \cdot \boldsymbol{\nabla} u^n\big)_{\Gamma^{out}} - \frac{\Delta t}{2}\big((\boldsymbol{a} \cdot \boldsymbol{n}) w, s^n\big)_{\Gamma^{out}} \\ -\big((\boldsymbol{a} \cdot \boldsymbol{n}) w, u^n\big)_{\Gamma^{in}_N} + \frac{\Delta t}{2}\big((\boldsymbol{a} \cdot \boldsymbol{n}) w, \boldsymbol{a} \cdot \boldsymbol{\nabla} u^n\big)_{\Gamma^{in}_N} - \frac{\Delta t}{2}\big((\boldsymbol{a} \cdot \boldsymbol{n}) w, s^n\big)_{\Gamma^{in}_N} \\ +\big(w, s^n\big)_\Omega + \frac{\Delta t}{2}\big(w, s_t^n\big)_\Omega\end{aligned} \tag{11}$$

For a problem with homogeneous Dirchlet boundary condition at the inlet, and assuming a source term that is not time-dependent, i.e. $s_t = 0$, therefore the weak form is simplified to:

$$\begin{aligned}\big(w, \frac{\Delta u}{\Delta t}\big)_\Omega = \big(\boldsymbol{a} \cdot \boldsymbol{\nabla} w, u^n\big)_\Omega - \frac{\Delta t}{2}\big(\boldsymbol{a} \cdot \boldsymbol{\nabla} w, \boldsymbol{a} \cdot \boldsymbol{\nabla} u^n\big)_\Omega + \frac{\Delta t}{2}\big(\boldsymbol{a} \cdot \boldsymbol{\nabla} w, s\big)_\Omega \\ -\big((\boldsymbol{a} \cdot \boldsymbol{n}) w, u^n\big)_{\Gamma^{out}} + \frac{\Delta t}{2}\big((\boldsymbol{a} \cdot \boldsymbol{n}) w, \boldsymbol{a} \cdot \boldsymbol{\nabla} u^n\big)_{\Gamma^{out}} - \frac{\Delta t}{2}\big((\boldsymbol{a} \cdot \boldsymbol{n}) w, s\big)_{\Gamma^{out}} + \big(w, s\big)_\Omega\end{aligned} \tag{12}$$

Using the following definitions:

$$\begin{aligned}
\boldsymbol{MU} &= \big(w, u\big)_{\Omega}, \\
\boldsymbol{CU} &= \big(\boldsymbol{a}\cdot\boldsymbol{\nabla} w, u\big)_{\Omega}, \\
\boldsymbol{KU} &= \big(\boldsymbol{a}\cdot\boldsymbol{\nabla} w, \boldsymbol{a}\cdot\boldsymbol{\nabla} u\big)_{\Omega}, \\
\boldsymbol{v_1} &= \big(w, s\big)_{\Omega}, \\
\boldsymbol{v_2} &= \big(\boldsymbol{a}\cdot\boldsymbol{\nabla} w, s\big)_{\Omega}, \\
\boldsymbol{M_oU} &= \big((\boldsymbol{a}\cdot\boldsymbol{n})w, u^n\big)_{\Gamma^{out}}, \\
\boldsymbol{C_oU} &= \big((\boldsymbol{a}\cdot\boldsymbol{n})w, \boldsymbol{a}\cdot\boldsymbol{\nabla} u^n\big)_{\Gamma^{out}}, \\
\boldsymbol{v_o} &= \big((\boldsymbol{a}\cdot\boldsymbol{n})w, s\big)_{\Omega}
\end{aligned} \tag{13}$$

Therefore, the weak form given by (12) is written in discrete form as:

$$\frac{1}{\Delta t}\boldsymbol{M\Delta U} = \boldsymbol{CU}^n - \frac{\Delta t}{2}\boldsymbol{KU}^n + \frac{\Delta t}{2}\boldsymbol{v_2} - \boldsymbol{M_oU}^n + \frac{\Delta t}{2}\boldsymbol{C_oU}^n - \frac{\Delta t}{2}\boldsymbol{v_2} + \boldsymbol{v_1} \tag{14}$$

and is further simplified to:

$$\boldsymbol{M\Delta U} = \Delta t\Big[\boldsymbol{C} - \frac{\Delta t}{2}\boldsymbol{K} - \boldsymbol{M_o} + \frac{\Delta t}{2}\boldsymbol{C_o}\Big]\boldsymbol{U}^n + \Delta t\Big[\boldsymbol{v_1} + \frac{\Delta t}{2}\big(\boldsymbol{v_2} - \boldsymbol{v_2}\big)\Big] \tag{15}$$

and written in the following format, for the ease of implementation in Matlab:

$$\boldsymbol{A\Delta U} = \boldsymbol{BU}^n + \boldsymbol{f}$$

and the corresponding part of the Matlab code is:

```
% COMPUTATION OF MATRICES FOR THE TRANSIENT ANALYSIS
if meth == 1
    A = M;
    B = dt*(C - (dt/2)*K - Mo + (dt/2)*Co);
    f = dt*(v1 + (dt/2)*(v2-vo));
```

If lumped mass matrix is used, the corresponding part of the Matlab code is:

```
elseif meth == 2
    Md = diag(M*ones(numnp,1));
    Mod = diag(Mo*ones(numnp,1));
    A = Md;
    B = dt*(C - (dt/2)*K - Mod + (dt/2)*Co);
    f = dt*(v1 + (dt/2)*(v2-vo));
```

2.2 Crank-Nicholson method + Galerkin formulation

The time discretization is given by:

$$\frac{\Delta u}{\Delta t} + \frac{1}{2}(\boldsymbol{a}\cdot\boldsymbol{\nabla})\Delta u = \frac{1}{2}s^{n+1} + \frac{1}{2}s^{n} - \boldsymbol{a}\cdot\boldsymbol{\nabla}u^{n} \tag{16}$$

The Galerkin weak form is:

$$\big(w, \frac{\Delta u}{\Delta t}\big)_{\Omega} + \frac{1}{2}\big(w, (\boldsymbol{a}\cdot\boldsymbol{\nabla})\Delta u\big)_{\Omega} = \frac{1}{2}\big(w, s^{n+1}\big)_{\Omega} + \frac{1}{2}\big(w, s^{n}\big)_{\Omega} - \big(w, \boldsymbol{a}\cdot\boldsymbol{\nabla}u^{n}\big)_{\Omega} \tag{17}$$

Integration by parts for the second term on the LHS and the last term on the RHS yields:

$$\begin{aligned}\big(w, \frac{\Delta u}{\Delta t}\big)_{\Omega} - \frac{1}{2}\big(\boldsymbol{a}\cdot\boldsymbol{\nabla}w, \Delta u\big)_{\Omega} + \frac{1}{2}\big((\boldsymbol{a}\cdot\boldsymbol{n})w, \Delta u\big)_{\Gamma} \\ = \frac{1}{2}\big(w, s^{n+1}\big)_{\Omega} + \frac{1}{2}\big(w, s^{n}\big)_{\Omega} + \big(\boldsymbol{a}\cdot\boldsymbol{\nabla}w, u^{n}\big)_{\Omega} - \big((\boldsymbol{a}\cdot\boldsymbol{n})w, u^{n}\big)_{\Gamma}\end{aligned} \tag{18}$$

Given that the boundary $\Gamma = \Gamma^{in} \cup \Gamma^{out}$, and recalling that for hyperbolic problems the boundary conditions are applied only to inflow boundaries. Furthermore, the inflow boundary could be a Neumann part and a Dirchlet part, i.e. $\Gamma^{in} = \Gamma^{in}_{N} \cup \Gamma^{in}_{D}$. Since $w = 0$ on the Dirchlet boundary, therefore the weak form is written as:

$$\begin{aligned}\big(w, \frac{\Delta u}{\Delta t}\big)_{\Omega} - \frac{1}{2}\big(\boldsymbol{a}\cdot\boldsymbol{\nabla}w, \Delta u\big)_{\Omega} + \frac{1}{2}\big((\boldsymbol{a}\cdot\boldsymbol{n})w, \Delta u\big)_{\Gamma^{out}} + \frac{1}{2}\big((\boldsymbol{a}\cdot\boldsymbol{n})w, \Delta u\big)_{\Gamma^{in}_{N}} \\ = \frac{1}{2}\big(w, s^{n+1}\big)_{\Omega} + \frac{1}{2}\big(w, s^{n}\big)_{\Omega} + \big(\boldsymbol{a}\cdot\boldsymbol{\nabla}w, u^{n}\big)_{\Omega} - \big((\boldsymbol{a}\cdot\boldsymbol{n})w, u^{n}\big)_{\Gamma^{out}} - \big((\boldsymbol{a}\cdot\boldsymbol{n})w, u^{n}\big)_{\Gamma^{in}_{N}}\end{aligned} \tag{19}$$

Moving the fourth term on the LHS to the RHS yields:

$$\begin{aligned}\big(w, \frac{\Delta u}{\Delta t}\big)_{\Omega} - \frac{1}{2}\big(\boldsymbol{a}\cdot\boldsymbol{\nabla}w, \Delta u\big)_{\Omega} + \frac{1}{2}\big((\boldsymbol{a}\cdot\boldsymbol{n})w, \Delta u\big)_{\Gamma^{out}} \\ = \frac{1}{2}\big(w, s^{n+1}\big)_{\Omega} + \frac{1}{2}\big(w, s^{n}\big)_{\Omega} + \big(\boldsymbol{a}\cdot\boldsymbol{\nabla}w, u^{n}\big)_{\Omega} - \big((\boldsymbol{a}\cdot\boldsymbol{n})w, u^{n}\big)_{\Gamma^{out}} \\ - \frac{1}{2}\big((\boldsymbol{a}\cdot\boldsymbol{n})w, u^{n+1}\big)_{\Gamma^{in}_{N}} - \frac{1}{2}\big((\boldsymbol{a}\cdot\boldsymbol{n})w, u^{n}\big)_{\Gamma^{in}_{N}}\end{aligned} \tag{20}$$

For a problem with homogeneous Dirchlet boundary condition at the inlet, and assuming a source term that is not time-dependent, i.e. $s^{n+1} = s^{n} = s$, therefore the weak form is simplified to:

$$\begin{aligned}\big(w, \frac{\Delta u}{\Delta t}\big)_{\Omega} - \frac{1}{2}\big(\boldsymbol{a}\cdot\boldsymbol{\nabla}w, \Delta u\big)_{\Omega} + \frac{1}{2}\big((\boldsymbol{a}\cdot\boldsymbol{n})w, \Delta u\big)_{\Gamma^{out}} \\ = \frac{1}{2}\big(w, s\big)_{\Omega} + \frac{1}{2}\big(w, s\big)_{\Omega} + \big(\boldsymbol{a}\cdot\boldsymbol{\nabla}w, u^{n}\big)_{\Omega} - \big((\boldsymbol{a}\cdot\boldsymbol{n})w, u^{n}\big)_{\Gamma^{out}}\end{aligned} \tag{21}$$

Using the previous definitions given by (13), the weak form given by (21) is written in discrete form as:

$$\frac{1}{\Delta t}\boldsymbol{M}\boldsymbol{\Delta U} - \frac{1}{2}\boldsymbol{C}\boldsymbol{\Delta U} + \frac{1}{2}\boldsymbol{M_o}\boldsymbol{\Delta U} = \boldsymbol{v_1} + \boldsymbol{C}\boldsymbol{U}^{n} - \boldsymbol{M_o}\boldsymbol{U}^{n} \tag{22}$$

further simplification yields:

$$\Big[\boldsymbol{M} - \frac{\Delta t}{2}\boldsymbol{C} + \frac{\Delta t}{2}\boldsymbol{M_o}\Big]\boldsymbol{\Delta U} = \Big[\Delta t\boldsymbol{C} - \Delta t\boldsymbol{M_o}\Big]\boldsymbol{U}^n + \Delta t\boldsymbol{v_1} \tag{23}$$

and for ease of implementation in Matlab, it is written as:

$$\boldsymbol{A\Delta U} = \boldsymbol{BU}^n + \boldsymbol{f}$$

The corresponding part of the Matlab code is:

```
elseif meth == 4
    A = M - (dt/2)*C + (dt/2)*Mo;
    B = dt*C - dt*Mo;
    f = dt*v1;
```

If lumped mass matrix is used, the corresponding part of the Matlab code is:

```
elseif meth == 5
    Md = diag(M*ones(numnp,1));
    Mod = diag(Mo*ones(numnp,1));
    A = Md - (dt/2)*C + (dt/2)*Mod;
    B = dt*C - dt*Mod;
    f = dt*v1;
```

2.3 Third-order Taylor-Galerkin (TG3) + Galerkin formulation

Assuming that the source term is not time dependent, therefore the time discretization using third-order Taylor-Galerkin method is given by:

$$\Big[1 - \frac{\Delta t^2}{6}(\boldsymbol{a}\cdot\boldsymbol{\nabla})^2\Big]\frac{\Delta u}{\Delta t} = -(\boldsymbol{a}\cdot\boldsymbol{\nabla})u^n + \frac{\Delta t}{2}(\boldsymbol{a}\cdot\boldsymbol{\nabla})^2u^n + s - \frac{\Delta t}{2}\boldsymbol{a}\cdot\boldsymbol{\nabla}s \tag{24}$$

The Galerkin weak form is given by:

$$\big(w, \frac{\Delta u}{\Delta t}\big)_\Omega - \frac{\Delta t^2}{6}\big(w, (\boldsymbol{a}\cdot\boldsymbol{\nabla})^2\frac{\Delta u}{\Delta t}\big)_\Omega = -\big(w, \boldsymbol{a}\cdot\boldsymbol{\nabla}u^n\big)_\Omega + \frac{\Delta t}{2}\big(w, (\boldsymbol{a}\cdot\boldsymbol{\nabla})^2u^n\big)_\Omega + \big(w, s\big)_\Omega - \frac{\Delta t}{2}\big(w, \boldsymbol{a}\cdot\boldsymbol{\nabla}s\big)_\Omega \tag{25}$$

Integration by parts yields:

$$\begin{aligned}\big(w, \frac{\Delta u}{\Delta t}\big)_\Omega + \frac{\Delta t^2}{6}\big(\boldsymbol{a}\cdot\boldsymbol{\nabla}w, \boldsymbol{a}\cdot\boldsymbol{\nabla}\frac{\Delta u}{\Delta t}\big)_\Omega - \frac{\Delta t^2}{6}\big((\boldsymbol{a}\cdot\boldsymbol{n})w, \boldsymbol{a}\cdot\boldsymbol{\nabla}\frac{\Delta u}{\Delta t}\big)_\Gamma \\ = \big(\boldsymbol{a}\cdot\boldsymbol{\nabla}w, u^n\big)_\Omega - \big((\boldsymbol{a}\cdot\boldsymbol{n})w, u^n\big)_\Gamma - \frac{\Delta t}{2}\big(\boldsymbol{a}\cdot\boldsymbol{\nabla}w, \boldsymbol{a}\cdot\boldsymbol{\nabla}u^n\big)_\Omega + \frac{\Delta t}{2}\big((\boldsymbol{a}\cdot\boldsymbol{n})w, \boldsymbol{a}\cdot\boldsymbol{\nabla}u^n\big)_\Gamma \\ + \big(w, s\big)_\Omega + \frac{\Delta t}{2}\big(\boldsymbol{a}\cdot\boldsymbol{\nabla}w, s\big)_\Omega - \frac{\Delta t}{2}\big((\boldsymbol{a}\cdot\boldsymbol{n})w, s\big)_\Gamma\end{aligned} \tag{26}$$

For a problem with homogeneous Dirchlet boundary condition at the inlet, the weak form is written as:

$$\begin{aligned}\big(w,\frac{\Delta u}{\Delta t}\big)_\Omega+\frac{\Delta t^2}{6}\big(\boldsymbol{a}\cdot\boldsymbol{\nabla}w,\boldsymbol{a}\cdot\boldsymbol{\nabla}\frac{\Delta u}{\Delta t}\big)_\Omega-\frac{\Delta t^2}{6}\big((\boldsymbol{a}\cdot\boldsymbol{n})w,\boldsymbol{a}\cdot\boldsymbol{\nabla}\frac{\Delta u}{\Delta t}\big)_{\Gamma^{out}}\\ =\big(\boldsymbol{a}\cdot\boldsymbol{\nabla}w,u^n\big)_\Omega-\big((\boldsymbol{a}\cdot\boldsymbol{n})w,u^n\big)_{\Gamma^{out}}-\frac{\Delta t}{2}\big(\boldsymbol{a}\cdot\boldsymbol{\nabla}w,\boldsymbol{a}\cdot\boldsymbol{\nabla}u^n\big)_\Omega+\frac{\Delta t}{2}\big((\boldsymbol{a}\cdot\boldsymbol{n})w,\boldsymbol{a}\cdot\boldsymbol{\nabla}u^n\big)_{\Gamma^{out}}\\ +\big(w,s\big)_\Omega+\frac{\Delta t}{2}\big(\boldsymbol{a}\cdot\boldsymbol{\nabla}w,s\big)_\Omega-\frac{\Delta t}{2}\big((\boldsymbol{a}\cdot\boldsymbol{n})w,s\big)_{\Gamma^{out}}\end{aligned}\tag{27}$$

Using the previous definitions given by (13), the weak form given by (27) is written in discrete form as:

$$\frac{1}{\Delta t}\boldsymbol{M}\boldsymbol{\Delta U}+\frac{\Delta t}{6}\boldsymbol{K}\boldsymbol{\Delta U}-\frac{\Delta t}{6}\boldsymbol{C_o}\boldsymbol{\Delta U}=\boldsymbol{C}\boldsymbol{U}^n-\boldsymbol{M_o}\boldsymbol{U}^n-\frac{\Delta t}{2}\boldsymbol{K}\boldsymbol{U}^n+\frac{\Delta t}{2}\boldsymbol{C_o}\boldsymbol{U}^n+\boldsymbol{v_1}+\frac{\Delta t}{2}\boldsymbol{v_2}-\frac{\Delta t}{2}\boldsymbol{v_o}\tag{28}$$

further simplification yields:

$$\Big[\boldsymbol{M}+\frac{\Delta t^2}{6}(\boldsymbol{K}-\boldsymbol{C_o})\Big]\boldsymbol{\Delta U}=\Delta t\Big[\boldsymbol{C}-\frac{\Delta t}{2}\boldsymbol{K}-\boldsymbol{M_o}+\frac{\Delta t}{2}\boldsymbol{C_o}\Big]\boldsymbol{U}^n+\Delta t\Big[\frac{\Delta t}{2}(\boldsymbol{v_2}-\boldsymbol{v_o})+\boldsymbol{v_1}\Big]\tag{29}$$

and for ease of implementation in Matlab, it is written as:

$$\boldsymbol{A}\boldsymbol{\Delta U}=\boldsymbol{B}\boldsymbol{U}^n+\boldsymbol{f}$$

The corresponding part of the Matlab code is:

```
elseif meth == 3
    A = M +(dt^2/6)*(K - Co);
    B = dt*(C - (dt/2)*K - Mo + (dt/2)*Co);
    f = dt*((dt/2)*(v2 - vo) +v1);
```

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

The time discretization of two-step third-order Taylor-Galerkin method is given by:

$$\tilde{u}^n=u^n+\frac{1}{3}\Delta t u_t^n+\alpha\Delta t^2u_{tt}^n,\tag{30a}$$

$$u^{n+1}=u^n+\Delta t u_t^n+\frac{1}{2}\Delta t^2\tilde{u}_{tt}^n\tag{30b}$$

It is important to recall that:

$$u_t^n=s^n-\boldsymbol{a}\cdot\boldsymbol{\nabla}u^n\tag{31}$$

$$u_{tt}^n=s_t^n-\boldsymbol{a}\cdot\boldsymbol{\nabla}u_t^n=s_t^n-\boldsymbol{a}\cdot\boldsymbol{\nabla}s^n+(\boldsymbol{a}\cdot\boldsymbol{\nabla})^2u^n\tag{32}$$

and similarly:

$$\tilde{u}_{tt}^n=s_t^n-\boldsymbol{a}\cdot\boldsymbol{\nabla}s^n+(\boldsymbol{a}\cdot\boldsymbol{\nabla})^2\tilde{u}^n\tag{33}$$

Substituting (31) and (32) into (30a), and (31) and (33) into (30b) yields:

$$\tilde{u}^n = u^n + \frac{1}{3}\Delta t s^n - \frac{1}{3}\Delta t \boldsymbol{a}\cdot\boldsymbol{\nabla}u^n + \alpha\Delta t^2 s_t^n - \alpha\Delta t^2 \boldsymbol{a}\cdot\boldsymbol{\nabla}s^n + \alpha\Delta t^2(\boldsymbol{a}\cdot\boldsymbol{\nabla})^2 u^n \tag{34a}$$

$$u^{n+1} = u^n + \Delta t s^n - \Delta t \boldsymbol{a}\cdot\boldsymbol{\nabla}u^n + \frac{1}{2}\Delta t^2 s_t^n - \frac{1}{2}\Delta t^2 \boldsymbol{a}\cdot\boldsymbol{\nabla}s^n + \frac{1}{2}\Delta t^2(\boldsymbol{a}\cdot\boldsymbol{\nabla})^2\tilde{u}^n \tag{34b}$$

Assuming that the source term is not time dependent, then the Galerkin weak form is given by:

$$\big(w,\tilde{u}^n\big)_\Omega = \big(w,u^n\big)_\Omega + \frac{1}{3}\Delta t\big(w,s^n\big)_\Omega - \frac{1}{3}\Delta t\big(w,\boldsymbol{a}\cdot\boldsymbol{\nabla}u^n\big)_\Omega - \alpha\Delta t^2\big(w,\boldsymbol{a}\cdot\boldsymbol{\nabla}s^n\big)_\Omega + \alpha\Delta t^2\big(w,(\boldsymbol{a}\cdot\boldsymbol{\nabla})^2u^n\big)_\Omega \tag{35a}$$

$$\big(w,u^{n+1}\big)_\Omega = \big(w,u^n\big)_\Omega + \Delta t\big(w,s^n\big)_\Omega - \Delta t\big(w,\boldsymbol{a}\cdot\boldsymbol{\nabla}u^n\big)_\Omega - \frac{1}{2}\Delta t^2\big(w,\boldsymbol{a}\cdot\boldsymbol{\nabla}s^n\big)_\Omega + \frac{1}{2}\Delta t^2\big(w,(\boldsymbol{a}\cdot\boldsymbol{\nabla})^2\tilde{u}^n\big)_\Omega \tag{35b}$$

Integration by parts, and applying homogeneous Dirchlet boundary condition at the inlet yields:

$$\begin{aligned}\big(w,\tilde{u}^n\big)_\Omega = \big(w,u^n\big)_\Omega &+ \frac{1}{3}\Delta t\big(w,s^n\big)_\Omega + \frac{1}{3}\Delta t\big(\boldsymbol{a}\cdot\boldsymbol{\nabla}w,u^n\big)_\Omega + \alpha\Delta t^2\big(\boldsymbol{a}\cdot\boldsymbol{\nabla}w,s^n\big)_\Omega \\ &- \alpha\Delta t^2\big(\boldsymbol{a}\cdot\boldsymbol{\nabla}w,\boldsymbol{a}\cdot\boldsymbol{\nabla}u^n\big)_\Omega - \frac{1}{3}\Delta t\big((\boldsymbol{a}\cdot\boldsymbol{n})w,u^n\big)_{\Gamma^{out}} \\ &- \alpha\Delta t^2\big((\boldsymbol{a}\cdot\boldsymbol{n})w,s^n\big)_{\Gamma^{out}} + \alpha\Delta t^2\big((\boldsymbol{a}\cdot\boldsymbol{n})w,\boldsymbol{a}\cdot\boldsymbol{\nabla}u^n\big)_{\Gamma^{out}}\end{aligned} \tag{36a}$$

$$\begin{aligned}\big(w,u^{n+1}\big)_\Omega = \big(w,u^n\big)_\Omega &+ \Delta t\big(w,s^n\big)_\Omega + \Delta t\big(\boldsymbol{a}\cdot\boldsymbol{\nabla}w,u^n\big)_\Omega + \frac{1}{2}\Delta t^2\big(\boldsymbol{a}\cdot\boldsymbol{\nabla}w,s^n\big)_\Omega \\ &- \frac{1}{2}\Delta t^2\big(\boldsymbol{a}\cdot\boldsymbol{\nabla}w,\boldsymbol{a}\cdot\boldsymbol{\nabla}\tilde{u}^n\big)_\Omega - \Delta t\big((\boldsymbol{a}\cdot\boldsymbol{n})w,u^n\big)_{\Gamma^{out}} \\ &- \frac{1}{2}\Delta t^2\big((\boldsymbol{a}\cdot\boldsymbol{n})w,s^n\big)_{\Gamma^{out}} + \frac{1}{2}\Delta t^2\big((\boldsymbol{a}\cdot\boldsymbol{n})w,\boldsymbol{a}\cdot\boldsymbol{\nabla}\tilde{u}^n\big)_{\Gamma^{out}}\end{aligned} \tag{36b}$$

Using the previous definitions given by (13), the weak form given by (36) is written in discrete form as:

$$\boldsymbol{M}\widetilde{\boldsymbol{U}}^n = \boldsymbol{M}\boldsymbol{U}^n + \frac{1}{3}\Delta t\boldsymbol{v_1} + \frac{1}{3}\Delta t\boldsymbol{C}\boldsymbol{U}^n + \alpha\Delta t^2\boldsymbol{v_2} - \alpha\Delta t^2\boldsymbol{K}\boldsymbol{U}^n - \frac{1}{3}\Delta t\boldsymbol{M_o}\boldsymbol{U}^n - \alpha\Delta t^2\boldsymbol{v_o} + \alpha\Delta t^2\boldsymbol{C_o}\boldsymbol{U}^n, \tag{37a}$$

$$\boldsymbol{M}\boldsymbol{U}^{n+1} = \boldsymbol{M}\boldsymbol{U}^n + \Delta t\boldsymbol{v_1} + \Delta t\boldsymbol{C}\boldsymbol{U}^n + \frac{1}{2}\Delta t^2\boldsymbol{v_2} - \frac{1}{2}\Delta t^2\boldsymbol{K}\widetilde{\boldsymbol{U}}^n - \Delta t\boldsymbol{M_o}\boldsymbol{U}^n - \frac{1}{2}\Delta t^2\boldsymbol{v_o} + \frac{1}{2}\Delta t^2\boldsymbol{C_o}\widetilde{\boldsymbol{U}}^n \tag{37b}$$

further simplification yields:

$$\boldsymbol{M}(\widetilde{\boldsymbol{U}}^n - \boldsymbol{U}^n) = \Big[\frac{\Delta t}{3}(\boldsymbol{C}-\boldsymbol{M_o}) - \alpha\Delta t^2(\boldsymbol{K}-\boldsymbol{C_o})\Big]\boldsymbol{U}^n + \Big[\frac{\Delta t}{3}\boldsymbol{v_1} + \alpha\Delta t^2(\boldsymbol{v_2}-\boldsymbol{v_o})\Big], \tag{38a}$$

$$\boldsymbol{M}(\boldsymbol{U}^{n+1} - \boldsymbol{U}^n) = \Big[\Delta t(\boldsymbol{C}-\boldsymbol{M_o})\Big]\boldsymbol{U}^n - \Big[\alpha\Delta t^2(\boldsymbol{K}-\boldsymbol{C_o})\Big]\widetilde{\boldsymbol{U}}^n + \Big[\Delta t\boldsymbol{v_1} + \frac{\Delta t^2}{2}(\boldsymbol{v_2}-\boldsymbol{v_o})\Big] \tag{38b}$$

and for ease of implementation in Matlab, it is written as:

$$\boldsymbol{A}_1(\widetilde{\boldsymbol{U}}^n - \boldsymbol{U}^n) = \boldsymbol{B}_1\boldsymbol{U}^n + \boldsymbol{f}_1, \tag{39a}$$

$$\boldsymbol{A}_2(\boldsymbol{U}^{n+1} - \boldsymbol{U}^n) = \boldsymbol{B}_2\boldsymbol{U}^n + \boldsymbol{C}_2\widetilde{\boldsymbol{U}}^n + \boldsymbol{f}_2 \tag{39b}$$

The corresponding part of the Matlab code is:

```
elseif meth == 7
    alpha = 1/9;
    A1 = M;
    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);
```

In the original code, the third term in the RHS of equation (35) wasn't integrated by parts. Both variations are valid and they give exactly the same results.

2.5 Fourth-order Taylor-Galerkin (TG4) + Galerkin formulation

The time discretization of fourth-order Taylor-Galerkin method is given by:

$$\frac{\Delta u}{\Delta t} = \frac{1}{2}(u_t^{n+1} + u_t^n) - \frac{\Delta t}{12}(u_{tt}^{n+1} - u_{tt}^n) \tag{40}$$

Substituting (31) and (32) into (40) yields:

$$\frac{\Delta u}{\Delta t} = \frac{1}{2}(s^{n+1} - \boldsymbol{a}\cdot\boldsymbol{\nabla}u^{n+1} + s^n - \boldsymbol{a}\cdot\boldsymbol{\nabla}u^n) - \frac{\Delta t}{12}(s_t^{n+1} - \boldsymbol{a}\cdot\boldsymbol{\nabla}s^{n+1} + (\boldsymbol{a}\cdot\boldsymbol{\nabla})^2u^{n+1} - s_t^n + \boldsymbol{a}\cdot\boldsymbol{\nabla}s^n - (\boldsymbol{a}\cdot\boldsymbol{\nabla})^2u^n) \tag{41}$$

Assuming that the source term is not time dependent, i.e. $s_t = 0$ and $s^{n+1} = s^n$, simplifies the equation into:

$$\frac{\Delta u}{\Delta t} = s - \frac{1}{2}\boldsymbol{a}\cdot\boldsymbol{\nabla}u^{n+1} - \frac{1}{2}\boldsymbol{a}\cdot\boldsymbol{\nabla}u^n - \frac{\Delta t}{12}(\boldsymbol{a}\cdot\boldsymbol{\nabla})^2\Delta u \tag{42}$$

Re-writing the third term on the RHS yields:

$$\frac{\Delta u}{\Delta t} = s - \frac{1}{2}\boldsymbol{a}\cdot\boldsymbol{\nabla}u^{n+1} + \frac{1}{2}\boldsymbol{a}\cdot\boldsymbol{\nabla}u^n - \boldsymbol{a}\cdot\boldsymbol{\nabla}u^n - \frac{\Delta t}{12}(\boldsymbol{a}\cdot\boldsymbol{\nabla})^2\Delta u \tag{43}$$

Gathering the 2nd and third term on the RHS yields:

$$\frac{\Delta u}{\Delta t} = s - \frac{1}{2}\boldsymbol{a}\cdot\boldsymbol{\nabla}\Delta u - \boldsymbol{a}\cdot\boldsymbol{\nabla}u^n - \frac{\Delta t}{12}(\boldsymbol{a}\cdot\boldsymbol{\nabla})^2\Delta u \tag{44}$$

Re-arranging:

$$\frac{\Delta u}{\Delta t} + \frac{1}{2}\boldsymbol{a}\cdot\boldsymbol{\nabla}\Delta u + \frac{\Delta t}{12}(\boldsymbol{a}\cdot\boldsymbol{\nabla})^2\Delta u = s - \boldsymbol{a}\cdot\boldsymbol{\nabla}u^n \tag{45}$$

The Galerkin weak form is given by:

$$\left(w, \frac{\Delta u}{\Delta t}\right)_\Omega + \frac{1}{2}\left(w, \boldsymbol{a}\cdot\boldsymbol{\nabla}\Delta u\right)_\Omega + \frac{\Delta t}{12}\left(w, (\boldsymbol{a}\cdot\boldsymbol{\nabla})^2\Delta u\right)_\Omega = \left(w, s\right)_\Omega - \left(w, \boldsymbol{a}\cdot\boldsymbol{\nabla}u^n\right)_\Omega \tag{46}$$

Integration by parts for the third term on the LHS, and considering only Dirchlet boundary conditions at the inlet boundary yields:

$$\begin{aligned}\left(w, \frac{\Delta u}{\Delta t}\right)_\Omega + \frac{1}{2}\left(w, \boldsymbol{a}\cdot\boldsymbol{\nabla}\Delta u\right)_\Omega - \frac{\Delta t}{12}\left(\boldsymbol{a}\cdot\boldsymbol{\nabla}w, \boldsymbol{a}\cdot\boldsymbol{\nabla}\Delta u\right)_\Omega + \frac{\Delta t}{12}\left((\boldsymbol{a}\cdot\boldsymbol{n})w, \boldsymbol{a}\cdot\boldsymbol{\nabla}\Delta u\right)_{\Gamma^{out}} \\ = \left(w, s\right)_\Omega - \left(w, \boldsymbol{a}\cdot\boldsymbol{\nabla}u^n\right)_\Omega\end{aligned} \tag{47}$$

Using the previous definitions given by (13), the weak form given by (47) is written in discrete form as:

$$\frac{1}{\Delta t}\boldsymbol{M}\boldsymbol{\Delta U} + \frac{1}{2}\boldsymbol{C}\boldsymbol{\Delta U} - \frac{\Delta t}{12}\boldsymbol{K}\boldsymbol{\Delta U} + \frac{\Delta t}{12}\boldsymbol{C_o}\boldsymbol{\Delta U} = \boldsymbol{v_1} - \boldsymbol{C}\boldsymbol{U}^n \tag{48}$$

further simplification yields:

$$\left[\boldsymbol{M} + \frac{\Delta t}{2}\boldsymbol{C} - \frac{\Delta t^2}{12}(\boldsymbol{K} - \boldsymbol{C_o})\right]\boldsymbol{\Delta U} = -\Delta t\boldsymbol{C}\boldsymbol{U}^n + \Delta t\boldsymbol{v_1} \tag{49}$$

and for ease of implementation in Matlab, it is written as:

$$\boldsymbol{A}\boldsymbol{\Delta U} = \boldsymbol{B}\boldsymbol{U}^n + \boldsymbol{f}$$

The corresponding part of the Matlab code is:

```
elseif meth == 8
    A = M +(dt/2)*C' - (dt^2/12)*(K - Co);
    B = -dt*C';
    f = dt*v1;
```

2.6 Two-step fourth-order Taylor-Galerkin (TG4-2S) + Galerkin formulation

By changing the value of α in (30) to 1/12, the two-step fourth-order Taylor-Galerkin time discretization is obtained.

```
elseif meth == 9
alpha = 1/12;
    A1 = M;
    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);
```

2.7 Numerical results of the rotating cone problem

The rotating cone problem presented in [1] is solved on a uniform mesh of 30×30 linear quadrilateral elements over the unit square $[\frac{-1}{2}, \frac{1}{2}] \times [\frac{-1}{2}, \frac{1}{2}]$. The numerical solution, using the different methods presented earlier, is obtained after a complete revolution of the cone, i.e. $t_{end} = 2\pi$. For the explicit methods, 200 time steps ($\Delta t = 2\pi/200$) are used, while for the implicit Crank-Nicholson method, three increasing values of the time step are employed ($\Delta t = 2\pi/120$, $2\pi/60$ and $2\pi/30$).

To compare the accuracy of the explicit methods (Lax-Wendroff/Lumped mass LW-FD, Lax-Wendroff/Consistent mass TG2, TG3, TG3-2S, TG4, TG4-2S), the maximum and minimum values of the computed solution are recorded in Table 1. Similarly, for the implicit Crank-Nicholson method, the maximum and minimum values of the computed solution are recorded in Table 1 using different values of time step.

It is noticed in Figure 10 that the TG2 and TG3 methods are more accurate than the LW-FD method. However, TG2 and TG3 uses consistent mass matrices which makes them computationally more expensive than LW-FD that has a diagonal mass matrix. The results using TG3-2S and the fourth-order methods TG4 and TG4-2S are shown in Figure 11. The high accuracy of the solutions is clearly apparent.

For the implicit Crank-Nicholson method which is unconditionally stable, the solutions obtained, using different values of the time step, are shown in Figure 3d. Higher values of Δt are used to check the behaviour of the method beyond the stability limit of the explicit methods. It is noticed that the method is not dissipative. However, the accuracy of the method decreases by increasing Δt which can be interpreted from the higher non-physical oscillations.

Method	Δt	u_{max}	u_{min}	**Figure**
LW-FD	$2\pi/200$	0.818575	-0.177432	Figure 10 (Top)
TG2	$2\pi/200$	0.983041	-0.018619	Figure 10 (Middle)
TG3	$2\pi/200$	0.983465	-0.014839	Figure 10 (Bottom)
TG3-2S	$2\pi/200$	0.982845	-0.014939	Figure 11 (Top)
TG4	$2\pi/200$	0.992350	-0.017285	Figure 11 (Middle)
TG4-2S	$2\pi/200$	0.982536	-0.014973	Figure 11 (Bottom)
CN	$2\pi/120$	0.996931	-0.045350	Figure 12 (Top)
CN	$2\pi/60$	0.969116	-0.109591	Figure 12 (Middle)
CN	$2\pi/30$	0.889308	-0.269427	Figure 12 (Bottom)

Table 1: The maximum and minimum values of the computed solution after one complete revolution using different methods.

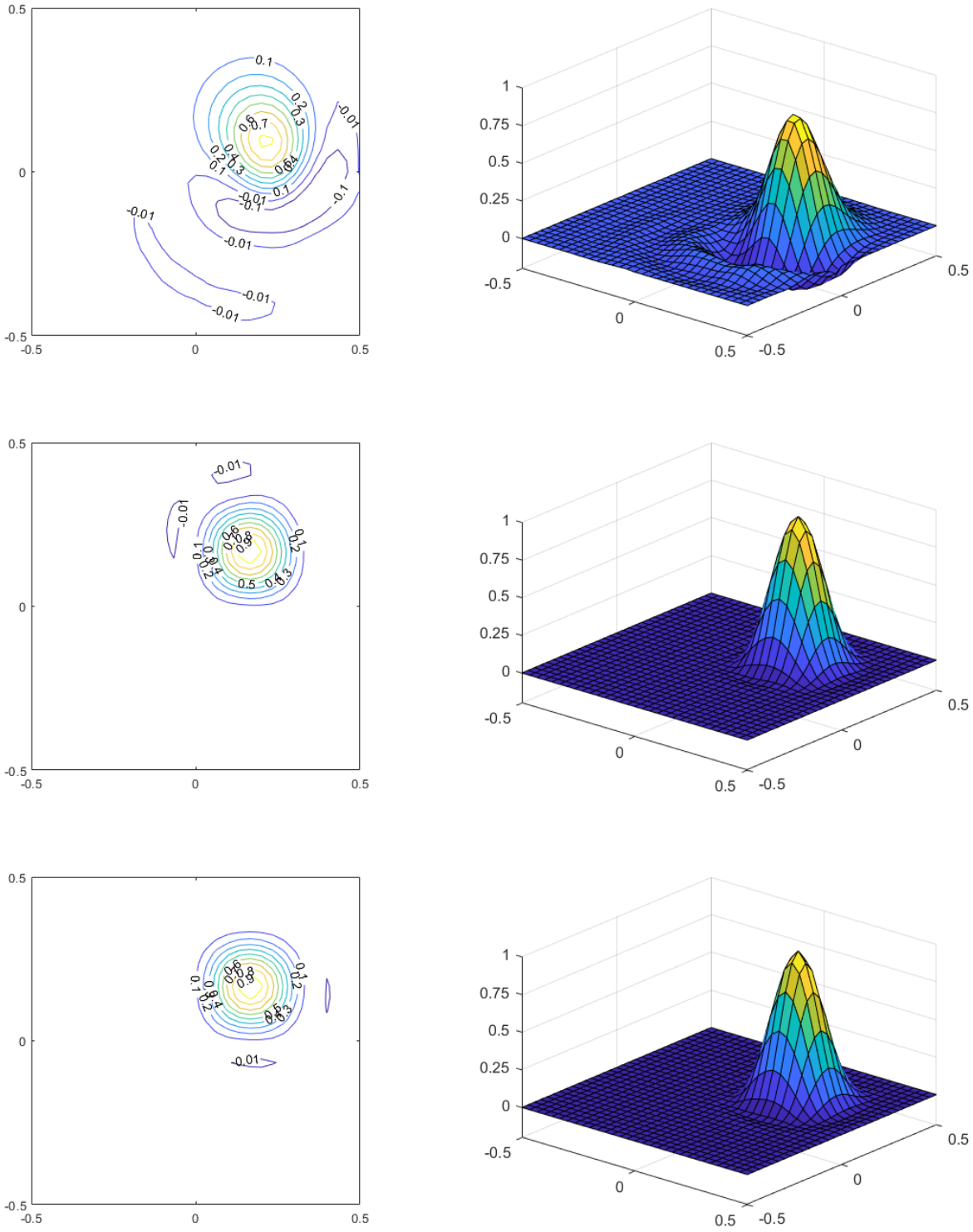


Figure 10: Convection of a cosine hill in a pure rotation velocity field: comparison of the numerical solutions after a complete revolution calculated with $\Delta t = 2\pi/200$ by means of (Top): Lax-Wendroff/Lumped mass matrix ($u_{max} = 0.818575, u_{min} = -0.177432$); (Middle): Lax-Wendroff/Consistent mass matrix ($u_{max} = 0.983041, u_{min} = -0.018619$); and (Bottom): TG3 ($u_{max} = 0.983465, u_{min} = -0.014839$).

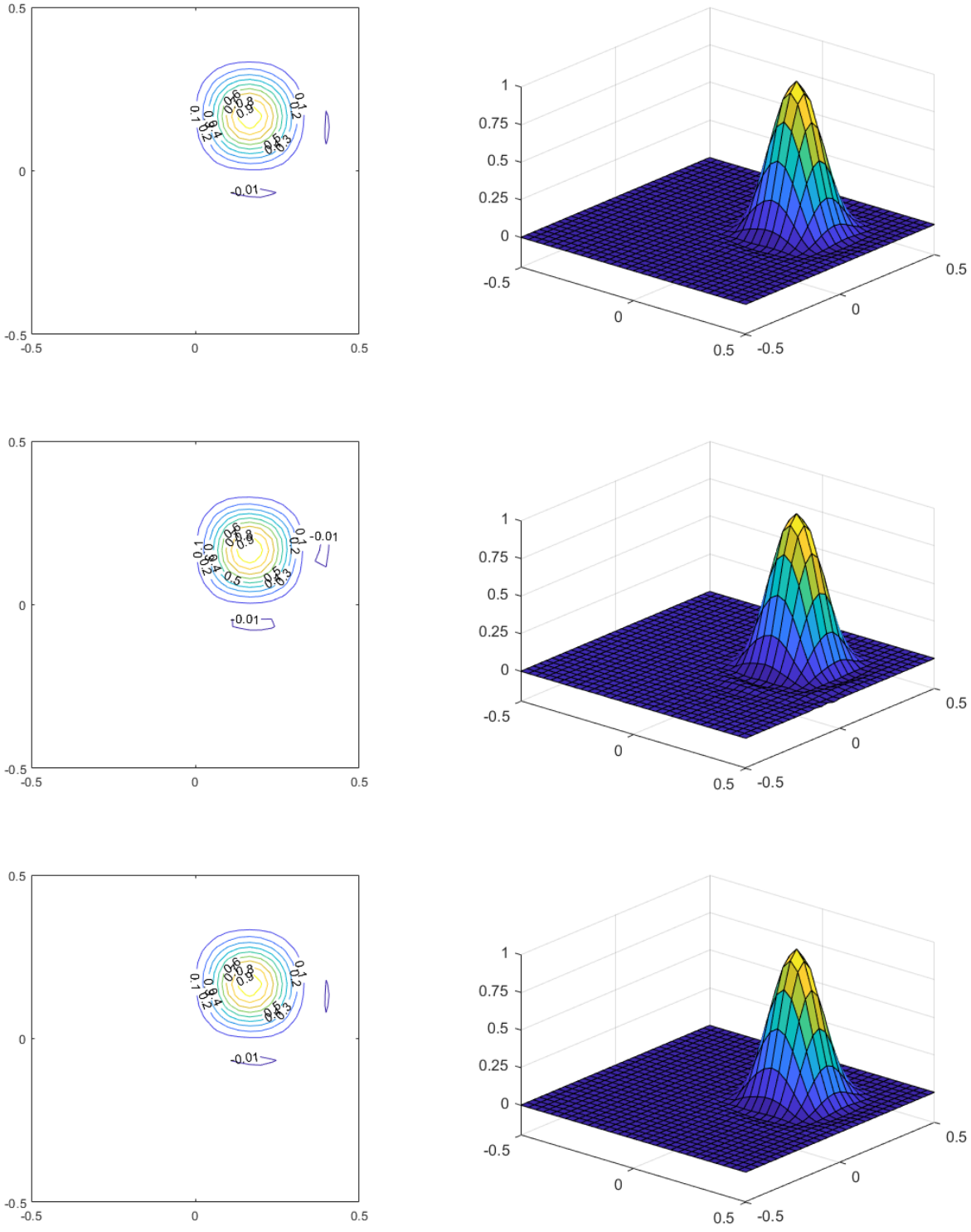


Figure 11: Convection of a cosine hill in a pure rotation velocity field: comparison of the numerical solutions after a complete revolution calculated with $\Delta t = 2\pi/200$ by means of (Top): TG3-2S ($u_{max} = 0.982845, u_{min} = -0.014939$); (Middle): TG4 ($u_{max} = 0.992350, u_{min} = -0.017285$); and (Bottom): TG4-2S ($u_{max} = 0.982536, u_{min} = -0.0124973$).

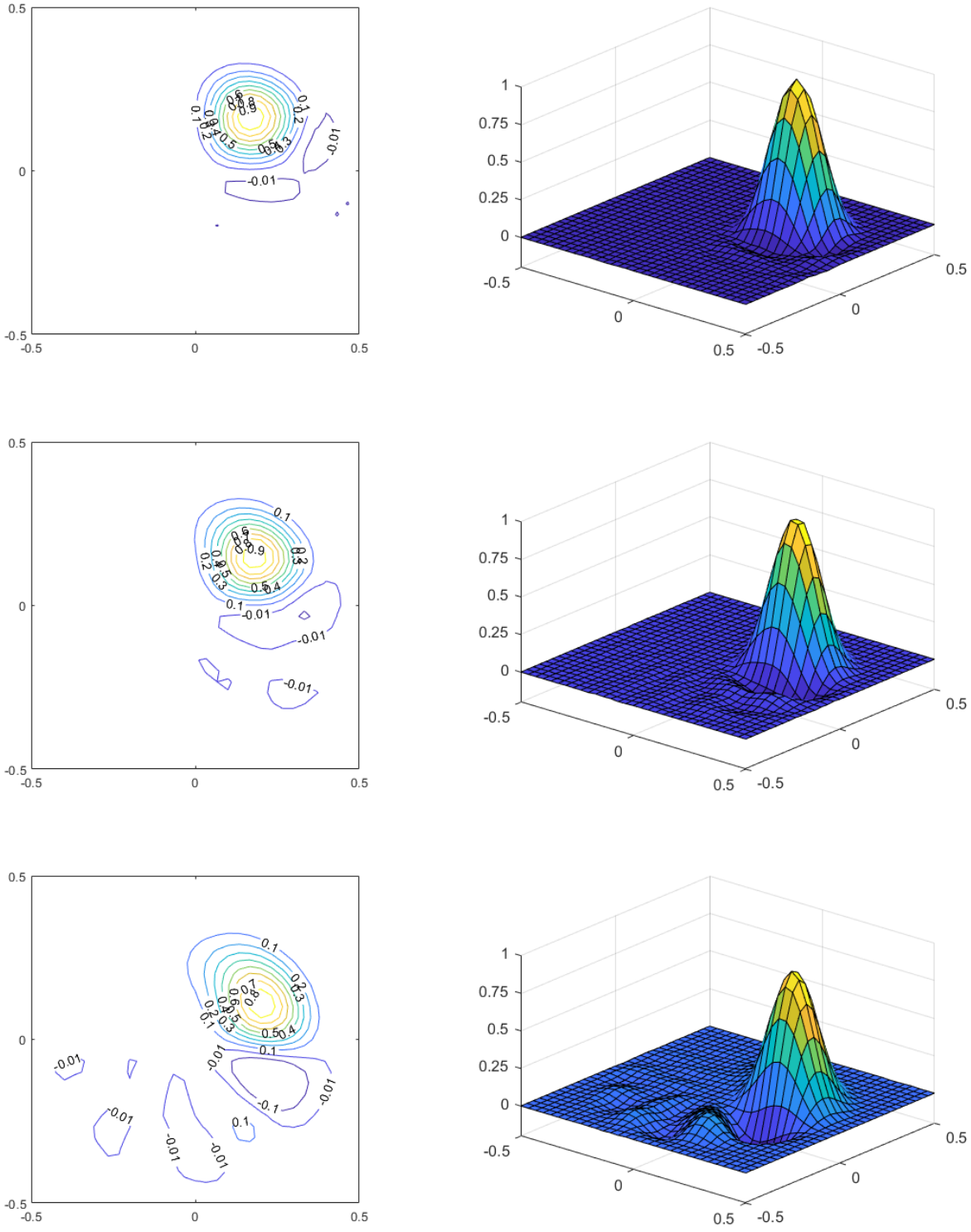


Figure 12: Convection of a cosine hill in a pure rotation velocity field using the Crank-Nicholson/Galerkin method: comparison of the numerical solutions after a complete revolution calculated with (Top): $\Delta t = 2\pi/120, u_{max} = 0.996931, u_{min} = -0.045350$; (Middle): $\Delta t = 2\pi/60, u_{max} = 0.969116, u_{min} = -0.109591$; and (Bottom): $\Delta t = 2\pi/30, u_{max} = 0.889308, u_{min} = -0.269427$.

A Developed codes

A.1 Developed codes for 2D steady transport equation

A.1.1 Modified `main.m`

The modifications were made to account for linear and quadratic elements, and also to account for zero Dirchlet boundary conditions at the outlet boundary.

```
% This program solves a convection-diffusion problem
% in a square domain [0,1]x[0,1]

clear; close all; clc

disp(' ')
disp('This program solves a convection-diffusion equation on [0,1]x[0,1]')
disp(' ')
disp('No source term is considered');

% PDE coefficients
disp('Convection velocity is');
a_magnitude = 10^-3;
velo = [cos(pi/6),sin(pi/6)];
disp(velo);
a = norm(velo);
nu = cinput('Diffusion coefficient', 0.0001);
sigma = cinput('Reaction coefficient sigma', 1);
disp(' ')

dom = [0,1,0,1];

example.dom = dom;
example.velo = velo;
example.a = a;
example.nu = nu;
example.sigma = sigma;

% Element type
elem = 1; p = 1;
referenceElement = SetReferenceElement(elem,p);

nx = cinput('Number of elements in each direction (multiple of 5)= ', 10);
ny = nx;
[X,T] = CreateMesh(dom,nx,ny,referenceElement);
PlotMesh(T,X,'b-');

h = 1/nx;
Pe = a*h/(2*nu);
disp(' ')
disp(strcat('Peclet number: ',num2str(Pe)))

disp(' ')
disp ('The following BCs at the outlet can be used:');
disp ('     [1] Homogeneous natural condition ');
disp ('     [2] zero Dirchlet condition');
```

```
47 disp ('     [3] zero Dirchlet condition everywhere (for this a source ...
       term s=1 is used)');
48 iOutletBC = cinput('Choose a BC at the outlet: ', 3);
49
50 disp(' ')
51 disp ('The following methods can be used:');
52 disp ('     [0] Galerkin');
53 disp ('     [1] Artificial diffusion');
54 disp ('     [2] SUPG');
55 disp ('     [3] GLS');
56 method = input ('Choose a method for solving the problem: ');
57 example.method = method;
58
59 % SYSTEM RESULTING OF DISCRETIZING THE WEAK FORM
60 [K,f] = FEM_system(X,T,referenceElement,example);
61
62 % BOUNDARY CONDITIONS
63 % Boundary conditions are imposed using Lagrange multipliers
64 nodes_x0 = find(abs(X(:,1)) < 10e-6);
65 nodes_x0 = nodes_x0(2:end-1);
66 nodes_x1 = find(abs(X(:,1)-1) < 10e-6);
67 nodes_x1 = nodes_x1(2:end);
68 nodes_y0 = find(abs(X(:,2)) < 10e-6);
69 nodes_y1 = find(abs(X(:,2)-1) < 10e-6);
70 nodes_y1 = nodes_y1(1:end-1);
71
72 if iOutletBC == 1
73     % nodes on which solution is u=1
74     nodesDir1 = nodes_x0( X(nodes_x0,2) > 0.2 );
75     % nodes on which solution is u=0
76     nodesDir0 = [nodes_x0( X(nodes_x0,2) ≤ 0.2 ); nodes_y0];
77     % Boundary condition matrix
78     C = [nodesDir1, ones(length(nodesDir1),1);
79          nodesDir0, zeros(length(nodesDir0),1)];
80 elseif iOutletBC == 2
81     % nodes on which solution is u=1
82     nodesDir1 = nodes_x0( X(nodes_x0,2) > 0.2 );
83     % nodes on which solution is u=0
84     nodesDir0 = [nodes_x0( X(nodes_x0,2) ≤ 0.2 ); nodes_y0; nodes_x1; ...
         nodes_y1];
85     % Boundary condition matrix
86     C = [nodesDir1, ones(length(nodesDir1),1);
87          nodesDir0, zeros(length(nodesDir0),1)];
88 elseif iOutletBC == 3
89     % nodes on which solution is u=0
90     nodesDir0 = [nodes_x0; nodes_y0; nodes_x1; nodes_y1];
91     % Boundary condition matrix
92     C = [nodesDir0, zeros(length(nodesDir0),1)];
93 end
94 nDir = size(C,1);
95 neq  = size(f,1);
96 A = zeros(nDir,neq);
97 A(:,C(:,1)) = eye(nDir);
98 b = C(:,2);
99
100 % SOLUTION OF THE LINEAR SYSTEM
101 % Entire matrix
102 Ktot = [K A';A zeros(nDir,nDir)];
```

```
ftot = [f;b];

sol = Ktot\ftot;
Temp = sol(1:neq);
multip = sol(neq+1:end);

% POSTPROCESS
figure(2), clf
xx  = reshape(X(:,1), p*ny+1, p*nx+1)';
yy  = reshape(X(:,2), p*ny+1, p*nx+1)';
sol = reshape(Temp, p*ny+1, p*nx+1)';
surface(xx,yy,sol,'FaceColor','interp');
set(gca, 'xTick',0:0.25:1, 'yTick',0:0.25:1, 'FontSize',12)
xlabel('x','FontSize',14);
ylabel('y','FontSize',14);
zlabel('u','FontSize',14);
zlim([-0.5 1.5])
% zlim([0 2])
colormap jet
grid on; view(3)
```

A.1.2 Modified `ShapeFunc.m`

The modifications were made to account for linear and quadratic elements. The second derivatives of the shape functions are added since they are needed in GLS formulation.

```
function [N,Nxi,Neta,N2xi,N2eta,N2xieta] = ShapeFunc(elem,p,z)
% [N,Nxi,Neta] = ShapeFunc(elem,z)
% N, Nxi, Neta: matrices storing the values of the shape functions on the ...
    Gauss points
%               of the reference element
%               Each row concerns to a Gauss point
% z: coordinates of Gauss points in the reference element
% elem:     type of element (0: quadrilatera, 1: triangles)
% p: interpolation degree

xi = z(:,1); eta = z(:,2); ngauss = length(z);
if elem == 0
    if p == 1
    N = [(1-xi).*(1-eta)/4, (1+xi).*(1-eta)/4, (1+xi).*(1+eta)/4, ...
        (1-xi).*(1+eta)/4];
    Nxi  = [(eta-1)/4, (1-eta)/4, (1+eta)/4, -(1+eta)/4];
    Neta = [(xi-1)/4, -(1+xi)/4,   (1+xi)/4,  (1-xi)/4 ];
    N2xi = zeros(ngauss,4);
    N2eta = zeros(ngauss,4);
    N2xieta = 1/4*[ones(size(xi)), -ones(size(xi)), ones(size(xi)), ...
        -ones(size(xi))];
    elseif p == 2
    N    = [xi.*(xi-1).*eta.*(eta-1)/4, xi.*(xi+1).*eta.*(eta-1)/4, ...
            xi.*(xi+1).*eta.*(eta+1)/4, xi.*(xi-1).*eta.*(eta+1)/4, ...
            (1-xi.^2).*eta.*(eta-1)/2,  xi.*(xi+1).*(1-eta.^2)/2,   ...
            (1-xi.^2).*eta.*(eta+1)/2,  xi.*(xi-1).*(1-eta.^2)/2,   ...
            (1-xi.^2).*(1-eta.^2)];
    Nxi  = [(xi-1/2).*eta.*(eta-1)/2,   (xi+1/2).*eta.*(eta-1)/2, ...
            (xi+1/2).*eta.*(eta+1)/2,   (xi-1/2).*eta.*(eta+1)/2, ...
```

```
            -xi.*eta.*(eta-1),          (xi+1/2).*(1-eta.^2),     ...
            -xi.*eta.*(eta+1),          (xi-1/2).*(1-eta.^2),     ...
            -2*xi.*(1-eta.^2)];
    Neta = [xi.*(xi-1).*(eta-1/2)/2,    xi.*(xi+1).*(eta-1/2)/2, ...
            xi.*(xi+1).*(eta+1/2)/2,    xi.*(xi-1).*(eta+1/2)/2, ...
            (1-xi.^2).*(eta-1/2),       xi.*(xi+1).*(-eta),      ...
            (1-xi.^2).*(eta+1/2),       xi.*(xi-1).*(-eta),      ...
            (1-xi.^2).*(-2*eta)];
    N2xi = [ eta.*(eta - 1)/2,    eta.*(eta - 1)/2, ...
             eta.*(eta + 1)/2,    eta.*(eta + 1)/2, ...
             -eta.*(eta - 1),     1 - eta.^2,       ...
             -eta.*(eta + 1),     1 - eta.^2,       ...
             2*eta.^2 - 2];
    N2eta = [ xi.*(xi - 1)/2, xi.*(xi + 1)/2, ...
              xi.*(xi + 1)/2, xi.*(xi - 1)/2, ...
              1 - xi.^2, -xi.*(xi + 1),       ...
              1 - xi.^2, -xi.*(xi - 1),       ...
              2*xi.^2 - 2];
    N2xieta = [ (eta.*(xi - 1/2))/2 + ((eta - 1).*(xi - 1/2))/2, ...
                (eta.*(xi + 1/2))/2 + ((eta - 1).*(xi + 1/2))/2, ...
                (eta.*(xi + 1/2))/2 + ((eta + 1).*(xi + 1/2))/2, ...
                (eta.*(xi - 1/2))/2 + ((eta + 1).*(xi - 1/2))/2, ...
                - eta.*xi - xi.*(eta - 1),  -2*eta.*(xi + 1/2), ...
                - eta.*xi - xi.*(eta + 1),  -2*eta.*(xi - 1/2), ...
                4*eta.*xi];
    else
    error('not available interpolation degree')
    end
elseif elem == 1
    if p == 1
        N = [1-xi-eta, xi, eta];
        Nxi = [-ones(size(xi)), ones(size(xi)), zeros(size(xi))];
        Neta = [-ones(size(xi)), zeros(size(xi)), ones(size(xi))];
        N2xi = zeros(ngauss,3);
        N2eta = zeros(ngauss,3);
        N2xieta = zeros(ngauss,3);
    elseif p == 2
        N = [2*(xi+eta).^2-3*(xi+eta)+1,  2*xi.^2-xi, ...
             2*eta.^2-eta,  -4*xi.^2 - 4*xi.*eta+4*xi, ...
             4*xi.*eta,  -4*eta.^2-4*xi.*eta + 4*eta];
        Nxi = [4*eta + 4*xi - 3,  4*xi - 1,  zeros(size(xi)), ...
               4 - 8*xi - 4*eta, 4*eta, -4*eta];
        Neta = [ 4*eta + 4*xi - 3, zeros(size(xi)), 4*eta - 1, ...
                 -4*xi,  4*xi,  4 - 4*xi - 8*eta];
        N2xi = [ 4*ones(size(xi)), 4*ones(size(xi)), zeros(size(xi)), ...
                 -8*ones(size(xi)), zeros(size(xi)), zeros(size(xi))];
        N2eta = [ 4*ones(size(xi)), zeros(size(xi)), 4*ones(size(xi)), ...
                  zeros(size(xi)), zeros(size(xi)), -8*ones(size(xi))];
        N2xieta = [ 4*ones(size(xi)), zeros(size(xi)), zeros(size(xi)), ...
                    -4*ones(size(xi)), 4*ones(size(xi)), -4*ones(size(xi))];
    else
        error('not available interpolation degree')
    end
else
    error('not available element')
end
```

A.1.3 Modified `FEMsystem.m`

The GLS method is added to the code where the stabilization parameter τ is computed in (lines 66 to 74) and the elemental matrices are computed in (lines 159 to 166). Another main addition to this function is the mapping of the second order derivatives from the reference element to the physical one which is added in (lines 104 to 139). The details of mapping of second order derivatives are shown in appendix B.

```
1  function [K,f] = FEM_system(X,T,referenceElement,example)
2  % [K,f] = FEM_system(X,T,referenceElement,example)
3  % Matrix K and r.h.s vector f obtained after discretizong a 2D ...
       convection-diffusion equation
4  %
5  % X:             nodal coordinates
6  % T:             connectivities (elements)
7  % referenceElement: reference element properties (quadrature, shape ...
       functions...)
8  % example: example properties
9
10 velo = example.velo;
11 a = example.a;
12 nu = example.nu;
13 sigma = example.sigma;
14 method = example.method;
15 ax = velo(1);
16 ay = velo(2);
17 Te = T(1,:); Xe = X(Te,:);
18 hx = max(Xe(:,1)) - min(Xe(:,1));
19 hy = max(Xe(:,2)) - min(Xe(:,2));
20 h = (hx+hy)/2;
21
22 nen = referenceElement.nen;
23 ngaus = referenceElement.ngaus;
24 wgp = referenceElement.GaussWeights;
25 N = referenceElement.N;
26 Nxi = referenceElement.Nxi;
27 Neta = referenceElement.Neta;
28 N2xi = referenceElement.N2xi;
29 N2eta = referenceElement.N2eta;
30 N2xieta = referenceElement.N2xieta;
31
32 % Number of elements and number of nodes in the mesh
33 nElem = size(T,1);
34 nPt = size(X,1);
35
36 K = zeros(nPt,nPt);
37 f = zeros(nPt,1);
38
39 if method == 0
40     % Galerkin
41     tau = 0;
42 elseif method == 1
43     % Artificial Diffusion
44     Pe_x = ax*h/(2*nu);
45     Pe_y = ay*h/(2*nu);
46     alpha_x = coth(Pe_x)-1/Pe_x;
```

```
47      alpha_y = coth(Pe_y)-1/Pe_y;
48      tau = h*(ax*alpha_x + ay*alpha_y)/2;
49      nubar_p = tau*a*a;
50      disp(strcat('Recommended value for the artificial diffusion = ...
            ',num2str(nubar_p)));
51      nubar = cinput('Artificial diffusion to be used', nubar_p);
52      if isempty(nubar)
53          nubar = nubar_p;
54      end
55      nu = nu + nubar;
56      tau=0;
57  elseif method == 2
58      % SUPG
59      Pe = a*h/(2*nu);
60      tau_p = h*(1 + 9/Pe^2)^(-1/2)/(2*a);
61      disp(strcat('Recommended stabilization parameter = ',num2str(tau_p)));
62      tau = cinput('Stabilization parameter',tau_p);
63      if isempty(tau)
64          tau = tau_p;
65      end
66  elseif method == 3
67      % GLS
68      Pe = a*h/(2*nu);
69      tau_p = h*(1 + 9/Pe^2)^(-1/2)/(2*a);
70      disp(strcat('Recommended stabilization parameter = ',num2str(tau_p)));
71      tau = cinput('Stabilization parameter',tau_p);
72      if isempty(tau)
73          tau = tau_p;
74      end
75  else
76      error ('Unavailable method')
77  end
78
79  % Loop on elements
80  for ielem = 1:nElem
81      % Te: global number of the nodes in the current element
82      Te = T(ielem,:);
83      % Xe: coordinates of the nodes in the current element
84      Xe = X(Te,:);
85      % Element matrices
86      [Ke,fe] = ...
            EleMat(Xe,nen,ngaus,wgp,N,Nxi,Neta,N2xi,N2eta,N2xieta,method,tau,velo,nu,sigma);
87      % Assemble the element matrices
88      K(Te,Te) = K(Te,Te) + Ke;
89      f(Te) = f(Te) + fe;
90  end
91
92
93
94  function [Ke,fe] = ...
        EleMat(Xe,nen,ngaus,wgp,N,Nxi,Neta,N2xi,N2eta,N2xieta,method,tau,velo,nu,sigma)
95  %
96  ax = velo(1);
97  ay = velo(2);
98
99  Ke = zeros(nen,nen);
100 fe = zeros(nen,1);
101
```

```
% Loop on Gauss points
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,:);
    if method == 0
        % Galerkin
        Ke = Ke + (nu*(Nx'*Nx+Ny'*Ny) + N_ig'*(ax*Nx+ay*Ny) + ...
            N_ig'*sigma*N_ig)*dvolu;
        aux = N_ig*Xe;
        f_ig = SourceTerm(aux);
        fe = fe + N_ig'*(f_ig*dvolu);
    elseif method == 1
        % Artifficial diffusion
        Ke = Ke + (nu*(Nx'*Nx+Ny'*Ny) + N_ig'*(ax*Nx+ay*Ny) + ...
            N_ig'*sigma*N_ig)*dvolu;
        aux = N_ig*Xe;
        f_ig = SourceTerm(aux);
        fe = fe + N_ig'*(f_ig*dvolu);
    elseif method == 2
        % SUPG
        Ke = Ke + ( nu*(Nx'*Nx+Ny'*Ny) + N_ig'*(ax*Nx+ay*Ny) + ...
            N_ig'*sigma*N_ig + ...
             tau*(ax*Nx+ay*Ny)'*( (ax*Nx+ay*Ny) - nu*(N2x+N2y) + ...
```

```
                    sigma*N_ig) )*dvolu;
156         aux = N_ig*Xe;
157         f_ig = SourceTerm(aux);
158         fe = fe + (N_ig + tau*(ax*Nx+ay*Ny))'*(f_ig*dvolu);
159     elseif method == 3
160         % GLS
161         Ke = Ke + (N_ig'*(ax*Nx+ay*Ny) + nu*(Nx'*Nx+Ny'*Ny) + ...
                N_ig'*sigma*N_ig + ...
162             tau*((ax*Nx+ay*Ny) - nu*(N2x+N2y) + sigma*N_ig)'*...
163             ((ax*Nx+ay*Ny) - nu*(N2x+N2y) + sigma*N_ig))*dvolu;
164         aux = N_ig*Xe;
165         f_ig = SourceTerm(aux);
166         fe = fe + (N_ig + tau*((ax*Nx+ay*Ny) - nu*(N2x+N2y) + sigma*N_ig ...
                ))'*(f_ig*dvolu);
167     end
168 end
```

A.2 Developed codes for 2D unsteady transport equation

Two higher-order methods are added to the code, which are TG4 and TG4-2S.

```
elseif meth == 8    % TG4
    A = M +(dt/2)*C' - (dt^2/12)*(K - Co);
    B = -dt*C';
    f = dt*v1;
elseif meth == 9    % TG4-2S
    alpha = 1/12;
    A1 = M;
    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);
```

B Mapping of second order derivatives from the reference element to the physical one

For 2D elements, the first order derivatives of the shape functions in the reference element can be written using chain-rule as:

$$\begin{aligned}\frac{\partial \boldsymbol{N}}{\partial \xi} &= \frac{\partial \boldsymbol{N}}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial \boldsymbol{N}}{\partial y}\frac{\partial y}{\partial \xi},\\ \frac{\partial \boldsymbol{N}}{\partial \eta} &= \frac{\partial \boldsymbol{N}}{\partial x}\frac{\partial x}{\partial \eta} + \frac{\partial \boldsymbol{N}}{\partial y}\frac{\partial y}{\partial \eta}.\end{aligned} \tag{B.1}$$

To derive an expression for a second derivative, consider for instance:

$$\begin{aligned}\frac{\partial^2 \boldsymbol{N}}{\partial \xi^2} &= \frac{\partial}{\partial \xi}\left(\frac{\partial \boldsymbol{N}}{\partial \xi}\right)\\ &= \frac{\partial}{\partial \xi}\left(\frac{\partial \boldsymbol{N}}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial \boldsymbol{N}}{\partial y}\frac{\partial y}{\partial \xi}\right)\\ &= \frac{\partial}{\partial \xi}\left(\frac{\partial \boldsymbol{N}}{\partial x}\right)\frac{\partial x}{\partial \xi} + \frac{\partial \boldsymbol{N}}{\partial x}\frac{\partial^2 x}{\partial \xi^2} + \frac{\partial}{\partial \xi}\left(\frac{\partial \boldsymbol{N}}{\partial y}\right)\frac{\partial y}{\partial \xi} + \frac{\partial \boldsymbol{N}}{\partial y}\frac{\partial^2 y}{\partial \xi^2}\\ &= \left[\frac{\partial}{\partial x}\left(\frac{\partial \boldsymbol{N}}{\partial x}\right)\frac{\partial x}{\partial \xi} + \frac{\partial}{\partial y}\left(\frac{\partial \boldsymbol{N}}{\partial x}\right)\frac{\partial y}{\partial \xi}\right]\frac{\partial x}{\partial \xi} + \frac{\partial \boldsymbol{N}}{\partial x}\frac{\partial^2 x}{\partial \xi^2} + \left[\frac{\partial}{\partial x}\left(\frac{\partial \boldsymbol{N}}{\partial y}\right)\frac{\partial x}{\partial \xi} + \frac{\partial}{\partial y}\left(\frac{\partial \boldsymbol{N}}{\partial y}\right)\frac{\partial y}{\partial \xi}\right]\frac{\partial y}{\partial \xi} + \frac{\partial \boldsymbol{N}}{\partial y}\frac{\partial^2 y}{\partial \xi^2}\\ &= \frac{\partial^2 \boldsymbol{N}}{\partial x^2}\left(\frac{\partial x}{\partial \xi}\right)^2 + \frac{\partial^2 \boldsymbol{N}}{\partial y \partial x}\frac{\partial y}{\partial \xi}\frac{\partial x}{\partial \xi} + \frac{\partial \boldsymbol{N}}{\partial x}\frac{\partial^2 x}{\partial \xi^2} + \frac{\partial^2 \boldsymbol{N}}{\partial x \partial y}\frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \xi} + \frac{\partial^2 \boldsymbol{N}}{\partial y^2}\left(\frac{\partial y}{\partial \xi}\right)^2 + \frac{\partial \boldsymbol{N}}{\partial y}\frac{\partial^2 y}{\partial \xi^2}\\ &= \frac{\partial^2 \boldsymbol{N}}{\partial x^2}\left(\frac{\partial x}{\partial \xi}\right)^2 + \frac{\partial^2 \boldsymbol{N}}{\partial y^2}\left(\frac{\partial y}{\partial \xi}\right)^2 + \frac{\partial^2 \boldsymbol{N}}{\partial x \partial y}\left(2\frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \xi}\right) + \frac{\partial \boldsymbol{N}}{\partial x}\frac{\partial^2 x}{\partial \xi^2} + \frac{\partial \boldsymbol{N}}{\partial y}\frac{\partial^2 y}{\partial \xi^2}\end{aligned} \tag{B.2}$$

Similarly the other second order derivatives are derived, which yields:

$$\frac{\partial^2 \boldsymbol{N}}{\partial \eta^2} = \frac{\partial^2 \boldsymbol{N}}{\partial x^2}\left(\frac{\partial x}{\partial \eta}\right)^2 + \frac{\partial^2 \boldsymbol{N}}{\partial y^2}\left(\frac{\partial y}{\partial \eta}\right)^2 + \frac{\partial^2 \boldsymbol{N}}{\partial x \partial y}\left(2\frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \eta}\right) + \frac{\partial \boldsymbol{N}}{\partial x}\frac{\partial^2 x}{\partial \eta^2} + \frac{\partial \boldsymbol{N}}{\partial y}\frac{\partial^2 y}{\partial \eta^2} \tag{B.3}$$

$$\frac{\partial^2 \boldsymbol{N}}{\partial \xi \partial \eta} = \frac{\partial^2 \boldsymbol{N}}{\partial x^2}\left(\frac{\partial x}{\partial \xi}\frac{\partial x}{\partial \eta}\right) + \frac{\partial^2 \boldsymbol{N}}{\partial y^2}\left(\frac{\partial y}{\partial \xi}\frac{\partial y}{\partial \eta}\right) + \frac{\partial^2 \boldsymbol{N}}{\partial x \partial y}\left(\frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} + \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi}\right) + \frac{\partial \boldsymbol{N}}{\partial x}\frac{\partial^2 x}{\partial \xi \partial \eta} + \frac{\partial \boldsymbol{N}}{\partial y}\frac{\partial^2 y}{\partial \xi \partial \eta} \tag{B.4}$$

By combining the three equations (B.2), (B.3) and (B.4) into a matrix form, it yields:

$$\begin{bmatrix}\left(\frac{\partial x}{\partial \xi}\right)^2 & \left(\frac{\partial x}{\partial \xi}\right)^2 & 2\frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \xi}\\ \left(\frac{\partial x}{\partial \eta}\right)^2 & \left(\frac{\partial x}{\partial \eta}\right)^2 & 2\frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \eta}\\ \frac{\partial x}{\partial \xi}\frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \xi}\frac{\partial y}{\partial \eta} & \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} + \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi}\end{bmatrix}\begin{Bmatrix}\frac{\partial^2 \boldsymbol{N}}{\partial x^2}\\ \frac{\partial^2 \boldsymbol{N}}{\partial y^2}\\ \frac{\partial^2 \boldsymbol{N}}{\partial x \partial y}\end{Bmatrix} = \begin{Bmatrix}\frac{\partial^2 \boldsymbol{N}}{\partial \xi^2} - \frac{\partial \boldsymbol{N}}{\partial x}\frac{\partial^2 x}{\partial \xi^2} - \frac{\partial \boldsymbol{N}}{\partial y}\frac{\partial^2 y}{\partial \xi^2}\\ \frac{\partial^2 \boldsymbol{N}}{\partial \eta^2} - \frac{\partial \boldsymbol{N}}{\partial x}\frac{\partial^2 x}{\partial \eta^2} - \frac{\partial \boldsymbol{N}}{\partial y}\frac{\partial^2 y}{\partial \eta^2}\\ \frac{\partial^2 \boldsymbol{N}}{\partial \xi \partial \eta} - \frac{\partial \boldsymbol{N}}{\partial x}\frac{\partial^2 x}{\partial \xi \partial \eta} - \frac{\partial \boldsymbol{N}}{\partial y}\frac{\partial^2 y}{\partial \xi \partial \eta}\end{Bmatrix} \tag{B.5}$$

Therefore, the second order derivatives in the physical element are obtained using the following mapping:

$$\begin{Bmatrix} \frac{\partial^2 \boldsymbol{N}}{\partial x^2} \\ \frac{\partial^2 \boldsymbol{N}}{\partial y^2} \\ \frac{\partial^2 \boldsymbol{N}}{\partial x \partial y} \end{Bmatrix} = \begin{bmatrix} (\frac{\partial x}{\partial \xi})^2 & (\frac{\partial x}{\partial \xi})^2 & 2\frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \xi} \\ (\frac{\partial x}{\partial \eta})^2 & (\frac{\partial x}{\partial \eta})^2 & 2\frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \eta} \\ \frac{\partial x}{\partial \xi}\frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \xi}\frac{\partial y}{\partial \eta} & \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} + \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi} \end{bmatrix}^{-1} \begin{Bmatrix} \frac{\partial^2 \boldsymbol{N}}{\partial \xi^2} - \frac{\partial \boldsymbol{N}}{\partial x}\frac{\partial^2 x}{\partial \xi^2} - \frac{\partial \boldsymbol{N}}{\partial y}\frac{\partial^2 y}{\partial \xi^2} \\ \frac{\partial^2 \boldsymbol{N}}{\partial \eta^2} - \frac{\partial \boldsymbol{N}}{\partial x}\frac{\partial^2 x}{\partial \eta^2} - \frac{\partial \boldsymbol{N}}{\partial y}\frac{\partial^2 y}{\partial \eta^2} \\ \frac{\partial^2 \boldsymbol{N}}{\partial \xi \partial \eta} - \frac{\partial \boldsymbol{N}}{\partial x}\frac{\partial^2 x}{\partial \xi \partial \eta} - \frac{\partial \boldsymbol{N}}{\partial y}\frac{\partial^2 y}{\partial \xi \partial \eta} \end{Bmatrix} \tag{B.6}$$

Recalling the isoparametric mapping between the reference element and a physical element e:

$$\begin{aligned} x^{(e)}(\xi,\eta) &= \sum_i x_i^{(e)} N_i(\xi,\eta), \\ y^{(e)}(\xi,\eta) &= \sum_i y_i^{(e)} N_i(\xi,\eta) \end{aligned} \tag{B.7}$$

Then, the first and second order derivatives of $x^{(e)}(\xi,\eta)$ and $y^{(e)}(\xi,\eta)$ w.r.t ξ and η are given by:

$$\begin{aligned} \frac{\partial x^{(e)}(\xi,\eta)}{\partial \xi} &= \sum_i x_i^{(e)} \frac{\partial N_i(\xi,\eta)}{\partial \xi}, \\ \frac{\partial y^{(e)}(\xi,\eta)}{\partial \xi} &= \sum_i y_i^{(e)} \frac{\partial N_i(\xi,\eta)}{\partial \xi}, \\ \frac{\partial x^{(e)}(\xi,\eta)}{\partial \eta} &= \sum_i x_i^{(e)} \frac{\partial N_i(\xi,\eta)}{\partial \eta}, \\ \frac{\partial y^{(e)}(\xi,\eta)}{\partial \eta} &= \sum_i y_i^{(e)} \frac{\partial N_i(\xi,\eta)}{\partial \eta} \end{aligned} \tag{B.8}$$

and

$$\begin{aligned} \frac{\partial^2 x^{(e)}(\xi,\eta)}{\partial \xi^2} &= \sum_i x_i^{(e)} \frac{\partial^2 N_i(\xi,\eta)}{\partial \xi^2}, \\ \frac{\partial^2 y^{(e)}(\xi,\eta)}{\partial \xi^2} &= \sum_i y_i^{(e)} \frac{\partial^2 N_i(\xi,\eta)}{\partial \xi^2}, \\ \frac{\partial^2 x^{(e)}(\xi,\eta)}{\partial \eta^2} &= \sum_i x_i^{(e)} \frac{\partial^2 N_i(\xi,\eta)}{\partial \eta^2}, \\ \frac{\partial^2 y^{(e)}(\xi,\eta)}{\partial \eta^2} &= \sum_i y_i^{(e)} \frac{\partial^2 N_i(\xi,\eta)}{\partial \eta^2}, \\ \frac{\partial^2 x^{(e)}(\xi,\eta)}{\partial \xi \partial \eta} &= \sum_i x_i^{(e)} \frac{\partial^2 N_i(\xi,\eta)}{\partial \xi \partial \eta}, \\ \frac{\partial^2 y^{(e)}(\xi,\eta)}{\partial \xi \partial \eta} &= \sum_i y_i^{(e)} \frac{\partial^2 N_i(\xi,\eta)}{\partial \xi \partial \eta} \end{aligned} \tag{B.9}$$

References

[1] Donea, J., Huerta, A. (2004). *Finite Element Methods for Flow Problems.* Chichester: Wiley, pp.135.