Project 1: Numerical solution of convection-diffusion equations

Mario Putti, Antonia Larese
Department of Mathematics – University of Padua

June 14, 2024

Content

1 Problem formulation

We consider the convection-diffusion (or advection-diffusion) equation, governing, for example, the fate of a contaminant dissolved in a fluid moving with velocity $\beta(x,t)$ in a domain $\Omega \subset \mathbb{R}^d$, $d = 1, 2, 3$, as:

$$\frac{\partial u}{\partial t} = \operatorname{div}(D \nabla u) - \operatorname{div}(\beta u) + f \qquad \text{in } \Omega \times (0,T) \tag{1.1}$$

$$u(x,0) = u_0(x) \qquad x \in \Omega, t = 0 \tag{1.2}$$

$$u(x,t) = u_D(x,t) \qquad x \in \Gamma_D, t > 0 \tag{1.3}$$

$$K \nabla u \cdot \vec{n} = q_N(x,t) \qquad x \in \Gamma_N, t > 0 \tag{1.4}$$

$$(\beta u + K \nabla c) \cdot \vec{n} = q_C(x,t) \qquad x \in \Gamma_C, t > 0 \tag{1.5}$$

where:

- u [-]: solute concentration (generally normalized to 1);
- x [L]: spatial coordinate with respect to a Eulerian reference system (for $d = 2$: $x = (x_1, x_2)$);
- t [T]: time;
- $\Omega \subset \mathbb{R}^d$ domain ($d = 1, 2,$ or 3);
- $\Gamma = \Gamma_D \cup \Gamma_N \cup \Gamma_C$ boundary of Ω;
- Γ_D: Dirichlet boundary;
- Γ_N: Neumann boundary;
- Γ_C: Robin (or Cauchy) boundary;
- K [L^2/T]: diffusion coefficient (assumed a scalar function of (x,t));
- f [1/T]: forcing function;
- $u_0(x)$: initial concentration;
- $u_D(x,t)$: value of the Dirichlet boundary condition;
- $q_N(x,t)$: value of the diffusive flux at Neumann boundary;
- $q_C(x,t)$: value of total flux at Cauchy boundary;
- $\vec{n}$: outward normal vector (for $d = 2$: $\vec{n} = (n_x, n_y)$);.

We want to solve the convection-diffusion equation (eq. (1.1)) in the stationary case ($\partial u/\partial t = 0$) in the domain Ω given by:

$$\Omega = \{(x, y) : 0 \leq x \leq 1, 0 \leq y \leq 1\}$$

with constant diffusion coefficient and velocity, and with zero forcing. The data and boundary conditions are appropriately chosen so that a boundary layer is obtained. Thus we want to solve:

$$\frac{\partial}{\partial x}\left(K\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(K\frac{\partial u}{\partial y}\right) - \frac{\partial \beta_x u}{\partial x} - \frac{\partial \beta_y u}{\partial y} = 0 \qquad (x, y) \in \Omega \tag{1.6}$$

$$u(x, t) = 1 \qquad x \in \Gamma_1, t > 0 \tag{1.7}$$

$$u(x, t) = 0 \qquad x \in \Gamma_2, t > 0 \tag{1.8}$$

where the boundaries Γ_1 and Γ_2 are given by:

$$\Gamma_1 = \{(x, y) : [x = 0, 0 \leq y \leq 1] \cup [0 \leq x \leq 0.3, y = 0]\}$$

$$\Gamma_2 = \{(x, y) : [x = 1, 0 \leq y \leq 1] \cup [0 \leq x \leq 1, y = 1] \cup [0.3 \leq x \leq 1, y = 0]\}$$

Note that there is no closed-formed solution for this problem. Thus we want to find the FEM numerical solution using linear Galerkin FEM on a triangular mesh.

1.1 Galerkin finite elements

Let u_h be the approximate solution given by:

$$u_h(x, y) = \sum_{i=1}^{n} u_i \phi_i (x, y) \tag{1.9}$$

where $\phi_i (x, y)$, $i = 1, \ldots, n$ are piecewise linear Lagrangean basis functions and u_i are the values of u_h at the n nodal points of the triangulation. Subsituting u_h into eq. (1.6) we have after application of Green's Lemma:

$$\int_\Omega \left(K_{xx}\frac{\partial u_h}{\partial x}\frac{\partial \phi_i}{\partial x} + K_{yy}\frac{\partial u_h}{\partial y}\frac{\partial \phi_i}{\partial y}\right) d\Omega - \int_{\Gamma_N} \left(K_{xx}\frac{\partial u_h}{\partial x}n_x + K_{yy}\frac{\partial u_h}{\partial y}n_y\right) \phi_i d\Gamma_N + \int_\Omega \left(\frac{\partial \beta_x u_h}{\partial x} + \frac{\partial \beta_y u_h}{\partial y}\right) \phi_i d\Omega = 0 \qquad i = 1, \ldots, n \tag{1.10}$$

where K_{xx} and K_{yy} are the components of the (diagonal) diffusion tensor along x and y, respectively. Using eq. (1.9) and the boundary conditions we have:

$$\int_\Omega \left[\sum_{j=1}^{n} \left(K_{xx}\frac{\partial \phi_j}{\partial x}\frac{\partial \phi_i}{\partial x} + K_{yy}\frac{\partial \phi_j}{\partial y}\frac{\partial \phi_i}{\partial y}\right) u_j\right] d\Omega + \int_\Omega \sum_{j=1}^{n} \left(\beta_x \frac{\partial \phi_i}{\partial x} + \beta_y \frac{\partial \phi_i}{\partial y}\right) \phi_i d\Omega u_j - \int_{\Gamma_N} q\phi_i d\Gamma_N = 0 \qquad i = 1, \ldots, n \tag{1.11}$$

which can be written in matrix form as:

$$(H + B)\mathbf{u} = \mathbf{q} \tag{1.12}$$

where:

$$H = \{h_{ij}\} \qquad h_{ij} = \int_\Omega \left[\left(K_{xx} \frac{\partial \phi_j}{\partial x} \frac{\partial \phi_i}{\partial x} + K_{yy} \frac{\partial \phi_j}{\partial y} \frac{\partial \phi_i}{\partial y} \right) \right] d\Omega$$

$$B = \{b_{ij}\} \qquad b_{ij} = \int_\Omega \left(\beta_x \frac{\partial \phi_j}{\partial x} + \beta_y \frac{\partial \phi_j}{\partial y} \right) \phi_i d\Omega$$

are stiffness and transport matrices, respectively. Note that now $B \neq B^T$.
We use the standard Lagrangean basis functions, and build these matrices using the standard element-by-element assembly procedure.

2 The FEM solver

The FEM solver shares the same structure of a pure diffusion code. The most important difference is the assembly of the non-symmetric transport matrix and the fact that the system matrix is now non-symmetric. Hence the Preconditioned Conjugate Gradients (PCG) method cannot be used, but we need to resort to the Generalized Minimal Residual Method (GMRES) method (see below).
In addition, the stabilizing Streamline Upwind Diffusion (SUD) term needs to be added to avoid oscillations in the numerical solution when convection dominates.

2.1 Solution of the linear system

The system matrix $H + B$ is sparse and has the same sparsity structure of the stiffness matrix H. The GMRES algorithm, as implemented in MATLAB, needs to be used for the solution of the linear system, as preconditioned by the incomplete LU factorization.
The call to the MATLAB GMRES function is as follows:

Algorithm 1 Prototype call to the GMRES Matlab function with the ILU Preconditioner.

```
1: restart=10; tol=1e-9; maxit=20;
2: setup.type='nofill';
3: setup.milu='off';
4: [L,U]=ilu(SYSMAT,setup);
5: x=gmres(SYSMAT,RHS,restart,tol,maxit,L,U);
```

For GMRES to work in Matlab, SYSMAT must be of type "sparse".

2.2 Dirichlet boundary conditions

Note that the use of the penalty method to impose the boundary conditions will, in this case cause convergence problems to the GMRES iteration. To see this, we try with a large value of the penalty parameter ($\lambda = 10^{40}$) and verify that the boundary condition is correctly imposed, but the real residual ($\|b - Ax_k\|$) is very large, indicating that the solution is not correct. This is due to cancellations that occur when calculating the exit norm $\|r_k\| / \|b\|$. Actually, with such a value for λ no iterations are performed, since the relative residual is smaller than the requested tolerance for any initial condition (λ cancels in the evaluation of r_k but $\|b\| \approx \lambda$. Hence the boundary lifting operator needs to be employed, or, alternatively but less accurately, a relatively small value of λ (e.g. $\approx 10^{10}$) needs to be employed.
You are asked to experiment with this problem, by looking at the different options.

2.3 Implementation of the streamline upwind stabilization term

The streamline upwind diffusion elemental matrix is given by:

$$s_{ij}^e = \int_{\Omega^{(e)}} \tau \frac{h^{(e)}}{K^{(e)}|\beta^{(e)}|} \left(\beta_x^{(e)} \frac{\partial \phi_i}{\partial x} + \beta_y^{(e)} \frac{\partial \phi_i}{\partial y}\right) \left(\beta_x^{(e)} \frac{\partial \phi_j}{\partial x} + \beta_y^{(e)} \frac{\partial \phi_j}{\partial y}\right) \, d\Omega$$

where $K^{(e)} = \max\left(K_{xx}^{(e)}, K_{yy}^{(e)}\right)$ is the norm of the diffusion tensor assumed constant in each element, $h^{(e)}$ is the length of the largest edge of the triangle $\Omega^{(e)}$, $|\beta^{(e)}| = \sqrt{\beta_x^2 + \beta_y^2}$ is the norm of the velocity vector (again elementwise constant) and the term τ is a safeguard coefficient used to calibrate the desired amount of numerical diffusion to be added.

2.4 Mass balance calculation

The correctness of the numerical solution is often tested by verifying that the global mass balance be satisfied. In essence, we calculate the mass entering and exiting the domain from the boundaries and verify that their algebraic sum is close to zero.
In our case of zero forcing, there are two flux components that need to be calculated: the diffusive and the convective flux. One is given by $q_D = -D \nabla u \cdot \vec{n}$ and the other by $q_C = u\beta \cdot \vec{n}$. The former can be easily calculated by multiplying the diffusion stiffness matrix H (without the penalty term) by the final solution. The values at the Dirichlet nodes are exactly the diffusive fluxes we are looking for. The other component is easily calculated from the boundary conditions.

3 Project 1

The student is asked to implement the solver as described before and to apply it to the solution of the described problem using the provided mesh. To verify the implementation, first try the

code with $\beta_x = \beta_y = 0$ and compare its solution with the solution obtained with the available Poisson solver.
The student is asked to explore the origin of oscillations by solving the problem for decreasing values of the diffusion coefficient $K = K_{xx} = K_{yy}$ varying it in the interval $[10^{-3} \div 1]$ with a constant velocity field $\beta = (1, 3)^T$. Find what is the minimum value below which oscillations occur, and determine a "pseudo" Peclet number for the considered mesh.
Apply the code with the SUD stabilization and explore its behavior varying K and τ.
In figs. 3.1 and 3.2 sample solutions to the proposed problem are shown. It is possible to identify in the plots the effects of the large numerical diffusion introduced in the stable case fig. 3.2, lower panel, along the y direction where velocity is largest. The need to calibrate $\tau > 0$ to reach a compromise between stability and numerical diffusion is evident.

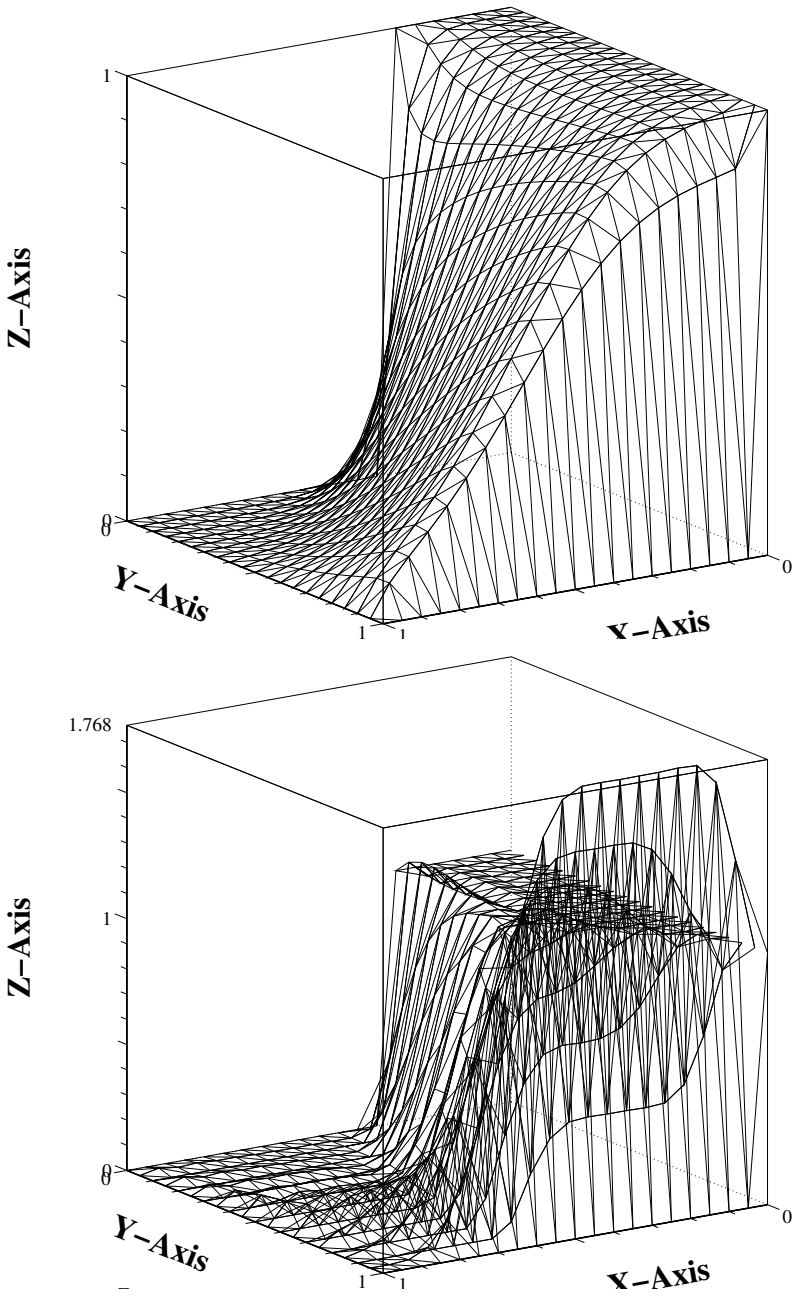

Figure 3.1: Convection-diffusion solution with linear Galerkin without stabilization. The upper and lower panels show the cases with $K = 0.1$ and $K = 0.01$, respectively. In both problems $\beta = (1, 3)^T$.

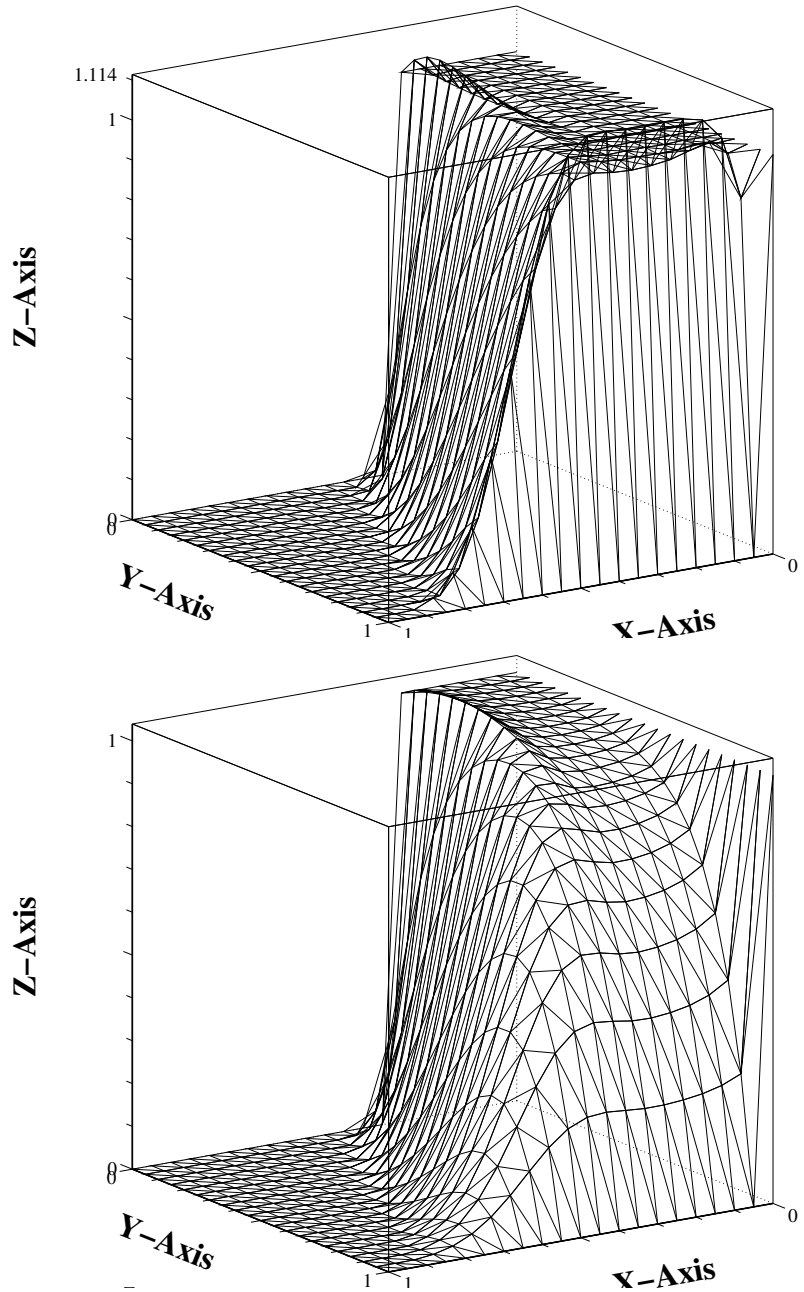

Figure 3.2: Convection-diffusion solution with linear Galerkin with stabilization in the case $\beta = (1, 3)^T$ and $K = 0.01$. The upper panel shows the solution with $\tau = 0.01$, while the lower panel shows the solution with $\tau = 1.0$. Note that numerical oscillations decrease as numerical diffusion increases.