Uppsala university, Division of Scientific computing

Stabilized finite element methods

1TD050: Advanced Numerical Methods, 10.0 hp

Murtazo Nazarov

September 4, 2023

Introduction

Let Ω be a fixed (open) domain in $\mathbb{R}^2$, with boundary $\partial\Omega$ over a time interval $[0, T]$ with initial time zero and the final time T. We are interested in solving the following time-dependent scalar conservation laws:

$$\begin{aligned} \partial_t u + \nabla\cdot \boldsymbol{f}(u) &= 0, \qquad && (\boldsymbol{x}, t) \in \Omega \times (0, T], \\ u(\boldsymbol{x}, 0) &= u_0(\boldsymbol{x}), && \boldsymbol{x} \in \Omega, \end{aligned} \tag{1}$$

with appropriate boundary conditions. Here u represents the unknown variable, $\boldsymbol{f} \in \mathcal{C}^1(\mathbb{R}, \mathbb{R}^2)$ is the flux term and $u_0 \in L^\infty(\mathbb{R}^2)$ is given initial data.

Let $0 = t_0 < t_1 < ... < t_N = T$ be a sequence of discrete time steps with associated time intervals $I_n = (t_{n-1}, t_n]$ of length $k_n = t_n - t_{n-1}$, $n = 1, 2, \cdots, N$. Let

$$\mathcal{X}_h := \{v : v \in H^1(\Omega),\ v(\boldsymbol{x}) - \text{cont. pw. linear in } \Omega\},$$

be a finite element space consisting of continuous piecewise linear polynomials on a mesh $\mathcal{T}_h = \{K\}$ of mesh-size $h(\boldsymbol{x})$ and let $V_{h,0}$ be the space of all functions in V_h vanishing on $\partial\Omega$. Next, let us denote by $U_0 = \tilde{\pi}_h u_0$ the interpolation of the initial data into the finite element space $\mathcal{X}_h$. We are now ready to formulate the following finite element approximation: for $n = 1, 2, \cdots, N$ find $U_n \in \mathcal{X}_h$ such that

$$\frac{1}{k_n}\Big(U_n - U_{n-1},\, v\Big) + \frac{1}{2}\Big(\nabla\cdot (\boldsymbol{f}(U_{n-1}) + \boldsymbol{f}(U_n)),\, v\Big) = 0, \quad \forall v \in \mathcal{X}_h, \tag{2}$$

where $U_n = U_h(t_n)$ is the solution at the discrete time steps t_n. The implicit Crank-Nicholson time-stepping is used for the time discretization.

It is well-known that the Galerkin space discretization of the flux term is equivalent to a central difference approximation, which is numerically unstable. A mesh-dependent, consistent numerical stabilization is usually added to the FEM discretization to fix this issue. A *residual based artificial viscosity method* (RV-method) is one of the techniques to cure this instability. This method reads: find $U_n \in \mathcal{X}_h$ such that

$$\begin{aligned}\frac{1}{k_n}\Big(U_n - U_{n-1},\, v\Big) + \frac{1}{2}\Big(\nabla\cdot (\boldsymbol{f}(U_{n-1}) + \boldsymbol{f}(U_n)),\, v\Big)&\\ + \frac{1}{2}\Big(\varepsilon_n(U_{n-1})\nabla(U_n + U_{n-1}), \nabla v\Big) &= 0, \quad \forall v \in \mathcal{X}_h,\end{aligned} \tag{3}$$

where the artificial viscosity $\varepsilon_n := \varepsilon_h(t_n)$ is computed as follows: for each cell $K \subset \mathcal{T}_h$ compute the finite element residuals as

$$R(U_n) = \frac{1}{k_n}(U_n - U_{n-1}) + \nabla\cdot \boldsymbol{f}(U_n), \tag{4}$$

then compute:

$$\varepsilon_{n,K} = \min\left(C_{\text{vel}} h_K \beta_K,\, C_{\text{RV}} h_K^2 \frac{\|R_{\text{RV}}\|_{\infty,K}}{\|U_n - \overline{U_n}\|_{\infty,\Omega}}\right), \tag{5}$$

where $\|U_n - \overline{U_n}\|_{\infty,\Omega}$ is a normalization term, with $\overline{U_n}$ denoting the space average of the solution over Ω, h_K is the mesh-size of the element K, that can be a diameter or the smallest edge of K, β_K denotes the local element wave speed that is computed as $\beta_K \equiv \|\boldsymbol{f}'(U_n)|\|_{L^\infty(K)} = \max_{N_i\in K, i=0,1,2}\Big(\big[(f_1'(U_n))^2 + (f_2'(U_n))^2\big]^{\frac{1}{2}}(N_i)\Big)$, where N_i, $i = 0,1,2$ is the nodes of the element K, $C_{\text{vel}} = 0.25$ and $C_{\text{RV}} = 1$ are stabilization parameters. The standard Galerkin solution can be obtained by setting $C_{\text{vel}} = 0$.

To generate a mesh in Matlab, one can use the `initmesh` function. This function is part of the PDE-Toolbox in Matlab that generates triangular meshes in two space dimensions. For example, the mesh for the unit disk can be constructed as:

```
g = @circleg;
hmax = 1/16;
[p,e,t] = initmesh(g,'hmax',hmax);
```

where `[p,e,t]` are point, edge and element data given by the triangulation in Matlab. Then, the residual based artificial viscosity is a piecewise constant function defined in each element `t`. This can be implemented in Matlab through the following loop:

```
...
nt = size(t,2);                      % get #-of elements
Cvel = 0.25;
Crv = 1.0;
Res = ...;                           % compute the residual
                                     % using the L2-projection
Res = Res/...;                       % normalize the residual
for K = 1:nt                         % iterate over all cells
    ...
    beta_K = ...;                    % get max-speed for K
    Res_K = ...;                          % get max-residual for K
    eps(K) = min(C1*h*beta_K, C2*h^2*Res_K);
end
```

For the rest of your computation you can set the time-step to $k_n = \mathrm{CFL}\frac{h_{\max}}{\|\boldsymbol{f}'(U_n)\|_{L^\infty(\Omega)}}$, where CFL=0.5 and $h_{\max}$ is the largest mesh-size in the mesh $\mathcal{T}_h$.

Linear advection equation

Let us consider the problem (1) in the unit disk $\Omega = \{\boldsymbol{x} : x_1^2 + x_2^2 \le 1\}$, and rewrite the flux term in the non-conservative form, i.e., $\nabla\cdot \boldsymbol{f}(u) := \boldsymbol{f}'(u)\cdot\nabla u$. Assume that $\boldsymbol{f}'(u) := 2\pi(-x_2, x_1)$ and in addition the homogenious Dirichlet boundary condition is applied on the entire boundary: $u(\boldsymbol{x}, t) = 0$, $\forall \boldsymbol{x} \in \partial\Omega$.

To apply the homogeneous Dirichlet boundary condition *strongly* (i.e. after assembling the linear system $\boldsymbol{A\xi} = \boldsymbol{b}$) one can do:

```
I = eye(length(p));         % construct the identity matrix
A = AssembleA(...);         % stiffness matrix
b = Assembleb(...);         % load vector
A(e(1,:),:) = I(e(1,:),:);  % replace the rows corresponding
                            % to the boundary nodes by
                            % corresponding rows of I
b(e(1,:)) = 0;              % put the boundary value into the RHS
```

Further, we need to compute the L^2-norm of the error several times in our computation: $\|e\|_{L^2(\Omega)} = (\int_\Omega e^2\,\mathrm{d}\boldsymbol{x})^{\frac{1}{2}}$, where $e \equiv u - U_h$. Provided that `M` is the mass matrix, this norm can easily be computed in Matlab as: `L2E=sqrt(e'*M*e);`.

Hint for the report: Add title and axises to the plots. Plot your solution in a readable format, for example work a bit when choosing the colormap, try to plot a 3D view of the 2D figure, etc..

Part 1. Galerkin FEM

Problem 1.1.

Implement a Matlab code to solve the finite element approximation (2). Set the initial data to:

$$u_0(\boldsymbol{x}, 0) = \frac{1}{2}\left(1 - \tanh\left(\frac{(x_1 - x_1^0)^2 + (x_2 - x_2^0)^2}{r_0^2} - 1\right)\right), \tag{6}$$

with $r_0 = 0.25$, $(x_1^0, x_2^0) = (0.3, 0)$, and solve the problem until $T = 1$. Solve the problem when CFL=0.5 in two meshes with $h_{\max} = \frac{1}{8}$ and $h_{\max} = \frac{1}{16}$. Plot and discuss your solution from two meshes at the final time.

Problem 1.2.

Compute the L^2-norm of the error for sequence of meshes obtained by setting: $h_{\max} = \{\frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}\}$. Find the convergence rate α of the approximation in the L^2-norm. Plot

$h_{\max}$ versus the L^2-norm of the errors and $h_{\max}$ versus $h^{\alpha}_{\max}$ at the same figure using the *loglog*-plot in Matlab. What is the convergence orders of the method?

Problem 1.3.

Now, set the initial data to:

$$u_0(\boldsymbol{x}, 0) = \begin{cases} 1 & \text{if } (x_1 - x_1^0)^2 + (x_2 - x_2^0)^2 \le r_0^2, \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

where, with $r_0 = 0.25$, $(x_1^0, x_2^0) = (0.3, 0)$. Perform the same analysis as in Problem 1.1 and 1.2, obtain the plots and report them. What are the convergence orders for this case? What happens with the solution?

Part 2. Stabilized FEM

Problem 2.1.

Another way of stabilizing the Galerkin approximation (2) is the Streamline-Upwind-Petrov-Galerkin (SUPG) method. Write down the SUPG method for (1) using Crank-Nicholson method in time. Prove that the SUPG method is stable.

Problem 2.2.

Implement a Matlab code to solve the finite element approximation (3) and the SUPG method from Problem 2.1. Solve the tasks in Problem 1.1 and 1.2 using these two methods, plot the corresponding results. It is useful to plot the corresponding artificial viscosity $\varepsilon_n(U_n)$ from the RV method. What are the convergence orders of the methods? Compare the methods and discuss your results.

Problem 2.3.

Repeat the tasks in Problem 2.2 with the discontinuous initial data (7). What are the convergence orders for this case? What happens with the solution? Which method performs better? Motivate.

Good luck!
Murtazo
Uppsala, September 2022