Uppsala university, Division of Scientific computing

Nonlinear conservation laws and goal-oriented adaptivity

1TD050: Advanced Numerical Methods, 10.0 hp

Murtazo Nazarov

October 2, 2023

Introduction

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

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

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

Part A. Finite element approximations of the KPP problem

In this part, we are interested in solving the so-called KPP rotating wave problem, where the flux and initial data in (1) are defined as

$$\boldsymbol{f}(u) = (\sin u, \cos u), \quad u_0(\boldsymbol{x}) = \begin{cases} \frac{14\pi}{4}, & \text{if } \sqrt{x^2 + y^2} \leq 1, \\ \frac{\pi}{4}, & \text{otherwise.} \end{cases} \tag{2}$$

This test was originally proposed in 2007 by Kurganov, Popov and Petrova. This is an interesting and challenging problem to test numerical schemes. For example it was tested in the literature that central-upwind schemes based on WENO5, Minmod 2 and SuperBee reconstructions converge to non-entropic solutions.

The computation for this problem is the square $[-2, 2] \times [-2.5, 1.5]$ and the problem is usually solved until $T = 1$. The solution of the KPP problem contains a strongly rotating shock wave. This phenomenon is difficult to capture numerically. The aim of this section to compare two types of stabilization techniques in finite elements, namely linear (SUPG) and nonlinear (RV) methods.

Problem A.1.

Implement Matlab code to solve problem (1) with data (2) using continuous finite element approximations in space and implicit Crank-Nicholson method in time. Use the Piccard iteration method to linearize the nonlinear part. Use the RV and SUPG methods to stabilize the Galerkin method. Solve the problem using three different methods for two grids with $h_{\max} = \frac{1}{8}$ and $h_{\max} = \frac{1}{16}$. Discuss your result.

Problem A.2.

Now, let us solve the problem (1) using the 4-th order classical explicit Runge-Kutta (RK4) method. Write down a finite element approximation of (1) using the RK4 method and implement it in Matlab. Solve the problem (1) with data (2) using the RK4 method for two meshes with $h_{\max} = \frac{1}{8}$ and $h_{\max} = \frac{1}{16}$. Is there any time-step restriction for these different methods? Compare your result with the one from Problem A.1. Which one is more accurate? Motivate your answer.

Part B. Goal-oriented error estimates

Consider the advection-diffusion equation in $\Omega = \{\boldsymbol{x} : x_1^2 + x_2^2 \leq 1\}$, $\boldsymbol{f}'(u) := \boldsymbol{\beta}(\boldsymbol{x}) = 2\pi(-x_2, x_1)$, a diffusion term has the form $-\nabla\cdot(\varepsilon\nabla u)$, $\varepsilon \geq 0$, with boundary condition $u(\boldsymbol{x}, t) = 0$, $\forall \boldsymbol{x} \in \partial\Omega$. And the initial condition is given as

$$u_0(\boldsymbol{x}, 0) = \begin{cases} 1 & \text{if } (x_1 - x_1^0)^2 + (x_2 - x_2^0)^2 \leq r_0^2, \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

where, with $r_0 = 0.25$, $(x_1^0, x_2^0) = (0.3, 0)$.

The so-called *the adjoint* or *the dual* problem corresponding to the problem (1) is defined as following: find $z := z(\boldsymbol{x}, t)$ such that

$$\begin{aligned} -\partial_t z - \boldsymbol{\beta}\cdot\nabla z - \nabla\cdot(\varepsilon\nabla z) &= \psi_\Omega, \qquad (\boldsymbol{x}, t) \in \Omega \times [T, 0), \\ z &= 0, \qquad (\boldsymbol{x}, t) \in \partial\Omega \times [T, 0), \\ z(\boldsymbol{x}, T) &= 0, \qquad \boldsymbol{x} \in \Omega, \end{aligned} \tag{4}$$

where $\psi_\Omega := \psi_\Omega(\boldsymbol{x}, t)$ is given functions with $\psi_\Omega \in L_2(\Omega \times [0, T])$, which defines a linear target functional by

$$\mathcal{M}(u) = \int_{\Omega\times[0,T]} u\psi_\Omega \,\mathrm{d}x\,\mathrm{d}t.$$

For example, the average of the error in domain Ω can be one example of the target functional, *i.e.*, set $\psi_\Omega = 1$, then:

$$\mathcal{M}(u) - \mathcal{M}(U) = \int_{\Omega\times[0,T]} (u - U) \,\mathrm{d}x\,\mathrm{d}t.$$

Theorem 1. *For the finite element approximation $U(t)$:*
1. the following error representation holds:

$$|\mathcal{M}(u) - \mathcal{M}(U)| = \int_0^T \sum_{K \in \mathcal{T}_h} \int_K \mathcal{R}(U(t))(z(t) - \pi_h z(t)) \,\mathrm{d}x \,\mathrm{d}t, \tag{5}$$

2. the following error estimates holds:

$$|\mathcal{M}(u) - \mathcal{M}(U)| \leq C \int_0^T \sum_{K \in \mathcal{T}_h} h_K \|\mathcal{R}(U(t))\|_{L^2(K)} \|Dz(t)\|_{L^2(K)} \,\mathrm{d}t, \tag{6}$$

where U is finite element approximation of u, π_h is the standard interpolation function, $\mathcal{R}(U)$ is the residual, Dz the first derivative of z, h_K is the cell diameter.

Let us now define the following *error indicator* for each cell K:

$$\eta_K(t) = C h_K \|\mathcal{R}(U(t))\|_{L^2(K)} \|Dz(t)\|_{L^2(K)}. \tag{7}$$

Then, the fully automatic goal oriented h-adaptive mesh refinement algorithm is:

1. Initialize coarse grid $\mathcal{T}_N$, choose $\mathrm{TOL} > 0$, and set $N := 0$.
2. Solve primal problem (1).
3. Solve dual problem (4).
4. For every $K \in \mathcal{T}_N$: compute a posteriori error estimator η_K.
5. If $\mathcal{E} = \int_0^T \sum_{K \in \mathcal{T}_h} \eta_K(t) \,\mathrm{d}t < \mathrm{TOL}$: STOP
6. Refine 20% of cells with the largest $\int_0^T \eta_K(t) \,\mathrm{d}t$.
7. Set $N := N + 1$ and goto step 2.

Note that η_K is a piecewise constant function on each cell K. Therefore, $\int_0^T \eta_K(t) \,\mathrm{d}t$ is also a piecewise constant function. To refine 20% of the cells with the largest error one can use the following Matlab code:

```
% adaptive mesh refinement:
tol = 0.8*max(eta);
% select elements for refinement
elements = find(eta > tol);
% refine elements using regular refinement
[p,e,t] = refinemesh(g,p,e,t,elements,'regular');
```

Here `eta` is $\int_0^T \eta_K(t) \,\mathrm{d}t$, and the array `elements` is the list 20% of cells with the largest error contribution.

Problem B.1

Prove Theorem 1.

Problem B.2

Formulate the Galerkin finite element method of (4) using a continuous piecewise linear polynomial approximation.

Problem B.3

Let us define the target function as the following function:

$$u_0(\boldsymbol{x}, 0) = \exp\left(\frac{(x_1 - x_1^0)^2 + (x_2 - x_2^0)^2}{r_0^2}\right), \tag{8}$$

which is the Gaussian of height 1, defined at the point $\boldsymbol{x}^0 = (x_1^0, x_2^0)$ with support on a circle of radius r_0.

Implement the goal-oriented adaptive algorithm in Matlab (or any other programming language you prefer). Report the convergence history with respect to the following target functions:

1. the average error in the domain, i.e., $\boldsymbol{x}^0 = (0, 0)$, $r_0 = 1$.
2. the average error in a circle with center $\boldsymbol{x}^0 = (0.3, 0)$ and radius $r_0 = 0.25$.
3. the average error in a circle with center $\boldsymbol{x}^0 = (0.6, 0)$ and radius $r_0 = 0.15$.
4. the average error in a circle with center $\boldsymbol{x}^0 = (-0.6, 0)$ and radius $r_0 = 0.15$.
5. the average error in two circles with centers and radii $\boldsymbol{x}^0 = (0, 0.6)$ and $r_0 = 0.15$, as well as $\boldsymbol{x}^0 = (0, -0.55)$ and $r_0 = 0.35$.

Stabilize your code using the Residual Viscosity method. Set $\varepsilon = 0$, TOL=0.001, and run your code until $T = 1$. Motivate your answer. What does the dual solution indicate?

For each case above, plot the primal solution U, the dual solution z, the total error indicator H, and the corresponding refined meshes. Additionally, plot the total error as a function of the number of points on a logarithmic plot as follows: Let $\mathbf{N}$ be a vector of the total number of mesh points and $\mathbf{E}$ be a vector of total errors for each mesh. Plot

```
loglog(1./N, E, 1./N, 1./sqrt(N))
```

What does this plot show?

Problem B.4

(Non-obligatory part, advanced) Now, let us consider the nonlinear scalar equation discussed in Part 3. Prove a posteriori error estimate for the KPP problem using a goal-oriented argument and implement it in Matlab.

Good luck!
Murtazo
Uppsala, October 2023