Final Project of The Practice of Mathematics

due 2023 AUG 18 (Friday), 22:00

1 The assignments

Write a `C++` package to implement the fourth-order FV-MOL algorithm in Section 12.3 and the approximate projection method in Section 12.4 to solve the advection-diffusion equation and the incompressible Navier-Stokes equations (INSE), respectively. You should use a multigrid method as in Chapter 9 to solve the linear systems.

Verify the validity of your program using the tests in Sections 1.1, 1.2, and 1.3. For each test, you should plot your solutions and report errors and convergence rates on successively refined grids.

1.1 Traveling sinusoidal waves

For $\nu = 0.01$, $\mathbf{u} = (1.0, 0.5, 0.25)$ and $\mathbf{k} = (2\pi, 4\pi, 6\pi)$, we use an expression of the form $\phi = \prod_d \sin\theta_d$ with $\theta_d = k_d x_d - u_d t$ as an exact solution to the advection-diffusion equation to derive the forcing term as

$$f(\mathbf{x},t) = \begin{array}{l}\left(\sum_d \nu k_d^2\right)\prod_d \sin\theta_d \\ +\sum_d \left\{u_d\,(k_d-1)\cos\theta_d \prod_{d'\neq d}\sin\theta_{d'}\right\}.\end{array} \tag{1}$$

On the unit domain $[0,1]^2$, the initial condition is calculated as the exact average $\langle\phi\rangle_{\mathbf{i}}$ evaluated at $t_0 = 0$, which is then advanced to $t_e = 1$ with the time-step chosen such that the Courant number $\mathrm{Cr} = 1.0$. For each dimension, you should use the Dirichlet boundary condition and the Neumann boundary condition for the low side and the high side, respectively. The test should be carried on four successively refined grids with $h = \frac{1}{64}, \frac{1}{128}, \frac{1}{256}, \frac{1}{512}$.

1.2 Gaussian patch in vortex shear

The velocity field of this test is steady and divergence-free:

$$\mathbf{u}(x,y) = a_V\left(\sin^2(\pi x)\sin(2\pi y), -\sin(2\pi x)\sin^2(\pi y)\right), \tag{2}$$

where $a_V = 0.1$ is a scaling parameter.

On the periodic domain $[0,1]^2$, the advection-diffusion equation is advanced with $\nu = 0.001$ and $\mathrm{Cr} = 1.0$ from $t_0 = 0$ to $t_e = 1/a_V$ on four successively refined grids with $h = \frac{1}{64}, \frac{1}{128}, \frac{1}{256}, \frac{1}{512}$. Errors and convergence rates should be calculated via Richardson extrapolation; see Section 5.4 in [Zhang et. al. 2012 *SIAM Journal on Scientific Computing* 34(2):B179-B201].

1.3 Taylor vortex

The exact solution of the INSE in this test is

$$\begin{aligned}\mathbf{u}(x,y,t) &= 1 + 2\exp(-8\pi^2\nu t)\begin{pmatrix} -\cos(2\pi(x-t))\sin(2\pi(y-t)) \\ \sin(2\pi(x-t))\cos(2\pi(y-t))\end{pmatrix} \\ p(x,y,t) &= -\exp(-16\pi^2\nu t)\big(\cos(4\pi(x-t)) + \cos(4\pi(y-t))\big).\end{aligned} \tag{3}$$

On the periodic domain $[0,1]^2$, (3) is advanced from $t_0 = 0$ to $t_e = 0.5$ for Courant number $\mathrm{Cr} = 0.75, 1.5$ and Reynolds numbers $\mathrm{Re} = \frac{U_0 L_0}{\nu} = 30, 300, 3000, 30000$ on four successively refined grids with $h = \frac{1}{64}, \frac{1}{128}, \frac{1}{256}, \frac{1}{512}$. The initial cell-averaged velocity is calculated by (3) and a sixth-order Newton-Cotes formula.

2 Extra credit and others

The total number of points is 100.

However, if the story in your report is extremely good in terms of software design, interesting new methods/tests, and insights that relate mathematical theory to numerical results, you can earn extra credit up to 15 points.

If you further implement the GePUP-SAV-DIRK schemes in Section 12.5.6 and test your programs appropriately using test in [Zhang 2016 Journal of Scientific Computing 67(3):1134-1180], you can earn extra credit up to 50 points.

In case of this project being way too difficult for you, here are some tradeoff rules to make your life easier:

- Instead of writing your own multigrid solver, you may use a third-party linear solver package such as `lapack`; but 15 points will be deducted from your total score.
- Instead of `C` or `C++`, you may use another programming language such as `matlab` or `python`, but 20 points will be deducted from your total score.

3 How to submit

Your submission must contain

(a) the LaTeX source code and its Makefile so that the command "`make story`" generates a document that contains the story required in Section 1,

(b) a `C++` package so that the command "`make run`" would trigger the compilation of your source code, the production of the executable, the running of your tests, the display of test results, and even the generation of the elements in your story.

You should archive your source code in a single gzipped tar ball (**format**: `YourName_finalProject.tar.gz`) and send it to `sxsj2023@126.com`. A number of tips are given as follows.

(i) You can use either `GNU Make` or `cmake` or a mixture of them.

(ii) You may use either `GNU plot` or `matlab` to plot your results.

(iii) You can use Chinese or English for writing the story document.

(iv) Your gzipped tar ball should neither contain anything that can be generated from your Makefile, nor contain anything irrelevant to this homework. In other words, your answers to this project should be both *sufficient* and *necessary*.

(v) You are encouraged to use a unit test framework such as `CppUnit`; of course you can choose your own unit-test framework as you see fit.