Adaptive time-stepping algorithms for the scalar auxiliary variable scheme of Navier-Stokes equations

Hongtao Chen¹,² and Weilong Wang¹,²

Abstract
Adaptive time stepping is an important tool for controlling the simulation error and enhancing its efficiency in computational fluid dynamics. In this paper, we introduce two adaptive time-stepping algorithms based on the scalar auxiliary variable scheme for the Navier-Stokes equation. One way is to take equidistribution of energy decay to control the variance of the time steps, and in the other way, the relative error is involved to produce time steps. The numerical experiments show that the computational time is significantly saved by these methods.

Keywords
Navier-Stokes equation, scalar auxiliary variable approach, adaptive time-stepping method, energy equidistribution, relative error

AMS subject classifications: 65M60, 76M10

Received 1 June 2020; Revised received 1 September 2020; accepted 28 March 2022

Introduction
We consider an incompressible flow contained in some domain \( \Omega \) in two dimensions, whose boundary is denoted by \( \partial \Omega \). The dynamics of the flow is described by the incompressible Navier-Stokes equations, given in a non-dimensional form as follows

\[
\begin{align*}
\frac{\partial u}{\partial t} + (u \cdot \nabla)u - \nu \nabla^2 u + \nabla p &= f, & \text{in } \Omega \times [0, T], \\
\nabla \cdot u &= 0, & \text{in } \Omega \times [0, T], \\
\frac{\partial u}{\partial t} &= g, & \text{on } \partial \Omega \times [0, T],
\end{align*}
\]

(1)

where \( u : \Omega \times [0, T] \to \mathbb{R}^2 \) is the fluid velocity, \( \mathbf{b} : \partial \Omega \times [0, T] \to \mathbb{R}^2 \) is the boundary velocity and \( p : \Omega \times [0, T] \to \mathbb{R} \) is the fluid pressure. The body force \( f \) is known, and \( \nu \) is the kinematic viscosity of the fluid. The system is supplemented by the initial condition

\[
u(x, 0) = u_0(x), \tag{2}\]

and the following often-used condition to fix the pressure values

\[
\int_{\Omega} p = 0. \tag{3}
\]

There have been extensive works for the Navier-Stokes equations. Semi-implicit splitting type schemes de-couple the computations for the velocity and the pressure fields and enjoy the constant coefficient matrices, but they are only conditionally stable (see Guermond et al.¹ and the references therein). To alleviate the time step size constraint, unconditionally energy-stable schemes are preferred, but involve the solution of a system of non-linear algebraic equations or a system of linear algebraic equations with a variable and time-dependent coefficient matrix (see e.g. Labovsky et al.² and Sanderse³).

Recently, by introducing a scalar auxiliary energy variable the authors in⁴ reformulated the incompressible Navier-Stokes equations (1) to (3) into an equivalent gradient-flow system including a linear constant coefficient system and a nonlinear algebraic equation. However, the cost by solving the above nonlinear equation is negligible since it is about a scalar number, not a field function. Numerical experiments show that the computational cost of this algorithm per time step is approximately twice that of the semi-implicit scheme, but the above auxiliary energy is proven to be unconditionally stable in⁴, which allows time steps based on the accuracy requirement only.

¹School of Mathematical Sciences, Xiamen University, Fujian, P.R. China ²Fujian Provincial Key Laboratory on Mathematical Modeling and High Performance Scientific Computing, Xiamen University, Fujian, P.R. China

Corresponding author:
Hongtao Chen, School of Mathematical Sciences, Xiamen University, Xiamen, Fujian, 361005, P.R. China.
Email: chenht@xmu.edu.cn
In many gradient-flow systems, energy evolution may undergo large and small variations at different time intervals. By enjoying the unconditionally energy stability, the scalar auxiliary variable (SAV) methods can be easily combined with an adaptive strategy. Hence, small time steps are only used when the energy variation is large, while larger time steps can be used when the energy variation is small (see e.g. Cheng and Shen\textsuperscript{3}). Unlike applying variable time steps to first-order schemes without affecting the unconditional stability, we shall construct a new SAV scheme by modifying the scheme in\textsuperscript{4} in order to preserve unconditional stability for second-order schemes with variable time steps.

The above adaptive time-stepping technique was developed based on the energy derivative in time in\textsuperscript{6} and also used in\textsuperscript{7} for the Cahn-Hilliard problem. In Xu and Tang\textsuperscript{8}, based on the stabilisation method there, one can use a large time step in the computation. The spectral deferred correction methods were proposed in\textsuperscript{9} and applied in\textsuperscript{10} to solve the phase field problems. Recently, adaptive time-stepping methods based on the energy were proposed in\textsuperscript{11,12} but only first-order accurate in time. Hence, we shall propose the first adaptive SAV time-stepping algorithm for the above reformulated Navier-Stokes problem (see Algorithm 1), which are based on the auxiliary energy and second-order accurate in time.

Methods to produce time steps in simulations by other ways is also widely considered. The authors in\textsuperscript{3} presented a multiple SAV (MSAV) scheme for the phase field equations, which are extremely efficient when coupled with adaptive time-stepping. Similarly based on the relative errors for the first- and second-order schemes, the second adaptive time stepping algorithm will be presented to shorten the simulation time (see Algorithm 2). Finally, we shall use the numerical examples to illustrate the algorithms.

Algorithm 1. SAV adaptive time-stepping method: Based on energy

Ensure:
1. Set $\Delta t_{\min}$ and $\Delta t_{\max}$, compute the problem (8) or (9) with the minimal time step $\Delta t_{min}$ for some steps. Set $\delta E$ as the average of the energy decay and set $m = 1$.
2. At the time $t_n$, compute the time step with formula (23), and then the problem (8) or (9) with time step $\Delta t_n = \min(\Delta t_{max}, \max(\Delta t_n, \Delta t_{\min}))$.
3. Check if energy decay of two successive steps $\leq \delta E$ and $\Delta t_n = \Delta t_{\min}$. If neither, return to Step 2 with time step $\Delta t_n = \min(\Delta t_{max}, \max(\Delta t_n/2, \Delta t_{\min}))$.
4. If $\Delta t_n = \Delta t_{\min}$, adjust $\delta E$ with an arithmetic average
   $$\delta E = \frac{m \times \delta E + \text{energy decay}}{m + 1},$$
   and set $m = m + 1$.
5. Set $t_{n+1} = t_n + \Delta t_n$, set $n = n + 1$ and go to Step 2.

Algorithm 2. SAV adaptive time-stepping method: based on error

Ensure:
1. Set $\Delta t_{\min}$ and $\Delta t_{\max}$, compute the problem (8) with the minimal time step $\Delta t_{\min}$ for some steps. Set tol and $\rho$.
2. At the time $t_n$, compute $(u_f, p_f)^{p+1}$ by the first order scheme (8) with $\Delta t_n$.
3. Compute $(u_s, p_s)^{p+1}$ by the second order schemes (9) with $\Delta t_{n+1}$ and $\Delta t_n$.
4. Calculate $e_{n+1}$ by formula (24).
5. If $e_{n+1} \leq \text{tol}$ return to Step 2 with time step $\Delta t_n = \max(\Delta t_{\min}, \min(\Delta t_{max}, \text{Adp}(e_{n+1}, \Delta t_n)))$, else update time step $\Delta t_n = \max(\Delta t_{\min}, \min(\Delta t_{max}, \text{Adp}(e_{n+1}, \Delta t_n)))$.
6. Set $t_{n+1} = t_n + \Delta t_n$, set $n = n + 1$ and go to Step 2.
(i) Let \( \{ t_n \} \) be an increasing sequence with \( t_0 = 0 \), and set \( \Delta t_n = t_{n+1} - t_n \). We set the ghost and initial condition as

\[
\Delta t_{-1} = \Delta t_0, \quad u^{-1} = u^0 = u_0, \quad \text{and} \quad R^{-1} = R^0 = \sqrt{E(0)} = \sqrt{c_0 + \int \frac{1}{2} |u^0|^2 \, dx}.
\]

Furthermore, \( \partial_t \)

If each time step is a constant, that is, \( \Delta t_n = \Delta t = \delta t \), we have

\[
\Delta t_{-1} = \Delta t_0 = \Delta t = \delta t, \quad u^{-1} = u^0 = u_0, \quad \text{and} \quad R^{-1} = R^0 = \sqrt{E(0)} = \sqrt{c_0 + \int \frac{1}{2} |u^0|^2 \, dx}.
\]

(ii) Then given \( (u^{n-1}, R^{n-1}) \) and \( (u^n, p^n, R^n) \), we can compute \( (u^{n+1}, p^{n+1}, R^{n+1}) \) through the first-order scheme:

\[
\left\{ \begin{array}{l}
\frac{u^{n+1} - u^n}{\Delta t} + \frac{R^{n+1}}{\sqrt{E^{n+1}}} (u^n \cdot \nabla) u^n - \nu \nabla^2 u^{n+1} + \nabla p^{n+1} = f(t_{n+1}), \\
\nabla \cdot u^{n+1} = 0,
\end{array} \right.
\]

\[
2R^{n+1} \frac{R^{n+1} - R^n}{\Delta t} = - \int_\Omega (\frac{\gamma_1 u^{n+1} - \gamma_2 u^n + \gamma_3 u^{n-1}}{\Delta t_n + \Delta t_{n-1}}) \cdot \nabla u^n - \int_{\partial \Omega} (u^{n+1} \cdot \mathbf{n}) \frac{1}{2} |u^{n+1}|^2
\]

or the second-order scheme:

\[
\left\{ \begin{array}{l}
\frac{\gamma_1 u^{n+1} - \gamma_2 u^n + \gamma_3 u^{n-1}}{\Delta t_n + \Delta t_{n-1}} + \frac{R^{n+1}}{\sqrt{E^{n+1}}} (u^n \cdot \nabla) u^n - \nu \nabla^2 u^{n+1} + \nabla p^{n+1} = f(t_{n+1}), \\
\nabla \cdot u^{n+1} = 0,
\end{array} \right.
\]

\[
2R^{n+1} \frac{R^{n+1} - R^n + \gamma_1 R^{n-1}}{\Delta t_n + \Delta t_{n-1}} = - \int_\Omega (\frac{\gamma_1 u^{n+1} - \gamma_2 u^n + \gamma_3 u^{n-1}}{\Delta t_n + \Delta t_{n-1}}) \cdot \nabla u^n + \int_{\partial \Omega} (u^{n+1} \cdot \mathbf{n}) \frac{1}{2} |u^{n+1}|^2
\]

with

\[
u^n = \frac{\Delta t_n + \Delta t_{n-1}}{\Delta t_n - \Delta t_{n-1}} u^n - \frac{\Delta t_n}{\Delta t_{n-1}} u^{n-1},
\]

and

\[
\gamma_1 = \frac{2 \Delta t_n + \Delta t_{n-1}}{\Delta t_n}, \quad \gamma_2 = \frac{(\Delta t_n + \Delta t_{n-1})^2}{\Delta t_n \Delta t_{n-1}}, \quad \gamma_3 = \frac{\Delta t_n}{\Delta t_{n-1}},
\]

with the boundary condition to be \( u^{n+1} = g(t_{n+1}) \) on \( \partial \Omega \).

Remark 0.1: If \( \nu \) denotes a generic variable, then in the above equations \( \frac{\gamma_1 u^{n+1} - \gamma_2 u^n + \gamma_3 u^{n-1}}{\Delta t_n + \Delta t_{n-1}} \) is the second order approximation of \( \partial_t g(t_{n+1}) \) with \( \gamma_i, i = 1, 2, 3 \) defined specifically by (11). Furthermore, \( \nu^n \) defined by (10) denotes a second-order explicit approximation of \( g(t_{n+1}) \).

Remark 0.2: If each time step is a constant, that is, \( \Delta t_{n-1} = \Delta t_n = \Delta t = \delta t \), we have

\[
\gamma_1 = \frac{3 \delta t^{n+1} - 4 \delta t^n + \delta t^{n-1}}{2 \delta t}, \quad \nu^n = 2 u^n - u^{n-1}.
\]

Thus the schemes (8) and (9) go back to the corresponding ones in 4.

The main advantage for the above schemes (8) and (9) lies in that they both enjoy the following unconditional energy stability. In the subsequent analysis, we let \( (u, v) \) and \( \langle u, v \rangle \) denote the \( L^2 \) inner product of \( u, v \) in the domain \( \Omega \) and \( \partial \Omega \), respectively, and the norm \( \| v \| = \sqrt{\langle v, v \rangle} \).

**Lemma 0.1:** 4 Theorem 2.1 In the absence of the external force \( f \) and with zero boundary velocity \( g \), the first-order scheme (8) satisfies the following property:

\[
Q^{n+1} - Q^n = - ||D^{n+1}||^2 - \nu \Delta t \| \nabla u^{n+1} \|^2,
\]

where

\[
Q^n = |R^n|^2, \quad D^{n+1} = R^{n+1} - R^n.
\]

The proof for the first-order scheme (8) was blackpro-vided by Lin et al.4 Theorem 2.1, thus we only give a proof for the second-order scheme (9) as follows.

**Lemma 0.2:** In the absence of the external force \( f \) and with zero boundary velocity \( g \), the second scheme (9) to (11) satisfies the following property:
\[ \dot{Q}^{n+1} - \dot{Q}^n = -\|D^{n+1}\|^2 - \nu(\Delta t_n + \Delta t_{n-1}) \|\nabla u^{n+1}\|^2, \] (14)

where

\[ \dot{Q}^n = \left[ \sqrt{\frac{t_n}{2}} - \sqrt{t_{n-1} - 4} R^n \right]^2 + \left[ \sqrt{\frac{t_{n-1}}{2}} - \sqrt{\frac{t_0}{2} - 4} R^{n-1} \right]^2, \]
\[ \dot{D}^{n+1} = \frac{2t_n}{\sqrt{t_n - 4}} R^n - \frac{2t_{n-1}}{\sqrt{t_{n-1} - 4}} R^{n-1}. \] (15)

Proof.: First, taking the inner product of the first equation in (11) with \( u^{n+1} \) leads to

\[
\begin{align*}
&\left( \gamma_1 u^{n+1} + \gamma_2 u^n + \gamma_3 u^{n-1} \right) + \frac{R^{n+1}}{\sqrt{E^{n+1}}} (\nabla u^n, u^{n+1}) \\
&\quad + \langle u^{n+1}, \nu \rangle_{n} > - \nu \| \nabla u^{n+1} \|_2^2 \\
&= (f(t_{n+1}), u^{n+1}),
\end{align*}
\]

where we used the divergence free condition. Then we sum up the above equation and the third equation in (11) to obtain

\[ 2R^{n+1} \frac{\gamma_1 R^{n+1} - \gamma_2 R^n + \gamma_3 R^{n-1}}{\Delta t_n + \Delta t_{n-1}} = -\nu \| \nabla u^{n+1} \|^2 - \frac{1}{2} \| g(t_{n+1}) \|^2_2, \]

Finally, combined with the following relation

\[ 2R^{n+1} (\gamma_1 R^{n+1} - \gamma_2 R^n + \gamma_3 R^{n-1}) = \dot{Q}^{n+1} - \dot{Q}^n + \|\dot{D}^{n+1}\|^2, \]

where we used the definitions (15), we obtain the desired stability result (14). \( \square \)

**Solution algorithm and implementation**

Even though at first glance both the schemes (8) and (9) look like coupled and implicit systems of equations, in fact they can be implemented in a very efficient way (cf. Lin et al.\(^4\)), thanks to the fact that \( R^{n+1} \) and \( E^{n+1} \) are scalar numbers, not field functions. We next only explain the details to implement the scheme (8), since the ones for the scheme (9) are similar.

First, we introduce a auxiliary variable \( S \) to separate \( u^{n+1} \) and \( p^{n+1} \) as follows

\[ S = \frac{R^{n+1}}{\sqrt{E^{n+1}}}, \quad u^{n+1} = u_1^{n+1} + Su_2^{n+1}, \]
\[ p^{n+1} = p_1^{n+1} + Sp_2^{n+1}. \]

Then from (4) we can see that

\[ E^{n+1} = E(S) = c_0 + \int_\Omega \frac{1}{2} |u^{n+1}|^2 \]
\[ = A_0 + A_1 S + A_2 S^2, \] (17)

with

\[ A_0 = c_0 + \int_\Omega \frac{1}{2} |u_1^{n+1}|^2, \quad A_1 = \int_\Omega u_1^{n+1} \cdot u_2^{n+1}, \]
\[ A_2 = \int_\Omega \frac{1}{2} |u_2^{n+1}|^2. \] (18)

**Implementation of algorithm:**

(a) By substituting (16) into (8), we can separate (8) by two linear subproblems, from which we can solve \( \{ u_i^{n+1}, p_i^{n+1} \}, i = 1, 2 \), respectively

\[
\begin{align*}
\frac{u_1^{n+1} - u_1^n}{\Delta t_n} - \nu \nabla^2 u_1^{n+1} + \nabla p_1^{n+1} &= f(t_{n+1}), \quad \text{in } \Omega, \\
\nabla \cdot u_1^{n+1} &= 0, \quad \text{in } \Omega, \\
u \nabla u_1^{n+1} &= g(t_{n+1}), \quad \text{on } \partial \Omega, 
\end{align*}
\]

and

\[
\begin{align*}
\frac{u_2^{n+1} - u_2^n}{\Delta t_n} - \nu \nabla^2 u_2^{n+1} + (u^n \cdot \nabla) u_2^n + \nabla p_2^{n+1} &= 0, \quad \text{in } \Omega, \\
\nabla \cdot u_2^{n+1} &= 0, \quad \text{in } \Omega, \\
u \nabla u_2^{n+1} &= g(t_{n+1}), \quad \text{on } \partial \Omega. 
\end{align*}
\] (19)

(b) Substituting (16) into the third equation of (8), then the unknown \( S \) can be obtained by solving the following algebraic equation

\[ F(S) = \frac{2}{\Delta t} E^{n+1} - \frac{2R^n}{\Delta t} S^2 \sqrt{E^{n+1}} \]
\[ + D_0 S + D_1 S^2 + D_2 S^3 = 0, \] (21)

with

\[
\begin{align*}
D_0 &= \nu \| \nabla u_1^{n+1} \|^2_2 - (f(t_{n+1}), u_1^{n+1}) + \frac{1}{2} |g(t_{n+1})|^2_2, \\
D_1 &= 2\nu (\nabla u_1^{n+1}, \nabla u_2^{n+1}) - (f(t_{n+1}), u_2^{n+1}) + \frac{1}{2} |g(t_{n+1})|^2_2, \\
D_2 &= \nu \| \nabla u_2^{n+1} \|^2_2. 
\end{align*}
\] (22)
Here we can use the Newton’s method to solve (21) with the initial guess $S = 1$, and the cost of the computation is very small compared to the total cost within a time step since it is a scalar nonlinear algebraic equation for $S$.

(c) With $S$ known, we can compute the scalar function $E^{n+1}$ from (17) and $\{R^{n+1}, u^{n+1}, p^{n+1}\}$ from (16).

### Adaptive time-stepping algorithm

We introduce two adaptive time-stepping methods in this section. First, we shall show how to produce the adaptive time step. Then we proposed the adaptive time-stepping algorithms.

1. **An adaptive time step: based on energy:** We first use the energy decay to control the variance of the time steps. A transform of discrete energy identities (12) and (14) are

   $$\Delta t_n = - \frac{Q^{n+1} - Q^n + ||D_{n+1}^e||^2}{\nu ||\nabla u^n||^2}$$

   and

   $$\Delta t_n = - \frac{\nabla u^{n+1} - \nabla u^n + ||D_{n+1}^p||^2}{\nu ||\nabla u^n||^2} - \Delta t_{n-1}$$

   for the first- and second-order schemes, respectively. As we aim at equidistribution of the energy decay, we allow the numerator $||Q^{n+1} - Q^n + ||D_{n+1}^e||^2||$ or $\nabla u^{n+1} - \nabla u^n + ||D_{n+1}^p||^2$ to be a constant $\delta E$. Thus the time step turns to

   $$\Delta t_n = \frac{\delta E}{\nu ||\nabla u^n||^2}$$

   or

   $$\Delta t_n = \frac{\delta E}{\nu ||\nabla u^n||^2} - \Delta t_{n-1}.$$  

   But we find there is still an unknown term $u_{n+1}$ in the denominator. We therefore replace $u_{n+1}$ by $u^n$ (or $u^n$ defined by (10)) for the first- (or second-) order scheme to get a computable adaptive time step

   $$\Delta t_n = \frac{\delta E}{\nu ||\nabla u^n||^2},$$

   or

   $$\Delta t_n = \frac{\delta E}{\nu ||\nabla u^n||^2} - \Delta t_{n-1}. \quad (23)$$

2. **An adaptive time step: based on error:** Given the solutions at $t_{n-1}$ and $t_n$, we compute $(u_f, p_f, R_f)^{n+1}$ by the first-order scheme (8) and $(u_s, p_s, R_s)^{n+1}$ by the second-order schemes (9), respectively. Then we can use the following relative error

   $$e_{n+1} = \max \left\{ \frac{||u_{f}^{n+1} - u_{f}^{n+1}||}{||u_{f}^{n+1}||}, \frac{||p_{f}^{n+1} - p_{f}^{n+1}||}{||p_{f}^{n+1}||} \right\} \quad (24)$$

   as the indicator to control the time step. One simple but effective strategy is to update the time step size by using the formula

   $$A_{d}(\varepsilon, \Delta t) = \rho \left( \frac{tol}{\varepsilon} \right)^{1/2} \Delta t,$$

   where $\varepsilon$ is a relative error, $\Delta t$ is the time step, $tol$ is the error tolerance and $\rho$ is an artificial parameter.
Figure 1. The contour of the Taylor-Green vortex problem using first-order scheme in Algorithm 1: (a) velocity (uniform step $\Delta t = 0.001$), (b) pressure (uniform step $\Delta t = 0.001$), (c) velocity (adaptive step $\Delta t_{\text{min}} = 0.001, \Delta t_{\text{max}} = 0.015$) and (d) pressure (adaptive step $\Delta t_{\text{min}} = 0.001, \Delta t_{\text{max}} = 0.015$).

Figure 2. The contour of the Taylor-Green vortex problem using second-order scheme in Algorithm 1: (a) velocity (uniform step $\Delta t = 0.001$), (b) pressure (uniform step $\Delta t = 0.001$), (c) velocity (adaptive step $\Delta t_{\text{min}} = 0.001, \Delta t_{\text{max}} = 0.1$) and (d) pressure (adaptive step $\Delta t_{\text{min}} = 0.001, \Delta t_{\text{max}} = 0.1$).

Figure 3. The contour of the Taylor-Green vortex problem using Algorithm 2: (a) velocity (adaptive step $\Delta t_{\text{min}} = 0.001, \Delta t_{\text{max}} = 0.1$) and (b) pressure (adaptive step $\Delta t_{\text{min}} = 0.001, \Delta t_{\text{max}} = 0.1$).
Figure 4. The evolution in Algorithm 1: (a) energy decay, (b) the evolution of time steps and (c) CPU time.

Figure 5. The evolution in Algorithm 2: (a) relative error between first-order and second-order schemes, (b) the evolution of time steps and (c) CPU time.

Figure 6. The evolution in Algorithm 1: (a) streamlines and (b) the evolution of time steps.
time variable in equation (1) and ‘CPU time’ labelled in the vertical coordinate-axis is the computation cost, from which we can find both adaptive methods do actually save computational time and only take less than one-third of the time of the uniform method.

Lid-driven cavity flow

In order to show validity of the adaptive SAV method, we test lid-driven cavity flow problem. The computational domain is the unite square domain, and the boundary conditions are

\[
\begin{align*}
  \mathbf{u} &= (0, 0), \quad \text{on } x = \pm 1 \text{ and } y = -1, \\
  \mathbf{u} &= (1, 0), \quad \text{on } y = 1.
\end{align*}
\]

(25)

We choose the \( \nu = 1/100, \Delta t_{\text{min}} = 10^{-6}, \Delta t_{\text{max}} = 5.0 \times 10^{-3}, h = 1/4000. \) The streamlines and the time steps by Algorithm 1 are shown in Figure 6.

Then we choose the \( \nu = 1/1000, \Delta t_{\text{min}} = 10^{-6}, \Delta t_{\text{max}} = 5.0 \times 10^{-3}, h = 1/4000. \) The streamlines and the time steps by Algorithm 1 are shown in Figure 7.

From the above figures, it can be found that at the beginning the accuracy can be guaranteed only when the time step is very small. With the development of time, the time step can be larger, which indeed saves the computation time compared with the constant time step.

Declaration of conflicting interests

The authors declared no potential conflicts of interest with respect to the research, authorship and/or publication of this article.

Funding

This work is in part supported by the National Natural Science Foundation of China (no. 11871410), the Natural Science Foundation of Fujian Province of China (nos. 2021J01034, 2018J01004), the Fundamental Research Funds for the Central Universities (no. 20720180001).

References

1. Guermond J, Minev P and Shen J. An overview of projection methods for incompressible flows. Comput Methods Appl Mech Eng 2006; 195: 6011–6045.
2. Labovsky A, Layton W, Manica C et al. The stabilized extrapolated trapezoidal finite-element method for the Navier-Stokes equations. Comput Methods Appl Mech Eng 2009; 198: 958–974.
3. Sanderse B. Energy-conserving Runge-Kutta methods for the incompressible Navier-Stokes equations. J Comput Phys 2013; 233: 100–131.
4. Lin L, Yang Z and Dong S. Numerical approximation of incompressible Navier-Stokes equations based on an auxiliary energy variable. J Comput Phys 2019; 388: 1–22.
5. Cheng Q and Shen J. Multiple scalar auxiliary variable (MSAV) approach and its application to the phase-field vesicle membrane model. SIAM J Sci Comput 2018; 40(6): 3982–4006.
6. Qiao Z, Zhang Z and Tang T. An adaptive time-stepping strategy for the molecular beam epitaxy models. SIAM J Sci Comput 2011; 33(3): 1395–1414.
7. Zhang Z and Qiao Z. An adaptive time-stepping strategy for the Cahn-Hilliard equation. Commun Comput Phys 2012; 11: 1261–1278.
8. Xu C and Tang T. Stability analysis of large time-stepping methods for epitaxial growth models. SIAM J Numer Anal 2006; 44: 1759–1779.
9. Tang T, Xie H and Yin X. High-order convergence of spectral deferred correction methods on general quadrature nodes. J Sci Comput 2013; 59(1): 1–13.
10. Feng X, Tang T and Yang J. Long time numerical simulations for phase-field problems using p-adaptive spectral deferred correction methods. SIAM J Sci Comput 2015; 37(1): 271–294.
11. Luo F, Tang T and Xie H. Parameter-free time adaptivity based on energy evolution for the Cahn-Hilliard equation. Commun Comput Phys 2016; 19(5): 1542–1563.

ORCID iD

Hongtao Chen https://orcid.org/0000-0002-3503-3975

Figure 7. The evolution in Algorithm 2: (a) streamlines, (b) the evolution of time steps when 0 \( \leq t \leq 245 \) and (c) the evolution of time steps when 0 \( \leq t \leq 0.06. \)
12. Luo F, Xie H, Xie M et al. Adaptive time-stepping algorithms for molecular beam epitaxy: Based on energy or roughness. *Appl Math Lett* 2020; 99: 105991.

13. Girault V and Raviart P. *Finite Element Methods for Navier-Stokes Equations*. Berlin Heidelberg: Springer-Verlag, 1986.

14. Chorin A. The numerical solution of the Navier-Stokes equations for an incompressible fluid. *Bull Amer Math Soc* 1967; 73(6): 928–931.

15. DeCaria V, Layton W and Zhao H. A time-accurate, adaptive discretization for fluid flow problems. *Int J Numer Anal Model* 2020; 17(2): 254–280.