Linearly Preconditioned Nonlinear Conjugate Gradient Solvers for the Epitaxial Thin Film Equation with Slope Selection

Wenqiang Feng* Cheng Wang† Steven M. Wise‡

June 7, 2017

Abstract

In this paper we present two efficient and practical Preconditioned Nonlinear Conjugate Gradient (PNCG) solvers for the nonlinear elliptic PDE systems. The main idea of the preconditioned solvers is to use a linearized version of the nonlinear operator as a pre-conditioner, or in other words, as a metric for choosing the search direction. In order to make the proposed solvers and scheme much more practical, we also investigate the adaptive time stepping strategy. Numerical simulations and comparisons for the important physical application problem – epitaxial thin film equation with slope selection– are carried out to verify the efficiency of the solvers.

1 Introduction

Many unconditionally energy stable schemes for the physical models will lead to a highly nonlinear elliptic PDE systems which arise from time discretization of parabolic equations.

Some review references for NCG [2, 3, 9, 10, 11, 17].

The remainder of the paper is organized as follows. In Section 2, we present the discrete spatial difference operators, function space, inner products and norms for the finite difference method. In Section 3, we introduce the SS model, recall the first-order-in-time fully discrete finite difference scheme in [6, 16] and prove that the fully discrete scheme is unconditionally stable. The PNCG solvers and the adaptive time-stepping strategy are proposed in Section 4 and Section 5, respectively. Finally, numerical experiments are presented Section 6, and some concluding remarks are given in Section 7.

2 The Discrete Spatial Difference Operators, Function Space, Inner Products and Norms in 2D

sec:fdm-pcg

In this section we define the discrete spatial difference operators, function space, inner products and norms, following the notation used in [5, 6, 16, 18, 19, 20]. Let $\Omega = (0, L_x) \times (0, L_y)$, where, for simplicity, we assume $L_x = L_y =: L > 0$. We write $L = m \cdot h$, where m is a positive integer. The parameter $h = \frac{L}{m}$ is called the mesh or grid spacing. We define the following two uniform, infinite grids with grid spacing $h > 0$:

$$E := \{x_{i+\frac{1}{2}} \mid i \in \mathbb{Z}\}, \quad C := \{x_i \mid i \in \mathbb{Z}\},$$

*Department of Mathematics, The University of Tennessee, Knoxville, TN 37996 (wfeng1@utk.edu)

†Department of Mathematics, The University of Massachusetts, North Dartmouth, MA 02747 (cwang1@umassd.edu)

‡Department of Mathematics, The University of Tennessee, Knoxville, TN 37996 (swise1@utk.edu)

where $x_i = x(i) := (i - \frac{1}{2}) \cdot h$. Consider the following 2D discrete periodic function spaces:

$$\begin{aligned}
\mathcal{V}_{\text{per}} &:= \left\{ \nu : E \times E \to \mathbb{R} \;\middle|\; \nu_{i+\frac{1}{2},j+\frac{1}{2}} = \nu_{i+\frac{1}{2}+\alpha m, j+\frac{1}{2}+\beta m},\ \forall\, i,j,\alpha,\beta \in \mathbb{Z} \right\}, \\
\mathcal{C}_{\text{per}} &:= \left\{ \nu : C \times C \to \mathbb{R} \;\middle|\; \nu_{i,j} = \nu_{i+\alpha m, j+\beta m},\ \forall\, i,j,\alpha,\beta \in \mathbb{Z} \right\}, \\
\mathcal{E}^{\text{ew}}_{\text{per}} &:= \left\{ \nu : E \times C \to \mathbb{R} \;\middle|\; \nu_{i+\frac{1}{2},j} = \nu_{i+\frac{1}{2}+\alpha m, j+\beta m},\ \forall\, i,j,\alpha,\beta \in \mathbb{Z} \right\}, \\
\mathcal{E}^{\text{ns}}_{\text{per}} &:= \left\{ \nu : C \times E \to \mathbb{R} \;\middle|\; \nu_{i,j+\frac{1}{2}} = \nu_{i+\alpha m, j+\frac{1}{2}+\beta m},\ \forall\, i,j,\alpha,\beta \in \mathbb{Z} \right\}.
\end{aligned}$$

The functions of $\mathcal{V}_{\text{per}}$ are called *vertex centered functions*; those of $\mathcal{C}_{\text{per}}$ are called *cell centered functions*. The functions of $\mathcal{E}^{\text{ew}}_{\text{per}}$ are called *east-west edge-centered functions*, and the functions of $\mathcal{E}^{\text{ns}}_{\text{per}}$ are called *north-south edge-centered functions*. We also define the mean zero space

$$\mathring{\mathcal{C}}_{\text{per}} := \left\{ \nu \in \mathcal{C}_{\text{per}} \;\middle|\; \frac{h^2}{|\Omega|} \sum_{i,j=1}^{m} \nu_{i,j} =: \overline{\nu} = 0 \right\}.$$

We now define the important difference and average operators on the spaces:

$$\begin{aligned}
A_x \nu_{i+\frac{1}{2},\square} &:= \frac{1}{2}\left(\nu_{i+1,\square} + \nu_{i,\square}\right), \quad D_x \nu_{i+\frac{1}{2},\square} := \frac{1}{h}\left(\nu_{i+1,\square} - \nu_{i,\square}\right), \\
A_y \nu_{\square,i+\frac{1}{2}} &:= \frac{1}{2}\left(\nu_{\square,i+1} + \nu_{\square,i}\right), \quad D_y \nu_{\square,i+\frac{1}{2}} := \frac{1}{h}\left(\nu_{\square,i+1} - \nu_{\square,i}\right),
\end{aligned}$$

with $A_x,\ D_x : \mathcal{C}_{\text{per}} \to \mathcal{E}^{\text{ew}}_{\text{per}}$ if $\square$ is an integer, and $A_x,\ D_x : \mathcal{E}^{\text{ns}}_{\text{per}} \to \mathcal{V}_{\text{per}}$ if $\square$ is a half-integer, with $A_y,\ D_y : \mathcal{C}_{\text{per}} \to \mathcal{E}^{\text{ns}}_{\text{per}}$ if $\square$ is an integer, and $A_y,\ D_y : \mathcal{E}^{\text{ew}}_{\text{per}} \to \mathcal{V}_{\text{per}}$ if $\square$ is a half-integer. Likewise,

$$\begin{aligned}
a_x \nu_{i,\square} &:= \frac{1}{2}\left(\nu_{i+\frac{1}{2},\square} + \nu_{i-\frac{1}{2},\square}\right), \quad d_x \nu_{i,\square} := \frac{1}{h}\left(\nu_{i+\frac{1}{2},\square} - \nu_{i-\frac{1}{2},\square}\right), \\
a_y \nu_{\square,j} &:= \frac{1}{2}\left(\nu_{\square,j+\frac{1}{2}} + \nu_{\square,j-\frac{1}{2}}\right), \quad d_y \nu_{\square,j} := \frac{1}{h}\left(\nu_{\square,j+\frac{1}{2}} - \nu_{\square,j-\frac{1}{2}}\right),
\end{aligned}$$

with $a_x,\ d_x : \mathcal{E}^{\text{ew}}_{\text{per}} \to \mathcal{C}_{\text{per}}$ if $\square$ is an integer, and $a_x,\ d_x : \mathcal{V}_{\text{per}} \to \mathcal{E}^{\text{ns}}_{\text{per}}$ if $\square$ is a half-integer; and with $a_y,\ d_y : \mathcal{E}^{\text{ns}}_{\text{per}} \to \mathcal{C}_{\text{per}}$ if $\square$ is an integer, and $a_y,\ d_y : \mathcal{V}_{\text{per}} \to \mathcal{E}^{\text{ew}}_{\text{per}}$ if $\square$ is a half-integer.

Define the 2D center-to-vertex derivatives $\mathfrak{D}_x,\ \mathfrak{D}_y : \mathcal{C}_{\text{per}} \to \mathcal{V}_{\text{per}}$ component-wise as

$$\begin{aligned}
\mathfrak{D}_x \nu_{i+\frac{1}{2},j+\frac{1}{2}} &:= A_y(D_x\nu)_{i+\frac{1}{2},j+\frac{1}{2}} = D_x(A_y\nu)_{i+\frac{1}{2},j+\frac{1}{2}} \\
&= \frac{1}{2h}\left(\nu_{i+1,j+1} - \nu_{i,j+1} + \nu_{i+1,j} - \nu_{i,j}\right), \\
\mathfrak{D}_y \nu_{i+\frac{1}{2},j+\frac{1}{2}} &:= A_x(D_y\nu)_{i+\frac{1}{2},j+\frac{1}{2}} = D_y(A_x\nu)_{i+\frac{1}{2},j+\frac{1}{2}} \\
&= \frac{1}{2h}\left(\nu_{i+1,j+1} - \nu_{i+1,j} + \nu_{i,j+1} - \nu_{i,j}\right).
\end{aligned}$$

The utility of these definitions is that the differences $\mathfrak{D}_x$ and $\mathfrak{D}_y$ are collocated on the grid, unlike the case for $D_x,\ D_y$. Define the 2D vertex-to-center derivatives $\mathfrak{d}_x,\ \mathfrak{d}_y : \mathcal{V}_{\text{per}} \to \mathcal{C}_{\text{per}}$ component-wise as

$$\begin{aligned}
\mathfrak{d}_x \nu_{i,j} &:= a_y(d_x\nu)_{i,j} = d_x(a_y\nu)_{i,j} \\
&= \frac{1}{2h}\left(\nu_{i+\frac{1}{2},j+\frac{1}{2}} - \nu_{i-\frac{1}{2},j+\frac{1}{2}} + \nu_{i+\frac{1}{2},j-\frac{1}{2}} - \nu_{i-\frac{1}{2},j-\frac{1}{2}}\right), \\
\mathfrak{d}_y \nu_{i,j} &:= a_x(d_y\nu)_{i,j} = d_y(a_x\nu)_{i,j} \\
&= \frac{1}{2h}\left(\nu_{i+\frac{1}{2},j+\frac{1}{2}} - \nu_{i+\frac{1}{2},j-\frac{1}{2}} + \nu_{i-\frac{1}{2},j+\frac{1}{2}} - \nu_{i-\frac{1}{2},j-\frac{1}{2}}\right).
\end{aligned}$$

Now the discrete gradient operator, ∇_h^v: $\mathcal{C}_{\text{per}} \to \mathcal{V}_{\text{per}} \times \mathcal{V}_{\text{per}}$, is defined as

$$\nabla_h^v \nu_{i+\frac{1}{2},j+\frac{1}{2}} := (\mathfrak{D}_x \nu_{i+\frac{1}{2},j+\frac{1}{2}}, \mathfrak{D}_y \nu_{i+\frac{1}{2},j+\frac{1}{2}}).$$

The standard 2D discrete Laplacian, $\Delta_h : \mathcal{C}_{\text{per}} \to \mathcal{C}_{\text{per}}$, is given by

$$\Delta_h \nu_{i,j} := d_x(D_x\nu)_{i,j} + d_y(D_y\nu)_{i,j} = \frac{1}{h^2}\left(\nu_{i+1,j} + \nu_{i-1,j} + \nu_{i,j+1} + \nu_{i,j-1} - 4\nu_{i,j}\right).$$

The 2D vertex-to-vertex average, $\mathcal{A} : \mathcal{V}_{\text{per}} \to \mathcal{C}_{\text{per}}$, is defined to be

$$\mathcal{A}\nu_{i,j} := \frac{1}{4}\left(\nu_{i+1,j} + \nu_{i-1,j} + \nu_{i,j+1} + \nu_{i,j-1}\right).$$

The 2D *skew* Laplacian, $\Delta_h^v : \mathcal{C}_{\text{per}} \to \mathcal{C}_{\text{per}}$, is defined as

$$\begin{aligned}\Delta_h^v \nu_{i,j} &= \mathfrak{d}_x(\mathfrak{D}_x\nu)_{i,j} + \mathfrak{d}_y(\mathfrak{D}_y\nu)_{i,j} \\ &= \frac{1}{2h^2}\left(\nu_{i+1,j+1} + \nu_{i-1,j+1} + \nu_{i+1,j-1} + \nu_{i-1,j-1} - 4\nu_{i,j}\right).\end{aligned}$$

The 2D discrete p-Laplacian operator is defined as

$$\nabla_h^v \cdot \left(|\nabla_h^v \nu|^{p-2}\, \nabla_h^v \nu\right)_{ij} := \mathfrak{d}_x(r\, \mathfrak{D}_x\nu)_{i,j} + \mathfrak{d}_y(r\, \mathfrak{D}_y\nu)_{i,j},$$

with

$$r_{i+\frac{1}{2},j+\frac{1}{2}} := \left[(\mathfrak{D}_x u)^2_{i+\frac{1}{2},j+\frac{1}{2}} + (\mathfrak{D}_y u)^2_{i+\frac{1}{2},j+\frac{1}{2}}\right]^{\frac{p-2}{2}}.$$

Clearly, for $p = 2$, $\Delta_h^v \nu = \nabla_h^v \cdot \left(\left|\nabla_h^v \nu\right|^{p-2} \nabla_h^v \nu\right)$.

Now we are ready to define the following grid inner products:

$$\begin{aligned}(\nu,\xi)_2 &:= h^2 \sum_{i=1}^{m}\sum_{j=1}^{n} \nu_{i,j}\psi_{i,j}, \quad \nu,\, \xi \in \mathcal{C}_{\text{per}}, \\ \langle \nu,\xi\rangle &:= (\mathcal{A}(\nu\xi), 1)_2\,, \quad \nu,\, \xi \in \mathcal{V}_{\text{per}}, \\ [\nu,\xi]_{\text{ew}} &:= (A_x(\nu\xi), 1)_2\,, \quad \nu,\, \xi \in \mathcal{E}^{\text{ew}}_{\text{per}}, \\ [\nu,\xi]_{\text{ns}} &:= (A_y(\nu\xi), 1)_2\,, \quad \nu,\, \xi \in \mathcal{E}^{\text{ns}}_{\text{per}}.\end{aligned}$$

We now define the following norms for cell-centered functions. If $\nu \in \mathcal{C}_{\text{per}}$, then $\|\nu\|_2^2 := (\nu,\nu)_2$; $\|\nu\|_p^p := (|\nu|^p, 1)_2$ $(1 \le p < \infty)$, and $\|\nu\|_\infty := \max_{\substack{1\le i\le m\\ 1\le j\le n}} \left|\nu_{i,j}\right|$. Similarly, we define the gradient norms: for $\nu \in \mathcal{C}_{\text{per}}$,

$$\|\nabla_h^v \nu\|_p^p := \langle |\nabla_h^v \nu|^p, 1\rangle, \quad |\nabla_h^v \nu|^p := [(\mathfrak{D}_x\nu)^2 + (\mathfrak{D}_y\nu)^2]^{\frac{p}{2}} = [\nabla_h^v\nu \cdot \nabla_h^v\nu]^{\frac{p}{2}} \in \mathcal{V}_{\text{per}}, \quad 2 \le p < \infty,$$

and

$$\|\nabla_h \nu\|_2^2 := [D_x\nu, D_x\nu]_{\text{ew}} + [D_y\nu, D_y\nu]_{\text{ns}}\,.$$

3 Epitaxial thin film equation with Slope Selection and the fully discrete scheme

:scheme-pcg

In this section we provide a brief introduction about SS model and recall the first-order-in-time unconditional energy stable numerical schemes in [6, 16].

This epitaxial thin film model was first proposed by P. Aviles and Y. Giga to study the dynamics of smestic liquid crystals in [1]. The energy of the SS model is as follows:

$$F(\phi) := \int_\Omega \left(\frac{1}{4} \left(|\nabla \phi|^2 - 1 \right)^2 + \frac{\varepsilon^2}{2} (\Delta \phi)^2 \right) \mathrm{d}\mathbf{x} \,, \tag{3.1}$$

energy-SS-1

where $\Omega = (0, L_x) \times (0, L_y)$, $\phi : \Omega \to \mathbb{R}$ is a scaled height function of thin film and ε is a constant which represents the width of the rounded corner. The corresponding chemical potential is defined to be the variational derivative of the energy (3.1), *i.e.*,

$$\mu := \delta_\phi F = -\nabla \cdot (|\nabla \phi|^2 \nabla \phi) + \Delta \phi + \varepsilon^2 \Delta^2 \phi. \tag{3.2}$$

chem-pot-SS

And the SS equation becomes the L^2 gradient flow associated with the energy (3.1):

$$\partial_t \phi = -\mu = \nabla \cdot (|\nabla \phi|^2 \nabla \phi) - \Delta \phi - \varepsilon^2 \Delta^2 \phi. \tag{3.3}$$

equation-SS

Periodic boundary conditions are assumed for ϕ and μ in both spatial directions for simplicity.

With the machinery in last section, the first-order-in-time unconditional energy stable scheme in [6, 16] can be formulated as follows: for $n \geq 0$, given $\phi^n \in \mathcal{C}_{\text{per}}$, find $\phi^{n+1} \in \mathcal{C}_{\text{per}}$ such that

$$\phi^{n+1} - s \nabla_h \cdot \left(\left| \nabla_h \phi^{n+1} \right|^2 \nabla^{n+1} \phi \right) + s \epsilon^2 \Delta_h^2 \phi^{n+1} \quad = \quad \phi^n - s \Delta_h \phi^n. \tag{3.4}$$

We now define a fully discrete energy that is consistent with the continuous space energy (3.1) as $h \to 0$. In particular, the discrete energy $F_h : \mathcal{C}_{\text{per}} \to \mathbb{R}$ is defined as:

$$F_h(\phi) = \frac{1}{4} \left\| \nabla_h \phi \right\|_4^4 - \frac{1}{2} \left\| \nabla_h \phi \right\|_2^2 + \frac{1}{2} \varepsilon^2 \left\| \Delta_h \phi \right\|_2^2 . \tag{3.5}$$

Theorem 3.1. *The numerical scheme* (3.4) *is unconditionally energy stable, i.e. the discrete energy F_h satisfies the following energy dissipation law:*

$$F_h(\phi^{n+1}) - F_h(\phi^n) \leq -s \left\| \mu^{n+1} \right\|_2^2 . \tag{3.6}$$

Proof. The numerical scheme (3.4) can be rewritten as a nonlinear system:

$$\begin{aligned} \phi^{n+1} - \phi^n &= -s \mu^{n+1} && (3.7) \\ \mu^{n+1} &= -\nabla_h \cdot (|\nabla_h \phi^{n+1}|^2 \nabla_h \phi^{n+1}) + \Delta_h \phi^n + \varepsilon^2 \Delta_h^2 \phi^{n+1}. && (3.8) \end{aligned}$$

By taking the L^2 inner product of (3.7) with μ^{n+1}, we obtain

$$-s \left\| \mu^{n+1} \right\|_2^2 = \left(\mu^{n+1}, \phi^{n+1} - \phi^n \right). \tag{3.9}$$

By taking the L^2 inner product of (3.8) with $\phi^{n+1} - \phi^n$ yields

$$\begin{aligned} \left(\mu^{n+1}, \phi^{n+1} - \phi^n \right) &= -\left(\nabla_h \cdot (|\nabla_h \phi^{n+1}|^2 \nabla_h \phi^{n+1}), \phi^{n+1} - \phi^n \right) \\ &\quad + \left(\Delta_h \phi^n, \phi^{n+1} - \phi^n \right) + \varepsilon^2 \left(\Delta_h^2 \phi^{n+1}, \phi^{n+1} - \phi^n \right) \\ &=: I_1 + I_2 + I_3. \end{aligned} \tag{3.10}$$

For the term I_1 which involves with 4-Laplacian term, we have

$$\begin{aligned} -\left(\nabla_h \cdot (|\nabla_h \phi^{n+1}|^2 \nabla_h \phi^{n+1}), \phi^{n+1} - \phi^n\right) &= \left(|\nabla_h \phi^{n+1}|^2 \nabla_h \phi^{n+1}, \nabla_h (\phi^{n+1} - \phi^n)\right) \\ &\geq \frac{1}{4}\left(\|\nabla_h \phi^{n+1}\|_4^4 - \|\nabla_h \phi^n\|_4^4\right). \end{aligned} \tag{3.11}$$

For the explicit linear term I_2, we have

$$\begin{aligned} \left(\Delta_h \phi^n, \phi^{n+1} - \phi^n\right) &= -\left(\nabla_h \phi^n, \nabla_h (\phi^{n+1} - \phi^n)\right) \\ &= -\frac{1}{2}\left\|\nabla_h \phi^{n+1}\right\|_2^2 + \frac{1}{2}\left\|\nabla_h \phi^n\right\|_2^2 + \frac{1}{2}\left\|\nabla_h (\phi^{n+1} - \phi^n)\right\|_2^2. \end{aligned} \tag{3.12}$$

For the highest-order diffusion term I_3, we have

$$\begin{aligned} \varepsilon^2\left(\Delta_h^2 \phi^{n+1}, \phi^{n+1} - \phi^n\right) &= \varepsilon^2\left(\Delta_h \phi^{n+1}, \Delta_h (\phi^{n+1} - \phi^n)\right) \\ &= \frac{1}{2}\varepsilon^2\left\|\Delta_h \phi^{n+1}\right\|_2^2 - \frac{1}{2}\varepsilon^2\left\|\Delta_h \phi^n\right\|_2^2 + \frac{1}{2}\varepsilon^2\left\|\Delta_h (\phi^{n+1} - \phi^n)\right\|_2^2. \end{aligned} \tag{3.13}$$

A combination of (3.9), (3.11)-(3.13) yields

$$F_h(\phi^{n+1}) - F_h(\phi^n) + \frac{1}{2}\left\|\nabla_h (\phi^{n+1} - \phi^n)\right\|_2^2 + \frac{1}{2}\varepsilon^2\left\|\Delta_h (\phi^{n+1} - \phi^n)\right\|_2^2 \leq -s\left\|\mu^{n+1}\right\|_2^2. \tag{3.14}$$

The desired result (3.6) follows from dropping some positive difference terms from (3.14). □

4 Preconditioned Methods

sec:psd-pcg

In this section we first recall PSD algorithm in [6]. Based on the PSD algorithm in [6], we propose two efficient preconditioned nonlinear conjugate gradient solvers for SS equation in this section.

The fully discrete scheme (3.4) can be recast as a minimization problem: For any $\phi \in \mathcal{C}_{\text{per}}$,

$$E_h[\phi] = \frac{1}{2}\|\phi\|_2^2 + \frac{s}{4}\|\nabla_h \phi\|_4^4 + \frac{s\varepsilon^2}{2}\|\Delta_h \phi\|_2^2. \tag{4.1}$$

One will observe that the fully discrete scheme (3.4) is the discrete variation of the strictly convex energy (4.1) set equal to zero. The nonlinear scheme at a fixed time level may be expressed as

$$\mathcal{N}_h[\phi] = f, \tag{4.2}$$

where

$$\mathcal{N}_h[\phi] = \phi^{n+1} - s\nabla_h \cdot (|\nabla_h \phi^{n+1}|^2 \nabla_h \phi^{n+1}) + s\varepsilon^2 \Delta_h^2 \phi^{n+1}, \tag{4.3}$$

and

$$f = \phi^n - s\Delta_h^2 \phi^n. \tag{4.4}$$

4.1 Preconditioned Steepest Descent Methods [6]

The main idea of the PSD solver is to use a linearized version of the nonlinear operator as a pre-conditioner, or in other words, as a metric for choosing the search direction. A linearized version of the nonlinear operator $\mathcal{N}$ is defined as follows: $\mathcal{L}_h : \mathring{\mathcal{C}}_{\text{per}} \to \mathring{\mathcal{C}}_{\text{per}}$,

$$\mathcal{L}_h[\psi] := \frac{1}{2}\psi - s\Delta_h\psi + s\varepsilon^2\Delta_h^2\psi.$$

Clearly, this is a positive, symmetric operator, and we use this as a pre-conditioner for the method. Specifically, this "metric" is used to find an appropriate search directtion for our steepest descent solver [6]. Given the current iterate $\phi^n \in \mathcal{C}_{\text{per}}$, we define the following *search direction* problem: find $d^k \in \mathring{\mathcal{C}}_{\text{per}}$ such that

$$\mathcal{L}_h[d^k] = f - \mathcal{N}_h[\phi^k] := r^k,$$

where r^k is the nonlinear residual of the k^{th} iterate ϕ^k. This last equation can be solved efficiently using the Fast Fourier Transform (FFT).

We then define the next iterate as

$$\phi^{k+1} = \phi^k + \overline{\alpha}_k d^k, \tag{4.5}$$

where $\overline{\alpha}_k \in \mathbb{R}$ is the unique solution to the steepest descent line minimization problem

$$\overline{\alpha}_k := \underset{\alpha\in\mathbb{R}}{\text{argmax}}\, E_h[\phi^k + \alpha d^k] = \underset{\alpha\in\mathbb{R}}{\text{argzero}}\, \delta E_h[\phi^k + \alpha d^k](d^k). \tag{4.6}$$

eqn-search

The complete algorithm can be found in Algorithm 1.

gorithm:psd

Algorithm 1 Preconditioned Steepest Descent (PSD) Method

1: Convex-splitting scheme: $\mathcal{N}_h(\phi) = f$
2: Linearize $\mathcal{N}_h(d_k)$ for search direction: $\mathcal{L}_h(d^k) = r^k := f - \mathcal{N}_h(\phi^k)$ ▷ solved by FFT
3: Search steps: $\overline{\alpha}_n = secantsearch(d^k)$ ▷ secant search
4: Update: $\phi^{k+1} = \phi^k + \overline{\alpha}_k d^k$ ▷ steepest descent algorithm

4.2 Preconditioned Nonlinear Conjugate Gradient

Based on the PSD algorithm, we define $\overline{g}_k = \mathcal{L}_h^{-1}(r^k)$. Then our PNCG algorithms are given by the following equations:

$$\begin{aligned} \phi^{k+1} &= \phi^k + \overline{\alpha}_k d^k & (4.7)\\ d^{k+1} &= -\overline{g}_{k+1} + \overline{\beta}_{k+1} d^k,\, d^0 = -\overline{g}_0. & (4.8) \end{aligned}$$

And more details can be found in Algorithm 2.

However, there several different ways to choose the scaling parameter $\overline{\beta}_{n+1}$. And two of the best known formulas for $\overline{\beta}_{n+1}$ are named after their deveiops:

Fletcher-Reeves [8]:

$$\overline{\beta}_{k+1}^{FR} = \frac{\overline{g}_{k+1}^T \overline{g}_{k+1}}{\overline{g}_k^T \overline{g}_k} \tag{4.9}$$

gorithm:pncg

Algorithm 2 Linearly Preconditioned Nonlinear Conjugate Gradient (PNCG) Method

Compute residual: $r^0 := f - \mathcal{N}_h(\phi^0)$
Set $\overline{g}_0 = \mathcal{L}_h^{-1}(r^0)$
Set $d^0 \leftarrow -\overline{g}_0$, $k \leftarrow 0$
while $\overline{g}_k \neq 0$ **do**
    Compute $\overline{\alpha}_k$ ▷ secant search
    $\phi^{k+1} \leftarrow \phi^k + \overline{\alpha}_k d^k$ ▷ steepest descent algorithm
    $\overline{g}_{k+1} \leftarrow \mathcal{L}_h^{-1}(r^{k+1}) = \mathcal{L}_h^{-1}(f - \mathcal{N}_h(\phi^{k+1}))$
    Compute $\overline{\beta}_{k+1}$
    $d^{k+1} \leftarrow -\overline{g}_{k+1} + \overline{\beta}_{k+1} d^k$
    $k \leftarrow k + 1$
end while

Polak-Ribière [14]:

$$\overline{\beta}_{k+1}^{PR} = \frac{\overline{g}_{k+1}^T(\overline{g}_{k+1} - \overline{g}_k)}{\overline{g}_k^T \overline{g}_k} \tag{4.10}$$

Based on those two best known formulas, we proposed the following two PNCG solvers:

PHCG1:

$$\overline{\beta}_{k+1} = \max\{0, \overline{\beta}_{k+1}^{PR}\} \tag{4.11}$$

PHCG2 :

$$\overline{\beta}_{k+1} = \max\{0, \min\{\overline{\beta}_{k+1}^{FR}, \overline{\beta}_{k+1}^{PR}\}\} \tag{4.12}$$

Remark 4.1. *The PNCG2 is also called hybrid conjugate gradient algorithm in [24].*

5 Adaptive time-stepping method

ec:adaptive

As we have proven in section 3, the proposed scheme is unconditionally energy stable which allows us to adopt large time steps to the simulations. However, for the sake of accuracy, large time steps are not proper for a rapidly phase transformation. In order to make the proposed scheme much more practical, we apply the adaptive time stepping strategy in [15, 22, 23] based on the change ratio of the free energy. The adaptive time steps is defined as follows:

$$s = \max\left(s_{\min}, \frac{s_{\max}}{\sqrt{1 + \alpha|\delta_t F_h(\phi)|^2}}\right), \tag{5.1}$$

where α is a constant and $s_{\min}$ and $s_{\max}$ are pre-set lower and upper bound of the time steps, respectively. By introducing the pre-set time steps, the $s_{\min}$ can force the adaptive time steps bounded from below to avoid too small time steps and the $s_{\max}$ gives the upper bound of the time steps to guarantee the accuracy. Consequently,

$$s_{\min} \leq s \leq s_{\max}.$$

6 Numerical Experiments

sec:num-pcg

In this section we demonstrate the accuracy, complexity and efficiency of the PNCG solvers. We present the results of the convergence tests and perform some sample computations to demonstrate the complexity and the efficiency of PNCG solvers. Moreover, We also provide some comparison results between the proposed PNCG solvers and the PSD solver in [6]. The stop tolerances for all of the following simulations are 1.0×10^{-10}.

6.1 Convergence tests and the complexity of the PNCG solvers

-convergence

To simultaneously demonstrate the spatial accuracy and the efficiency of the solver, we perform a typical time-space convergence test for the fully discrete scheme (3.4) for the SS model. As in [6, 16, 18], we perform the Cauchy-type convergence test using the following periodic initial data [16]:

$$\begin{aligned} u(x,y,0) &= 0.1\sin^2\left(\frac{2\pi x}{L}\right)\cdot\sin\left(\frac{4\pi(y-1.4)}{L}\right) \\ &\quad -0.1\cos\left(\frac{2\pi(x-2.0)}{L}\right)\cdot\sin\left(\frac{2\pi y}{L}\right), \end{aligned} \tag{6.1}$$

with $\Omega = [0, 3.2]^2$, $\varepsilon = 1 \times 10^{-1}$, $s = 0.1h^2$ and $T = 0.32$. We use a quadratic refinement path, *i.e.*, $s = Ch^2$. At the final time $T = 0.32$, we expect the global error to be $\mathcal{O}(s) + \mathcal{O}(h^2) = \mathcal{O}(h^2)$ under either the ℓ^2 or ℓ^∞ norm, as $h, s \to 0$. The Cauchy difference is defined as $\delta_\phi := \phi_{h_f} - \mathcal{I}_c^f(\phi_{h_c})$, where $\mathcal{I}_c^f$ is a bilinear interpolation operator (We applied Nearest Neighbor Interpolation in Matlab, which is similar to the 2D case in [5, 6] the 3D case in [4]). This requires having a relatively coarse solution, parametrized by h_c, and a relatively fine solution, parametrized by h_f, in particular $h_c = 2h_f$, at the same final time T. The ℓ^2 norms of Cauchy difference and the convergence rates can be found in Table 1 which confirms our expectation for the first order in time and second order in space convergence.

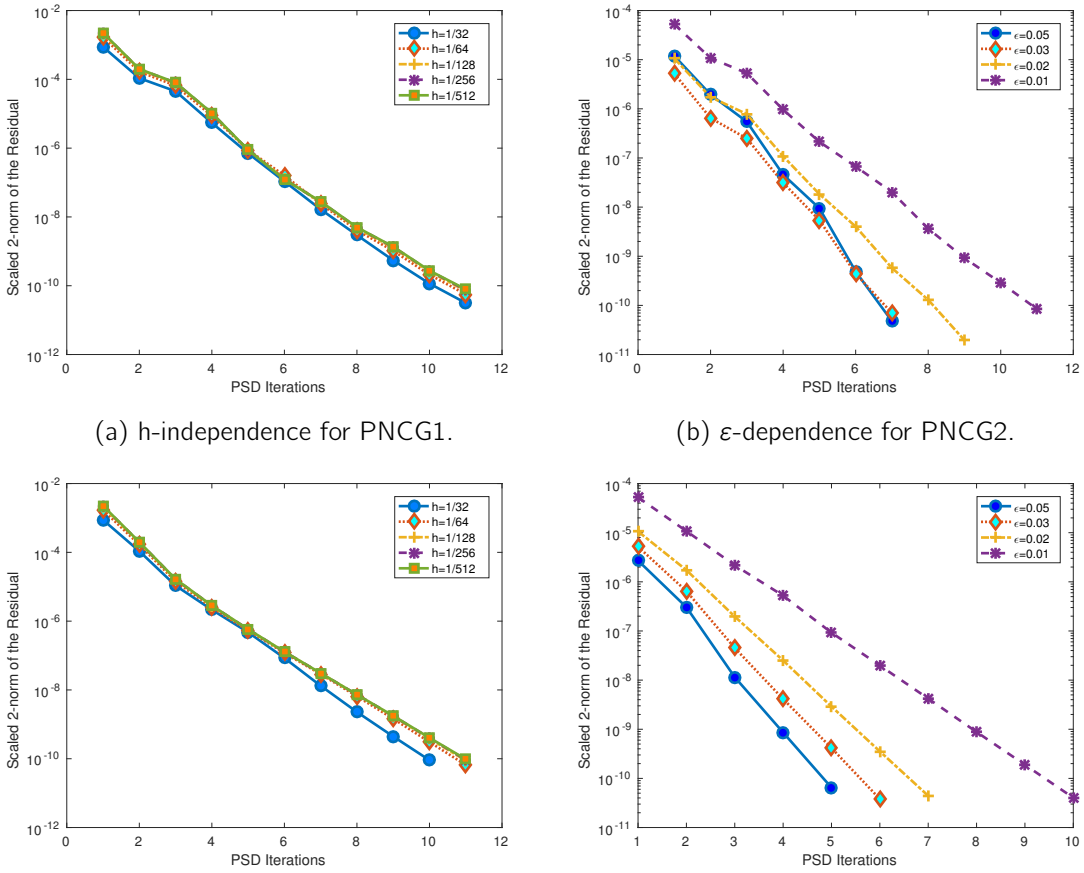
Table 1: Errors, convergence rates, average iteration numbers and average CPU time (in seconds) for each time step. Parameters are given in the text, and the initial data is defined in (6.1). The refinement path is $s = 0.1h^2$.

		PNCG1				PNCG2			
h_c	h_f	$\|\delta_\phi\|_2$	Rate	$\#_{iter}$	$T_{cpu}(h_f)$	$\|\delta_\phi\|_2$	Rate	$\#_{iter}$	$T_{cpu}(h_f)$
$\frac{3.2}{16}$	$\frac{3.2}{32}$	5.9944×10^{-3}	-	4	0.0007	5.9944×10^{-3}	-	4	0.0007
$\frac{3.2}{32}$	$\frac{3.2}{64}$	1.1500×10^{-3}	2.38	2	0.0026	1.1500×10^{-3}	2.38	2	0.0025
$\frac{3.2}{64}$	$\frac{3.2}{128}$	2.4689×10^{-4}	2.22	2	0.0220	2.4688×10^{-4}	2.22	2	0.0129
$\frac{3.2}{128}$	$\frac{3.2}{256}$	5.8656×10^{-5}	2.07	2	0.0674	5.8656×10^{-5}	2.07	2	0.0653
$\frac{3.2}{256}$	$\frac{3.2}{512}$	1.4463×10^{-5}	2.02	2	0.4542	1.4463×10^{-5}	2.02	2	0.4533

tab:cov-pcg

In the second part of this test, we demonstrate the complexity of the PNCG solvers with initial data (6.1). In Figure 1 (a) and (c), we plot the semi-log scale of the relative residuals versus PNCG1 and PNCG2 iteration numbers for various values of h, respectively. The other common parameters for the h-independence are $\Omega = [0, 3.2]^2$, $\varepsilon = 1\times10^{-1}$, $s = 0.001$, $T = 1\times10^{-2}$ with time steps $s = 1\times10^{-3}$ and the initial data (6.1). And the semi-log scale of the relative residuals versus PNCG1 and PNCG2 iteration numbers for various values of ε can be found in Figure 1 (b) and (d), respectively. The other common parameters for the ε-dependence are $\Omega = [0, 3.2]^2$, $h = {}^{3.2}/_{512}$, $s = 0.001$, $T = 1\times10^{-2}$ with time steps $s = 1 \times 10^{-3}$ and the initial data (6.1). Figure 1(a) and (c) indicate that the convergence

rates of PNCG solvers (as gleaned from the error reduction) are nearly uniform and nearly independent of h for a fixed ε. Figure 1(b) and (d) show that the number of PNCG iterations increases with the decreasing of ε. Moreover, Figure 1(b) and (d) indicate that the PNCG2 is more efficient and robust than PNCG1. Figure 1 provides the similar geometric convergence rate of the PNCG solvers predicted by the theory in [6].



(a) h-independence for PNCG1.

(b) ε-dependence for PNCG2.

(c) h-independence for PNCG1.

(d) ε-dependence for PNCG2.

Figure 1: Complexity tests showing the solver performance for changing values of h and ε. Parameters are given in the text, and the initial data is defined in (6.1).

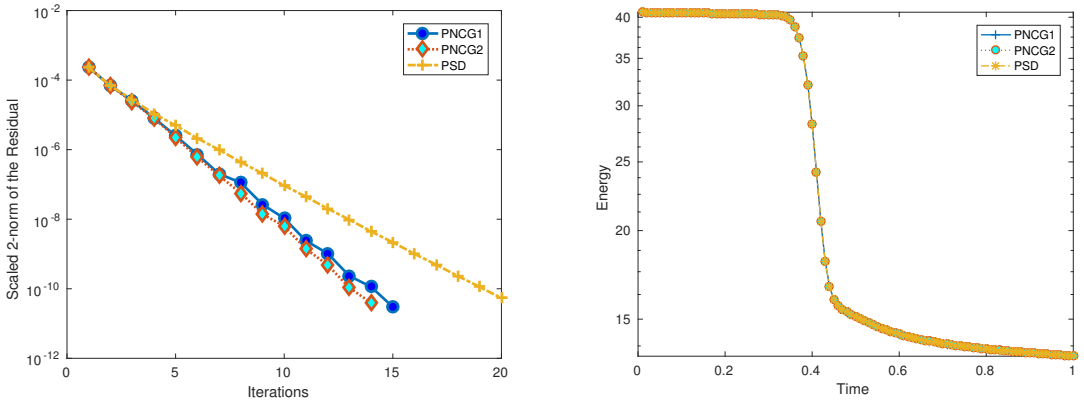
plexity-pcg

In the third part of this test, we perform some comparisons between the proposed PNCG solvers and the PSD solver. The parameters for the comparison simulations are $\Omega = [0, 12.8]^2$, $\varepsilon = 3 \times 10^{-2}$, $h = {}^{12.8}/_{512}$, $s = 0.01$, and $T = 0.32$. The average iteration numbers, total CPU time (in seconds) and speedups for the preconditioned methods can be found in Table 2. The Table 2 indicates that the PNCG1 solver and PNCG2 solver have provided a 1.34x and 1.39x speedup over PSD solver, respectively. The error reductions at $T = 1$ (100th iteration) in Figure 2 (a) indicates the PNCG solvers have less iteration numbers and faster error reduction at each time iteration. And energy evolutions in Figure 2 (a) show that the preconditioned solvers have the same energy evolutions.

Table 2: The average iteration numbers and total CPU time (in seconds) for the preconditioned methods with fixed time steps $s = 0.01$ and initial data (6.1). Parameters are given in the text.

tab:cpu-fixed

Methods	PSD	PNCG1	PNCG2
$\#_{iter}$	17	13	12
T_{cpu}(s)	232.2665	173.2407	167.6591
Speedup	-	1.34	1.39



(a) The error reduction at $T = 1$.

(b) Energy evolutions.

Figure 2: The error reductions at $T = 1$ and energy evolutions for the preconditioned solvers. Parameters are given in the text, and the initial data is defined in (6.1).

fig:pcg-psd

6.2 Long-time coarsening process, energy dissipation and mass conservation

Coarsening processes in thin film systems can take place on very long time scales [12]. In this subsection, we perform long time behavior tests for SS model. Such test, which have been performed in many places, will confirm the expected coarsening rates and serve as benchmarks for our solver. See, for example, [16, 18]. The initial data for the simulations are taken as essentially random:

$$u^0_{i,j} = 0.05 \cdot (2r_{i,j} - 1), \tag{6.2}$$

eqn:init-et

where the $r_{i,j}$ are uniformly distributed random numbers in $[0, 1]$. Since all of the solvers give similar results, we only present the results from PNCG2 in the fellowing content of this subsection. Time snapshots of the evolution for the epitaxial thin film growth model can be found in Figure 3. The coarsening rates are given in Figure 4. The interface width or roughness is defined as

$$W(t_n) = \sqrt{\frac{h^2}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}(\phi^n_{i,j} - \bar{\phi})^2}, \tag{6.3}$$

where m and n are the number of the grid points in x and y direction and $\bar{\phi}$ is the average value of ϕ on the uniform grid. The log-log plots of roughness and energy evolution and the corresponding linear regression are presented in Figure. 4. The linear regressions in Figure. 4 indicate that the surface roughness grows like $t^{1/3}$ and the energy decays like $t^{1/3}$. In particular, the linear fits have the form $a_e t^{b_e}$ with $a_e = 11.2, b_e = -0.3315$ for energy evolution and $a_r t^{b_r}$ with $a_r = 0.0025, b_r = 0.3255$

for roughness evolution. Those decay properties confirm the one-third power law in [13]. Moreover, these simulation results are consistent with earlier work on this topic in [6, 16, 18, 21].

The PNCG2 iterations and mass difference at each time steps for the simulation depicted in Figure 3 are presented in 5. The PNCG2 iterations indicate that the average iteration number of PNCG2 solver is only 3. And the mass difference at each time steps clearly shows the mass Conservative property.

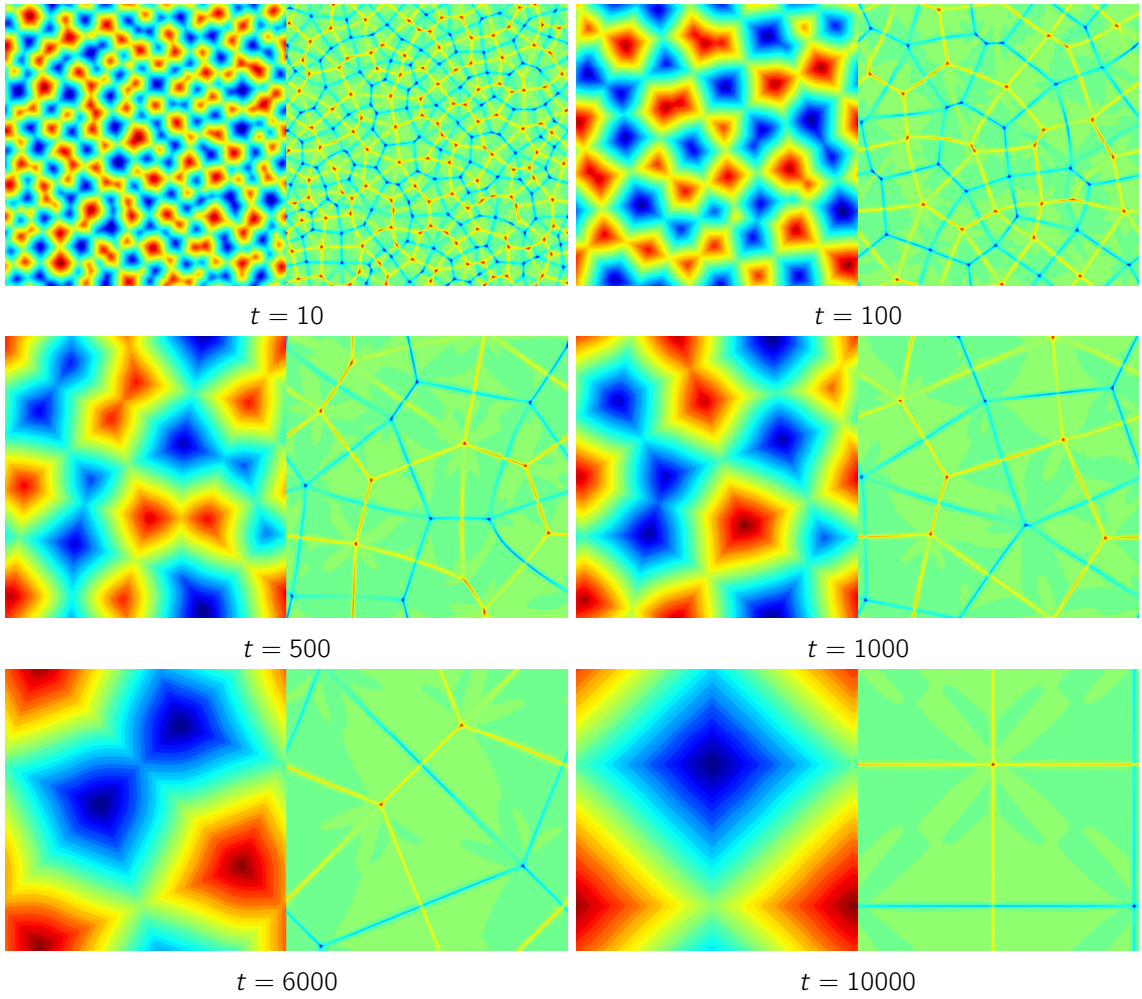


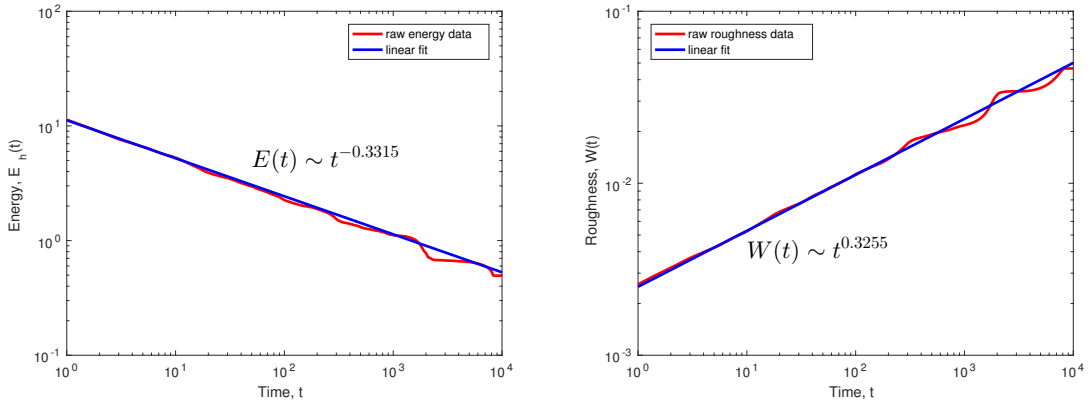
Figure 3: Time snapshots of the evolution with PSD solver for the epitaxial thin film growth model at $t = 10, 100, 500, 1000, 6000$ and 10000. Left: contour plot of u, Right: contour plot of Δu. The parameters are $\varepsilon = 0.03, \Omega = [12.8]^2$ and $s = 0.01$. These simulation results are consistent with earlier work on this topic in [6, 7, 16, 18, 21].

ng-time-pcg

6.3 Adaptive time strategy

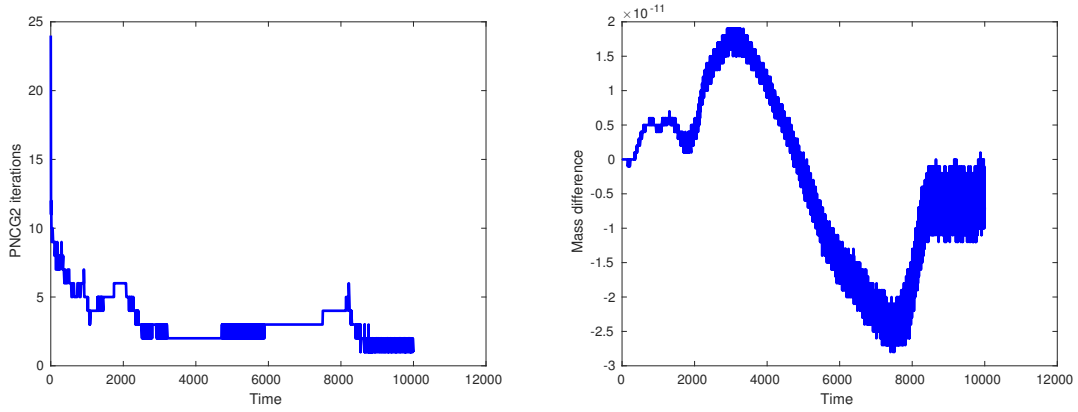
:adaptivedt

In order to make the proposed scheme much more practical, we investigate the adaptive time stepping strategy in this subsection. The Figure 5.1 (a) and (c) indicate that the energy decays and adaptive time steps are the same, which demonstrate the robustness of the proposed preconditioned methods. Moreover, according to the $t^{-1/3}$ reference line in Figure 5.1 (a), we can conclude that the energy E_h decays like $t^{-1/3}$. From the Figure 5.1 (b), we can observe that the masses are conservative in sense



(a) Energy evolution (b) Roughness evolution

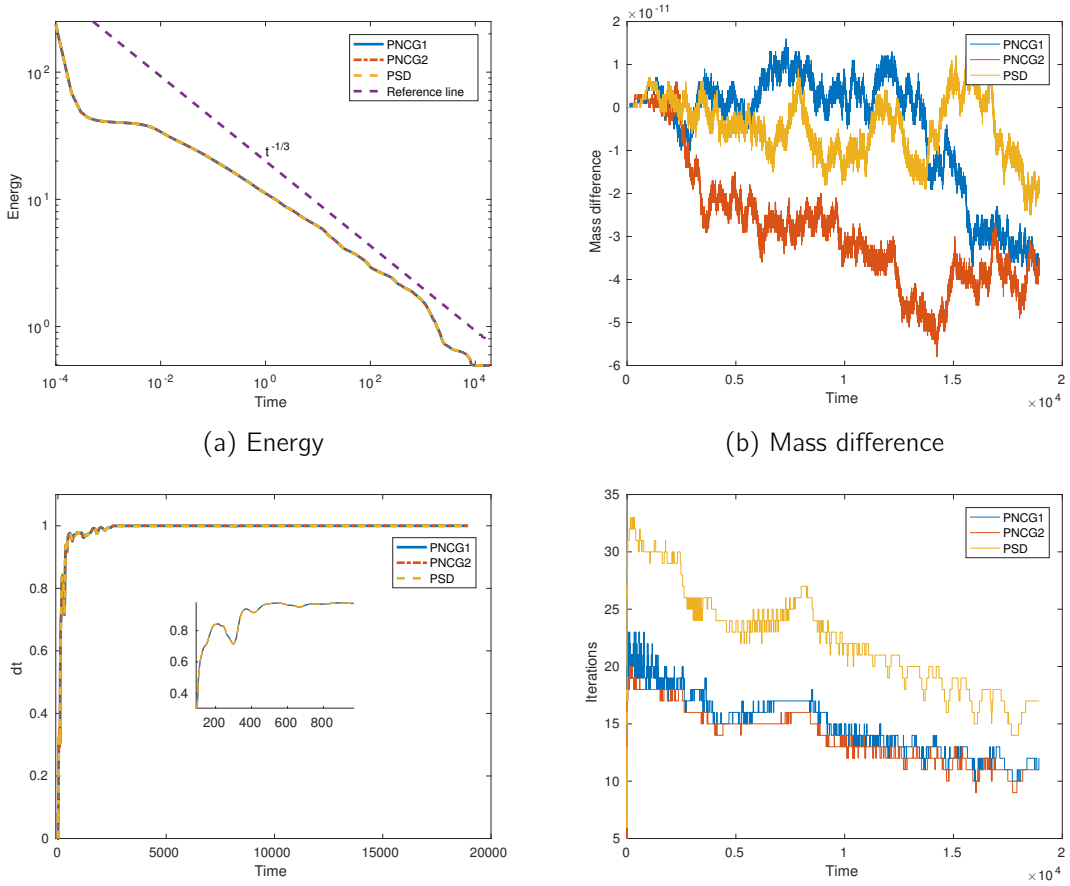
Figure 4: The log-log plots of energy and roughness evolution and the corresponding linear regression for the simulation depicted in Figure 3.



(a) PNCG2 iterations (b) Mass difference

Figure 5: PNCG2 iterations and mass difference at each time steps for the simulation depicted in Figure 3.

of 10^{-11}. The adaptive time steps and the number of iterations at each time steps are presented in Figure 5.1 (c) and (d), respective. From Figure 5.1 (d), we can clearly see that the PNCG solvers have less iteration numbers than the PSD solver. Moreover, the total CPU time in Table 3 shows that the adaptive time stepping approach can greatly save CPU time without losing accuracy.



(a) Energy

(b) Mass difference

(c) Adaptive time steps

(d) Number of iterations

Figure 6: Adaptive time-stepping methods with adaptive time steps (5.1). The rest of the parameters are $\Omega = [0, 12.8]^2$, $\varepsilon = 3.0 \times 10^{-2}$, $h = {}^{12.8}/_{512}$, $smin = 1.0^{-4}$, $smax = 1.0$.

adaptiveTime

Table 3: The total CPU time (in seconds) and speedups for the preconditioned methods with adaptive time steps for the simulation depicted in Figure 6. Parameters are given in the text.

tab:cpu-adaptive

Methods	PSD	PNCG1	PNCG2
T_{cpu}(s)	106968.8356	75029.2439	70657.7581
Speedup	-	1.43	1.51

7 Conclusions

clusion-pcg

In this paper, we have proposed two efficient PNCG solvers for solving the two-dimensional epitaxial thin film with SS equation base on the PSD method in [6]. Since the proposed scheme is unconditionally energy stable, large time steps are allowed in the long time numerical simulations. In order to guarantee the accuracy and make our solver more efficient, we also adopted an adaptive time-stepping strategy which the energy is used to monitor the change of the time steps. Various numerical results are also presented, including the first-order-in-time accuracy test, energy-dissipation, mass-conservation test and comparison tests. Moreover, numerical experiments for the adaptive time-stepping method demonstrated that the adaptive time stepping approach can greatly save CPU time without losing accuracy.

8 Acknowledgments

The first author would like to thank Jian Sun at the University of Tennessee, Knoxville for the valuable discussions. This work is supported in part by NSF DMS-1418692 (S. Wise).

References

mathematical [1] P. Aviles, Y. Giga, et al. A mathematical problem related to the physical theory of liquid crystal configurations. In *Proc. Centre Math. Anal. Austral. Nat. Univ*, volume 12, pages 1–16, 1987.

011nonlinear [2] Y. Dai. *Nonlinear conjugate gradient methods*. Wiley Online Library, 2011.

999nonlinear [3] Y. Dai and Y. Yuan. A nonlinear conjugate gradient method with a strong global convergence property. *SIAM J. Optimiz.*, 10(1):177–182, 1999.

Sconvergence [4] L. Dong, W. Feng, C. Wang, S. M. Wise, and Z. Zhang. Convergence analysis and numerical implementation of a second order numerical scheme for the three-dimensional phase field crystal equation. *arXiv preprint arXiv:1611.06288*, 2016.

feng2016fch [5] W. Feng, Z. Guan, J. Lowengrub, S.M. Wise, and C. Wang. An energy stable finite-difference scheme for Functionalized Cahn-Hilliard Equation and its convergence analysis. *arXiv preprint arXiv:1610.02473*, 2016.

econditioned [6] W. Feng, A.J. Salgado, C. Wang, and S.M. Wise. Preconditioned steepest descent methods for some nonlinear elliptic equations involving p-Laplacian terms. *arXiv preprint arXiv:1607.01475*, 2016.

ng2017second [7] W. Feng, C. Wang, S. M. Wise, and Z. Zhang. A second-order energy stable backward differentiation formula method for the epitaxial thin film equation with slope selectio. *In preparation*, 2017.

1964function [8] R. Fletcher and C.M. Reeves. Function minimization by conjugate gradients. *Comput. J.*, 7(2):149–154, 1964.

er2006survey [9] W. W. Hager and H. Zhang. A survey of nonlinear conjugate gradient methods. *Pac. J. Optim.*, 2(1):35–58, 2006.

995iterative [10] C. T. Kelley. Iterative methodsforlinearand nonlinear equations. *Raleigh N. C.: NorthCarolina State University*, 1995.

999iterative [11] C. T. Kelley. *Iterative methods for optimization*, volume 18. SIAM, 1999.

an2006energy [12] R.V. Kohn. Energy-driven pattern formation. In *International Congress of Mathematicians*, volume 1, pages 359–383, 2006.

kohn03 [13] R.V. Kohn and X. Yan. Upper bound on the coarsening rate for an epitaxial growth model. *Comm. Pure Appl. Math.*, 56:1549–1564, 2003.

ere1969note [14] E. Polak and G. Ribiere. Note sur la convergence de directions conjuges. *ESAIM Math. Model Num.*, 3(R1):35–43, 1969.

2011adaptive [15] Z. Qiao, Z. Zhang, and T. Tang. An adaptive time-stepping strategy for the molecular beam epitaxy models. *SIAM J. Sci. Comput.*, 33(3):1395–1414, 2011.

en2012second [16] J. Shen, C. Wang, X. Wang, and S.M. Wise. Second-order convex splitting schemes for gradient flows with Ehrlich-Schwoebel type energy: application to thin film epitaxy. *SIAM J. Numer. Anal.*, 50(1):105–125, 2012.

5nonlinearly [17] H.D. Sterck and M. Winlaw. A nonlinearly preconditioned conjugate gradient algorithm for rank-R canonical tensor approximation. *Numer. Linear Algebra Appl.*, 22(3):410–432, 2015.

onditionally [18] C. Wang, X. Wang, and S.M. Wise. Unconditionally stable schemes for equations of thin film epitaxy. *Discrete Contin. Dyn. Syst.*, 28(1):405–423, 2010.

ag2011energy [19] C. Wang and S.M. Wise. An energy stable and convergent finite-difference scheme for the modified phase field crystal equation. *SIAM J. Numer. Anal.*, 49(3):945–969, 2011.

se2009energy [20] S.M. Wise, C. Wang, and J.S. Lowengrub. An energy-stable and convergent finite-difference scheme for the phase field crystal equation. *SIAM J. Numer. Anal.*, 47(3):2269–2288, 2009.

006stability [21] C. Xu and T. Tang. Stability analysis of large time-stepping methods for epitaxial growth models. *SIAM J. Numer. Anal.*, 44(4):1759–1779, 2006.

2013adaptive [22] Z. Zhang, Y. Ma, and Z. Qiao. An adaptive time-stepping strategy for solving the phase field crystal model. *J. Comput. Phys.*, 249:204–215, 2013.

2012adaptive [23] Z. Zhang and Z. Qiao. An adaptive time-stepping strategy for the Cahn-Hilliard equation. *Commun. Comput. Phys.*, 11(04):1261–1278, 2012.

econditioned [24] G. Zhou, Y. Huang, and C. Feng. Preconditioned hybrid conjugate gradient algorithm for p-Laplacian. *Int. J. Numer. Anal. Model*, 2:123–130, 2005.