Commun. Comput. Phys.
doi: 10.4208/cicp.

Vol. **x**, No. x, pp. 1-32
xxx 2022

Benchmark Computations of the Phase Field Crystal and Functionalized Cahn-Hilliard Equations via Fully Implicit, Nesterov Accelerated Schemes

Jea-Hyun Park[1,*], Abner J. Salgado[2] and Steven M. Wise[2]

[1] *Department of Mathematics, University of California Santa Barbara, CA 93106, USA.*
[2] *Department of Mathematics, University of Tennessee Knoxville, TN 37996, USA.*

Received ; Accepted (in revised version)

Abstract. We introduce a fast solver for the phase field crystal (PFC) and functionalized Cahn-Hilliard (FCH) equations with periodic boundary conditions on a rectangular domain that features the preconditioned Nesterov's accelerated gradient descent (PAGD) method. We discretize these problems with a Fourier collocation method in space, and employ various second-order schemes in time. We observe a significant speedup with this solver when compared to the preconditioned gradient descent (PGD) method. With the PAGD solver, fully implicit, second-order-in-time schemes are not only feasible to solve the PFC and FCH equations, but also do so more efficiently than some semi-implicit schemes in some cases where accuracy issues are taken into account. Benchmark computations of five different schemes for the PFC and FCH equations are conducted and the results indicate that, for the FCH experiments, the fully implicit schemes (midpoint rule and BDF2 equipped with the PAGD as a nonlinear time marching solver) perform better than their IMEX versions in terms of computational cost needed to achieve a certain precision. For the PFC, the results are not as conclusive as in the FCH experiments, which, we believe, is due to the fact that the nonlinearity in the PFC is milder nature compared to the FCH equation. We also discuss some practical matters in applying the PAGD. We introduce an *averaged Newton preconditioner* and a *sweeping-friction* strategy as heuristic ways to choose good preconditioner parameters. The sweeping-friction strategy exhibits almost as good a performance as the case of the best manually tuned parameters.

AMS subject classifications: 74A50, 65M22, 65F08, 65B99

Key words: Phase field crystal, functionalized Cahn-Hilliard, preconditioning, Nesterov acceleration, nonlinear solver.

*Corresponding author. *Email addresses:* `jhpark1@ucsb.edu` (J.-H. Park), `asalgad1@utk.edu` (A.J. Salgado), `swise1@utk.edu` (S.M. Wise)

1 Introduction

We are interested in fast and accurate numerical solvers for initial value problems (IVPs) for nonlinear parabolic partial differential equations of the form

$$\partial_t u = M\Delta \frac{\delta \mathcal{E}}{\delta u}(u), \quad t>0, \quad u|_{t=0} = u_0, \tag{1.1}$$

supplemented with periodic boundary conditions. Here, $\frac{\delta \mathcal{E}}{\delta u}$ denotes the variational derivative of the energy

$$\mathcal{E}(u) = \int_\Omega f(u, \nabla u, \Delta u) \mathrm{d}x.$$

The spatial domain is Ω, which is assumed to be rectangular throughout this paper, and $M : \mathbb{R} \to \mathbb{R}$ is the so-called *mobility constant*. While the mobility may depend on the unknown u in general, we confine ourselves to the case of constant mobility $M \equiv 1$ in this work. Two real world applications, the *phase field crystal* (PFC) and *functionalized Cahn-Hilliard* (FCH) equations (see Section 2 for more details) take this form and are of our main interest.

Our focus is on the numerical solvers. Nevertheless, for completeness, let us briefly mention existing works about the phenomena that the PFC and FCH equations model and their PDE analyses. These two equations are important models in materials science. The PFC equation describes crystal formation in a liquid bath, crack propagations in a crystal layer, and elastic and plastic deformations of a crystal lattice, to name a few. The FCH equations, on the other hand, describes network formation in a binary mixture and is a useful tool for modeling bilayer membrane formation and polymer electrolyte membrane evolution. The reader interested in applications is referred to [1,15–17] for the PFC, and [22, 23, 30] for the FCH model, respectively. There is some limited amount of work about these equations at the PDE level. For the PFC equation see [11, 34]; whereas for the FCH see [6, 12].

Both the PFC and FCH are nonlinear, sixth-order 'parabolic' equations. As such, they share common numerical difficulties, such as accuracy and stability, and there have been efforts to overcome them; see, for example, [7,24,26,35,39] for the PFC, and [20,27,37,38] for the FCH, respectively. If one wishes to have a long time evolution of the equations, explicit discretization schemes in time must typically be excluded due to their stringent restriction on the time step size, ($\mathrm{d}t \approx \mathrm{d}x^6$), for stability. On the other hand implicit schemes, which are more robust in terms of stability and accuracy, as a rule lead to a large, highly nonlinear system that must be solved at every time step. A substantial amount of work has been dedicated to developing schemes that mitigate the numerical difficulties or instabilities of either of these extreme approaches, fully explicit schemes, on one hand, and fully implicit schemes, on the other. Examples of this are the convex splitting technique [7, 24, 26, 27, 35, 37, 38], and the SAV technique [5, 8, 28], to name a few. Both of these approaches, however, are known to create larger local truncation errors than implicit schemes [36, 38]. If a reliable, robust, and efficient iterative solver is

available to handle the nonlinear equations resulting from fully implicit schemes, a good balance between accuracy and the efficiency may be within reach.

In previous work [29], we showed that the preconditioned Nesterov's accelerated gradient descent method (PAGD; see Algorithm 3 for definition) can be applied to approximate the minimizer of a strongly convex objective that is *locally Lipschitz* smooth as opposed to *globally Lipschitz* smooth ones as most of the literature assumes. This significantly extends the applicability of the PAGD as a numerical PDE solver. In [29] it is also reported that the PAGD's performance can be significantly better than that of the preconditioned gradient descent method (PGD; see Algorithm 2 for definition), especially on harder problems.

In light of our previous discussion, the construction and analysis of efficient, time-adaptive, implicit schemes — with the PAGD solver as the central engine — for high-order nonlinear parabolic equations, such as the PFC and FCH equations, is an underdeveloped subject, and our main motivation in writing this contribution. Our first goal is to establish that the PAGD makes an efficient solver for real world problems (see Section 4.2). But this begs the question: *Does the PAGD make implicit schemes more attractive than, say, semi-implicit ones? What should one compare to answer this question?* These questions are addressed in Section 4.3. To compare schemes we measure the computational cost needed to achieve a certain precision. Under this metric, our experiments indicate that implicit schemes are indeed a better choice when nonlinearity of the problem is "strong". If one compares 'dollars per digit' cost — that is, the number of flops to achieve a desired level of accuracy in a computed solution — as we do in Section 4.3, our experiments indicate that the implicit schemes are often a better choice over linear semi-implicit methods.

In the course of achieving our first goal, we also discuss two practical issues. One is about how to choose parameters involved in the PAGD scheme. To implement the PGD method, only the step size s needs to be set. In contrast, PAGD contains an additional tunable parameter, which herein we call *friction* and label η (see [29] for motivation behind this naming convention). We suggest what we call the *sweeping-η* or *sweeping-friction* strategy rather than finding a single optimal constant by trial and error. It turns out that the sweeping-η strategy is almost as efficient as the best-tuned constant friction setup and it is more robust than the latter in the sense that its performance depends less on different ranges for η to sweep than that of the constant-friction setup does on different fixed values of η. A detailed discussion and its intuition is explained in Section 3.4. The second practical issue is how to choose a good preconditioner. Again, rather than finding necessary constants by trial and error, we suggest what we call the *averaged Newton* preconditioner, which computes the parameters involved in the preconditioner in such a way that it mimics the second variation of the objective functional among a certain type of linear operators. See Section 3.4 for a detailed discussion.

The rest of this paper is organized as follows: Section 2 summarizes the mathematical formulations of the two models of interest, namely the FCH and PFC equations. In Section 3, we detail how to discretize (in time) the PDEs in four different ways, whose result-

ing solvers are used in the numerical experiments. We also talk about how to construct the numerical solvers in the same section and explain the sweeping-friction strategy and the averaged Newton preconditioner in detail. Section 4 summarizes the benchmark problems, the results of the numerical experiments, and our interpretation of the results. Finally, we make concluding remarks in Section 5.

2 The phase field crystal and functionalized Cahn-Hilliard equations

We begin our discussion by providing some details regarding the models that we shall be interested in.

2.1 Phase field crystal equation

There are several versions of the phase field crystal (PFC) equations [1,14–16,32]. We will use only the prototypical version, as presented in [15]. The other variants of the model bring similar numerical challenges. The PFC model, at its heart, describes solidification of a unary crystal from its liquid phase. The model captures atomic-scale features on a diffusive time scale. Suppose that $u:\Omega\to\mathbb{R}$ defines an atom density. The free energy of the system (at constant temperature) is

$$\mathcal{E}_{\rm PFC}(u):=\int_\Omega\left[\frac{1}{4}u^4+\frac{1-\varepsilon}{2}u^2-|\nabla u|^2+\frac{1}{2}(\Delta u)^2\right]\mathrm{d}x, \tag{2.1}$$

where ε is a parameter that mimics the temperature variation. We assume that u satisfies periodic boundary conditions on Ω, for simplicity, and that the dynamics for u are mass conserving and free energy dissipative. This leads to the following system of equations:

$$\begin{aligned}&\partial_t u=M\Delta\mu,\\&\mu=\frac{\delta\mathcal{E}_{\rm PFC}}{\delta u}=u^3+(1-\varepsilon)u+2\Delta u+\Delta^2 u,\end{aligned} \tag{2.2}$$

which is an H^{-1}-gradient flow with a constant mobility $M>0$. Mass is conserved, i.e., $\mathrm{d}_t\int_\Omega u(x,t)\,\mathrm{d}x=0$, and energy is dissipated at the rate $\mathrm{d}_t\mathcal{E}_{\rm PFC}(u)=-\int_\Omega|\nabla\mu|^2\,\mathrm{d}x$.

2.2 The functionalized Cahn-Hilliard equation

The functionalized Cahn-Hilliard equation is a phase field model that describes network formation of amphiphilic di-block co-polymer mixtures [22,23]. As with the PFC, there are several versions of the FCH equations, as the model needs to be fine-tuned to the physical system of interest [10,22,23,38]. We will use the same model as that used in the computational benchmark paper [38]. (See also [10].) Let $u:\Omega\to\mathbb{R}$ denote the volume

fraction of component A in a binary mixture of molecules. The free energy of the mixture (at constant temperature) is

$$\mathcal{E}_{\rm FCH}(u) := \int_\Omega \left[\frac{1}{2}\left(\varepsilon^2\Delta u - F'(u)\right)^2 - \left(\frac{\varepsilon^2}{2}\eta_1|\nabla u|^2 + \eta_2 F(u)\right)\right] \mathrm{d}x, \tag{2.3}$$

where F is a double well potential, ε is an interface thickness parameter; and $\eta_1, \eta_2 > 0$ are material parameters. We assume, for simplicity that u is periodic on the square domain Ω. The corresponding FCH equation, with constant mobility $M > 0$, reads

$$\partial_t u = M\Delta \frac{\delta \mathcal{E}_{\rm FCH}}{\delta u} = M\Delta\left[\left(\varepsilon^2\Delta u - F''(u)\right)\left(\varepsilon^2\Delta u - F'(u)\right) - \left(-\varepsilon^2\eta_1\Delta u + \eta_2 F'(u)\right)\right]. \tag{2.4}$$

Written as a system of three second-order equations, we have, equivalently,

$$\begin{aligned} &\partial_t u = M\Delta\mu, \\ &\mu = \frac{\delta \mathcal{E}_{\rm FCH}}{\delta u} = \left(\varepsilon^2\Delta u - F''(u) + \eta_1\right)\omega + (\eta_1 - \eta_2)F'(u), \\ &\omega = \varepsilon^2\Delta u - F''(u). \end{aligned} \tag{2.5}$$

We assume that $F : \mathbb{R} \to \mathbb{R}$ is a polynomial double-well potential of the form

$$F(\zeta) = \frac{1}{2}(\zeta+1)^2\left(\frac{1}{2}(\zeta-1)^2 + \frac{2}{3}\tau(\zeta-2)\right),$$

whose symmetry can be tuned by adjusting $\tau \in \mathbb{R}$.

Similar to PFC, the FCH system can be seen as a H^{-1}-gradient flow of the energy $\mathcal{E}_{\rm FCH}$. As before, mass is conserved, $\mathrm{d}_t \int_\Omega u(x,t)\,\mathrm{d}x = 0$, and energy is dissipated at the rate $\mathrm{d}_t \mathcal{E}_{\rm FCH}(u) = -\int_\Omega |\nabla\mu|^2\,\mathrm{d}x$.

3 Discretization and numerical solvers

In this section, we describe our numerical approach. First, in Section 3.1, we present our adaptive time discretizations: the fully and semi-implicit BDF2 and midpoint rule (MP). This reduces our problem to a sequence of time independent, sixth-order, nonlinear elliptic equations, which are then discretized by a Fourier collocation method as detailed in Section 3.2. Finally, the ensuing nonlinear systems of equations are solved using the linear and nonlinear solvers described in Sections 3.3 and 3.4.

3.1 Time discretization

3.1.1 Fully and semi- implicit BDF2 and midpoint rule

We choose four different schemes for time discretization: the fully implicit second-order backward differentiation formula (BDF2), fully implicit midpoint rule (MP), a (linear) semi-implicit second-order backward differentiation formula (LBDF2), and a (linear)

semi-implicit midpoint rule (LMP). There are several reasons for these choices. First, we do not consider explicit schemes since, to be stable, they require extremely stringent time step size restrictions of the form $\mathrm{d}t \approx \mathrm{d}x^6$; see [20]. Second, we want to compare the performance of fully implicit schemes and their semi-implicit versions. Both of these classes of schemes are known to be unconditionally stable and accurate. However, at first glance, one might expect that fully implicit schemes will not be computationally efficient, as they require solving a nonlinear system every time step. On the other hand, while *semi-implicit* schemes, like the first-order convex splitting scheme [2, 9, 18, 19], are known to be often fast and stable, these properties always come at the expense of accuracy. Thus, a fair comparison between these two classes of schemes must be made by considering both speed and accuracy.

Let us now describe our schemes in more detail. We introduce a nonuniform time grid $\{t_n\}_{n\geq 0}$ with (variable) time step defined by $\mathrm{dt}_{n+1} = t_{n+1} - t_n$. The sequence of functions $\{u^n : \Omega \to \mathbb{R}\}_{n\geq 0}$ is meant to be an approximation of u, the solution of (1.1), at the time grid points, i.e., $u^n(\cdot) \approx u(t_n, \cdot)$ for all $n \geq 0$.

The fully implicit schemes are defined as follows: given the initial value $u^0 = u_0$, find u^{n+1}, for $n \geq 0$, as the solution of

$$a_n u^{n+1} + b_n u^n + c_n u^{n-1} = M\Delta \frac{\delta \mathcal{E}}{\delta u}(\breve{u}^{n+1}), \tag{3.1}$$

where the coefficients $\{a_n\}_{n\geq 0}, \{b_n\}_{n\geq 0}, \{c_n\}_{n\geq 0}$, and the choice of the function $\breve{u}^{n+1} : \Omega \to \mathbb{R}$ define the various fully implicit schemes. In particular, for the BDF2 scheme, we have

$$a_0 = \frac{1}{\mathrm{dt}_1}, \quad b_0 = -\frac{1}{\mathrm{dt}_1}, \quad c_0 = 0, \quad \breve{u}^1 = u^1,$$

and, for $n \geq 1$,

$$\begin{aligned} a_n &= \frac{1}{\mathrm{dt}_{n+1}} + \frac{1}{\mathrm{dt}_{n+1} + \mathrm{dt}_n}, \quad b_n = -\frac{1}{\mathrm{dt}_{n+1}} - \frac{1}{\mathrm{dt}_n}, \\ c_n &= \frac{1}{\mathrm{dt}_n} - \frac{1}{\mathrm{dt}_{n+1} + \mathrm{dt}_n}, \qquad \breve{u}^{n+1} = u^{n+1}. \end{aligned} \tag{3.2}$$

The MP scheme is defined by

$$a_n = \frac{1}{\mathrm{dt}_{n+1}}, \quad b_n = -\frac{1}{\mathrm{dt}_{n+1}}, \quad c_n = 0, \quad \breve{u}^{n+1} = \frac{u^{n+1} + u^n}{2}, \tag{3.3}$$

for all $n \geq 0$.

The semi-implicit schemes we shall use are of linear IMEX type (see, e.g., [10] for details). They only require, at each time step, the solution of a linear system of equations. To achieve this, these methods decompose the chemical potential, $\frac{\delta \mathcal{E}}{\delta u}$, into two parts

$$\frac{\delta \mathcal{E}}{\delta u} = \mathfrak{L}(u) + \mathfrak{N}(u),$$

where the *linear* part $\mathfrak{L}$ is a linear, positive semi-definite operator. The remaining terms

constitute the *nonlinear* part. We have chosen the following decompositions. For the PFC model (2.2), we set

$$\begin{aligned}\mathfrak{L}(u) &= (1+\Delta)^2 u,\\ \mathfrak{N}(u) &= u^3 - \varepsilon u.\end{aligned}$$

For the FCH model (2.4), we add and subtract a second and a zeroth order term and obtain

$$\begin{aligned}\mathfrak{L}(u) &= \varepsilon^4\Delta^2 u + \kappa_2(-\Delta)u + \kappa_0 u\\ \mathfrak{N}(u) &= -\kappa_0 u - \varepsilon^2\Delta F'(u) - (\varepsilon^2(F''(u)-\eta_1)-\kappa_2)\Delta u + (F''(u)-\eta_2)F'(u),\end{aligned}$$

where the parameters are set to $\kappa_0 = (1-2\tau^2+\eta_2)$ and $\kappa_2 = 1$, respectively. The parameter κ_0 is equal to the linear coefficient of the zeroth-order term $(F''(u)-\eta_2)F'(u)$, which is a quintic polynomial in u. The value of the parameter κ_2 was found by trial and error. We note that these parameters are not optimized and that they differ from those in [10] because it turned out that, in our setting, the values in [10] made some of our schemes extremely slow. However, we tried several reasonable options and have chosen a combination that yields an expected evolution of the FCH model. The semi-implicit schemes are obtained by treating the linear part, $\mathfrak{L}(u)$, implicitly and the nonlinear part, $\mathfrak{N}(u)$, explicitly, using a second-order extrapolation of the previous approximations. That is,

$$a_n u^{n+1} + b_n u^n + c_n u^{n-1} = M\Delta\big(\mathfrak{L}(\bar{u}^{n+1}) + \mathfrak{N}(\breve{u}^{n+1})\big), \tag{3.4}$$

where a_n, b_n, and c_n are the same as in (3.2) and (3.3). The extrapolations, for LBDF2 are given by

$$\begin{aligned}\bar{u}^{n+1} &= u^{n+1},\\ \breve{u}^{n+1} &= u^n + \rho_{n+1}(u^n - u^{n-1}),\end{aligned}$$

whereas for LMP they are

$$\begin{aligned}\bar{u}^{n+1} &= \frac{1}{2}\big(u^n + u^{n+1}\big),\\ \breve{u}^{n+1} &= \frac{1}{2}\big((2+\rho_{n+1})u^n - \rho_{n+1}u^{n-1}\big)\end{aligned}$$

with $\rho_{n+1} = \frac{\mathrm{dt}_{n+1}}{\mathrm{dt}_n}$. When $n=0$, an artificial time iterate $u^{-1} := u^0$ is used, which reduces the explicit treatment using the extrapolation to a pure explicit one, u^0, without extrapolating. Observe that (3.4) is linear in u^{n+1}.

3.1.2 Adaptive time stepping

To be able to accurately carry out long time simulations, we employ variable time step sizes, which are chosen adaptively [25, Chapter III.5]. At every time step, after finding

our numerical solution, we compute an error indicator and, if it is not smaller than our prescribed tolerance the current approximation is discarded, the step size reduced, and a new numerical solution is computed.

For the BDF2, LMP, or LBDF2 schemes, we follow the adaptive strategy detailed in [10, Section 3.2], which we refer to as *AM3 stepping*. One exception is that the midAB2 stepping (see below) is used for PFC2 experiment when it is solved by the LMP solver. This is because midAB2 stepping yields a way better result than AM3 for this computation. We now describe the AM3 stepping in Algorithm 1. See also Algorithm 4. We first introduce a predetermined stepping tolerance $\mathrm{TOL}>0$, as well as maximum and minimum time step sizes, denoted by $\mathrm{dt}_{\min}$ and $\mathrm{dt}_{\max}$, respectively.

We comment that, in (3.7), the number 0.9 is a so-called *safety factor*, and that the power of a third in is related to the fact that the local truncation error of our schemes of interest (MP, BDF2, LMP, and LBDF2) is of order two.

Following a suggestion found in [3], we use a different error estimator in the case of the MP scheme (and the LMP when solving PFC2 as mentioned above as an exception), which we call *midAB2 stepping*. Using the computed values of the solution at t_{n-2}, t_{n-1}, and t_n one can compute approximations at the midpoints $t_{n-1/2}$ and $t_{n-3/2}$, these are then used to construct a second-order polynomial that is then evaluated at $\tilde{t}_{n+1}=t_n+\tilde{\mathrm{dt}}_{n+1}$ to obtain

$$\begin{aligned}\hat{u}^{n+1}_{AB2}=u^n&\frac{\left(\tilde{\mathrm{dt}}_{n+1}+\mathrm{dt}_n\right)\left(\tilde{\mathrm{dt}}_{n+1}+\mathrm{dt}_n+\mathrm{dt}_{n-1}\right)}{\mathrm{dt}_n\left(\mathrm{dt}_n+\mathrm{dt}_{n-1}\right)}\\&-u^{n-1}\frac{\tilde{\mathrm{dt}}_{n+1}\left(\tilde{\mathrm{dt}}_{n+1}+\mathrm{dt}_n+\mathrm{dt}_{n-1}\right)}{\mathrm{dt}_n\mathrm{dt}_{n-1}}\\&+u^{n-2}\frac{\tilde{\mathrm{dt}}_{n+1}\left(\tilde{\mathrm{dt}}_{n+1}+\mathrm{dt}_n\right)}{\mathrm{dt}_{n-1}\left(\mathrm{dt}_n+\mathrm{dt}_{n-1}\right)}.\end{aligned}\tag{3.9}$$

Then, the local truncation error can be computed by

$$T_{n+1}=\left(\tilde{u}^{n+1}_{MP}-\hat{u}^{n+1}_{AB2}\right)\frac{1}{1-1/\left(24R_n\right)},\tag{3.10}$$

where $\tilde{u}^{n+1}_{MP}$ is a tentative solution at $\tilde{t}_{n+1}$ using the MP and

$$R_n=\frac{1}{24}+\frac{1}{8}\left(1+\frac{\mathrm{dt}_n}{\tilde{\mathrm{dt}}_{n+1}}\right)\left(1+2\frac{\mathrm{dt}_n}{\tilde{\mathrm{dt}}_{n+1}}+\frac{\mathrm{dt}_{n-1}}{\tilde{\mathrm{dt}}_{n+1}}\right).\tag{3.11}$$

To match the scaling of the error with the AM3 stepping case, we use a L^2-normalized error estimator

$$\mathrm{ERR}=\frac{\left\|\tilde{u}^{n+1}_{MP}-\hat{u}^{n+1}_{AB2}\right\|_{L^2}}{\left\|\hat{u}^{n+1}_{AB2}\right\|_{L^2}}\frac{1}{1-1/\left(24R_n\right)},\tag{3.12}$$

when determining the tentative step size. In the numerical experiments, the midAB2 stepping applies from the third time step $\mathrm{dt}_3=t_3-t_2$ because it requires three previous approximations. For $n=1,2$, we use AM3 stepping. We refer the reader to [3] for further details on the midAB2 stepping strategy.

Algorithm 1: AM3 stepping.

Set $n=0$, $\tilde{\mathrm{dt}}_1=\mathrm{dt}_{\min}$, $t_0=0$.
while $t_n<T$ **do**
  Compute $\tilde{u}^{n+1}$. This is a tentative solution at $\tilde{t}_{n+1}=t_n+\tilde{\mathrm{dt}}_{n+1}$, and is obtained using one of the main schemes (BDF2, LMP, or LBDF2).
  Compute $\hat{u}^{n+1}$. This is a solution of a higher order accuracy obtained using an explicit variant of the AM3 scheme

$$\hat{u}^{n+1}=u^n+\frac{\tilde{\mathrm{dt}}_{n+1}}{6}\left[\frac{3+2\rho_{n+1}}{1+\rho_{n+1}}R\left(\tilde{u}^{n+1}\right)+(3+\rho_{n+1})R\left(u^n\right)-\frac{\rho_{n+1}^2}{1+\rho_{n+1}}R\left(u^{n-1}\right)\right] \tag{3.5}$$

  with

$$\rho_{n+1}=\frac{\tilde{\mathrm{dt}}_{n+1}}{\mathrm{dt}_n},\quad R(v)=M\Delta\frac{\delta\mathcal{E}}{\delta u}(v).$$

  Estimate the error with

$$\mathrm{ERR}=\frac{\|\tilde{u}^{n+1}-\hat{u}^{n+1}\|_{L^2}}{\|\hat{u}^{n+1}\|_{L^2}}. \tag{3.6}$$

  if $\mathrm{ERR}\le\mathrm{TOL}$ **then**
    $\mathrm{dt}_{n+1}=\tilde{\mathrm{dt}}_{n+1}$,
    $t_{n+1}=t_n+\mathrm{dt}_{n+1}$,
    $u^{n+1}=\tilde{u}^{n+1}$.
    Increment n.
  else
    Compute a *tentative* time step by

$$\bar{\mathrm{dt}}_{n+1}=0.9\left(\frac{\mathrm{TOL}}{\mathrm{ERR}}\right)^{\frac{1}{3}}\mathrm{dt}_n, \tag{3.7}$$

$$\tilde{\mathrm{dt}}_{n+1}=\max\left(\mathrm{dt}_{\min},\min(\bar{\mathrm{dt}}_{n+1},\mathrm{dt}_{\max})\right). \tag{3.8}$$

    Goto 3.
  end
end

3.2 Spatial discretization via the Fourier collocation method

To take advantage of the fact that our PDEs are supplemented with periodic boundary conditions on a square, we use the Fourier collocation method for spatial differentiation and integration.

We introduce $K\in\mathbb{N}$, so that the grid resolution is $N=2K+1$, and the grid spacing is $h=1/N$. We define

$$\begin{aligned}\mathbb{N}_{0,N}^2&=\left\{\mathbf{m}=(m_1,m_2)\in\mathbb{Z}^2\;\middle|\;0\le m_1,m_2\le N\right\},\\ \mathbb{N}_N^2&=\left\{\mathbf{m}=(m_1,m_2)\in\mathbb{Z}^2\;\middle|\;1\le m_1,m_2\le N\right\},\\ \mathbb{Z}_K^2&=\left\{\mathbf{r}=(r_1,r_2)\in\mathbb{Z}^2\;\middle|\;-K\le r_1,r_2\le K\right\},\end{aligned}$$

and introduce the uniform grid domain

$$\Omega_N=[0,L]\cap h\mathbb{N}_{0,N}^2. \tag{3.13}$$

We also define the *trial space* of periodic grid functions

$$\begin{aligned}\mathbb{H}_N=\Big\{v_N:\Omega_N\to\mathbb{C}\;\mid\;&v_N(0,hm)=v_N(L,hm),\\ &v_N(h\ell,0)=v_N(h\ell,L),(m,\ell)\in\mathbb{N}_{0,N}^2\Big\}.\end{aligned} \tag{3.14}$$

In the numerical experiment for the PFC, the domain is translated so that $\Omega=[-L/2,L/2]$ and the grid domain is also shifted accordingly. This is purely a cosmetic matter since we are dealing with the periodic boundary conditions.

We omit the details of the case where $N=2K$ for brevity but, up to a slight difference in indexing, a similar construction can be carried out.

Finally, we replace the differential operators in our problems with so-called *Fourier interpolation differentiation*; see [4, pp. 123–124], which we now describe in some detail for the FCH case. For the PFC, a shift of $L/2$ in each coordinate direction is necessary. First, we endow $\mathbb{H}_N$ with the discrete L^2-inner product

$$(u_N,v_N)_N=h^2\sum_{\mathbf{s}\in\mathbb{N}_N^2}u_N(\mathbf{x_s})\overline{v_N(\mathbf{x_s})},$$

where $\overline{v_N(\mathbf{x_s})}$ denotes the complex conjugate.

The Fourier interpolation differentiation can be defined and computed via its diagonalization using the *discrete Fourier transform* (DFT) and the *inverse discrete Fourier transform* (IDFT). The DFT $\hat{w}_K$ of $w_N\in\mathbb{H}_N$ is defined by

$$\hat{w}_K(\mathbf{r})=\left(w_N,e^{\frac{2\pi}{L}\mathrm{i}\mathbf{r}\cdot(\cdot)}\right)_N=h^2\sum_{\mathbf{s}\in\mathbb{N}_N^2}w_N(\mathbf{x_s})e^{-\frac{2\pi}{L}\mathrm{i}\mathbf{r}\cdot\mathbf{x_s}},\quad\mathbf{r}\in\mathbb{Z}_K^2.$$

In particular, given $w_N\in\mathbb{H}_N$ and $\alpha\in\{-1,1,2\}$, we set

$$\left[(-\Delta_N)^\alpha w_N\right](\mathbf{x_m})=\sum_{\mathbf{r}\in\mathbb{Z}_K^2}\left(\frac{4\pi^2|\mathbf{r}|^2}{L^2}\right)^\alpha\hat{w}_K(\mathbf{r})e^{\frac{2\pi}{L}\mathrm{i}\mathbf{r}\cdot\mathbf{x_m}}. \tag{3.15}$$

This defines the discrete Laplacian if $\alpha=1$, the discrete biharmonic operator if $\alpha=2$, and the inverse Laplacian if $\alpha=-1$ and $(w_N,1)_N=0$. In this case, however, $\mathbf{r}=\mathbf{0}$ must be excluded in the summation though it is present for notational convenience. Define the following mesh-dependent *negative norm*:

$$\|w_N\|_{-1,N}=\sqrt{\left((-\Delta_N)^{-1}w_N,w_N\right)_N}. \tag{3.16}$$

In addition to replacing differential operators, the spatial integration is replaced with the *composite trapezoidal rule*. This is indicated by the symbol $(\cdot,\cdot)_N$.

There are two significant advantages of using the Fourier collocation method. First, it is accurate. For smooth, periodic functions, the two aforementioned operations are known to be spectrally accurate (see [4, pp. 53, 272], [33]). Second, it is fast. We can take advantage of the fast Fourier transform (FFT) when computing the Fourier interpolation differentiation, which reduces the computational cost significantly (see [4, pp. 52–54]).

3.3 Linear solvers for semi-implicit schemes

As mentioned above, the semi-implicit schemes require us to solve a linear equation at every time step. It turns out that the coefficient matrix of the linear system results only from differentiation. Since we are using a Fourier collocation method, we apply the FFT to solve the linear equations involved in the LMP and LBDF2 schemes.

3.4 Nonlinear solvers for fully-implicit schemes

Let us now discuss solvers for the fully implicit schemes. We employ the *preconditioned Nesterov's accelerated gradient descent method* (PAGD) as our main nonlinear solver, whose convergence theory and applications to nonlinear PDEs are found in [29], and the *preconditioned gradient descent method* (PGD) for comparison, which is studied in [21]. To summarize how these solvers work, let us explicitly state the fully discrete problem required for time marching, where we drop the superscript for the new time marching for ease of notation and so that it can be viewed as a time-independent problem on its own: given u_N^{n-1}, $u_N^n\in\mathbb{H}_N$, find $u_N\in\mathbb{H}_N$ such that

$$a_nu_N+b_nu_N^n+c_nu_N^{n-1}=M\Delta_N\frac{\delta\mathcal{E}_N}{\delta u}(\breve{u}_N), \tag{3.17}$$

where, as before, $\breve{u}_N=u_N$ or $\frac{u_N+u_N^n}{2}$ if the BDF2 or MP is used, respectively. By $\mathcal{E}_N$ we denote the discrete version of either the PFC (2.1) of FCH (2.3) energy. Namely, the one that is defined by replacing the differential operators by Fourier interpolation differentiation, and integrals by the trapezoidal rule.

Since the problem is nonlinear, we need to employ an iterative method. To this end, we recast (3.17) as a minimization problem.

Proposition 3.1 (Minimization problem)**.** Let Ω_N be given by (3.13). Then, $u_N \in \mathbb{H}_N$ solves the discrete PDE (3.17) if and only if it is a critical point of the objective

$$G_N(v_N) = \frac{1}{2Ma_n} \|a_n v_N + b_n u_N^n + c_n u_N^{n-1}\|_{-1,N}^2 + \widetilde{\mathcal{E}}_N(\check{v}_N), \tag{3.18}$$

where

$$\widetilde{\mathcal{E}}_N(\check{v}_N) = \begin{cases} \mathcal{E}_N(v_N), & \text{if BDF2 is used,} \\ 2\mathcal{E}_N\left(\dfrac{v_N + u_N^n}{2}\right), & \text{if MP is used.} \end{cases} \tag{3.19}$$

Proof. First, if v_N solves (3.17), the inverse Laplacian of $a_n v_N + b_n u_N^n + c_n u_N^{n-1}$ is well-defined since v_N has zero mean. To see this, we take the discrete inner product $(\cdot,1)_N$ on both sides of (3.17) and use (3.15) to conclude that the discrete Laplacian of any periodic grid function must have zero mean.

Next, an explicit calculation shows

$$\left\langle G_N'(v_N), w_N \right\rangle = \frac{1}{M}\left(a_n v_N + b_n u_N^n + c_n u_N^{n-1}, w_N\right)_{-1,N} + \left\langle \frac{\delta \widetilde{\mathcal{E}}_N}{\delta u}(\check{v}_N), w_N \right\rangle,$$

or, in other words,

$$G_N'(v_N) = \frac{1}{M}(-\Delta_N)^{-1}\left(a_n v_N + b_n u_N^n + c_n u_N^{n-1}\right) + \frac{\delta \widetilde{\mathcal{E}}_N}{\delta u}(\check{v}_N). \tag{3.20}$$

The factor 2 in the MP case comes from the chain rule. □

Remark 3.1 (Convexity of G_N)**.** The objective functional G_N, given in (3.18), is not convex in general. For this reason, we speak of critical points rather than minimizers.

Remark 3.2 (Discrete mass conservation)**.** The iterative solvers that we consider use a slight variant of the gradient (3.20). Namely, they use the mean-zero projection of intermediate grid functions to ensure mass conservation at the discrete level and to properly compute the discrete inverse Laplacian. More specifically, we use the negative of the following gradient as the residual:

$$G_N'(v_N) = \frac{1}{M}(-\Delta_N)^{-1}\left[\Pi_0\left(a_n v_N + b_n u_N^n + c_n u_N^{n-1}\right)\right] + \Pi_0 \frac{\delta \widetilde{\mathcal{E}}_N}{\delta u}(\check{v}_N). \tag{3.21}$$

The first mean-zero projection is not needed if we have an infinite precision since they must be mean-zero. However, due to round-off error, we need it to keep the mass conservation at the discrete level. The second projection really changes the discrete chemical potential. However, as can be seen from (3.17), adding a constant to the chemical potential does not change the main unknown u_N, i.e., the discrete Laplacian annihilates the constant added to the discrete chemical potential. Moreover, the second mean-zero projection allows the inversion of the preconditioner (see below) to be well-defined, which involves the discrete inverse Laplacian.

3.4.1 The averaged Newton preconditioner

As the names indicate, the iterative solvers used in this work involve a preconditioner. There is no general way to construct a suitable preconditioner. For this reason, we make use the energy structure of the PFC and FCH models to develop what we call an *averaged Newton preconditioner*.

To present the idea without introducing unnecessary technicalities, consider the following problem. Let $m \in \mathbb{N}$ and suppose that $G : \mathbb{R}^m \to \mathbb{R}$ is a smooth, convex objective functional with a unique minimizer $x = \text{argmin}_{\tilde{x}\in\mathbb{R}^m} G(\tilde{x})$. To find it, our starting point is to view the Newton's method as a generalization of the preconditioned gradient descent method, where the preconditioner is the second variation of the objective. That is,

$$x_{n+1} = x_n - G''(x_n)^{-1} G'(x_n).$$

Now, the goal is to come up with a preconditioner $\mathcal{G}$ (that is independent of n) that resembles $G''(x_n)$ but is easier to invert. For that, we exploit the structure of the objective function G. We know that, necessarily, $G''(x_n)$ is a linear operator, but that it may depend on the entries of x_n. To make it even simpler to invert, we remove this dependence by replacing these by averaged quantities.

Let us now move on to our case of interest. The second variation for the PFC model reads

$$G_N''(v_N)w_N = \frac{a_n}{M}(-\Delta_N)^{-1}w_N + (3u^2+1-\varepsilon)w_N + 2\Delta_N w_N + \Delta_N^2 w_N, \tag{3.22}$$

whereas the one for the FCH model is

$$\begin{aligned} G_N''(v_N)w_N = {} & \frac{a_n}{M}(-\Delta_N)^{-1}w_N \\ & + \left[F''(v_N)^2 - \eta_2 F''(v_N) - \varepsilon^2\Delta_N F''(v_N) - F'''(v_N)\left(\varepsilon^2\Delta_N v_N - F'(v_N)\right)\right] w_N \\ & + \left(F''(v_N)\varepsilon^2 - \eta_1\varepsilon^2\right)(-\Delta_N w_N) + \varepsilon^4\Delta_N^2 w_N. \end{aligned}$$

Both have the form

$$G_N''(v_N)w_N = \beta_{-2}(-\Delta_N)^{-1}w_N + \tilde{\beta}_0 w_N + \tilde{\beta}_2(-\Delta_N)w_N + \beta_4\Delta_N^2 w_N,$$

where the parameters β_{-2} and β_4 are constants while $\tilde{\beta}_0$ and $\tilde{\beta}_2$ are functions of v_N for the FCH equation. The same is true for the PFC model except $\tilde{\beta}_2$ is also a constant. Based on this observation, we consider a preconditioner of the same form, but where all the parameters are constants, that is,

$$\mathcal{L}w_N := \beta_{-2}(-\Delta_N)^{-1}w_N + \beta_0 w_N + \beta_2(-\Delta_N)w_N + \beta_4\Delta_N^2 w_N.$$

The question that remains then, is how to choose these constants. Our approach, for the PFC models, is that if the parameter is constant, then we keep the value, whereas for

those that are variable we set them to be the absolute value of its average across the domain. For the FCH model, however, we have chosen to drop several terms from $\beta_0(v_N)$ when computing its average for practical purposes. More specifically, $-\varepsilon^2\Delta_N F''(v_N)$ and $-\varepsilon^2 F'''(v_N)\Delta_N v_N$ are not included since their contributions are small (they involve ε^2) and to save computations (they involve computing a Laplacian). At every time-step, these constants are computed using the initial guess, and kept fixed throughout the iterative process. They are only recomputed once we advance in time.

3.4.2 PGD and PAGD

The PGD method is given in Algorithm 2. This method works the same way as usual gradient descent methods to minimize (3.18) except that, as mentioned above in Remark 3.2, we take the mean-zero projection on some parts of the gradient when computing the residual and apply a preconditioner to get the search direction.

Algorithm 2: Preconditioned gradient descent method (PGD).

Data: x_0, $s>0$, $\text{TOL}_{iter}>0$
initial guess, step size, tolerance;
Result: x_∞
approximate solution/minimzer;
$i=0$ initialization;
while $\|d_i\|_\infty < \text{TOL}_{iter}$ **do**
  $r_i = -G_N'(x_i)$ find the residual using (3.21);
  $d_i = \mathcal{L}^{-1}(r_i)$ find the search direction, i.e., solve $\mathcal{L}d_i = r_i$;
  $x_{i+1} = x_i + sd_i$ descent step;
  $i \leftarrow i+1$;
end

The PAGD is an accelerated version of the PGD. As explained in [29], it requires an additional parameter η. This corresponds to the *friction coefficient* of the rolling ball system associated to the PAGD. That is, the PAGD is nothing but a discretization of a second-order ordinary differential equation (ODE) describing the motion of a ball as it rolls down to the bottom of a well (the graph of the objective functional) in the presence of constant friction. For more details about this relation, we refer interested readers to [29]. As shown in [29], the choice of friction parameter η can be justified theoretically if the objective is strongly convex and the strong convexity constant is known. However, this is not the case for the PFC nor FCH equations and a different approach is needed. We could have tried to find an optimal value for the friction coefficient η by trial and error. Instead, we propose the scheme given in Algorithm 3, which is a slight variant of the PAGD and we call the *sweeping-η* or *sweeping-friction* strategy. Here, the friction coefficient η, takes a different value from a prescribed list of values at every iteration.

We note that this strategy, upon choosing a reasonable range of values, works very well in practice. In fact, it is almost as efficient as the optimally tuned, fixed, choice in terms of the number of iterations required to reach a prescribed tolerance. Moreover, the efficiency is less sensitive to a change of the range than it is to that of the fixed value. Based on the convergence theory of the PAGD in [29], an equally spaced range starting from some small number (e.g., 0.1) ending at $1/\sqrt{s}$ is possible. However, from our experience, the right endpoint can be slightly larger than $1/\sqrt{s}$. The intuition behind this can be explained by the rolling ball analogy. As opposed to the constant friction case, the sweeping-η strategy corresponds to "putting on the brake" repeatedly, say, from softly to hard. If the ball is rolling down to the bottom in this manner, our experience from driving a vehicle suggests that it will effectively stop the ball near the bottom of the valley of the landscape, an analogy to converging to a local minimum. Finally, when the PAGD or PGD is used as a nonlinear solver for time marching equations, a good option for the initial guess is the extrapolation of the previous two histories, and in fact, this is adopted in the numerical experiments that follow.

Algorithm 3: Preconditioned Nesterov's accelerated gradient descent method (PAGD) with sweeping-η.

Data: x_0, $x_{-1} := x_0$, $s > 0$, $(\eta_0, \eta_1, \ldots, \eta_{k-1}) \in \mathbb{R}^k$, $\text{TOL}_{iter} > 0$;
initial guess, fictitious previous iterate, step size, range of sweeping-η with $0 < \eta_j$ $(0 \le j \le k-1)$, tolerance;
Result: x_∞;
approximate local minimizer;
$i = 0$ initialization;
while $\|d_i\|_\infty < TOL_{iter}$ **do**
  $j \leftarrow i \mod k$ sweep over the possible choices of η;
  $\lambda_i = \frac{1-\eta_j\sqrt{s}}{1+\eta_j\sqrt{s}}$ compute the extrapolation coefficient;
  $y_i = x_i + \lambda_i(x_i - x_{i-1})$ compute the extrapolated position;
  $r_i = -G_N'(y_i)$ find the residual using (3.21);
  $d_i = \mathcal{L}^{-1}(r_i)$ find the search direction, i.e., solve $\mathcal{L}d_i = r_i$;
  $x_{i+1} = y_i + sd_i$ gradient step;
  $i \leftarrow i+1$;
end

4 Numerical experiments

4.1 Benchmark problems

Our numerical experiments can be divided into two parts: in the first part (Section 4.2), the performances of the PGD and PAGD are compared, while in the second part (Sec-

Algorithm 4: Solvers. Since BDF2 and LBDF2 require two previous solutions, the backward Euler scheme or its semi-implicit version is used for the first time-step, respectively.

Data: Model, u^0, T, time discretization, $\mathrm{dt}_{\min}$, $\mathrm{dt}_{\max}$, TOL;
PFC or FCH; initial condition; final time; MP, BDF2, LMP, or LBDF2; minimum and maximum time step sizes; stepping tolderance;
Result: $(t_1,t_2,t_3,\ldots)$, $\left(u^1,u^2,u^3,\ldots\right)$;
a sequence of times, a sequence of approximate solutions at the prescribed times;
$n=0$, $t_0=0$ Initialization. Set initial time;
$\tilde{\mathrm{dt}}_1=\mathrm{dt}_{\min}$ Initialization. Set step size;
while $t_n < T$ **do**
  $\tilde{t}_{n+1}=t_n+\tilde{\mathrm{dt}}_{n+1}$ Tentative new time;
  $\tilde{u}^{n+1}$ Tentative solution at $\tilde{t}_{n+1}$. The solution depends on the Model and Solver;
  $\hat{u}^{n+1}$ The higher order solution: (3.5) or (3.9);
  ERR The error estimator, computed via (3.6) or (3.12);
  if ERR $\leq$ TOL **then**
    Copy history, and time advances;
    $\mathrm{dt}_{n+1} \leftarrow \tilde{\mathrm{dt}}_{n+1}$;
    $t_{n+1} \leftarrow t_n+\mathrm{dt}_{n+1}$;
    $u^{n+1} \leftarrow \tilde{u}^{n+1}$;
    $n \leftarrow n+1$;
  end
  $\tilde{\mathrm{dt}}_{n+1}$ Recompute using (3.7) and (3.8);
end

tion 4.3), comparisons of fully implicit schemes and semi-implicit schemes are made. For such comparisons, we have chosen five benchmark problems, where we measure the computational cost taken by each of the proposed solvers for each problem (see corresponding sections for details on how to measure the cost). For the first part, we have ten combinations: five problems numerically are solved by two solvers, PGD and PAGD, respectively. For the second part, we have twenty combinations: four different numerical solvers, i.e., fully implicit MP and BDF2 (both equipped with the PAGD iterative solver), and semi-implicit LMP and LBDF2, are used to solve five benchmark problems. Three of the problems are evolutions according to the FCH model with different combinations of initial condition and parameters of the model, which we refer to as FCH1, FCH2, and FCH3, respectively, and the other two are according to the PFC model.

The details of the initial conditions of FCH1-3 are found in [38, Sections 5.2 (FCH1), 5.3 (FCH2), and 5.5.1 (FCH3)] and the significance of evolutions similar to FCH2 and FCH3

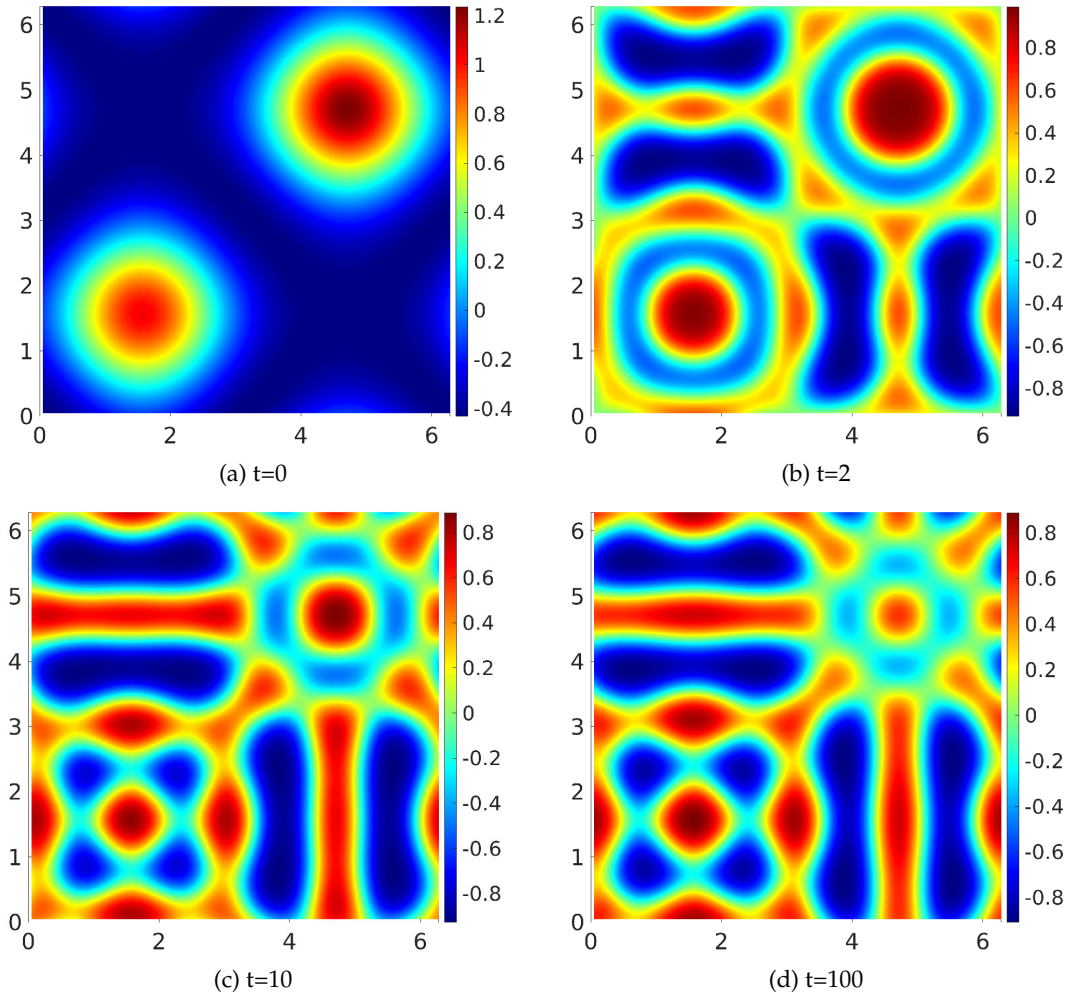


Figure 1: FCH1 evolution.

is studied in [13]. The plots of these three are displayed in Fig. 1(a) (FCH1), Fig. 2(a) (FCH2), and Fig. 3(a) (FCH3).

The other two benchmark problems are simulations of crack propagation in a crystal strip system and that of crystal growth in a supercooled liquid using the PFC model for both, which we refer to as PFC1 and PFC2, respectively. Similar computations are common in the literature, since they vividly illustrate interesting physical phenomena and highlight the versatility of the model: by tuning some model parameters this single model is able to describe many different experimentally observed states and transformations. For more discussions of these simulations, see [15–17,24,31,32].

In the following two paragraphs, we detail the initial conditions of PFC1-2 since they are not exactly the same as in the literature that they are based on. The initial condition

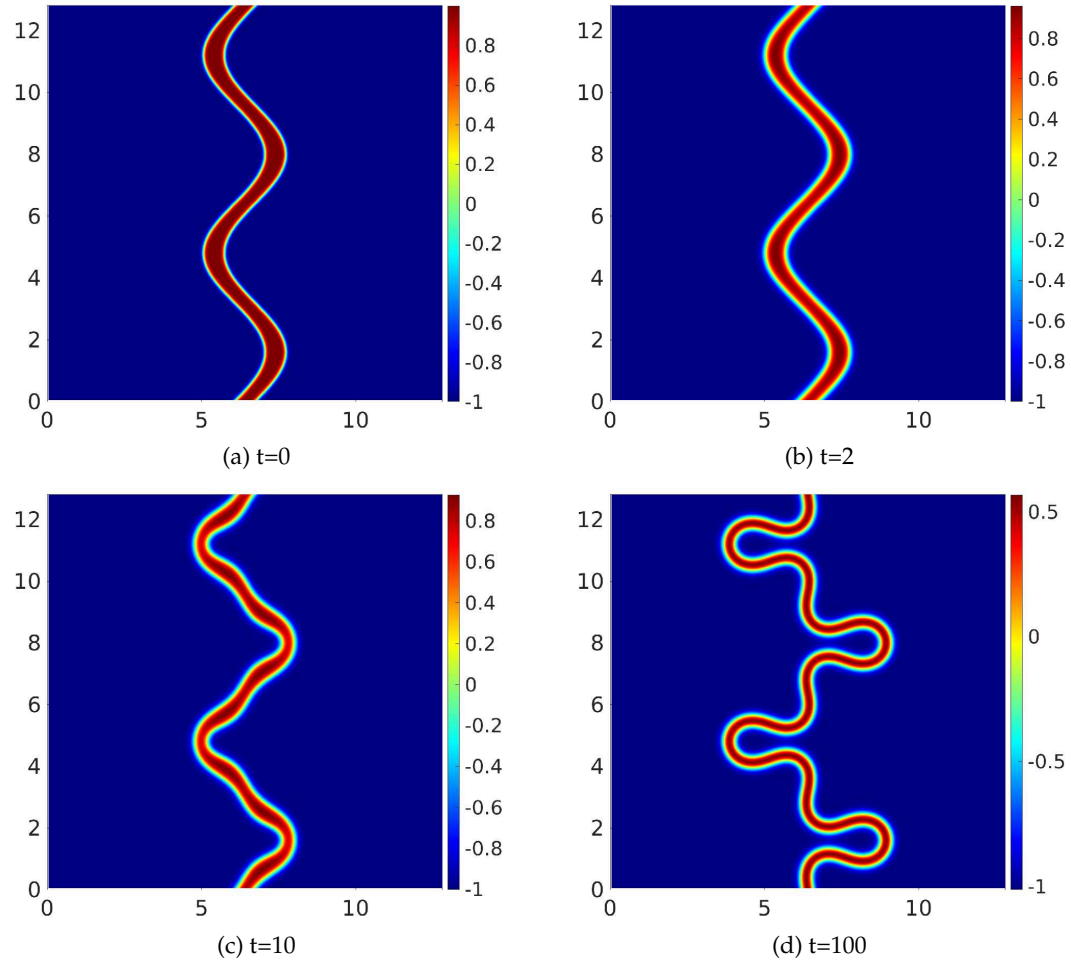


Figure 2: FCH2 evolution.

of PFC1 (see Fig. 4(a) for its plot) is set by

$$u_0(x,y) = \phi_s \cdot \psi(x,y) + \phi_\ell \big(1 - \psi(x,y)\big) + A \cdot \psi(x,y) \cdot \left[\cos\left(q_t \frac{x}{1.2}\right) \cos\left(\frac{q_t y}{\sqrt{3}}\right) - \frac{1}{2} \cos\left(\frac{2 q_t y}{\sqrt{3}}\right) \right], \tag{4.1}$$

where A and q_t are constants needed to describe the steady state density field of the solid type (see [15, Capter II. C]) and set to

$$A = \frac{4}{5}\phi_s + \frac{4}{15}\sqrt{15\epsilon - 36\phi_s^2}, \quad q_t = \frac{\sqrt{3}}{2}. \tag{4.2}$$

Following [15, Capter III. D. 2], we set $\phi_s = 0.49$, and $\phi_\ell = 0.79$. ϕ_s and ϕ_ℓ represent the temporal average of the number density of atoms for the solid and liquid state of a crys-

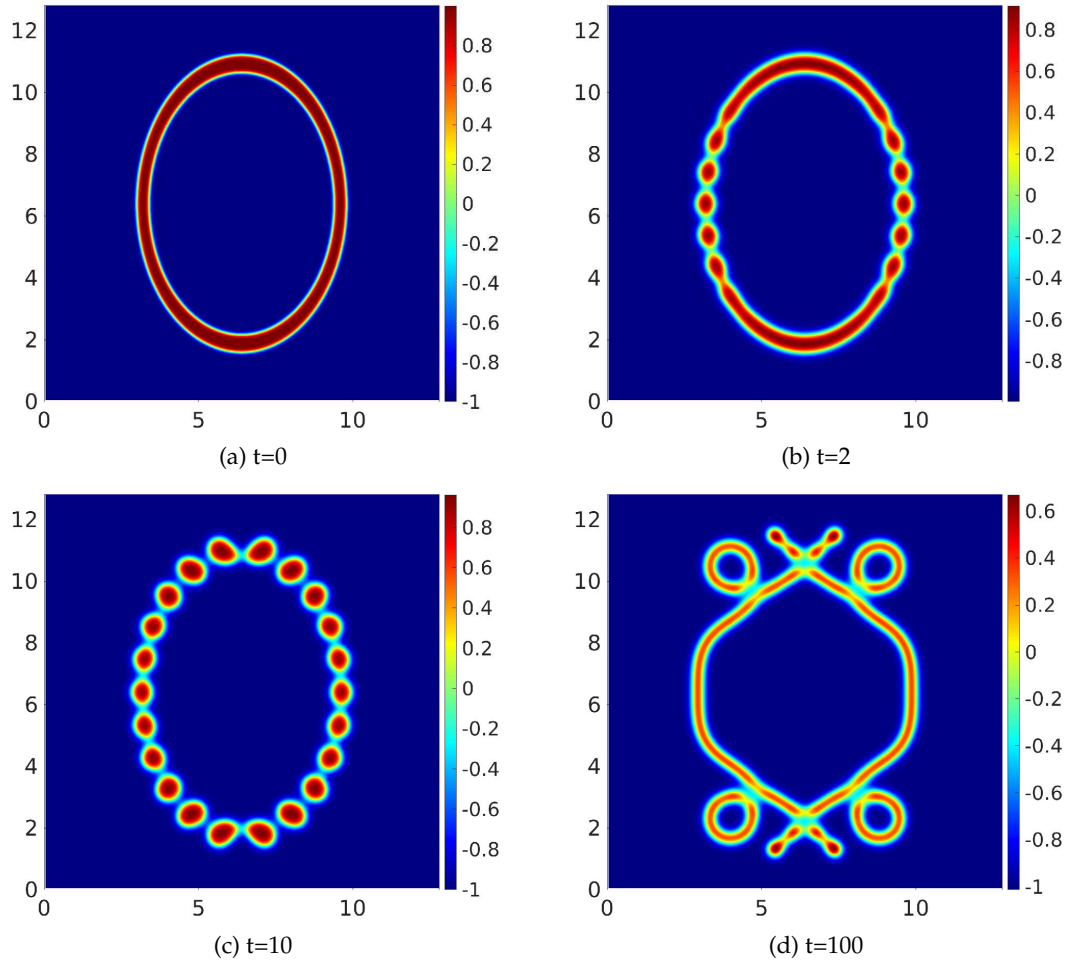


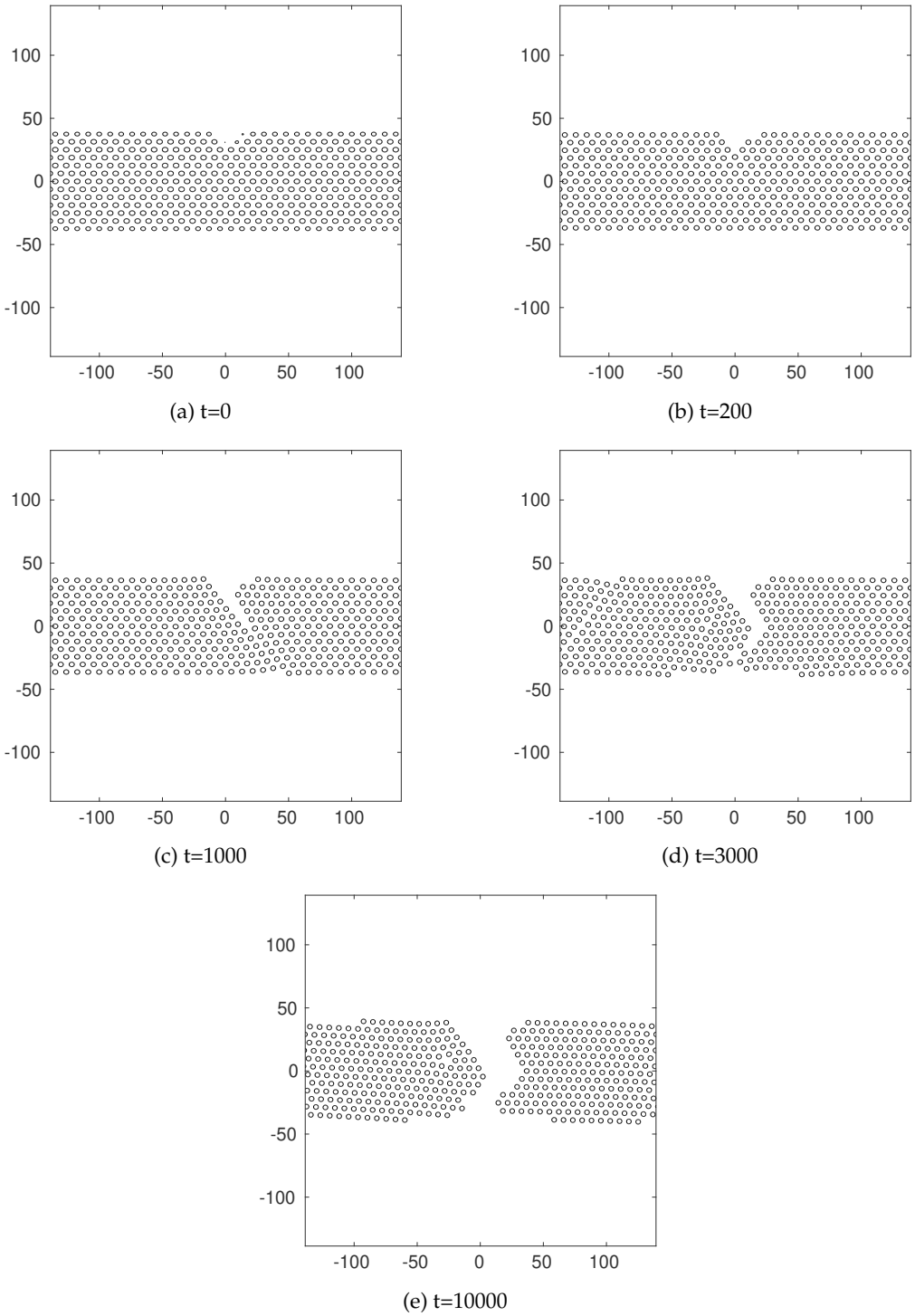
Figure 3: FCH3 evolution.

tal strip system in a liquid bath, respectively. The function ψ is a smoothed Heavyside function defined by

$$\psi(x,y) := \left(\frac{1}{2} - \frac{1}{2}\tanh\left(\frac{|y|-\gamma_1}{4}\right)\right)\left(\frac{1}{2} + \frac{1}{2}\tanh\left(\frac{\sqrt{(x-x_0)^2+(y-y_0)^2}-\gamma_2}{4}\right)\right). \quad (4.3)$$

The function ψ is used to create the chip (or a hole) on the strip as well as where the liquid regions are. The parameters x_0 and y_0 determine the location of the chip, γ_2 how big it is, and γ_1 the liquid regions.

Lastly, the initial condition for the crystal growth simulation (PFC2) used for our experiments is a miniature version of the one implemented in [24, Section 4.1]. The simulation of the whole domain, i.e., the same one as in [24] except the locations and sizes



(a) t=0 (b) t=200

(c) t=1000 (d) t=3000

(e) t=10000

Figure 4: PFC1 evolution: zero level curves of the phase variable.

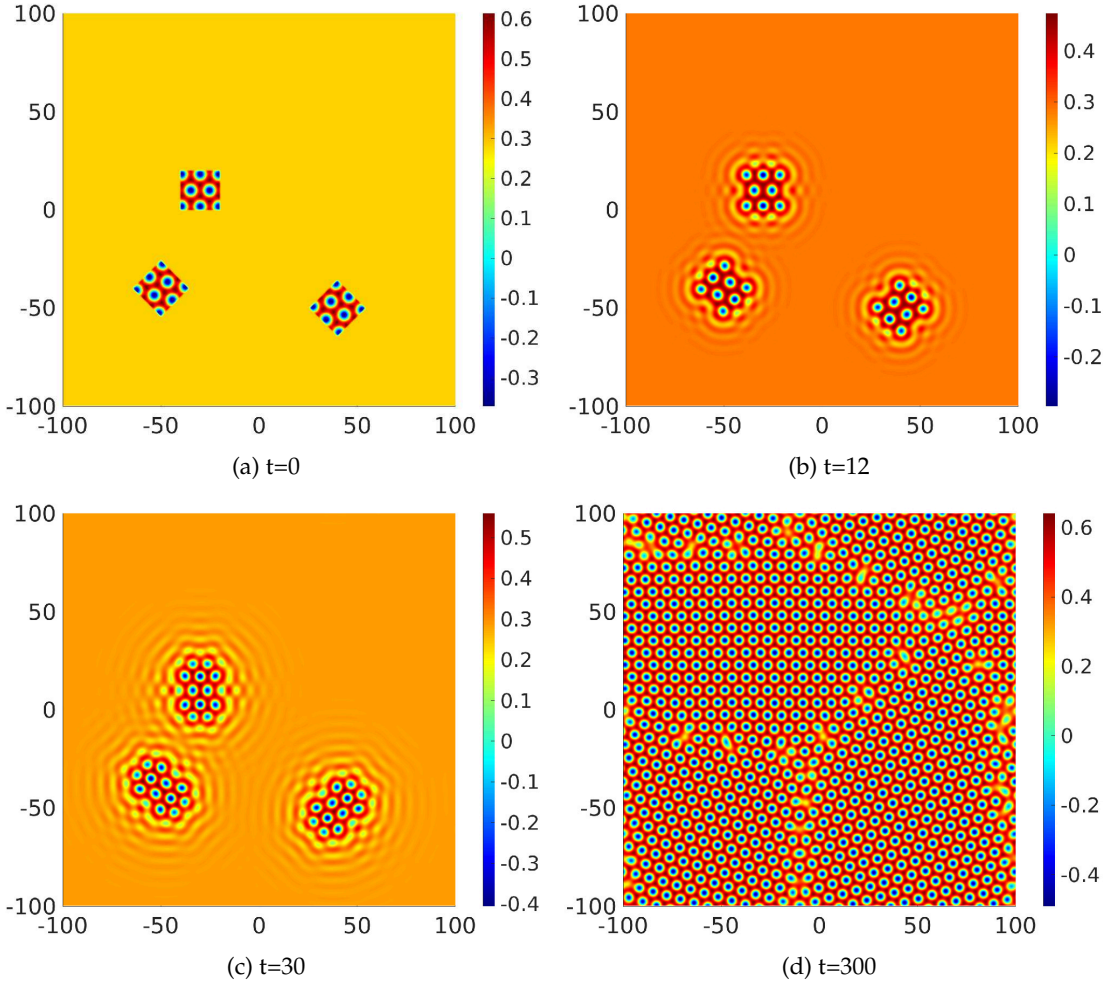


Figure 5: PFC2 evolution.

of crystallites, is reproduced in Figs. 6(a)-(e), and the PFC2 simulation considered here is displayed in Figs. 5(a)-(d). The detailed setting of PFC2 is the same as in [24, Section 4.1] except that the spatial domain and the final time are reduced to $[-L/2,L/2]^2$ with $L=200$ and $T=300$, respectively. Also, to smooth out the initial condition, it is filtered using a Gaussian filtering that is used in [38, p. 15], which is used also for FCH2-3. In the case of PFC2, however, an eight-times-finer resolution is used $\hat{N}=8N$ while $\hat{N}=2N$ for FCH2-3, where N is the original resolution set for the main experiment. See [38, p. 15] for the details about the filtering.

The snap shots of the evolutions displayed in Figs. 1-5 are generated using the best performing scheme in terms of CPU time when applied to the experiment conducted in Section 4.3. However, all solvers produce visually the same evolution with their differences detected only through numerical errors.

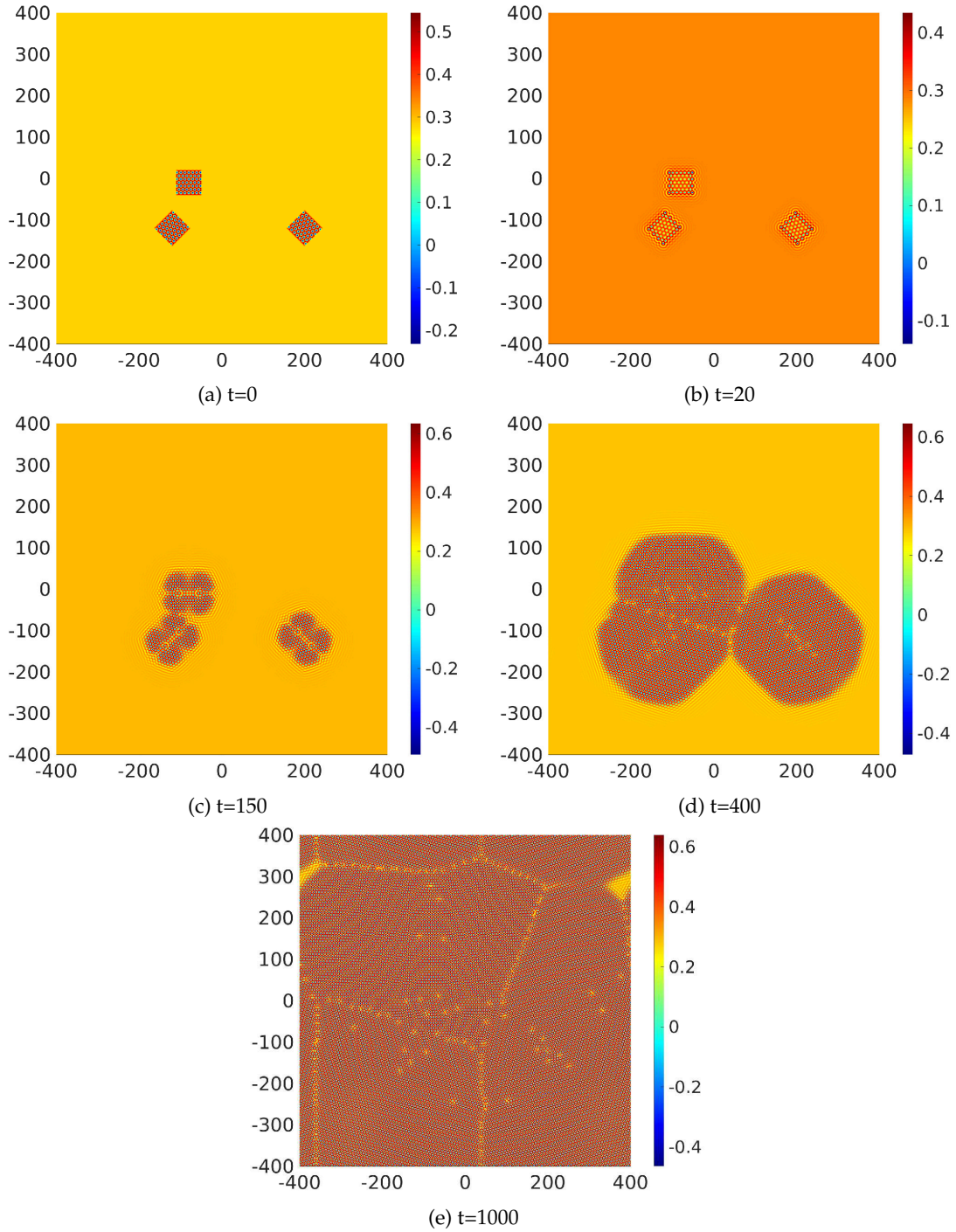
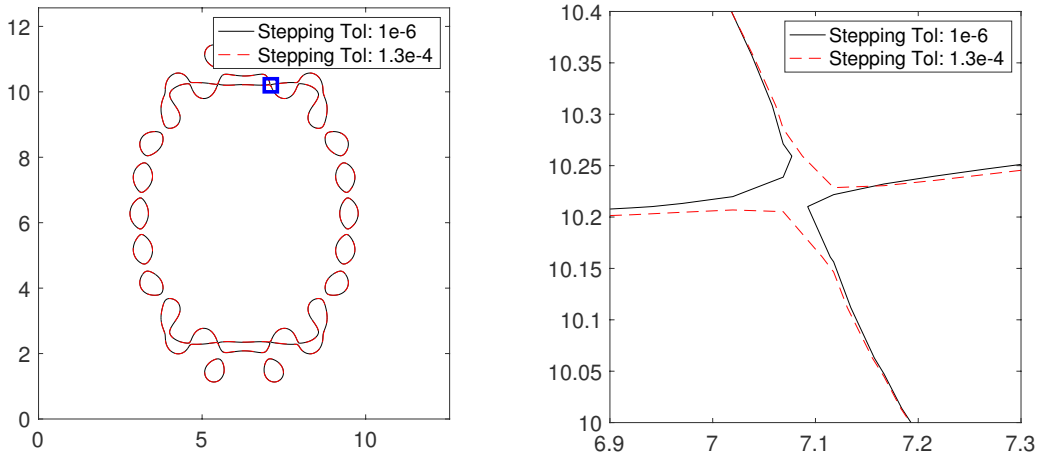


Figure 6: Simulation of crystal growth in a supercooled liquid, as originally implemented in [24]. Its miniature version is used for PFC2 benchmark computations.



(a) t=18 (b) t=18 zoom-in

Figure 7: Different evolutions of FCH3 depending on different time stepping tolerances. Both are level curves of $u(x,y,t=18)=-0.01$ obtained by BDF2 with time stepping tolerance 10^{-6} (solid black) and 1.3×10^{-4} (dashed red), respectively. The right figure is a zoom-in around the point $(x,y)=(7.1,10.2)$ (blue box).

To help put things into perspective about overall range of parameters in pursuit of accuracy (e.g., time step sizes, time stepping tolerance, etc.), Fig. 7 illustrates how a milder time stepping restriction (BDF2 with stepping tolerance 1.3×10^{-4}) leads to a different evolution in FCH3 than the one achieving the 5-digit objective (BDF2 with stepping tolerance 10^{-6}; see Section 4.3 for details on 5-digit objective): the connectivity of the level curves of the solutions at $t=18$ is different. When the tolerance is set to 10^{-6}, the maximum time step size that is actually used by the algorithm is 0.1028 while when the tolerance is 1.3×10^{-4}, it is as large as 0.5482. If the time stepping tolerance is slightly larger, say 1.5×10^{-4}, near $t=10.1561$ our algorithm does not reach the iteration tolerance before the maximum number of iterations, which is set to 1000.

Detailed settings for the benchmark comparisons are summarized in Table 1. Let us make a comment regarding the chosen range for the sweeping-η strategy. This is obtained by taking square root of a collection of equally spaced 5 numbers in the interval $[0.1,2]$. The square root is taken due to PAGD convergence theory. If the minimization problem is applied to a μ-strongly convex functional, $\eta=\sqrt{\mu}$ is the optimal choice from the convergence analysis; see [29] for more details.

4.2 Performance of PAGD and PGD

One of our main goals in this work is to show that the PAGD is a viable solver for certain types of challenging PDEs such as the PFC and FCH equations. To achieve this goal, for each one of these models, we compare the total computational cost of using PAGD vs. PGD, the latter of which is known to be an efficient solver for such problems; see [10,21]. As a measure of cost, we count the number of FFTs needed to finish the evolution. Each

Table 1: Parameter settings for the benchmark comparison.

	PDE setting	Solver setting
Common		$\mathrm{dt}_{\max}=0.5$, $\mathrm{TOL}_{iter}=10^{-10}$, sweeping-$\eta$ = ($\sqrt{0.1}$, $\sqrt{0.575}$, $\sqrt{1.05}$, $\sqrt{1.525}$, $\sqrt{2}$)
FCH1	$L=2\pi$, $\Omega=(0,L)^2$, $\epsilon=0.18$, $\eta_1=\epsilon^2$, $\eta_2=\epsilon^2$, $\tau=0$	$N=2^7$, $s=0.4$, $\mathrm{dt}_{\min}=10^{-5}$
FCH2	$L=12.8$, $\Omega=(0,L)^2$, $\epsilon=0.1$, $\eta_1=0.2$, $\eta_2=0.2$, $\tau=0$	$N=2^8$, $s=0.9$, $\mathrm{dt}_{\min}=10^{-5}$
FCH3	$L=4\pi$, $\Omega=(0,L)^2$, $\epsilon=0.1$, $\eta_1=1.45\epsilon$, $\eta_2=2\epsilon$, $\tau=0.125$;	$N=2^8$, $s=0.9$, $\mathrm{dt}_{\min}=10^{-5}$
PFC1	$L=1.2\times32\times4\times\frac{\pi}{\sqrt{3}}$, $\Omega=(-L/2,L/2)^2$, $\varepsilon=1.0$	$N=2^9$, $s=0.9$, $\mathrm{dt}_{\min}=10^{-4}$
PFC2	$L=200$, $\Omega=(-L/2,L/2)^2$, $\varepsilon=0.25$	$N=2^9$, $s=0.9$, $\mathrm{dt}_{\min}=10^{-4}$

of the benchmark problems is simulated using the solver described in Algorithm 4 with the BDF2 time discretization scheme except that the MP scheme is used for PFC2,[†] once equipped with the PGD and another time with the PAGD as a solver. The AM3 adaptive time stepping is used for FCH1-3 and PFC1 with a stepping tolerance 10^{-4} while the midAB2 stepping is utilized for PFC2 with the same stepping tolerance. For the FCH evolutions, the final time is set to $T=100$, whereas for the PFC problems, it is set to $T=10{,}000$ and $T=300$, respectively. All remaining parameter settings are the same as described in the previous section. The results are shown in Table 2.

As Table 2 shows, the PAGD solver takes as few as half the number of FFTs needed for the PGD to carry out the same simulations for most of the cases. The exception of PFC2 is discussed in the next paragraph. We emphasize once more that PGD itself is known to be an efficient solver for the FCH model (e.g., [10, 38]) and that PAGD needs only one more vector addition per iteration than PGD. Upon further inspection we observe that, as is the case at the beginning of the evolution, when the time step size is small, the two solvers perform almost equally. However, when the time step size is relatively large, the PAGD costs much less than the PGD does for each time marching. This is in line with what is reported in [29] in the sense that the acceleration comes into play when the problem is "hard".

Table 2: Number of FFTs needed for the PGD and PAGD solvers to complete the evolution. The time stepping is carried out via BDF2.

FFT	FCH1	FCH2	FCH3	PFC1	PFC2
PGD	139841	367246	883543	330908.5	24811.5
PAGD	87097	240374	440807	176887.5	25284.0

[†]BDF2 somehow makes both solvers extremely slow for PFC2.

The discussion in the preceding paragraph does not explain the results obtained for PFC2, where the computational cost (number of FFTs) of both solvers is similar. We speculate that this peculiarity is due to a well-behaving landscape of the physical energy functional associated to PFC2. From an intuitive perspective, the evolution described by PFC2 does not involve many possible bifurcations since the crystallites only grow as portions of a supercooled liquid (i.e., regions of constant phase variable) coagulate and continue the crystal pattern near the boundary of the grains. On the other hand, other simulations bear a certain symmetry in the system so that there are many possible bifurcations. As a result, there can be many more local minima in the energy functional, making solving them harder than PFC2. As reported in [29], the acceleration of PAGD (in comparison to the PGD) tends to play a bigger role in "harder" problems. The same tendency mentioned in the last two sentences of the previous paragraph is also observed in PFC2 case. However, a slightly better performance of the PGD in the beginning of the evolution (when the time step size is small) outweighs a marginally better performance of the PAGD toward the end of the simulation (when the time step size is big).

4.3 Computational cost

In order to make our comparisons as fair as possible, we take into account both accuracy and efficiency. To this end, the experiment starts by preparing preliminary data. To be specific, we choose time t of evolution (long enough for a certain morphological change to emerge) for each problem, then we choose a point (x,y) in our domain, and the triple (x,y,t) is formed. Then, we find a highly accurate solution, which is computed by the implicit Euler method with a constant time step that is so small that the difference between the computed values of $u(x,y,t)$ with the current and a ten times smaller time step is no larger than 10^{-6}, while keeping a fixed spatial grid spacing.

The implicit Euler method was chosen because, in our experience, this scheme is very robust. It is possible that the results of the numerical experiment may be different if one uses another method. However, our preliminary computations showed little difference in the point values at the reference coordinates. Following the same scheme, the difference in the reference point values computed by the implicit Euler and the LMP or LBDF2 method ranges from 1.86×10^{-9} to 7.90×10^{-7} across the five simulations,[‡] suggesting we obtain a 6-digit precision.

The procedure we now describe aims at a 6-digit precision for the highly accurate solution at the reference point (x,y,t). Let $u_{[\ell]}(x,y,t)$ be the point value at the reference point computed by setting the constant step size to 0.1^ℓ ($\ell=1,2,3,\ldots$). Suppose that we have obtained that $|u_{[\ell]}(x,y,t)-u_{[\ell+1]}(x,y,t)|<10^{-6}$. The second smallest constant time step size, i.e. 0.1^ℓ, is considered "small enough" for a 6-digit precision. Then, the reference point value is computed once again on a finer grid that has spacing smaller by a factor of 0.5 than before, and with this small enough time step, i.e., 0.1^ℓ. If this point

[‡]For FCH1-3 and PFC1, the implicit Euler is compared with the LMP while the LBDF2 is compared for PFC2.

value still differs by less than 10^{-6} from the original approximation, i.e., the one before refining the grid spacing, then it is selected as the highly accurate solution, and its point value is used in the main experiment. In fact, in all problems, this is the case. The reference coordinates (x,y,t) for each benchmark problem are as listed below and also shown in Fig. 8 (black dot).

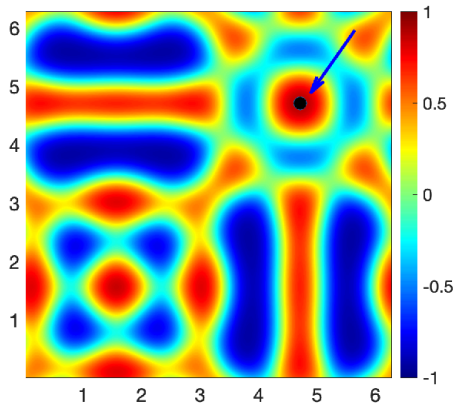
$$\begin{array}{lrrr} \text{FCH1}: & (4.71239, & 4.71239, & 10) \\ \text{FCH2}: & (7.1, & 8.85, & 10) \\ \text{FCH3}: & (6.92132, & 10.7501, & 10) \\ \text{PFC1}: & (20.6773, & 5.4414, & 1000) \\ \text{PFC2}: & (43.3594, & 14.4531, & 300) \end{array}$$

Now, the main experiment is done as Fig. 9 shows. For each combination of a problem and a solver, we start with a generous time stepping tolerance 1, which will suggest rather large time step sizes through the adaptive time stepping. We simulate the evolution with this setting until it reaches the reference time t, and obtain the point value at the spatial reference coordinate (x,y). Then, the *objective error* is computed by the difference between $u(x,y,t)$ and the value of the precomputed, highly accurate solution at our reference point. If this error is larger than the *objective tolerance* 10^{-5}, which aims at a 5-digit precision, we restart the experiment with a time stepping tolerance that is reduced by a factor of 0.1. This process is repeated until the objective error is smaller than the objective tolerance. When this occurs, we record the cost: FFT count (total number of the FFT and iFFT divided by two), the wall-clock time, and the CPU time consumed.

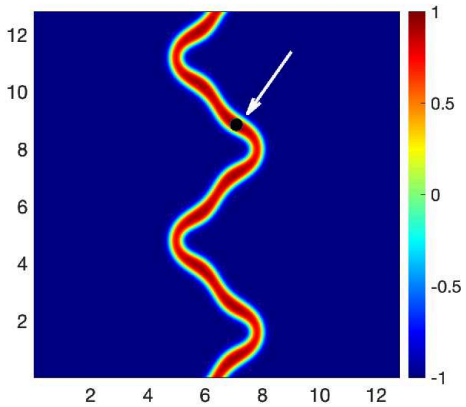
The results of this experiment are summarized in Table 3. Charts that reorganize our findings are also given in Fig. 10 so that one can compare the performance easily. As can be seen in Fig. 10, for FCH1-3, implicit schemes perform significantly better than semi-implicit schemes. In an extreme case, the MP takes less than a fifteenth CPU time than the LBDF2 does for the FCH3 benchmark problem. Interestingly, for PFC1, semi-implicit schemes perform either very poorly (LMP) or very well (LBDF2) while the performance of implicit schemes is somewhere in the middle. Even more interestingly, for PFC2, the difference in performance diverges according to the time discretization, MP-based vs. BDF2-based, rather than fully- vs. semi-implicit nature. And the worst performing solver for PFC1, namely LMP, shows an extremely good result. We suspect that the fact that the nonlinearity of the PFC equation is milder than that of the FCH equation (as one can see from their chemical potentials) explains the good performance of implicit solvers on FCH1-3. Also, although further investigations are needed, the well-behaving nature of the PFC2 evolution mentioned in the last paragraph of Section 4.2 seems to make MP-based schemes more efficient than the BDF2-based ones.

5 Conclusion

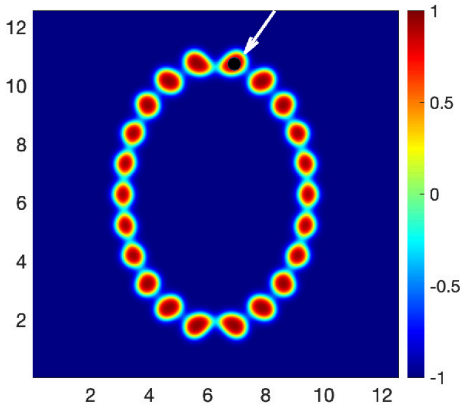
In this work, we have introduced an efficient, time-adaptive solver for the FCH and PFC equations featuring the PAGD as solver. We observed that, if solver parameters are ap-



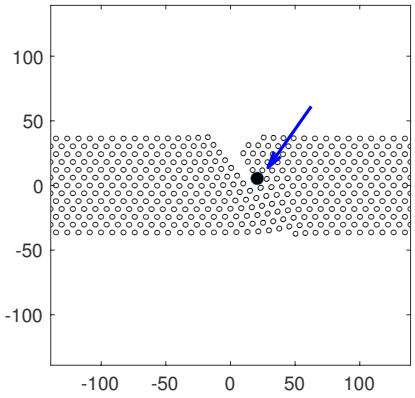
(a) FCH1: (x,y,t) = (4.71239, 4.71239, 10)



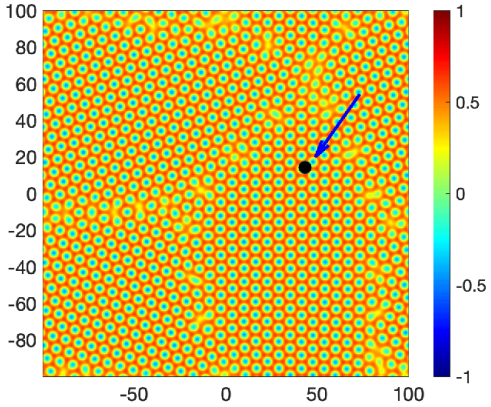
(b) FCH2: (x,y,t) = (7.1, 8.85, 10)



(c) FCH3: (x,y,t) = (6.92132, 10.7501, 10)



(d) PFC1: (x,y,t) = (20.6773, 5.4414, 1000)



(e) PFC2: (x,y,t) = (43.3594, 14.4531, 300)

Figure 8: Reference coordinates for determining accuracy in the benchmark comparisons.

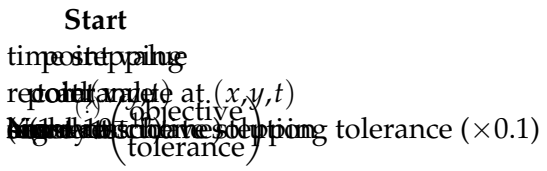


Figure 9: A flowchart to determine computational cost.

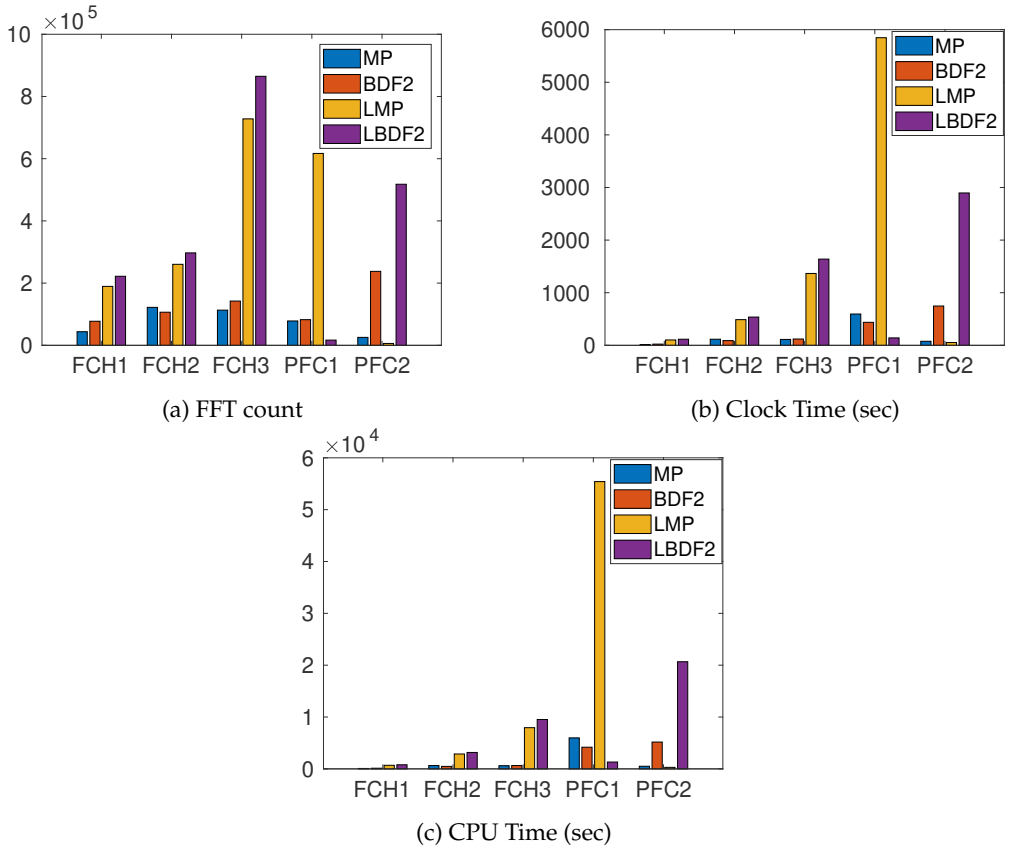


Figure 10: Comparisons between number of FFTs, the wall-clock time, and CPU time of the MP, BDF2, LMP, and LBDF2. Implicit schemes perform better than semi-implicit schemes.

Table 3: Results of benchmark experiment.

Prob	Scheme	Step tol.	Point value	Obj. err.	FFT	Clock (sec)	CPU (sec)
FCH1	MP	0.0001	0.8886817507	-2.44E-07	43769	14	81
FCH1	BDF2	1E-06	0.8886738923	-8.1E-06	77385	23	128
FCH1	LMP	1E-08	0.8886791836	-2.81E-06	189479	102	691
FCH1	LBDF2	1E-08	0.8886785837	-3.41E-06	221987	116	791
FCH2	MP	1E-05	0.9254124244	7.77E-06	122049	116	639
FCH2	BDF2	1E-07	0.9254110119	6.36E-06	106425	90	480
FCH2	LMP	1E-08	0.9254006926	-3.96E-06	260438	488	2871
FCH2	LBDF2	1E-08	0.9253997437	-4.91E-06	297146	537	3169
FCH3	MP	1E-05	0.9597908022	-8.85E-06	113089	112	611
FCH3	BDF2	1E-06	0.9598029022	3.25E-06	142293	120	643
FCH3	LMP	1E-09	0.9597965984	-3.05E-06	727975	1366	7946
FCH3	LBDF2	1E-09	0.959796049	-3.6E-06	864888	1639	9519
PFC1	MP	1	-1.190098311	-1.38E-06	78289	595	5980
PFC1	BDF2	1	-1.190094247	2.68E-06	82485	437	4178
PFC1	LMP2	0.0001	-1.190101571	-4.64E-06	616922	5847	55413
PFC1	LBDF22	0.001	-1.190105174	-8.24E-06	16785	141	1319
PFC2	MP	1	0.609267849	9.07E-08	25551	77	505
PFC2	BDF2	0.01	0.609267701	-5.75E-08	237792	747	5174
PFC2	LMP2	0.001	0.609276681	8.92E-06	6026	53	291
PFC2	LBDF22	0.001	0.609269078	1.32E-06	517858	2896	20668

propriatly chosen, a PAGD-based solver outperforms a PGD-based solver, which has been recently developed and proven to be efficient on its own.

We have also conducted an experiment so that both the accuracy and the efficiency are measured in some way, and compared the performance of two fully implicit schemes, the MP and BDF2 equipped with the PAGD, and those of two semi-implicit schemes, the LMP and LBDF2. Our results show that the implicit schemes can be a good choice from a practical point of view, particularly for highly nonlinear problems. For such problems, although some desirable properties (e.g., unique solvability) may be not available for highly nonlinear, nonconvex problems, implicit schemes tend to take less cost to yield similarly accurate simulations than the semi-implicit schemes considered in this work, provided the implicit schemes are equipped with an efficient nonlinear solver such as the PAGD. This efficient solver can be harnessed without much of tedious paramter tuning with the help of averaged Newton preconditioner and the sweeping-friction strategy. Semi-implicit schemes can be very efficient for solving equations of a milder nonlinearity if an appropriate time discretization is chosen and a *good* decomposition of the linear part and nonlinear part is chosen.

Acknowledgements

The work of J.-H. Park was partially supported by NSF grants DMS-1720213, DMS-1719854, and DMS-2012634. The work of A. J. Salgado was partially supported by NSF grants DMS-1720213 and DMS-2111228. The work of S. M. Wise was partially supported by DMS-1719854 and DMS-2012634.

References

[1] E. Asadi and M. Asle Zaeem, A review of quantitative phase-field crystal modeling of solid-liquid structures, *JOM*, 67(1):186–201, 2015.
[2] R. Backofen, S. M. Wise, M. Salvalaglio, and A. Voigt, Convexity splitting in a phase field model for surface diffusion, *Int. J. Numer. Anal. Model.*, 16(2):192–209, 2019.
[3] J. Burkardt and C. Trenchea, Refactorization of the midpoint rule, *Applied Mathematics Letters*, 107:106438, 2020.
[4] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, *Spectral methods*, Scientific Computation, Springer-Verlag, 2006. Fundamentals in single domains.
[5] H. Chen, J. Mao, and J. Shen, Optimal error estimates for the scalar auxiliary variable finite-element schemes for gradient flows, *Numer. Math.*, 145(1):167–196, 2020.
[6] K. Cheng, C. Wang, S. Wise, and Z. Yuan, Global-in-time gevrey regularity solutions for the functionalized Cahn-Hilliard equation, *Discrete Cont. Dyn. Sys. S*, 13:Paper No. 48, 20, 2020.
[7] M. Cheng and J. A. Warren, An efficient algorithm for solving the phase field crystal model, *J. Comput. Phys.*, 227(12):6241–6248, 2008.
[8] Q. Cheng, C. Liu, and J. Shen, Generalized SAV approaches for gradient systems, *J. Comput. Appl. Math.*, 394:Paper No. 113532, 19, 2021.
[9] L. Cherfils, H. Fakih, M. Grasselli, and A. Miranville, A convergent convex splitting scheme for a nonlocal Cahn–Hilliard–Oono type equation with a transport term, *ESAIM Math. Model. Numer. Anal.*, 55(suppl.):S225–S250, 2021.
[10] A. Christlieb, K. Promislow, Z. Tan, S. Wang, B. Wetton, and S. M. Wise, Benchmark computation of morphological complexity in the functionalized Cahn-Hilliard gradient flow, 2020.
[11] M. Conti, A. Giorgini, and M. Grasselli, Phase-field crystal equation with memory, *J. Math. Anal. Appl.*, 436(2):1297–1331, 2016.
[12] S. Dai, Q. Liu, and K. Promislow, Weak solutions for the functionalized Cahn-Hilliard equation with degenerate mobility, *Appl. Anal.*, 100(1):1–16, 2021.
[13] A. Doelman, G. Hayrapetyan, K. Promislow, and B. Wetton, Meander and pearling of single-curvature bilayer interfaces in the functionalized Cahn-Hilliard equation, *SIAM J. Math. Anal.*, 46(6):3640–3677, 2014.
[14] K. Elder, N. Provatas, J. Berry, P. Stefanovic, and M. Grant, Phase-field crystal modeling and classical density functional theory of freezing, *Phys. Rev. B*, 77:064107, 2007.
[15] K. R. Elder and M. Grant, Modeling elastic and plastic deformations in nonequilibrium processing using phase field crystals, *Physical review. E, Statistical, nonlinear, and soft matter physics*, 70(5 Pt 1):051605–051605, 2004.
[16] K. R. Elder, M. Katakowski, M. Haataja, and M. Grant, Modeling elasticity in crystal growth, *Phys. Rev. Lett.*, 88:245701, 2002.

[17] H. Emmerich, H. Löwen, R. Wittkowski, T. Gruhn, G. I. Tóth, G. Tegze, and L. Gránásy, Phase-field-crystal models for condensed matter dynamics on atomic length and diffusive time scales: an overview, *Advances in Physics*, 61(6):665–743, 2012.
[18] D. J. Eyre, Unconditionally gradient stable time marching the Cahn-Hilliard equation, In: *Computational and mathematical models of microstructural evolution*, Vol. 529 of *Mater. Res. Soc. Sympos. Proc.*, 39–46. MRS, Warrendale, PA, 1998.
[19] D. J. Eyre, An unconditionally stable one-step scheme for gradient systems, *Unpublished article*, 1–15, 1998.
[20] W. Feng, Z. Guan, J. Lowengrub, C. Wang, S. M. Wise, and Y. Chen, A uniquely solvable, energy stable numerical scheme for the functionalized Cahn-Hilliard equation and its convergence analysis, *J. Sci. Comput.*, 76(3):1938–1967, 2018.
[21] W. Feng, A. J. Salgado, C. Wang, and S. M. Wise, Preconditioned steepest descent methods for some nonlinear elliptic equations involving p-Laplacian terms, *J. Comput. Phys.*, 334:45–67, 2017.
[22] N. Gavish, G. Hayrapetyan, K. Promislow, and L. Yang, Curvature driven flow of bi-layer interfaces, *Physica D: Nonlinear Phenomena*, 240(7):675–693, 2011.
[23] N. Gavish, J. Jones, Z. Xu, A. Christlieb, and K. Promislow, Variational models of network formation and ion transport: Applications to perfluorosulfonate ionomer membranes, *Polymers*, 4(1):630–655, 2012.
[24] H. Gomez and X. Nogueira, An unconditionally energy-stable method for the phase field crystal equation, *Comput. Methods Appl. Mech. Engrg.*, 249/252:52–61, 2012.
[25] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving ordinary differential equations. I*, Vol. 8 of *Springer Series in Computational Mathematics*, Springer-Verlag, Second edition, 1993. Nonstiff problems.
[26] Z. Hu, S. M. Wise, C. Wang, and J. S. Lowengrub, Stable and efficient finite-difference nonlinear-multigrid schemes for the phase field crystal equation, *J. Comput. Phys.*, 228(15): 5323–5339, 2009.
[27] J. S. Jones, *Development of a fast and accurate time stepping scheme for the functionalized Cahn-Hilliard equation and application to a graphics processing unit*, ProQuest LLC, Ann Arbor, MI, 2013, Ph.D. Thesis, Michigan State University.
[28] X. Li and J. Shen, Stability and error estimates of the SAV Fourier-spectral method for the phase field crystal equation, *Adv. Comput. Math.*, 46(3):Paper No. 48, 20, 2020.
[29] J.-H. Park, A. J. Salgado, and S. M. Wise, Preconditioned accelerated gradient descent methods for locally Lipschitz smooth objectives with applications to the solution of nonlinear PDEs, *J. Sci. Comput.*, 89(17), 2021.
[30] K. Promislow and B. Wetton, PEM fuel cells: a mathematical overview, *SIAM J. Appl. Math.*, 70(2):369–409, 2009.
[31] N. Provatas, J. A. Dantzig, B. Athreya, P. Chan, P. Stefanovic, N. Goldenfeld, and K. R. Elder, Using the phase-field crystal method in the multi-scale modeling of microstructure evolution, *JOM*, 59(7):83–90, 2007.
[32] N. Provatas and K. Elder, *Phase-Field Methods in Materials Science and Engineering*, Wiley-VCH Verlag, 2010.
[33] L. N. Trefethen and J. A. C. Weideman, The exponentially convergent trapezoidal rule, *SIAM Rev.*, 56(3):385–458, 2014.
[34] C. Wang and S. M. Wise, Global smooth solutions of the three-dimensional modified phase field crystal equation, *Methods Appl. Anal.*, 17(2):191–211, 2010.
[35] 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.

[36] J. Xu, Y. Li, and S. Wu, Convex splitting schemes interpreted as fully implicit schemes in disguise for phase field modeling, 04 2016.

[37] C. Zhang and J. Ouyang, Unconditionally energy stable second-order numerical schemes for the functionalized Cahn-Hilliard gradient flow equation based on the SAV approach, *Comput. Math. Appl.*, 84:16–38, 2021.

[38] C. Zhang, J. Ouyang, C. Wang, and S. M. Wise, Numerical comparison of modified-energy stable SAV-type schemes and classical BDF methods on benchmark problems for the functionalized Cahn-Hilliard equation, *J. Comput. Phys.*, 423:109772, 35, 2020.

[39] 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.