Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 219 (2006) 404–417
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

Development of a 2-D algorithm to simulate convection and phase transition efficiently

Katherine J. Evans [a,*], D.A. Knoll [a], Michael Pernice [b]

[a] Theoretical Division, Fluid Dynamics Group (T-3), Los Alamos National Laboratory, MS B216, Los Alamos, NM 87545, USA
[b] Computer and Computational Sciences Division, Modeling, Algorithms and Informatics Group (CCS-3), Los Alamos National Laboratory, MS B265, Los Alamos, NM 87545, USA

Received 13 January 2006; received in revised form 21 March 2006; accepted 23 March 2006

Abstract

We develop a Jacobian-Free Newton–Krylov (JFNK) method for the solution of a two-dimensional convection phase change model using the incompressible Navier–Stokes equation set and enthalpy as the energy conservation variable. The SIMPLE algorithm acts as a physics-based preconditioner to JFNK. This combined algorithm is compared to solutions using SIMPLE as the main solver. Algorithm performance is assessed for two benchmark problems of phase change convection of a pure material, one melting and one freezing. The JFNK-SIMPLE method is shown to be more efficient per time step and more robust at larger time steps. Overall CPU savings of more than an order of magnitude are realized.

Keywords: Newton–Krylov; SIMPLE preconditioner; Phase change convection

1. Introduction

Accurate modeling of phase change interfaces during solidifying flows has applications to materials science and metallurgical processes. For example, castings of metal alloys can experience undesirable macrosegregation, which reduces strength and quality control. The goal is to minimize this effect through optimum design and procedure. However, monitoring solidification processes is difficult to achieve experimentally, so state of the art mathematical models of this effect are crucial.

Capturing realistic time dependent phase change interfaces within convective flow regimes is a challenge. First of all, the phase front in melting and solidifying flows is dynamic, which necessitates an accurate tracking method. Also, the release of latent heat associated with phase change creates locally stiff nonlinearities at the liquid–solid interface. Further, in the case of multi-component alloy solidification, the numerical method used must provide an accurate time evolution of the coupled temperature and concentration fields. This present

* Corresponding author. Tel.: +1 505 667 4613; fax: +1 505 665 5926.
E-mail address: kevans@lanl.gov (K.J. Evans).

0021-9991/$ - see front matter
doi:10.1016/j.jcp.2006.03.025

work is an extension of the phase change conduction problem using Jacobian-Free Newton–Krylov (JFNK) [8,10].

Phase change and the energy balance need to be incorporated into the equation set to be solved and there are several options to consider. Two of the primary options are temperature versus enthalpy formulations and the choice of velocity attenuation model. Here we use an enthalpy framework to be consistent with our previous work. The attenuation (commencement) of velocity as the material solidifies (liquefies) can be treated by incorporating an additional forcing term in the momentum equation that damps the velocities at the phase change front such that the deceleration and solidification rate within a cell are linked. For a simple pure material, the percent of latent heat released is analogous to the amount the material has solidified within a cell, so a term including liquid fraction can be used. The velocities can also be attenuated by prescribing that the viscosity becomes large as the material solidifies. For a detailed discussion refer to Gartling [4] and Voller et al. [19]. Both of these velocity treatments are models, and we use the momentum sink approach.

Once the equations to be solved are established for phase change convection, there are numerical issues to consider. This is the current focus of our research. An accurate discretization of variables in space is necessary to properly resolve the scale of convection in recirculating flow. The use of second order accurate spatial discretization on melting problems has been studied by Hannoun et al. [6], where it is established that second order spatial discretization is important for obtaining spatially converged melting simulations. Numerical accuracy issues can also be mitigated with improved time discretization. There has been less focus on this issue as compared to spatial discretization. We will briefly touch on this issue here, and focus on it in more detail in a follow-on paper.

The primary focus of this study is the solution algorithm applied to the nonlinear algebraic system, which needs to be solved at each implicit time step. The conventional approach solves time dependent equations implicitly. The SIMPLE algorithm [13] is a common solution method for phase change and natural convection problems [19,1,16,6]. The basic algorithm and its progeny (e.g. SIMPLER [13] and SIMPLEC [18]) are well developed and the basics are outlined in Section 2.1.

Recent studies have applied preconditioned JFNK solvers to the Navier–Stokes equation set [9,14] and phase change heat conduction [8,10]. In this study we utilize JFNK for phase change convection, and consider SIMPLE as a preconditioner. The details of this solution method are explained in Section 2.3. The model equations developed for the present analysis of time dependent phase change in pure materials is described in Section 2 and the JFNK and SIMPLE solution algorithms are outlined in Sections 2.1 and 2.2. The results and discussion are presented in Section 3 and summarized as conclusions in Section 4.

2. Convection phase change algorithm

The equations of motion governing the model are the time-dependent two-dimensional incompressible Navier–Stokes equations with continuity and thermodynamics. The mathematical representation of phase change is expressed with the enthalpy form of the energy equation including latent heat. The basic formulation of phase change convection is designed to match earlier phase change models. The model equations are given by

$$\frac{\partial u}{\partial t} + \nabla \cdot (\mathbf{v}u) - \frac{1}{Re}\Delta u + \frac{1}{\rho}\frac{\partial p}{\partial x} - A(H)u = 0, \tag{1}$$

$$\frac{\partial v}{\partial t} + \nabla \cdot (\mathbf{v}v) - \frac{1}{Re}\Delta v + \frac{1}{\rho}\frac{\partial p}{\partial y} - A(H)v = f_v, \tag{2}$$

$$\nabla \cdot \mathbf{v} = 0, \tag{3}$$

$$\frac{\partial H}{\partial t} + \nabla \cdot (\mathbf{v}H) + \frac{c_p}{RePr}\Delta\tau(H) = 0, \tag{4}$$

where the dependent variables are the horizontal and vertical velocities ($\mathbf{v} = \{u,v\}$), pressure (p), and total enthalpy (H), and make up the state vector $\mathbf{x} = \{u,v,p,H\}^T$ to be solved at each time step. The term $\tau(H)$ in Eq. (4) is the temperature function expressed solely in terms of H. The symbols $\boldsymbol{\Delta}$ and $\nabla\cdot$ represent the two-dimensional Laplacian and divergence operators, respectively. Refer to Table 1 for additional parameter definitions.

Table 1
Parameters and domain configuration for solidification within a square cavity (problem 1) and gallium melting within a rectangular cavity (problem 2)

Parameter	Symbol	Units (problem 2)	Problem 1	Problem 2
Domain width	$[0{:}b_x]$	m	[0:4]	[0:0.0356]
Domain length	$[0{:}b_y]$	m	[0:4]	[0:0.0635]
Rayleigh #	Ra	ND	3×10^3	7.0×10^5
Latent heat	L	J/kg	1.0	8.016×10^4
Specific heat	c_p	J/kg/K	1.0	381.5
Prandtl #	Pr	ND	1.0	0.0216
Reynolds #	Re	ND	1.0	3.37×10^6
Density #	ρ	kg/m^3	1.0	6.093×10^3
Melt temperature	T_m	K	0.0	302.78
Right wall temperature	T_l	K	0.5	301.0
Left wall temperature	T_r	K	−0.5	311.0

'ND' refers to a nondimensional value and the melt, left, and right wall temperature values are converted to enthalpy values H_l and H_r when implemented.

To treat the velocity attenuation in the solid region of the domain, a forcing term in the momentum equation as discussed in Section 1 is included. Because we are simulating pure material behavior, a simple linear relationship to the solid fraction, ϵ_s, of the material is used to damp the material velocity in the vicinity of the phase front, so $A(H) = A'\epsilon_s$. The constant A' is set to a value for each problem based on the latent heat release and other thermodynamic properties of the material being simulated. The material in the domain is either solid ($\epsilon_s = 1$), liquid ($\epsilon_s = 0$), or within a mushy zone ($0 < \epsilon_s < 1$) where ϵ_s is set to the portion of latent heat that has been released or absorbed.

In part, the behavior of the flow of the phase front velocity are characterized by the Rayleigh number, Ra, and latent heat of fusion, L, respectively. The buoyancy force due to gravity used is the Boussinesq approximation, given by $\mathbf{f}(H) = \left\{ f_u = 0, f_v = \frac{Ra}{Re^2 Pr}\tau(H) \right\}$ where the temperature is calculated as a function of enthalpy. The total enthalpy

$$H = c_p T + (1 - \epsilon_s)L, \tag{5}$$

accounts for energy changes associated with a change in temperature (T) and phase. For this algorithm study, c_p, L, and ρ are constant between phases. For a pure material, the temperature can be extracted from H as follows:

$$T = \tau(H) = \begin{cases} H/c_p & \text{if } H < c_p T_m, \\ T_m & \text{if } c_p T_m \leqslant H \leqslant c_p T_m + L, \\ (H - L)/c_p & \text{if } H > c_p T_m + L, \end{cases}$$

where T_m is the melting temperature. The problem at hand is discretized spatially using a two-dimensional Cartesian staggered fixed grid algorithm based on (1)–(4). For this mesh layout the pressure and enthalpy values lie at the center of each cell and the horizontal and vertical velocities are defined at the western and southern edges of the cells, respectively. The domain, Ω, currently consists of a rectangular cavity of equal sized cells along one direction, so $\Omega = [0{:}b_x, 0{:}b_y]$ with boundary conditions of

$$u = v = 0, \quad \text{on } \partial\Omega, \tag{6}$$

$$H(0,y) = H_l, \quad H(b_x,y) = H_r, \quad y \in [0,b_y], \tag{7}$$

$$\frac{\partial H}{\partial y}(x,0) = \frac{\partial H}{\partial x}(x,b_y) = 0, \quad x \in [0,b_x]. \tag{8}$$

This corresponds to insulating top and bottom walls and left (H_l) and right (H_r) sidewalls with a specified constant value. As mentioned, latent heat release associated with liquid to solid phase change is tracked empirically within the enthalpy framework. For the spatial orientation of the domain, the finite volume approach is used with second order centered spatial differencing. This provides a direct comparison with the majority of phase change models using the SIMPLE algorithm.

We implement two options for time discretization, which are explained as a discretized form of the general partial differential equation

$$\frac{\partial \theta}{\partial t} = \Psi(\theta), \tag{9}$$

where θ represents any one of the variables in the dependent variable state vector $\mathbf{x}$ and $\Psi(\theta)$ represents two-dimensional smooth spatial derivatives of θ. The first order accurate backward Euler (BE, also known as fully implicit) can be written as

$$\theta^{n+1} = \theta^n + \Psi(\theta^{n+1})\Delta t. \tag{10}$$

Δt is the time step size taken, and the superscript refers to the time level of the variable relative to the current time level n. All the physics within the model equations that occurs between the old and new time steps are included when using the new time level in the discretized terms. The benefit of this approach is that a larger time step can be utilized for the solution process while retaining stability.

For second order accuracy we use the BDF2 time discretization (also known as the three level fully implicit scheme), which is expressed as

$$\theta^{n+1} = \theta^n + \frac{1}{3}(\theta^n - \theta^{n-1}) + \frac{2}{3}\Psi(\theta^{n+1})\Delta t \tag{11}$$

for a constant time step size. BDF2 has a second order truncation error in time, damps spurious oscillations, and is unconditionally stable. Note that the benefit of an implicit scheme is lost with a numerical solver that cannot converge when large time steps are taken or is too expensive. A good explanation of the details of these numerical schemes can be found elsewhere [3].

2.1. SIMPLE

The SIMPLE algorithm, or semi-implicit method for pressure linked equations, solves a pressure correction equation to update the flow field and any additional state variables linked to the flow field through an iterative procedure that maintains mass continuity. The essence of the SIMPLE algorithm within the control volume framework is summarized below. Additional details regarding the basis of the SIMPLE family of solvers can be found in Patankar [13]. The model Eqs. (1)–(4) to be solved can be written more conveniently for this discussion in matrix operator form as

$$\mathbf{Q}_1(\mathbf{v}) + \rho^{-1}\nabla(p) - \mathbf{f}(H) = F(\mathbf{v}), \tag{12}$$

$$\nabla \cdot \mathbf{v} = F(p), \tag{13}$$

$$\mathbf{Q}_2(H) = F(H), \tag{14}$$

where

$$\mathbf{Q}_1 = \frac{\partial()}{\partial t} + \nabla \cdot (\tilde{\mathbf{v}}()) - \frac{1}{Re}\Delta() - A(H)(), \tag{15}$$

$$\mathbf{Q}_2 = \frac{\partial()}{\partial t} + \nabla \cdot (\tilde{\mathbf{v}}()) - \frac{1}{RePr}\Delta\tau(). \tag{16}$$

$F(\mathbf{x})$ is the set of nonlinear residuals, and the ∇ operator is the two-dimensional gradient. Note that $\tilde{\mathbf{v}}$ in operators $\mathbf{Q}_1$ and $\mathbf{Q}_2$ refers to the two-dimensional velocity vector from the previous iterative loop of SIMPLE, a linearization necessary to perform the sequential solution procedure. SIMPLE is a series of linear segregated solves of the individual equations in (12)–(14) built around a specific approximation relating velocity updates and pressure gradients. These equations can be written as one 3 by 3 matrix $\mathbf{M}$ of operators on the state vector $\mathbf{x}$,

$$\underbrace{\begin{pmatrix} \mathbf{Q}_1 & \rho^{-1}\nabla & 0 \\ \nabla\cdot & 0 & 0 \\ 0 & 0 & \mathbf{Q}_2 \end{pmatrix}}_{\mathbf{M}} \begin{pmatrix} \mathbf{v} \\ p \\ H \end{pmatrix} - \begin{pmatrix} \mathbf{f} \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} F(\mathbf{v}) \\ F(p) \\ F(H) \end{pmatrix}.$$

In practice, these equations are applied to updates to the current state vector, $\delta\mathbf{x} = \{\delta u,\delta v,\delta p,\delta H\}^{\mathrm{T}}$ rather than $\mathbf{x}$ itself. The process for building $\delta\mathbf{x}$ using this version of SIMPLE is performed as follows;

(1) evaluate δH^{n+1} with $\mathbf{Q}_2$, $\delta H^{n+1} = F(H)$
(2) calculate $\mathbf{f}^{n+1} = \{0, f_v = \frac{Ra}{Re^2 Pr}(\tau(\delta H^{n+1}))\}$
(3) evaluate $\delta\mathbf{v}^{n+1/2}$ with $\mathbf{Q}_1$, $(\delta\mathbf{v}^{n+1/2}) + \rho^{-1}\nabla p = F(\mathbf{v})$
(4) calculate $F(p)$ from (13) using $\delta\mathbf{v}^{n+1/2}$

Step (3) is applied to the u and v components of $\mathbf{v}$ sequentially and step (1) and both applications of step (3) are performed for a prescribed number of iterations. There is no explicit equation defining pressure variations, only the constraint specified by the continuity equation to be mass balanced, so matrix $\mathbf{M}$ has a zero along the main diagonal at the (2,2) location (analogous to a zero a_p for p in Patankar's notation). Thus basic linear solution techniques encounter a singularity and provide no solution. With SIMPLE, an update for pressure and velocity is determined by solving a simplified version of the momentum equation that assumes

$$\mathbf{D}\delta\mathbf{v} \simeq -\frac{1}{\rho}\nabla\delta p, \tag{17}$$

where $\mathbf{D}$ is a diagonal approximation to the discrete momentum operator $\mathbf{Q}_1$. Eq. (17) decouples the momentum at adjacent grid points and the thermal forcing $\mathbf{f}$ because it assumes their updates are zero. This is considered a reasonable assumption mainly because it keeps the iteration procedure tractable, but also because the forcing term is retained in step (3), so the final solution remains mass-balanced.

The approximate momentum equation (17) can be solved for $\delta\mathbf{v}$ with

$$\delta\mathbf{v} = -\mathbf{D}^{-1}\left(\frac{1}{\rho}\nabla\delta p\right), \tag{18}$$

and substituted into the updated version of Eq. (13) to give an expression for pressure as the dependent variable,

$$\nabla\cdot\mathbf{D}^{-1}\frac{1}{\rho}\nabla\delta p = F(p) = \mathbf{Q}_3\delta p. \tag{19}$$

Then, a linear solution for δp as well as velocity corrections can be found as follows:

(5) evaluate δp^{n+1} with $\mathbf{Q}_3$, $\delta p^{n+1} = F(p)$
(6) $\delta\mathbf{v}^{n+1} = \delta\mathbf{v}^{n+1/2} - \mathbf{D}^{-1}(\nabla\delta p^{n+1})$

As with steps (1) and (3), step (5) is performed for a prescribed number of iterations. The resulting update vector $\delta\mathbf{x}$ is added to an overall update, $\mathrm{d}\mathbf{x}$,

$$\mathrm{d}\mathbf{v} = \mathrm{d}\mathbf{v} + \alpha_v\delta\mathbf{v}, \tag{20}$$

$$\mathrm{d}p = \mathrm{d}p + \alpha_{\mathrm{p}}\delta p, \tag{21}$$

$$\mathrm{d}H = \mathrm{d}H + \alpha_{\mathrm{H}}\delta H. \tag{22}$$

The pressure update is underrelaxed by $\alpha_{\mathrm{p}} = 0.2$ for δp to balance the overestimating assumption made in Eq. (17). For all the other updates, $\alpha_v = \alpha_{\mathrm{H}} = 1$. Steps (1)–(6) and the build of $\mathrm{d}\mathbf{x}$ are repeated for a specified number of sweeps. When SIMPLE is used as the solver, the resulting $\mathrm{d}\mathbf{x}$ is added to $\mathbf{x}$, which is used to calculate a new $F(\mathbf{x})$. This whole process is repeated until the system reaches a specified convergence criteria. Typically, this criteria is met when one or two of the updates are reduced a specified amount. Rather than address phase change with linearized methods such as the effective heat capacity, here we have included the phase change directly in the iterative solve loop by using the most recent $\delta\mathbf{x}$ and computing $\tau(H)$ in the conduction term in $\mathbf{Q}_2$.

Proceeding through SIMPLE using steps (1)–(6) can be thought of as a process to approximate the solution matrix $\mathbf{M}^{-1}$ for a given time step. Because each equation builds an update individually with a set of sweeps, SIMPLE can converge slowly. The slow convergence becomes even more dramatic with increased spatial

resolution. Note that fine grids are necessary to resolve the fine scale structure of phase change convection problems [2,6]. The sequential nature of the SIMPLE solution procedure may also limit the nonlinearity of the final solution that can be found for the equation set. Next we present a solution method applied to phase change convection that can take advantage of SIMPLE but increase the robustness, accuracy, and speed of solution to nonlinear phase change convection problems.

2.2. JFNK

The Jacobian-Free Newton–Krylov (JFNK) algorithm is a nested iteration solution method, whereby an inner linear Krylov solver provides an approximation for the outer nonlinear inexact Newton's method. The details of this solution technique have been documented extensively elsewhere (e.g. Knoll and Keyes [7]) and will only be summarized here. The solution package used to implement the JFNK algorithm is an adapted form of the NITSOL solver package [15].

Inexact Newton's method solves nonlinear equation sets of the form

$$\mathbf{F}(\mathbf{x}) = 0, \tag{23}$$

where $\mathbf{x}$ is the state vector at the time advanced solution and $\mathbf{F}$ is the function of nonlinear residuals. When $\mathbf{F}(\mathbf{x})$ is expanded in a first order Taylor series, Eq. (23) becomes

$$\mathbf{J}(\mathbf{x}^k)\delta\mathbf{x}^k = -\mathbf{F}(\mathbf{x}^k), \quad \mathbf{x}^{k+1} = \mathbf{x}^k + \delta\mathbf{x}^k, \tag{24}$$

where $\mathbf{J}$ is the Jacobian for $\mathbf{F}$ and k is the nonlinear iteration index. The iteration cycle is complete when the L_2 norm of the nonlinear residual drops below the convergence criteria

$$\frac{\|\mathbf{F}(\mathbf{x}^k)\|_2}{\|\mathbf{F}(\mathbf{x}^0)\|_2} < \eta_k. \tag{25}$$

For this analysis, $\eta_k = 1 \times 10^{-5}$ except where time step convergence calculations are performed. $\mathbf{F}(\mathbf{x}^0)$ is the initial residual.

To solve the linear problem in (24) a Krylov method is used. The Krylov solution update process is performed with GMRES (generalized minimum residual), a common method for nonsymmetric matrices [17]. The 'Jacobian-Free' designation refers to a major benefit of this solver; the full Jacobian is not needed; GMRES utilizes the Jacobian times a vector, which is approximated with finite differencing as

$$\mathbf{J}\mathbf{v} \simeq \frac{\mathbf{F}(\mathbf{x} + \epsilon\mathbf{v}) - \mathbf{F}(\mathbf{x})}{\epsilon}, \tag{26}$$

where ϵ is a small perturbation.

The linear GMRES solution update is sent to the Newton solver when it drops below a specified level of convergence

$$\|\mathbf{J}\delta\mathbf{x}^k + \mathbf{F}(\mathbf{x}^k)\|_2 < \eta_l\|\mathbf{F}(\mathbf{x}^k)\|_2. \tag{27}$$

Thus we are using an inexact Newton's method. Note that the update $\delta\mathbf{x}$ is generated with the use of a preconditioner (Section 2.3). The value of the forcing term, η_l, determines the degree of efficiency and robustness with which Eq. (24) is solved. For time dependent problems, a constant value of η_l can be more efficient by insuring the right hand side of (27) does not become too small and retard convergence. This demonstrably successful version of the inexact Newton family of solvers for phase change applications [8,10] is used here.

One caveat to GMRES is that the required storage grows with each Krylov iteration; each previous basis vector element must be retained to provide the next solution approximation. The key to avoiding expensive storage overhead is to precondition the Krylov solver in order to reduce the number of Krylov iterations (and thus the number of stored vector elements) to reach a solution.

2.3. JFNK preconditioned with SIMPLE

There are a number of reasonable choices for a preconditioner and it is not often an obvious choice what type will provide superior results. One way to improve the solution efficiency and accuracy of phase change

convection problems with JFNK, but without starting from scratch is to employ SIMPLE as a preconditioner to JFNK. Recent studies show that the SIMPLE method is effective as a preconditioner for incompressible Navier–Stokes problems [14,20,11].

When a right preconditioning option is used in GMRES, the Newton loop (left equation of (24)) becomes

$$\mathbf{J}\widetilde{\mathbf{M}}^{-1}(\widetilde{\mathbf{M}}\delta\mathbf{x}) = -\mathbf{F}(\mathbf{x}), \tag{28}$$

where $\widetilde{\mathbf{M}}^{-1}$ is the preconditioning operator, represented symbolically in matrix form. The $(\widetilde{\mathbf{M}}\delta\mathbf{x})$ portion of Eq. (28) is combined during the preconditioning process, so we assign

$$\delta\mathbf{z} = \widetilde{\mathbf{M}}\delta\mathbf{x}. \tag{29}$$

First a solution is found for $\delta\mathbf{z}$ with

$$\mathbf{J}(\widetilde{\mathbf{M}}^{-1}\delta\mathbf{z}) = -\mathbf{F}(\mathbf{x}), \tag{30}$$

where $\widetilde{\mathbf{M}}^{-1}\delta\mathbf{z}$ is the latest Krylov vector element, $\mathbf{v}$. If the resulting test for tolerance using Eq. (27) is satisfied, we back out of $\delta\mathbf{z}$ to get the desired update by building $\delta\mathbf{x}$

$$\widetilde{\mathbf{M}}^{-1}(\delta\mathbf{z}) = \delta\mathbf{x} \tag{31}$$

using the preconditioner once again. The preconditioner $\widetilde{\mathbf{M}}^{-1}$ as used here is an approximate inverse, built through a prescribed number of SIMPLE sweeps. The operator $\widetilde{\mathbf{M}}$ is never actually used, rather Eq. (30) is the working equation in the algorithm.

This preconditioner is referred to as 'physics-based' because it can be written as a simplified linear version of the original nonlinear equation set to be solved. Intelligent choices can be made as to how to simplify and thus minimize the work done to create $\widetilde{\mathbf{M}}^{-1}$ yet still contain enough physical terms to provide a good approximate solution. In the preconditioner for this algorithm, first order upwind spatial discretization and first order BE time discretization is used. This makes the solution process more robust because the inverse is more diagonally dominant. Also to reduce the nonlinear coupling of equations, latent heat release and the associated velocity attenuation is set constant within a call to the preconditioner, so no phase change is created and the function $T = \tau(H)$ is restricted to be linear. The same value of damping ($\alpha_p = 0.2$) as with SIMPLE as a solver is used. Because these simplifications are applied to the preconditioner for the linear update, they only affect the efficiency of convergence to the final solution, not the accuracy.

3. Results and discussion

To assess the efficiency of the JFNK-SIMPLE algorithm for convection phase change applications, two model problems are investigated. A benchmark nondimensional freezing problem (problem 1) is selected to focus on algorithm performance for a range of temporal and spatial scales. The early melting of Gallium in a rectangular cavity (problem 2), which is a more complex model problem, is also analyzed in detail. This provides an assessment of the JFNK-SIMPLE algorithm for solutions with more detailed convection and phase change behavior.

3.1. Problem 1: Natural convection and solidification

The freezing of a nondimensional pure material in a benchmark two-dimensional square cavity problem is performed. A domain configuration matching earlier studies [12,2,19] is adopted and the parameters are listed in Table 1. Thermodynamic parameters such as specific heat are held constant except where noted. The entire domain is initially a fluid at temperature +0.5 and at time zero the left wall is set to temperature −0.5, which is below the freezing temperature (set to $T_m = 0$).

In the JFNK algorithm, the nonlinear convergence tolerance is set to $\eta_k = 1 \times 10^{-5}$ and the inexact linear tolerance defined in Eq. (27) is constant at $\eta_l = 1 \times 10^{-2}$. For the SIMPLE preconditioner, the approximate enthalpy update (step 1 in Section 2.1) was calculated with 20 sweeps using Successive Overrelaxation Routine (SOR) with an overrelaxation parameter $\omega = 1.2$. The momentum (step 3) and pressure (step 5) correction

updates were calculated with 10 and 5 sweeps, respectively, and the overall outer SIMPLE build of d**x** (steps 1–6) is set to 5 sweeps.

The simulation is run for time = 200 and the resulting temperature contours and velocity vectors are displayed in Fig. 1. Initially the whole cavity undergoes counterclockwise rotation due to buoyancy. At completion, the left wall and an adjacent layer of control volumes have frozen and exhibit negligible velocity. In the reduced domain containing cells above the melting temperature, convection is still occurring and acts to advect colder temperatures further right in the lower portion of the domain. As a result, the phase front extends further out into the main cavity.

Problem 1 is run with a set of varying time step sizes using the fully implicit backward Euler (BE) time integration formulation from Eq. (10) and centered second order spatial differencing using the JFNK-SIMPLE solver. Table 2 displays performance statistics for the JFNK-SIMPLE solver for a series of time steps. Note that as the time step increases, the number of nonlinear iterations per time step required to converge grows weakly. The speed of the simulation is normalized to the $\Delta t = 1$ run to aid comparison between algorithms.

The number of linear iterations performed per nonlinear iteration is dependent on the number of sweeps of SIMPLE in the preconditioner call and this effect is illustrated in Fig. 2. The time to complete the simulation on a 50^2 grid with a $\Delta t = 1$ is a minimum when 5 sweeps of SIMPLE in each preconditioner call are performed. As with Table 2, the CPU is normalized to the run for $\Delta t = 1$ with 5 sweeps. As indicated in the right y-axis, the number of iterations (the average number of linear GMRES iterations performed per time step) is reduced in a logarithmic fashion as the number of preconditioner calls is increased. Above 5 SIMPLE sweeps, the reduction in iterations is outweighed by the increase in cost to perform them. The cost to benefit ratio varies for each type of problem and even the same problem with different time step size. Generally, the smaller

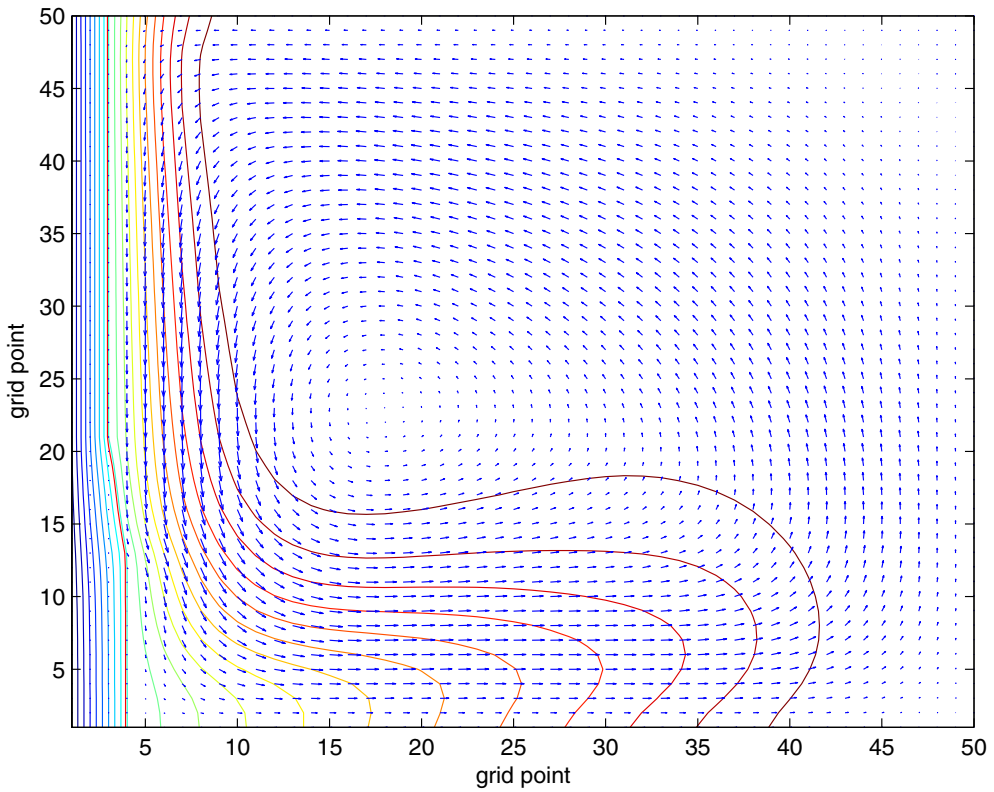

Fig. 1. Freezing of a nondimensional pure material undergoing natural convection in a 50×50 square grid after time = 200. The contours are temperature and the vectors are velocity. The red line denotes the cold edge of the phase front ($T = -0.01$). Material parameters are listed in Table 1.

Table 2
Performance of freezing benchmark problem 1 run to time = 200 using the JFNK-SIMPLE solver with BE time discretization

Time step	nlin/tstep	lin/nlin	CPU
0.01	2.66	5.06	32.7
0.05	2.89	4.12	6.0
0.1	3.33	4.20	3.6
0.5	4.31	6.95	1.4
1.0	5.26	8.48	≡1
2.0	6.27	11.70	0.86

The CPU units are normalized with the $\Delta t = 1$ run. Labels 'nlin/tstep' and 'lin/nlin' refer to the average number of nonlinear iterations per time step and linear iterations per nonlinear iteration, respectively.

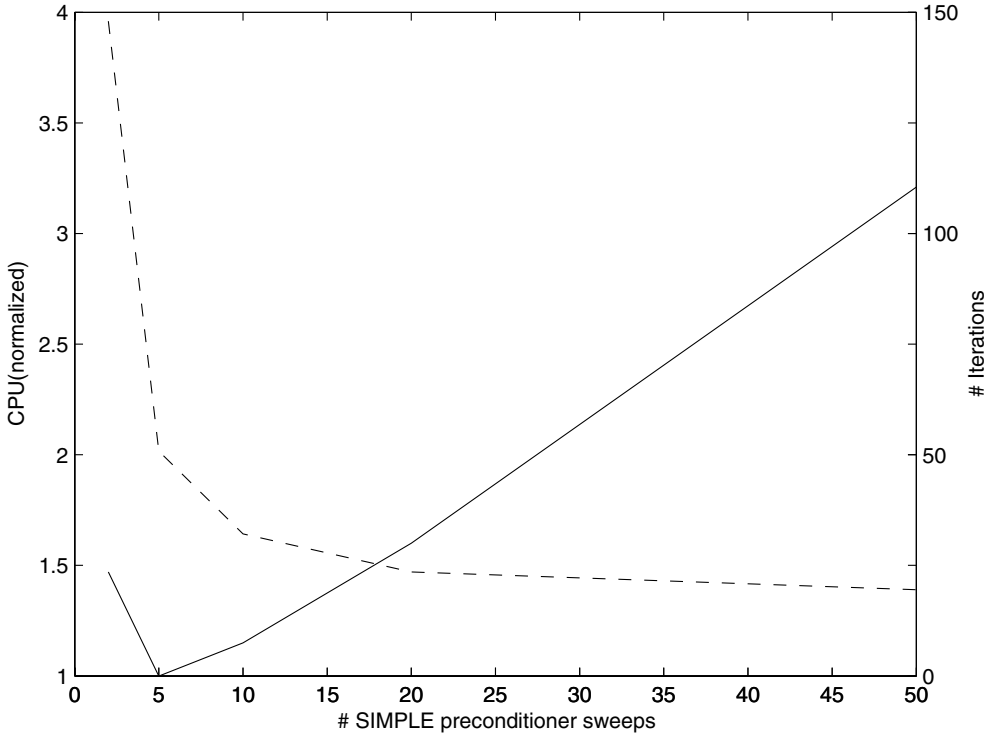

Fig. 2. CPU (normalized to $\Delta t = 1$ and 5 sweeps) of problem 1 (solid line, left y-axis) and average sum of nonlinear plus linear iterations (dashed line, right y-axis) per time step on a 50^2 grid for $\Delta t = 1$ as a function of SIMPLE sweeps within each call to the preconditioner.

time step size requires less work from the preconditioner to reach a minimum in CPU time because the problem is less stiff.

The same simulation is performed using the SIMPLE algorithm as the main solver (no JFNK). The full BE discretized equation set with phase change used in the JFNK algorithm is used for SIMPLE. Instead of 5 sweeps of the SIMPLE steps outlined in Section 2.1, the number of sweeps necessary to converge to the same criteria as JFNK-SIMPLE ($\eta_k = 1 \times 10^{-5}$) in Eq. (25) are performed. After a prescribed number of SIMPLE sweeps, the d$\mathbf{x}$ update is added to $\mathbf{x}$ to calculate the nonlinear residual using Eq. (25). The necessary number of sweeps through SIMPLE is found via trial and error and varies with time step and the parameters of the problem being solved. Often, SIMPLE-based simulation studies only converge a combination of the mass balance condition and another criteria such as energy balance [1]. However, determining convergence by measuring the nonlinear residual is more rigorous and allows a direct comparison to JFNK-SIMPLE simulations.

Results with SIMPLE as the solver are displayed in Table 3 with corresponding runs and convergence tolerance using JFNK-SIMPLE. SIMPLE does not converge for this configuration above Δt of about 0.5. On a

Table 3
Comparison of algorithm performance for JFNK-SIMPLE and SIMPLE with the freezing benchmark problem 1 run to time = 200 on a 50^2 grid using the same set-up as described in Table 2, including CPU units normalized to the JFNK-SIMPLE run for $\Delta t = 1$

Solver	Time step	Iterations/tstep	CPU
JFNK	0.01	17.2	32.7
SIMPLE	0.01	600	193.3
JFNK	0.05	15.8	6.0
SIMPLE	0.05	400	28.0
JFNK	0.1	18.3	3.6
SIMPLE	0.1	500	16.7
JFNK	0.5	35.3	1.4
SIMPLE	0.5	1600	10.5

50^2 grid, for $\Delta t = 0.05$ and 0.1, the SIMPLE solver completes the simulation about 4.6 times slower than JFNK-SIMPLE and for $\Delta t = 0.5$ SIMPLE is 7.5 times slower. For the 0.01 time step, the ratio of CPU time for SIMPLE and JFNK-SIMPLE is 5.8:1. The initial norm is small so both JFNK-SIMPLE and SIMPLE require more effort to reduce the relative norm sufficiently.

Another angle to analyze algorithm efficiency can be found by comparing the number of SIMPLE sweeps needed to complete the simulation for SIMPLE and JFNK-SIMPLE for a given time step size. Unlike the linear GMRES loop within JFNK, the effort of SIMPLE is not minimized with a linear tolerance test. When the number of SIMPLE loops for an average time step are summed, the ratio of loops in SIMPLE verses JFNK-SIMPLE ranges from 6.7:1 to 10.6:1. This ratio compared to the ratio of average CPU time per time step is almost a constant, 1.48 ± 0.07:1. Thus the reduction of SIMPLE sweeps by using JFNK-SIMPLE is likely responsible for the reduction in CPU, with an 'overhead' of about 48%. This value could be probed for improvement through optimization efforts, however the reduction in SIMPLE sweeps is far more dramatic than the 'overhead' cost. The salient point is that JFNK acts as an accelerator to the SIMPLE method for problem 1 and SIMPLE algorithm's efficiency degrades relative to JFNK-SIMPLE with increasing time step size.

The JFNK algorithm becomes more efficient and robust relative to SIMPLE with finer grids. Finer grid simulations were performed for problem 1 and the algorithm performance is displayed in Table 4. There is no additional convection structure, although the front position moves because it can be resolved more accurately. For a 100^2 grid, the largest time step where SIMPLE achieves convergence was at $\Delta t = 0.05$, and at this time step for both algorithms, JFNK-SIMPLE is 8.8 times faster than SIMPLE. Run at the largest time step size of convergence, JFNK-SIMPLE is 59 times faster on a 100^2 grid. For a 200^2 grid, SIMPLE converged for Δt no greater than 0.001 whereas JFNK-SIMPLE converged at a maximum $\Delta t = 0.1$. Using these time step sizes, JFNK-SIMPLE was almost 200 times faster than SIMPLE.

Along with increased efficiency, the algorithm exhibits first and second order accuracy with respect to time. Because there is no analytical solution from which to compare, the simulation accuracy is assessed by comparing runs to a base run with a time step size of 1×10^{-3} s using BDF2 time discretization, centered spatial differencing, and a coarser 10^2 grid. Error is measured with a global L_2 norm on a fixed spatial grid, and the order of error as a function of time step is given by the slope of the line of error verses time step size on a log–log plot. The slopes for BE and BDF2 time discretizations for this problem are 0.99 and 1.92, which indicates first and second order accurate methods in time. The difference in error magnitude for a given time step ranges from about 1.5 to 2.5 orders of magnitude for time steps from 1.0 to 0.1, respectively. Thus, running the simulation at a time step of 1.0 using BDF2 is more accurate than running at an order of magnitude smaller time step using BE.

3.2. Problem 2: Natural convection and melting of Gallium

The convection phase change algorithm described in Section 2 is also applied to the melting of pure Gallium (problem 2). The parameters and physical properties of Gallium used are listed in Table 1. This problem was chosen because of previous simulation studies [1,2,6] and a comprehensive experimental study by Gau and

Table 4
Algorithm performance applied to problem 1 with the simulation run using time step sizes close to the largest time step of convergence (noted as 'Max TSS') for increasingly finer grids

Grid	Max TSS SIMPLE	Max TSS JFNK-SIMPLE	CPU (ratio)
50×50	0.5	2.0	38:1
100×100	0.05	1.0	59:1
200×200	0.001	0.1	191:1

Here CPU units are presented as ratios of SIMPLE to JFNK-SIMPLE. The finer grids were run for a portion of the full simulation time.

Vistanka [5] from which to compare. We set the domain size to a portion of these studies because the current focus is at the left edge where melting is occurring at early times. Below, we discuss a fine grid simulation of the full domain.

The melting simulation is run for 200 s using a grid size of 40 by 100 cells, which corresponds to a grid with cell sizes of 0.89 mm horizontal by 0.635 mm vertical. The resulting streamfunction values are displayed in Fig. 3 as contours in the melt along with the location of the phase front. As with problem 1, the nonlinear and linear tolerance parameters are set to $\eta_k = 1 \times 10^{-5}$ and $\eta_l = 1 \times 10^{-2}$, respectively. An analysis to maximize effectiveness of the preconditioner for problem 2 analogous to Fig. 2 determined that 20 sweeps of SIMPLE minimized the CPU time required to perform the simulation for the JFNK-SIMPLE solver. Within each SIMPLE loop the same number of sweeps of the individual steps were taken as with problem 1.

Table 5 displays statistics for problem 2 using the JFNK-SIMPLE and SIMPLE solvers across a range of time step sizes. For consistency, first order time discretization (BE) is used. Like the freezing simulation, JFNK-SIMPLE displays a relatively weak growth of iterations as the time step is increased (about a factor of 10 over a time step increase of 200). The maximum time step size where the problem 2 simulation can be run to convergence using a 40×100 size grid is about 100 times greater with JFNK-SIMPLE than SIM-

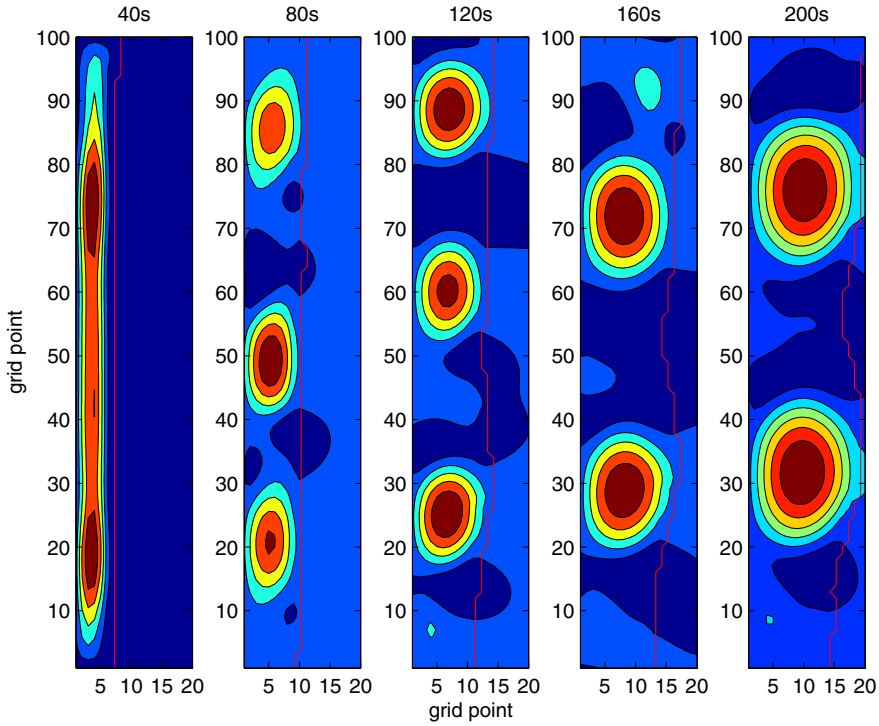

Fig. 3. Melting and convection of pure Gallium in a 40×100 grid every 40 s. Left 20×100 half of the domain is displayed. Material parameters (problem 2) are listed in Table 1. The contours are values of streamfunction and the red line denotes the cold edge of the phase front ($T = 302.77$ K). For this figure BDF2 second order time differencing and a time step of 0.1 s is used.

Table 5
Performance of Gallium melting problem 2 run to 5 s using both JFNK-SIMPLE and SIMPLE solvers with FI time discretization on a 40×100 grid

Solver	Time step (s)	nlin/tstep	Avg. lin/nlin	CPU (ratio)
SIMPLE	0.001	4.	50[a]	41.9
JFNK-S	0.001	1.15	2.2	11.4
JFNK-S	0.005	1.84	2.9	4.5
JFNK-S	0.01	2.03	2.9	2.5
JFNK-S	0.05	2.76	7.0	1.4
JFNK-S	0.1	2.92	9.9	≡1

CPU time is normalized to the 0.1 s simulation using JFNK-SIMPLE.
[a] For the SIMPLE solver, the number of linear SIMPLE sweeps within each nonlinear function evaluation is prescribed to the minimum number whereby nonlinear tolerance can be achieved.

PLE. As a result, this simulation can be completed almost 42 times faster using JFNK-SIMPLE than SIMPLE on this coarse grid. Referring to Table 5, 50 sweeps of SIMPLE through the steps outlined in Section 2.1 is the minimum number needed to converge the SIMPLE solver for problem 2. At about a time step of 0.005 s, SIMPLE could reduce the residual norm by about 10^3 before beginning to increase and eventually diverge.

Performance results for JFNK-SIMPLE are similar at finer grids and matching the full domain of Gau and Vistanka's [5] experimental study. With the same configuration as for Table 5 but with a full geometry 200^2 grid, JFNK-SIMPLE took an average of 3.32 nonlinear iterations per time step and 18.4 linear per nonlinear iterations to reach convergence at a time step of 0.05 s. Using a mesh size equal to the Hannoun et al. [6] study,

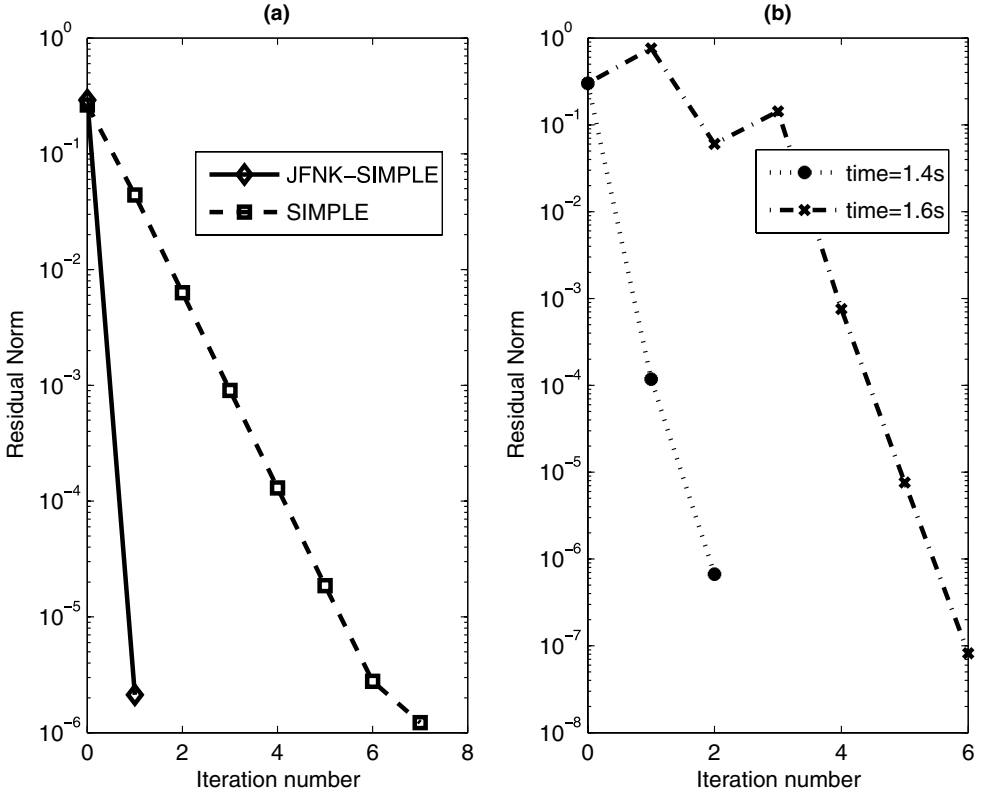

Fig. 4. Nonlinear residual norm for individual iterations of problem 2 using BE time discretization and $\eta_k = 1 \times 10^5$. (a) Convergence history of JFNK-SIMPLE and SIMPLE at time = 0.2 s using a time step size of 0.005 s. (b) JFNK-SIMPLE at time 1.4 and 1.6 s using a time step size of 0.2 s.

which they demonstrated to be spatially converged, and the current domain size (240×600) the algorithm converged with an average of 4.36 nonlinear and 19.6 linear per nonlinear iterations per step of 0.01 s after 10 s.

The convergence history for JFNK-SIMPLE and SIMPLE provides some insight to the algorithm behavior for problem 2. Fig. 4 displays the nonlinear residual norm as a function of iteration number k for a set of time steps. In Fig. 4(a), JFNK-SIMPLE and SIMPLE are presented at time 0.2 s using a time step size of 0.005 s. Both solution methods converge at this time, but the SIMPLE residual norm takes seven iterations to reach the same nonlinear tolerance as JFNK-SIMPLE with one iteration. At large time steps using JFNK-SIMPLE, the residual norm occasionally increases within a time step before eventually converging. This is possibly associated with new grid cells changing phase or additional convection structure. One example of this is displayed in Fig. 4(b) using a time step of 0.2 s at time 1.4 and 1.6 s. At 1.4 s, the algorithm requires 2 iterations to drop below the convergence tolerance (typical value for this run is 3). At each new iteration, the norm is reduced. At the 1.6 s time step however, the norm fluctuates near the initial norm for several iterations before locking onto the solution. A minimum of quantified error and CPU time of the final solution are the two main goals for a simulation, and JFNK-SIMPLE achieves both.

Problem 2 complexities arise early in the melting simulation because the aspect ratio of the melted region is very high. As a result, instabilities in the convection melt induce multiple roll cell structure that significantly influences the shape of the phase front. Therefore an accurate solution method to resolve these structures is required. Earlier studies that have been able to take advantage of sufficiently fine scale grids and second order spatial discretization [2,6] have also demonstrated the existence of multiple cell structures. The present work has focused on algorithm efficiency within reasonable parameters for solidification; further work addressing time accuracy issues on spatially converged grids is forthcoming.

4. Conclusions

The Jacobian-Free Newton–Krylov (JFNK) solution method preconditioned with the SIMPLE algorithm is applied to two-dimensional phase change convection; the qualitative solution structure matches earlier studies with similar parameters. The low storage benefit of the segregated algorithm SIMPLE is exploited as a preconditioner, which creates an update for the dependent variable set within the inner Krylov loop. When compared SIMPLE as a stand alone solver, the performance of JFNK-SIMPLE is superior in terms of efficiency. For a simple nondimensional freezing problem, JFNK is two orders of magnitude faster. For a more complex Gallium melting problem with multiple celled convection phase change behavior on a coarse grid, JFNK can run more than 40 times faster.

When implemented with BE and BDF2 time discretizations schemes, JFNK-SIMPLE has also demonstrated first and second order accuracy, respectively, and the error of the model is quantified. The error of the second order method is up to two and a half orders of magnitude less than first order at a 0.2 time step for a nondimensional freezing simulation, with the exact gain being time step dependent. The implications of these results using second order methods is that the ability to model more complicated physics can be performed with increased confidence. The nature of multiple celled convection and its interaction with the shape of the phase front can be probed with increased accuracy. By using the JFNK-SIMPLE algorithm, the full nonlinear equations for phase change convection are solved simultaneously. Thus, JFNK-SIMPLE is more robust and efficient with increased time step and more refined grid.

Acknowledgments

This work was carried out under the auspices of the National Nuclear Security Administration of the US Department of Energy at Los Alamos National Laboratory under Contract No. W-7405-ENG-36 (LA-UR-06-0051).

References

[1] A.D. Brent, V.R. Voller, K.J. Reid, Enthalpy porosity technique for modeling convection-diffusion phase change: Application to the melting of a pure metal, Numer. Heat Transfer 13 (1988) 297–318.

[2] J. Danzig, Modelling liquid–solid phase changes with melt convection, Int. J. Numer. Methods Eng. 28 (1989) 1769–1785.
[3] C. Fletcher, Computational Techniques for Fluid Dynamics, Springer-Verlag, Berlin, 1991.
[4] D.K. Gartling, Finite element analysis of convective heat transfer problems with change of phase, in: K. Morgan, C. Taylor, C. Brebbia (Eds.), Computer Methods in Fluids, Pentech Press, London, 1980, pp. 257–284.
[5] C. Gau, R. Vistanka, Melting and solidification of a pure metal from a vertical wall, J. Heat Transfer 108 (1986) 171–174.
[6] N. Hannoun, V. Alexiades, T.Z. Mai, Resolving the controversy over tin and gallium melting in a rectangular cavity heated from the side, Numer. Heat Transfer B 44 (2003) 253–276.
[7] D. Knoll, D. Keyes, Jacobian-free Newton–Krylov methods: a survey of approaches and applications, J. Comput. Phys. 193 (2004) 357–397.
[8] D.A. Knoll, D. Kothe, B. Lally, A new nonlinear solution method for phase-change problems, Numer. Heat Transfer B 35 (1999) 439–459.
[9] D.A. Knoll, V.A. Mousseau, On Newton–Krylov methods for the incompressible Navier–Stokes equations, J. Comput. Phys. 163 (2000) 262–267.
[10] D.A. Knoll, W.B. Vanderheyden, V.A. Mousseau, D. Kothe, On preconditioning Newton–Krylov methods in solidifying flow applications, SIAM J. Sci. Comput. 23 (2001) 381–397.
[11] C. Li, C. Vuik, Eigenvalue analysis of the SIMPLE preconditioning for incompressible flow, Numer. Lin. A. Appl. 11 (2004) 511–523.
[12] K. Morgan, A numerical analysis of freezing and melting with convection, Comp. Meth. App. Eng. 28 (1981) 275–284.
[13] S.V. Patankar, Numerical Heat Transfer and Fluid Flow, Hemisphere Publishing Corp., New York, 1980.
[14] M. Pernice, M. Tocci, A multigrid-preconditioned Newton–Krylov method for the incompressible Navier–Stokes equations, SIAM J. Sci. Comput. 23 (2001) 398–418.
[15] M. Pernice, H.F. Walker, NITSOL: a Newton iterative solver for nonlinear systems, SIAM J. Sci. Comput. 19 (1998) 302–318.
[16] P. Prescott, F.P. Incropera, Numerical simulation of a solidifying Pb–Sn alloy: The effects of cooling rate on thermosolutal convection and macrosegregation, Metall. Trans. B 22B (1991) 529–540.
[17] Y. Saad, M.H. Schultz, GMRES: generalized minimum residual algorithm for solving non-symmetric systems, SIAM J. Sci. Stat. Comput. 7 (1986) 856–869.
[18] J.P. van Doormaal, G.D. Raithby, Enhancements of the SIMPLE method for predicting incompressible fluid flows, Numer. Heat Transfer 7 (1984) 417–440.
[19] V. Voller, M. Cross, N.C. Markatos, An enthalpy method for convection/diffusion phase change, Int. J. Numer. Methods Eng. 24 (1987) 271–284.
[20] C. Vuik, A. Saghir, G. Boerstoel, The Krylov accelerated SIMPLE(R) method for flow problems in industrial furnaces, Int. J. Numer. Methods Fluids 33 (2000) 1027–1040.