Journal of Computational Physics 228 (2009) 8566–8588

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

An approximate block Newton method for coupled iterations of nonlinear solvers: Theory and conjugate heat transfer applications

Andrew Yeckel *, Lisa Lun, Jeffrey J. Derby

Department of Chemical Engineering and Materials Science and Minnesota Supercomputing Institute, University of Minnesota, 421 Washington Avenue SE, Minneapolis, MN 55455-0132, USA

ARTICLE INFO

Article history:
Received 12 March 2009
Received in revised form 15 July 2009
Accepted 3 August 2009
Available online 12 August 2009

PACS:
81.10.−h
02.60.−x
02.70.−c

Keywords:
Crystal growth
Multiscale coupling
Multiphysics coupling
Approximate Newton methods
Block Gauss–Seidel methods
Modular iterations

ABSTRACT

A new, approximate block Newton (ABN) method is derived and tested for the coupled solution of nonlinear models, each of which is treated as a modular, black box. Such an approach is motivated by a desire to maintain software flexibility without sacrificing solution efficiency or robustness. Though block Newton methods of similar type have been proposed and studied, we present a unique derivation and use it to sort out some of the more confusing points in the literature. In particular, we show that our ABN method behaves like a Newton iteration preconditioned by an inexact Newton solver derived from subproblem Jacobians. The method is demonstrated on several conjugate heat transfer problems modeled after melt crystal growth processes. These problems are represented by partitioned spatial regions, each modeled by independent heat transfer codes and linked by temperature and flux matching conditions at the boundaries common to the partitions. Whereas a typical block Gauss–Seidel iteration fails about half the time for the model problem, quadratic convergence is achieved by the ABN method under all conditions studied here. Additional performance advantages over existing methods are demonstrated and discussed.

1. Introduction

There has long been interest in efficient and flexible techniques for coupling systems of equations that are solved in some segregated fashion. The most powerful techniques are those that can be applied when the equations are solved by black box computer codes that allow no intervention in their algorithms. This opens the possibility to orchestrate the actions of two or more computer codes to solve problems of greater complexity than is possible using the codes individually. Such methods can be used to link together existing best-in-class tools to tackle complex multiphysics and multiscale problems, without requiring extraordinary programming effort. In addition, such methods can be also be used within a single code, when it proves advantageous over solving the global set of equations in some integrated manner.

One type of application arises when physical problems in non-overlapping domains are coupled through interactions at their common boundary. A typical example is the fluid–structure interaction problem, in which the fluid and structure equations are solved separately from one another, subject to boundary conditions that represent self-consistent matching of forces and displacements at the fluid–structure interface [1–4]. Another example of this type is a conjugate heat transfer problem studied in our earlier work [5–8], in which a furnace radiation model is coupled to a melt crystal growth model

* Corresponding author.
E-mail address: yecke003@umn.edu (A. Yeckel).

0021-9991/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.08.003

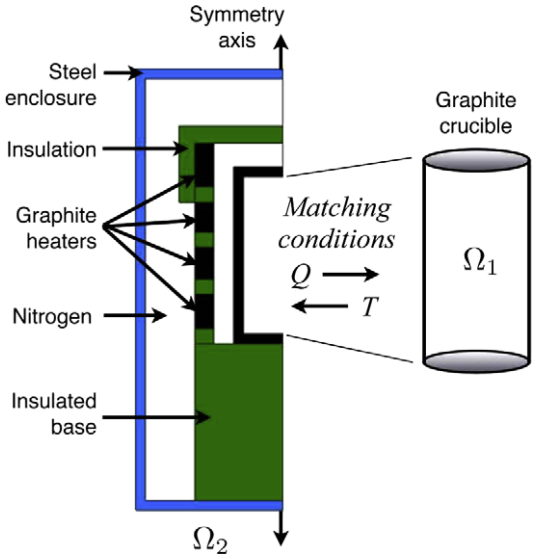

Fig. 1. Generic simplified schematic of a melt crystal growth furnace, depicting a crystal growth process divided into submodels representing growth chamber (Ω_1) and furnace (Ω_2), which are coupled via conjugate heat transfer conditions.

via temperature and flux matching conditions. The situation is illustrated in Fig. 1. This example plays a central role in the work presented here.

A different type of application is the segregated calculation of multiphysics within a common domain, illustrated by a transport-reaction problem in which species, temperature, and flow equations are each solved independently, but are coupled through reaction source terms and temperature-dependent properties [9,10]. This approach facilitates building a complete physical model from a library of submodels [11]. Although we do not treat examples of this type, the method used here is quite general and equally applicable to such problems.

Another completely different type of application arises when a set of equations representing a physical problem is subjected to constraint equations that are necessarily solved separately. In this way arclength and other constrained parameter methods, as well as constrained optimization, can be imposed in situations where it is not possible to modify the means of solving the physical problem to include the desired constraints. Chan [12], whose work plays an important role in this paper, developed an early and innovative way to tackle this type of application.

Other motivations exist. Artlich and Mackens [13] solve a transport-reaction problem of two variables on a common domain that are coupled through an integral term, which introduces dense blocks to an otherwise sparse global Jacobian. To avoid forming these dense blocks, they developed a segregated solver approach based on a problem splitting that preserves sparsity of the Jacobians of the subproblems. The effect of the integral term is isolated to an outer, fixed-point iteration.

The block Gauss–Seidel (BGS) method, with its back-and-forth character, is the most intuitive and conceptually simplest way to link two computer codes to solve a coupled problem, and is widely used for this reason. But the method performs poorly under many conditions, particularly the conjugate heat transfer examples studied in [7,8], so we seek an alternative. A superior family of methods has been developed that approximates a Newton iteration on the coupled system of equations using the solvers of the individual systems. Notable examples include the approximate Newton method of Chan [12], the iterative approximate Newton method of Artlich and Mackens [13], the tangential block Newton method of Menck [14], and the approximative block Newton method of Matthies and Steindorf [15]. These methods, which appear to originate with Chan's ANM paper [12], are quite similar to one another in their mechanics. All of them begin with a block Jacobi or Gauss–Seidel step, and so can be thought of as accelerators to the basic iteration. The implementations vary, however, with some more general than others.

We draw upon this family of methods to seek a general form, which we label simply the approximate block Newton (ABN) method. Below, we state concisely a practical implementation of the method that is completely general, provided that the information needed to effect the coupling can be extracted and exchanged between the solvers. The method bears strong similarities to the methods mentioned above, but we demonstrate some clear performance advantages of our ABN method.

In the sections to come, we provide an original derivation that sheds light on key relationships and differences among these closely related methods. In particular, we show that methods of this type behave like a Newton iteration preconditioned by an inexact Newton solver based on the subproblem Jacobian matrices. The preconditioner is incorporated directly into the formulation of the method, which yields a well-conditioned Schur complement-type problem that can easily be solved using an iterative linear solver without further preconditioning. We also discuss various ways of iterating on a transformed set of variables, rather than the primitive variables of the subproblems. This issue has been glossed over in previous works, yet is essential to understanding practical implementation of the method.

Our method is demonstrated on a problem in melt crystal growth that is partitioned into furnace and growth chamber regions that are modeled by independent heat transfer codes; these are linked by temperature and flux matching conditions at the boundaries common to the partitions. Whereas a typical block Gauss–Seidel iteration fails about half the time for this model problem, quadratic convergence can be achieved by the ABN method under all conditions studied here. Both one- and two-dimensional examples are considered. The results demonstrate that the ABN method is robust and efficient for problems of this type.

2. Coupled nonlinear models and fixed-point iterations

We first motivate the various solution strategies to be discussed below by defining the underlying problem of the coupling of two, nonlinear models. In this section and what follows, matrices are denoted by boldface upper-case Roman letters and vectors by boldface lower-case Roman letters. Other quantities are scalars unless noted otherwise. Subscripts in boldface indicate differentiation, e.g. $\mathbf{r}_\mathbf{x}$ is a matrix of derivatives of a vector $\mathbf{r}(\mathbf{x})$ with respect to $\mathbf{x}$. Function evaluations are indicated by parentheses $(\cdot)$, and operations such as multiplication and differentiation are denoted by square brackets, $[\cdot]$.

We seek the solution to the global, coupled problem

$$\mathbf{f}(\mathbf{x},\mathbf{y}) = \mathbf{0} \tag{1}$$

$$\mathbf{g}(\mathbf{x},\mathbf{y}) = \mathbf{0} \tag{2}$$

We assume that the subproblems $\mathbf{f}$ and $\mathbf{g}$ are segregated such that $\mathbf{x}$ is the solution to $\mathbf{f}$, with $\mathbf{y}$ as input parameters, and $\mathbf{y}$ is the solution to $\mathbf{g}$, with $\mathbf{x}$ as input parameters. It makes sense to rewrite the problem to indicate this situation:

$$\mathbf{f}(\mathbf{x},\tilde{\mathbf{y}}) = \mathbf{0} \tag{3}$$

$$\mathbf{g}(\tilde{\mathbf{x}},\mathbf{y}) = \mathbf{0} \tag{4}$$

Variables carrying the tilde represent the coupling of one subproblem to the other.

For now we assume that the dependence of one subproblem on the solution to the other subproblem is explicit, i.e. $\mathbf{x}$ and $\mathbf{y}$ can be substituted directly for $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$. In many cases, however, coupling of the problems is expressed implicitly, as in

$$\mathbf{f}(\mathbf{x},\tilde{\mathbf{y}}(\mathbf{y})) = \mathbf{0} \tag{5}$$

$$\mathbf{g}(\tilde{\mathbf{x}}(\mathbf{x}),\mathbf{y}) = \mathbf{0} \tag{6}$$

Each subproblem depends on a transformation of the other subproblem's variables. This introduces some complications, but since a function $\mathbf{f}(\mathbf{x},\tilde{\mathbf{y}}(\mathbf{y}))$ can always be written $\mathbf{f}(\mathbf{x},\mathbf{y})$, much of the mathematical development below is unaffected, so we defer discussion of transformed variables until Section 3.5.

We assume that black box solvers are available for each subproblem. We write the action of these solvers in the form of a fixed-point iteration:

$$\mathbf{x}^{(k+1)} = \mathcal{F}(\mathbf{x}^{(k)},\tilde{\mathbf{y}}) \tag{7}$$

$$\mathbf{y}^{(k+1)} = \mathcal{G}(\tilde{\mathbf{x}},\mathbf{y}^{(k)}) \tag{8}$$

The counter k is introduced with the expectation that these fixed-point solvers will be used within an iterative scheme to solve a coupled problem; these outer iterations are not to be confused with any iterations occurring within the solvers themselves. The solution updates given by these solvers can be written as

$$\Delta^{\mathcal{F}}\mathbf{x}(\tilde{\mathbf{y}}) = \mathcal{F}(\mathbf{x}^{(k)},\tilde{\mathbf{y}}) - \mathbf{x}^{(k)} \tag{9}$$

$$\Delta^{\mathcal{G}}\mathbf{y}(\tilde{\mathbf{x}}) = \mathcal{G}(\tilde{\mathbf{x}},\mathbf{y}^{(k)}) - \mathbf{y}^{(k)} \tag{10}$$

A subproblem is converged when its update falls below some suitably small tolerance. The global problem is converged when the solutions to Eqs. (3) and (4) are consistent, namely both subproblems are converged with $\tilde{\mathbf{x}} = \mathbf{x}$ and $\tilde{\mathbf{y}} = \mathbf{y}$.

A fixed-point form is chosen to describe the subproblem solvers because of its generality. Any iterative solver can be written in this form; specifically, for Newton iteration applied to Eqs. (3) and (4) we have

$$\mathcal{F}^N(\mathbf{x}^{(k)},\tilde{\mathbf{y}}) = \mathbf{x}^{(k)} - \mathbf{f}_\mathbf{x}^{-1}(\mathbf{x}^{(k)},\tilde{\mathbf{y}})\mathbf{f}(\mathbf{x}^{(k)},\tilde{\mathbf{y}}) \tag{11}$$

$$\mathcal{G}^N(\tilde{\mathbf{x}},\mathbf{y}^{(k)}) = \mathbf{y}^{(k)} - \mathbf{g}_\mathbf{y}^{-1}(\tilde{\mathbf{x}},\mathbf{y}^{(k)})\mathbf{g}(\tilde{\mathbf{x}},\mathbf{y}^{(k)}) \tag{12}$$

The essence of a fixed-point solver is very simple: the user provides some input parameters to the solver, which returns an updated estimate of the solution to its subproblem. How the solver does so is unimportant. A solver in this abstract sense can encompass the sum of many algorithmic steps—whatever might be hidden inside a computer code, for example. In many situations, access to the internal workings of a code is limited, leaving us unable to obtain useful data that reside there, hence our use of the term 'black box' solver. To broaden applicability, therefore, a coupling method should rely entirely on fixed-point forms of the solvers, without the need to know any details such as residuals or Jacobians.

We also note that the individual solvers could be iterated several times before updating $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$, but conceptually this is redundant, since we can simply define these inner iterations to constitute a single step of our abstract solver [12]. Then there

is no need to reference inner iterations, and k can be used to denote an outer iteration over which $\mathbf{x}^{(k)}$ and $\mathbf{y}^{(k)}$ are updated, rather than an inner iteration over the individual solvers at fixed $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$. It is in this sense we will use the iteration counter from now on.

3. Solution strategies for coupled models

As laid out in the prior section, we adopt a fixed-point iteration strategy to solve for the coupled, nonlinear problem of interest. We are then faced with an important decision on exactly what type of iteration should be implemented. In the rather long ensuing discussion on this matter, we present three different formulations, taking care to provide derivations and comment on implementation issues. We also provide a section that focuses on historical issues related to prior, similar implementations. We believe that this extended discussion is warranted. The seemingly simple idea of coupling codes is fraught with practical challenges, and the block Newton algorithms developed to address these challenges, though elegant, are not simple.

3.1. The block Gauss–Seidel (BGS) method

The simplest type of coupling iteration involves repeated application of the update formulas, Eqs. (9) and (10). Note that in these equations $\Delta^{\mathcal{F}}\mathbf{x}$ depends on $\tilde{\mathbf{y}}$ and $\Delta^{\mathcal{G}}\mathbf{y}$ on $\tilde{\mathbf{x}}$. We have written this dependence explicitly to emphasize that these values must somehow be chosen before evaluating the updates. If one step of each solver is taken with $\tilde{\mathbf{x}} = \mathbf{x}^{(k)}$ and $\tilde{\mathbf{y}} = \mathbf{y}^{(k)}$, one iteration of the well-known Jacobi method is obtained. For the iterations used here and in our prior efforts [5–8], a block form of the Gauss–Seidel method is obtained by setting $\tilde{\mathbf{y}} = \mathbf{y}^{(k)}$, solving for $\mathbf{x}^{(k+1)}$, then setting $\tilde{\mathbf{x}} = \mathbf{x}^{(k+1)}$ to solve for $\mathbf{y}^{(k+1)}$. Hence, the BGS method consists of sequentially solving each model, rewriting Eqs. (7) and (8) as,

$$\mathbf{x}^{(k+1)} = \mathcal{F}(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}) \tag{13}$$

$$\mathbf{y}^{(k+1)} = \mathcal{G}(\mathbf{x}^{(k+1)}, \mathbf{y}^{(k)}) \tag{14}$$

Note that there is still ambiguity within this algorithm, since a choice must be made about the order in which these equations are solved. This issue is examined in practical terms for several test problems in subsequent sections.

3.2. The approximate block Newton (ABN) method

Newton's method is often the method of choice for solving nonlinear problems because of the desirable properties of guaranteed convergence and quadratic rates of convergence for initial guesses sufficiently close to the solution. However, Newton's method requires the formulation of the Jacobian matrix, whose components may not be accessible in a black box solver, and its solution, which may be quite expensive. The approximate block Newton method (ABN) discussed in this section is formulated to address these challenges while providing a good approximation to a full Newton iteration for the entire, coupled problem.

3.2.1. Derivation

In the interest of clarifying the big picture without getting bogged down in details, we take a somewhat ad hoc approach here to derive the ABN method. A more thorough analysis will be presented in the following sections. We begin, then, with a special case of Eqs. (7) and (8):

$$\mathbf{x}^{(k+1)} = \mathcal{F}(\mathbf{x}^{(k)}, \tilde{\mathbf{y}}) = \mathcal{F}(\tilde{\mathbf{y}}) \tag{15}$$

$$\mathbf{y}^{(k+1)} = \mathcal{G}(\tilde{\mathbf{x}}, \mathbf{y}^{(k)}) = \mathcal{G}(\tilde{\mathbf{x}}) \tag{16}$$

The right hand sides of the equations are taken to depend only on the parameters input from the other subproblem, not on the internal variables iterated over by the subproblem solver. One interpretation is that $\mathcal{F}$ and $\mathcal{G}$ are exact solvers of $\mathbf{f}$ and $\mathbf{g}$ [13]. In principle, this result can always be attained by iterating each nonlinear subproblem to convergence before exchanging coupling information with the other subproblem.

The central strategy employed in previous derivations of block Newton-type methods [13–15] is to recast a fixed-point solver as a root-finding problem to be tackled by Newton iteration. The manner in which this strategy is employed varies greatly, however. In terms of the special case considered here, the root finding problem is given by setting $\tilde{\mathbf{x}} = \mathbf{x}^{(k+1)}$, $\tilde{\mathbf{y}} = \mathbf{y}^{(k)}$, and substituting the first equation into the second to obtain:

$$\mathbf{y}^{(k+1)} = \mathcal{G}(\mathcal{F}(\mathbf{y}^{(k)})) \tag{17}$$

(note that this is simply Gauss–Seidel iteration written in a compact form). Let us recast this equation into residual form as,

$$\mathbf{r} = \mathbf{y} - \mathcal{G}(\mathcal{F}(\mathbf{y})) = \mathbf{0} \tag{18}$$

Solving this equation via Newton's method computes an update $\Delta\mathbf{y}$.

The first challenge is to simply implement Newton's method on Eq. (18), since we do not have access to the inner workings of either model and thus do not have direct knowledge of a Jacobian matrix. Adopting a strategy similar to that employed by Artlich and Mackens [13], we choose to solve Eq. (18) using a Jacobian–free Newton–Krylov (JFNK) [16] approach. Thus, the Newton step is given by the solution to

$$\mathbf{S}\Delta\mathbf{y} = -\mathbf{r} \tag{19}$$

where

$$\mathbf{S} \equiv \mathbf{r}_\mathbf{y} = \mathbf{I} - \partial_\mathbf{y}[\mathcal{G}(\mathcal{F}(\mathbf{y}))] \tag{20}$$

Eq. (19), which we will see later is of the Schur complement-type, is solved using a Krylov subspace-based solver such as GMRES or BiCGSTAB [17]. Solvers of this type are advantageous for requiring only the product of **S** with a Krylov vector **w**, rather than **S** itself. This product can be approximated by finite differences:

$$\mathbf{S}\mathbf{w} = \mathbf{w} - \partial_\mathbf{y}[\mathcal{G}(\mathcal{F}(\mathbf{y}))]\mathbf{w} \approx \mathbf{w} - \frac{1}{\epsilon}[\mathcal{G}(\mathcal{F}(\mathbf{y} + \epsilon\mathbf{w})) - \mathcal{G}(\mathcal{F}(\mathbf{y}))] \tag{21}$$

where ϵ is a suitably small finite-differencing parameter.

Using the JFNK approach described above, we are able to solve for **y** via an approximate Newton method that needs only information provided by the individual solvers $\mathcal{F}$ and $\mathcal{G}$. In the course of solving for **y**, we also obtain a converged value of **x** determined from Eq. (15) by the evaluation $\mathcal{F}(\mathbf{y})$. Thus, all that is needed to solve for **x** and **y** by this method are Eqs. (18) and (21), which can be computed using the current iterates of the solutions and their fixed-point solvers, plus whatever machinery is needed to perturb and exchange the solutions between solvers. This simplicity is essential if the method is to be applied to black box computer codes.

Per the simplified scheme discussed at the start of this section, the full dependence of the solver evaluations $\mathcal{F}$ and $\mathcal{G}$ on **x** and **y** can be reintroduced by substituting Eqs. (7) and (8) for Eqs. (15) and (16), even though the steps in the current, ABN method were derived from the latter, simplified equations. The justification for this generalization will be provided in the ensuing sections.

The cost of implementing the ABN method is substantially larger than that of the BGS method. To complete a single block Newton step, solver $\mathcal{F}$ must be executed $2 + N_K$ times and solver $\mathcal{G}$ must be executed $1 + N_K$ times, where N_K is the number of Krylov vectors. The extra cost per iteration can be large, approximately N_K times the cost of one BGS iteration. However, there are several ways to mitigate the extra cost per iteration, and results presented below demonstrate that the ABN method dramatically accelerates convergence and is notably more robust than the BGS method for the conjugate heat transfer problems studied here.

3.2.2. Implementation

Based on the outcome of the simplified analysis above, we propose the following approach to solve for the general case of Eqs. (7) and (8). We refer to these procedures as the ABN method.

The first step is to define

$$\mathbf{x}^\mathcal{F} = \mathcal{F}(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}) \tag{22}$$

$$\mathbf{y}^\mathcal{G} = \mathcal{G}(\mathbf{x}^\mathcal{F}, \mathbf{y}^{(k)}) \tag{23}$$

using the current iterates of the solution unknowns $\mathbf{x}^{(k)}$ and $\mathbf{y}^{(k)}$ and the fixed-point solvers. Note that this is equivalent to an iteration using the block Gauss–Seidel approach. From this point on, the method can be viewed as a correction to the BGS steps $\mathbf{x}^\mathcal{F}$ and $\mathbf{y}^\mathcal{G}$, the intent of which is to accelerate convergence.

Next, following the idea of finding the root for the Gauss–Seidel form of the coupled problem, we define an appropriate residual and solve via Newton iterations. First, a residual is computed

$$\mathbf{r} = \mathbf{y}^{(k)} - \mathbf{y}^\mathcal{G} \tag{24}$$

for the Newton iteration:

$$\mathbf{S}\Delta\mathbf{y} = -\mathbf{r} \tag{25}$$

Eq. (25) is solved for $\Delta\mathbf{y}$, using any Krylov subspace-based method, with the product of **S** and Krylov vector **w** computed by the following steps:

$$\mathbf{p} = \mathcal{F}(\mathbf{x}^{(k)}, \mathbf{y}^{(k)} + \epsilon\mathbf{w}) \tag{26}$$

$$\mathbf{S}\mathbf{w} = \mathbf{w} - \frac{1}{\epsilon}[\mathcal{G}(\mathbf{p}, \mathbf{y}^{(k)}) - \mathbf{y}^\mathcal{G}] \tag{27}$$

where ϵ is a small finite-differencing parameter. Note that **S** corresponds to the Jacobian matrix defined by the residual of Eq. (24). Owing to the application of the JFNK method, defined by Eqs. (26) and (27) used with the Krylov solver, we need not know the specific form of **S**.

Once $\Delta\mathbf{y}$ has been computed, the block Newton updates are computed from:

$$\mathbf{y}^{(k+1)} = \mathbf{y}^{(k)} + \Delta\mathbf{y} \tag{28}$$

$$\mathbf{x}^{(k+1)} = \mathcal{F}(\mathbf{x}^{(k)}, \mathbf{y}^{(k+1)}) \tag{29}$$

and the ABN step is complete. The iteration is repeated by following the sequence of computations starting with Eqs. (22) and (23). Convergence of the global problem is attained when a chosen norm of either $\Delta\mathbf{y}$ or $\mathbf{r}$ falls below a specified tolerance.

3.3. An alternative approximate block Newton (ABN-J) method

The prior discussions, though based around a specialized problem, were meant to demonstrate the elegant ideas behind Newton-based methods for solving coupled models. Here we present a more formal derivation that leads to a slightly different approximate block Newton method. We call this the ABN-J method because it starts from a block Jacobi iteration, whereas the ABN method starts from a Gauss–Seidel iteration. Following the derivation of the ABN-J method, we formally show how the ABN method can be interpreted using the same theoretical framework.

3.3.1. Derivation

If we return to the coupled problem, stated as Eqs. (1) and (2), and consider a monolithic approach to its solution, we would write an exact Newton step as:

$$\begin{bmatrix} \mathbf{f}_\mathbf{x} & \mathbf{f}_\mathbf{y} \\ \mathbf{g}_\mathbf{x} & \mathbf{g}_\mathbf{y} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{y} \end{bmatrix} = -\begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix} \tag{30}$$

where $\Delta\mathbf{x} \equiv \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$ and $\Delta\mathbf{y} \equiv \mathbf{y}^{(k+1)} - \mathbf{y}^{(k)}$ are the solution update vectors and all residuals and Jacobians are evaluated at $(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})$. If $\mathbf{f}_\mathbf{y}$ and $\mathbf{g}_\mathbf{x}$ are replaced by zeros, a block Jacobi iteration is obtained with exact Newton iterations used on the subproblems. Capturing the effect of these off-diagonal blocks, which represent the sensitivity of one subproblem to the other, is critical to achieving desired Newton-like features, such as accelerating convergence and improving robustness. However, according to our assumption of not being able to access the internal workings of each solver, these blocks are unavailable and must be approximated using only the current iterates of the solutions and their fixed-point solver updates, given in Eqs. (7) and (8) (or equivalently Eqs. (9) and (10)).

For reasons that will soon become clear, we precondition the Newton step using the Jacobian inverses of the subproblems:

$$\begin{bmatrix} \mathbf{f}_\mathbf{x}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{g}_\mathbf{y}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{f}_\mathbf{x} & \mathbf{f}_\mathbf{y} \\ \mathbf{g}_\mathbf{x} & \mathbf{g}_\mathbf{y} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{y} \end{bmatrix} = -\begin{bmatrix} \mathbf{f}_\mathbf{x}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{g}_\mathbf{y}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix} \tag{31}$$

This preconditioner can be viewed as an inexact Newton step that transforms the matrix closer to a diagonal form.

Multiplication followed by block elimination gives:

$$\begin{bmatrix} \mathbf{I} & \mathbf{C} \\ \mathbf{0} & \mathbf{S} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{q} \\ -\mathbf{r} \end{bmatrix} \tag{32}$$

with the following definitions: the Schur complement is given by

$$\mathbf{S} \equiv \mathbf{I} - \mathbf{g}_\mathbf{y}^{-1}\mathbf{g}_\mathbf{x}\mathbf{C} \tag{33}$$

and its right hand side by

$$\mathbf{r} \equiv \mathbf{s} + \mathbf{g}_\mathbf{y}^{-1}\mathbf{g}_\mathbf{x}\mathbf{q} \tag{34}$$

The remaining block matrix and vectors are given as

$$\mathbf{C} \equiv \mathbf{f}_\mathbf{x}^{-1}\mathbf{f}_\mathbf{y}, \quad \mathbf{q} \equiv -\mathbf{f}_\mathbf{x}^{-1}\mathbf{f}, \quad \mathbf{s} \equiv \mathbf{g}_\mathbf{y}^{-1}\mathbf{g} \tag{35}$$

where $\mathbf{q}$ and $\mathbf{s}$ are equivalent to block Jacobi updates for the subproblems $\mathbf{f}$ and $\mathbf{g}$, and the matrix $\mathbf{C}$ represents a coupling of the subproblems.

We obtain the solution updates via block back-substitution, yielding:

$$\Delta\mathbf{y} = -\mathbf{S}^{-1}\mathbf{r} \tag{36}$$

$$\Delta\mathbf{x} = \mathbf{q} - \mathbf{C}\Delta\mathbf{y} \tag{37}$$

As in the prior simplified case, solving just the first of these, Eq. (36), is the key to computing the Newton step. Once $\Delta\mathbf{y}$ is known, it is always possible to compute $\Delta\mathbf{x}$ directly, by using the subproblem solver $\mathcal{F}$ to compute $\mathbf{x}^{(k+1)}$ with the updated solution $\mathbf{y}$ as input. So, in practice, it is not necessary to solve Eq. (37) directly (although it can be done that way, if $\mathbf{C}$ is available, as in Ref. [12]).

We thus desire to solve the Schur complement problem, Eq. (36), using the same JFNK approach espoused before, but we encounter a problem. For the more general case considered here, both **S** and **r** comprise the products of many unknown terms. We now employ an approach similar to that used by Artlich and Mackens [13] to derive their IANM method (for more on the IANM, please refer to Section 3.6.1). Namely, a Taylor expansion of **g** with respect to **x** is used to apply finite differences to approximate the term $\mathbf{g_x C w}$ (cf. Eq. (33)), which arises from the multiplication of **S** by the vector **w** in forming the Krylov subspace. We obtain

$$\mathbf{S w} = \left(\mathbf{I} - \mathbf{g_y}^{-1}\mathbf{g_x C}\right)\mathbf{w} \approx \mathbf{w} + \frac{1}{\epsilon}\mathbf{g_y}^{-1}\left(\mathbf{x},\mathbf{y}\right)\left[\mathbf{g}\left(\mathbf{x} - \epsilon\mathbf{C w},\mathbf{y}\right) - \mathbf{g}\left(\mathbf{x},\mathbf{y}\right)\right] \tag{38}$$

In a similar fashion, the Taylor expansion $\mathbf{g_x q} \approx \mathbf{g}(\mathbf{x}+\mathbf{q},\mathbf{y}) - \mathbf{g}(\mathbf{x},\mathbf{y})$ is applied to Eq. (34) to give us **r** of Eq. (36)

$$\mathbf{r} \approx \mathbf{g_y}^{-1}(\mathbf{x},\mathbf{y})\mathbf{g}(\mathbf{x}+\mathbf{q},\mathbf{y}) \tag{39}$$

after terms cancel. Here no finite-differencing parameter is used (equivalent to setting $\epsilon = 1$) because **x** is perturbed by **q**, which is expected to be quadratically small.

Furthermore, the matrix–vector product **Cw** needed to compute $\mathbf{g}(\mathbf{x} - \epsilon\mathbf{C w},\mathbf{y})$ in Eq. (38) can be similarly approximated by applying finite differences to $\mathbf{f_y w}$ (cf. Eq. (35))

$$\mathbf{C w} = \mathbf{f_x}^{-1}\mathbf{f_y w} \approx \frac{1}{\epsilon}\mathbf{f_x}^{-1}(\mathbf{x},\mathbf{y})[\mathbf{f}(\mathbf{x},\mathbf{y}+\epsilon\mathbf{w}) - \mathbf{f}(\mathbf{x},\mathbf{y})] \tag{40}$$

The three preceding equations are all that is needed to solve the Schur complement problem using a JFNK approach. In keeping with our general goal, we seek to evaluate these equations using only the general fixed-point solvers $\mathcal{F}$ and $\mathcal{G}$. Clearly, this is not possible as written above. However, the genius of their current form (from insight attributed to Artlich and Mackens [13]) is that all products of multiple blocks of the global Jacobian in the original equations (cf. Eqs. (33)–(35)) have been significantly simplified using a single block matrix multiplying finite-difference approximations involving only solver evaluations. Significantly, these single blocks simply involve the Jacobian matrices of each subproblem. Indeed, $\mathbf{g_y}$ in Eqs. (38) and (39) and $\mathbf{f_x}$ in Eq. (40) are the Jacobian matrices associated with Newton solves of each individual model, as represented in Eqs. (11) and (12). Accordingly, we will use insight from the these fixed-point Newton formulas to replace $\mathbf{g_y}$ and $\mathbf{f_x}$ with suitable approximations.

We first approach the approximation to **Cw** represented by Eq. (40). We see by rearranging Eq. (11) that

$$\mathbf{f_x}^{-1}(\mathbf{x},\mathbf{y})\mathbf{f}(\mathbf{x},\mathbf{y}) = \mathbf{x} - \mathcal{F}^N(\mathbf{x},\mathbf{y}) \tag{41}$$

so the second term on the right hand side is readily computed using only an evaluation of the solver. The first term is more problematic, however. Consider the fixed-point form of the exact Newton solver evaluated at $(\mathbf{x},\mathbf{y}+\epsilon\mathbf{w})$:

$$\mathcal{F}^N(\mathbf{x},\mathbf{y}+\epsilon\mathbf{w}) = \mathbf{x} - \mathbf{f_x}^{-1}(\mathbf{x},\mathbf{y}+\epsilon\mathbf{w})\mathbf{f}(\mathbf{x},\mathbf{y}+\epsilon\mathbf{w}) \tag{42}$$

Note in the above equation, the Jacobian inverse $\mathbf{f_x}^{-1}$ is evaluated at $(\mathbf{x},\mathbf{y}+\epsilon\mathbf{w})$ rather than $(\mathbf{x},\mathbf{y})$. Rearranged, this equation yields the relationship,

$$\mathbf{f_x}^{-1}(\mathbf{x},\mathbf{y}+\epsilon\mathbf{w})\mathbf{f}(\mathbf{x},\mathbf{y}+\epsilon\mathbf{w}) = \mathbf{x} - \mathcal{F}^N(\mathbf{x},\mathbf{y}+\epsilon\mathbf{w}) \tag{43}$$

Now, if we claim that

$$\mathbf{f_x}^{-1}(\mathbf{x},\mathbf{y})\mathbf{f}(\mathbf{x},\mathbf{y}+\epsilon\mathbf{w}) \approx \mathbf{f_x}^{-1}(\mathbf{x},\mathbf{y}+\epsilon\mathbf{w})\mathbf{f}(\mathbf{x},\mathbf{y}+\epsilon\mathbf{w}) \tag{44}$$

which is tantamount to substituting a secant step for an exact Newton step, we can employ the above series of relationships to rewrite Eq. (40) using the much simpler expression

$$\mathbf{C w} \approx -\frac{1}{\epsilon}[\mathcal{F}^N(\mathbf{x},\mathbf{y}+\epsilon\mathbf{w}) - \mathcal{F}^N(\mathbf{x},\mathbf{y})] \tag{45}$$

that does comply with our stated goal of using only solver evaluations to compute needed quantities.

The discrepancy introduced by the use of the secant approximation, Eq. (44), between these two approximations for **Cw** is given by subtracting the original Eq. (40) from the simpler Eq. (45), which, to leading order in ϵ, is $\approx \mathbf{w}[\mathbf{f_x}^{-1}]_{\mathbf{y}}[\mathbf{f} + \mathbf{f_y}\epsilon\mathbf{w}]$. In essence, this error is introduced by evaluating the Jacobian needed in Eq. (40) at the "wrong" location. The effect of this error is assessed by considering the size of both terms as the solution is approached. Since the residuals **f** vanish quadratically at convergence, the first error term, $\mathbf{w}[\mathbf{f_x}^{-1}]_{\mathbf{y}}\mathbf{f}$, is quadratically small about the sought-after solution. The second term, $\epsilon\mathbf{w}^2[\mathbf{f_x}^{-1}]_{\mathbf{y}}\mathbf{f_y}$, is first-order but can be made small by ϵ. Convergence at quadratic rates can often be achieved with careful choice of ϵ, however.

The derivation proceeds similarly for **S** and **r**. Substituting $\mathcal{G}^N$ from Eq. (12) for the exact Newton steps in Eqs. (38) and (39) and making a similar secant approximation yields

$$\mathbf{Sw} \approx \mathbf{w} - \frac{1}{\epsilon}\left[\mathcal{G}^N(\mathbf{x} - \epsilon\mathbf{Cw}, \mathbf{y}) - \mathcal{G}^N(\mathbf{x}, \mathbf{y})\right] \tag{46}$$

$$\mathbf{r} \approx \mathbf{y} - \mathcal{G}^N(\mathbf{x} + \mathbf{q}, \mathbf{y}) \tag{47}$$

where $\mathbf{q} = \mathcal{F}^N(\mathbf{x}, \mathbf{y}) - \mathbf{x}$. The above two equations again achieve our goal of using only solver evaluations in these approximations.

The same error considerations discussed with regard to Eq. (45) apply to Eqs. (46) and (47), except that secant error in Eq. (47) is proportional to $\mathbf{q}$ rather than ϵ. Although $\mathbf{q}$ vanishes quadratically near the solution, this error cannot be arbitrarily reduced by control of ϵ. The significance of this error will be explored further in the following section.

In the derivations above, we have used the exact Newton forms of the model solvers, given by $\mathcal{F}^N$ and $\mathcal{G}^N$ in Eqs. (11) and (12), to yield simplified approximations. This approach is perfectly suitable, if, indeed, the solvers of the individual models are Newton methods. However, we desire to generalize this framework for any fixed-point solvers. Conceptually, we do this by simply using $\mathcal{F}$ and $\mathcal{G}$ (rather than $\mathcal{F}^N$ and $\mathcal{G}^N$) in the above formulas, irrespective of their form. An important corollary of this action is that the original solution updates, given by Eqs. (9) and (10), must also be assumed to be represented by their Newton-based analogs, namely that $\Delta^{\mathcal{F}}\mathbf{x}$ is equivalent to $\mathbf{q}$ and $\Delta^{\mathcal{G}}\mathbf{y}$ is equivalent to $\mathbf{s}$. These approximations should not degrade convergence provided these solvers compute updates that are quadratically convergent for their respective subproblems. As Chan [12] has noted, any contractive fixed-point solver can attain this requirement by repeated iteration (we speak of the solver's internal iteration here). With this stipulation, we can write the method solely in terms of the current solution iterate, $(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})$ and general fixed-point solvers $\mathcal{F}$ and $\mathcal{G}$.

In terms of computational effort, the fixed-point solvers $\mathcal{F}$ and $\mathcal{G}$ must each be executed a total of $2 + N_K$ times per block Newton step of the ABN-J method, the same as the ABN method for $\mathcal{F}$, but one greater for $\mathcal{G}$. The additional evaluation is the Jacobi-type step $\mathcal{G}(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})$ needed to evaluate Eq. (46). The equivalent term in the ABN method is provided by the Gauss–Seidel step already computed in Eq. (23), giving the ABN method a slight performance advantage.

Eqs. (45)–(47) form the foundation of the approach that we term the ABN-J method. The basis of this name refers to the starting point for deriving this formulation, which is a Jacobi step rather than the BGS step that started the ABN method. We provide additional insight between the ABN and ABN-J methods in Section 3.4.

3.3.2. Implementation

The ABN-J method is carried as follows. With current iterates of the solution unknowns $\mathbf{x}^{(k)}$ and $\mathbf{y}^{(k)}$ and the fixed-point solvers, we compute

$$\mathbf{q} = \mathbf{x}^{(k)} - \mathcal{F}(\mathbf{x}^{(k)}, \mathbf{y}^{(k)}) \tag{48}$$

which is used to compute a residual

$$\mathbf{r} = \mathbf{y}^{(k)} - \mathcal{G}(\mathbf{x}^{(k)} + \mathbf{q}, \mathbf{y}^{(k)}) \tag{49}$$

for the Newton iteration:

$$\mathbf{S}\Delta\mathbf{y} = -\mathbf{r} \tag{50}$$

Eq. (50) is then solved for $\Delta\mathbf{y}$, using a Krylov subspace-based method. The product of the Schur complement $\mathbf{S}$ and Krylov vector $\mathbf{w}$ is computed by the following relations:

$$\mathbf{Cw} = -\frac{1}{\epsilon}\left[\mathcal{F}(\mathbf{x}^{(k)}, \mathbf{y}^{(k)} + \epsilon\mathbf{w}) - \mathcal{F}(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})\right] \tag{51}$$

$$\mathbf{Sw} = \mathbf{w} - \frac{1}{\epsilon}\left[\mathcal{G}(\mathbf{x}^{(k)} - \epsilon\mathbf{Cw}, \mathbf{y}^{(k)}) - \mathcal{G}(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})\right] \tag{52}$$

where ϵ is a small finite-differencing parameter.

With $\Delta\mathbf{y}$ computed, we compute updates as:

$$\mathbf{y}^{(k+1)} = \mathbf{y}^{(k)} + \Delta\mathbf{y} \tag{53}$$

$$\mathbf{x}^{(k+1)} = \mathcal{F}(\mathbf{x}^{(k)}, \mathbf{y}^{(k+1)}) \tag{54}$$

completing an iteration by the ABN-J method. Subsequently, iterations are carried out until convergence.

3.4. The ABN and ABN-J methods compared

The ABN and ABN-J methods differ only in the approximation employed to compute the product of the Schur complement and Krylov vector $\mathbf{Sw}$ in Eq. (27) for ABN and Eq. (52) for ABN-J. To allow for a direct comparison of these expressions, we represent each below after some substitution and rearrangement of terms:

$$[\mathbf{Sw}]_{\text{ABN}} = \mathbf{w} - \frac{1}{\epsilon}\left[\mathcal{G}(\mathbf{p}, \mathbf{y}^{(k)}) - \mathcal{G}(\mathbf{x}^{\mathcal{F}}, \mathbf{y}^{(k)})\right] \tag{55}$$

$$[\mathbf{Sw}]_{\text{ABN-J}} = \mathbf{w} - \frac{1}{\epsilon}\left[\mathcal{G}(\mathbf{p} + \mathbf{x}^{(k)} - \mathbf{x}^{\mathcal{F}}, \mathbf{y}^{(k)}) - \mathcal{G}(\mathbf{x}^{\mathcal{F}} + \mathbf{x}^{(k)} - \mathbf{x}^{\mathcal{F}}, \mathbf{y}^{(k)})\right] \tag{56}$$

where $\mathbf{p} = \mathcal{F}(\mathbf{x}^{(k)}, \mathbf{y}^{(k)} + \epsilon\mathbf{w})$ and $\mathbf{x}^{\mathcal{F}} = \mathcal{F}(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})$ as defined originally in the ABN algorithm.

From these expressions, we see that the only difference is the location at which the function $\mathcal{G}$ is evaluated. Noting that $\mathbf{q} = \mathbf{x}^{\mathcal{F}} - \mathbf{x}^{(k)}$, we see that the ABN-J expression, Eq. (56), is evaluated at an $\mathbf{x}$-value that is shifted by a factor of $-\mathbf{q}$ from that employed by the ABN. This shift is another error that arises from the secant approximations that were made in the derivation of the ABN-J method; see, e.g. Eq. (44). Nevertheless, this discrepancy vanishes quadratically, and the methods are equivalent to one another at convergence.

A final observation comparing these approaches is compelling. Considering the rather ad hoc manner in which the ABN method was derived in Section 3.2, it may be surprising that the formally-derived ABN-J method is nearly identical in form (except for the difference noted above). To explain this fact, we return to the starting point of the ABN derivations, Eqs. (15) and (16), where each subproblem is assumed to depend only on the variables of the other problem. This assumption may not be as restrictive as it might initially appear. For example, if $\mathcal{F}$ and $\mathcal{G}$ are affected only to second-order by $\mathbf{x}^{(k)}$ and $\mathbf{y}^{(k)}$, respectively, our starting assumption is perfectly valid, since only first-order interactions are subsequently considered in the ensuing derivation. Notably, this will be the case when each solver is based on a Newton solver, an assumption that is used to derive the approximations employed in the ABN-J method.

3.5. Transformed variables

Each subproblem consists of some internal variables, $\mathbf{x}$ or $\mathbf{y}$. These primitive variables do not necessarily correspond to the input parameters $\tilde{\mathbf{x}}$ or $\tilde{\mathbf{y}}$ required by the other subproblem. Often the parameters to one subproblem will be derived from the other subproblem by a transformation, i.e. $\tilde{\mathbf{x}}(\mathbf{x})$ or $\tilde{\mathbf{y}}(\mathbf{y})$. Eqs. (7) and (8) can be rewritten in terms of these transformed variables:

$$\mathbf{x}^{(k+1)} = \mathcal{F}(\mathbf{x}^{(k)}, \tilde{\mathbf{y}}(\mathbf{y}^{(k)})) \tag{57}$$

$$\mathbf{y}^{(k+1)} = \mathcal{G}(\tilde{\mathbf{x}}(\mathbf{x}^{(k)}), \mathbf{y}^{(k)}) \tag{58}$$

In many situations, such as dividing a multiphysics simulation on a common domain among different black box solvers, an interpolation between grids of the respective subproblems is the only transformation that is needed. In other situations, a transformation changes the physical meaning of a variable, e.g. one subproblem computes the temperature, but the other subproblem requires its heat flux instead. Since $\mathcal{F}(\mathbf{x}, \tilde{\mathbf{y}}(\mathbf{y}))$ can always be denoted $\mathcal{F}(\mathbf{x}, \mathbf{y})$, these transformations do not affect the validity of the ABN equations presented above. The presence of such transformations can have an impact on how best to implement the method, however, particularly if working with black box computer codes.

Under some scenarios $\mathbf{x}$ or $\mathbf{y}$ will be unavailable, making it impossible to perturb the internal unknowns to compute transformations such as $\tilde{\mathbf{y}}(\mathbf{y} + \epsilon\mathbf{w})$ in Eq. (26). We can make the following approximation on the basis that the perturbation is small:

$$\tilde{\mathbf{y}}(\mathbf{y} + \epsilon\mathbf{w}) \approx \tilde{\mathbf{y}}(\mathbf{y}) + \tilde{\mathbf{y}}(\epsilon\mathbf{w}) \tag{59}$$

We must know how to compute $\tilde{\mathbf{y}}(\epsilon\mathbf{w})$ to use this approach, but in many cases the transformation $\tilde{\mathbf{y}}$ will reside within the black box along with $\mathbf{y}$, making this approach impractical. We can always perturb the transformed variable itself, however, so a better approach is to switch over to $\tilde{\mathbf{x}}$, $\tilde{\mathbf{y}}$ as the variables in the outer iteration, namely

$$\mathbf{f}(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) = \mathbf{0} \tag{60}$$

$$\mathbf{g}(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) = \mathbf{0} \tag{61}$$

The method is written exactly as before, with the substitutions

$$\tilde{\mathbf{x}}^{(k)} \to \mathbf{x}^{(k)} \tag{62}$$

$$\tilde{\mathbf{y}}^{(k)} \to \mathbf{y}^{(k)} \tag{63}$$

The only practical implication for implementation of the method is that $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$ are perturbed instead of $\mathbf{x}$ or $\mathbf{y}$. Any underlying differences are hidden within the abstract solvers, so the method should remain quadratically convergent, barring a pathological variable transformation.

When the subproblem domains intersect only along a shared boundary, the data shared between subproblems is typically related to matching conditions, such as continuity of temperature and heat flux, that are one order lower in spatial dimension than the problem from which these conditions are derived. As an example, if $\mathbf{x}$ and $\mathbf{y}$ are the solutions to a two-dimensional heat transfer problem, $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$ will be one-dimensional boundary data for temperature and flux. In this case using a transformed variable in place of its primitive counterpart reduces the size of the Schur matrix $\mathbf{S}$ by at least one order of magnitude. In terms of the method, boundary data shared this way fully characterize the subproblem from which they are derived. Thinking then of the boundary data as a perfect representation of the global data on a much smaller subspace

consisting only of the most relevant vector directions, it seems likely that the number of Krylov vectors needed to adequately represent **S** will also be reduced. Since the cost of an ABN iteration is approximately proportional to N_K, the potential savings is large. This argues in favor of basing the implementation on the transformed variables, whenever doing so substantially reduces the size of **S**.

3.6. Historical perspectives

There have been several key historical developments for block approximations to Newton's method. In this section, we elucidate the ideas behind these methods and point out their differences. The pragmatic reader may choose to skip this section and continue to the example problems; however, we believe that such background is important to fully appreciate the theoretical underpinnings of the methods developed here.

3.6.1. The methods of Chan and of Artlich and Mackens

Chan [12] and Artlich and Mackens [13] both apply block elimination directly to the Newton step in Eq. (30), without the preconditioning step in Eq. (31). The Schur matrix and its right hand side are defined differently than in Eqs. (33) and (34), leading to these forms:

$$\mathbf{S}_{\mathrm{ANM}} \equiv \mathbf{g}_{\mathbf{y}} - \mathbf{g}_{\mathbf{x}}\mathbf{C} \tag{64}$$

$$\mathbf{r}_{\mathrm{ANM}} \equiv \mathbf{g} + \mathbf{g}_{\mathbf{x}}\mathbf{q} \tag{65}$$

where **C** and **q** are defined as previously in Eq. (35). These forms lack premultiplication by $\mathbf{g}_{\mathbf{y}}^{-1}$ introduced by the preconditioning step, without which there is difficulty introducing the fixed-point solver form of Eq. (12) to the right hand side of these equations, as was done in deriving Eqs. (46) and (47) above.

Chan was interested primarily in arclength continuation methods, for which subproblem **g** consists of a few constraint equations applied to control the parameters of subproblem **f**. In this situation, **g** and its Jacobian blocks $\mathbf{g}_{\mathbf{x}}$ and $\mathbf{g}_{\mathbf{y}}$ can be computed directly, but knowledge of **f** is restricted to its fixed-point solver. Then, to compute **S** and **r** in Eqs. (64) and (65), it is only necessary to find **q** and **C**, after which Eq. (36) is solved directly to determine $\Delta\mathbf{y}$. Chan introduces a fixed-point solver to solve for **q**, just as above, and derives an approximation to the columns of **C** identical to Eq. (51) with **w** replaced by $\mathbf{e}_j$. Chan's approach to deriving this equation differs significantly from the approach taken here, however, a point we return to below.

Artlich and Mackens sought to extend Chan's method to systems in which both subproblems are large, at least thousands of variables, for which it is infeasible to calculate every column of **C**. Introducing a Krylov projection-based solver mitigates this problem by reducing it to computing **Cw** for N_K Krylov vectors in place of every column of **C**.

Artlich and Mackens calculated **q** the same way that Chan did and might have adapted Chan's result to calculate **Cw** as well. Instead, they chose Eq. (40) as their starting point. They note that Eq. (40) is an exact Newton step at position $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ to solve $\mathbf{f}(\tilde{\mathbf{x}}, \tilde{\mathbf{y}} + \epsilon\mathbf{w}) - \mathbf{f}(\mathbf{x}, \tilde{\mathbf{y}}) = \mathbf{0}$ for **x**, where extra care has been taken to denote by the tilde which variables remain fixed during this step. This problem is equivalent to an iteration on $\mathbf{f}(\mathbf{x}, \tilde{\mathbf{y}})$ with $\mathbf{f}(\tilde{\mathbf{x}}, \tilde{\mathbf{y}} + \epsilon\mathbf{w})$ held fixed, which leads to:

$$\mathbf{Cw} \approx \frac{1}{\epsilon}[\mathbf{f}_{\mathbf{x}}^{-1}(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})\mathbf{f}(\tilde{\mathbf{x}}, \tilde{\mathbf{y}} + \epsilon\mathbf{w}) - \mathbf{x} + \mathcal{F}(\mathbf{x}, \tilde{\mathbf{y}})] \tag{66}$$

where Eq. (11) has been used to replace the second term on the right hand side of Eq. (40), the same as we did earlier in deriving Eq. (51).

At this point they introduce the following restrictive relationship:

$$\mathbf{f} = \mathbf{x} - \mathcal{F}(\mathbf{x}, \mathbf{y}) \tag{67}$$

which states that the residuals of subproblem **f** are equal to the negative of the Jacobi update of solver $\mathcal{F}$. They assert this relationship on the basis that both sides of the equation have the same solution, i.e. if $\mathbf{f}(\mathbf{x}, \mathbf{y}) = \mathbf{0}$, then $\mathbf{x} - \mathcal{F}(\mathbf{x}, \mathbf{y}) = \mathbf{0}$ also. But this relationship is correct only when $\mathbf{f}_{\mathbf{x}}^{-1} = \mathbf{I}$ (cf. Eq. (11)), an approximation that will be wrong to first-order except under very special circumstances. Invoking this relationship makes the Jacobian inverse simply disappear from Eq. (66), with the result that

$$\mathbf{Cw} \approx -\frac{1}{\epsilon}[\mathbf{f}(\tilde{\mathbf{x}}, \tilde{\mathbf{y}} + \epsilon\mathbf{w}) + \mathbf{x} - \mathcal{F}(\mathbf{x}, \tilde{\mathbf{y}})] \tag{68}$$

A similar argument is invoked to swap each instance of the residuals**g** with its fixed-point solver $\mathcal{G}$ by the relationship

$$\mathbf{g} = \mathbf{y} - \mathcal{G}(\mathbf{x}, \mathbf{y}) \tag{69}$$

to derive expressions for **Sw** and **r**. These expressions are not necessarily wrong, but are arrived at by an ad hoc argument that lacks generality.

Eq. (68) and other steps in the algorithm require evaluation of the residuals, restricting it to applications that allow access to these values. For this reason the method was not considered in our tests below.

3.6.2. The method of Matthies and Steindorf

Matthies and Steindorf [15] avoided the pitfalls encountered by Artlich and Mackens by adopting an indirect approach to solving Eq. (30). Their method is to approximate a Newton step of a system based on the residuals of the fixed-point solvers themselves, $\Delta^{\mathcal{F}}\mathbf{x}(\mathbf{y}) = \mathbf{0}$, $\Delta^{\mathcal{G}}\mathbf{y}(\mathbf{x}) = \mathbf{0}$ (Eqs. (9) and (10)), which they note have the same solutions as $\mathbf{f}$ and $\mathbf{g}$. This Newton step is written:

$$\begin{bmatrix} [\mathbf{x}-\mathcal{F}]_{\mathbf{x}} & -\mathcal{F}_{\mathbf{y}} \\ -\mathcal{G}_{\mathbf{x}} & [\mathbf{y}-\mathcal{G}]_{\mathbf{y}} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{y} \end{bmatrix} = -\begin{bmatrix} \mathbf{x}-\mathcal{F} \\ \mathbf{y}-\mathcal{G} \end{bmatrix} \tag{70}$$

The resemblance of the lower component of the right hand side to Eq. (18), the starting point of our derivation of the ABN method in Section 3.2, should not be overlooked. Also notable is that Chan's derivation of $\mathbf{Ce}_j$, alluded to above, is based on the same operations appearing in the upper blocks of Eq. (70), even though his starting point was Eq. (30). These observations emphasize the ever-present role of the fixed-point solver forms in this family of methods.

Matthies and Steindorf factorize this system using block elimination, and, by straightforward finite differences, obtain Eq. (51) for $\mathbf{Cw}$, Eq. (49) for $\mathbf{r}$, and an expression for $\mathbf{Sw}$ that differs slightly from Eq. (52), namely:

$$\mathbf{Sw} \approx \mathbf{w} - \frac{1}{\epsilon}\left[\mathcal{G}(\mathbf{x}-\epsilon\mathbf{Cw}, \mathbf{y}+\epsilon\mathbf{w}) - \mathcal{G}(\mathbf{x},\mathbf{y})\right] \tag{71}$$

The Jacobian blocks are never formed, but are purely conceptual. When a solver of the fixed-point forms is needed, the solver of the original equations is used in its place, e.g. the solver for $\mathbf{f} = \mathbf{0}$ is used to solve the problem $\mathbf{x} - \mathcal{F} = \mathbf{0}$ and so on.

From these results Matthies and Steindorf outline an algorithm similar to the ABN-J algorithm in Section 3.3, except that the $\mathbf{y}$ unknowns in Eq. (71) are perturbed by $\epsilon\mathbf{w}$, something not done in Eq. (52). A Taylor expansion in $\mathbf{y}$ applied to $\mathcal{G}(\mathbf{x}-\epsilon\mathbf{Cw}, \mathbf{y}+\epsilon\mathbf{w})$ about the point $(\mathbf{x},\mathbf{y})$ shows that this perturbation introduces a term proportional to $[\mathbf{g}_{\mathbf{y}}^{-1}]_{\mathbf{y}}\mathbf{g}$ that vanishes quadratically. Not only can this term can be neglected, somewhat simplifying the method, its inclusion can lead to inferior performance of the method in certain cases, as we show in the examples below.

To better understand the relationship between our derivation and that of Matthies and Steindorf, we insert the exact Newton solvers of Eqs. (11) and (12) into the Jacobian of Eq. (70) and apply the chain-rule to obtain:

$$\begin{bmatrix} [\mathbf{f}_{\mathbf{x}}^{-1}\mathbf{f}]_{\mathbf{x}} & [\mathbf{f}_{\mathbf{x}}^{-1}\mathbf{f}]_{\mathbf{y}} \\ [\mathbf{g}_{\mathbf{y}}^{-1}\mathbf{g}]_{\mathbf{x}} & [\mathbf{g}_{\mathbf{y}}^{-1}\mathbf{g}]_{\mathbf{y}} \end{bmatrix} = \begin{bmatrix} [\mathbf{I}-[\mathbf{f}_{\mathbf{x}}^{-1}]_{\mathbf{x}}\mathbf{f}] & [\mathbf{f}_{\mathbf{x}}^{-1}\mathbf{f}_{\mathbf{y}} + [\mathbf{f}_{\mathbf{x}}^{-1}]_{\mathbf{y}}\mathbf{f}] \\ [\mathbf{g}_{\mathbf{y}}^{-1}\mathbf{g}_{\mathbf{x}} + [\mathbf{g}_{\mathbf{y}}^{-1}]_{\mathbf{x}}\mathbf{g}] & [\mathbf{I}-[\mathbf{g}_{\mathbf{y}}^{-1}]_{\mathbf{y}}\mathbf{g}] \end{bmatrix} \tag{72}$$

Terms with second derivatives, such as $[\mathbf{f}_{\mathbf{x}}^{-1}]_{\mathbf{x}}\mathbf{f}$, vanish quadratically. Dropping these terms produces Eq. (31), the starting point of our derivation in Section 3.3. Matthies and Steindorf do not formally neglect these terms in deriving their method, but achieve an equivalent result by using solvers based on $\mathbf{f}$ and $\mathbf{g}$ to compute the solution to the fixed-point forms in Eq. (70). The extra perturbation of $\mathbf{y}$ in Eq. (71) persists as an artifact of this approach, however.

Matthies and Steindorf [15] note that the fixed-point equations are numerically better conditioned, because "they implicitly already have the effect of the single system solvers in them." Eq. (31) explains this observation in terms of an explicit preconditioning by exact Newton solvers on the subproblems. This inexact Newton preconditioning leads to a strongly diagonal Schur matrix that can be well-represented by a projection onto a few vector directions. Hence, we expect that a Krylov subspace-based method will require relatively few Krylov vectors $\mathbf{w}$ to converge for our formulations (via the solution of Eqs. (25) and (50)), a conclusion that is supported by example results presented below.

4. One-dimensional heat transfer example

We begin our evaluation of the ABN method by using it to solve a steady-state, one-dimensional heat transfer problem intended to mimic lateral heat transport at the crucible wall in a melt crystal growth system. Pandy and co-workers [5,7] found that block Gauss–Seidel iteration performed poorly for this problem, often failing to converge at all. Derby et al. [8] showed that under-relaxed BGS iterations were equally ineffective applied to this same problem. We show here that ABN iteration dramatically outperforms BGS iteration, converging in all cases at a small fraction of the computational effort. This example is also used to illustrate how the ABN method can be applied to transformed variables when the effect of one subproblem on the other subproblem is reduced to its boundary data, in this case temperature and heat flux.

4.1. Model equations

The global model is partitioned into two subdomains: Ω_1 $(0 \leqslant x \leqslant 1)$ is a simple one-dimensional analog of a crystal melt and Ω_2 $(1 \leqslant x \leqslant 2)$ is a similar analog of a crystal growth furnace. The model is described in detail elsewhere [5,7]; we simply repeat the non-dimensionalized equations developed there. Heat transfer occurs by conduction and convection in Ω_1 and by conduction in Ω_2:

$$\frac{d^2T}{dx^2} - \mathrm{Pe}\frac{dT}{dx} = 0 \quad \text{in } \Omega_1 \tag{73}$$

$$\kappa\frac{d^2T}{dx^2} = 0 \quad \text{in } \Omega_2 \tag{74}$$

where T is temperature, x is length, κ is thermal conductivity ratio, and Pe is Peclet number. These subdomains and their model solutions will constitute the subproblems of the ABN iteration, of course.

Dirichlet boundary conditions are specified on the external ends of the system, $x = 0$ and $x = 2$:

$$T = T_L \quad \text{at } x = 0 \tag{75}$$

$$T = T_R \quad \text{at } x = 2 \tag{76}$$

It is also necessary to furnish a boundary condition to each subproblem at the boundary separating Ω_1 and Ω_2. Applying temperatures, for example

$$T = T_1 \quad \text{at } x = 1 \text{ in } \Omega_1 \tag{77}$$

$$T = T_2 \quad \text{at } x = 1 \text{ in } \Omega_2 \tag{78}$$

yields the solutions [7]:

$$T = \frac{(T_1 - T_L e^{\mathrm{Pe}}) + (T_L - T_1)e^{\mathrm{Pe}x}}{1 - e^{\mathrm{Pe}}} \quad \text{in } \Omega_1 \tag{79}$$

$$T = (T_R - T_2)(x - 2) + T_R \quad \text{in } \Omega_2 \tag{80}$$

Radiation effects are incorporated in the model by treating the subdomain Ω_2 as transparent to radiation, with the boundary at $x = 1$ exchanging radiation with the boundary at $x = 2$. With this provision, fluxes at the shared boundary are given by inserting the solutions into the following expressions:

$$Q_1 = -\left.\frac{dT}{dx}\right|_{x=1-} \tag{81}$$

$$Q_2 = -\kappa\left.\frac{dT}{dx}\right|_{x=1+} + \mathrm{Rd}\left(T_2^4 - T_R^4\right) \tag{82}$$

where Rd is radiation number.

Up to this point, the subproblems have been solved independently, assuming that T_1 (or T_2) is given at $x = 1$. To obtain a globally consistent solution we must construct an iteration that matches temperature and flux at the boundary separating Ω_1 and Ω_2:

$$T_1 = T_2 \tag{83}$$

$$Q_1 = Q_2 \tag{84}$$

We desire to solve the problem in a segregated manner, with Eqs. (79) and (80) serving as solvers to the subproblems. In the next section we describe a procedure to do this using the ABN method of Section 3.2.

4.2. Iteration procedures

We assume that the solutions in each domain can be used to compute a heat flux at the boundary given a temperature there, or vice-versa, but are otherwise inaccessible. This sort of black box scenario is typical, particularly when using commercial codes. For example, we might use Eq. (81) to compute the flux from the temperature in Ω_1, which we denote $Q_1(T_1)$. The same equation can be rearranged to compute $T_1(Q_1)$ instead. Similarly in Ω_2 we can compute either $Q_2(T_2)$ or $T_2(Q_2)$ using Eq. (82).

The boundary temperature and flux are transformed variables in the sense described in Section 3.5. In a practical scenario, these variables are likely to be computed using heat transfer codes based on discretized solution of transport equations. The internal solutions maintained by these codes are unlikely to be available, so the practical approach is to construct the ABN iteration around the transformed variables. This is done using the matching conditions in Eqs. (83) and (84), which at convergence must be satisfied. The unknowns of the global iteration are designated Q_m and T_m, which must satisfy the relationships

$$Q_m = Q_1 = Q_2 \tag{85}$$

$$T_m = T_1 = T_2 \tag{86}$$

The ABN method begins with a BGS step, which can be arranged several ways. Using Q_1 and T_2 as fixed-point solvers, for example, the iteration can be written:

$$Q_m^{(k+1)} = Q_1(T_m^{(k)}) \tag{87}$$

$$T_m^{(k+1)} = T_2\left(Q_m^{(k+1)}\right) \tag{88}$$

These are just Eqs. (22) and (23) with $\mathbf{x} \equiv Q_m$, $\mathbf{y} \equiv T_m$. These same solvers might be applied in reverse order, as well, with $\mathbf{x} \equiv T_m$, $\mathbf{y} \equiv Q_m$:

$$T_m^{(k+1)} = T_2(Q_m^{(k)}) \tag{89}$$

$$Q_m^{(k+1)} = Q_1\left(T_m^{(k+1)}\right) \tag{90}$$

We refer to these formulations as coupling configurations and denote them Q_1–T_2 and T_2–Q_1, respectively. By swapping the roles of the domains, i.e. using Q_2 and T_1 as solvers, two other coupling configurations are obtained, denoted T_1–Q_2 and Q_2–T_1.

After the BGS step is completed by any of these coupling configurations, the matrix $\mathbf{S}$ is computed using Eqs. (26) and (27). Since $\mathbf{S}$ is simply a scalar in this example, these equations are used with $\mathbf{w} = 1$, and a Krylov-based solver is unnecessary. Once $\mathbf{S}$ is known, the solution updates are computed by Eqs. (28) and (29).

The coupling configurations used here have some differences that warrant commenting on before proceeding to the results. One difference is whether the first step of the method requires T_m or Q_m as its initial guess; this is determined by the order in which the solvers are evaluated in the method. Another difference is whether the solvers are exact. The solution in Ω_1 is linear, making it easy to form exact solvers for both $Q_1(T_1)$ and $T_1(Q_1)$. In contrast, the flux–temperature relationship in Ω_2 is nonlinear due to radiation heat transfer. Although Eq. (82) is an exact solver for $Q_2(T_2)$, it cannot be factorized easily to obtain an exact solver for $T_2(Q_2)$. Instead we use local Newton iteration to compute $T_2(Q_2)$ when needed. This treatment adds realism to the example and allows studying the effects of subproblem iteration on convergence of the global iteration. The implications of the initial guess and differences in the behavior of the subproblem solvers will be explored in the next section.

4.3. Results and discussion

BGS and ABN algorithms were tested to study their convergence behavior for each of the four coupling configurations just described. Parameter values representative of a melt crystal growth system were used in the model equations: $\mathrm{Pe} = 9$, $\kappa = 0.1$, $T_L = 0.98$, and $T_R = 1$. Iterations were deemed converged when the update satisfied the criterion $|\Delta\mathbf{y}| < 10^{-6}$. A finite-differencing parameter $\epsilon = 10^{-4}$ was used in the ABN algorithm for the results shown; the effect of varying this parameter in the range 10^{-3}–10^{-7} proved insignificant.

Fig. 2(a) shows the effect of radiation heat transfer on the number of iterations required by the BGS algorithm to converge. Values ranging from $\mathrm{Rd} = 0$ (linear) to $\mathrm{Rd} = 10$ (highly nonlinear) are considered. Coupling configurations based on T_1 and Q_2 converge only at low values of Rd, regardless of order in which the solvers are evaluated. Conversely, configurations based

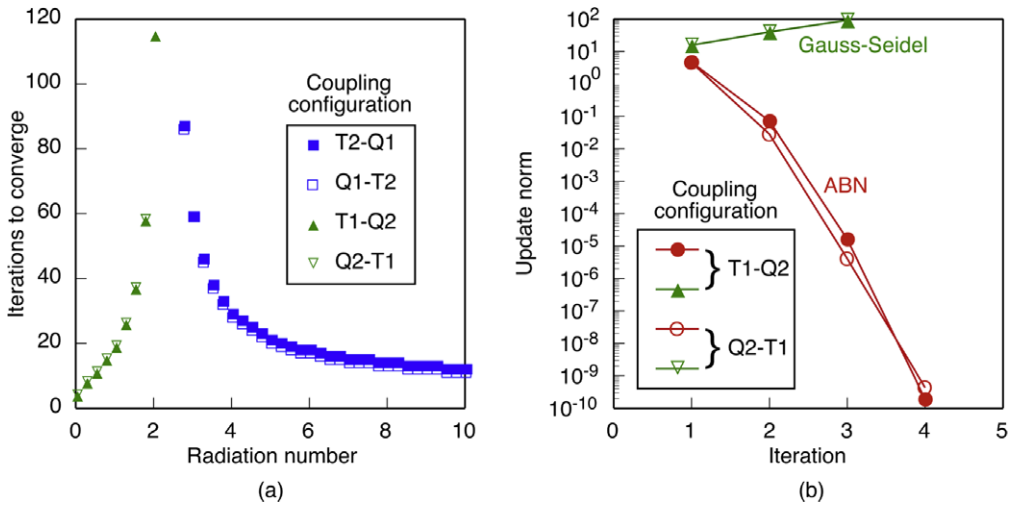

Fig. 2. (a) Effect of radiation number Rd on convergence behavior of BGS iteration, for four different coupling configurations of the one-dimensional model. T_2–Q_1 defines $X \equiv T_2$ and $Y \equiv Q_1$ and Q_1–T_2 defines $X \equiv Q_1$ and $Y \equiv T_2$; in both cases T_2 is computed from the model in domain 2, and the flux Q_1 is computed from the model in domain 1. Conversely, T_1–Q_2 and Q_2–T_1 use T_1 from domain 1 and Q_2 from domain 2, differing only by which variable is assigned to X in the algorithm. (b) Comparison of ABN iteration to BGS iteration for the T_1–Q_2 and Q_2–T_1 configurations, starting with initial guess $T_m = 1$, at radiation number $\mathrm{Rd} = 5.67$.

on T_2 and Q_1 converge only at high values of Rd. Convergence is very slow at intermediate values of Rd regardless of the configuration used. We conclude that BGS iteration is largely a failure for this problem.

In contrast, the ABN method converges in 3–4 iterations for all values of the radiation number, equivalent in effort to 8–10 BGS iterations for this problem (based on five solver evaluations per ABN iteration versus two per BGS iteration). A typical convergence history is shown in Fig. 2(b), for Rd = 5.67, using the configurations based on T_1 and Q_2, both of which are exact solvers. Note that all BGS iterations diverge for this case, while the ABN method converges quadratically. Order of solver evaluation (i.e. T_1—Q_2 versus Q_2—T_1) is unimportant here.

Order of solver evaluation does matter for the T_2—Q_1 and Q_1—T_2 configurations, however, as shown in Fig. 3. Here the T_2 solver consists of a local Newton iteration applied to solve Eq. (82). The T_2—Q_1 configuration (Fig. 3(a)) is sensitive to the number of these subiterations, but the Q_1—T_2 configuration (Fig. 3(b)) is not. In the former case, the iteration is seeded with an initial guess $Q_m = 0$, whereas in the latter the iteration starts from $T_m = 1$ (these are most plausible initial guesses for each formulation, in our opinion). Interestingly, for the T_2—Q_1 ABN case shown in Fig. 3(a), the algorithm begins with Q_m as input to the nonlinear solver T_2, and convergence is slightly compromised for the initial guess made here (see the iteration history marked by square symbols). This effect vanishes if we use our foreknowledge of the solution to select a better initial guess for Q_m, but it also vanishes if the T_2 solver is subiterated more than once, at which point it becomes nearly an exact solver (see the iteration history marked by circles). As shown by the convergence history for the ABN method in Fig. 3(b) for the Q_1—T_2 configuration, the starting value of T_m is unimportant, because Q_1 is an exact solver, the output of which is independent of initial guess. In all cases, the BGS method converges, though relatively slowly.

Two lessons are learned from the ABN method results shown in Fig. 3. One is that subiteration of a subproblem solver can accelerate convergence in some situations, albeit at a computational price (though it can be harmful in other situations, as shown below). The other is that order of evaluation of the solvers, though it should not affect the asymptotic convergence properties of the algorithm, can be important for strategic reasons related to initial guesses for the subproblem solvers. Some judgement and experimentation may be necessary to determine the most effective coupling configuration in a given situation, though any of them may be quadratically convergent. The issue of convergence rate outside the quadratic zone is an important one that will be taken up in the next test cases.

5. One-dimensional heat transfer example revisited

The conjugate heat transfer problem of Section 4 is revisited here in a slightly different form to compare and contrast the convergence behavior of the various block Newton variants discussed above. All perform well compared to BGS iteration, but some differences are observed, particularly with regard to behavior outside the asymptotic zone of quadratic convergence.

5.1. Model equations

The model equations are the same as outlined in Section 4.1, except that the matching conditions of Eqs. (83) and (84) are replaced by an equivalent form

$$\alpha Q_1 = \alpha Q_2 + (1-\alpha)(T_2 - T_1) \tag{91}$$

$$\beta Q_2 = \beta Q_1 + (1-\beta)(T_1 - T_2) \tag{92}$$

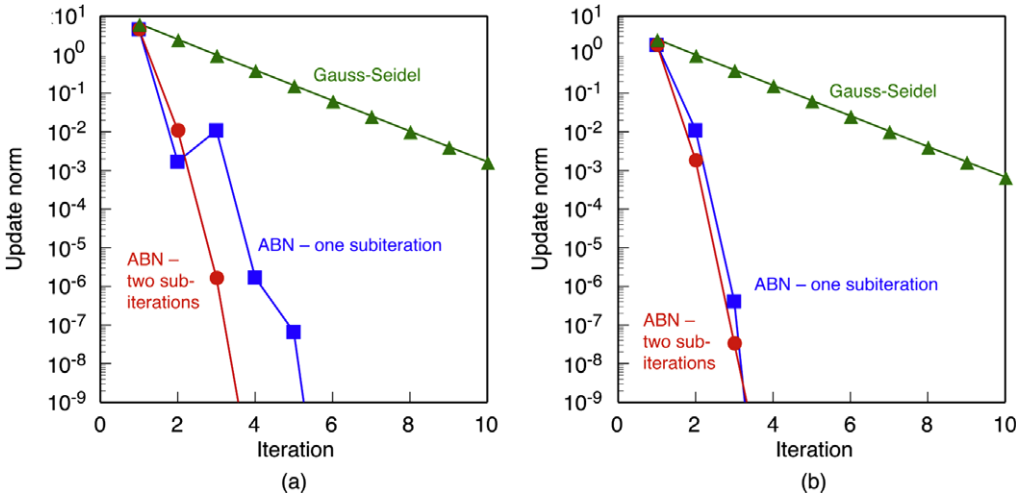

Fig. 3. Comparison of ABN iteration to BGS iteration for (a) the T_2—Q_1 configuration and (b) the Q_1—T_2 configuration, starting with initial guess $T_m = 1$, at radiation number Rd = 5.67.

where α and β are arbitrary parameters that allow tuning of the fixed-point iterations. These were termed coupling parameters and were successfully employed to coax convergence from a BGS implementation of a coupled crystal growth model in [6]. Subsequent analyses [7,8] demonstrated, however, that this approach alone was insufficient to overcome the poor convergence properties of the BGS method, hence motivating the work presented here. Note that the case $\alpha = \beta$ is singular and to be avoided. The cases $\alpha,\ \beta = (0,1)$ and $\alpha,\ \beta = (1,0)$ recover Eqs. (83) and (84).

The examples studied in the previous section suggest that the ABN method is affected only slightly by the coupling configuration chosen; we desire to test this observation more thoroughly, however, so we return to this formulation with the notion that every $\alpha,\ \beta$ pair constitutes a different coupling configuration.

5.2. Iteration procedures

To more thoroughly test the coupling algorithms, we arrange this formulation into an iteration in which the fixed-point solvers intentionally do not satisfy the special form of fixed-point solvers in Eqs. (15) and (16). To do this we substitute the solutions given by Eqs. (79) and (80), together with the flux definitions in Eqs. (81) and (82), into Eqs. (91) and (92). The result is rearranged into the following residuals:

$$r_1 = T_1 - a - bT_1 - cT_2 + \mathrm{Rd}T_2^4 \tag{93}$$

$$r_2 = T_2 - a - dT_1 - eT_2 + \mathrm{Rd}T_2^4 \tag{94}$$

where

$$a = fT_L + \kappa T_R + \mathrm{Rd}T_R^4, \quad b = -\frac{1-\alpha}{\alpha} - f, \quad c = \frac{1-\alpha}{\alpha} - \kappa$$

$$d = -\frac{1-\beta}{\beta} - f, \quad e = \frac{1-\beta}{\beta} - \kappa, \quad f = -\frac{\mathrm{Pe}\, e^{\mathrm{Pe}}}{1 - e^{\mathrm{Pe}}}$$

These nonlinear equations are solved locally by Newton iteration: Eq. (93) is iterated for T_1, with T_2 as input, and Eq. (94) is iterated for T_2, with T_1 as input. The local Newton steps of these iterations represent the subproblem solvers for the various coupling algorithms tested.

5.3. Results and discussion

All results were computed with Rd = 5.67, and all other parameters the same as in Section 4.3. Varying the finite-differencing parameter ϵ in the range 10^{-3}–10^{-7} again had no significant effect on the outcome.

Fig. 4 shows the effect of $\alpha,\ \beta$ on the convergence of the BGS and ABN iterations. These parameters were each varied from 0 to 1 by increments of 0.01, resulting in a total sample of 10,201 pairs. Shaded regions represent $\alpha,\ \beta$ pairs for which iteration fails to converge within 1000 iterations (nearly all of which are divergent). Approximately half of all BGS iterations never converge, and those that do require 105 iterations on average. The results reinforce the lesson that BGS iteration performs poorly for this type of problem. In contrast, all cases of ABN iteration converge within four iterations, except for the $\alpha = \beta$ cases (the shaded diagonal in the figure), which, as noted above, are ill-posed.

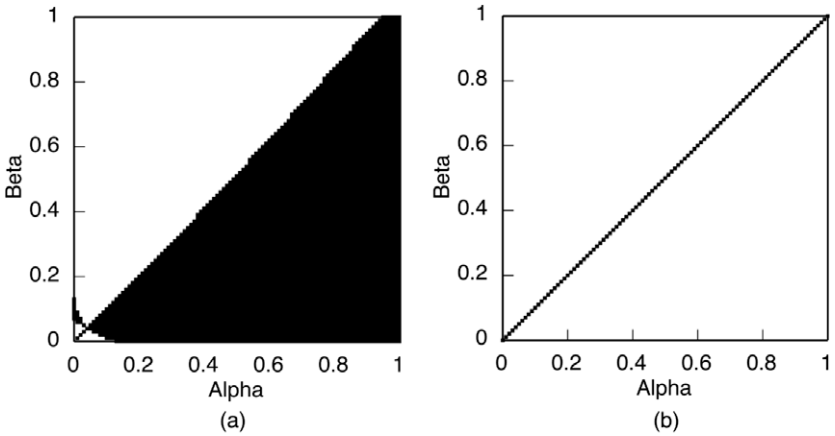

Fig. 4. Effect of temperature- and flux-matching parameters $\alpha,\ \beta$ on convergence behavior of (a) BGS iteration and (b) ABN iteration (ABN-J iteration produces an identical result). Shaded regions indicate cases that fail to converge within 1000 iterations, starting with initial guess $T_m = 1$, at radiation number Rd = 5.67.

The performance of the method of Matthies and Steindorf (MS) [15] is examined for various cases in Fig. 5 for the same problem considered previously. Fig. 5(a) shows the case when a good initial guess, $T_m = 1$, is employed and the total number of iterations is limited to 10. Most cases converge within five iterations, but a small number of stubborn cases converge more slowly. Approximately 1% of cases require more than 10 iterations, up to 70 iterations in the worst case. Approximately 0.5% of the cases converged to a second root of the problem, a behavior that was not observed for the BGS or ABN algorithms.

Fig. 5(b) and (c) show what happens to the MS method if a rather poor initial guess is used to start the iteration, in this case $T_m = 0$, as compared to $T_m = 1$ used in the case shown in Fig. 5(a). A mere 11% of cases converge within 10 iterations; see Fig. 5(b). The success rate improves when up to 50 iterations are allowed, as shown in Fig. 5(c); however, a significant number of cases, approximately 12%, remain unconverged. After 1000 iterations, the MS iteration converges to the first root in 61% of the cases and to the second root in 30% of the cases. The remaining 9% are those cases that diverge using this algorithm. In contrast, the ABN and ABN-J methods converge to the first root within 8 or fewer iterations, starting from the same initial guess of $T_m = 0$, in all cases with $\alpha \neq \beta$.

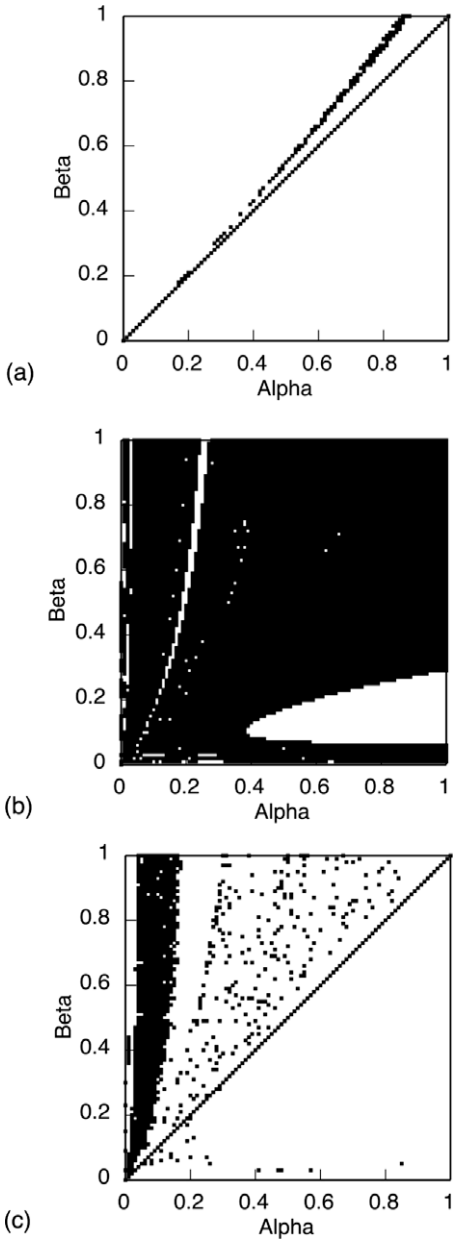

Fig. 5. Effect of temperature- and flux-matching parameters α, β on convergence behavior of the Matthies–Steindorf (MS) algorithm, at radiation number Rd = 5.67. Shaded regions indicate cases that fail to converge: (a) within 10 iterations, starting with initial guess $T_m = 1$; starting with initial guess $T_m = 0$, (b) within 10 iterations and (c) within 50 iterations.

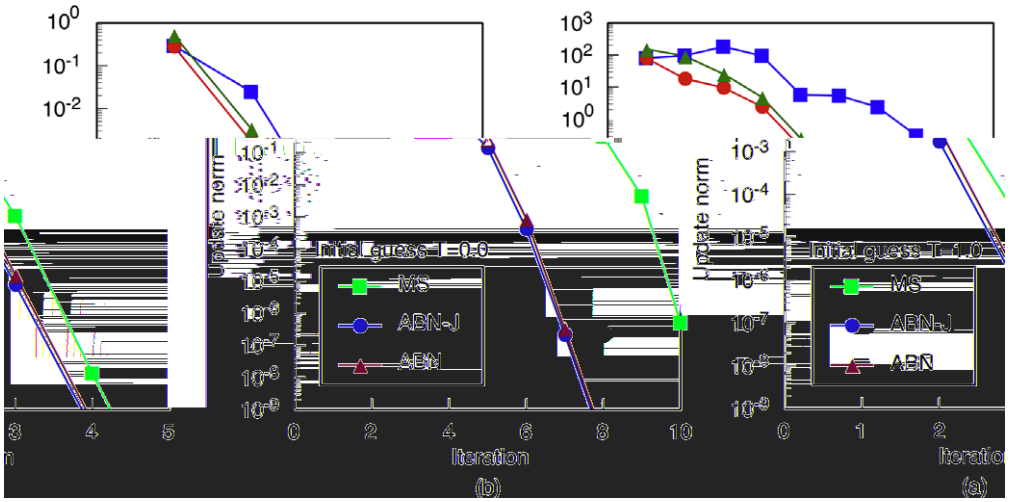

Fig. 6. Update norm versus iteration, for MS, ABN-J, and ABN iterations, at $\alpha = 0.8,\ \beta = 0.2,\ \mathrm{Rd} = 5.67$, starting with initial guess (a) $T_m = 1$ and (b) $T_m = 0$.

Fig. 6 sheds some light on the different behaviors exhibited by each of the block Newton-type methods by plotting the update norm as a function of iteration for the specific case of $\alpha,\ \beta = (0.8, 0.2)$ and for the good and poor initial guesses. Fig. 6(a) shows that all the methods converge quadratically from the second iteration onward when starting with a good initial guess $T_m = 1$. Fig. 6(b) shows that a poor initial guess affects the methods differently, however. Under this condition, the ABN and ABN-J methods outperform the MS method, with an update norm that diminishes quickly and monotonically, reaching the quadratic regime by the fifth iteration. The update norm for the MS method wanders, rising and falling before reaching the quadratic regime after nine iterations.

Fig. 7 attempts to quantify the issue of convergence success by plotting the fraction of all $\alpha,\ \beta$ pairs converged as a function of the maximum iterations allowed for the different methods tested here. Fig. 7(a) summarizes the results of Figs. 4 and 5, showing that the ABN and ABN-J methods perform extremely well, converging within eight iterations in all cases, whereas convergence of the MS method is slow for most $\alpha,\ \beta$ pairs and is unsuccessful in a significant number of cases, even after very many iterations.

Fig. 7(b) shows the results of iteration schemes that subiterate twice per global iteration. It was shown earlier, in regard to Fig. 3, that this strategy sometimes reduces the number of global iterations. Here the effects are largely deleterious. The MS method converges slightly more often, whereas the ABN and ABN-J methods are made significantly worse. Some $\alpha,\ \beta$ pairs no longer converge, and a moderate number of other pairs now converge to the second root (9% for the ABN method and 12% for the ABN-J method, as compared to 35% for the MS method). Increasing the number of subiterations per global iteration degrades performance further, in terms of total cases converged, for all the methods.

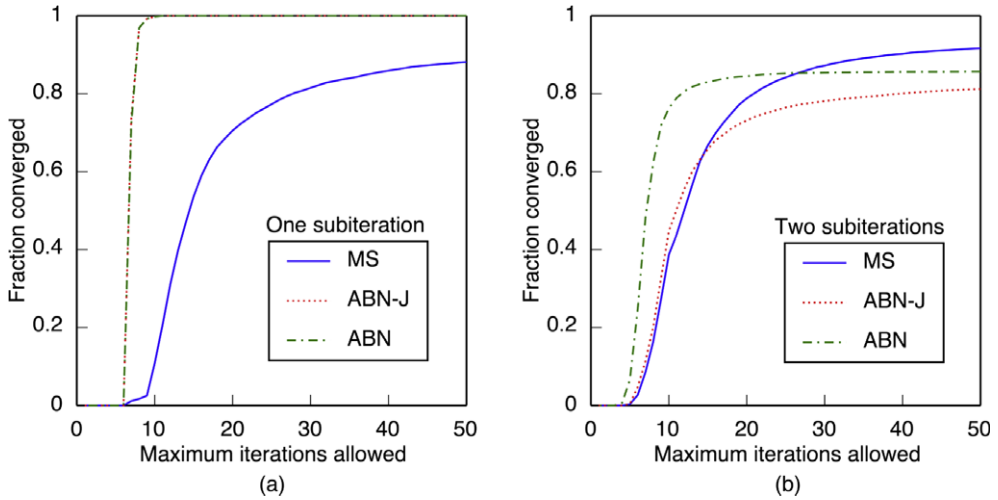

Fig. 7. Fraction of cases converged versus maximum iterations allowed to reach convergence, for 10,201 pair $\alpha,\ \beta$ sample at $\mathrm{Rd} = 5.67$, starting with initial guess $T_m = 0$, using (a) one and (b) two subiterations of the nonlinear subproblem in domain 2.

The iteration behavior for individual $\alpha,\ \beta$ pairs is not as simple as Fig. 7(b) might suggest, however. Whereas increasing the number of subiterations causes many new $\alpha,\ \beta$ pairs to diverge, some pairs are helped by additional subiteration. The $\alpha,\ \beta = (0.18, 0.26)$ case, for example, converges if three or more subiterations are executed per global iteration, whereas it diverges if only two subiterations are executed. At issue here is whether the nonlinear subproblem solver is contractive or not at the initial guess given. Tests performed using up to 30 subiterations per global iteration showed that the nonlinear subproblem is locally contractive for an initial guess $T_m = 0$ for only a slight majority of $\alpha,\ \beta$ pairs. The global iteration converges for 98% of these pairs, provided the subproblem is subiterated all the way to convergence. Subiterating just a few times is not adequate to achieve local convergence, however, which can cause the global iteration to fail in some cases.

In contrast, those cases for which the nonlinear subproblem is non-contractive are always harmed by additional subiteration. What is perhaps most interesting is that all of these cases do converge by the ABN and ABN-J methods if subiterated only once. This outcome is consistent with Chan's [12] convergence analysis of his ANM method, in which he shows that the global iteration can converge despite a non-contractive subproblem solver.

6. Two-dimensional convective heat transfer example

While the one-dimensional problems of the prior sections were useful to consider many issues involving problem formulation and iteration performance, they are admittedly very simple. In this final results section, we consider a more realistic two-dimensional representation of heat transfer in melt crystal growth systems. Fig. 1 shows a simplified schematic depicting a crystal growth process divided into subproblems representing furnace and growth chamber, which are coupled via conjugate heat transfer conditions. A highly detailed model of such a system was presented in our earlier work on solver coupling [6,7]. Here we study a simplified system consisting of a cylinder within a cylinder, deferring analysis of the full system to a future paper. Nevertheless, the example studied here is sufficiently rich in detail to provide a substantial test of the ABN method.

6.1. Model equations

The model of the growth chamber (domain Ω_1) represents heat and momentum transport in a high-temperature liquid. Convective–diffusion and incompressible Navier–Stokes equations for this problem can be written [18]:

$$\nabla^2 T - \mathbf{u}\cdot\nabla T = 0 \tag{95}$$

$$\frac{1}{\mathrm{Pr}}\mathbf{u}\cdot\nabla\mathbf{u} + \nabla p - \nabla\cdot(\nabla\mathbf{u} + (\nabla\mathbf{u})^T) - \mathrm{Ra}(1-T)\mathbf{g} = 0 \tag{96}$$

$$\nabla\cdot\mathbf{u} = 0 \tag{97}$$

where T is temperature, p is pressure, and $\mathbf{u}$ is velocity, all dimensionless. The Prandtl number Pr is set to unity, characteristic of many II–VI semiconductor melts. The magnitude of the Rayleigh number, Ra, scales the intensity of buoyant thermal convection. The equations are solved in 2D axisymmetric coordinates $r,\ z$. Except for the symmetry axis, the boundary is shared with the other domain and is meant to represent the chamber wall. A condition of zero flow velocity is imposed there, and heat transfer conditions are supplied by the matching process described in detail below.

The model of the furnace (domain Ω_2) is greatly simplified, consisting essentially of a high-temperature heat bath in which heat transfer is by conduction only:

$$\kappa\nabla^2 T = 0 \tag{98}$$

where κ is thermal conductivity ratio. A destabilizing thermal gradient is imposed by applying a hot temperature at the bottom surface, a low temperature at the top surface, and zero heat flux on the side surface of the domain. The dimensionless temperature difference is unity; the actual magnitude is embodied in the Rayleigh number.

6.2. Iteration procedures

Two separate computer codes are employed to compute the subproblems. The growth chamber model is solved using Cats2D [19], our finite element-based multiphysics transport code, on a structured mesh of nine-noded quadrilateral elements. The heat bath model is solved using CrysVUn [20–23], a commercial finite volume-based heat transfer analysis code, on an unstructured mesh of triangles. Both codes use Newton iteration in conjunction with a linear equation solver to compute their respective solutions.

The ABN algorithm is implemented inside Cats2D, to which CrysVUn is linked as a compiled library. Access to CrysVUn is limited to computing heat flux at the boundary in response to changing its Dirichlet temperature boundary condition, making it essentially a black box of the $Q_2(T_1)$ type (in the sense of Section 4.2). Variables exchanged between the codes are interpolated using a third-order piecewise fit of discrete data along the shared boundary. With the ABN approach implemented here, the Schur problem is solved using GMRES [24] without preconditioning.

As in the previous example, the fixed-point solvers of the coupled iteration are based on Eqs. (91) and (92). The temperature computed by Cats2D at the shared boundary is imposed as a Dirichlet temperature condition on the CrysVUn heat bath

calculation, equivalent to Eq. (91) with $\alpha = 0$. The boundary heat flux returned by CrysVUn is used by Cats2D in Eq. (92), with $\beta = 0.9$. The equation is divided by β and imposed on Cats2D as a weak boundary condition in the finite element sense.

Our purpose in choosing $\beta = 0.9$ rather than $\beta = 1$ is to avoid specifying a pure flux boundary condition everywhere at the boundary. To do so would leave the Cats2D problem underspecified: with T appearing only in the form of its derivatives in the governing equations, the solution can be shifted by an arbitrary constant and remain a valid solution. By setting $\beta \neq 1$, the Cats2D problem is well specified, and a consistent temperature level is enforced between the subproblems.

Per the issues discussed in Section 3.5, rather than view the problem as one of matching Q and T at the boundary, it is more natural to consider the transformed variables, $\tilde{x} \equiv \alpha Q + (1-\alpha)T$ and $\tilde{y} \equiv \beta Q + (1-\beta)T$, and view the problem as one of matching $\tilde{x}$ and $\tilde{y}$ at the boundary. The case considered here, $\alpha = 0$, $\beta \neq 1$, reduces to $\tilde{x} = T_1$ and $\tilde{y} = \beta Q_2 + (1-\beta)T_2$. Cats2D generates $\tilde{\mathbf{x}} = [\tilde{x}_i]$ as its output, in the form of T_1 evaluated at some points i along the shared boundary in Ω_1. Similarly, CrysVUn implicitly generates $\tilde{\mathbf{y}} = [\tilde{y}_j]$ as its output, computed from T_2 and Q_2 evaluated at some points j along the shared boundary in Ω_2. These transformed variables form the basis of the ABN iteration.

Though CrysVUn is a solver of type $Q_2(T_1)$, in order to compute $\tilde{\mathbf{y}}$ it must also be viewed to return T_2, at least in an abstract sense. Of course this is simply the value of the Dirichlet condition T_1 that was imposed on CrysVUn to begin with, but the distinction between T_1 and T_2 is not trivial. Because $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$ are independent variables, by implication T_1 and T_2 are also independent of one another, even though we state that $T_1 = T_2$ by setting $\alpha = 0$ in one of the coupling conditions. Of course this condition is satisfied at convergence, but perturbations of $\tilde{\mathbf{x}}$, with $\tilde{\mathbf{y}}$ held constant, and vice-versa, result in a non-zero value of $(1-\beta)(T_1 - T_2)$ in the Cats2D matching condition.

6.3. Results and discussion

All cases are computed with $\kappa = 2$, based on growth chamber conductivity of 1 W/m^2 and bath conductivity of 2 W/m^2. The temperature difference between top and bottom of the heat bath is taken to be 50 K. Two Rayleigh numbers are considered: Ra = 0, for which flow is zero, and Ra $= 6 \times 10^3$, for which a buoyant flow occurs. Iterations at Ra = 0 are started with the solution initialized to zero. At Ra $= 6 \times 10^3$, the Cats2D problem is sufficiently nonlinear that Newton's method will not converge from a zero initial guess. The solution is reached instead by a parameter continuation method, in which iteration is started from a converged solution computed at the lower value Ra $= 1 \times 10^3$. The finite-differencing parameter is $\epsilon = 10^{-4}$ in all calculations. The dimension of the Schur problem is 185, which is the number of points j along the boundary at which CrysVUn outputs its heat flux. In comparison, the problem in Ω_1 solved by Cats2D is discretized by 7883 unknowns, and the problem in Ω_2 solved by CrysVUn is discretized by 1047 unknowns.

Fig. 8 shows temperature contours and streamline contours at Ra $= 6 \times 10^3$. Isotherms, shown in growth chamber and bath, are spaced at temperature intervals of 1 K. Dashed streamline contours indicate clockwise flow and solid streamline contours indicate counter-clockwise flow. The flow consists of a pair of co-rotating toroidal vortices. The flow structure, though unremarkable, represents a typical situation of convective heat transfer by thermal buoyancy in melt crystal growth systems.

Fig. 9 shows temperatures and heat fluxes computed by both codes at the shared boundary for both values of Ra considered. Temperature contours in the melt appear little affected by the flow, but the impact on heat flux at the boundary is quite large. The temperatures reported by the two codes agree to within a small interpolation error between the subproblem grids.

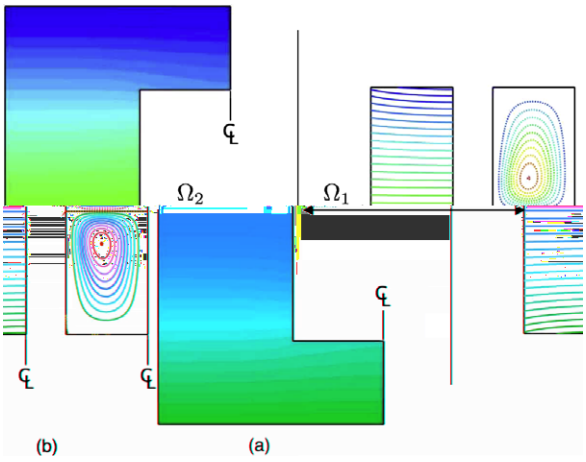

Fig. 8. Simulation of two-dimensional convective heat transfer for cylinder-in-cylinder geometry: (a) temperature contours in the bath (Ω_2), and (b) temperature and streamline contours in the growth chamber (Ω_1), for Ra $= 6 \times 10^3$. Isotherms are spaced at $\Delta T = 1$ K. Dashed streamline contours indicate clockwise flow, with a spacing of $\Delta\psi = \psi_{max}/10$, and solid streamline contours indicate counter-clockwise flow, with a spacing of $\Delta\psi = \psi_{min}/10$, and $\psi_{max} = 1.6579$ and $\psi_{min} = -1.6576$, dimensionless.

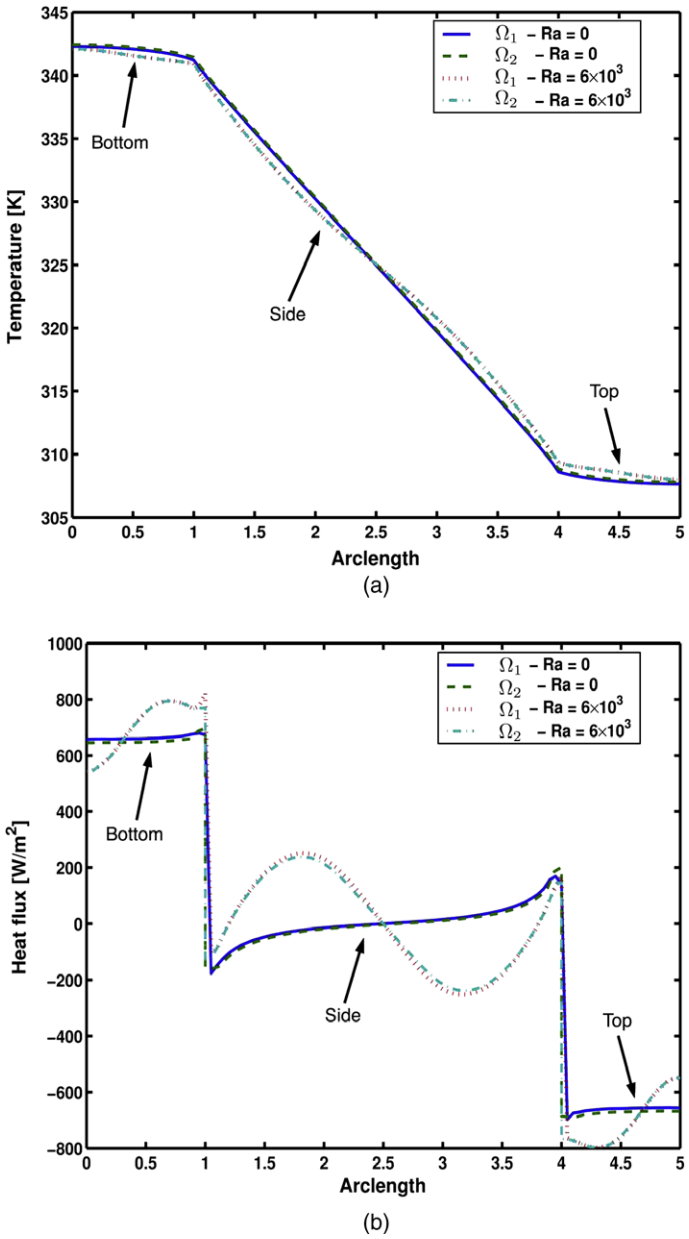

Fig. 9. Comparison of (a) temperature [K] and (b) heat flux [W/m^2] as a function of arc length along the shared boundary, for the cylinder-in-cylinder model with Ra = 0 and Ra = 6×10^3.

The fluxes are nearly indistinguishable, too, except at the corners where the cylinder side meets the top and bottom. The cause of the local disagreement is a post-processing error: the flux reported by the codes on output represents a local, pointwise differentiation of the temperature field that is not identical to the flux that would be applied in a boundary condition to achieve the identical solution. This discrepancy, ordinarily small, can be quite large at corners where a discontinuity in the normal vector makes the heat flux ill-defined. Cats2D uses a special procedure [25] for post-processing accurate fluxes at the corners, but CrysVUn does not, leading to the apparent discrepancy in the flux plots.

Fig. 10 shows the update norm versus iteration for both values of Ra considered. The effect of Krylov subspace size is studied by varying N_K. Both problems diverge under BGS iteration (not shown), but converge under ABN iteration provided $N_K \geqslant 2$. The linear case at Ra = 0, shown in Fig. 10(a), is expected to converge in a single iteration by exact Newton iteration. A comparable result is achieved with $N_K = 18$, which reduces the update norm by 10 orders of magnitude in a single iteration. The method performs nearly as well with $N_K = 14$ and $N_K = 10$, both of which reduce the update norm below 10^{-5} in a single iteration. Fig. 10(b) shows the nonlinear case for Ra = 6×10^3. The method performs well, and convergence is qua-

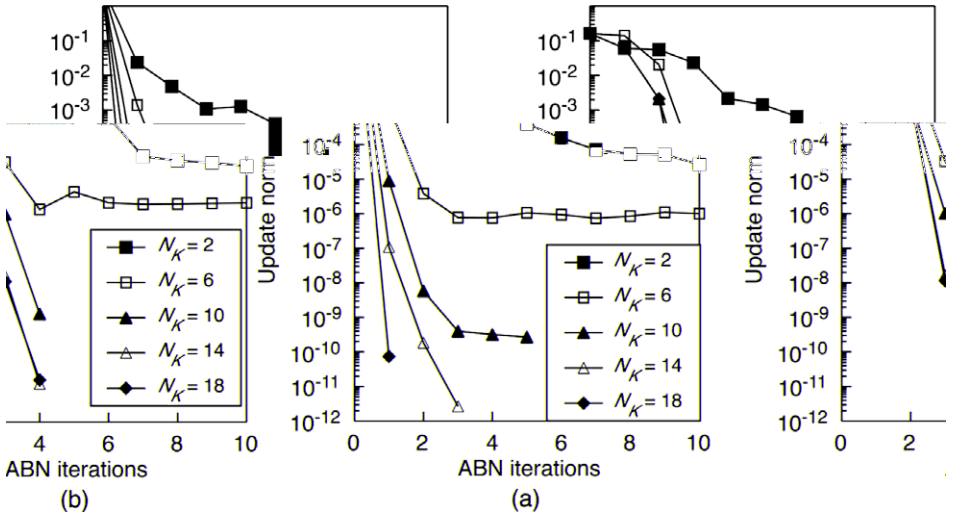

Fig. 10. The dependence of convergence rate on Krylov subspace size is determined for the cylinder-in-cylinder model by plotting the L_2-norm of the temperature update versus ABN iteration at various values of N_K for (a) Ra = 0 and (b) Ra = 6×10^3.

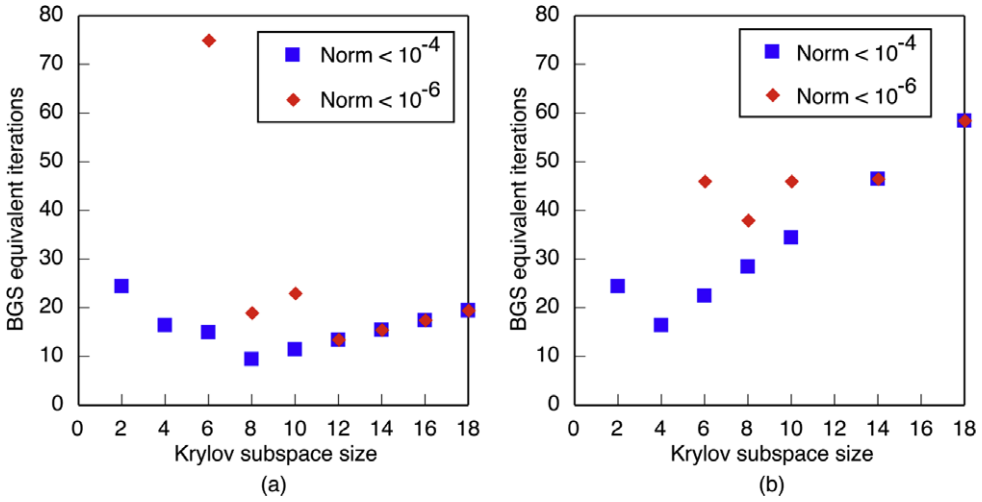

Fig. 11. Computational effort of ABN iteration, measured in BGS-equivalent iterations, is plotted versus size of Krylov subspace N_K for (a) Ra = 0 and (b) Ra = 6×10^3.

dratic with $N_K \geqslant 10$. These subspace sizes are at least an order of magnitude smaller than the dimension of **S**, which we believe is a consequence of the high quality preconditioning intrinsic to the ABN method.

Optimal convergence, in the sense of minimizing ABN iterations, is achieved in these problems by setting $N_K > 10$, but the computational cost per iteration rises with N_K. To determine optimum behavior in terms of computational effort, we convert ABN iterations into BGS-equivalent iterations as measured by the number of times each solver must be evaluated per iteration ($2 + N_K$ for $\mathcal{F}$ and $1 + N_K$ for $\mathcal{G}$ in the ABN method versus one evaluation of each solver in the BGS method). While this is a reasonable metric with which to evaluate computational effort, we note that the BGS algorithm itself diverges when applied to this problem. The results are plotted in Fig. 11 for both values of Ra. Two different update norm tolerances for determining convergence are considered, $|\Delta \mathbf{y}|_2 < 10^{-6}$ and $|\Delta \mathbf{y}|_2 < 10^{-4}$. The lowest cost in these examples occurs when N_K is somewhere between 4 and 8, which is too small to realize quadratic convergence, but large enough to achieve superlinear convergence.

7. Concluding remarks

We have derived, implemented, and tested several block Newton methods for the modular solution of black box models arising from the coupling of nonlinear, conjugate heat transfer problems. These approaches offer markedly superior perfor-

mance over simpler implementations that are based on block Gauss–Seidel (BGS) iterations. In particular, we identify two new and promising approximate block Newton algorithms, termed the ABN and ABN-J methods, that follow from prior implementations [12–15]. The performance of the ABN and ABN-J methods is indistinguishable for the examples studied here, yielding robust quadratically convergent solution algorithms. The method of Matthies and Steindorf also converges quadratically, but sometimes performs poorly for poor initial solution guesses, making it somewhat less robust than the ABN or ABN-J methods. For the larger-scale, two-dimensional problem considered here, the ABN method proved to be robust and to converge quadratically, provided the Krylov subspace size is large enough to solve the Schur problem accurately.

Of the two approaches derived here, we advocate the use of the ABN method: it is simple to derive and comprehend, it requires one less solver evaluation per iteration, and its implementation is a natural extension of the block Gauss–Seidel method. Indeed, the heart of the procedure, Eq. (19) or (25), can be written:

$$\Delta\mathbf{y} = \mathbf{S}^{-1}\Delta^{\mathcal{G}}\mathbf{y}(\mathbf{x}+\mathbf{q},\mathbf{y}) \tag{99}$$

which makes it clear that **S** constitutes a mapping between the BGS step and the ABN step.

It is interesting, though hardly a coincidence, that all three methods studied here arrive at similar forms by different pathways. The derivation of Matthies and Steindorf starts with a residual based on the block Jacobi step (cf. Eq. (70)), but block elimination to the Schur complement form leads to the BGS form of the residual (cf. Eq. (47)). In contrast, the ABN-J derivation starts with the exact Newton step instead of a fixed-point form of the residuals; here the key to reaching the BGS form is the preconditioning in Eq. (31). Both methods arrive at Eq. (99) by approximation, however, whereas the ABN method uses this equation as its starting point (which itself is an approximation, in consideration of the special case outlined in Eqs. (15) and (16)).

The difference in the methods lies in the evaluation of **S**, or more specifically the approximation to **Sw**. The ABN method uses a form based directly on Eq. (99), with no further approximation except to apply finite differences in the JFNK procedure used to solve the Schur problem. The other methods arrive at their approximations by more circuitous paths. The ABN-J derivation reaches a form in which the evaluation is shifted from the ABN form by the Jacobi update **q** (cf. Eq. (52)); the methods are equivalent at convergence, however. The derivation of Matthies and Steindorf introduces a spurious perturbation of **y**, a consequence of using the subproblem solvers in place of solvers for the fixed-point forms, which are unavailable. Their method also is equivalent to ABN and ABN-J methods at convergence, but the spurious perturbation results in somewhat diminished performance outside the quadratic regime.

While a significant motivation for this work has focused on quadratic convergence of the algorithm, iteration of the two-dimensional heat transfer example was most computationally efficient when the Schur problem was solved approximately on a small subspace, resulting in superlinear convergence of ABN iterations. For example, Fig. 11 showed that more ABN iterations are needed to converge, but fewer solver evaluations are required at each iteration due to the decreased Krylov subspace. A related issue arises when the subproblem solvers themselves fail to achieve quadratic convergence. Along these lines Menck [14] has proposed a control mechanism to optimize convergence when the subproblem solvers converge linearly at best. Based on an ABN-type algorithm similar to that of Matthies and Steindorf, he outlines the theory for determining optimum values of under-relaxation parameters and convergence tolerances for the subproblems. Such an approach applied to the ABN method could be beneficial to optimizing its performance.

Other simple cost-saving strategies exist. For example, if a direct solver can be used on either subproblem, great savings can be realized by reusing the Jacobian to compute the perturbed solutions. For computing

$$\mathcal{F}(\mathbf{x},\mathbf{y}+\epsilon\mathbf{w}) \approx \mathbf{x} - \mathbf{f}_{\mathbf{x}}^{-1}(\mathbf{x},\mathbf{y})\mathbf{f}(\mathbf{x},\mathbf{y}+\epsilon\mathbf{w}) \tag{100}$$

the Jacobian inverse $\mathbf{f}_{\mathbf{x}}^{-1}(\mathbf{x},\mathbf{y})$ could be reused for each perturbation, requiring only the effort of back-substitution, typically much faster than refactorizing the Jacobian. This approach also improves slightly on the basic algorithm, in that it eliminates the error incurred in adopting Eq. (45) as an approximation to Eq. (40). A similar consideration applies to solver $\mathcal{G}$. Along these same lines, computational effort may be even further reduced by employing an iterative solver for perturbation steps, such as Eq. (100) above, rather than a direct solver. Certainly an excellent initial guess for this problem is readily available, only a perturbation ϵ away, with which to seed the solution iterations.

Another important performance consideration is selection of the finite difference parameter ϵ. Varying this parameter had little effect on the calculations here, which we attribute to careful scaling of the subproblems. The scaling situation will be much less favorable in many practical applications, however. In such cases careful selection of ϵ may be necessary to assure accurate finite difference expressions that are neither too large for accuracy, nor too small for numerical stability. Though we have treated ϵ as a constant, it may be advantageous to set this parameter to a different value for each Krylov vector **w**, in order to control the relative norm of the perturbation. Methods for controlling ϵ for JFNK applications are discussed in Ref. [16].

The algorithms presented here show great promise for the efficient and robust solution of coupled, nonlinear problems of conjugate heat transfer, problems for which simpler iteration approaches have proven ineffective. However, the outcome of this development is potentially much broader. Such algorithms may allow for innovative ways to couple existing software packages that have been developed to solve specific problems, especially for modeling multiscale and multiphysics problems. Not only will effective block Newton coupling methods allow for significant savings of effort in software development,

their increased robustness and increased computational effectiveness will enable the study of new worlds of complex physical phenomena via computing.

Acknowledgments

This work has been supported in part by a seed grant from the Minnesota Supercomputing Institute and by the Department of Energy, National Nuclear Security Administration, under Award Numbers DE-FG52-06NA27498 and DE-FG52-08NA28768, the content of which does not necessarily reflect the position or policy of the United States Government, and no official endorsement should be inferred. Andrew Yeckel gives thanks to Fraunhofer Gesellschaft for financial support through a PROF.X^2 scholarship, to Jochen Friedrich for hosting his stay at the Crystal Growth Laboratory at Fraunhofer IISB, and to Thomas Jung for advice on research direction at a critical juncture.

References

[1] G. Dubini, R. Pietrabissi, F.M. Montevecchi, Fluid–structure interaction in bio-fluid mechanics, Med. Eng. Phys. 17 (1995) 609–617.
[2] S. Piperno, C. Farhat, B. Larrouturou, Partitioned procedures for the transient solution of coupled aeroelastic problems – Part I: Model problem, theory, and two-dimensional application, Comput. Method Appl. Mech. Eng. 124 (1995) 79–112.
[3] S. Piperno, C. Farhat, Partitioned procedures for the transient solution of coupled aeroelastic problems – Part II: Energy transfer analysis and three-dimensional applications, Comput. Method Appl. Mech. Eng. 190 (2001) 3147–3170.
[4] C. Farhat, M. Lesoinne, P. LeTallec, Load and motion transfer algorithms for fluid/structure interaction problems with non-matching discrete interfaces: momentum and energy conservation, optimal discretization and application to aeroelasticity, Comput. Method Appl. Mech. Eng. 157 (1998) 95–114.
[5] A. Pandy, Global Modeling of Bulk Crystal Growth of Cadmium Zinc Telluride in Industrial VB/VGF Systems, Ph.D. Thesis, University of Minnesota, 2004.
[6] A. Pandy, A. Yeckel, M. Reed, C. Szeles, M. Hainke, G. Müller, J.J. Derby, Analysis of the growth of cadmium zinc telluride in an electrodynamic gradient freeze furnace via a self-consistent, multi-scale numerical model, J. Cryst. Growth 276 (2005) 133–147.
[7] A. Yeckel, A. Pandy, J.J. Derby, Fixed-point convergence of modular steady-state heat transfer models coupling multiple scales and phenomena for melt-crystal growth, Int. J. Numer. Method Eng. 67 (2006) 1768–1789.
[8] J.J. Derby, L. Lun, A. Yeckel, Strategies for the coupling of global and local crystal growth models, J. Cryst. Growth 303 (2007) 114–123.
[9] R.W. Hooper, M.H. Hopkins, R.P. Pawlowski, Enabling Newton-based coupling within a multi-physics environment using NOX – an object-oriented nonlinear solver library, in: M. Papadrakakis, E. Oñate, B. Schrefler (Eds.), International Conference on Computational Methods for Coupled Problems in Science and Engineering, CIMNE, Barcelona, 2005.
[10] R.W. Hooper, M.H. Hopkins, R.P. Pawlowski, B. Carnes, H.K. Moffat, Final Report on LDRD Project: Coupling Strategies for Multi-physics Applications, Tech. Rep. SAND2007-7146, Sandia National Laboratories, 2008.
[11] T.G. Kolda, R.P. Pawlowski, NOX Home Page. <http://software.sandia.gov>.
[12] T.F. Chan, An approximate Newton method for coupled nonlinear systems, SIAM J. Numer. Anal. 22 (5) (1985) 904–913.
[13] S. Artlich, W. Mackens, Newton-coupling of fixed point iterations, in: W. Hackbusch, G. Wittum (Eds.), Numerical Treatment of Coupled Systems, Vieweg-Verlag, Braunschweig, Wiesbaden, 1995, pp. 1–10.
[14] J. Menck, An approximate Newton-like coupling of subsystems, Z. Angew. Math. Mech. 82 (2) (2002) 101–114.
[15] H.G. Matthies, J. Steindorf, Partitioned but strongly coupled iteration schemes for nonlinear fluid–structure interaction, Comput. Struct. 80 (2002) 1991–1999.
[16] D.A. Knoll, D.E. Keyes, Jacobian-free Newton–Krylov methods: a survey of approaches and applications, J. Comput. Phys. 193 (2004) 357–397.
[17] Y. Saad, Iterative Methods for Sparse Linear Systems, PWS Pub. Co., Boston, 1996.
[18] A. Yeckel, J.J. Derby, Computer modelling of bulk crystal growth, in: P. Capper (Ed.), Bulk Crystal Growth of Electronic, Optical and Optoelectronic Materials, Springer, Berlin, 2005, pp. 73–119.
[19] A. Yeckel, R.T. Goodwin, Cats2D (Crystallization and Transport Simulator), User Manual, 2003. <http://www.msi.umn.edu/yeckel/cats2d.html>.
[20] M. Kurz, A. Pusztai, G. Müller, Development of a new powerful computer code CrysVUN++ especially designed for fast simulation of bulk crystal growth processes, J. Cryst. Growth 198/199 (1999) 101–106.
[21] M. Kurz, G. Müller, Control of thermal conditions during crystal growth by inverse modeling, J. Cryst. Growth 208 (2000) 341–349.
[22] J. Fainberg, D. Vizman, J. Friedrich, G. Müller, A new hybrid method for the global modeling of convection in CZ crystal growth configurations, J. Cryst. Growth 303 (2007) 124–134.
[23] CrysMAS. <http://www.iisb.fraunhofer.de>.
[24] Y. Saad, M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (1986) 856–869.
[25] R.L. Lee, P.M. Gresho, R.L. Sani, Smoothing techniques for certain primitive variable solutions of the Navier–Stokes equations, Int. J. Num. Method Eng. 14 (1979) 1785–1804.