Operator splittings for solving nonlinear, coupled multiphysics problems with an application to the numerical solution of an interface problem

Owe Axelsson* Maya Neytcheva†

April 8, 2011

This paper is dedicated to the memory of Jan Verwer, who recently suddenly and unexpectedly deceased, for his impressive contributions to numerical analysis and for his strict, but fair and friendly, manner in dealing with other scientists work.

Abstract

This work summarizes operator splitting methods to solve various kinds of coupled multiphysics problems. Such coupled problems are usually stiff. Furthermore, one is often interested in obtaining stationary solutions, which require integration over long time intervals. Therefore, an implicit and stable time-stepping method of at least second order of accuracy must be used, to allow for larger timesteps. To control the operator splitting errors for nonlinear problems, an approximate Newton solution method is proposed for each separate equation. After completion of some (normally few) Newton steps, the equations are updated with the current solution, thereby preparing for the next sequence of Newton steps.

An application for a nonlinear model of interface tracking problem arising in a multiphase flow is described. Hereby an inner–outer iterative solution method with a proper preconditioning for solving the arising linearized algebraic equations, which results in few iterations, is analyzed. There is no need to update the preconditioner during the iterations.

1 Introduction

In the modeling of many problems arising in practice, coupled multiphysics phenomena are involved, that is, partial differential equations of different types appear in the model. An example is the interaction of an elastic body with a fluid flow, see, e.g., Glowinski et

*Institute of Geonics AS CR, Ostrava, The Czech Republic and Department of Information Technology, Uppsala University owe.axelsson@it.uu.se

†Department of Information Technology, Uppsala University, maya.neytcheva@it.uu.se

al. ([17]). Another example arises in the modeling of an interface between two (or more) phases, such as fluid and gas phase of some substance, see, e.g., Novick–Cohen ([34]). Both problems involve nonlinear equations for the concentration, the pressure and the velocity, where the latter two are modelled by Navier-Stokes equations, see, e.g., [17] and [34] for further details. A model involving exceptionally many variables and corresponding partial differential equations arises in the transport of air pollutants, where subprocesses such as advection, diffusion, emission, deposition, heat conduction, and chemical reactions are coupled, see, e.g., Verwer at al. ([38]), Dimov et al. ([13]) and Zlatev ([43]).

Clearly, such problems can be computationally very demanding and complicated to solve and it is not, in general, viable to handle them as one coupled system using say, an iterative solution method based on linearizations as in Newton's method for the whole system. In fact, the arising linearized equation matrix may be indefinite, or even singular. Instead, one can treat each subvariable and corresponding partial differential equation separately and couple the obtained solutions together after each iteration step, in this way forming the new, updated equations as well as initial conditions for the next iteration step.

Normally, the processes evolve with time, i.e., time-dependent partial differential equations, mostly of parabolic type arise. In such a case, one must use some time-stepping method and the decoupling, and following recoupling of the equations can then take place at the beginning, respectively at the end of each time step.

Such methods, where each equation is solved separately, have been used since long. Early methods were named *fractional step methods*, see, e.g., Marchuk ([31]) and Yanenko ([42]). Incidently, the alternating direction method can also be seen as such a method. In that case one solves alternatively a parabolic equation along each coordinate axes direction, see, e.g., Douglas and Gunn ([14]). Indeed, separation of the solution components can also take place within each equation if it involves different physical phenomena, such as diffusion, advection, and reaction. Such splitting methods have been used, e.g., in Hundsdorfer and Verwer ([27]), Glowinski et al. ([17]). In the present paper, however, we are more concerned with multiphysics problems which are defined by separate equations. More generally, the methods can be defined as *operator splitting methods*. There are several variants of such methods and they are discussed in Section 2.

We remark here that the choice of timesteps is crucial for the efficiency of the methods. Choosing very small timesteps results in small splitting errors but causes many costly timesteps and large computer run times. For instance, the computer times reported in [17], lasted for several days or weeks, even on a modern computer. The choice of a proper efficient splitting method, which allows larger timesteps, is therefore crucial.

In Section 3 some multiphysics problems are shortly presented. The Cahn-Hilliard equation coupled with the Navier–Stokes equation, describing the interface dynamics between two phases of some substance in a moving fluid is presented in greater detail. Its numerical solution is discussed in Section 4. In particular, a preconditioning method for the linearized equations, occurring at each Newton step, is analysed. In Section 5 suitable preconditioners, which were originally constructed to solve linear systems arising in complex valued equations are presented. Section 6 contains a summary of an iterative solution method for the saddle point problem such as in the Navier–Stokes equation. Section 7 con-

tains some numerical tests using the methods from the previous sections. Some concluding remarks are found in the final section.

2 Operator splitting methods

Operator splitting methods to solve a system of coupled variables and corresponding partial differential equations can take place in several ways, which we now survey with some background theory.

2.1 Splitting methods

(PS) *Separate (or parallel) splitting method*
In the simplest version of operator splitting methods, for each timestep the coupled equations are solved separately and the solutions of the different variables are then used to update the equations and form initial conditions for the next timestep. This method is similar to the Jacobi iteration method. The advantage with the method is its simplicity as well as that it allows parallel computation of the solutions to the equations. Clearly, in general, there appear splitting errors and time–stepping errors which must be controlled by choosing sufficiently small timesteps. As already mentioned, however, too small timesteps can lead to large computer simulation times.

(SO) *Sequential ordering method*
In a sequential procedure, one chooses first an ordering of the equations, then solves the first equation and substitutes its solution into the other equations, where it appears. Then the second equation is solved and its solution used to update the other equations, etc. In this way one continues sequentially, equation by equation. This method is similar to the Gauss–Seidel iteration method and is the more traditional one, see, e.g., [15].

(SS) *Symmetric splitting method*
In order to improve the accuracy of the sequential splitting, Strang ([36]) and Marchuk ([31]) proposed a method, where for the case of two equations, one solves the first equation for the first half of the timestep, substitutes the solution into the second equation, which is then solved on the whole timestep, its solution substituted into the first equation which is finally solved for the second half of the timestep.

As pointed out in e.g. Faragó et al. ([11]), it is recommended to let the second equation be the one which depends more strongly on the solution of the first equation and is more difficult to solve. In this way, the splitting error depends on half of the timestep and the most difficult equation is solved only once during each timestep. In air pollution models one chooses often the stiff chemistry operator equation for the second equation, see, e.g., Verwer et al. ([38]).

On the other hand, the second equation is normally solved also in two parts for half a sized timestep. Then the solution cost is the same for both equations. This method can be seen as carrying out the two half–steps with reversed order of the equations. In this way, the scheme becomes symmetrized and this leads to a second order method, as we shall see.

(SW) *Symmetrized weighted sequential splitting method*
As described above, at the cost of solving all equations twice during each timestep, the sequential splitting method can be symmetrized. Applying also a weighting of the solution, allows for reducing the error further, see, e.g., Faragó et al. ([11]).

For the case of two equations, ordered as 1 and 2, one solves first equation 1, then equation 2, and after that solves them in the opposite order, equation 2 followed by equation 1. Finally, a weighted average of the obtained solutions is taken, which then forms the initial conditions to update the equations and to start the computations for the next timestep.

Typically, the parallel and sequential methods lead to splitting errors of first order $(O(\tau))$, while the improved, i.e., symmetrized and symmetrized weighted methods lead to second order $(O(\tau^2))$ methods, where τ is the timestep. The time–stepping method to be used can be the Euler forward method and some second order method, respectively, to match those orders. For reasons of numerical stability, for stiff equations one must use an implicit time-stepping method. The simplest methods are then the Euler backward method $(O(\tau))$ and the Crank Nicholson $(O(\tau^2))$, or a weighted time-stepping, sometimes called the θ-method, see, e.g., [3].

Remark 1 In large scale real-life multiphysics problems one is often interested in the stationary state of the underlying system, which might be attained after a long time. Therefore, to speed up the numerical simulations we need to progress in time with as large time steps as possible. Time integration schemes, based on the θ method do not allow us to use very long time steps because those would ultimately destroy the discretization error, guaranteed for the solution. The largest time step one can use is $\tau = h$ provided that we have second order accurate scheme in time and that the splitting error is not prevailing the discretization error.

The alternative is to use higher order methods, such as Radau quadrature based implicit Runge-Kutta methods, see, e.g., [2]. Frequently, the differential equations are stiff and for the convergence analysis it is then important to use one-sided Lipschitz constants, see, for instance [3], [23]. However, enabling the utilization of higher order time integration schemes may require to treat the coupled equations in the context of differential-algebraic systems (DAE) context.

2.2 Operator splitting errors

We now give an outline of the derivation of the splitting errors. The simultaneous splitting method leads to splitting errors of order $O(\tau)$. Likewise, the sequential method leads

to a splitting error of that order. As mentioned, for sufficiently regular solutions, the symmetrized method leads to a splitting error $O(\tau^2)$. This holds also for the symmetrized, weighted method.

We present first an algebraic formulation of the splitting methods.

Algebraic formulation of the splitting methods

Consider the equation

$$\frac{\partial \mathbf{u}(t)}{\partial t} = \mathscr{A}(t, \mathbf{u}(t)),\ 0 < t \leq T\,, \tag{1}$$

where $\mathbf{u}(0) = \mathbf{u}_0$ is given and where $\mathscr{A}$ is a differential operator, frequently nonlinear, in the n components of the unknown vector function $\mathbf{u}(t)$. Some component may itself have d components when we work with differential operators in a d-dimensional space. We assume that (1) has a unique solution and that the operator $\mathscr{A}$ can be split in a sum of operators,

$$\mathscr{A}(t, \mathbf{u}(t)) = \sum_{i=1}^{n} \mathscr{A}^{(i)}(t, \mathbf{u}(t))$$

Thereby, each equation

$$\frac{\partial \mathbf{u}^{(i)}(t)}{\partial t} = \mathscr{A}^{(i)}(t, \mathbf{u}(t)),\ 0 < t \leq T\,,$$

where the components $u^j(t)\,,\ j \neq i$ are given, is also assumed to have a unique solution.

The time interval $[0, T]$ is divided in subintervals. For notational simplicity we assume that these are of equal length, i.e., $\tau = T/m$. Let $t_k = k\tau\,,\ k = 0, 1, \ldots, m$.

The different operator splitting methods can now be defined algebraically in the following way.

(PS) Simultaneous (parallel) splitting method

Let $\widehat{\mathbf{u}}(t_0) = \mathbf{u}_0$. At timestep k, given initial values $\widehat{\mathbf{u}}(t_{k-1})$ for $k = 1, 2, \ldots, n$, solve

$$\frac{\partial u^{(i)}(t)}{\partial t} = \mathscr{A}^{(i)}(t, \widehat{\mathbf{u}}(t)),\ t_{k-1} < t \leq t_k\,,$$

where $\widehat{u}^{(j)}(t) = u^{(j)}(t_{k-1})$, $j = 1, \ldots, n$, $j \neq i$ and with initial condition $u^{(j)}(t_{k-1}) = \widehat{u}^{(j)}(t_{k-1})$, given. Then let $\widehat{\mathbf{u}}(t_k) = [u^{(j)}(t_k)]_{j=1}^{n}$ be the initial condition for the next timestep t_{k+1}.

(SO) Sequential splitting

Let $\widehat{\mathbf{u}}(t_0) = \mathbf{u}_0$. At timestep k, given initial values $\widehat{\mathbf{u}}(t_{k-1})$, for $i = 1, 2, \ldots, n$, solve

$$\frac{\partial u^{(i)}(t)}{\partial t} = \mathscr{A}^{(i)}(t,\ \widehat{\mathbf{u}}(t)),\ t_{k-1} < t \leq t_k\,,$$

where

$$\begin{cases} \widehat{u}^{(j)}(t) = u^{(j)}(t_k)\,, & j = 1, \ldots, i-1 \\ \widehat{u}^{(j)}(t) = u^{(j)}(t_{k-1})\,, & j = i+1, \ldots, n \end{cases}$$

and with an initial condition $u^{(j)}(t_{k-1}) = \widehat{u}^{(j)}(t_{k-1})$ given. Then let $\widehat{\mathbf{u}}(t_k) = [u^{(j)}(t_k)]_{j=1}^n$ be the initial condition for the next timestep.

To describe the improved schemes with smaller splitting errors we consider now just two equations, $n = 2$.

(SS) The symmetrized splitting method

Let $\widehat{\mathbf{u}}(t_0) = \mathbf{u}_0$. At timestep k, given initial values $\widehat{\mathbf{u}}(t_{k-1})$, solve in order,

(i)

$$\frac{\partial u^{(1)}(t)}{\partial t} = \mathscr{A}^{(1)}(t, \widehat{\mathbf{u}}(t)),\ t_{k-1} < t \le t_{k-1/2} := t_{k-1} + \frac{1}{2}\tau\,,$$

where $\widehat{u}^{(2)}(t) := \widehat{u}^{(2)}(t_{k-1})$, and $u^{(1)}(t_{k-1}) = \widehat{u}^{(1)}(t_{k-1})$;

(ii)

$$\frac{\partial u^{(2)}(t)}{\partial t} = \mathscr{A}^{(2)}(t, \widehat{\mathbf{u}}(t)),\ t_{k-1} < t \le t_k\,,$$

where $\widehat{u}^{(1)}(t) = u^{(1)}(t_{k-1/2})$, and $u^{(2)}(t_{k-1}) = \widehat{u}^{(2)}(t_{k-1})$;

(iii)

$$\frac{\partial u^{(1)}(t)}{\partial t} = \mathscr{A}^{(1)}(t, \widehat{\mathbf{u}}(t)),\ t_{k-1/2} < t \le t_k\,,$$

where $\widehat{u}^{(2)}(t) = u^{(2)}(t_{k-1/2})$, and the initial value $u^{(1)}(t_{k-1/2})$ is given from the first step (i).

(SW) The symmetrized weighted scheme

Let $\widehat{\mathbf{u}}(t_0) = \mathbf{u}_0$.
At timestep k, given initial values $\widehat{\mathbf{u}}(t_{k-1})$, solve in order, for $\widehat{u}^{(1)}(t)\,, t_{k-1} < t \le t_{k-1/2}$,
$\widehat{u}^{(2)}(t)$, $t_{k-1} < t \le t_k$ and then $\widehat{u}^{(1)}(t)$, $t_{k-1/2} < t \le t_k$ as in the symmetrized step method. Then repeat this method but in the opposite order

$$\widehat{\widehat{u}}^{(2)}(t),\ t_{k-1} < t \le t_{k-1/2},\ \widehat{\widehat{u}}^{(1)}(t),\ t_{k-1} \le t < t_k \quad \text{and}\, \widehat{\widehat{u}}^{(2)}(t),\ t_{k-1/2} \le t \le t_k\,.$$

Finally, take a weighted average of the so–obtained solutions

$$\mathbf{u}(t_k) = \theta\widehat{\mathbf{u}}(t_k) + (1-\theta)\widehat{\widehat{\mathbf{u}}}(t_k)\,,$$

for some θ, $0 < \theta < 1$. Then $\mathbf{u}(t_k)$ form initial values for the next timestep.

Under certain conditions, the splitting errors in the previous two methods are $O(\tau^2)$. The "hidden" constant in the expression can be expected to be smaller for the second method. The symmetrized weighted method requires, however, a double amount of computations per timestep.

Operator splitting errors

For the following derivation we assume sufficient differentiability and regularity to hold. Consider a nonlinear system consisting of two coupled equations,

$$\begin{aligned} \frac{du^{(1)}(t)}{dt} &= \mathscr{A}^{(1)}(t, u^{(1)}(t), u^{(2)}(t)) \\ \frac{du^{(2)}(t)}{dt} &= \mathscr{A}^{(2)}(t, u^{(1)}(t), u^{(2)}(t)) \,, \end{aligned}$$

$t > 0$, where $u^{(1)}(0), u^{(2)}(0)$ are given.
For given initial values at $t = t_{k-1}$, $k = 1, 2, \ldots$ the sequential operator splitting method in the interval (t_{k-1}, t_k), where $t_k - t_{k-1} = \tau > 0$, takes the form

$$\begin{aligned} \frac{d\widehat{u}^{(1)}(t)}{dt} &= \mathscr{A}^{(1)}(t, \widehat{u}_1(t), u^{(2)}(t_{k-1})),\ t_{k-1} < t \le t_k \\ \frac{d\widehat{u}^{(2)}(t)}{dt} &= \mathscr{A}^{(2)}(t, \widehat{u}_1(t_k), \widehat{u}^{(2)}(t)),\ t_{k-1} < t \le t_k \,, \end{aligned} \tag{2}$$

with initial values $\widehat{u}^{(1)}(t_{k-1}) = u^{(1)}(t_{k-1})$, $\widehat{u}^{(2)}(t_{k-1}) = u^{(2)}(t_{k-1})$, respectively. To find the local splitting errors, i.e., local truncation errors due to the splitting, we insert the exact solution $\mathbf{u}(t)$ into (2), and obtain

$$\begin{aligned} \frac{du^{(1)}(t)}{dt} &- \mathscr{A}^{(1)}(t, u^{(1)}(t), u^{(2)}(t_{k-1})) \\ &= \mathscr{A}^{(1)}(t, u^{(1)}(t), u^{(2)}(t)) - \mathscr{A}^{(1)}(t, u^{(1)}(t), u^{(2)}(t_{k-1})) \\ &= \frac{\partial \mathscr{A}^{(1)}}{\partial u^{(2)}}(u^{(2)}(t) - u^{(2)}(t_{k-1})) + O(\|u^{(2)}(t) - u^{(2)}(t_{k-1})\|^2) \\ &= (t - t_{k-1})\frac{\partial \mathscr{A}^{(1)}}{\partial u^{(2)}}\mathscr{A}^{(2)} + O(\|u^{(2)}(t) - u^{(2)}(t_{k-1})\|^2) \end{aligned} \tag{3}$$

Similarly, for the second equation, we get

$$\begin{aligned} \frac{du^{(2)}(t)}{dt} - \mathscr{A}^{(2)}(t, u^{(1)}(t_k), u^{(2)}(t)) &= \frac{\partial \mathscr{A}^{(2)}}{\partial u^{(1)}}(u^{(1)}(t) - u^{(1)}(t_k)) \\ &\quad + O(\|u^{(1)}(t) - u^{(1)}(t_{k-1})\|^2) \\ &= (t - t_k)\frac{\partial \mathscr{A}^{(2)}}{\partial u^{(1)}}\mathscr{A}^{(1)} + O(\|u^{(1)}(t) - u^{(1)}(t_k)\|^2) \,. \end{aligned} \tag{4}$$

Hence, the local truncation errors are $O(\tau)$. To find the corresponding local discretization errors in $u^{(1)}(t_k)$ and $u^{(2)}(t_k)$, we perform time integrations from t_{k-1} to t_k in (3) and (4), and find that the error factor of $\frac{\partial \mathscr{A}^{(1)}}{\partial u^{(2)}}\mathscr{A}^{(2)}$ in the major error term, equals

$$\int_{t_{k-1}}^{t_k} (t - t_{k-1})dt = \frac{1}{2}(t_k - t_{k-1})^2 = O(\tau^2)$$

and, similarly,

$$\int_{t_{k-1}}^{t_k} (t - t_k)dt = -\frac{1}{2}(t_k - t_{k-1})^2$$

for $u^{(1)}(t_k)$ and $u^{(2)}(t_k)$, respectively. Note here the negative sign in the second factor. Hence, the local discretization errors are $O(\tau^2)$.

Clearly, the corresponding global discretization errors equal $O(\tau)$, in general.

Following Strang ([36]) and Marchuk ([31]), to improve on the error order we use the symmetrical splitting. We then carry out the above splitting but only for the first half timestep, i.e., in the interval $(t_{k-1}, t_{k-1/2})$, where $t_{k-1/2} = t_{k-1} + \frac{1}{2}\tau$, followed by the equations in the reverse order on the second half, $(t_{k-1/2}, t_k)$ of the time interval.

It follows from the above that the factor in the leading term of the local splitting errors in the solutions becomes

$$\frac{1}{2}(t_{k-1/2} - t_{k-1})^2 - \frac{1}{2}(t_k - t_{k-1/2})^2 = 0\,.$$

Hence, since the major factors of $\frac{\partial \mathscr{A}^{(1)}}{\partial u^{(2)}}\mathscr{A}^{(2)}$ and $\frac{\partial \mathscr{A}^{(2)}}{\partial u^{(1)}}\mathscr{A}^{(1)}$ respectively, in the two components cancel, it follows that the local and global discretization errors now become $O(\tau^3)$ and $O(\tau^2)$, respectively.

Since, due to a normally smaller condition number of the system matrix involved, the computational cost of performing a half timestep can be lower than that for a full timestep, the expense of the symmetrical splitting method can be less than twice that for the sequential splitting and, furthermore, allows in general much larger timesteps to get a certain prescribed accuracy. Clearly, the time stepping method should also be a second or higher order method.

Operator splitting methods can also take place within a given equation if it contains certain terms of different physical behavior. For instance, in an advection–diffusion–reaction equation one can split the solution in three parts to be computed separately during each time step. This is a method favored, e.g., in [27], see also [17]. However, such problems give normally rise to interior and boundary layers which may be difficult to resolve unless one treats the equation as a whole equation.

Similarly, for parabolic partial differential equations one can use locally one–dimensional splittings, which enables solutions of just one–dimensional problems at each timestep. This is the so–called alternating direction method, where the splitting is performed according

to the directions of the coordinate axes. For the equivalence of locally one–dimensional splittings and alternating direction methods, see [19].

Such methods will not be further discussed in this paper. We just mention here, as can also be seen from the previous derivations, that for a splitting of a single equation in two terms,

$$\mathscr{A}(t, u(t)) = \mathscr{A}^{(1)}(t, u(t)) + \mathscr{A}^{(2)}(t, u(t))$$

the operator factor in the major term in the local splitting error takes the form

$$\frac{\partial \mathscr{A}^{(2)}}{\partial u}\mathscr{A}^{(1)} - \frac{\partial \mathscr{A}^{(1)}}{\partial u}\mathscr{A}^{(2)} .$$

For a linear problem, this equals $\mathscr{A}^{(2)}\mathscr{A}^{(1)} - \mathscr{A}^{(1)}\mathscr{A}^{(2)}$. Hence, it cancels for commuting operators, see [11] for more general results regarding operator commutativity. Clearly, commutativity does rarely occur in practical problems. For some conditions where it occurs, see, e.g., [27].

To get the above splitting for a single equation on the same form as for the separate equation splittings, one can introduce separate variables, say $u^{(1)}$ for the diffusion part, $u^{(2)}$ for the advection part and $u^{(3)}$ for the reaction part. Then one forces $u^{(1)} = u^{(2)}$ and $u^{(1)} \equiv u^{(3)}$ by a penalty or by an augmented or regularized Lagrangian formulation method, see, e.g., [18].

Not only do real–life systems involve interactions between many physical phenomena, but the time and length scales of the individual processes involved often differ by orders of magnitude. Therefore, for many real–life problems both multiphysics and multiscale phenomena must be handled. Different time and length scales can be used for different equations and the solution from a coarse mesh can be interpolated to a finer mesh for some other equations, or vice versa. For efficiency, for some equations it can be necessary to use finer meshes in parts of the domain only. Multiscale methods have been discussed in many papers, for a recent publication, see [41], see also [26]. Due to space limitations we will not address this aspect further in this paper.

2.3 Differential-algebraic equations

The operator splitting techniques, as presented above, hold for differential equations in standard form. Many models in practice include, however, an algebraic constraint. An example to mention is the divergence-free condition for the velocity, which appears whenever incompressible fluids are included in the models (see Section 3 for some examples). Such models result in differential-algebraic systems (DAEs) to be solved, which are of the following form:

$$\begin{aligned} \mathscr{A}_1(u', u, v, t) &= 0 \\ \mathscr{A}_2(u, v, t) &= 0, \end{aligned} \tag{5}$$

where the second equation defines the algebraic constraint.

A DAE is said to have index s if it must (and can) be differentiated s times to obtain an equivalent standard differential equation form, without algebraic constraints. This

transformation can be done by differentiating the constraint equation,

$$\frac{d\mathscr{A}_2}{d\,t} = \frac{\partial \mathscr{A}_2}{\partial u}u' + \frac{\partial \mathscr{A}_2}{\partial v}v' + \frac{\partial \mathscr{A}_2}{\partial t}.$$

(Note here, that $\frac{\partial \mathscr{A}_2}{\partial u}$ and/or $\frac{\partial \mathscr{A}_2}{\partial v}$ may be singular or even identically zero.)

The reduction of (5) to a standard form can take place by eliminating the variable v. To illustrate this approach and indicate some possible difficulties, we look at a prototype example of the type we are concerned with in this paper. Consider

$$\begin{array}{rcl} M\mathbf{u}' + A\mathbf{u} + B^T\mathbf{v} &=& \mathbf{f} \\ C\mathbf{u} - D\mathbf{v} &=& \mathbf{g} \end{array}, \tag{6}$$

where M, A, B, C, D are linear operators (matrices) and (for simplicity) do not depend on time. We assume that M is nonsingular and that B and C have full rank. Also, we assume that M and A are of order $n \times n$ while B and C are of order $m \times n$, with $m < n$.

If $D = 0$, then $C\mathbf{u}' = \mathbf{g}'$ and

$$C\mathbf{u}' + CM^{-1}A\mathbf{u} + CM^{-1}B^T\mathbf{v} = CM^{-1}\mathbf{f}, \quad \text{or} \quad CM^{-1}B^T\mathbf{v} = CM^{-1}(\mathbf{f} - A\mathbf{u}) - \mathbf{g}'.$$

Then, $CM^{-1}B^T$ is nonsingular, so $\mathbf{v} = \left(CM^{-1}B^T\right)^{-1}(CM^{-1}(\mathbf{f} - A\mathbf{u}) - \mathbf{g}')$, and substituting the latter in the first equation in (6) gives us a differential equation in a standard form,

$$M\mathbf{u}' + A\mathbf{u} - B^T\left(CM^{-1}B^T\right)^{-1}CM^{-1}A\mathbf{u} = \mathbf{f} - B^T\left(CM^{-1}B^T\right)^{-1}(CM^{-1}\mathbf{f} + \mathbf{g}') \tag{7}$$

In this case, the DAE (6) has index 1.

Each differentiation increases the requirements of regularity of the right hand side function to achieve a certain order of the discretization error. If the index is higher than 1, then it increases also the requirements of regularity of the solution itself, that is, the lower the index is, the greater the possibility to achieve a certain order of the discretization is.

If D is nonsingular, than we can substitute $\mathbf{v} = D^{-1}C\mathbf{u} + \mathbf{g}$ in the first equation in (6). The operator D is nonsingular for compressible fluids or for regularized problems. In this case we do not need to require that B and C have full rank. The DAE has here index zero. For an exposure of numerical solution methods for DAEs, see [23] and also [20], particularly for a discussion on the solution of the Stokes problem.

Converting (6) to (7) enables us to use any high order stable time integration scheme, for instance order 4, or 6, which enables us to use very large time steps and still guarantee enough accuracy in the obtained solution. The form (7) indicates, however, that we may face significant numerical difficulties when solving the transformed equation. Therefore, in the numerical process we normally do not perform substitutions of the above type but rather solve the equations, discretized in space and time, by proper iterative solution methods.

2.4 Controlling the splitting error: combined timestepping and Newton's method

Operator splitting errors can in general drive the numerical solution to have an unphysical behavior and must therefore be controlled.

The most straightforward way to ascertain such a control is by choosing sufficiently small timesteps but, as remarked in the introduction, too small timesteps can make the solution method too expensive. Small timesteps can be avoided by using implicit time-stepping. This is also needed because the problems are normally stiff, at least due to the presence of diffusion. Second order, or higher order methods are preferred.

To control the operator splitting errors for nonlinear problems one can use some steps of a Newton method for each separate operator equation. After every Newton step, the solutions in all equations can be updated. In this way, we get smaller splitting errors than would otherwise occur for each full timestep.

However, the linearized Newton matrix may be indefinite or even singular. In such a case, the Newton method must be approximated, either by use of some approximation of the linearized Jacobian matrix or by using damped Newton steps. It can be combined with any of the splitting methods presented in this section.

In this paper we advocate the use of higher order time integration methods, which enable the use of much larger timesteps to get a requested accuracy. In such a case, one must in general solve the equations arising at each timestep performing several steps of some (approximate) version of a Newton method. In order to ensure that the savings in number of timesteps is not outweighed by too costly Newton steps, it is then essential that the linearized equations are solved efficiently. This is discussed for the particular application to follow.

In this paper we treat only the case of two coupled equations. This does not prevent that one, or both, of the equations themselves consist of subprocesses, involving different variables and corresponding split equations. As mentioned previously, one can also use a splitting of each equation for a single variable, if it involves different physical processes. Typical examples are the convection-diffusion or velocity-pressure as in the Navier–Stokes equations. For example, one can treat the convection part operator separate from the diffusion part, see, e.g., [17]. We mention also the variable density time-dependent Navier-Stokes equations, which include one more mass conservation equation. For recent results on operator splitting schemes for that equation, where second order discretization in time is guaranteed, see [22].

Nonetheless, we recommend not to use unnecessarily many splittings of the operators, but rather solve for instance the Navier–Stokes equation for velocity and pressure as a whole, using some efficient method for saddle point problems, briefly discussed in Section 6. As remarked previously, for widely different time and/or length scales it may, however, be efficient to use additional splittings.

In the numerical test problem section we test the combined time stepping and approximate Newton method, where the linearized operator is replaced with a simpler and better conditioned operator (matrix).

3 Some multiphysics models

We present now four examples of multiphysics models. The first three are described only shortly with the basic equations.

3.1 Fluid – solid structure interaction

Following [17], we let $\Omega \subset \mathcal{R}^d$, $d = 2, 3$ be a space region, which is filled with an incompressible fluid of density ρ_f and contains J moving rigid bodies, $B_1, B_2 \ldots B_J$. We assume that the only external force acting on this mixture is gravity (g). Then, between collisions (assuming that collisions take place), the fluid flow is modeled by the Navier–Stokes equations, for the velocity $\mathbf{u}$,

$$\begin{aligned} \rho_f[\tfrac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}] &= \rho_f \mathbf{g} + \nabla\cdot\sigma \qquad && \text{in}\, \Omega(t) \\ \nabla\cdot\mathbf{u} &= 0\,, && \text{in}\, \Omega_{(t)} \\ \mathbf{u}(\mathbf{x},0) &= \mathbf{u}_0(x) && \forall \mathbf{x} \in \Omega(t)\, \text{and}\ \ \nabla\cdot\mathbf{u}_0 = 0\,. \end{aligned}$$

Here $\Omega(t) = \Omega \setminus \bigcup_{j=1}^{J} \bar{B}_j(t)$.

The system has to be completed with proper boundary conditions, such as $\mathbf{u} = \mathbf{g}_0$ on Γ where $\int_\Gamma \mathbf{g}_0 \cdot \mathbf{n} d\,\Gamma = 0$. Here Γ is the outer boundary, $\Gamma = \partial\Omega$ and $\mathbf{n}$ denotes the unit normal vector pointing outward to the flow region.

One must also impose no–slip boundary conditions on the boundary ∂B_j of B_j, see [17] for details. Further, the stress–tensor, denoted here by σ, satisfies

$$\sigma = \tau - pI\,,$$

where p is the pressure and

$$\tau = \nu(\nabla\mathbf{u} + \nabla\mathbf{u}^T)$$

in the Newtonian case or τ is a nonlinear function of ∇u in the non–Newtonian case. Here ν denotes the viscosity of the fluid. The motion of the rigid bodies is modeled by the Newton–Euler equations. The equations involve the velocity of the center and the angular velocity of each mass.

Hence, this set of equations involves three velocities as well as the stress tensor and some additional functions, such as a second order evolutionary variable to model repulsive force, which takes place at collisions between bodies or collisions with the bottom of the fluid region, see [17] for details.

It is seen that several splittings must take place. In [17], even an advection-diffusion splitting has been used. Extensive numerical simulations are reported there, with computer run times up to more than a week.

3.2 A thermal stress, Joule heating problem

When an electric current flows through a device, it induces the so-called Joule heating of the device. The heating leads to thermal stresses which can be utilized to force the device to bend or move in a desired way. An application of this effect is to move gears and ratchet mechanisms (e.g., for mirror positioning) on a microscopic scale.

The model includes equations for the electric field, heat transfer and thermal stresses. For the electric field it holds

$$\begin{aligned} -\nabla \cdot \sigma(V)\nabla V &= 0 \qquad \text{in} \quad \Omega \\ \mathbf{n} \cdot \mathbf{J} &= 0 \\ \mathbf{J} &= \sigma \nabla V\,, \quad \text{on } \partial\Omega \setminus \partial\Omega_0\,, \end{aligned}$$

with suitable boundary conditions on a part $(\partial\Omega_0)$ of $\partial\Omega$. Here V is the electric potential, σ is the electric conductivity and J is the current.

The heat transfer is modelled by

$$\begin{aligned} -\nabla \cdot k\nabla T &= Q(\sigma, V) \quad \text{in } \Omega \\ \mathbf{n} \cdot k\nabla T &= h(T_{\text{inf}} - T) \quad \text{on } \partial\Omega \end{aligned}$$

where k is the thermal conductivity, Q is the resistance heating, h is a heat transfer coefficient and T is the external temperature. Typically

$$Q(\sigma, V) = \frac{\sigma}{\mid J \mid^2} = \sigma \mid \nabla V \mid^2, \quad \text{in } \Omega\,.$$

Finally, the thermal stresses are determined by certain, more standard relations. This problem involves, hence, three sets of partly nonlinear equations. An artificial time may be introduced to solve the arising equations via an (approximate) Newton method thereby introducing an operator splitting in, at least, the three sets. For a numerical study of the problem, see, e.g., [25].

3.3 An electrolytic cell problem

An interface problem arises in the modelling of industrial production of aluminium, which is based on an electrolytic process and produced from alumina (bauxite). This model gives raise to several coupled time-dependent differential equations. They include multi-fluid Navier–Stokes equations, electromagnetic (Maxwell's) equations and equations for the concentration, $c(\mathbf{x}, t)$ of the liquid alumina and n_p, the particle density of solid alumina. The latter is dissolved in the electrolytic process. The equations for c and n_p are similar to the equations for the concentration (c) and chemical potential (η) in the Cahn–Hilliard equation, to be described below. For further details, see [12, 16].

3.4 The Cahn–Hilliard interface model equation

Moving interface problems can be modelled by the Cahn–Hilliard equation (C-H), see, e.g. [1, 9, 10, 34]. In its original form, C-H is a fourth order nonlinear equation of parabolic type, defined in some bounded domain Ω. The variable to solve for is referred to as the *phase field*, the *order parameter* or the *concentration*, which represents the amount of certain substance in space at a given time. The concentration attains distinct values in the different substances. For example, for a binary fluid, it equals 1 in one of the phases, -1 on the other, and forms a smooth but steep transition front between them. It evolves with time due to diffusion or diffusion and convection processes.

In practice, in order to avoid dealing with fourth order derivatives, C-H is often reformulated as a coupled system of two equations with the concentration (c) and chemical potential (η) as main variable functions.

We follow here [34] and describe some details of the phase-field model. We assume that the chemical potential is given by the pointwise value in the following equations, due to van der Waals ([40]),

$$\eta = \frac{\partial \widetilde{F}}{\partial c}\,,$$

where $\widetilde{F}$ is the free energy, defined by

$$\widetilde{F}(c) = \int_\Omega (\beta \mid \nabla c \mid^2 + \alpha\psi(c))dx$$

Here $\beta \mid \nabla c \mid^2$ is the interfacial energy, which competes with the bulk free energy $\alpha\psi(c)$. The coefficients α, β are related to the surface tension coefficient σ and the interface width as $\alpha \sim \sigma/\varepsilon$ and $\beta \sim \sigma\varepsilon$. The free energy function ψ is normally assumed to have the form of a double–well potential,

$$\psi(c) = (c^2 - 1)^2\,.$$

Hence it holds

$$\psi'(c) = 4c(c^2 - 1)$$

and, taking $\alpha = 1/4$, we find

$$\eta = (c^3 - c) - \beta\Delta c\,.$$

The concentration is coupled to a diffusive term for the chemical potential. The equations take then the form

$$\left.\begin{aligned} \eta - \frac{d\psi}{dc} + \beta\nabla^2 c &= 0 \\ -\nabla\cdot(\kappa\nabla\eta) + \frac{\partial c}{\partial t} + (\mathbf{u}\cdot\nabla)c &= 0 \end{aligned}\right\} \; x \in \Omega,\; t > 0, \qquad (8)$$
$$\frac{\partial c}{\partial \mathbf{n}} = 0\,,\; \frac{\partial \eta}{\partial \mathbf{n}} = 0\,,\; x \in \partial\Omega\,,\; t > 0,$$

where $\mathbf{n}$ denotes the outward pointing normal vector. The problem parameter κ is the mobility and the coefficient β is proportional to the interface thickness. In dimensionless

form, β is proportional to the square of the so-called Cahn number Cn and κ is proportional to the inverse of the Peclet number Pe. An initial condition $c(x,0) = c_0(x)$, $x \in \Omega$ is assumed to be given.

Above, no influx, i.e., pure Neumann type boundary conditions are assumed for both variables. In general, the boundary conditions include the so-called wetting angle (cf. [10]), however details of that are not relevant for this work.

As is seen, the double–well potential has two minima, at $+1$ and -1, respectively, which corresponds to the two stable phases.

In general, the velocity field vector $\mathbf{u}$ varies with time and space, and is determined by solving the Navier–Stokes equations in its full complexity, namely, time-dependent with variable viscosity and variable density. A brief discussion on the numerical solution of those is included in Section 6.

The aim of the Cahn-Hilliard equation is to describe the interface dynamics of a system in which the final state is dominated by two (or more) phases, separated by a transition region. The goal is to prescribe as accurately as possible the evolution of the interface and the location of the phases, which appear after an initial stage of separation. The system is mass-conservative. (As remarked in [34], the model, as stated in (8), may not accurately capture any particular physical system, since effects of impurities, thermal effects, viscous and elastic effects are not accounted for. However, C-H is obtained as a limit equation of many more detailed models and therefore, studying and understanding its features is relevant.)

The coupled system (8)-(20) is nonlinear and involves five variables, η, c, $\mathbf{u}$, p and ρ. As already remarked, it is not viable to solve the whole, coupled nonlinear system by a Newton method. Instead, some form of an operator splitting method should be used, i.e., the equations (8) and (20) are split at each time step. In addition, another operator splitting scheme is applied when solving (20) itself. To get a sufficient accuracy, in particular a sufficiently small splitting error, one may then be forced to use many small time-steps $\tau_k = O(h^2)$, where h is a space mesh parameter.

On the other side, as discussed in Section 3, we advocate to use larger timesteps, in the order $\tau = O(h)$, and instead use an (approximate Jacobian) Newton iteration method during each time step. The straightforward approach is to split the coupled system in two parts, one in the form of the Cahn–Hilliard equation for the variables c and η, and one for the Navier–Stokes equation with main variables $\mathbf{u}$, p and ρ. Clearly, since the two sets are coupled, during each Newton step we update the velocity field $\mathbf{u}$, in the Cahn-Hilliard equation and then the computed c and η in the Navier-Stokes equation.

4 Numerical solution of the Cahn – Hilliard equation

Consider now the solution of the Cahn–Hilliard equation alone. Some comments on the solution of the stationary Navier–Stokes equations for the case of constant viscosity and constant density are presented in Section 6.

The solution of the Cahn-Hilliard equation comprises a time-stepping procedure, a

nonlinear problem to be solved during each time step using Newton's method in this case, and a solution of a linear system of algebraic equations with the Jacobian matrix (or some approximation of it), arising from the Newton iterations. To solve systems with the Jacobian, we use an inner–outer iteration procedure. The inner iterations are based on an idea in [7], see also [2] and [35], to solve complex valued systems, and is presented in Section 5.

4.1 Iterative solution of the Cahn-Hilliard equation

We discretize the Cahn–Hilliard equation by a standard finite element method. At every Newton linearization step during each timestep a matrix in the form

$$\mathcal{C} = \begin{bmatrix} M & -\beta K - J \\ \tau\kappa K & M + \tau W \end{bmatrix} \tag{9}$$

then arises. Here, the Jacobian matrix term J is derived from the nonlinear term $d\,\psi/d\,c$ in (8), M, K and W denote in order, the Gramian mass matrix, the stiffness, Laplacian operator matrix and the advection term matrix, respectively, all corresponding to the given finite element mesh. In general, the timestep $\tau = \tau_k$ is variable, but for notational simplicity we keep the notation τ without a time-step number index.

We split the matrix $\mathcal{C}$ as

$$\mathcal{C} = \mathcal{C}_0 + \begin{bmatrix} 0 & -J \\ 0 & \tau W \end{bmatrix},$$

where

$$\mathcal{C}_0 = \begin{bmatrix} M & -\widetilde{K} \\ \widehat{\beta}^2\widetilde{K} & M \end{bmatrix}, \quad \widetilde{K} = \beta K \text{ and } \widehat{\beta}^2 = \frac{\tau\kappa}{\beta}. \tag{10}$$

The matrix equation $\mathcal{C}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}$ is then solved by a generalized conjugate gradient or minimal residual method with $\mathcal{C}_0$ as preconditioner. Note that if the coefficients κ and β are constant during each time step, then there is no need to update the preconditioner $\mathcal{C}_0$. This holds also in a larger time–interval, if we choose constant timesteps in that interval.

Proposition 4.1 *Let C, C_0 be defined in (9) and (10), respectively. The preconditioned matrix satisfies*

$$\mathcal{C}_0^{-1}\mathcal{C} = I + \begin{bmatrix} 0 & (-S^{-1}J + \tau M^{-1}\widetilde{K}S^{-1}W \\ 0 & (\widetilde{\beta}^2 M^{-1}\widetilde{K}S^{-1}J + \tau S^{-1}W) \end{bmatrix}$$

where $S = M + \widetilde{\beta}^2\widetilde{K}M^{-1}\widetilde{K}$, $\widetilde{K} = \beta K$, and $\widetilde{\beta}^2 = \frac{\tau\kappa}{\beta}$.
It holds

$$\|\tau M^{-1}\widetilde{K}S^{-1}W\| \le \frac{\tau}{2\widetilde{\beta}} = O(\tau^{1/2}),\ \tau \to 0\,.$$

$$\|\widetilde{\beta}^2 M^{-1}\widetilde{K}S^{-1}J\| \le \frac{3}{2}\widetilde{\beta} = O(\tau^{1/2}),\ \tau \to 0\,.$$

and

$$\|\tau S^{-1}W\| \leq \tau \|M^{-1}W\| \,.$$

Since

$$\lim_{\tau \to 0} \mathcal{C}_0^{-1} C = \begin{bmatrix} 0 & S^{-1}J \\ 0 & 0 \end{bmatrix} ,$$

it follows that for sufficiently small values of τ *a generalized preconditioned conjugate gradient method for the preconditioned matrix* $\mathcal{C}_0^{-1}C$ *converges in two or three steps.*

Proof: It holds

$$\begin{aligned} \mathcal{C}_0^{-1} \begin{bmatrix} 0 & -J \\ 0 & \tau W \end{bmatrix} &= \begin{bmatrix} S^{-1} & M^{-1}\widetilde{K}S^{-1} \\ -\widehat{\beta}^2 M^{-1}\widetilde{K}S^{-1} & S^{-1} \end{bmatrix} \begin{bmatrix} 0 & -J \\ 0 & \tau W \end{bmatrix} \\ &= \begin{bmatrix} 0 & (-S^{-1}J + \tau M^{-1}\widetilde{K}S^{-1}W) \\ 0 & (\widetilde{\beta}^2 M^{-1}\widehat{K}S^{-1}J + \tau S^{-1}W) \end{bmatrix} , \end{aligned} \tag{11}$$

where $S = M + \widetilde{\beta}^2 \widetilde{K} M^{-1} \widetilde{K}$. To analyse the rate of convergence of the iteration method we rewrite the matrix $M^{-1}\widetilde{K}S^{-1}$ in the form $(M^{-1}\widetilde{K}S^{-1}M)M^{-1}$ and make a similarity transformation of the first factor,

$$M^{1/2}(M^{-1}\widetilde{K}S^{-1}M)M^{-1/2} = \widehat{K}(I + \widetilde{\beta}^2 \widehat{K}^2)^{-1} \,,$$

where $\widehat{K} = M^{-1/2}\widetilde{K}M^{-1/2}$.

Since $|a| \diagup (1 + \widetilde{\beta}^2 a^2) \leq 1/(2\widetilde{\beta})$, where a is a real number, it follows that

$$\|\tau M^{-1}\widetilde{K}S^{-1}M\| \leq \frac{\tau}{2\widetilde{\beta}} = O(\tau^{1/2}),\ \tau \to 0$$

and

$$\|\widetilde{\beta}^2 M^{-1}\widetilde{K}S^{-1}M\| \leq \frac{1}{2}\widetilde{\beta} = O(\tau^{1/2}),\ \tau \to 0 \,.$$

Since $\|\tau S^{-1}W\| = \|\tau S^{-1}M\, M^{-1}W\| \leq \tau \|M^{-1}W\|$, it follows then that all terms, except $S^{-1}J$, in (11) decay at least as $O(\tau^{1/2})$ when $\tau \to 0$. Therefore, for sufficiency small timesteps, the iteration method converges approximately with an error matrix as $\begin{bmatrix} 0 & S^{-1}J \\ 0 & 0 \end{bmatrix}$, i.e. in two iteration steps. Clearly, for larger values of τ , there will be some additional iterations but their number can be controlled by choosing τ sufficiently small. ■

Remark 2 The constants involved in the upper bounds in Proposition 4.1 equal the inverses of each other, because $\frac{\tau}{\widetilde{\beta}^2} = O((\beta/\kappa)^{1/2})\tau^{1/2}$ and $\widetilde{\beta}^2 = O((\kappa/\beta)^{1/2})\tau^{1/2}$, respectively. Here $\beta/\kappa = \sigma\varepsilon Pe$, where Pe is the Peclet number. To get a bound valid for all thicknesses and Peclet numbers, we must therefore choose.

$$\tau \ll \min\{\beta/\kappa,\, \kappa/\beta\} = \min\{\sigma\varepsilon Pe,\, 1/(\sigma\varepsilon Pe)\}$$

This means that if $Pe \cong O(1/\sigma\varepsilon)$, we can choose τ independently of these parameters, but if $Pe \ll O(1/\sigma\varepsilon)$, then we must choose τ sufficiently small.

In practice, in order to resolve the interface profile numerically, we choose the space mesh parameter h such that $h = \beta/r$, where, typically, r is taken as a number, $5 \leq r \leq 10$. Then, if $Pe \cong O(\beta^{-1})$, we can choose the timestep τ such that the timestepping discretization error is of the same order as the space discretization error, which normally means that we take $\tau = O(h)$.

Such large Peclet numbers hold for convection dominated flows. If, however, the convection is small or even zero, so that $Pe \approx 1$, say, then we must choose

$$\tau \ll \min\{\sigma\varepsilon, 1/(\sigma\varepsilon)\} = \sigma\varepsilon\,, \text{ i.e.,}$$

τ must be taken significantly smaller than $\varepsilon = O(h)$, to avoid a slow rate of convergence of the iterative solution method. We note namely that the troublesome term in the preconditioned matrix arises in its 2,2–block, i.e., it directly influences the eigenvalues of the matrix. This is also clearly seen from the numerical tests.

Finally, we note that for finite element discretized operators one must balance τ with respect to the space discretization parameter η to get $\tau\|M^{-1}W\|$ bounded for all values of h. Since W is a first order (convection) term, we should choose $\tau = O(h)$. Numerical tests indicate that already $\tau = \frac{1}{2}h$ can be efficient.

4.2 Inner iterations

To solve the systems with the preconditions $\mathcal{C}_0$ which, as outlined above, occurs at few iteration steps, we use inner iterations. The preconditioner for the inner iterations is the matrix,

$$\mathcal{B}_0 = \begin{bmatrix} M & -\widetilde{K} \\ \widetilde{\beta}^2\widetilde{K} & M + 2\widetilde{\beta}\widetilde{K} \end{bmatrix},$$

which can be factorized as

$$\mathcal{B}_0 = \begin{bmatrix} M & 0 \\ \widetilde{\beta}^2\widetilde{K} & M + \widetilde{\beta}\widetilde{K} \end{bmatrix} \begin{bmatrix} I & -M^{-1}\widetilde{K} \\ 0 & M^{-1}(M + \widetilde{\beta}\widetilde{K}) \end{bmatrix}, \tag{12}$$

As is shown in Section 5, the eigenvalues of $\mathcal{B}_0^{-1}\mathcal{C}_0$ are contained in the interval $[0.5, 1]$ or even in a smaller interval. Therefore, a conjugate gradient method to solve the arising inner iteration systems with matrix $\mathcal{C}_0$ will also converge rapidly.

As is shown in Section 5, each inner iteration requires just two solutions with the matrix $M + \widetilde{\beta}\widetilde{K} = M + \sqrt{\tau\kappa/\beta}K$.

4.3 An approximate Newton iteration method

The above works well only for not too large timesteps. The reason is that the Newton iterations use a matrix $\mathcal{C}$ which can be singular, or even indefinite. This is seen from the Schur complement $S(\mathcal{C})$ of $\mathcal{C}$,

$$\begin{aligned} S(\mathcal{C}) &= M + \tau W + \tau\kappa K M^{-1}(\beta K + J) \\ &= M + \tau W + \tau\kappa\beta K M^{-1} K + \tau\kappa K M^{-1} J \end{aligned} \tag{13}$$

Normally the advection operator matrix W does not cause any problem, at least not for timesteps $\tau \leq O(h)$, where h is the mesh parameter for the FEM method. The last term in (13), however, may cause a loss of positive definiteness, since J is an indefinite matrix. This may cause divergence or bifurcation of the Newton iterations. Since $\|KM^{-1}\| = O(h^{-2})$, this holds in particular for sufficiently small stepsizes h. Therefore, we may be forced to choose $\tau = O(h^2)$, which is clearly undesirable.

For the problem under consideration, it is therefore better to use an approximate Newton method, where we use $\mathcal{C}_0$ as a replacement of $\mathcal{C}$ in the nonlinear iteration solver steps. The approximate Newton method takes then the form

$$\mathcal{C}_0(x^{(k+1)} - x^{(k)}) = \xi_k \mathcal{A}(t_k, x^{(k)}),\ k = 0, 1, \cdots$$

for an implicit Euler method and a similar form for a Crank–Nicholson or a $\theta-$method. Here $\xi_k \leq 1$, is a damping parameter. Since $\mathcal{A} = C_0 + \begin{bmatrix} 0 & -J \\ 0 & \tau W \end{bmatrix}$, so

$\|C_0^{-1}(\mathcal{A} - C_0)\| \leq \left\| \begin{bmatrix} 0 & -S^{-1}J \\ 0 & 0 \end{bmatrix} \right\| = O(\tau^{1/2})$,

it can even happen that one can choose full length steps with $\xi_k = 1$.

5 Application of a preconditioner for complex valued systems

To analyse the preconditioner $\mathcal{B}_0$ defined in the previous section, we present first an analysis for general matrices on a similar form. To solve a complex valued system

$$(A + iB)(x + iy) = (f + ig) \tag{14}$$

it was suggested in [7] to rewrite (14) in the form

$$\begin{bmatrix} A & -B \\ B & A \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix},$$

whereby one avoids the use of complex number arithmetics. We assume first that A is symmetric and positive definite (spd). In the simplest form, the matrix $(A+B)A^{-1}(A+B)$ was used as an approximation of the Schur complement matrix $A + BA^{-1}B$, see [2, 3]. Here we generalize the method to solve systems with a matrix in the form

$$\mathcal{A} = \begin{bmatrix} A & -B^T \\ \beta^2 B & \alpha^2 A \end{bmatrix}, \tag{15}$$

where α, β are positive numbers. (These have no relation to the parameters α, β used in the Cahn–Hilliard equation.) Clearly, A and B have the same order.

As a preconditioner to $\mathcal{A}$ we choose

$$\mathcal{B} = \begin{bmatrix} A & 0 \\ \beta^2 B & (\alpha A + \beta B) \end{bmatrix} \begin{bmatrix} A^{-1} & 0 \\ 0 & A^{-1} \end{bmatrix} \begin{bmatrix} A & -B^T \\ 0 & (\alpha A + \beta B^T) \end{bmatrix} . \tag{16}$$

It holds

$$\mathcal{B} = \begin{bmatrix} A & 0 \\ \beta^2 B & \alpha A + \beta B \end{bmatrix} \begin{bmatrix} I & -A^{-1}B^T \\ 0 & \alpha I + \beta A^{-1} B^T \end{bmatrix}$$

so

$$\mathcal{B} = \begin{bmatrix} A & -B^T \\ \beta^2 B & \alpha^2 A + \alpha\beta(B + B^T) \end{bmatrix}$$

or

$$\mathcal{B} = \mathcal{A} + \begin{bmatrix} 0 & 0 \\ 0 & \alpha\beta(B + B^T) \end{bmatrix}$$

We show now first that $\mathcal{B}^{-1}$ can be written in such a form that each action of it on a vector only requires one solution with matrices $\alpha A + \beta B$ and $\alpha A + \beta B^T$ and that no other systems must be solved.

Using (16) we find

$$\mathcal{B} = \begin{bmatrix} I & 0 \\ \beta^2 B A^{-1} & (\alpha I + \beta B A^{-1}) \end{bmatrix} \begin{bmatrix} A & 0 \\ 0 & A \end{bmatrix} \begin{bmatrix} I & -A^{-1}B^T \\ 0 & (\alpha I + \beta A^{-1} B^T) \end{bmatrix} .$$

Hence,

$$\begin{aligned} \mathcal{B}^{-1} &= \begin{bmatrix} I & (\alpha I + \beta A^{-1}B^T)^{-1} A^{-1} B^T \\ 0 & (\alpha I + \beta A^{-1} B^T)^{-1} \end{bmatrix} \begin{bmatrix} A^{-1} & 0 \\ 0 & A^{-1} \end{bmatrix} \times \\ &\qquad \begin{bmatrix} I & 0 \\ -\beta^2 B A^{-1} (\alpha I + \beta B A^{-1})^{-1} & (\alpha I + \beta B A^{-1})^{-1} \end{bmatrix} \\ &= \begin{bmatrix} A^{-1} & \frac{1}{\beta}\left[I - \alpha(\alpha I + \beta A^{-1}B^T)^{-1}\right] A^{-1} \\ 0 & (\alpha A + \beta B^T)^{-1} \end{bmatrix} \times \\ &\qquad \begin{bmatrix} I & 0 \\ -\beta[I - \alpha(\alpha I + \beta B A^{-1})^{-1} & A(\alpha A + \beta B)^{-1} \end{bmatrix} \\ &= \begin{bmatrix} \alpha(\alpha A + \beta B^T)^{-1}[I - \alpha A(\alpha A + \beta B^T)^{-1}] & \frac{1}{\beta}[I - \alpha(\alpha A + \beta B^T)^{-1}] A(\alpha A + \beta B)^{-1} \\ \qquad + \alpha(\alpha A + \beta B)^{-1} & \\ -\beta(\alpha A + \beta B^T)^{-1}[I - \alpha A(\alpha A + \beta B)^{-1}] & (\alpha A + \beta B^T)^{-1} A(\alpha A + \beta B)^{-1} \end{bmatrix} \end{aligned}$$

Multiplying a vector $\begin{bmatrix} f \\ g \end{bmatrix}$ with $\mathcal{B}^{-1}$ we find

$$\mathcal{B}^{-1} \begin{bmatrix} f \\ g \end{bmatrix} = \begin{bmatrix} \alpha(\alpha A + \beta B^T)^{-1}(f - \alpha f^{(1)}) + \alpha f^{(1)} + \frac{1}{\beta}[I - \alpha(\alpha A + \beta B^T)^{-1}] g^{(1)} \\ -\beta(\alpha A + \beta B^T)^{-1}(f - \alpha f^{(1)}) + (\alpha A + \beta B^T)^{-1} g^{(1)} \end{bmatrix}$$

where

$$f^{(1)} = A(\alpha A + \beta B)^{-1} f\,, \quad g^{(1)} = A(\alpha A + \beta B)^{-1} g$$

Hence,

$$\begin{aligned} \mathcal{B}^{-1} \begin{bmatrix} f \\ g \end{bmatrix} &= \begin{bmatrix} \frac{1}{\beta}[\alpha(\alpha A + \beta B^T)^{-1}(\beta(f - \alpha f^{(1)}) - g^{(1)}) + \beta\alpha f^{(1)} + g^{(1)}] \\ -(\alpha A + \beta B^T)^{-1}(\beta(f - \alpha f^{(1)}) - g^{(1)}) \end{bmatrix} \\ &= \begin{bmatrix} \frac{\alpha}{\beta}(\alpha A + \beta B^T)^{-1}(\beta f - g^{(2)}) + g^{(2)} \\ -(\alpha A + \beta B^T)^{-1}(\beta(f - g^{(2)}) \end{bmatrix} = \begin{bmatrix} \frac{\alpha}{\beta} g^{(3)} + g^{(2)} \\ -g^{(3)} \end{bmatrix} \end{aligned}$$

where $g^{(2)} = \alpha\beta f^{(1)} + g^{(1)} = A(\alpha A + \beta B^T)^{-1}(\alpha\beta f + g)$ and $g^{(3)} = (\alpha A + \beta B^T)^{-1}(\beta f - g^{(2)})$. Therefore, besides matrix–vector multiplications (with normally sparse matrices) and vector additions, only one action of $(\alpha A + \beta B)^{-1}$ is involved, to compute $g^{(2)}$ and one action of $(\alpha A + \beta B^T)^{-1}$ to compute $g^{(3)}$ and the final vector.

We now analyse the eigenvalues of the preconditioned matrix.

Proposition 5.1 *Assume that A is spd and the symmetric part of B is positive semidefinite. The relation*

$$\mathcal{B} = \mathcal{A} + \begin{bmatrix} 0 & 0 \\ 0 & \alpha\beta(B + B^T) \end{bmatrix},$$

where $\alpha > 0$, shows that the eigenvalues of the generalized eigenvalue problem,

$$\lambda \mathcal{B} \begin{bmatrix} x \\ y \end{bmatrix} = \mathcal{A} \begin{bmatrix} x \\ y \end{bmatrix}$$

satisfy

$$\lambda = 1 \quad \textit{for eigenvectors} \quad y \in \mathcal{N}(B + B^T), x \in \mathcal{R}^n$$

and all eigenvalues are contained in the interval $[0.5, 1]$.

Proof: If $\mathcal{A} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, then

$$Ax = B^T y \qquad \text{and}$$

$$\beta^2 Bx + \alpha^2 Ay = 0$$

so

$$(\alpha^2 A + \beta^2 B A^{-1} B^T) y = 0\,,$$

which implies $y = 0$ and, hence, $x = 0$. Therefore $\lambda \neq 0$.
It holds,

$$(\lambda - 1)\mathcal{B} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ -\alpha\beta(B + B^T) y \end{bmatrix}$$

so $\lambda = 1$ for $y \in \mathcal{N}(B + B^T)$.
If $\lambda \neq 1$, then

$$Ax - B^T y = 0$$

and a computation shows that

$$(\lambda - 1)(\beta^2 Bx + \left(\alpha^2 A + \alpha\beta(B + B^T)\right) y = -\alpha\beta(B + B^T)y$$

Hence $x = A^{-1}B^T y$ and

$$(\lambda - 1)\left(\beta^2 BA^{-1}B^T + \alpha^2 A + \alpha\beta(B + B^T)\right) y = -\alpha\beta(B + B^T)y\,,$$

that is,

$$(\lambda - 1)\left(\alpha^2 A + \beta^2 BA^{-1}B^T\right) y = -\lambda\alpha\beta(B + B^T)y$$

or

$$\left(\frac{1}{\lambda} - 1\right)\left(\alpha^2 A + \beta^2 BA^{-1}B^T\right) y = \alpha\beta(B + B^T)y\,.$$

It follows

$$\left(\frac{1}{\lambda} - 1\right)\left(I + \widetilde{B}\widetilde{B}^T\right)\widetilde{y} = (\widetilde{B} + \widetilde{B}^T)\widetilde{y}\,,$$

where $\widetilde{B} = \frac{\beta}{\alpha}A^{-1/2}BA^{-1/2}$, $\widetilde{y} = A^{1/2}y$.

Hence

$$\frac{1}{\lambda} - 1 = \frac{\widetilde{y}^T(\widetilde{B} + \widetilde{B}^T)\widetilde{y}}{\widetilde{y}^T\widetilde{y} + (\widetilde{B}^T\widetilde{y}, \widetilde{B}^T\widetilde{y})}\,.$$

By assumption, $\widetilde{y}^T(\widetilde{B} + \widehat{B}^T)\widetilde{y} \geq 0$. Hence, by Cauchy–Schwarz inequality,

$$0 \leq \widetilde{y}^T(\widetilde{B} + \widetilde{B}^T)\widetilde{y} = (\widetilde{B}^T\widetilde{y}, \widetilde{y}) + (\widetilde{y}^T, \widetilde{B}^T\widetilde{y}) \leq \widetilde{y}^T\widetilde{y} + (\widetilde{B}^T\widetilde{y}, \widetilde{B}^T\widetilde{y})$$

so

$$0 \leq \frac{1}{\lambda} - 1 \leq 1$$

that is,

$$1 \leq \frac{1}{\lambda} \leq 2\,.$$

■

It has been shown that solutions with the preconditioner for systems with a matrix of the form (15) involve only solutions with matrices $\alpha A + \beta B$ and $\alpha A + \beta B^T$. In the application of this method for the Cahn–Hilliard equations these matrices are combinations of mass and Laplace operator matrices, and can therefore normally be solved efficiently by some preconditioned iterative solution method, or sometimes even by a direct solution method. Note that they correspond to homogeneous coefficients as there is no strong variation of the coefficients in these matrices.

To improve the preconditioner further, we let now the preconditioner depend on a parameter. Thereby we replace the problem parameter α in (16) with a variable parameter $\widetilde{\alpha}$, to get

$$\begin{aligned}\mathcal{B} &= \begin{bmatrix} A & 0 \\ \beta^2 B & \widetilde{\alpha}A + \beta B \end{bmatrix} \begin{bmatrix} A^{-1} & 0 \\ 0 & A^{-1} \end{bmatrix} \begin{bmatrix} A & -B^T \\ 0 & \widetilde{\alpha}A + \beta B^T \end{bmatrix} \\ &= \begin{bmatrix} A & -B^T \\ \beta^2 B & \widetilde{\alpha}A + \widetilde{\alpha}\beta(B + B^T) \end{bmatrix} \\ &= \mathcal{A} + \begin{bmatrix} 0 & 0 \\ 0 & (\widetilde{\alpha}^2 - \alpha^2)A + \widetilde{\alpha}\beta(B + B^T) \end{bmatrix}\end{aligned} \tag{17}$$

Clearly, this preconditioner can be implemented in the same way as (16), hence each action of $\mathcal{B}^{-1}$ on a vector requires only a solution with matrices $(\widetilde{\alpha}A + \beta B)$ and $(\widetilde{\alpha}A + \beta B^T)$. For clarity, we state this result as

Proposition 5.2 *The actions* $\mathcal{B}^{-1} \begin{bmatrix} f \\ g \end{bmatrix}$ *of the inverse of the preconditioner* $\mathcal{B}$, *defined in (17) can be implemented as*

$$\mathcal{B}^{-1} \begin{bmatrix} f \\ g \end{bmatrix} = \begin{bmatrix} \frac{\alpha}{\beta} g^{(3)} + g^{(2)} \\ -g^{(3)} \end{bmatrix},$$

where

$$\begin{aligned} g^{(2)} &= A(\alpha A + \beta B^T)^{-1}(\widetilde{\alpha}\beta f + g), \\ g^{(3)} &= (\alpha A + \beta B)^{-1}(\beta f - g^{(2)}). \end{aligned}$$

Proof: See the previous derivation after (16). ∎

The generalized eigenvalue problem

$$\lambda \mathcal{B} \begin{bmatrix} x \\ y \end{bmatrix} = \mathcal{A} \begin{bmatrix} x \\ y \end{bmatrix},$$

where $\lambda \neq 0$, gives

$$\left(\frac{1}{\lambda} - 1\right) \mathcal{A} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ \left[(\widetilde{\alpha}^2 - \alpha^2)A + \widetilde{\alpha}\beta(B + B^T)\right] y \end{bmatrix}$$

or, using $Ax = B^T y$,

$$\left(\frac{1}{\lambda} - 1\right) (\beta^2 B A^{-1} B^T + \alpha^2 A) y = \left[(\widetilde{\alpha}^2 - \alpha^2)A + \widetilde{\alpha}\beta(B + B^T)\right] y \tag{18}$$

Assuming that A is *spd* and B is *spsd* (symmetric positive semidefinite) and letting

$$\widetilde{B} = \beta A^{-1/2} B A^{-1/2}, \; \widetilde{y} = A^{1/2} y,$$

we find

$$\left(\frac{1}{\lambda}-1\right)(\widetilde{B}\widetilde{B}^T+\alpha^2 I)\widetilde{y}=\left[(\widetilde{\alpha}^2-\alpha^2)I+\widetilde{\alpha}(\widetilde{B}+\widetilde{B}^T)\right]\widetilde{y}\,,$$

or

$$\frac{1}{\lambda}(\alpha^2 I+\widetilde{B}\widetilde{B}^T)\widetilde{y}=[\widetilde{\alpha}^2 I+\widetilde{B}\widetilde{B}^T+\widetilde{\alpha}(\widetilde{B}+\widetilde{B}^T)]y\,.$$

Letting $\widetilde{B}\widetilde{y}=\mu\widetilde{y}$, where μ denotes an eigenvalue of $\widetilde{B}$, it follows that

$$\lambda=\lambda(\mu)=\frac{\alpha^2+\mu^2}{\widetilde{\alpha}^2+\mu^2+2\widetilde{\alpha}\mu}=\frac{\alpha^2+\mu^2}{(\widetilde{\alpha}+\mu)^2}\,. \tag{19}$$

We want to find the optimal value of $\widetilde{\alpha}$ to minimize the spectral condition number of $\mathcal{B}^{-1}\mathcal{A}$.

Proposition 5.3 *Let $\mathcal{B}$ be defined in (17) and assume that A is symmetric and positive definite and B is symmetric and positive semidefinite. Let $\mu_{\max}$ be the maximal eigenvalue of $\beta A^{-1}B$ and assume that $\mu_{\max}\geq\alpha^2/\widetilde{\alpha}$. Then the optimal value of $\widetilde{\alpha}$ to minimize the spectral condition number $\varkappa$ of $\mathcal{B}^{-1}\mathcal{A}$ equals*

$$\widetilde{\alpha}_{opt}=\alpha\left[\left(\frac{\alpha}{\mu_{\max}}+\sqrt{(\frac{\alpha}{\mu_{\max}})^2+1}\right)\right]$$

and

$$\min_{\widetilde{\alpha}}\varkappa(\mathcal{B}^{-1}\mathcal{A})=1+(\alpha/\widetilde{\alpha}_{opt})^2\,.$$

Proof: It holds $\lambda(0)=(\alpha/\widetilde{\alpha})^2$ and $\lim_{\mu\to\infty}\lambda(\mu)\leq 1$. To find the extreme value of $\lambda(\mu)$ in (19), we see that

$$\lambda'(\mu)=\frac{2}{(\widetilde{\alpha}+\mu)^3}(\widetilde{\alpha}\mu-\alpha^2)=0$$

gives

$$\lambda_{\min}=\lambda(\alpha^2/\widetilde{\alpha})=\frac{\alpha^2+\alpha^4/\widetilde{\alpha}^2}{(\widetilde{\alpha}+\alpha^2/\widetilde{\alpha})^2}=\frac{1}{1+(\widetilde{\alpha}/\alpha)^2}\,.$$

We have $\mu\in[0,\mu_{\max}]$, where $\mu_{\max}\geq\alpha^2/\widetilde{\alpha}$. It holds $\lambda(0)=(\alpha/\widetilde{\alpha})^2$ and $\lambda(0)\geq\lambda(\alpha^2/\widetilde{\alpha})$. Since $\varkappa(\mathcal{B}^{-1}\mathcal{A})=2$ if $\widetilde{\alpha}=\alpha$, it follows that we must choose $\widetilde{\alpha}>\alpha$ to make $\varkappa(\mathcal{B}^{-1}\mathcal{A})<2$. The variation of the function $\lambda(\mu)$ is shown in Figure 1.

To minimize the condition number we choose $\widetilde{\alpha}$ such that

$$\left(\frac{\alpha}{\widetilde{\alpha}}\right)^2=\lambda(\mu_{\max})=\frac{\alpha^2+\mu_{\max}^2}{(\widetilde{\alpha}+\mu_{\max})^2}$$

which, after simplification, gives

$$\frac{\widetilde{\alpha}}{\alpha}=\frac{\alpha}{\mu_{\max}}+\left\{(\frac{\alpha}{\mu_{\max}})^2+1\right\}^{1/2},\quad\text{where }\widetilde{\alpha}=\widetilde{\alpha}_{opt}\,.$$

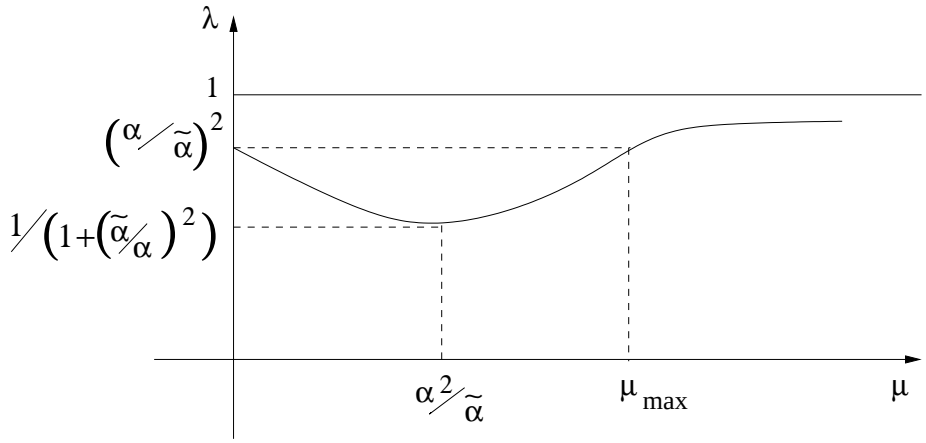


Figure 1: $\lambda(\mu) = (\alpha^2 + \mu^2)/(\widetilde{\alpha} + \mu)^2$

Hence

$$\min_{\widetilde{\alpha}} \varkappa(\mathcal{B}^{-1}A) = (\alpha/\widetilde{\alpha})^2(1 + (\widetilde{\alpha}/\alpha)^2) = 1 + (\alpha/\widetilde{\alpha}_{opt})^2 \,.$$

■

Remark 3 If $\mu_{\max}/\alpha \to \infty$, then $\varkappa(\mathcal{B}^{-1}\mathcal{A}) \to 2$ and if $\mu_{\max} = \alpha$, then $\varkappa(\mathcal{B}^{-1}\mathcal{A}) = \sqrt{2}$.

6 Solving the variable viscosity, variable density time-dependent incompressible Navier-Stokes equations

The time-dependent Navier-Stokes equations for flows with variable viscosity and density read as follows:

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) &= 0 \\ \rho \left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} \right) - \nabla \cdot [\nu(\nabla \mathbf{u} + \nabla \mathbf{u}^T)] + \nabla p &= \mathbf{F} - \eta \nabla c \\ \nabla \cdot \mathbf{u} &= 0 \end{aligned} \tag{20}$$

with typical boundary and initial conditions

$$\begin{aligned} &\rho(\mathbf{x}, 0) = \rho_0, \quad \rho(\mathbf{x}, t)\,|_{\Gamma^-} = r(\mathbf{x}, t) \\ &\mathbf{u}(\mathbf{x}, 0) = \mathbf{u}_0, \quad \mathbf{u}(\mathbf{x}, t)\,|_{\partial\Omega} = \mathbf{g} \end{aligned} \tag{21}$$

Here ρ, ν and p denote the density, viscosity and the pressure of the fluid. $\mathbf{F}$ is the volume force density and $\Gamma^- = \{\mathbf{x} \in \partial\Omega : \mathbf{u}(\mathbf{x}) \cdot \mathbf{n} < 0\}$ is the inflow boundary.

As noted in [22], there are quite a few papers, dealing with the above problem, the major difficulty being the fact that the equations possess elliptic, parabolic and hyperbolic

properties. The existing work addresses mainly various splitting and projection schemes, their stability and accuracy, see the the founding work by Chorin (1968) and Themam (1969), see, e.g., [22] and the references therein. Up to the knowledge of the authors, there are also few studies of the numerical solution aspects of the solution of the discrete algebraic systems, arising from (20). We mention as an example, two papers, where preconditioning techniques for N-S problem with variable density are studied - [32] and [30]. Close to the raised topic is the work in [21], where variable viscosity is present in the model, however the stationary equations are discussed and the convective term is handled as a part of the right-hand side, thus the problem is more like Stokes with variable viscosity. In [24], the stationary Navier-Stokes equation, including the convection term, with variable density is analysed, showing that the preconditioners for the stationary Navier-Stokes equations, linearized using Picard iterations, are straightforwardly applicable to the case of variable viscosity.

We sketch below a state-of-art splitting method for the time-dependent, variable density incompressible Navier-Stokes equations, which is derived in [22], for which a second order splitting error is shown. The operator splitting scheme is of incremental type and consists of the following four steps:
Set $\rho^{(0)} = \rho_0$, $\mathbf{u}^{(0)}\mathbf{u}_0$, $p^{(0)} = p_0$ (to be deduced from ρ_0 and $\mathbf{u}_0$) and $\phi^{(0)} = 0$.
For $n = 1, \cdots$ compute

$$\frac{\rho^{(n+1)} - \rho^{(n)}}{\tau} + \nabla \cdot (\rho^{(n+1)}\mathbf{u}^{(n)}) - \frac{\rho^{(n+1)}}{2}\nabla \cdot \mathbf{u}^{(n)} = 0 \tag{22}$$

Update velocity as

$$\begin{aligned} &\frac{1}{\tau}\left[\frac{1}{2}(\rho^{(n+1)} + \rho^{(n)})\mathbf{u}^{(n+1)} - \rho^{(n)}\mathbf{u}^{(n)}\right] + \rho^{(n+1)}\mathbf{u}^{(n)} \cdot \mathbf{u}^{(n+1)} \\ &+\frac{1}{2}\nabla \cdot (\rho^{(n+1)}\mathbf{u}^{(n)})\mathbf{u}^{(n+1)} - \nu\Delta\mathbf{u}^{(n+1)} + \nabla(p^{(n)} - \phi^{(n)}) = \mathbf{f}^{(n+1)} \\ &\mathbf{u}^{(n+1)}\,|_{\partial\Omega} = 0 \end{aligned} \tag{23}$$

Next, update the auxiliary variable ϕ:

$$\begin{aligned} &\Delta\phi^{(n+1)} = \frac{\chi}{\tau}\nabla \cdot \mathbf{u}^{(n+1)} \\ &\partial_n\phi^{(n+1)}\,|_{\partial\Omega} = 0 \end{aligned} \tag{24}$$

and, finally, update the pressure as

$$p^{(n+1)} = p^{(n)} + \phi^{(n+1)} \tag{25}$$

Above, $\chi \in [0, min(\rho_0)]$ and in practice, the usual choice is $\chi = min(\rho_0)$. For more details and a higher-order time integration scheme, see [22].

As can be observed, in the operator splitting scheme (22)-(25), no solution of a saddle point system is involved. It can be reintroduced by unifying steps (23) and (24). Then,

the saddle point system to be solved, has a nonzero block in its (2,2) position, which corresponds to a stabilized stationary Navier-Stokes formulation. It is worth mentioning the comment, made in [22], that still, stable LBB discretization spaces for velocity and pressure have to be used.

Below we briefly state some results, relevant to the iterative solution of saddle point problems.

6.1 Preconditionings for saddle point problems

Saddle point matrices are special cases of matrices in two-by-two block form $A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$ that arise in various applications. If $A_{22} = 0$, then A is a saddle point matrix. We assume that A_{11} is nonsingular and $S_2(A) = A_{22} - A_{21}A_{11}^{-1}A_{12}$ (the Schur complement) is nonsingular. Following [6], we give here first an outline of the analysis of a general, approximate block matrix factorization of A and apply then directly this result for saddle point matrices.

A general form of an approximate block factorization of the matrix A takes the form

$$C = \begin{bmatrix} I_1 & 0 \\ A_{21}C_{11} & I_2 \end{bmatrix} \begin{bmatrix} \widetilde{A}_{11} & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I_1 & B_{11}A_{12} \\ 0 & I_2 \end{bmatrix} . \tag{26}$$

Here B_{11} and C_{11} are approximations of A_{11}^{-1} (normally sparse and given on explicit form) while $\widetilde{A}_{11}^{-1}$ denotes some approximation of A_{11}^{-1}, possibly implicitly defined via inner iterations. Further S is a nonsingular approximation of S_2.

An analysis of the above approach with details regarding saddle-point matrices can be found in [8]. Below we briefly summarize the main conclusions.

Since we can use inner iterations for the arising systems with A_{11} and S_2, it is of interest to consider the case where we let $\widetilde{A}_{11} = A_{11}$ and $S = S_2$. The following result has been shown in [6].

Proposition 6.1 *Let C be defined by (26), where $\widetilde{A}_{11} = A_{11}$ and $S = S_2$. Then there is a multiple eigenvalue $\lambda = 1$ for eigenvectors $x = \left[x_1^T\ ,\ x_2^T\right]$, where*

$$x_1 \in ker(A_{21}(I_1 - C_{11}A_{11})), x_2 \in ker((A_{11}^{-1} - B_{11})A_{12}) .$$

The remaining eigenvalues equal

$$\lambda = 1 + \frac{1}{2}\zeta(1 \pm \sqrt{1 + 4/\zeta})$$

for ζ equal to any nonzero eigenvalue of the matrix product $\widetilde{A}_{12}\widetilde{A}_{21}$, where $\widetilde{A}_{12} = (I_1 - B_{11}A_{11})A_{11}^{-1}A_{12}$ and $\widetilde{A}_{21} = S_2^{-1}A_{21}(I_1 - C_{11}A_{11})$.

If $C_{11} = B_{11}$, A is symmetric and S_2 is positive definite, then the eigenvalues ζ are real and positive and

$$\lambda_{\max} = 1 + \frac{1}{2}\sigma(1 + \sqrt{1 + 4/\sigma})\,, \;\; \lambda_{\min} = \frac{\sqrt{1 + 4/\sigma} - 1}{\sqrt{1 + 4/\sigma} + 1}\,,$$

where $0 < \zeta \leq \sigma$, $\sigma = \rho(\widetilde{A}_{12}\widetilde{A}_{21})$.

In the general case of nonsymmetric matrices, if we let $\sigma = \|\widetilde{A}_{12}\widetilde{A}_{21}\|$, *then for the absolute values of the eigenvalues it holds that* $\lambda_{\min} \leq \mid \lambda \mid \leq \lambda_{\max}$.

Remark 4 The method parameter $\sigma = \rho(\widetilde{A}_{12}\widetilde{A}_{21})$ can be said to replace γ, or rather $\gamma^2/(1-\gamma^2)$, where $\gamma = \rho(A_{11}^{-1}A_{12}A_{22}^{-1}A_{21})$, which is the familiar CBS constant, defined for symmetric positive definite matrices A (see, e.g., [4]). Note that γ is fixed while the value of σ is controllable by the construction of the preconditioner (the choice of B_{11} and C_{11}). Furthermore, the new parameter is also applicable for nonsymmetric and indefinite matrices.

For a saddle point problem, it is readily seen that $\sigma = 1$ or $\sigma = -1$, depending on the sign of the approximate Schur complement.

Remark 5 From both a computational complexity and preconditioning efficiency point of view, the most efficient form of the preconditioner is of block–triangular form. If we let $C_{11} = \widetilde{A}_{11}^{-1}$, and let $B_{11} = 0$, then it takes the form

$$C = \begin{bmatrix} \widetilde{A}_{11} & 0 \\ A_{21} & S \end{bmatrix} \quad \text{and } \sigma = \|A_{11}^{-1}A_{12}S_2^{-1}A_{21}(I_1 - \widetilde{A}_{11}^{-1}A_{11})\|\,.$$

Here we can control the value of σ, and make it arbitrarily small, by performing a sufficient number of inner iterations in solving the arising systems with matrix A_{11}. In the limit case, where $\widetilde{A}_{11}^{-1} = A_{11}^{-1}$ and $S = S_2$, we get $\sigma = 0$ and the preconditioned matrix takes the form

$$C^{-1}A = \begin{bmatrix} I_1 & A_{11}^{-1}A_{12} \\ 0 & I_2 \end{bmatrix}\,.$$

In this case, the minimal polynomial to $C^{-1}A$ is simply $\mathcal{P}_2(t) = (1-t)^2$, and only two iterations in a generalized outer conjugate gradient iteration method will occur. There will in general be more iterations when $\widetilde{A}_{11}^{-1}$ is an approximation of A_{11}^{-1} but, as shown above, we can control this number by making a sufficient number of inner iterations.

In the general case, where $\widetilde{A}_{11}^{-1} \approx A_{11}^{-1}$ and $S \approx S_2$, it can be shown that the eigenvalues λ_1 of $C^{-1}A$ are located in discs around $1 + \mu_i$ where

$$\mu_i - \frac{1}{2}(\zeta_i \pm \sqrt{\zeta_i^2 + 4\zeta_i})$$

and ζ_i are the nonzero eigenvalues of $A_{11}^{-1}A_{12}S_2^{-1}A_{21}(I_1 - \widetilde{A}^{-1}A_{11})$. The radius of the disks is bounded by $q(1 + \mid \mu_i \mid)$, where

$$q = \|Q^{-1}\begin{bmatrix} \widetilde{A}^{-1}A_{11} - I_1 & 0 \\ 0 & S^{-1}S_2 - I_2 \end{bmatrix} Q\|\,.$$

The radius can be controlled by making a sufficient number of inner iterations when solving the arising systems with the pivot block A_{11} and by choosing a sufficiently accurate approximation S of S_2.

For the saddle point systems, arising from discretizing the stationary Navier-Stokes equation, for not strongly convection-dominated flows, Schur complement matrices can be approximated quite efficiently using the assembly of local, exact Schur complements and possibly extending this approximation to a sequence of coarser matrices as in a multilevel preconditioning method, see [33].

7 Numerical tests

We consider the Cahn-Hilliard equation in the form (8), in two space dimensions, discretized using triangular meshes and standard linear conforming finite elements for both variables. We illustrate the numerical efficiency of the proposed preconditioner $\mathcal{B}_0$ on the following test problem.

Problem 1 The model test setting is with no convection (i.e., the coefficient vector $\mathbf{b}$ in (8) is identically zero) and the phase interface moves due to diffusion processes only.

The arising Jacobian matrices are of the form $\mathcal{C}$ as in (9). We consider two cases.
Case 1: We solve the problem, applying Newton's method using the exact Jacobian matrix $\mathcal{C}$. During each nonlinear iteration, $\mathcal{C}$ is solver iteratively using `gcg-mr`, preconditioned by $\mathcal{C}_0$, given in (10), neglecting the block J. The results are in Tables 1 and 2.
Case 2: We solve the problem using inexact Newton's method, i.e., we use the matrix $\mathcal{C}_0$ as an approximate Jacobian. This leads on one side to an increase of the nonlinear iterations, and on another side, to significantly less computational effort per nonlinear iteration. The results are presented in Table 3.

We present experiments, where systems with $\mathcal{C}_0$ are solved with a direct method (Table 1) as well as experiments, where $\mathcal{C}$ is preconditioned by $\mathcal{B}_0$ as in (12). In its turn, when solving systems with $\mathcal{B}_0$, the block $M + \widetilde{\beta}\widetilde{K}$ in $\mathcal{B}_0$ is solved by an AMG-preconditioned conjugate gradient method to an inner relative accuracy 10^{-3} (Table 2).

For Case 1, we also illustrate the performance of $\mathcal{C}_0$ and $\mathcal{B}_0$ when the mass matrix is replaced by its diagonal (the lower part in Tables 1 and 2). We see that replacing the mass matrix in the preconditioner by its diagonal leads to an increase of the number N_2 of linear iterations. Numerical experiments, not included here, indicate that replacing M by its diagonal in Case 2, leads to a significant increase of the nonlinear iterations. (For nonconforming Crouzeix-Raviart elements the mass matrix is diagonal.)

In all tables, two or three iteration counts are presented in the form N_1/N_2 or $N_1/N_2/N_3$. Here N_1 denotes the average number of nonlinear iterations for 10 time steps. The count N_2 is the average number of linear iterations per nonlinear iteration for 10 time steps to solve a linear system with $\mathcal{C}$, preconditioned by $\mathcal{B}_0$. The third iteration count N_3 is the average number of inner iterations to solve $\mathcal{C}_0$, preconditioned by $\mathcal{B}_0$ per linear iteration when solving the Jacobian in Table 2 and the average number of AMG-preconditioned

	Δt					
Size	h	$h/2$	$h/4$	$h/10$	h^2	h^3
			M			
8450	8 / 19	3 / 10	3 / 8	3 / 6	2 / 5	2 / 4
33282	3 / 9	3 / 7	3 / 6	3 / 5	3 / 4	2 / 3
132098	3 / 7	3 / 6	3 / 5	3 / 5	3 / 4	3 / 3
			$diag(M)$			
8450	7 / 22	3 / 20	3 / 17	3 / 15	3 / 13	2 / 13
33282	3 / 18	3 / 17	3 / 14	3 / 13	3 / 13	3 / 14

Table 1: Equations (8), no convection: Exact Newton method, $\mathcal{C}$ is preconditioned by $\mathcal{C}_0$, direct solve with $\mathcal{C}_0$

	Δt					
Size	h	$h/2$	$h/4$	$h/10$	h^2	h^3
			M			
8450	7 / 19 / 6	3 / 10/ 5	3 / 8 / 5	3 / 6 / 5	3 / 5 / 4	3 / 5 / 4
33282	3 / 10 / 5	3 / 8 / 5	3 / 6 / 5	3 / 5 / 4	3 / 4 / 4	3 / 4 / 5
132098	3 / 7 / 5	3 / 6 / 5	3 / 5 / 4	3 / 5 / 4	3 / 4 / 4	3 / 4 / 5
			$diag(M)$			
8450	7 / 20 / 7	3 / 10 / 7	3 / 8 / 6	3 / 6 / 6	3 / 5 / 6	3 / 5 / 8
33282	3 / 10 / 7	3 / 8 / 6	3 / 6 / 6	3 / 6 / 6	3 / 5 / 6	3 / 4 / 7
132098	3 / 7 / 6	3 / 6 / 6	3 / 5 / 6	3 / 5 / 6	3 / 5 / 6	3 / 4 / 7

Table 2: Equations (8), no convection: Exact Newton method, $\mathcal{C}$ is preconditioned by $\mathcal{C}_0$, inner solver for $\mathcal{C}_0$, preconditioned by $\mathcal{B}_0$

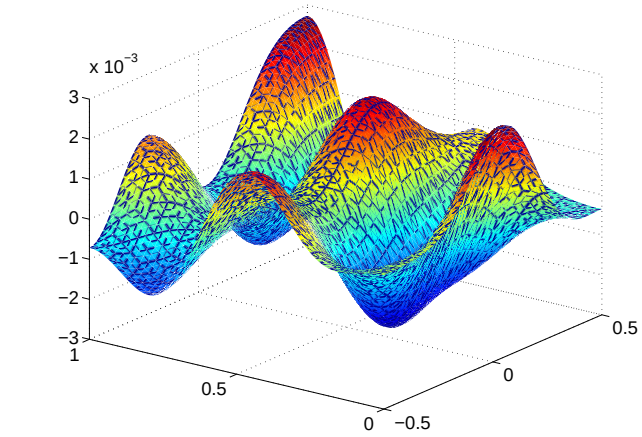
conjugate gradient method to solve the block $M + \widetilde{\beta}\widetilde{K}$ in Table 2, per nonlinear iteration, per linear iteration for 10 time steps.

We see that neglecting the nonlinear term in the Jacobian leads to a very efficient preconditioner, which furthermore (for fixed meshes) does not have to be recomputed.

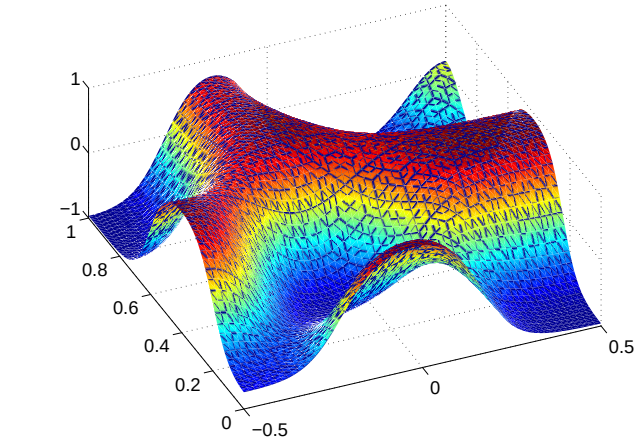
It is seen that the iteration number counts are rather insensitive to the mesh size h and to the time step $\tau \leq h$.

The relative stopping criteria for the nonlinear, linear and inner iterations, were chosen as 10^{-6}, 10^{-6} and 10^{-3}, respectively. As the number of iterations are already small, we did not tune these parameters, i.e., we did not test for more efficient combinations of them and some further reduction of the number of iterations.

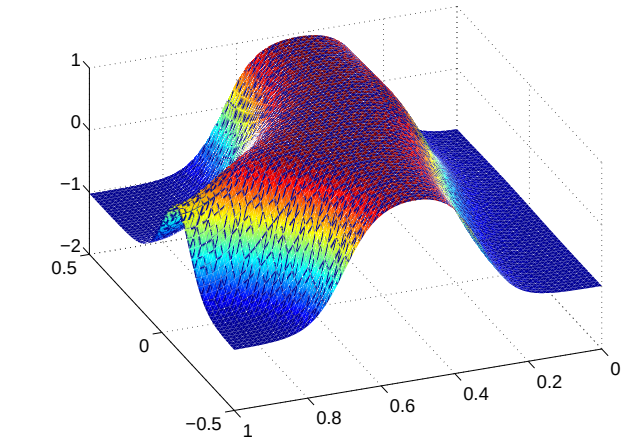
Remark 6 Numerical experiments in [9] show, that, in Case 1, if we precondition $\mathcal{C}$ by cB_0 instead of $\mathcal{C}_0$, the nonlinear iterations are not increased, while the resulting solver is less computationally intensive.



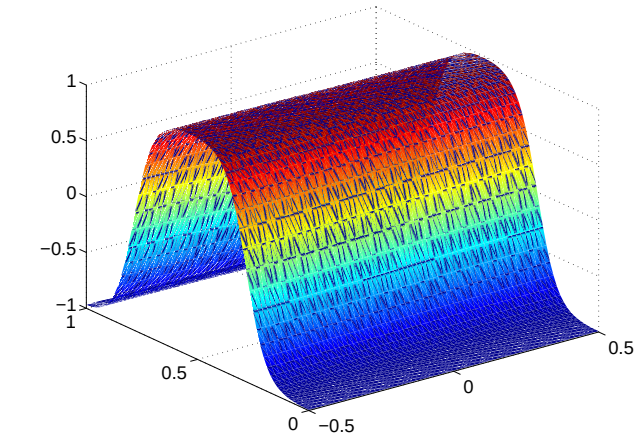
(a) $t = 2\Delta t$



(b) $t = 10\Delta t$



(c) $t = 15\Delta t$



(d) $t = 85\Delta t$

Figure 2: Equations (8), no convection: the evolution of the solution at different time steps, $\Delta t = h/10$

	Δt				
Size	h	$h/2$	$h/4$	$h/10$	h^2
direct solve with $\mathcal{C}_0$					
8450	>100	30	13	8	6
33282	29	13	9	7	6
132098	13	9	7	6	5
direct solve with $\mathcal{B}_0$					
8450	> 100	30 / 4	13 / 4	8 / 4	6 / 4
33282	30 / 4	13 / 4	9 / 4	7 / 4	6 / 4
132098	13 / 4	9 / 4	7 / 4	6 / 4	5 / 3
AMG-preconditioned CG for the block $M + \widetilde{\beta}\widetilde{K}$					
8450	> 100	31 / 5 / 5	13 / 5 / 5	8 / 5 / 5	6 / 4 / 6
33282	31 / 5 / 5	14 / 5 / 5	9 / 6 / 5	7 / 5 / 5	6 / 4 / 5
132098	13 / 6 / 5	9 / 6 / 5	7 / 6 / 5	7 / 5 / 5	5 / 4 / 5

Table 3: Equations (8), no convection: Inexact Newton method, inner solve with $\mathcal{C}_0$, preconditioned by $\mathcal{B}_0$

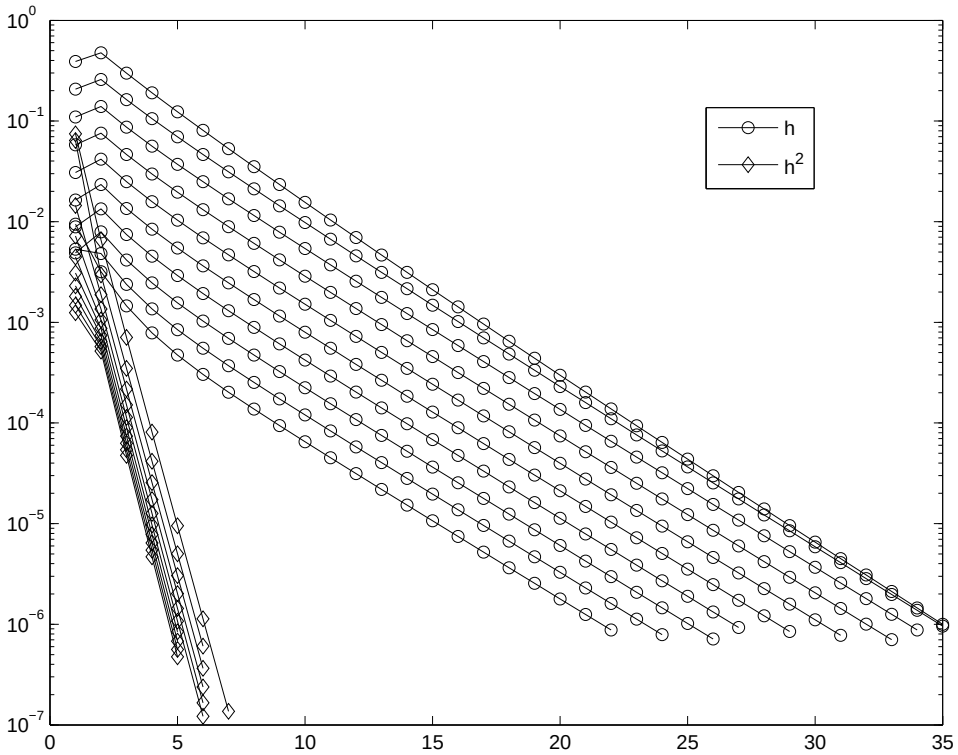


Figure 3: Equations (8), no convection: Inexact Newton method, nonlinear convergence for 10 timesteps for $\tau = h$ and $\tau = h^2$ (problem size 8450)

The AMG implementation, used for the numerical experiments is from [44].

8 Conclusions

It has been shown that the use of efficient preconditioners can avoid the need to use many operator splittings, in this way decreasing the total operator splitting error and, hence, allowing for larger timesteps.

In addition, combined with symmetrized operator splitting methods even larger timesteps can be used. Properly approximated Newton iteration methods helps avoiding the use of a pure Newton linearized operator, which is desirable when this is nearly singular or indefinite.

The linear systems arising at each time step can be preconditioned by a preconditioner in a form which resembles that arising for complex valued systems rewritten in real valued form, and for which the corresponding linear systems are best solved by inner iterations and decoupled preconditioners. In this way, very few iterations are required also for the inner iterations.

9 Acknowledgments

The work of the second author is partly supported by the Swedish Research Council (VR) via the grant *Finite element preconditioners for algebraic problems as arising in modelling of multiphase microstructures*, 2009-2011. The support is hereby gratefully acknowledged.

We also thank the developers of the HSL MI-20 AMG package for kindly providing us with the AMG-`Matlab` interface.

References

[1] D.M. Anderson, G.B. McFadden, A.A. Wheeler, Diffusive–interface methods in fluid mechanics, *Ann. Rev. Fluid. Mech.*, 30 (1998), 139–165.

[2] O. Axelsson, On the efficiency of a class of A–stable methods, *BIT* 14 (1974), 279–287.

[3] O. Axelsson, Error estimates over infinite intervals of some discretization of evolution equations, *BIT* 24 (1984), 413–424.

[4] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, 1994.

[5] O. Axelsson, Globally convergent continuation methods for non–linear equations. In:*Large Scale Scientific Computations of Engineering and Environmental Problems*, Proceedings, Varna, Bulgaria, June 7–11,1997. Edited by M. Griebel, Oleg P. Iliev, Svetozar Margenov, Panayot S. Vassilevski, Vieweg Verlag 1998, 3–17.

[6] O. Axelsson, R. Blaheta, Preconditioning of matrices partitioned in two by two block form; Eigenvalue estimates and Schwarz DD mixed FEM, *Numer. Linear Algebra Appl.*, 17 (2010), 787–810.

[7] O. Axelsson, A. Kucherov, Real valued iterative methods for solving complex symmetric linear systems. *Numer. Linear Algebra Appl.*, 7 (2000), 197–218.

[8] O. Axelsson, M. Neytcheva. A General Approach to Analyse Preconditioners for Two-by-Two Block Matrices TR 2010-029, Institute for Information Technology, Uppsala University, November 2010. Submitted.

[9] P. Boyanova, M. Do–Quang, M. Neytcheva, On solution methods for Cahn–Hilliard equation discretized by conforming and non–conforming FEM, TR 2011-004, Institute for Information Technology, Uppsala University, March 2011.

[10] M. Do–Quang, G. Amberg, The splash of a ball hitting a liquid surface: Numerical simulation of the influence of wetting, *Physics of Fluids*, 2008.

[11] P. Csomós, I. Faragó, Á. Havasi, Weighted sequential splitting and their analysis, *Comp.and Math. with Appl.*, 50, 1017–1031.

[12] J. Descloux, M. Flück, M. V. Romerio, A problem of magnetostatics related to thin plates, *Mathematical Modelling and Numerical Analysis*(M^2AN), 32 (1998), 859–876.

[13] I. Dimov, I. Faragó, Z. Zlatev, L-Commutativity of the operators in splitting methods for air pollution models, *Annales. Univ. Sci. Sec. Math.* 44 (2001), 127–148.

[14] J. Douglas Jr., J.E. Gunn, A general formulation of alternating direction methods. *Numer. Math.*, 6 (1964), 428–453.

[15] E.G. D'yakonov, Difference schemes with splitting operator for multi–dimensional non–stationary problems. *Zh. Vychisl. mat. i Mat. Fiz.* 2 (1962), 549–568.

[16] M. Flück, T. Hofer, M. Picasso, J. Rappaz, G. Steiner, Scientific computing for aluminium production, *Int. J. Numerical Analysis and Modelling* 6 (2009), 489–504.

[17] R. Glowinski, T. W. Pan, T. I. Hesla, D. D. Joseph, J. Périaux, A fictitious domain approach to the direct numerical simulation of incompressible viscous flow past moving rigid bodies; Application to particulate flow, *Journal of Computational Physics*, 169 (2001), 363–426.

[18] R. Glowinski, P. Le Tallec, *Augmented Lagrangian and Operator Splitting Methods in Nonlinear Mechanics*, SIAM, Philadelphia, 1989.

[19] A.R. Gourlay, A.R. Mitchell, The equivalence of certain alternating direction and locally one–dimensional difference methods. *SIAM Journal on Numerical Analysis*, 6 (1969), 17–46.

[20] P.M. Gresho, R.L. Sani, *Incompressible Flow and the Finite Element Method. II* ,John Wiley & Sons, 2000.

[21] P.P. Grinevich, M.A. Olshanskii, An iterative method for the Navier-Stokes-type problem with variable viscosity. *SIAM Journal on Scientific Computing*, 31 (2009), 3959–3978.

[22] J.-L. Guermond, A. Salgado, A splitting method for incompressible flows with variable density based on a pressure Poisson equation. *Journal of Computational Physics*, 2009, doi:10.1016/j.jcp.2008.12.036

[23] E. Hairer, G. Wanner, *Solving Ordinary Differential Equations II.Stiff and Differential-Algebraic Problems*, Springer Series in Comput. Mathematics, Vol. 14, Springer-Verlag, Second revised edition 1996.

[24] X. He, Preconditioning the variable viscosity incompressible Navier-Stokes equations. In progress.

[25] O. Axelsson, I. Gustafsson M.J. Holst, M.G. Larson, A. Målqvist, R. Söderlund Convergence analysis of finite element approximations of the Joule heating problem in three spatial dimensions. BIT Numerical Mathematics, 50 (2010), 781-795.

[26] T.Y. Hou, X.-H. Wu, A multiscale finite element method for elliptic problems in composite materials and porous media. *Journal of Computational Physics*, 134 (1997), 169–189.

[27] W. Hundsdorfer, J.G. Verwer, *Numerical Solution of Time–Dependent Advection–Diffusion–Reaction Equations*, Springer, Berlin (2003).

[28] D. Lanser, J.G. Verwer, Analysis of operator splitting for advection-diffusion-reaction problems in air pollution modeling, *J. Comput. Appl. Math.* 111 (1999), 201–216.

[29] P.-L. Lions. *Mathematical Topics in Fluid Mechanics: Volume 1: Incompressible Models* Oxford Lecture Series in Mathematics and Its Applications, 3, 1996.

[30] F. Losasso, F. Gibou, R. Fedkiw, Simulating water and smoke with an octree data structure. *ACM Transactions on Graphics (TOG)*, Proceedings of ACM SIGGRAPH 2004 23, August 2004.

[31] G.I. Marchuk, Some application of splitting–up methods to the solution of mathematical physics problems, *Appl. Mat.* 13(2) (1968), 103–132.

[32] D.J. Mavriplis, Multigrid strategies for viscous flow solvers on anisotropic unstructured meshes. *Journal of Computational Physics*, 145 (1998), 141–165.

[33] M. Neytcheva, M. Do-Quang, X. He. Element-by-Element Schur Complement Approximations for General Nonsymmetric Matrices of Two-by-Two Block Form. In I. Lirkov, S. Margenov, J. Was'niewski (Eds.): Lecture Notes in Computer Science Vol. 5910, LSSC 2009, pp. 108–115. Springer, Heidelberg (2010). Refereed.

[34] A. Novick–Cohen, The Cahn–Hilliard equation: Mathematical and modelling perspectives. *Advances in Mathematical Sciences and Applications*, Gakkotosho, Tokyo, 8(2) (1998), 965–985.

[35] U. van Rienen, Numerical Methods in Computational Electrodynamics - Linear Systems in Practical Applications, Springer *Lecture Notes in Computational Science and Engineering*, 15 (2000).

[36] G. Strang, On the construction and comparison of difference schemes, *SIAM J. Numer. Anal.* 5(3)(1968), 506–517.

[37] J.G. Verwer, Contractivity of locally one–dimensional splitting methods, *Numer. Math.* 44(1984), 247–259.

[38] J.G. Verwer, W.H. Hundsdorfer, J.G. Blom, Numerical time integration for air pollution models, *Surveys on Mathematics for Industry* 10 (2002), 107–114.

[39] W. Villanueva, G. Amberg, Some generic capillary–driven flows. *Int. J. of Multiphase Flow*, 32 (2006), 1072-1086.

[40] J.D. van der Waals, The thermodynamic theory of capillary flow under the hypothesis of a continuous variation of density. Verhandel/Konik. Akad. Weten. 1 (1893), (in Dutch). *J. Stat. Phys.* 20 (1979) (in English).

[41] M.F. Wheeler, T. Wildey, Guangri Xue, Recent advances in multiscale mortar methods. *Numer. Linear Algebra Appl.* 17 (2010), 1–6.

[42] N.N. Yanenko, *The Method of Fractional Steps*, Springer–Verlag, Berlin (1971).

[43] Z. Zlatev, *Computer Treatment of Large Pollution Models*. Kluwer, Dordrecht, 1995.

[44] The HSL Mathematical Software Library, `http://www.hsl.rl.ac.uk/`