European Conference on Computational Fluid Dynamics
ECCOMAS CFD 2006
P. Wesseling, E. Oñate and J. Périaux (Eds)

BLOCK PRECONDITIONERS FOR FULLY IMPLICIT RUNGE-KUTTA SCHEMES APPLIED TO THE BIDOMAIN EQUATIONS

Gunnar A. Staff[*] and Kent-Andre Mardal[†] and Trygve K. Nilssen[††]

[*] Simula Research Laboratory
1325 Lysaker, Norway
e-mail: gunnaran@simula.no
web page:http://www.simula.no/portal_memberdata/gunnaran
[†] Simula Research Laboratory
1325 Lysaker, Norway
e-mail:kent-and@simula.no
web page:http://www.simula.no/portal_memberdata/kent-and
[††] Scandpower Petrolium Technology AS
N-2027 Kjeller, Norway
e-mail:trygve.nilssen@scandpowerpt.com

Key words: Block preconditioners Runge-Kutta methods, order-optimal methods

Abstract. *Recently, the authors presented different block preconditioners for implicit Runge-Kutta discretization of the heat equation. The preconditioners were block Jacobi and block Gauss-Seidel preconditoners where the blocks reused existing preconditioners for the implicit Euler discretization of the same equation. In this paper we will introduce similar block preconditioners for the implicit Runge-Kutta discretization of the Bidomain equation. We will, by numerical experiments, show the properties of the preconditoners, and that higher-order Runge-Kutta discretization of the Bidomain equation may be superior to lower-order in some cases.*

1 Introduction

Since Runge published his paper in 1895[15], the Runge-Kutta methods have proven to be effective solvers of different ODEs. However, when the system of ODEs originates from a parabolic PDE, these methods are not that common. The majority of these problems are computed using single step methods like Implicit Euler, Crank-Nicolson or some higher-order Backward Differential Formulas (BDF) in time.

When a system of ODEs with dimension m is discretized using an s stage Runge-Kutta method, the system to be solved is of dimension $sm \times sm$. In addition, even though the system of ODEs is symmetric positive definite, the resulting Runge-Kutta discretization is in general nonsymmetric and non-positive definite. This calls for different linear solvers and often prevents the use of well known preconditioners.

In[14,17], we addressed this problem for the heat equation by introducing a block preconditioner that reuses standard preconditioners. It was proven that the block Jacobi preconditioner is order optimal with respect to the timestep and the spatial discretization parameters, assuming that the preconditioner for the implicit Euler discretization is.

In this paper we will present block preconditioners for the Runge-Kutta discretization of the Bidomain equations which is a mathematical model of the electrical activity in the heart. The Bidomain equations are an index 1 problem, in contrast to the heat equation from[14,17] which is index 0.

The remaining of this paper is organized like this: Section 2 explains the Bidomain equation, and emphasises the block structure of the resulting discretized system. The preconditioners are presented in Section 3. In Section 4 we present a variety of numerical experiments, demonstrating the effectiveness of the preconditioner, while in Section 5 we summarize the results.

2 Discretization of the Problem

Computer based simulations of the electrical activity in the heart is an active research field of applied mathematics. The electrical activity of the heart can be modelled using the Bidomain equations[19], where relevant application of the model are e.g. studies of arrhytmia and defibrillation.

To discretized and solve with high accuracy in space is a simple task. To do the same for the time variable is more challenging since it prevents the use of standard preconditioners. This is the problem we will address by introducing block preconditioners for the Runge-Kutta discretization.

The mathematical model can be written as:

$$\frac{\partial \mathfrak{s}}{\partial t} = F(t, \mathfrak{s}, v) \quad x \in \Omega, \tag{1}$$

$$\frac{\partial v}{\partial t} = \nabla \cdot (\sigma_{\text{in}} \nabla v) + \nabla \cdot (\sigma_{\text{in}} \nabla u) - I(v, \mathfrak{s}) \quad x \in \Omega, \tag{2}$$

$$0 = \nabla \cdot (\sigma_{\text{in}} \nabla v) + \nabla \cdot ((\sigma_{\text{in}} + \sigma_{\text{ex}}) \nabla u) \quad x \in \Omega, \tag{3}$$

where the unknowns are the trans-membrane potential v, the extra cellular potential u and the state variables $\mathfrak{s}$, which varies from 1 and up to 40 in the most realistic models[19]. The intra– and extra cellular conductivity tensors are denoted σ_{in} and σ_{ex}, respectively. For notational convenience, the tensors are scaled by the membrane capacitance and the membrane surface area, see[18] for details. Depending on the membrane model, the rate function F might describe ionic fluxes, enzyme kinetics and possibly other entities. The function I is current flux across the cell membrane. The computational domain is denoted Ω.

A common way to solve these systems of equations is to use an operator splitting technique to split (1) from (2)-(3)[18]. We will not focus on this splitting in this paper. We will assume that this splitting can be done, and instead study how we can solve

(2)-(3) using implicit Runge-Kutta methods in time, and how we can reuse well known preconditioners to create efficient preconditioners for this reduced system.

We start by discretize (2)-(3) in space using e.g. a finite element method. The resulting system can be written as

$$I_h \frac{\mathrm{d}v}{\mathrm{d}t} = -M_{\mathrm{in}} v - M_{\mathrm{in}} u + f_v, \tag{4}$$

$$0 = -M_{\mathrm{in}} v - M_{\mathrm{tot}} u + f_u, \tag{5}$$

where $I_h \in \mathbb{R}^{n\times n}$ is a mass matrix, $M_{\mathrm{in}} \in \mathbb{R}^{n\times n}$ is the spatial discretization matrix of $\nabla \cdot (\sigma_{\mathrm{in}} \nabla)$ (a stiffness matrix) and $M_{\mathrm{tot}} \in \mathbb{R}^{n\times n}$ is the discretization of $\nabla \cdot ((\sigma_{\mathrm{in}} + \sigma_{\mathrm{ex}})\nabla)$ (also a stiffness matrix) . The resulting system is a differential-algebraic equation (DAE) of index 1, which requires solution methods in time which are stiffly accurate. Suitable methods are the Backward Differentiation Formulae (BDF), and Runge-Kutta methods with RadauIIA collocation methods[4,9].

When the equations (4)-(5) are discretized using single stage discretization schemes such as implicit Euler or higher-order BDF schemes, we arrive at the following linear system to be solved at each time step:

$$\begin{bmatrix} I_h + \delta t M_{\mathrm{in}} & \delta t M_{\mathrm{in}} \\ \delta t M_{\mathrm{in}} & \delta t M_{\mathrm{tot}} \end{bmatrix} \begin{bmatrix} v \\ u \end{bmatrix}^{n+1} = \begin{bmatrix} v \\ 0 \end{bmatrix}^{n} + \delta t \begin{bmatrix} f_v \\ f_u \end{bmatrix}^{n+1}.$$

All higher-order single stage methods result in more terms on the right hand side, leaving the left hand side unchanged (except that δt on the left hand side will be substituted with $\delta t \alpha_0$).

For a Runge-Kutta discretization of our problem, the stage variables can be computed as

$$V_i = v_{n-1} + \delta t \sum_{j=1}^{s} a_{ij}(-M_{\mathrm{in}} V_j - M_{\mathrm{in}} U_j + f_v(t_j)), \quad i = 1, \ldots, s \tag{6}$$

$$0 = -M_{\mathrm{in}} V_i - M_{\mathrm{tot}} U_i + f_u(t_i), \quad i = 1, \ldots, s, \tag{7}$$

where the s is the number of quadrature points and a_{ij} are the Runge-Kutta coefficients, b_i are the quadrature weights and c_i are the quadrature nodes organized in the Butcher tableau

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array}.$$

Note that the reduction from the standard formulation

$$0 = \delta t \sum_{j=1}^{s} a_{ij} \left(-M_{\mathrm{in}} V_j - M_{\mathrm{tot}} U_j + f_u(t_j)\right), \quad i = 1, \ldots, s, \tag{8}$$

to (7) assumes that the Runge-Kutta coefficient matrix is invertible[9]. Then the next timestep is found by

$$v^{n+1} = V_s,$$
$$u^{n+1} = U_s,$$

assuming that the chosen Runge-Kutta scheme is stiffly accurate ($a_{si} = b_i$, $i = 1, \ldots, s$). The implicit Runge-Kutta methods with RadauIIA quadrature have both an invertible A and are stiffly accurate scheme. By using these methods the global error for v_n and u_n is $\mathcal{O}(\delta t^{s2-1})$[9]. The implicit Runge-Kutta methods with LobattoIIIC quadrature have also an invertible A, and are stiffly accurate. These methods have an global error for v_n and u_n of $\mathcal{O}(\delta t^{s2-2})$[9].

To better understand the block structure arising from the Runge-Kutta discretization, we write the scheme on block matrix form for $s = 2$:

$$\begin{bmatrix} I_h + \delta t a_{11} M_{\text{in}} & \delta t a_{11} M_{\text{in}} & \delta t a_{12} M_{\text{in}} & \delta t a_{12} M_{\text{in}} \\ \delta t a_{11} M_{\text{in}} & \delta t a_{11} M_{\text{tot}} & 0 & 0 \\ \delta t a_{21} M_{\text{in}} & \delta t a_{21} M_{\text{in}} & I_h + \delta t a_{22} M_{\text{in}} & \delta t a_{22} M_{\text{in}} \\ 0 & 0 & \delta t a_{22} M_{\text{in}} & \delta t a_{22} M_{\text{tot}} \end{bmatrix} \begin{bmatrix} V_1 \\ U_1 \\ V_2 \\ U_2 \end{bmatrix}$$
$$= \begin{bmatrix} u_{n-1} + \delta t (a_{11} f_{u,1} + a_{12} f_{u,2}) \\ \delta t a_{11} f_{v,1} \\ u_{n-1} + \delta t (a_{21} f_{u,1} + a_{22} f_{u,2}) \\ \delta t a_{22} f_{v,2} \end{bmatrix}.$$

For reasons that will be explained later, we have multiplied (7) with $\delta t a_{ii}$.

This can be written on the compact tensor-product form

$$\mathcal{A} = I^{s \times s} \otimes \begin{bmatrix} I_h & 0 \\ 0 & 0 \end{bmatrix} + \delta t \left(\text{diag}(A) \otimes \begin{bmatrix} 0 & 0 \\ M_{\text{in}} & M_{\text{tot}} \end{bmatrix} + A \otimes \begin{bmatrix} M_{\text{in}} & M_{\text{in}} \\ 0 & 0 \end{bmatrix} \right), \quad (9)$$

where $I^{s \times s}$ is the identity matrix in s dimensions. This will be denote the *standard* formulation.

Since Gauss-Seidel depends on the numbering of the unknowns, we may arrange the system with the quadrature nodes in increasing order and sort the system for U and then V. This results in an alternative formulation:

$$\begin{bmatrix} \delta t a_{11} M_{\text{tot}} & 0 & \delta t a_{11} M_{\text{in}} & 0 \\ 0 & \delta t a_{22} M_{\text{tot}} & 0 & \delta t a_{22} M_{\text{in}} \\ \delta t a_{11} M_{\text{in}} & \delta t a_{12} M_{\text{in}} & I_h + \delta t a_{11} M_{\text{in}} & \delta t a_{12} M_{\text{in}} \\ \delta t a_{21} M_{\text{in}} & \delta t a_{22} M_{\text{in}} & \delta t a_{21} M_{\text{in}} & I_h + \delta t a_{22} M_{\text{in}} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ V_1 \\ V_2 \end{bmatrix}$$
$$= \begin{bmatrix} \delta t a_{11} f_{v,1} \\ \delta t a_{22} f_{v,2} \\ u_{n-1} + \delta t (a_{11} f_{u,1} + a_{12} f_{u,2}) \\ u_{n-1} + \delta t (a_{21} f_{u,1} + a_{22} f_{u,2}) \end{bmatrix}.$$

In tensor-product notation this simply means that we have changed the order of the factors in the tensor-product, and then changed the position of v and u.

$$\mathcal{A} = \begin{bmatrix} 0 & 0 \\ 0 & I_h \end{bmatrix} \otimes I^{s\times s} + \delta t \left(\begin{bmatrix} M_{\text{tot}} & M_{\text{in}} \\ 0 & 0 \end{bmatrix} \otimes \text{diag}(A) + \begin{bmatrix} 0 & 0 \\ M_{\text{in}} & M_{\text{in}} \end{bmatrix} \otimes A \right). \tag{10}$$

We will denote this the UV formulation.

3 The Preconditioner

A general description of the preconditioned problem is:

$$\mathcal{B}_L \mathcal{A} \mathcal{B}_R \mathbf{x} = \mathcal{B}_L \mathbf{b},$$

where $\mathcal{B}_L$ and $\mathcal{B}_R$ are the left and right preconditioners, respectively. It will be made clear from the context whether $\mathcal{B}$ is a left or a right preconditioner, so we will from now on drop the subscript for left and right.

When we write $\mathcal{B}$, it means exact preconditioning, meaning that the inversion is done exactly. $\tilde{\mathcal{B}}$ means that we compute a cheap approximation of $\mathcal{B}$ using in our case multigrid, but it may equally well be a domain decomposition method or other well known preconditioning techniques.

3.1 Block preconditioners for the Bidomain model

We will start by presenting a block preconditioner for the Bidomain equation, first presented by Sundnes et al.[20] and then proven by Mardal et al.[13]. They present a block Jacobi and a symmetric block Gauss-Seidel preconditioner, and prove that it is order optimal assuming that the preconditioner for each block is order optimal.

Assume that we can write our problem (2)-(3) on a general block matrix form

$$\mathcal{A} = \begin{bmatrix} A & B \\ C & D \end{bmatrix}. \tag{11}$$

$$\tag{12}$$

where A in this case is a general matrix, and not the Runge-Kutta coefficient matrix. Then the block Jacobi and block symmetric-Gauss-Seidel can be written as

$$\mathcal{B}_J^{-1} = \begin{bmatrix} A & 0 \\ 0 & D \end{bmatrix}, \tag{13}$$

$$\mathcal{B}_{SGS}^{-1} = \mathcal{B}_{GSL}^{-1} \mathcal{B}_J \mathcal{B}_{GSU}^{-1}, \quad \mathcal{B}_{GSL}^{-1} = \begin{bmatrix} A & 0 \\ C & D \end{bmatrix}, \quad \mathcal{B}_{GSU}^{-1} = \begin{bmatrix} A & B \\ 0 & D \end{bmatrix}. \tag{14}$$

By using a Runge-Kutta discretization, $\mathcal{A}$ will in general be nonsymmetric, meaning that we do not necessarily need symmetric preconditioners. Therefore we will also use

the block Gauss-Seidel preconditioner $\mathcal{B}_{GSL}$ which from now on will be denoted $\mathcal{B}_{GS}$. The nonsymmetric Gauss-Seidel preconditioners are not discussed in the paper by Mardal et al.[13], and we will therefore indicate an order-optimal behaviour by presenting numerical results.

3.2 Block preconditioners for the Runge-Kutta discretization

We will investigate both a block Jacobi and a block Gauss-Seidel preconditioner for the standard formulation (9) and the UV formulation (10). These two formulations gives two different approaches for the construction of the preconditioners. We start with the UV-formulation (10).

$$\mathcal{B}_J^{-1} = \begin{bmatrix} \delta t \mathrm{diag}(A) \otimes M_{\mathrm{tot}} & 0 \\ 0 & I^{s\times s} \otimes I_h + \delta t \mathrm{diag}(A) \otimes M_{\mathrm{in}} \end{bmatrix},$$
$$\mathcal{B}_{GS}^{-1} = \begin{bmatrix} \delta t \mathrm{diag}(A) \otimes M_{\mathrm{tot}} & 0 \\ \delta t A \otimes M_{\mathrm{in}} & I^{s\times s} \otimes I_h + \delta t \mathrm{ltri}(A) \otimes M_{\mathrm{in}} \end{bmatrix},$$

where $\mathrm{diag}(\cdot)$ means the diagonal of a matrix, while $\mathrm{ltri}(\cdot)$ means the lower triangular part of a matrix. Both $\mathcal{B}_J$ and $\mathcal{B}_{GS}$ are a straightforward formulation of a block Jacobi or a block Gauss-Seidel preconditioner based on $\mathcal{A}$.

It is now time to explain why we multiplied (7) with $\delta t a_{ii}$. Assume that we didn't, and that we applied the block Jacobi preconditioner $\mathcal{B}_J$. Then we would have $2s$ blocks which looked like

$$(I_h + \delta t a_{ii} M_{\mathrm{in}})^{-1} M_{\mathrm{in}} \approx (\delta t a_{ii} M_{\mathrm{in}})^{-1} M_{\mathrm{in}} = (\delta t a_{ii})^{-1} I,$$

where the first step assumes that the eigenvalues in $\delta t a_{ii} M_{\mathrm{in}}$ dominates the ones in I_h. We would then have $2s$ off-diagonal block with eigenvalues scaling as $(\delta t a_{ii})^{-1}$. This would be disastrous for the condition number, and is obviously something we want to prevent. We therefore multiply (7) with $\delta t a_{ii}$.

If we instead consider the standard-formulation (9), we are in the position to formulate a block-block preconditioner. Let us denote

$$\mathcal{A}_{BD,i} = \begin{bmatrix} I_h + \delta t a_{ii} M_{\mathrm{in}} & \delta t a_{ii} M_{\mathrm{in}} \\ \delta t a_{ii} M_{\mathrm{in}} & \delta t a_{ii} M_{\mathrm{tot}} \end{bmatrix},$$
$$\mathcal{A}_{BD,ij} = \begin{bmatrix} \delta t a_{ij} M_{\mathrm{in}} & \delta t a_{ij} M_{\mathrm{in}} \\ 0 & 0 \end{bmatrix},$$

where $\mathcal{A}_{BD,i}$ is the diagonal blocks in the Runge-Kutta formulation, while $\mathcal{A}_{BD,ij}$ is the off-diagonal blocks. If we assume that we have a preconditioner for $\mathcal{A}_{BD,i}$, then the construction of the preconditioner is similar to the one presented in[14,17].

We therefore introduce the block preconditioning elements

$$\mathcal{B}_{BD,i,J}^{-1} = \begin{bmatrix} I_h + \delta t a_{ii} M_{\text{in}} & 0 \\ 0 & \delta t a_{ii} M_{\text{tot}} \end{bmatrix},$$
$$\mathcal{B}_{BD,i,GS}^{-1} = \begin{bmatrix} I_h + \delta t a_{ii} M_{\text{in}} & 0 \\ \delta t a_{ii} M_{\text{in}} & \delta t a_{ii} M_{\text{tot}} \end{bmatrix},$$

where $\mathcal{B}_{BD,i,J}$ is the block Jacobi preconditioner presented by Mardal et al.[13], and $\mathcal{B}_{BD,i,GS}$ is the lower block Gauss-Seidel preconditioner. For the off-diagonal block elements we also have two options, namely

$$\mathcal{B}_{BD,ij,J}^{-1} = \begin{bmatrix} \delta t a_{ij} M_{\text{in}} & 0 \\ 0 & 0 \end{bmatrix},$$
$$\mathcal{B}_{BD,ij,F}^{-1} = \begin{bmatrix} \delta t a_{ij} M_{\text{in}} & \delta t a_{ij} M_{\text{in}} \\ 0 & 0 \end{bmatrix},$$

where $\mathcal{B}_{BD,ij,J}$ is the off-diagonal block in Jacobi style, while $\mathcal{B}_{BD,ij,F}$ is the full block. In this setting $\mathcal{B}_{BD,ij,J} \equiv \mathcal{B}_{BD,ij,GS}$. The Runge-Kutta Jacobi preconditioners can then be written as

$$\mathcal{B}_{J,X}^{-1} = \begin{bmatrix} \mathcal{B}_{BD,1,X}^{-1} & & & \\ & \mathcal{B}_{BD,2,X}^{-1} & & \\ & & \ddots & \\ & & & \mathcal{B}_{BD,s,X}^{-1} \end{bmatrix},$$

where X is the chosen Bidomain block preconditioner (Jacobi or Gauss-Seidel). The Runge-Kutta Gauss-Seidel preconditioner can be written as

$$\mathcal{B}_{GS,X,Y}^{-1} = \begin{bmatrix} \mathcal{B}_{BD,1,X}^{-1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \mathcal{B}_{BD,s1,Y}^{-1} & \cdots & \mathcal{B}_{BD,s,X}^{-1} \end{bmatrix}, \tag{15}$$

where X as above is the chosen Bidomain block preconditioner, and Y is the chosen block element for the off-diagonal elements (Jacobi or full). Since X and Y may be unlike, this introduces an extra degree of freedom in the choice of preconditioner. For instance is $\mathcal{B}_{GS,GS,F}$ the standard block Gauss-Seidel preconditioner of (10), while $\mathcal{B}_{GS,GS,GS}$ is the one where the block Gauss-Seidel formulation is reused on the off-diagonals.

Conjecture 1 *By using an order optimal preconditioner for each of the diagonal blocks, the block preconditioner $\tilde{\mathcal{B}}_J$ and $\tilde{\mathcal{B}}_{GS}$ will also be order optimal.*

The proof of this is work in progress.

Another interesting property is how well the approximated preconditioners actually work. An indication is given by the Property 1, also stated in[17].

Property 1 *Assuming that $\tilde{\mathcal{B}}$ is the approximation of the exact preconditioner $\mathcal{B}$, using e.g. multigrid or domain decomposition, and that $\mathcal{B}$ is one of the presented block preconditioners. Then the condition number can be bounded by*

$$\kappa(\tilde{\mathcal{B}}\mathcal{A}) \leq \kappa(\tilde{\mathcal{B}}\mathcal{B}^{-1})\kappa(\mathcal{B}\mathcal{A}).$$

The proof is given in[17], and is a simple Cauchy-Schwarz like argument.

We will now try to estimate the sensitivity of the factor $\kappa(\tilde{\mathcal{B}}\mathcal{B}^{-1})$ which measures the effect of the inexact inversion of the preconditioner. Assume that we use a pure block Jacobi preconditioner ($\mathcal{B}_J$ or $\mathcal{B}_{J,J}$), and that M_{in} and M_{tot} are symmetric positive definite, then

$$\kappa(\tilde{\mathcal{B}}\mathcal{B}) = \frac{\max_i(\max(eig(\tilde{\mathcal{B}}_i\mathcal{B}_i^{-1})))}{\min_i(\min(eig(\tilde{\mathcal{B}}_i\mathcal{B}_i^{-1})))}.$$

For the different variety of block Gauss-Seidel preconditioners, the estimate is somehow more complicated. The inverse of a lower triangular matrix is still lower triangular. Assume that the block Gauss-Seidel preconditioner can be written on the general form

$$\mathcal{B}_{GS}^{-1} = \begin{bmatrix} \hat{\mathcal{B}}_{11} & \dots & 0 \\ \vdots & \ddots & \vdots \\ \hat{\mathcal{B}}_{s1} & \dots & \hat{\mathcal{B}}_{ss} \end{bmatrix}.$$

Then we find

$$\left(\tilde{\mathcal{B}}\mathcal{B}^{-1}\right)_{ij} = \begin{cases} \sum_{k=j}^{i} \tilde{\mathcal{B}}_{ik}\hat{\mathcal{B}}_{kj}, & i \geq j \\ 0 & \text{else.} \end{cases}$$

Obviously the block Gauss-Seidel preconditioner is less robust to a poor approximation of the preconditioner then the block Jacobi preconditioner. This is something we will investigate numerically.

4 Results

All implementation is done in the framework of PyCC[2], which is a Python library interfacing compiled packages for matrix-storage, preconditioners etc. It supports higher-order Lagrange elements[5], generated using SyFi[3]. As preconditioner we use the algebraic multigrid preconditioner ML[1] with one V-cycle and symmetric Gauss-Seidel as point smoother. All computations will be done on the unit square $\Omega = (-1, 1)^2$.

4.1 ML preconditioner on a Poisson problem

Initially, we will demonstrate the behaviour of the ML preconditioner for a Poisson problem, for the Lagrange elements of order 2,3 and 4. The results can be found in Figure 1.

We notice that the number of required iterations is oscillating for increasing spatial discretization. We can not expect the heuristic algorithms finding the coarser grids to

work equally well for all possible grids, and this is therefore not surprising. For the Lagrange elements of order 3 and 4, we notice a small increase in the condition number since we have not reached the asymptotic region yet. This can be reduced by tuning the aggregation parameters of ML. This has not been done, since the condition number already is very small. It is never larger then 1.4, 2.5 and 4.1 for the 2nd 3rd and 4th order element respectively.

4.2 Order optimality of the Bidomain block Gauss-Seidel preconditioner

Since the Runge-Kutta discretization of our problem (4)-(5) is in general nonsymmetric, we do not have a particular need for symmetric preconditioners. We will therefore use the lower block triangular Gauss-Seidel preconditioner for the Bidomain equation. Since it is not discussed in the paper by Mardal et al.[13], we first check the order-optimal behaviour with respect to the spatial discretization parameter h, and the timestep δt by numerical experiments. The result can be seen in Figure 2.

Again we notice that we have not reached the asymptotic limit yet, but BiCGStab never requires more then 15 iterations to reach convergence for our testproblem.

4.3 Order optimality of the Runge-Kutta block preconditioner

In this next experiment we show numerically that the Runge-Kutta block preconditioner is order optimal. We test for both the block Jacobian preconditioner $\mathcal{B}_{J,J}$ and for the block Gauss-Seidel preconditioner $\mathcal{B}_{GS,GS,GS}$ for the three stage Runge-Kutta RadauIIA scheme. The result can be seen in Figure 3

The results are similar for the other preconditioners, and for different numbers of Runge-Kutta quadrature nodes s.

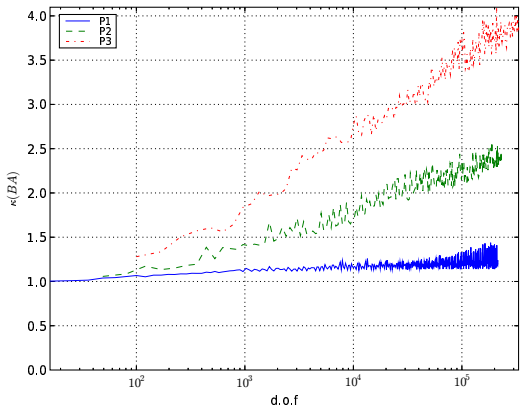


Figure 1: The condition number $\kappa(\mathcal{BA})$ for a Poisson problem $-\Delta u = f$. The example illustrates the behaviour of the ML preconditioner for an elliptic problem.

4.4 Numerical comparison of the different preconditioners

We will now compare the different preconditioners for different number of Runge-Kutta stages for both the RadauIIA and the LobattoIIIC family of schemes. This is done by

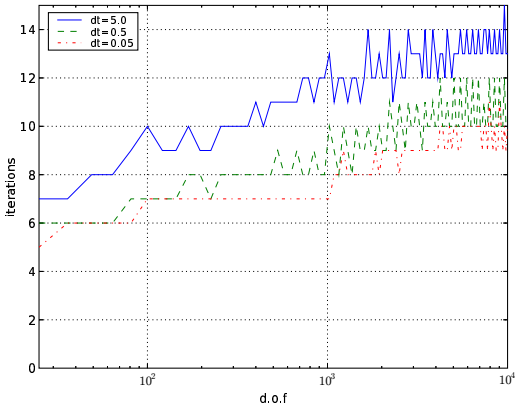
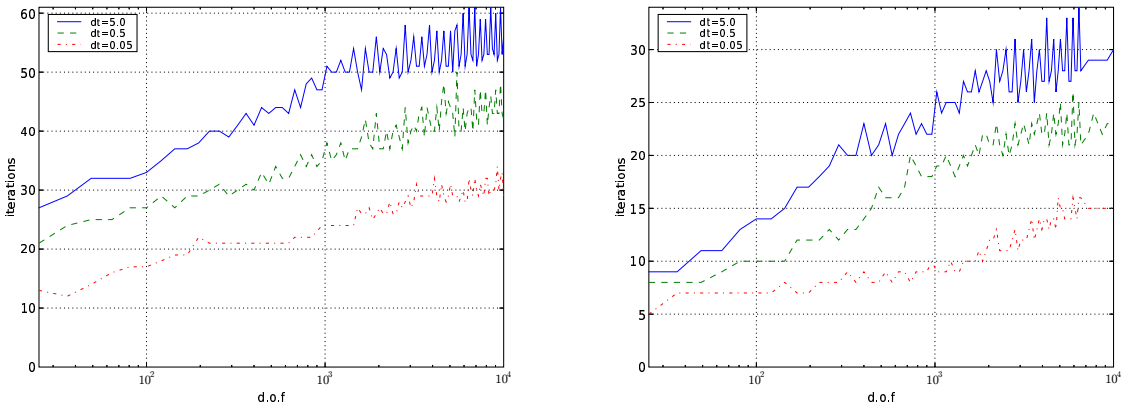


Figure 2: The number of iterations used by BiCGStab for the 2D problem (2)–(3) using linear finite elements in space, and implicit Euler in time. The preconditioner is the lower block diagonal Gauss-Seidel preconditioner. The figure indicate an order optimal behaviour with respect to the spatial discretization parameter h and the timestep δt.



(a) $\mathcal{B}_{J,J}$ (b) $\mathcal{B}_{GS,GS,GS}$

Figure 3: The number of iterations used by BiCGStab for the 2D problem (2)–(3) using linear finite elements in space, and three nodes RadauIIA method in time

solving (4)-(5) using linear finite elements in space, and the specified Runge-Kutta scheme in time. The grid is a triangulation of 201×201 nodes on the unit square. We choose a large timestep, $\delta t = 5.0$, to be close to the asymptotic limit. We count the number of iterations required for the right preconditioned BiCGStab to reach absolute error of 10^{-8}. The result can be seen in Table 1. As a point of reference, we compare with the number

	UV		standard					
Method	$\tilde{\mathcal{B}}_J$	$\tilde{\mathcal{B}}_{GS}$	$\tilde{\mathcal{B}}_{J,J}$	$\tilde{\mathcal{B}}_{J,GS}$	$\tilde{\mathcal{B}}_{GS,J,J}$	$\tilde{\mathcal{B}}_{GS,J,F}$	$\tilde{\mathcal{B}}_{GS,GS,GS}$	$\tilde{\mathcal{B}}_{GS,GS,F}$
RII 2	42	26	43	40	32	35	25	26
RII 3	75	45	74	87	49	58	42	44
RII 4	119	66	116	183	69	84	65	67
LC 2	55	31	54	49	38	42	31	31
LC 3	118	59	115	167	60	72	56	54
LC 4	253	94	256	566	99	125	98	96

Table 1: Number of iterations required for BiCGStab to reach absolute tolerance of 10^{-8}. As a point of reference, we compare with the solution of the implicit Euler discretization which requires 23, 16 and 18 iterations for block Jacobi, block Gauss-Seidel and symmetric block Gauss-Seidel respectively. *RII s* is the RadauIIA scheme with s stages, while *LC s* is the LobattoIIIC scheme, also with s stages. The $\mathcal{B}_{GS,GS,GS}$ seems to be the most efficient preconditioner for the RadauIIA schemes, while $\mathcal{B}_{GS,GS,F}$ seems to be slightly better for the LobattoIIIC schemes.

of iterations required for solving the same problem discretized with implicit Euler in time. By using a block Jacobi Bidomain preconditioner, BiCGStab uses 23 iterations. By using a lower block Gauss-Seidel Bidomain preconditioner, BiCGStab uses 16 iterations. Note that by using the symmetric Gauss-Seidel Bidomain preconditioner, BiCGStab uses 18 iterations, which is two iterations more then the lower block Gauss-Seidel preconditioner.

First we notice that the block Jacobi preconditioner for the UV formulation $\tilde{\mathcal{B}}_J$ performs comparable with the block Jacobi preconditioner for the standard formulation $\tilde{\mathcal{B}}_{J,J}$. This is no surprise since they are only permutations of each other. The difference between them is probably due to round-off errors.

The block Gauss Seidel preconditioners $\tilde{\mathcal{B}}_{GS}$ and $\tilde{\mathcal{B}}_{GS,GS,F}$ are also comparable in performance. $\tilde{\mathcal{B}}_{GS,GS,GS}$ reuses the Bidomain block structure for the off-diagonal Runge-Kutta blocks, and it is slightly better then $\tilde{\mathcal{B}}_{GS,GS,F}$ in the case of the RadauIIA method. This is probably because the increase in $\kappa(\tilde{\mathcal{B}}\mathcal{B}^{-1})$ due to more non-zeros blocks in (15) outweighs the decrease in $\kappa(\mathcal{B}\mathcal{A})$. The same behaviour is even more obvious for $\tilde{\mathcal{B}}_{GS,J,J}$ versus $\tilde{\mathcal{B}}_{GS,J,F}$. For the LobattoIIIC method, the $\tilde{\mathcal{B}}_{GS,GS,F}$ preconditioner is best.

An interesting observation is the deterioration of $\mathcal{B}_{J,GS}$ for increasing number of Runge-Kutta stages. The same behaviour is even more present for the $\tilde{\mathcal{B}}_{J,SGS}$, and the numbers are therefore not presented. This behaviour is not fully understood, but again we believe that this is because of the increase in the factor $\kappa(\tilde{\mathcal{B}}\mathcal{B}^{-1})$.

4.5 Timing results

Obviously our preconditioner is not order optimal with respect to the number of Runge-Kutta stages s. The question is whether the decrease in the number of required timesteps compensate for the increase in the number of iterations. We will try to answer this by comparing the wall clock time (wct) for a given test problem. We solve (2)-(3) with a source terms f such that the exact solution is

$$\begin{aligned} v(x,y,t) &= \sin(\omega_x x)\sin(\omega_y y)\sin(\omega_t t) \\ u(x,y,t) &= -\sin(\omega_x x)\sin(\omega_y y)\sin(\omega_t t) \\ (\omega_x,\omega_y,\omega_t) &= (\pi,\pi,20.5\pi), \quad \Omega = (-1,1)^2, \quad t \in [0,1] \end{aligned}$$

The high number of oscillation in time is used to create a certain degree of complexity in time. We also vary both of the discretization parameters h and p in space. The discretization parameters are chosen such that the L^2 error in time and space is less then 10^{-5}. We use right preconditioned BiCGStab with an absolute tolerance of 10^{-7}, and $\mathcal{B}_{GS,GS,GS}$ is used as preconditioner. The computation is done on a Intel Xeon 2.0GHz with 4GB RAM. The results can be seen in Table 2

		3rd order (5625)		4th order (3721)	
Method	δt	wct	k	wct	k
RII 1	$5 \cdot 10^{-7}$	10^5	1.0	10^5	1.0
RII 2	$2 \cdot 10^{-3}$	230	3.8	224	4.4
RII 3	$1 \cdot 10^{-2}$	151	7.5	141	8.3
RII 4	$2.5 \cdot 10^{-2}$	160	13.6	144	14.3

Table 2: The wall clock time (wct) measured in seconds, and the average number of iterations k for solving the 2D Bidomain equation (4)–(5) for RadauIIA schemes with various number of stages. The discretization parameters are chosen such that the errors from the discretizations are approximately 10^{-5}. We compute for both the 3rd and the 4rt order Lagrangian element. The number of required degrees of freedom in space is written in parenthesis. The preconditioner $\mathcal{B}_{GS,GS,GS}$ is approximated by using one algebraic multigrid V–cycle. The higher-order schemes outperforms the lower-order schemes.

For this small testproblem, the higher-order schemes outperforms the lower-order. Note that these results depend on the chosen problem, the order of accuracy required, the implementation etc., and is therefore not necessarily true in all cases. But it is an indication that the saving in the number of required timesteps outweighs the increase in the number of iterations. We have only used the previous timestep as initial guess for the linear solver. Additional saving can be made for higher-order schemes by using a better guess. By choosing a more strict error tolerance, the higher-order schemes would have been even more favourable.

5 Final Remarks

In this paper we have shown that the systems arising from fully implicit Runge-Kutta schemes applied to Bidomain equations (2)-(3) can be efficiently preconditioned using well known preconditioners for parabolic and elliptic problems. The preconditioning block elements are arranged in a block Jacobi or a block Gauss-Seidel fashion. Suitable block preconditioning elements may be multigrid, domain decomposition or other standard preconditioners.

In several numerical experiments we have indicated that the block preconditioner for the Runge-Kutta discretization is order optimal, assuming that the preconditioner for the blocks is so. We have also shown that the increase in the number of required iterations for higher-order schemes is inferior to the decrease in number of required timesteps. This shows that higher-order Runge-Kutta schemes are beneficial in computing the Bidomain equation (4)-(5).

In[17], we introduced a generalized block Gauss-Seidel preconditioner where the Runge-Kutta coefficient matrix A is changed with $\tilde{A}$ in the preconditioner. $\tilde{A}$ was found by minimizing the criterion

$$\kappa(\tilde{A}^{-1}A)$$

where $\tilde{A}$ was a lower triangular matrix. This resulted in a reduction of the condition number of a factor up to 5 (for the six stage RadauIIA scheme). We tried this optimization principle for the Bidomain equation as well, but it only made the condition number larger. We also tried to optimize for the full problem $\kappa(\mathcal{BA})$ instead for the simplified optimization principle $\kappa(\tilde{A}^{-1}A)$, but the reduction in the number of BiCGStab iterations was never more then two. The reason for this is not fully understood, but is probably due to the conductivities' influence on the condition number.

We have only shown numerical results for the RadauIIA and the LobattoIIIC family of the Runge-Kutta schemes. But the only assumptions we have made are that the scheme have to be stiffly accurate, and that the Runge-Kutta coefficient matrix should be invertible. This means that the presented preconditioners also works for other Runge-Kutta schemes satisfying these two properties.

The presented preconditioners will not work on the LobattoIIIA family[9], since the the Runge-Kutta coefficient matrix A is not invertible in this case. The equation for the algebraic constraint is then (8) instead of (7) leading to a $\mathcal{A}$ being a dense block matrix. The preconditioners can be modified thereafter, but the condition number will in general be larger.

References

[1] Ml – Multilevel Preconditioning Package.
http://software.sandia.gov/trilinos/packages/ml/index.html.

[2] PyCC – Python Cardiac Computation. http://heim.ifi.uio.no/∼skavhaug/heart_simulations.html.

[3] SyFi – Symbolic Finite Elements. http://syfi.sf.net.

[4] Uri M. Ascher and Linda R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM, 1998.

[5] S.C. Brenner and L.R. Scott. *The Mathematical Theory of Finite Element Methods*. Number 15 in Texts in Applied Mathematics. Springer, 2nd edition, 2002.

[6] Lawrence C. Evans. *Partial Differential Equations*. Number 19. American Mathematical Society, 1998.

[7] Wolfgang Hackbusch. *Iterative Solution of Large Sparse Systems of Equations*. Number 95. Springer Verlag, 1994.

[8] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I - Nonstiff Problems*. Springer Verlag, 2nd edition, 1992.

[9] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II - Stiff and Differential-Algebraic Problems*. Springer Verlag, 2nd edition, 1996.

[10] Ernst Hairer and Gerhard Wanner. Stiff differential equations solved by Radau methods. *Journal of Computational and Applied Mathematics*, 111:93–111, 1999.

[11] J.E. Dennis Jr. and R.B. Schnabel. A View of Unconstrained Optimization. In G.L. Nemhauser, A.H.G. Rinnooy Kan, and H.J. Todd, editors, *Optimization*, pages 1–72. Elsevier, 1989.

[12] J. Van lent and S. Vandewalle. Multigrid methods for implicit Runge-Kutta and boundary value method discretizations of PDEs. to appear in SIAM J. Sci. Comput, 2004.

[13] K.A. Mardal, B. F. Nielsen, X. Cai, and A. Tveito. An order optimal solver for the discretized Bidomain equations. Simula Research Laboratory, Research Report 04–2005. URL:http://www.simula.no/departments/scientific/publications/Mardal.2005.2, 2005.

[14] K.A. Mardal, T.K. Nilssen, and G.A. Staff. Order optimal preconditioners for implicit Runge-Kutta schemes applied to parabolic PDE's. Simula Research Laboratory, Research Report 08–2005. Submitted to SIAM J. Sci. Computing.

[15] C. Runge. Über die numerische Auflösung von Differentialgleichungen. *Math, Ann,*, 46:167–178, 1895.

[16] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2nd edition, 2003.

[17] Gunnar A. Staff, Kent-Andre Mardal, and Trygve K. Nilssen. Preconditioning of fully implicit Runge-Kutta schemes for parabolic PDEs. *Modeling, Identification and Control*, 27(2):109–123, 2006.

[18] J. Sundnes, G. T. Lines, and A. Tveito. An operator splitting method for solving the Bidomain equations coupled to a volume conductor model for the torso. *Mathematical biosciences*, 194(2):233–248, 2005.

[19] J. Sundnes, G.T. Lines, X. Cai, B.F. Nielsen, K.-A. Mardal, and A. Tveito. *Computing the Electrical Activity in the Heart*. Springer, 2006.

[20] J. Sundnes, G.T. Lines, K-.A. Mardal, and A. Tveito. Multigrid block preconditioning for the coupled Bidomain and forward problem. *Computer Methods in Biomechanics and Biomedical Engineering*, 5(6):397–409, 2003.

[21] V. Thomée. *Galerkin Finite Element Methods for Parabolic Problems*, volume 2nd. Springer-Verlag, 1997.