Available online at www.sciencedirect.com

ScienceDirect

Journal of Computational Physics 227 (2008) 5272–5285

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

A stabilized explicit Lagrange multiplier based domain decomposition method for parabolic problems [☆]

Zheming Zheng [a,*], Bernd Simeon [b], Linda Petzold [a]

[a] *Department of Mechanical Engineering, University of California Santa Barbara, Santa Barbara, CA 93106, USA*
[b] *Zentrum Mathematik, Technische Universität München, Boltzmannstrasse 3, D-85748 Garching bei München, Germany*

Received 22 June 2007; received in revised form 23 January 2008; accepted 24 January 2008
Available online 14 February 2008

Abstract

A fully explicit, stabilized domain decomposition method for solving moderately stiff parabolic partial differential equations (PDEs) is presented. Writing the semi-discretized equations as a differential-algebraic equation (DAE) system where the interface continuity constraints between subdomains are enforced by Lagrange multipliers, the method uses the Runge–Kutta–Chebyshev projection scheme to integrate the DAE explicitly and to enforce the constraints by a projection. With mass lumping techniques and node-to-node matching grids, the method is fully explicit without solving any linear system. A stability analysis is presented to show the extended stability property of the method. The method is straightforward to implement and to parallelize. Numerical results demonstrate that it has excellent performance.
© 2008 Elsevier Inc. All rights reserved.

Keywords: Non-overlapping domain decomposition; Lagrange multiplier; Runge–Kutta–Chebyshev; Stability; Parallelization; Parabolic PDE

1. Introduction

Domain decomposition methods (DDM) for the solution of partial differential equations (PDEs) have attracted a great deal of research interest, due to their flexibility for both multiphysics and parallel computing. For complex multiphysics problems, domain decomposition methods help simplify the problem and reduce the complexity. Then different strategies can be applied to different subdomains for best performance. The DDM can be cast into two categories: overlapping domain decomposition methods, including the classical Schwarz alternating method [1] and the two-level overlapping methods – the additive/multiplicative Schwarz methods [2–8]; and non-overlapping domain decomposition methods, including the Dirichlet–Neumann algorithm [9–12], the Neumann–Neumann algorithm [12–16] and the Lagrange multiplier based finite element

☆ This work was supported by the National Science Foundation under NSF award NSF/ITR CCF-0428912.
* Corresponding author.
E-mail addresses: zhengzhm@engineering.ucsb.edu (Z. Zheng), simeon@ma.tum.de (B. Simeon), petzold@engineering.ucsb.edu (L. Petzold).

tearing and interconnecting (FETI) method [17–21]. In the Lagrange multiplier based methods, the problem domain is decomposed into non-overlapping subdomains where the interface continuity condition between subdomains is enforced via Lagrange multipliers.

Most of the above methods are designed for elliptic PDEs. Parabolic problems can be solved by implicit semi-discretization in time, yielding an elliptic problem at each time step [22,23]. Parabolic problems can also be solved by fractional step methods [24–27]. A class of methods for solving parabolic problems with a finite difference discretization is the hybrid explicit–implicit scheme, including Dawson's method [28,29], the explicit prediction and implicit correction (EPIC) method [30] and the implicit prediction and implicit correction (IPIC) method [31]. In the above explicit–implicit schemes, the unknowns at the interior of each subdomain are solved implicitly, while the unknowns at the interface are treated differently. Dawson et al. [28,29] calculated the interface values with a half implicit, half explicit scheme. However Dawson's algorithm is only conditionally stable [31]. Zhu and Qian [30] proposed the EPIC method, in which an explicit predictor is first used to obtain the interface values. Then the interior subdomains are solved for new interior values, and finally the interface values are corrected with new interior values by an implicit scheme. The EPIC method has better stability than Dawson's method. To achieve even better stability, Jun and Mai [31] proposed the IPIC method which replaces the explicit predictor in the EPIC method with an implicit scheme.

In this paper we develop a fully explicit but stabilized domain decomposition method for solving parabolic problems. A major concern for explicit methods is stability. Stabilized explicit methods, e.g. the Runge–Kutta–Chebyshev (RKC) method [32–34] for solving moderately stiff ordinary differential equations (ODEs) and the Runge–Kutta–Chebyshev projection (RKCP) method [35] for solving differential-algebraic equations (DAEs) have been proposed. In the class of Runge–Kutta–Chebyshev methods only two stages are used to obtain second-order accuracy and all other stages are exploited to extend the real stability boundary along the negative real axis. Thus the RKC method is best for diffusion-dominated problems, but a small convection (or other non-diffusion) term is allowed by introducing a damping factor [36]. For reaction–diffusion problems, an implicit–explicit (IMEX) extension of the explicit RKC method has been proposed [37,36], where the diffusion terms are treated explicitly by the RKC algorithm and the highly stiff reaction terms implicitly. When the system is convection-dominated, a semi-Lagrangian formulation can be used with the RKC method to form a finite element semi-Lagrangian explicit RKC method [38] for solving convection-dominated reaction–diffusion problems. A variable step-size selection strategy is also presented in the RKC method [34], which can be easily extended to the RKCP method.

In the explicit domain decomposition method of this paper, we decompose the problem domain into non-overlapping subdomains, discretize the parabolic equation with the finite element method (FEM) and enforce the interface continuity condition via Lagrange multipliers. Hence we formulate the problem into a DAE system which is to be solved by the RKCP method. By applying the mass lumping technique [39], it is possible and very promising to develop a stabilized explicit domain decomposition finite element method. In fact the explicit domain decomposition method also works for finite difference discretization.

The rest of this paper is organized as follows. In Section 2 the domain decomposition problem is defined with an example of two subdomains for a time-dependent heat equation. The problem is then formulated into a DAE through weak form and finite element discretization. In Section 3 we present the explicit integration scheme for the DAE and the stability analysis of this method, as well as a discussion of parallelization issues. Numerical results are presented in Section 4, followed by conclusions in Section 5.

2. Background

2.1. Model problem

In this section we briefly review the domain decomposition method for a parabolic problem. The parabolic problems we consider in this paper are diffusion-dominated, thus the stiffness comes mainly from the diffusion term. For simplicity we use the heat equation with a 2-subdomain decomposition as an example in the following. Consider the domain $\Omega \subset \mathbb{R}^d$ where d is the dimension and the heat equation

$$u_t = \Delta u + f(u,t) \tag{1}$$

for the unknown heat distribution $u(x,t)$ in the domain Ω. For simplicity of presentation and without loss of generality, we neglect the dependence of f on u whenever it is appropriate. We assume zero Dirichlet boundary conditions $u = 0$ on $\partial\Omega$ and initial values $u(x,0) = u_0(x)$.

A standard approach to solve Eq. (1) is the method of lines, e.g. in combination with finite elements in space. Eq. (1) is replaced by its weak form and the solution is projected onto a finite dimensional subspace of $H^1(\Omega)$. Thus we set $u(x,t) \doteq N(x)\mathbf{y}(t)$ with basis functions written in matrix notation as $N(x)$ and unknown coefficient vector $\mathbf{y}(t)$ (the nodal variables in the FEM), to arrive at an ODE system

$$M\mathbf{y}_t = -A\mathbf{y} + \mathbf{b} \tag{2}$$

with mass matrix M, stiffness matrix A, vector $\mathbf{b}$ corresponding to the discretization of $f(u,t)$, and initial conditions $\mathbf{y}(0) = \mathbf{y}_0$. If mass lumping is used, the mass matrix M is diagonal, hence explicit time integration becomes attractive.

In the next step, we assume that the domain Ω is decomposed into several non-overlapping subdomains, for example two subdomains Ω_1 and Ω_2,

$$\Omega = \Omega_1 \cup \Omega_2$$

with common interface Γ_I, shown in Fig. 1. Formulating a non-overlapping domain decomposition, we seek the solution $u_1(x,t)$ on Ω_1 and $u_2(x,t)$ on Ω_2 such that the heat equation holds on both subdomains along with boundary and interface conditions [12]

$$u_{1,t} = \Delta u_1 + f_1 \quad \text{in } \Omega_1, \tag{3a}$$

$$u_{2,t} = \Delta u_2 + f_2 \quad \text{in } \Omega_2, \tag{3b}$$

$$u_1 = 0 \quad \text{on } \Gamma_1, \tag{3c}$$

$$u_2 = 0 \quad \text{on } \Gamma_2, \tag{3d}$$

$$u_1 = u_2 \quad \text{on } \Gamma_I, \tag{3e}$$

$$\frac{\partial u_1}{\partial v_1} = -\frac{\partial u_2}{\partial v_2} \quad \text{on } \Gamma_I, \tag{3f}$$

where v_1 and v_2 ($v_1 = -v_2$) are the outward normal vectors on the interface Γ_I towards Ω_1 and Ω_2, respectively. We denote the external boundaries of the subdomains by $\partial\Omega = \Gamma_1 \cup \Gamma_2$. Formally, under suitable assumptions on the domain, we have $u|_{\Omega_1} = u_1$ and $u|_{\Omega_2} = u_2$. In other words, the solutions of the original problem (1) and of the domain-decomposed problem (3) are equivalent.

2.2. Weak form and finite element discretization

In this section we transform the problem (3) to weak form and apply the method of lines. Test functions v_i defined in Ω_i are elements of $H^1(\Omega_i)$ and vanish on the boundary Γ_i, $i = 1,2$. The same holds for $u_1(\cdot,t)$ and

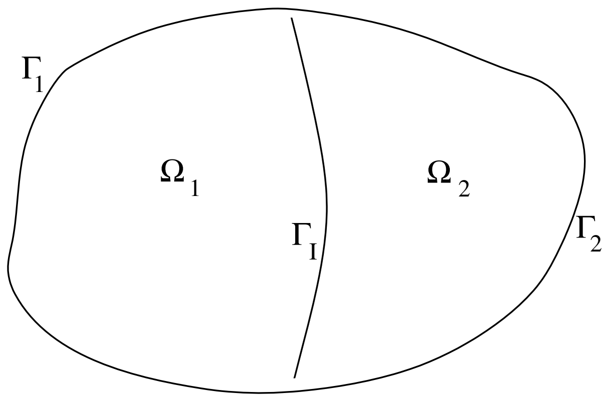

Fig. 1. Decomposed domain with interface Γ_I.

$u_2(\cdot, t)$ at each instance of time t. On the interface Γ_I, however, test functions and subdomain solutions do not vanish.

Application of Green's theorem gives

$$\int_{\Omega_i} \langle \nabla u_i, \nabla v_i \rangle \mathrm{d}x - \int_{\Gamma_I} v_i \langle \nabla u_i, v_i \rangle \mathrm{d}s = -\int_{\Omega_i} v_i \Delta u_i \mathrm{d}x, \tag{4}$$

and the weak form of the domain decomposition problem reads

$$\int_{\Omega_1} v_1 u_{1,t} \mathrm{d}x + \int_{\Omega_1} \langle \nabla v_1, \nabla u_1 \rangle \mathrm{d}x + \int_{\Gamma_I} v_1 \lambda \mathrm{d}s = \int_{\Omega_1} v_1 f_1 \mathrm{d}x \quad \text{for all } v_1, \tag{5a}$$

$$\int_{\Omega_2} v_2 u_{2,t} \mathrm{d}x + \int_{\Omega_2} \langle \nabla v_2, \nabla u_2 \rangle \mathrm{d}x - \int_{\Gamma_I} v_2 \lambda \mathrm{d}s = \int_{\Omega_1} v_2 f_2 \mathrm{d}x \quad \text{for all } v_2, \tag{5b}$$

$$\int_{\Gamma_I} \tau (u_1 - u_2) \mathrm{d}s = 0 \quad \text{for all } \tau, \tag{5c}$$

where v_1, v_2 and τ are the test functions and

$$\lambda = -\langle \nabla u_1, v_1 \rangle = \langle \nabla u_2, v_2 \rangle,$$

defined on Γ_I, is the Lagrange multiplier. We can thus view the Lagrange multiplier as a negative flux (or contact pressure) through the interface. By construction, the Neumann boundary condition (3f) is automatically satisfied. While the Dirichlet boundary conditions (3c) and (3d) have already been included in this formulation by the choice of v_1 and v_2, the interface condition $u_1 - u_2 = 0$ is still present in the form of the weak constraint Eq. (5c). The correct function space for the test function $\tau(x)$ as well as for the Lagrange multiplier $\lambda(x, t)$ (with time t frozen) would be the dual of the trace space $H^{1/2}(\Gamma_I)$.

Eq. (5) form a time-dependent saddle point problem or a partial differential-algebraic equation (PDAE). We can write the system in the abstract form as

$$\begin{aligned} \mathbf{u}_t + A\mathbf{u} + B^T \boldsymbol{\lambda} &= \mathbf{l}, \\ B\mathbf{u} &= 0, \end{aligned}$$

with $\mathbf{u} = (u_1, u_2)$ and corresponding operators A, B.

As above in the unconstrained problem, finite dimensional subspaces are chosen to project the solution. While the subdomain variables can be treated as before by a finite element ansatz, i.e., $u_1(x, t) \doteq N_1(x)\mathbf{y}_1(t)$ and $u_2(x, t) \doteq N_2(x)\mathbf{y}_2(t)$, the Lagrange multiplier λ on the boundary requires special attention. Several possibilities exist.

- *Matching grids*
  If the grids for Ω_1 and Ω_2 match on the interface Γ_I, the simplest choice is node-to-node constraints. This means that the integral $\int_{\Gamma_I} \tau(u_1 - u_2)\mathrm{d}s$ is replaced by the discrete sum $\sum_i \{\tau(u_1 - u_2)\}|_{x_i}$, with the index i running over all nodes on Γ_I. In turn, we get the pointwise condition $u_1(x_i, t) = u_2(x_i, t)$ at all interface nodes, and the Lagrange multiplier $\lambda(x, t)$ is discretized to a vector $\lambda(t)$ defined on the interface. One could also speak of a quadrature rule for the boundary integral.
- *Non-matching grids*
  Unlike the matching grids method, the non-matching grids method does not share nodes points between subdomains on the interface. The continuity between subdomains can be enforced by the Lagrange multiplier method. However a special finite element space for the Lagrange multiplier must be chosen to stabilize the domain decomposition problem.
- *Mortar element*
  The mortar method [40] is related to the above approach. It adds extra flexibility, in particular for non-matching grids.

We skip the details here and proceed with the resulting DAE system. All the above three methods reach the following form:

$$M_1\mathbf{y}_{1,t} = -A_1\mathbf{y}_1 + \mathbf{b}_1 - B_1^T\lambda, \tag{6a}$$

$$M_2\mathbf{y}_{2,t} = -A_2\mathbf{y}_2 + \mathbf{b}_2 + B_2^T\lambda, \tag{6b}$$

$$0 = B_1\mathbf{y}_1 - B_2\mathbf{y}_2, \tag{6c}$$

or

$$M\mathbf{y}_t = -A\mathbf{y} + \mathbf{b} - B^T\lambda, \tag{7a}$$

$$0 = B\mathbf{y}, \tag{7b}$$

where

$$M = \begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix}, \quad A = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix}, \quad B = [B_1 \quad -B_2], \quad \mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}.$$

Here, M_i and A_i are the mass and stiffness matrices for the subdomains, and the interface discretization leads to constraint matrices B_1 and B_2. If the matrix B has full rank, Eq. (6) or Eq. (7) is a DAE of index 2 with block-diagonal structure in the differential equations. Moreover, in the case of mass lumping, the mass matrices are diagonal. Hence explicit time integration becomes an option.

3. Explicit integration method

3.1. RKCP

In this section we introduce an explicit method for integrating Eq. (7). The RKC method [32–34], an explicit method designed for solving moderately stiff ODEs, has good stability properties while maintaining second-order accuracy. The real stability boundary of the RKC method increases quadratically with the number of stages used. Hence it can use a much larger time step-size than traditional explicit Runge–Kutta methods. Combining the RKC method with the projection method [41,42], the RKCP method [35] was developed as an explicit method for solving DAEs, and it was shown that it preserves the zero-stability and absolute stability properties of the RKC method. Thus we use the RKCP method to integrate the DAE (6). The RKCP algorithm for integrating from $\{\mathbf{y}^n, \lambda^n\}$ at t^n to $\{\mathbf{y}^{n+1}, \lambda^{n+1}\}$ at t^{n+1} can be formulated as follows:

$$\begin{aligned}
&\mathbf{Y}^0 = \mathbf{y}^n,\\
&\mathbf{F}^0 = -M^{-1}A\mathbf{Y}^0 + M^{-1}\mathbf{b} - M^{-1}B^T\lambda^n,\\
&\mathbf{Y}^1 = \mathbf{Y}^0 + \tilde{\mu}_1\Delta t\mathbf{F}^0,\\
&\mathbf{F}^{j-1} = -M^{-1}A\mathbf{Y}^{j-1} + M^{-1}\mathbf{b} - M^{-1}B^T\lambda^n,\\
&\mathbf{Y}^j = (1-\mu_j-\nu_j)\mathbf{Y}^0 + \mu_j\mathbf{Y}^{j-1} + \nu_j\mathbf{Y}^{j-2} + \tilde{\mu}_j\Delta t\mathbf{F}^{j-1} + \tilde{\gamma}_j\Delta t\mathbf{F}^0, \quad (j = 2,3,\ldots,s).\\
&\text{Solve } (BM^{-1}B^T)\phi = B\mathbf{Y}^s \quad \text{for } \phi.\\
&\text{Update } \mathbf{y}^{n+1} = \mathbf{Y}^s - M^{-1}B^T\phi.\\
&\text{Update } \lambda^{n+1} = \lambda^n + \frac{2\phi}{\Delta t}.
\end{aligned} \tag{8}$$

Here s is the number of Runge–Kutta stages and $\mu_j, \tilde{\mu}_j, \nu_j, \tilde{\gamma}_j$ are the coefficients, determined for accuracy and stability of the method. What the above algorithm does is to use the RKC method to integrate Eq. (7a) from t^n to t^{n+1} without regard to the constraint (7b) with a "frozen" λ value at t^n, i.e., λ^n. At the last stage of the RKC method, a projection procedure is performed to update $\mathbf{y}$ so that it satisfies the constraint (7b) and to update λ^n to λ^{n+1}.

Choosing the matching grids method, the continuity constraint (6c) reads

$$(\mathbf{y}_1)_{\Gamma_I} - (\mathbf{y}_2)_{\Gamma_I} = 0, \tag{9}$$

where $(\mathbf{y}_1)_{\Gamma_I}$ and $(\mathbf{y}_2)_{\Gamma_I}$ denote the interface nodes on Γ_I of $\mathbf{y}_1$ and $\mathbf{y}_2$, respectively, and Eqs. (6a) and (6b) can be elaborated as

$$M_1\mathbf{y}_{1,t} = \begin{cases} 0 & \text{Zero Dirichlet boundary condition on } \Gamma_1 \\ -A_1\mathbf{y}_1 + \mathbf{b}_1 & \text{Interior nodes in } \Omega_1 \\ -A_1\mathbf{y}_1 + \mathbf{b}_1 - \lambda & \text{Interface nodes on } \Gamma_I \end{cases}, \tag{10}$$

and

$$M_2\mathbf{y}_{2,t} = \begin{cases} 0 & \text{Zero Dirichlet boundary condition on } \Gamma_2 \\ -A_2\mathbf{y}_2 + \mathbf{b}_2 & \text{Interior nodes in } \Omega_2 \\ -A_2\mathbf{y}_2 + \mathbf{b}_2 - \lambda & \text{Interface nodes on } \Gamma_I \end{cases}. \tag{11}$$

If additionally mass lumping techniques [39,43] are used in the finite element discretization, the mass matrices M_1 and M_2 can be replaced by diagonal matrices $\widetilde{M}_1$ and $\widetilde{M}_2$. Approximating a consistent mass matrix (M_1, M_2) with a lumped mass matrix $(\widetilde{M}_1, \widetilde{M}_2)$ will reduce the accuracy of the finite element method, however it is shown in [39] that the lumped mass matrix method converges with the order $O(h^2)$ where h is the size of the finite elements. Thus the loss of spatial accuracy can be recovered by a refined grid. With a diagonal mass matrix, this method yields a fully explicit domain decomposition finite element method. For linear elements, $\widetilde{M}_1$ (or $\widetilde{M}_2$) can be obtained simply by a row (or column) summation of M_1 (or M_2) and assigning the results onto the corresponding diagonal positions.

With the above simplifications, the projection step in scheme (8) is rewritten as

$$[(\widetilde{M}_1^{-1})_{\Gamma_I} + (\widetilde{M}_2^{-1})_{\Gamma_I}]\phi = (\mathbf{Y}_1^s)_{\Gamma_I} - (\mathbf{Y}_2^s)_{\Gamma_I}, \tag{12}$$

where $(\cdot)_{\Gamma_I}$ denotes the part of vectors or matrices defined on Γ_I. Since both $\widetilde{M}_1$ and $\widetilde{M}_2$ are diagonal, $[(\widetilde{M}_1^{-1})_{\Gamma_I} + (\widetilde{M}_2^{-1})_{\Gamma_I}]$ is also diagonal. It is trivial to solve Eq. (12). Hence the RKCP method applied to the domain decomposition problem (6) is fully explicit. After solving the above equation for ϕ, only the variables on the interface Γ_I need to be updated,

$$(\mathbf{y}_1^{n+1})_{\Gamma_I} = (\mathbf{Y}_1^s)_{\Gamma_I} - (\widetilde{M}_1^{-1})_{\Gamma_I}\phi, \tag{13}$$

$$(\mathbf{y}_2^{n+1})_{\Gamma_I} = (\mathbf{Y}_2^s)_{\Gamma_I} - (\widetilde{M}_2^{-1})_{\Gamma_I}\phi, \tag{14}$$

$$\lambda^{n+1} = \lambda^n + \frac{2\phi}{\Delta t}. \tag{15}$$

Overall, the RKCP method applied to the domain decomposition method is a fully explicit integrator, requiring no Newton iteration or linear system solver. At each time step, it consists of s explicit stages plus a final projection step which "synchronizes" the subdomains. It thus can be viewed as a fractional-step method which includes an explicit integration step and a final synchronization step.

The algorithm has an excellent potential for parallelization. In the integration step, the computation in each subdomain is totally independent, hence no communication between processors (subdomains) is needed. In the synchronization step, a continuity constraint must be satisfied on each interface which requires information exchange between the subdomains bordering on the same interface. In other words, the communication is local, only across the interface between the two subdomains. Therefore the parallelization should be highly scalable. Because the computation in each subdomain is totally independent, the method is also highly suitable for multiphysics problems, where the model and/or the discretization may be different in different subdomains.

3.2. Stability analysis

In this section we analyze the zero-stability and the absolute stability properties of the RKCP method applied to the domain decomposition problem. We show that the method is zero-stable, and that, for stiff problems, its stability region is at least as large as that of the corresponding RKC method.

Without loss of generality, we neglect the forcing term $\mathbf{b}$ in Eq. (7) and point out that the following analysis holds both for consistent and lumped mass matrices. One step integration of the ODE (7a) (with λ frozen to λ^n) by the RKC method from t^n to t^{n+1} yields the intermediate solution [35]

$$\mathbf{y}_*^{n+1} = P_s(-\Delta t M^{-1}A)\mathbf{y}_n + [I - P_s(-\Delta t M^{-1}A)](-M^{-1}A)^{-1}M^{-1}B^T\lambda_n, \tag{16}$$

where P_s is the stability polynomial of the RKC method, in the form of [33]

$$P_s(z) = 1 + z + \frac{z^2}{2} + \sum_{k=3}^{s} C_k z^k, \tag{17}$$

with C_k being the coefficients. Note here that M must be nonsingular for M^{-1} to exist, however A can be singular, because $[I - P_s(-\Delta t M^{-1}A)](-M^{-1}A)^{-1}$ is a polynomial of A with non-negative exponents. The solution $\mathbf{y}^{n+1}$ is obtained by a projection of $\mathbf{y}_*^{n+1}$, written as

$$\mathbf{y}^{n+1} = (I - Q)\mathbf{y}_*^{n+1}, \tag{18}$$

where $Q = M^{-1}B^T(BM^{-1}B^T)^{-1}B$. Denoting $P_s(-\Delta t M^{-1}A)$ as R, it is derived from Eq. (16) that

$$\mathbf{y}_*^{n+1} - \mathbf{y}_*^n = R(\mathbf{y}^n - \mathbf{y}^{n-1}) + (I - R)(-M^{-1}A)^{-1}M^{-1}B^T(\lambda^n - \lambda^{n-1}). \tag{19}$$

Since $\lambda^n - \lambda^{n-1} = \frac{2\phi}{\Delta t}$, where $\phi = (BM^{-1}B^T)^{-1}B\mathbf{y}_*^n$, Eq. (19) is rewritten as

$$\mathbf{y}_*^{n+1} - \mathbf{y}_*^n = R(I - Q)(\mathbf{y}_*^n - \mathbf{y}_*^{n-1}) + (I - R)\frac{2}{\Delta t}(-M^{-1}A)^{-1}Q\mathbf{y}_n^*. \tag{20}$$

Let $\mathbf{y}_*^n = \mathbf{v}^n + \mathbf{w}^n$, where $\mathbf{v}^n = (I - Q)\mathbf{y}_*^n$ and $\mathbf{w}_n = Q\mathbf{y}_*^n$. We rewrite Eq. (20) as

$$\mathbf{v}^{n+1} + \mathbf{w}^{n+1} - \mathbf{v}^n - \mathbf{w}^n = R(\mathbf{v}^n - \mathbf{v}^{n-1}) + (I - R)\frac{2}{\Delta t}(-M^{-1}A)^{-1}\mathbf{w}^n. \tag{21}$$

Consider the following two recurrences:

$$\mathbf{v}^{n+1} - \mathbf{v}^n = R(\mathbf{v}^n - \mathbf{v}^{n-1}), \tag{22}$$

$$\mathbf{w}^{n+1} = \left(I + (I - R)\frac{2}{\Delta t}(-M^{-1}A)^{-1}\right)\mathbf{w}^n. \tag{23}$$

The propagation matrices for the recurrences (22) and (23) are R and $I + (I - R)\frac{2}{\Delta t}(-M^{-1}A)^{-1}$, respectively.

Expanding $R = P_s(-\Delta t M^{-1}A)$ in the form of Eq. (17) and considering Δt small, we have

$$R = I - \Delta t M^{-1}A + \frac{\Delta t^2}{2}(M^{-1}A)^2 + \mathrm{O}(\Delta t^3), \tag{24}$$

and thus

$$I + (I - R)\frac{2}{\Delta t}(-M^{-1}A)^{-1} = -I + \Delta t M^{-1}A + \mathrm{O}(\Delta t^2). \tag{25}$$

Zero-stability for recurrences (22) and (23), and hence for the RKCP method, is easily established, following the standard theory for numerical ODEs (e.g. [44]).

For absolute stability, let $z = \Delta t \lambda_s$, where λ_s denotes an eigenvalue of $-M^{-1}A$. It can be shown that for z in the stability region of the RKC method, i.e., $\|P_s(z)\| \leqslant 1$, we always have $-1 \leqslant \frac{1-P_s(z)}{z} \leqslant 0$. It follows that $-1 \leqslant 1 + 2\frac{1-P_s(z)}{z} \leqslant 1$. So we see that both recurrence (22) and recurrence (23) are stable. The sum of these two recurrences, Eq. (21), is therefore stable.

Stability analysis for a general projection method (implicit or explicit) can be found in [45]. It is mentioned in [46] that mass lumping may affect the dissipation and dispersion properties of the classical time integration methods. It is correct that mass lumping affects the eigenvalues and thus leads to a somewhat different stability of the semi-discretized equations. In second-order problems (e.g. elasto-dynamics) where we have mostly imaginary eigenvalues, this shift of the eigenvalues can be a severe difficulty, in particular if resonance frequencies for vibrational analysis play a role. However, in our first-order problem (parabolic case), we have mostly negative real eigenvalues, and the mass lumping results in perturbations of decaying exponential functions. As the problem is stable anyway, the perturbations have no strong influence on the solution behavior. Yet they do influence the behavior of explicit integrators as the stability regions change somewhat. Thus the stability analysis in this section can be extended to the lumped mass matrix case for the parabolic problems we consider.

4. Numerical examples

4.1. 1D Brusselator problem

We first applied our method to the following 1D Brusselator problem on $\Omega = [0, 1]$, using a finite difference discretization:

$$\frac{\partial u}{\partial t} = D_1 \frac{\partial^2 u}{\partial x^2} + \alpha - (\beta + 1)u + u^2 v, \tag{26a}$$

$$\frac{\partial v}{\partial t} = D_2 \frac{\partial^2 v}{\partial x^2} + \beta u - u^2 v, \tag{26b}$$

with boundary conditions

$$u(0,t) = u(1,t) = \alpha,$$
$$v(0,t) = v(1,t) = \beta/\alpha,$$

and initial conditions

$$u(x,0) = \alpha + x(1-x),$$
$$v(x,0) = \beta/\alpha + x^2(1-x).$$

The problem domain Ω was decomposed into two non-overlapping subdomains $\Omega_1 = [0, 0.5]$ and $\Omega_2 = [0.5, 1]$ which in general may have different physics, namely different diffusion coefficients. The parameters were set as $D_1 = D_2 = \frac{1}{40}$ in Ω_1, $D_1 = D_2 = \frac{1}{800}$ in Ω_2, and $\alpha = 0.6$, $\beta = 2$ in both Ω_1 and Ω_2. The problem domain was discretized in space with a second-order central difference for the diffusion term with spacing $\Delta x = 0.01$. The time span for the simulation is $T = [0, 32]$.

We implemented the adaptive Runge–Kutta–Chebyshev projection method (see [34] for the adaptive strategy) in Matlab scripts, called ARKCP and compared the results with the Matlab ODE solvers ODE15S, ODE45 and ODE23. In the case of the Matlab ODE solvers, the problem (26) was solved as an ODE by semi-discretization in space. This can be viewed as an overlapping domain decomposition method where the interface continuity constraint is implemented as a boundary condition for each subdomain. We supplied a sparse Jacobian matrix for the implicit ODE solver ODE15S. In the tests shown in Table 1, the error is a vector of u error and v error, computed as $\frac{\|\mathbf{u}-\mathbf{u}_{\text{ref}}\|_2}{\|\mathbf{u}_{\text{ref}}\|_2}$, where $\mathbf{u}$ is the numerical solution and $\mathbf{u}_{\text{ref}}$ is the reference solution which is computed by ODE15S with a very small error tolerance (RTOL=ATOL=1.0E−13). Under the condition that each method yielded the same level of accuracy, we compared the computation times. We see that from Table 1 ARKCP is much faster than the explicit solvers ODE45 and ODE23, and is also faster than the implicit stiff solver ODE15S with the sparse Jacobian matrix. We note that such a PDE in one spatial dimension yields a tightly banded Jacobian matrix which can be factored and solved very efficiently in ODE15S. That will no longer be the case in more than one spatial dimension.

Figs. 2 and 3 are plots of number of stages and step-size, respectively, chosen by the adaptive Runge–Kutta–Chebyshev projection (ARKCP) code. It is seen that within the error tolerance the code adjusts the time step-size and thus the number of stages in response to the stiffness of the problem. Note that the solution of the problem (26) is oscillatory with an approximate period of 12.

Table 1
Numerical results for 1D Brusselator problem

Method	RTOL	ATOL	Error	CPU time (s)
ARKCP	1.0E−2	1.0E−2	[1.1E−3 1.2E−3]	1.8
ODE15S	1.0E−3	1.0E−3	[2.2E−3 1.8E−3]	2.8
ODE45	1.0E−2	1.0E−2	[7.6E−4 6.3E−4]	104.6
ODE23	1.0E−2	1.0E−2	[1.4E−3 1.9E−3]	17.8

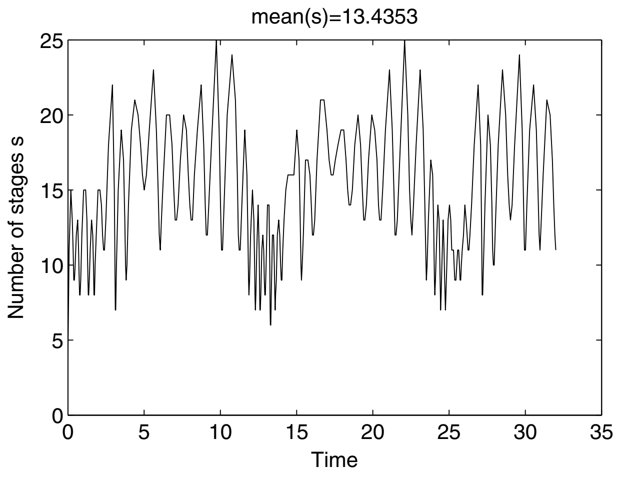

Fig. 2. Number of stages vs time for 1D Brusselator problem.

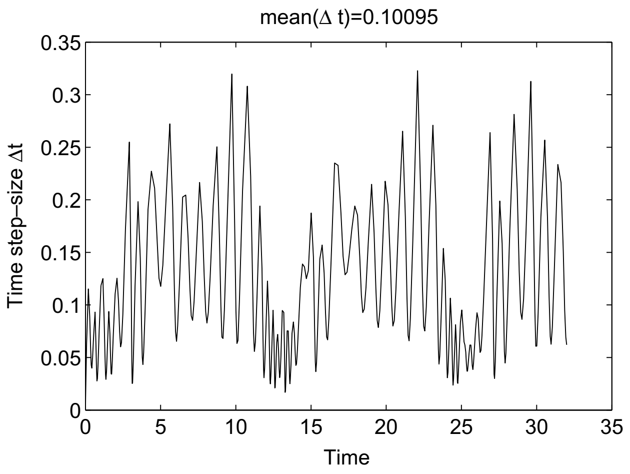

Fig. 3. Time step-size vs time for 1D Brusselator problem.

4.2. 2D Brusselator problem

We then applied our method to the 2D Brusselator problem (27) with a finite elment discretization,

$$\frac{\partial u}{\partial t} = D_1\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + \alpha - (\beta + 1)u + u^2 v, \tag{27a}$$

$$\frac{\partial v}{\partial t} = D_2\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) + \beta u - u^2 v, \tag{27b}$$

with boundary conditions

$$\frac{\partial u}{\partial \mathbf{n}} = \frac{\partial v}{\partial \mathbf{n}} = 0,$$

at Γ_1 and Γ_2, and initial conditions

$$u(t = 0) = 0.5 + y,$$
$$v(t = 0) = 1 + 5x.$$

Decomposed into two subdomains as shown in Fig. 4 and implementing the continuity constraint by Lagrange multipliers, the problem (27) is formulated into a DAE in the form of Eq. (7). The parameters were set as $D_1 = D_2 = 0.02$, and $\alpha = 1$, $\beta = 3.4$ in both subdomains. The time span for the simulation is $T = [0, 8]$.

We used COMSOL Multiphysics to implement the finite element model. After building the model in COMSOL Multiphysics, we used the commands `femlin` and `assemble` to extract the mass matrix M, the stiffness matrix A and the constraint matrix B. We then applied the RKCP method to obtain the solution of the resulting DAE. In the case of the Matlab ODE solvers, we first found the underlying ODE (this is basically the ODE which results from using the constraints of the DAE to eliminate the corresponding degrees of freedom [47]) of the DAE and solved that. We note that this may not be the most efficient way to solve the DAE, but it is the best we can do if we want to use the Matlab ODE solvers. Linear Lagrange elements were used in the COMSOL model.

The decomposed grids are shown in Fig. 4, in which the problem domain is discretized to 2014 elements and 1048 nodes, decomposed into two subdomains.

The numerical results with a consistent mass matrix and with a lumped mass matrix are shown in Tables 2 and 3, respectively. The error in the tables was again computed as $\frac{\|\mathbf{u}-\mathbf{u}_{\text{ref}}\|_2}{\|\mathbf{u}_{\text{ref}}\|_2}$, where $\mathbf{u}$ is the numerical solution and $\mathbf{u}_{\text{ref}}$ is the reference solution which is the solution computed by COMSOL Multiphysics with a tight error tolerance. It is observed that with the same error tolerance, all the methods yield about the same level of accuracy, but the ARKCP code is the most efficient.

The CPU time comparison with the Matlab solvers may not be the best proof of the performance of the RKCP method, since the problems used for our numerical tests are simple. However the numerical results

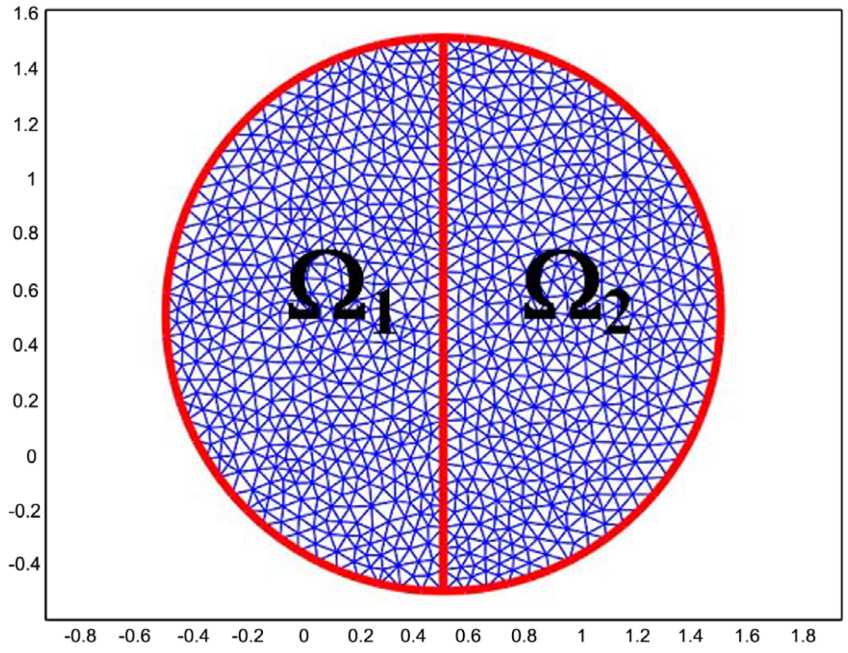

Fig. 4. Finite element grids for 2D Brusselator problem.

Table 2
Numerical results for 2D Brusselator problem with consistent mass matrix

Method	RTOL	ATOL	Error	CPU time (s)
ARKCP	1.0E−2	1.0E−2	2.9E−2	113
ODE15S	1.0E−2	1.0E−2	2.5E−2	276
ODE45	1.0E−2	1.0E−2	1.4E−2	1005
ODE23	1.0E−2	1.0E−2	1.4E−2	541

Table 3
Numerical results for 2D Brusselator problem with lumped mass matrix

Method	RTOL	ATOL	Error	CPU time (s)
ARKCP	1.0E−2	1.0E−2	4.6E−2	102
ODE15S	1.0E−2	1.0E−2	6.1E−2	291
ODE45	1.0E−2	1.0E−2	4.8E−2	329
ODE23	1.0E−2	1.0E−2	4.8E−2	198

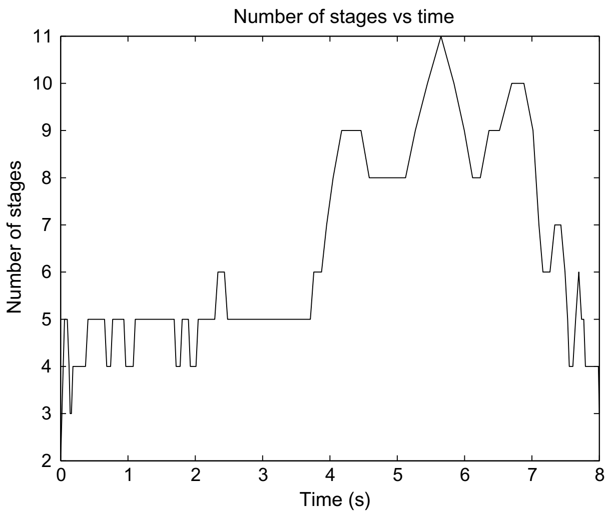

Fig. 5. Number of stages vs time with consistent mass matrix for 2D Brusselator problem.

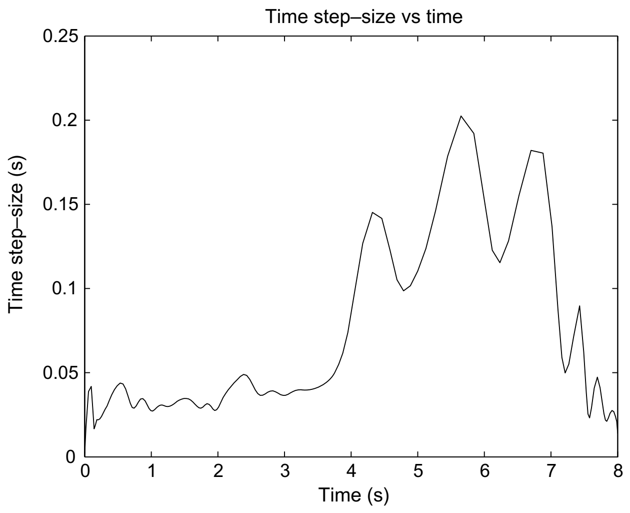

Fig. 6. Time step-size vs time with consistent mass matrix for 2D Brusselator problem.

clearly show that the efficiency of the RKCP method is at least comparable to that of the implicit method. We would expect the relative advantages of the RKCP to increase, for larger more complex problems.

We also plotted the number of stages and the time step-size as a function of time for both the consistent mass matrix method (without lumping) and the lumped mass matrix method [39] in Figs. 5–8.

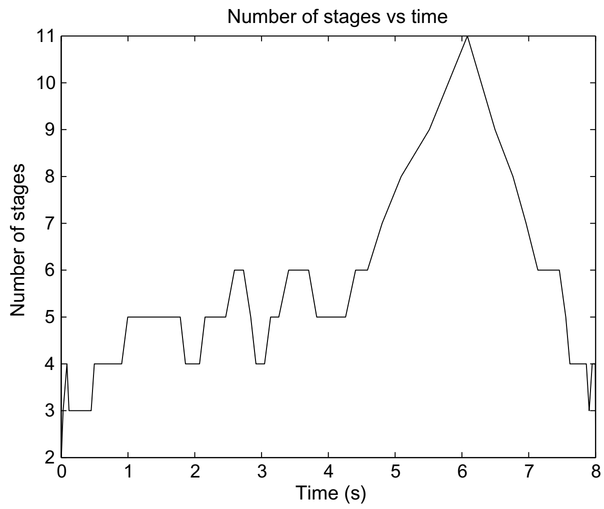

Fig. 7. Number of stages vs time with lumped mass matrix for 2D Brusselator problem.

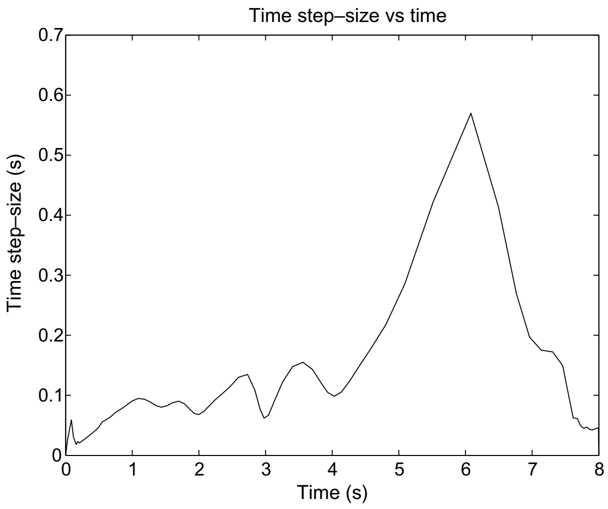

Fig. 8. Time step-size vs time with lumped mass matrix for 2D Brusselator problem.

Our numerical experiments verify that the lumped mass matrix method converges to the second-order accuracy, order $O(h^2)$ where h is the finite element size.

5. Conclusions

In this paper a fully explicit, stabilized domain decomposition method for parabolic PDEs is presented. Through semi-discretization in space, the parabolic problem is formulated into a differential-algebraic equation (DAE) system where the interface continuity constraints between subdomains are enforced by Lagrange multipliers. The Runge–Kutta–Chebyshev projection (RKCP) method is used to integrate the DAE explicitly and includes one projection per step to enforce the constraints. With mass lumping techniques and node-to-node matching grids, this method is fully explicit without solving any linear system. A stability analysis is presented to show the extended stability property of the method. The method is straightforward to implement and to parallelize, with high scalability and low communication. Numerical results demonstrate that the method has excellent performance.

References

[1] H.A. Schwarz, Gesammelte Mathematicsche Abhandlungen, Springer-Verlag, 1890, vol. 2, p. 133.

[2] M. Dryja, An additive Schwarz algorithm for two- and three-dimensional finite element elliptic problems, in: T.F. Chan, R. Glowinski, J. Périaux, O. Widlund (Eds.), Proceedings of the Second International Symposium on Domain Decomposition Methods for Partial Differential Equations, SIAM, Philadelphia, PA, 1989, p. 168.

[3] P.E. Bjørstad, M. Skogen, Domain decomposition algorithms of Schwarz type, designed for massively parallel computers, in: D.E. Keyes, T.F. Chan, G.A. Meurant, J.S. Scroggs, R.G. Voigt (Eds.), Proceedings of the Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations, SIAM, Philadelphia, PA, 1992, p. 362.

[4] X.C. Cai, Additive Schwarz algorithms for parabolic convection–diffusion equations, Numer. Math. 60 (1) (1991) 41.

[5] X.C. Cai, O. Widlund, Multiplicative Schwarz algorithms for nonsymmetric and indefinite problems, SIAM J. Numer. Anal. 30 (1993) 936.

[6] X.C. Cai, Multiplicative Schwarz algorithms for parabolic problems, SIAM J. Sci. Comput. 15 (1994) 587.

[7] X. Zhang, Multilevel Schwarz methods, Numer. Math. 63 (4) (1992) 521.

[8] T.P. Mathew, Schwarz alternating and iterative refinement methods for mixed formulations of elliptic problems, part I: Algorithms and numerical results, Numer. Math. 65 (4) (1993) 445.

[9] P.E. Bjørstad, O.B. Widlund, Iterative methods for the solution of elliptic problems on regions partitioned into substructures, SIAM J. Numer. Anal. 10 (5) (1986) 1053.

[10] J.H. Bramble, J.E. Pasciak, A.H. Schatz, An iterative method for elliptic problems on regions partitioned into substructures, Math. Comp. 46 (173) (1986) 361.

[11] D. Funaro, A. Quarteroni, P. Zanolli, An iterative procedure with interface relaxation for domain decomposition methods, SIAM J. Numer. Anal. 10 (5) (1988) 1053.

[12] A. Toselli, O. Widlund, Domain Decomposition Methods – Algorithms and Theory, Springer, 2005.

[13] J.F. Bourgat, R. Glowinski, P.L. Tallec, M. Vidrascu, Variational formulation and algorithm for trace operator in domain decomposition calculations, in: T.F. Chan, R. Glowinski, J. Périaux, O. Widlund (Eds.), Proceedings of the Second International Symposium on Domain Decomposition Methods for Partial Differential Equations, SIAM, Philadelphia, PA, 1989.

[14] Y.D. Roeck, P.L. Tallec, Analysis and test of a local domain decomposition preconditioner, in: R. Glowinski, Y.A. Kuznetsov, G.A. Meurant, J. Périaux, O. Widlund (Eds.), Proceedings of the Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations, SIAM, Philadelphia, PA, 1991.

[15] P.L. Tallec, Y.D. Roeck, M. Vidrascu, Domain decomposition methods for large linearly elliptic three-dimensional problems, J. Comput. Appl. Math. 34 (1) (1991) 93.

[16] J. Xu, J. Zou, Some nonoverlapping domain decomposition methods, SIAM Rev. 40 (4) (1998) 857.

[17] C. Farhat, F.X. Roux, A method of finite element tearing and interconnecting and its parallel solution algorithm, Int. J. Numer. Meth. Eng. 32 (1991) 1205.

[18] J. Mandel, R. Tezaur, C. Farhat, An optimal Lagrange multiplier based domain decomposition method for plate bending problems, Technical Report UCD-CCM-061, Center for Computational Mathematics, University of Colorado at Denver, 1995.

[19] M. Lesoinne, K. Pierson, An efficient FETI implementation on distributed shared memory machines with independent numbers of subdomains and processors, Contemp. Math. 218 (1998) 318.

[20] A. Klawonn, O.B. Widlund, A domain decomposition method with Lagrange multipliers for linear elasticity, in: C.H. Lai, P.E. Bjørstad, M. Cross, O.B. Widlund (Eds.), Proceedings of the Eleventh International Conference on Domain Decomposition Methods, Greenwich, UK, 1998.

[21] R. Tezaur, Analysis of Lagrange Multiplier Based Domain Decomposition. PhD thesis, University of Colorado at Denver, 1998.

[22] G. Lube, L. Müller, F.-C. Otto, A non-overlapping DDM of Robin-Robin type for parabolic problems, in: C.H. Lai, P.E. Bjørstad, M. Cross, O.B. Widlund (Eds.), Proceedings of the Eleventh International Conference on Domain Decomposition Methods, Greenwich, UK, 1998.

[23] M. Israeli, L. Vozovoi, A. Averbuch, Parallelizing implicit algorithms for time dependent problems by parabolic domain decomposition, J. Sci. Comput. 8 (1993) 151.

[24] M. Dryja, X. Tu, A domain decomposition discretization of parabolic problems, Technical Report 860, Courant Institute of Mathematical Sciences, New York University, 2005.

[25] A.A. Samarskii, P.N. Vabishchevich, Domain decomposition methods for parabolic problems, in: C.H. Lai, P.E. Bjørstad, M. Cross, O.B. Widlund (Eds.), Proceedings of the Eleventh International Conference on Domain Decomposition Methods, Greenwich, UK, 1998.

[26] Y. Zhuang, X. Sun, Stabilized explicit–implicit domain decomposition methods for the numerical solution of parabolic equations, SIAM J. Sci. Comput. 24 (2002) 335.

[27] H. Shi, H. Liao, Unconditional stability of corrected explicit–implicit domain decomposition algorithms for parallel approximation of heat equations, SIAM J. Numer. Anal. 44 (4) (2006) 1584.

[28] C.N. Dawson, Q. Du, T.F. Dupont, A finite difference domain decomposition algorithm for numerical solution of the heat equation, Math. Comput. 57 (195) (1991) 63.

[29] C.N. Dawson, T.F. Dupont, Explicit/implicit, conservative domain decomposition procedures for parabolic problems based on block-centered finite differences, SIAM J. Numer. Anal. 31 (1994) 1045.

[30] J. Zhu, H. Qian, On an efficient parallel algorithm for solving time dependent partial differential equations, in: Proceedings of the PDPTA'98 International Conference, 1998, p. 394.

[31] Y. Jun, T.-Z. Mai, IPIC Domain decomposition algorithm for parabolic problems, Appl. Math. Comput. 177 (1) (2006) 352.
[32] P.J. van der Houwen, B.P. Sommeijer, On the internal stability of explicit, *m*-stage Runge–Kutta methods for large *m*-values, Z. Angew. Math. Mech. 60 (1980) 479.
[33] J.G. Verwer, Explicit Runge–Kutta methods for parabolic differential equations, Appl. Numer. Math. 22 (1996) 359.
[34] B.P. Sommeijer, L.F. Shampine, J.G. Verwer, RKC: an explicit solver for parabolic PDEs, J. Comp. Appl. Math. 88 (1997) 315.
[35] Z. Zheng, L.R. Petzold, Runge–Kutta–Chebyshev projection method, J. Comp. Phys. 219 (2006) 976.
[36] J.G. Verwer, B.P. Sommeijer, W. Hundsdorfer, RKC time-stepping for advection–diffusion–reaction problems, J. Comp. Phys. 201 (2004) 61.
[37] J.G. Verwer, B.P. Sommeijer, An implicit–explicit Runge–Kutta–Chebyshev scheme for diffusion–reaction equations, SIAM J. Sci. Comput. 25 (2004) 1824.
[38] R. Bermejo, M. El Amrani, A finite element semi-Lagrangian explicit Runge–Kutta–Chebyshev method for convection dominated reaction–diffusion problems, J. Comput. Appl. Math. 154 (2003) 27.
[39] S.R. Wu, Lumped mass matrix in explicit finite element method for transient dynamics of elasticity, Comput. Meth. Appl. Mech. Eng. 195 (2006) 5983.
[40] F. Ben Belgacem, The mortar finite element method with Lagrange multipliers, Numer. Math. 84 (1999) 73.
[41] A.J. Chorin, Numerical solution of the Navier–Stokes equations, Math. Comp. 22 (104) (1968) 745.
[42] R. Témam, Sur l'approximation de la solution des équations de Navier–Stokes par la méthode des pas fractionnaires (II), Arch. Rat. Mech. Anal. 33 (1969) 377.
[43] O. Botella, A high-order mass-lumping procedure for B-spline collocation method with application to incompressible flow simulations, Int. J. Numer. Meth. Fluids 41 (2003) 1295.
[44] U.M. Ascher, L.R. Petzold, Computer methods for ordinary differential equations and differential-algebraic equations, Soc. Ind. Appl. Math. (1998).
[45] Z. Zheng, L.R. Petzold, A framework for the analysis of second order projection methods, submitted for publication.
[46] T.J.R. Hughes, The Finite Element Method: Linear Static and Dynamic Finite Element Analysis, Dover Publications, 2000.
[47] U.M. Ascher, L.R. Petzold, Stability of computational methods for constrained dynamics systems, SIAM J. Sci. Statist. Comput. 14 (1993) 95.