SIAM J. Sci. Comput. © 2021 Society for Industrial and Applied Mathematics
Vol. 43, No. 5, pp. S637–S654

A LOW-RANK MATRIX EQUATION METHOD FOR SOLVING PDE-CONSTRAINED OPTIMIZATION PROBLEMS[*]

ALEXANDRA BÜNGER[†], VALERIA SIMONCINI[‡], and MARTIN STOLL[†]

Abstract. PDE-constrained optimization problems arise in a broad number of applications such as hyperthermia cancer treatment and blood flow simulation. Discretization of the optimization problem and using a Lagrangian approach result in a large-scale saddle-point system, which is challenging to solve, and acquiring a full space-time solution is often infeasible. We present a new framework to efficiently compute a low-rank approximation to the solution by reformulating the KKT system into a Sylvester-like matrix equation. This matrix equation is subsequently projected onto a small subspace via an iterative rational Krylov method, and we obtain a reduced problem by imposing a Galerkin condition on its residual. In our work we discuss implementation details and dependence on the various problem parameters. Numerical experiments illustrate the performance of the new strategy also when compared to other low-rank approaches.

Key words. PDE-constrained optimization, matrix equation, rational Krylov subspace

AMS subject classifications. 65F10, 49M41, 65F45, 65F55

DOI. 10.1137/20M1341210

1. Introduction.

A vast number of today's technological advancements are due to the ever advancing improvement in modeling and solving of partial differential equation (PDE) problems. One class of especially challenging problems is PDE-constrained optimization, which has a host of applications, reaching from engineering design and control problems to medical applications like cancer treatment [21, 4, 9].

We consider a typical PDE-constrained optimization problem on a space-time cylinder $[0,T]\times\Omega$ given by

$$\min_{y,u} J(y,u) \tag{1.1}$$

$$\text{subject to } \dot{y} = \mathcal{L}(y) + u, \tag{1.2}$$

where we are interested in the minimization of the functional $J(y,u)$ constrained by a differential operator $\mathcal{L}(y)$ with space and time dependent state y and control u. An extensive analysis of this type of problem can be found, e.g., in [27] or [11]. In this work, we will follow the popular approach of first discretizing the problem and then formulating the discrete first-order optimality conditions as described in [12]. Finding an efficient numerical solution to the large system of equations resulting from these conditions is of high interest, and many approaches exist to solve the resulting saddle-point system such as using a block preconditioner to subsequently solve the preconditioned system with iterative solvers like Minres (cf. [20, 17, 19, 22]).

[*]Received by the editors May 28, 2020; accepted for publication (in revised form) June 4, 2021; published electronically August 19, 2021.
https://doi.org/10.1137/20M1341210

Funding: The work of the first and third authors was supported by the German Science Foundation (DFG) through grant 1742243256 - TRR 9. The second author is a member of Indam-GNCS. Its support is gratefully acknowledged. This work was also supported by the DAAD-MIUR Mobility Program 2018 "Optimization and low-rank solvers for isogeometric analysis (34876)" between TU-Chemnitz and the Università di Bologna.

[†]Fakultät für Mathematik, Technische Universität Chemnitz, Chemnitz, 09126, Germany (alexandra.buenger@mathematik.tu-chemnitz.de, martin.stoll@mathematik.tu-chemnitz.de).

[‡]Alma Mater Studiorum - Università di Bologna, Bologna, 40126, Italy, and IMATI-CNR, Pavia, 27100, Italy (valeria.simoncini@unibo.it).

However, today's PDE models often result in millions of variables, and solving optimization problems governed by such huge models is still a challenging task due to their size and complexity. They often even prove to be impossible to solve on a normal computer due to the memory required to store the usually large and dense solution computed by standard algorithms. Therefore, memory-efficient solvers, which compute reduced approximations to the solution, are a desirable approach to solving these challenging problems. We here propose a widely applicable low-rank solver, which computes a projection of the solution onto a small subspace.

The method relies on the reformulation of the problem into a matrix equation, which will be illustrated in section 2. The numerical solution of this PDE-constrained optimization problem poses a significant challenge regarding the complexity of both storage and computational resources. We show that it is possible to separate spatial and temporal data via a novel low-rank matrix equation solver in section 3. We introduce a method to reduce the system of equations into a single low-rank matrix equation. In section 4 we discuss the choice of the approximation space, while in section 5 we analyze stopping criteria and a fast update scheme of the residual computation. And finally, in section 6, we examine the performance and applicability of our method on various numerical examples. We show the method's robustness with respect to the various problem parameters as well as assess the performance of our method compared with a previously proposed low-rank approach on a distributed heat equation model as well as a boundary control problem. Further, we show the applicability of our method to more challenging problems such as a partial state observation and a nonsymmetric PDE constraint.

2. Problem formulation.

We consider a PDE-constrained optimal control problem on a space-time cylinder with time interval $[0, T]$ and a spatial domain Ω as in (1.1)–(1.2). The functional we want to minimize reads

$$J(y,u) = \frac{1}{2}\int_0^T \int_{\Omega_1} (y-\hat{y})^2 \,\mathrm{d}x\,\mathrm{d}t + \frac{\beta}{2}\int_0^T \int_{\Omega_u} u^2 \,\mathrm{d}x\,\mathrm{d}t. \tag{2.1}$$

Here y is the state, $\hat{y}$ is the desired state given on a subset Ω_1 of Ω, and u is the control on a subset Ω_u of Ω, which is regularized by the control cost parameter β.

For the PDE subject to which we want to minimize the functional $J(y, u)$, let us exemplarily consider the heat equation with $\mathcal{L}(y) = \Delta y$ and Dirichlet boundary condition,

$$\dot{y} - \Delta y = u \quad \text{in} \quad \Omega_u, \tag{2.2}$$

$$\dot{y} - \Delta y = 0 \quad \text{in} \quad \Omega/\Omega_u, \tag{2.3}$$

$$y = 0 \quad \text{on} \quad \partial\Omega. \tag{2.4}$$

There are two distinct options in proceeding with this problem. Either we formulate the optimality conditions and discretize them subsequently, or we first discretize the problem and then formulate the discrete first-order optimality conditions, known as the Karush–Kuhn–Tucker (KKT) conditions [12]. In this work we follow the second approach.

We discretize space and time in an all-at-once approach. The spatial discretization is done with finite elements, and to discretize in time, we split the time interval into n_T intervals of length $\tau = \frac{T}{n_T}$. Using a rectangle rule, the discretization of (2.1) becomes

$$\sum_{t=1}^{n_T} \frac{\tau}{2}(y_t - \hat{y}_t)^T M_1 (y_t - \hat{y}_t) + \frac{\tau\beta}{2} u_t^T M_u u_t, \tag{2.5}$$

where $y_t, \hat{y}_t$, and u_t are spatial discretizations of $y, \hat{y}$ of size n and u of size n_u for each time step $t = 1, \ldots, n_T$. Using an implicit Euler scheme [14] the discrete formulation of the PDE (2.2) reads

$$\frac{M(y_t - y_{t-1})}{\tau} + Ky_t = Nu_t \quad \text{for } t = 1, \ldots, n_T, \tag{2.6}$$

$$y_0 = 0. \tag{2.7}$$

Here, $M, M_1 \in \mathbb{R}^{n\times n}$, $M_u \in \mathbb{R}^{n_u \times n_u}$, and $N \in \mathbb{R}^{n\times n_u}$ are the mass matrices of the spatial discretization, and $K \in \mathbb{R}^{n\times n}$ denotes the so-called stiffness matrix resulting from the discretization of the differential operator $\mathcal{L}(y)$. The boundary constraints are incorporated into the stiffness matrix. For further details about the discretization of PDE operators we refer the reader to [23].

In the course of this paper we cover three distinct setups for this problem:

(i) All domains are the same, $\Omega = \Omega_1 = \Omega_u$, and thus we have a full domain observation and control, leading to $n = n_u$ and $M = M_1 = M_u = N$.

(ii) We have a full observation $\Omega_1 = \Omega$ but only a partial control, e.g., a boundary control $\Omega_u = \partial\Omega$ leading to $n_u < n$ and $M = M_1 \neq M_u$, and N being rectangular.

(iii) We have a full domain control $\Omega_u = \Omega$ but only a partial observation on a smaller domain Ω_1, leading to $M_1 \neq M$ with M_1 being singular due to entries set to zero on the nonobserved parts of the state.

Additionally, we will address the case of $K \neq K^T$ later on.

We collect the discretizations of the variables in matrices $Y = [y_1, \ldots, y_{n_T}] \in \mathbb{R}^{n\times n_T}$ and denote their vectorizations by $\underline{Y} = \text{vec}(Y)$, respectively for $\hat{y}$ and u. With this we write the optimization problem in compact form as

$$\min_{Y,U} \frac{\tau}{2}(\underline{Y} - \underline{\hat{Y}})^T \mathcal{M}_1 (\underline{Y} - \underline{\hat{Y}}) + \frac{\tau\beta}{2} \underline{U}^T \mathcal{M}_u \underline{U}, \tag{2.8}$$

$$\text{s.t. } \mathcal{K}\underline{Y} - \tau\mathcal{N}\underline{U} = 0, \tag{2.9}$$

with the matrices

$$\mathcal{M} = \begin{bmatrix} M & & & \\ & M & & \\ & & \ddots & \\ & & & M \end{bmatrix}, \qquad \mathcal{K} = \begin{bmatrix} M + \tau K & & & \\ -M & \ddots & & \\ & \ddots & \ddots & \\ & & -M & M + \tau K \end{bmatrix}, \tag{2.10}$$

where mass matrices $M \in \mathbb{R}^{n\times n}$ and stiffness matrix $K \in \mathbb{R}^{n\times n}$ repeat n_T times each. The matrices $\mathcal{M}_u$ and $\mathcal{N}$ are block diagonal matrices like $\mathcal{M}$ but with the different mass matrices M_u and N, respectively, on the diagonal.

To solve the discrete problem (2.8)–(2.9), we have to solve the system of equations resulting from the first-order optimality conditions [3]. They state that an optimal solution must be a saddle point of the Lagrangian of the problem

$$\nabla\mathcal{L}(\underline{Y}^*, \underline{U}^*, \underline{\Lambda}^*) = 0. \tag{2.11}$$

The Lagrangian of this problem reads

$$\mathcal{L}(\underline{Y},\underline{U},\underline{\Lambda}) = \frac{\tau}{2}(\underline{Y}-\underline{\hat{Y}})^T\mathcal{M}_1(\underline{Y}-\underline{\hat{Y}}) + \frac{\tau\beta}{2}\underline{U}^T\mathcal{M}_u\underline{U} + \underline{\Lambda}^T(\mathcal{K}\underline{Y}-\tau\mathcal{N}\underline{U}), \tag{2.12}$$

with Lagrange multipliers Λ. Thus, the optimal solution solves the following set of linear equations:

$$0 = \nabla_Y\mathcal{L}(\underline{Y},\underline{U},\underline{\Lambda}) = \tau\mathcal{M}_1(\underline{Y}-\underline{\hat{Y}}) + \mathcal{K}^T\underline{\Lambda}, \tag{2.13}$$

$$0 = \nabla_U\mathcal{L}(\underline{Y},\underline{U},\underline{\Lambda}) = \tau\beta\mathcal{M}_u\underline{U} - \tau\mathcal{N}^T\underline{\Lambda}, \tag{2.14}$$

$$0 = \nabla_\Lambda\mathcal{L}(\underline{Y},\underline{U},\underline{\Lambda}) = \mathcal{K}\underline{Y} - \tau\mathcal{N}\underline{U}. \tag{2.15}$$

Memorywise, an effective approach to solving this large-scale problem is to use a low-rank approach, which finds a cheap representation of the solution matrix within a low-rank subspace range(V) and a reduced solution Z, whose number of columns is related to the number of employed time steps,

$$Y \approx VZ, \tag{2.16}$$

and similarly for the other matrix variables U and Λ.

In this work we show that we can compute such a low-rank solution based on a matrix equation rather than using the saddle-point formulation used in [26]. This is achieved by rearranging the large system of equations (2.13)–(2.15) into a generalized Sylvester matrix equation of the form

$$A_1X + XC + A_2XI_0 - A_3XD - F_1F_2 = 0. \tag{2.17}$$

The resulting matrix equation can be efficiently solved by using a tailored low-rank Krylov subspace method. This new approach greatly reduces storage and time requirements while being robust to parameter changes. This matrix equation oriented methodology allows one to clearly identify space and time dimensions and aims at reducing the problem dimension in the space variables; the time variable is handled using an all-at-once procedure. Similar strategies have been used, for instance, in [5, 16].

One assumption we make is that we have a low-rank approximation or representation of the desired state as

$$\hat{Y} \approx Y_1Y_2, \tag{2.18}$$

with $Y_1 \in \mathbb{R}^{n\times r}$, $Y_2 \in \mathbb{R}^{r\times n_T}$, and $r < n_T$. First, we introduce the auxiliary matrices

$$\mathcal{I} = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix} \text{ and } \mathcal{C} = \begin{bmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{bmatrix} \tag{2.19}$$

to rewrite the system matrices as Kronecker products $\mathcal{M} = \mathcal{I}\otimes M$, $\mathcal{M}_1 = \mathcal{I}\otimes M_1$, $\mathcal{N} = \mathcal{I}\otimes N$, and $\mathcal{K} = \mathcal{I}\otimes\tau K + \mathcal{C}\otimes M$. Note that here $\mathcal{C}$ results from the implicit Euler scheme. This scheme can be replaced by a different time stepping method like a Crank–Nicolson scheme [13], in which case $\mathcal{C}$ and the identities $\mathcal{I}$ in $\mathcal{N}$ and $\mathcal{K}$ will become different block matrices. This would maintain the structure of the following steps but would naturally make solving the resulting matrix equation more challenging.

With this our system of KKT conditions (2.13)–(2.15) becomes

$$\tau(\mathcal{I} \otimes M_1)\underline{Y} + (\mathcal{I} \otimes \tau K^T + \mathcal{C}^T \otimes M^T)\underline{\Lambda} - \tau(\mathcal{I} \otimes M_1)\underline{\hat{Y}} = 0, \tag{2.20}$$

$$\tau\beta(\mathcal{I} \otimes M_u)\underline{U} - \tau(\mathcal{I} \otimes N^T)\underline{\Lambda} = 0, \tag{2.21}$$

$$(\mathcal{I} \otimes \tau K + \mathcal{C} \otimes M)\underline{Y} - \tau(\mathcal{I} \otimes N)\underline{U} = 0. \tag{2.22}$$

We exploit the relation

$$(W^T \otimes V)\mathrm{vec}(Y) = \mathrm{vec}(VYW) \tag{2.23}$$

to rewrite the equations (2.20)–(2.22) as

$$\tau M_1 Y + \tau K^T \Lambda + M^T \Lambda C - \tau M_1 \hat{Y} = 0, \tag{2.24}$$

$$\tau\beta M_u U - \tau N^T \Lambda = 0, \tag{2.25}$$

$$\tau K Y + M Y C^T - \tau N U = 0. \tag{2.26}$$

The mass matrix M arising from a standard Galerkin method is symmetric and can be considered lumped, i.e., diagonal. Therefore, we can eliminate (2.25) by setting $U = \frac{1}{\beta} M_u^{-1} N^T \Lambda$ in the remaining two equations,

$$\tau M_1 Y + \tau K^T \Lambda + M^T \Lambda C - \tau M_1 \hat{Y} = 0, \tag{2.27}$$

$$\tau K Y + M Y C^T - \frac{\tau}{\beta} N M_u^{-1} N^T \Lambda = 0, \tag{2.28}$$

which are equivalent to

$$M^{-1} M_1 Y + M^{-1} K^T \Lambda + \Lambda \tilde{C} - M^{-1} M_1 \hat{Y} = 0, \tag{2.29}$$

$$M^{-1} K Y + Y \tilde{C}^T - \frac{1}{\beta} M^{-1} N M_u^{-1} N^T \Lambda = 0, \tag{2.30}$$

where $\tilde{C} = \frac{1}{\tau} C$. For now, let us assume that $K = K^T$, but our approach can be easily generalized for nonsymmetric K as we will demonstrate later on. Now this representation corresponds to

$$\begin{aligned} &M^{-1}K \begin{bmatrix} Y & \Lambda \end{bmatrix} + \begin{bmatrix} Y & \Lambda \end{bmatrix} \underbrace{\begin{bmatrix} \tilde{C}^T & 0 \\ 0 & \tilde{C} \end{bmatrix}}_{C_1} + M^{-1} M_1 \begin{bmatrix} Y & \Lambda \end{bmatrix} \underbrace{\begin{bmatrix} 0 & \mathcal{I} \\ 0 & 0 \end{bmatrix}}_{I_0} \\ &+ \frac{1}{\beta} M^{-1} N M_u^{-1} N^T \begin{bmatrix} Y & \Lambda \end{bmatrix} \underbrace{\begin{bmatrix} 0 & 0 \\ -\mathcal{I} & 0 \end{bmatrix}}_{D} - \underbrace{\begin{bmatrix} 0 & M^{-1} M_1 \hat{Y} \end{bmatrix}}_{F_1 F_2} = \begin{bmatrix} 0 & 0 \end{bmatrix}. \end{aligned} \tag{2.31}$$

We denote $X = \begin{bmatrix} Y & \Lambda \end{bmatrix}$, $A_1 = M^{-1}K$, $A_2 = M^{-1}M_1$, $A_3 = \frac{1}{\beta} M^{-1} N M_u^{-1} N^T$, $F_1 = M^{-1} M_1 Y_1$, and $F_2 = \begin{bmatrix} 0_{r \times n_T} & Y_2 \end{bmatrix}$, where $\hat{Y} = Y_1 Y_2$ with Y_1, Y_2 of low column and row rank, respectively. Then we get the desired format from (2.17) as

$$A_1 X + X C_1 + A_2 X I_0 + A_3 X D - F_1 F_2 = 0, \tag{2.32}$$

where the left-hand coefficient matrices A_1, A_2, and A_3 have size $n \times n$ while the right-hand ones C_1, I_0, and D have size $2n_T \times 2n_T$, so that $X \in \mathbb{R}^{n \times 2n_T}$.

3. Low-rank solution. The generalized Sylvester equation in (2.32) replaces the large Kronecker product based system of equations (2.20)–(2.22). Since the solution matrix $X \in \mathbb{R}^{n\times 2n_T}$ will be dense and potentially very large, to exploit the new setting it is desirable to find an appropriate approximation space and a low-rank reduced matrix approximation $Z \in \mathbb{R}^{p\times 2n_T}$ such that

$$X \approx V_p Z, \tag{3.1}$$

where the orthonormal columns of $V_p \in \mathbb{R}^{n\times p}$ generate the approximation space. With this setting we can construct a reduced version of the matrix equation (2.32). Let us assume we compute an approximation as in (3.1). Then the residual matrix associated with (2.32) reads

$$R = A_1V_pZ + V_pZC_1 + A_2V_pZI_0 + A_3V_pZD - F_1F_2. \tag{3.2}$$

We impose the Galerkin orthogonality of the residual matrix with respect to the approximation space, which in the matrix inner product is equivalent to writing $V_p^TR = 0$, so that our equation becomes

$$V_p^TA_1V_pZ + V_p^TV_pZC_1 + V_p^TA_2V_pZI_0 + V_p^TA_3V_pZD - V_p^TF_1F_2 = 0. \tag{3.3}$$

Let us denote the reduced $p \times p$ coefficient matrices as $A_{1,r} := V_p^TA_1V_p$, $A_{2,r} := V_p^TA_2V_p$, $A_{3,r} := V_p^TA_3V_p$ and set $F_{1,r} = V_p^TF_1 \in \mathbb{R}^{p\times 1}$. The resulting reduced equation

$$A_{1,r}Z + \mathcal{I}_pZC_1 + A_{2,r}ZI_0 + A_{3,r}ZD - F_{1,r}F_2 = 0 \tag{3.4}$$

has the same structure as the original matrix equation (2.32), but its size is reduced to $p \times 2n_T$. By exploiting once again the relation in (2.23), we get the small linear system of equations

$$\big((\mathcal{I}_{2n_T} \otimes A_{1,r}) + (C_1^T \otimes \mathcal{I}_p) + (I_0^T \otimes A_{2,r}) + (D^T \otimes A_{3,r})\big)\underline{Z} = \mathrm{vec}(F_{1,r}F_2), \tag{3.5}$$

with $\underline{Z} = \mathrm{vec}(Z)$. For a small subspace size $p \ll n$ this system of equations is significantly easier to solve, and we can use either a direct or an iterative method to do so. If the obtained approximate solution V_pZ is not sufficiently good, then the approximation space can be expanded and a new approximation constructed, giving rise to an iterative method. The use of low-rank methods within optimization of large-scale systems has been successfully documented in several articles, and we refer the reader to [26, 7, 6, 2] for recent accounts.

4. Subspace computation. To construct the projection (3.1) we need an iterative subspace method, which constructs a relevant subspace for our problem. For this we make use of rational Krylov subspaces

$$\mathcal{K}_p(A, v, \mathbf{s}) = \mathrm{span}\left\{v, (A + s_1I)^{-1}v, \ldots, \prod_{j=1}^{p-1}(A + s_jI)^{-1}v\right\} \tag{4.1}$$

with shifts $\mathbf{s} = [s_1, \ldots, s_j]$ as described in [25]. Rational Krylov subspaces have become an important tool in many dimension reduction problems such as eigenvalue problems, dynamical systems, approximation of matrix functions, and multiterm matrix equations. The main advantage is that they are able to capture crucial spectral information on the matrix A in (4.1) already with small values of p, compared with standard (polynomial) Krylov subspaces. The reduced problem obtained by projection onto a rational Krylov subspace is thus able to preserve the leading spectral and functional properties of the original problem, but in a dramatically smaller dimensional space. As an initial vector (or block of vectors, in the case of $\mathrm{rank}(F_1) > 1$) we take the right-hand side, $v_0 = F_1$, and to construct the subspace (4.1) we employ a tailored strategy to adapt to the different settings.

The idea is to generate an approximation space that contains spectral information of all matrices A_i, $i = 1, \ldots, 3$, so as to be effective already for a small dimension. Without further information, this goal would be achieved by generating a subspace that is the sum of $\mathcal{K}_p(A_i, v_0, \mathbf{s})$, of dimension at most three times p. However, by accurately selecting the matrices involved in the generation of the Krylov subspaces, smaller dimensional spaces can be determined. We recall the three previously described setups and make distinct subspace choices for each of them:

(i) *$M_1 = M = N = M_u$, all full rank.* In this case $A_2 = I$ and A_3 is a diagonal nonsingular matrix. We construct a rational Krylov subspace from $A = A_1$.

(ii) *$M_1 \neq M$, $M = M_u = N$, and M_1 of lower rank.* We construct a mixed subspace where we add the following two new vectors in step k:

$$\{(A_1 + s_k^{(1)} I)^{-1} v_k,\ (A_2 + s_k^{(2)} I)^{-1} v_k\}, \tag{4.2}$$

so that the space dimension grows by at most two per iteration, instead of one. This adds the influence of M_1 to the constructed subspace.

(iii) *$M_1 = M$, N tall, $n_u < n$.* Here, $A_2 = I$ while $A_3 = M^{-1} N M_u^{-1} N^T$ is not invertible. We thus define $A_3(\alpha) = A_3 + \alpha A_1$; this selection is justified below. In this case we use the following two new vectors in step k:

$$\{(A_1 + s_k^{(1)} A_3(\alpha))^{-1} v_k,\ (A_3(\alpha) + s_k^{(2)} A_1)^{-1} v_k\}, \tag{4.3}$$

and once again the space dimension grows by at most two per iteration, instead of one.

(iv) *$M = M_1 = M_u = N$, all full rank, $K \neq K^T$.* For a number of PDE constraints, like convection-diffusion problems, the stiffness matrix is nonsymmetric. In this case, we have to slightly modify the first component $A_1 X$ in (2.32) to $A_1 X [\begin{smallmatrix} \mathcal{I} & 0 \\ 0 & 0 \end{smallmatrix}] + M^{-1} K^T X [\begin{smallmatrix} 0 & 0 \\ 0 & \mathcal{I} \end{smallmatrix}]$. Here again we construct a mixed subspace where we add two new vectors in step k:

$$\{(A_1 + s_k^{(1)} I)^{-1} v_k,\ (M^{-1} K^T + s_k^{(2)} I)^{-1} v_k\}. \tag{4.4}$$

A strategy similar to (ii)–(iv) was successfully adopted in [18] for a multiterm linear matrix equation in a different context. The effectiveness of the mixed approaches in (ii)–(iii) relies on the fact that the generated space is rich in eigencomponents of both matrices. In (ii), where A_2 is diagonal and singular with a bounded and tight nonzero spectral interval, a good spectral homogeneity in the space construction can be reached by shifting the matrix A_2 by some α so that the spectra of A_1 and $A_2 + \alpha_2 I_n$ are contained in the same interval. Hence, we introduce a shift parameter α_2 in (2.32) to get

$$A_1 X + X(C_1 - \alpha_2 I_0) + (A_2 + \alpha_2 I_n) X I_0 - A_3 X D - F_1 F_2 = 0. \tag{4.5}$$

With these premises, an appropriate choice for α_2 is a rough approximation to the largest eigenvalue of A_1. Due to the good properties of the transformed spectrum, the shifts were computed by only using the projection of the matrix A_1 onto the generated space; more details on the shift selection can be found in [8]. In cases (i) and (iv), A_2 is not singular, but still the spectra of A_1 and A_2 differ greatly. Applying the same strategy to shift A_2 as in (4.5) also proved to be beneficial in these cases.

In (iii), the structure and singularity of A_3 was significantly more challenging, because $\|A_3\|$ inversely depends on β; hence the spectrum of a shifted version of A_3 may not be enclosed in that of A_1. We propose considering the equivalent problem

$$A_1X(I-\alpha_3D)+XC_1+A_2XI_0-(A_3+\alpha_3A_1)XD-F_1F_2=0, \tag{4.6}$$

where $\alpha_3=\frac{1}{\sqrt{\beta}}\|A_3\|_F/\|A_1\|_F$. With this choice the Frobenius norms of α_3A_1 and A_3 have similar magnitude, and the influence of β is reflected realistically in the equation. With this formulation, we found particularly successful the use of the projection of the pair $(A_1, A_3+\alpha_3A_1)$ onto the current approximation space to determine the next shifts during the iteration.

Further subspace reduction. By computing $X\approx V_pZ$ we want to reduce storage requirements and computation time. This goal is only achieved if the approximation space dimension remains considerably smaller than n_T. By enriching the subspace as outlined above, however, the space dimension may in principle grow up to n in case the solution is not sufficiently accurate. To keep the subspace as small as possible when it is not optimal we include a truncation scheme which bounds the subspace dimension to less than n_T. Thus, in the worst case the solution will have maximum rank. To reduce the dimension we compute a singular value decomposition of $Z=U\Sigma V^T$, with $\Sigma=\mathrm{diag}(\sigma_1,\ldots,\sigma_p)$, where $\sigma_1\geq\cdots\geq\sigma_p$ are the singular values of Z. In case some of the singular values are too small, say $\sigma_i<\varepsilon$ for some i, it is clear that some of the subspace information is not needed for computing a good approximation to the solution. If this occurs, we truncate the subspace with $\tilde{p}=i-1$ by setting

$$V_{\tilde{p}}=V_pU_{\tilde{p}}, \tag{4.7}$$

with $U_{\tilde{p}}$ denoting the first $\tilde{p}$ columns of U. Note that the next subspace additions (4.2) are still conducted using v_k from the latest generated vectors in the original subspace V_p, and orthogonalized with respect to the reduced subspace. We refer the reader to the discussion of Table 6.4 for an illustration of the achievable memory savings by adopting this strategy.

5. Residuals and stopping criteria.

To determine a computable stopping criterion we monitor the residuals of the two equations (2.27) and (2.28),

$$R_1=\tau M_1Y+\tau K^T\Lambda+M^T\Lambda C-\tau M_1\hat{Y}, \tag{5.1}$$

$$R_2=\tau KY+MYC^T-\frac{\tau}{\beta}NM^{-1}N^T\Lambda, \tag{5.2}$$

which are closely related to the original system.

Computing the residuals poses a bottleneck in this scheme as straightforward computation is time consuming and would require forming the full solution $[Y\Lambda]=V_pZ$. To avoid this and substantially speed up the computation time, we rely on a low-rank representation of the residual and calculate its norm, making use of an updated QR method. The matrix residuals can be rewritten as

$$R_1 = \tau M_1 V_p Z_Y + \tau K^T V_p Z_\Lambda + M V_p Z_\Lambda C - \tau M_1 \hat{Y} \tag{5.3}$$

$$= \begin{bmatrix} \tau M_1 V_p & \tau K^T V_p & M V_p & -\tau M_1 Y_1 \end{bmatrix} \cdot \begin{bmatrix} Z_Y \\ Z_\Lambda \\ Z_\Lambda C \\ Y_2 \end{bmatrix}, \tag{5.4}$$

$$R_2 = \tau K V_p Z_Y + M V_p Z_Y C^T - \frac{\tau}{\beta} N M^{-1} N^T V_p Z_\Lambda \tag{5.5}$$

$$= \begin{bmatrix} \tau K V_p & M V_p & -\frac{\tau}{\beta} N M^{-1} N^T V_p \end{bmatrix} \cdot \begin{bmatrix} Z_Y \\ Z_Y C^T \\ Z_\Lambda \end{bmatrix}, \tag{5.6}$$

where Z_Y denotes the Y-component of Z and Z_Λ denotes the component associated with Λ, respectively. Let either of the two quantities be denoted by

$$R = R_L R_R. \tag{5.7}$$

Here the matrices have dimensions $R_L \in \mathbb{R}^{n \times (3p+1)}$ and $R_R \in \mathbb{R}^{(3p+1) \times n_T}$. We consider a reduced QR decomposition of the tall and skinny matrix R_L,

$$R_L = Q_1 R_{1,1}, \quad \text{with} \quad Q_1 \in \mathbb{R}^{n \times (4p+1)}, \quad R_{1,1} \in \mathbb{R}^{(4p+1) \times (4p+1)}. \tag{5.8}$$

At each iteration new columns are added to the matrices R_L and new rows to R_R. To further reduce the computational cost, we update the QR decomposition so as to avoid a new factorization from scratch at each iteration. Assume that we have a current subspace of size p so that $V_p = [v_1, \ldots, v_p]$ and we add another vector v_{p+1} to the space. Thus, four new columns are added to R_L, which we add to the end of R_L, while keeping the same reordering for R_R. Adding a new column v_{p+1} gives

$$\begin{bmatrix} R_L & v_{p+1} \end{bmatrix} = \begin{bmatrix} Q_1 R_{1,1} & v_{p+1} \end{bmatrix} = \begin{bmatrix} Q_1 & q_2 \end{bmatrix} \begin{bmatrix} R_{1,1} & R_{1,2} \\ 0 & R_{2,2} \end{bmatrix}, \tag{5.9}$$

where only two column vectors q_2 and $R_{1,2}$ and a scalar value $R_{2,2}$ have to be computed. We have

$$Q_1 R_{1,2} + q_2 R_{2,2} = v_{p+1}. \tag{5.10}$$

Setting $R_{1,2} = Q_1^T v_{p+1}$ and constructing the vector

$$\hat{q}_2 = v - Q_1(Q_1^T v_{p+1}) = v_{p+1} - Q_1 R_{1,2}, \tag{5.11}$$

we can set $q_2 = \frac{\hat{q}_2}{\|\hat{q}_2\|}$ and thus $R_{2,2} = \|\hat{q}_2\|$. With this, (5.10) holds, and the new column q_2 is orthogonal to Q_1,

$$Q_1^T q_2 = Q_1^T \frac{\hat{q}_2}{R_{2,2}} = \frac{Q_1^T v_{p+1} - Q_1^T Q_1 Q_1^T v_{p+1}}{R_{2,2}} = 0. \tag{5.12}$$

Now we can completely avoid forming the full residuals, as with $R = R_L R_R = Q_1 R_{1,1} R_R$ the desired Frobenius norm of the residual becomes

$$\|R\|_F = \sqrt{\operatorname{trace}(R^T R)} = \sqrt{\operatorname{trace}(R_R^T R_{1,1}^T Q_1^T Q_1 R_{1,1} R_R)} = \sqrt{\operatorname{trace}(R_R^T R_{1,1}^T R_{1,1} R_R)}. \tag{5.13}$$

To compute the residual norms in this form only a very small matrix of size $n_T \times n_T$ has to be computed. With this, the norm of both residuals in (5.1) and (5.2), $\|R_1\|_F$ and $\|R_2\|_F$, can be computed cheaply without forming the approximations Y and Λ.

Following the guidelines in [10] for linear matrix equations, we also monitor a scaled backward error norm of the two equations, that is,

$$\begin{aligned}\rho_3 = &\frac{\|R_1\|_F}{\tau(\|M_1\|_F\|Z_Y\|_F + \|K\|_F\|Z_\Lambda\|_F + \|M\|_F\|\hat{Y}\|_F) + \|M\|_F\|Z_Y\|_F\|C\|_F} \\ &+ \frac{\|R_2\|_F}{\tau(\|K\|_F\|Z_Y\|_F + \frac{1}{\beta}\|M\|_F\|Z_\Lambda\|_F + \|M\|_F\|Z_Y\|_F\|C\|_F},\end{aligned} \tag{5.14}$$

which takes into account the data order of magnitude. Summarizing, we stop our iteration whenever

$$\max\{\|R_1\|_F, \|R_2\|_F, \rho_3\} \leq \texttt{tol}, \tag{5.15}$$

where `tol` is a prescribed tolerance. Note that this criterion requires that both the relative and absolute quantities satisfy the condition for the iteration to stop. We also remark that the use of the formulation in (5.13) may lead to small inaccuracies adding up to be significantly larger than machine precision. However, these inaccuracies always stayed far below our convergence tolerance, which was usually set even larger, say `tol` $=10^{-6}$.

6. Numerical results.

We now present the performance and flexibility of our method on multiple examples for different PDE-constrained optimization problems. The spatial finite element discretizations of the PDE operator were conducted using the deal.II framework [1] with Q1 finite elements, and the method described in the previous section was implemented in MATLAB R2018b. All experiments were run on a desktop computer with an INTEL Core i7-4770 Quad-core processor running at $4 \times$ 3400 MHz with 32 GB of RAM.

We will show robustness with respect to the discretization sizes n and n_T as well as the control parameter β. Furthermore, we demonstrate the applicability of our method to multiple different setups such as partial observation of the desired state, boundary control, and a different nonsymmetric PDE operator.

Other low-rank solvers. To emphasize the competitiveness of our new approach we compare it with another low-rank method aimed at the same problem class. The idea of exploiting the system structure to avoid forming a large system of equations and finding a low-rank way to compactly represent the solution is not an entirely new approach. Earlier, in [26] the authors developed a technique involving rewriting the problem in a low-rank context and solving it with a preconditioned Krylov subspace solver, namely MINRES, introduced in [15]. We denote this solver as LRMINRES (low-rank MINRES).[1]

Here, the variables are represented in a low-rank matrix format rather than vectors, as $Y = W_Y V_Y^T$, $U = W_U V_U^T$, and $\Lambda = W_\Lambda V_\Lambda^T$, and the system of equations (2.24)–(2.26) is split into a matrix product as

$$\begin{bmatrix}\tau M_1 W_Y & \tau K^T W_\Lambda & M\Lambda\end{bmatrix}\begin{bmatrix}V_Y^T \\ V_\Lambda^T \\ V_\Lambda^T C\end{bmatrix} = \tau M_1 \hat{Y} \tag{6.1}$$

[1]The LRMINRES MATLAB code used for our comparison is available under https://www.tu-chemnitz.de/mathematik/wire/pubs/PubCodes.zip.

for (2.24) and respectively for (2.25) and (2.26). From here, the system of equations is solved with a tailored version of Minres. As opposed to the original version of Minres, the residual and all other upcoming variables are formulated as low-rank matrices rather than vectors and subsequently truncated to keep memory requirements low. Combined with exploiting the structure of the low-rank equation system in the upcoming matrix vector products and preconditioning, this method provides a memory-efficient and fast alternative to established approaches for solving PDE-constrained optimization problems.

A bottleneck in this approach, however, is the setup of a preconditioner. This becomes increasingly difficult for decreasing β. To construct the necessary low-rank preconditioner, a large-scale matrix equation needs to be solved, which significantly affects the overall performance. As opposed to this, our approach directly starts from a matrix equation and gains convergence speed from carefully selecting the subspace, which makes additional preconditioning obsolete. This appears to be more stable with respect to changing β. Therefore, our approach proves to be superior in many setups. Theoretical justification for the effectiveness of rational spaces can be found in the literature; see related references in [25].

6.1. Setup (i): Fully observed heat equation. As a first example PDE we use the heat equation with a full observation; thus $\Omega_1 = \Omega$ and $M_1 = M$. This leads to a simplified version of (2.17) as

$$A_1 X + X C_1 + X I_0 + A_3 X D - Y_1 F_2^T = 0. \tag{6.2}$$

First, we will use this simple setup to investigate the convergence behavior and our choice of stopping criteria by comparing the results to a solution obtained by a direct solver accurate up to machine precision. The constant-in-time desired state $\hat{Y}$ is displayed in Figure 6.1. This constant desired state leads to a low-rank representation of the right-hand side of rank 1. The reference solution was obtained by solving the linear equation system of a small setup with the MATLAB *backslash* operator. We used a discretization of $n = 1089$ and $n_T = 100$.

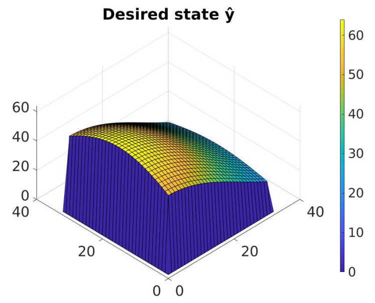


Fig. 6.1. *Constant-in-time desired state for the experiments in section* 6.1 *with* Q_1*—finite elements on a square domain. Here, the desired state is realized with* $n = 1089$ *spatial nodes and Dirichlet boundary conditions set to zero.*

The monitored quantities (5.1), (5.2), (5.14) and the actual relative errors to the reference solution are displayed in Figure 6.2. We see that the monitored quantities are a good estimation for the magnitude of the errors as R_1 stays close above the actual errors. Inserting the direct solution into (2.32) gives a residual of $5.11 \cdot 10^{-9}$. Thus, the different scaling of the matrix equation prohibits further accuracy gains

TABLE 6.1
Example 6.1. *Subspace size p and required time for different discretizations.*

n	1089		4225		16641		66049		263169	
n_T	p	time (s)	p	time (s)	p	time (s)	p	time (s)	p	time (s)
20	8	0.02	11	0.07	11	0.23	11	0.95	12	5.37
100	7	0.03	11	0.09	11	0.25	11	1.02	12	5.37
500	6	0.04	11	0.19	11	0.36	11	1.11	13	6.06
2500	6	0.18	11	0.89	11	1.04	10	1.47	15	8.85

once a high accuracy is reached. This effect is reflected in Figure 6.2 as the last iterations do not provide further accuracy gains.

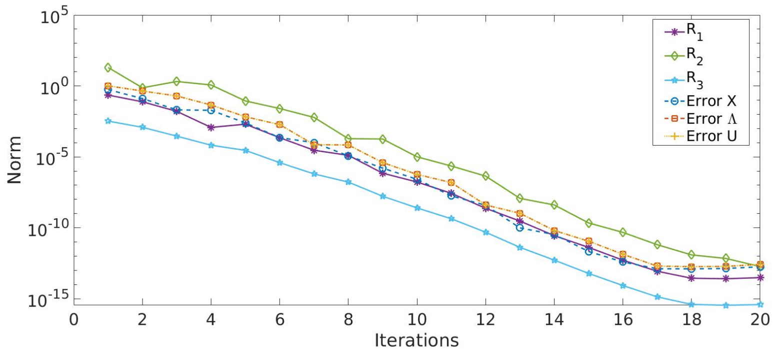


FIG. 6.2. *Observed stopping criteria and actual error progression.*

Example 6.1. We continue with the above simple example, i.e., the heat equation with full observation, $\Omega_1 = \Omega$ and $M_1 = M$, and a constant-in-time desired state. We now investigate the performance of our method with respect to time and space discretization. We vary the number of time steps from 20 to 2500 and the number of spatial discretization nodes from $n = 1089$ to $n = 263169$, which roughly resembles up to a total of 78 million degrees of freedom. Here again we fix the control parameter to $\beta = 10^{-4}$. As seen in Table 6.1, increasing the discretization size barely impacts the resulting subspace sizes p. Additionally, the time needed to solve the optimization problems increases considerably slowly.

TABLE 6.2
Example 6.2. *Comparison of the resulting subspace size p, the solution rank r, and required CPU time between the new* SYS2MATEQ *and another low-rank scheme,* LRMINRES, *for $\hat{Y}$ with rank one.*

n		1089			4225			16641		
β	method	p	r	time (s)	p	r	time (s)	p	r	time (s)
10^{-1}	SYS2MATEQ	6	6	0.09	5	5	0.05	5	5	0.14
	LRMINRES	20	17	4.00	22	21	7.49	22	21	31.59
10^{-3}	SYS2MATEQ	7	7	0.07	7	7	0.09	7	7	0.22
	LRMINRES	11	7	1.80	14	11	5.46	11	10	24.75
10^{-5}	SYS2MATEQ	14	13	0.19	14	13	0.23	14	13	0.53
	LRMINRES	7	6	1.08	7	6	4.75	-	-	-

Example 6.2. With the same data as in Example 6.1 we report on performance comparisons with the low-rank preconditioned Minres (lrminres) proposed in [26] and introduced above. We fixed the number of time steps to $n_T = 100$ and computed solutions for different discretization sizes and control parameters β. The maximum discretization size here is $n = 16641$. The results in Table 6.2 reveal that our method's performance is superior with regard to both required memory and CPU time. Here, our method is labeled as sys2mateq. This acronym stands for *system to matrix equation*. The column denoted by p states the subspace size for sys2mateq, while for lrminres it denotes the maximum rank per vector needed during the iterations, of which up to 15 are required. The column r states the rank of the final solution in both methods. Note that even though sometimes the ranks achieved by lrminres are smaller, the required memory to store the solution is still greater. This is because the lrminres scheme needs to store a low-rank representation $U_1U_2^T$ for each of the three variables, as opposed to sys2mateq, where the same subspace V_p is used for all variables. Additionally, during the iterations our method needs to store only one subspace of size p, whereas lrminres is required to store multiple vectors of size p. Also, our scheme does not rely on the often time-consuming computation of a preconditioner, which is needed in lrminres. The convergence speed depends on the quality of the preconditioner, and for lrminres this requires the solution of two matrix equations as part of the Schur-complement approximation of the saddle-point system. In order to achieve this, lrminres relies on the KPIK method (see [24]) introduced in [5] for a similar situation. This gets harder for large spatial discretizations combined with decreasing β. As a result, the last entry of Table 6.2 does not show convergence for lrminres.

With the same settings as before, we now raise the rank of the desired state $\hat{Y}$ to 6—leading to a larger rank matrix $F_1F_2^T$—and compare both methods once again. Table 6.3 displays the results as before. Again, our method successfully converges with a small subspace size within a very short amount of time, whereas lrminres is considerably slower, especially for larger problem sizes, and does not converge for the largest discretization with β being very small.

Table 6.3
Example 6.2. *Comparison of* sys2mateq *and* lrminres *for rank*$(\hat{Y}) = 6$. *The resulting subspace size* p, *the solution rank* r, *and CPU time are displayed.*

n		1089			4225			16641		
β	method	p	r	time (s)	p	r	time (s)	p	r	time (s)
10^{-1}	sys2mateq	21	21	0.38	21	21	0.47	19	19	0.72
	lrminres	29	22	20.77	27	25	13.38	29	29	23.63
10^{-3}	sys2mateq	31	31	1.08	30	30	1.09	30	29	1.77
	lrminres	15	11	2.22	17	11	11.19	19	15	31.05
10^{-5}	sys2mateq	49	43	3.75	49	44	3.97	51	41	5.88
	lrminres	11	8	1.49	11	9	6.14	-	-	-

We are also interested in the memory savings our method provides. Therefore, we monitor the memory consumption of sys2mateq, lrminres, and a full rank Minres method for the same setups displayed in Table 6.3. For sys2mateq we monitor the memory needed to construct the subspace, the reduced solution, and the setup of the reduced equation system (3.5). For lrminres we monitor the memory for all vectors that are used. For comparison we additionally report the memory consumed by a standard Minres method only to store the required vectors, not taking into account the system matrix and preconditioner. The results are shown in Table 6.4. The quantities refer to the overall memory consumption in megabytes during the process of solving the equation system with the respective method. We see that even with the construction of the reduced equation (3.5), the memory requirements stay well below those of the competing approaches.

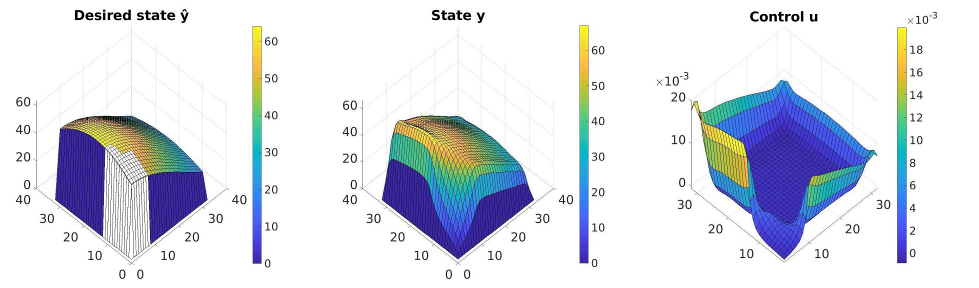
Table 6.4
Example 6.2. *Memory requirements of all considered methods.*

		Memory (MB)		
n	β	SYS2MATEQ	LRMINRES	MINRES
1089	1e-1	4.46	8.11	15.39
	1e-3	8.06	6.14	15.39
	1e-5	17.58	4.54	15.39
4225	1e-1	8.58	19.21	60.46
	1e-3	12.57	19.35	60.46
	1e-5	24.36	16.15	60.46
16641	1e-1	23.50	71.74	241.84
	1e-3	31.86	80.56	241.84
	1e-5	51.91	205.85	241.84

6.2. Setup (ii): Partial observation on heat equation. Until now we only investigated the case where $M = M_1$ and therefore the matrix A_2 is the identity. One highly interesting and challenging problem is the optimization under partial observation, meaning the desired state is only of interest in certain measurement areas on a partial domain Ω_1. This leads to a singular matrix M_1, which further increases the difficulty of the PDE-constrained optimization problem. When we set up the rational Krylov subspace with only A_1 for this setting, we do not reach convergence. Therefore, we add up to two new vectors in each iteration as outlined in section 4.

Example 6.3. We solved the problem with control parameter $\beta = 10^{-4}$ and 100 time steps on a grid of 1089 spatial degrees of freedom. In Figure 6.3a we see the desired state for this problem, and the white area is an example of 100 nonobserved nodes, roughly 10 %. Hence, the corresponding entries in M_1 are set to 0. Figures 6.3b and 6.3c show the resulting state and control, respectively, for a fixed time step $t = 25$. Table 6.5 shows the results when increasing the number of unobserved nodes from $n_0 = 0$ to $n_0 = 900$, which is about 90% of the nodes. Here, the constructed subspaces are far from optimal as the resulting solution rank r is much smaller than the achieved subspace size p in the first column of Table 6.5. Thus, we make use of the truncation modification outlined in section 4. We see that the time and iterations needed to solve this more challenging problem are higher than those for the fully observed case in the table's first row. For the nontruncated case with full V_p, the columns denoted by p for the subspace size and r for the solution rank show a discrepancy which disappears with stricter truncation. When truncating with a tolerance of 10^{-10} both values coincide for all cases. In some cases applying this truncation greatly increases the number of iterations needed to find a solution, but the end result is a higher memory reduction. Thus, this option can be toggled to better reflect the desired behavior.

Unfortunately, the LRMINRES scheme from [26] is not adapted to the case of $M \neq M_1$. Hence, we have no other low-rank method to compare the performance of our method with.



(a) Partially observed desired state (b) State solution for partial observation (c) Control solution for partial observation

Fig. 6.3. *Example* 6.3. *Solution at a fixed time instant for a partial observation.*

Table 6.5
Example 6.3. *Results for different levels of partial observation with subspace truncation. We compare the number of iterations, the resulting subspace size p, and the solution rank r.*

	Full V_p				Truncated V_p, 10^{-12}				Truncated V_p, 10^{-10}			
n_0	time (s)	iters	p	r	time (s)	iters	p	r	time (s)	iters	p	r
0	0.06	7	7	7	0.05	7	7	7	0.06	7	7	7
100	0.67	19	31	29	0.67	19	30	29	0.69	19	23	23
300	1.46	26	42	34	1.41	26	35	34	1.20	28	26	26
500	2.55	32	51	37	2.52	34	38	36	24.41	142	29	29
700	1.50	26	42	37	1.59	27	38	36	2.46	42	29	29
900	0.82	21	34	34	0.84	21	33	33	0.75	21	25	25

6.3. Setup (iii): Boundary control. Another setup of high interest is having a nondistributed control. Here, we take a boundary control problem as an example. This leads to the control u not being distributed on Ω but only on a smaller part of the domain Ω_b. Therefore, we get a smaller mass matrix $M_u \in \mathbb{R}^{n_b \times n_b}$ associated with U present on $n_b < n$ spatial nodes, and $N \in \mathbb{R}^{n \times n_b}$ is now rectangular (tall). Therefore, we have a modification of (2.32) as

$$A_1 X + XC_1 + XI_0 + A_3 XD - F_1 F_2^T = 0 \tag{6.3}$$

with $A_1 = M^{-1}K$ and $A_3 = M^{-1}NM_u^{-1}N^T$. Here, the subspace we compute is derived from A_1 and A_3 as in (4.2).

Example 6.4. As before, we use $n_T = 100$ time steps and a convergence threshold of 10^{-4} for this setup. Results with a constant-in-time desired state for different levels of spatial discretization and different values for β are displayed in Table 6.6. sys2mateq produces robust results with respect to the discretization size as we can only see a small increase in the ranks throughout discretization. Additionally, the results are acquired within a short amount of time even for small β. For larger discretization sizes the required time grows only slightly, as opposed to lrminres, where time requirements increase substantially for larger discretization sizes.

TABLE 6.6
Example 6.4. *Comparisons between the new* SYS2MATEQ *and* LRMINRES *for a boundary control problem with regard to the subspace size p, the solution rank r, and required CPU time.*

n		1089			4225			16641		
β	method	p	r	time (s)	p	r	time (s)	p	r	time (s)
10^{-1}	SYS2MATEQ	72	47	3.92	80	47	7.33	80	47	16.35
	LRMINRES	29	15	29.47	28	16	113.99	28	16	496.70
10^{-3}	SYS2MATEQ	64	46	3.68	68	46	5.08	74	46	14.35
	LRMINRES	25	12	12.70	24	14	35.53	24	14	158.82
10^{-5}	SYS2MATEQ	104	45	12.36	116	44	18.95	107	44	28.48
	LRMINRES	21	16	34.20	21	16	142.83	20	16	694.93

6.4. Setup (iv): Nonsymmetric convection-diffusion PDE.

Until now we only investigated the performance of our method under the constraint of a heat equation. But for more general PDEs our method works just as well, e.g., the convection-diffusion equation,

$$\dot{y} = \varepsilon\nabla^2 y - v \cdot \nabla y + u, \tag{6.4}$$

with diffusion coefficient ε and velocity field v. When $\varepsilon \ll 1$, the equation is convection-dominated, and solving it is a challenging task. As the stiffness matrix K of the resulting equation system is nonsymmetric, we have to adjust the matrix equation accordingly. Thus, we get another modification of (2.32) for this setup as

$$A_1 X \begin{bmatrix} \mathcal{I} & 0 \\ 0 & 0 \end{bmatrix} + M^{-1}K^T X \begin{bmatrix} 0 & 0 \\ 0 & \mathcal{I} \end{bmatrix} + XC_1 + A_2 X I_0 + A_3 X D - F_1 F_2^T = 0. \tag{6.5}$$

Example 6.5. We consider Example 3.1.4 from [23] with a recirculating wind $v(x, y) = (2y(1 - x^2), -2x(1 - y^2))$ as the underlying model. The velocity field v is displayed in Figure 6.4a. In Figures 6.4b–6.4c the desired state and a snapshot of a control solution for $\beta = 10^{-5}$ are displayed for $n = 4225$ and $n_T = 100$. The SYS2MATEQ method produces reliable results throughout a range of different values for ε and β as reported in Table 6.7. Even very small values in ε do not pose a problem to the solver.

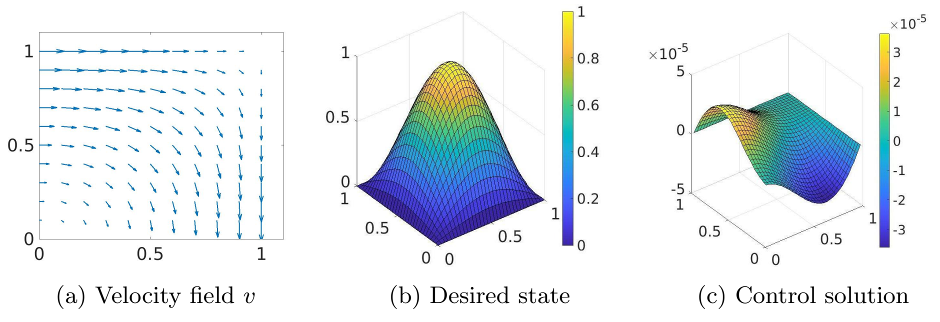


FIG. 6.4. *Example* 6.5. *Solution at a fixed time instant for convection-diffusion problem.*

TABLE 6.7
Example 6.5. *Results for different diffusion coefficients ε for the convection-diffusion problem with different parameters β and different spatial discretizations with regard to required subspace size p, solution rank r, and CPU time.*

n		4225			16641			66049		
β	ε	p	r	time	p	r	time	p	r	time
10^{-1}	10^{-0}	2	2	0.18	2	2	0.37	2	2	4.44
	10^{-1}	10	10	0.27	8	8	0.79	2	2	3.07
	10^{-2}	22	22	0.72	18	18	2.31	10	10	7.78
	10^{-3}	24	24	0.91	18	18	3.15	10	10	11.04
10^{-3}	10^{-0}	12	12	0.44	14	14	1.67	10	10	9.92
	10^{-1}	18	18	0.53	16	16	1.71	18	18	14.63
	10^{-2}	6	6	0.14	4	4	0.49	4	4	3.10
	10^{-3}	56	43	5.82	6	6	1.49	4	4	6.00
10^{-5}	10^{-0}	52	23	4.47	62	22	13.45	46	20	42.52
	10^{-1}	52	22	4.67	46	20	7.92	38	19	32.55
	10^{-2}	6	6	0.12	4	4	0.51	4	4	3.03
	10^{-3}	6	6	0.19	4	4	1.15	4	4	5.90

7. Conclusion and outlook.

We proposed a new scheme to solve a class of large-scale PDE-constrained optimization problems in a low-rank format. This method relied on the reformulation of the KKT saddle-point system into a matrix equation, which was subsequently projected onto a low dimensional space generated with rational Krylov-type iterations. We showed that the method's convergence is robust with respect to different discretizations and parameters. Furthermore, we demonstrated higher memory and time savings compared to an established low-rank scheme. Additionally, SYS2MATEQ is very flexible with respect to different constraints such as nonsymmetric PDE operators and partial state observation.

In the future, we plan on further investigating the subspace choice and the performance of our scheme for other challenging setups. Further improvements are expected from realizing a truncation or restarting mechanism in cases where the subspaces become unexpectedly large.

REFERENCES

[1] W. BANGERTH, R. HARTMANN, AND G. KANSCHAT, *deal.II—a general-purpose object-oriented finite element library*, ACM Trans. Math. Software, 33 (2007), 24, https://doi.org/10.1145/1268776.1268779.

[2] P. BENNER, A. ONWUNTA, AND M. STOLL, *Block-diagonal preconditioning for optimal control problems constrained by PDEs with uncertain inputs*, SIAM J. Matrix Anal. Appl., 37 (2016), pp. 491–518, https://doi.org/10.1137/15M1018502.

[3] M. BENZI, G. H. GOLUB, AND J. LIESEN, *Numerical solution of saddle point problems*, Acta Numer., 14 (2005), pp. 1–137, https://doi.org/10.1017/S0962492904000212.

[4] L. T. BIEGLER, O. GHATTAS, M. HEINKENSCHLOSS, AND B. VAN BLOEMEN WAANDERS, *Large-scale PDE-constrained optimization: An introduction*, in Large-Scale PDE-Constrained Optimization, Springer, New York, 2003, pp. 3–13, https://doi.org/10.1007/978-3-642-55508-4.

[5] T. BREITEN, V. SIMONCINI, AND M. STOLL, *Fast iterative solvers for fractional differential equations*, Electron. Trans. Numer. Anal., 45 (2016), pp. 107–132.

[6] A. CICHOCKI, A.-H. PHAN, Q. ZHAO, N. LEE, I. V. OSELEDETS, M. SUGIYAMA, AND D. MANDIC, *Tensor networks for dimensionality reduction and large-scale optimizations. Part* 2. *Applications and future perspectives*, Found. Trends Mach. Learn., 9 (2017), pp. 431–673, https://doi.org/10.1561/2200000067.

[7] S. DOLGOV, J. W. PEARSON, D. V. SAVOSTYANOV, AND M. STOLL, *Fast tensor product solvers for optimization problems with fractional differential equations as constraints*, Appl. Math.

Comput., 273 (2016), pp. 604–623, https://doi.org/10.1016/j.amc.2015.09.042.

[8] V. Druskin and V. Simoncini, *Adaptive rational Krylov subspaces for large-scale dynamical systems*, Systems Control Lett., 60 (2011), pp. 546–560, https://doi.org/10.1016/j.sysconle.2011.04.013.

[9] M. Fisher, J. Nocedal, Y. Trémolet, and S. J. Wright, *Data assimilation in weather forecasting: A case study in PDE-constrained optimization*, Optim. Engrg., 10 (2009), pp. 409–426, https://doi.org/10.1007/s11081-008-9051-5.

[10] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, Philadelphia, 2002, https://doi.org/10.1137/1.9780898718027.

[11] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich, *Optimization with PDE Constraints*, Math. Model. Theory Appl. 23, Springer-Verlag, New York, 2009, https://doi.org/10.1007/978-1-4020-8839-1.

[12] K. Ito and K. Kunisch, *Lagrange Multiplier Approach to Variational Problems and Applications*, Adv. Des. Control 15, SIAM, Philadelphia, 2008, https://doi.org/10.1137/1.9780898718614.

[13] M. L. Juncosa and D. Young, *On the Crank-Nicolson procedure for solving parabolic partial differential equations*, Proc. Cambridge Philos. Soc., 53 (1957), p. 448–461, https://doi.org/10.1017/S0305004100032436.

[14] H. Mingyou and V. Thomée, *On the backward Euler method for parabolic equations with rough initial data*, SIAM J. Numer. Anal., 19 (1982), pp. 599–603, https://doi.org/10.1137/0719040.

[15] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629, https://doi.org/10.1137/0712047.

[16] D. Palitta, *Matrix equation techniques for certain evolutionary partial differential equations*, J. Sci. Comput., 87 (2021), 99, https://doi.org/10.1007/s10915-021-01515-x.

[17] J. W. Pearson, M. Stoll, and A. J. Wathen, *Regularization-robust preconditioners for time-dependent PDE-constrained optimization problems*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 1126–1152, https://doi.org/10.1137/110847949.

[18] C. E. Powell, D. Silvester, and V. Simoncini, *An efficient reduced basis solver for stochastic Galerkin matrix equations*, SIAM J. Sci. Comput., 39 (2017), pp. A141–A163, https://doi.org/10.1137/15M1032399.

[19] T. Rees, H. S. Dollar, and A. J. Wathen, *Optimal solvers for PDE-constrained optimization*, SIAM J. Sci. Comput., 32 (2010), pp. 271–298, https://doi.org/10.1137/080727154.

[20] T. Rees and M. Stoll, *Block-triangular preconditioners for PDE-constrained optimization*, Numer. Linear Algebra Appl., 17 (2010), pp. 977–996, https://doi.org/10.1002/nla.693.

[21] O. Schenk, M. Manguoglu, A. Sameh, M. Christen, and M. Sathe, *Parallel scalable PDE-constrained optimization: Antenna identification in hyperthermia cancer treatment planning*, Comput. Sci. Res. Develop., 23 (2009), pp. 177–183, https://doi.org/10.1007/s00450-009-0080-x.

[22] J. Schöberl and W. Zulehner, *Symmetric indefinite preconditioners for saddle point problems with applications to PDE-constrained optimization problems*, SIAM J. Matrix Anal. Appl., 29 (2007), pp. 752–773, https://doi.org/10.1137/060660977.

[23] D. Silvester, H. Elman, and A. Wathen, *Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics*, Oxford University Press, Oxford, UK, 2005, https://doi.org/10.1093/acprof:oso/9780199678792.001.0001.

[24] V. Simoncini, *A new iterative method for solving large-scale Lyapunov matrix equations*, SIAM J. Sci. Comput., 29 (2007), pp. 1268–1288, https://doi.org/10.1137/06066120X.

[25] V. Simoncini, *Computational methods for linear matrix equations*, SIAM Rev., 58 (2016), pp. 377–441, https://doi.org/10.1137/130912839.

[26] M. Stoll and T. Breiten, *A low-rank in time approach to PDE-constrained optimization*, SIAM J. Sci. Comput., 37 (2015), pp. B1–B29, https://doi.org/10.1137/130926365.

[27] F. Tröltzsch, *Optimal Control of Partial Differential Equations: Theory, Methods and Applications*, Grad. Stud. Math. 112, AMS, Providence, RI, 2010, https://doi.org/10.1090/gsm/112.