RC24403 (W0711-032) November 6, 2007
Mathematics

IBM Research Report

Inertia Revealing Preconditioning for Large-Scale Nonconvex Constrained Optimization

Olaf Schenk
Department of Computer Science
University of Basel
Klingelbergstrasse 50
CH-4056 Basel, Switzerland

Andreas Wächter
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598, USA

Martin Weiser
Zuse Institute Berlin (ZIB)
14195 Berlin, Germany

Inertia Revealing Preconditioning For Large-Scale Nonconvex Constrained Optimization

Olaf Schenk* Andreas Wächter† Martin Weiser‡

November 5, 2007

Abstract

Fast nonlinear programming methods following the all-at-once approach usually employ Newton's method for solving linearized Karush-Kuhn-Tucker (KKT) systems. In nonconvex problems, the Newton direction is only guaranteed to be a descent direction if the Hessian of the Lagrange function is positive definite on the nullspace of the active constraints, otherwise some modifications to Newton's method are necessary. This condition can be verified using the signs of the KKT's eigenvalues (inertia), which are usually available from direct solvers for the arising linear saddle point problems. Iterative solvers are mandatory for very large-scale problems, but in general do not provide the inertia. Here we present a preconditioner based on a multilevel incomplete LBL^T factorization, from which an approximation of the inertia can be obtained. The suitability of the heuristics for application in optimization methods is verified on an interior point method applied to the CUTE and COPS test problems, on large-scale 3D PDE-constrained optimal control problems, as well as 3D PDE-constrained optimization in biomedical cancer hyperthermia treatment planning. The efficiency of the preconditioner is demonstrated on convex and nonconvex problems with 150^3 state variables and 150^2 control variables, both subject to bound constraints.

1 Introduction

In this paper we address the numerical solution of nonlinear optimization problems (NLPs) given as

$$\min_{x\in\mathbb{R}^n} f(x) \quad \text{subject to } c(x)=0, \quad x_L \le x \le x_U \tag{1}$$

where the objective function $f:\mathbb{R}^n \longrightarrow \mathbb{R}$ and the constraints function $c:\mathbb{R}^n \longrightarrow \mathbb{R}^m$ are assumed to be C^2. Note that problems with general nonlinear inequality constraints can be equivalently reformulated into the above statement by means of slack variables.

Despite the vast amount of literature published in this area (cf. [25] and the references therein), large-scale nonconvex nonlinear constrained programming remains to be algorithmically and computationally challenging. Here, we address one specific open aspect of that class of problems: How to compute or approximate the inertia of the Hessian of the Lagrange functional (a.k.a. KKT matrix) when an iterative linear solver is used to obtain the search directions of the optimization algorithm.

*Department of Computer Science, University of Basel, Klingelbergstrasse 50, CH-4056 Basel, Switzerland (olaf.schenk@unibas.ch).

†IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA (andreasw@us.ibm.com).

‡Zuse Institute Berlin (ZIB), 14195 Berlin, Germany (weiser@zib.de). The work of this author was supported by the DFG research center MATHEON "Mathematics for Key Technologies" in Berlin.

This piece of information can be employed to generate optimization steps that promote convergence to minimizers of a nonconvex optimization problem, and not merely to any stationary point, which is often unsatisfactory.

A rich source for large-scale nonconvex NLPs are PDE-constrained optimal control problems. Here, the unknowns $x = (y, u)$ can often be partitioned into control variables $u \in \mathbb{R}^{n-m}$ and state variables $y \in \mathbb{R}^m$. Usually, the state equation $c(y, u) = 0$ defines (locally) unique states $y = y(u)$ in terms of given controls. The *reduced* solution approach, which considers the reformulation

$$\min_{u\in\mathbb{R}^{n-m}} f(y(u), u) \quad \text{subject to} \quad y_L \le y \le y_U, \quad u_L \le u \le u_U$$

is formally attractive, but requires to solve the nonlinear state equation to high accuracy in every step of the optimization algorithm. In addition, the PDE solver has to provide the derivative information $\frac{\partial y}{\partial u}$ (and possibly $\frac{\partial^2 y}{\partial u^2}$ for an optimization algorithm with fast local convergence), which is often implementationally and computationally challenging or impossible. Furthermore, in problems where a partitioning $x = (y, u)$ of the unknowns is not induced by the problem structure, computing a reduced basis is usually only practical for small problems.

On the other hand, in the *all-at-once* approach, optimality and feasibility are reached simultaneously. The infinite-dimensional differential equations are discretized, and only linearized state equations have to be solved in every optimization step, which can improve the overall performance tremendously (cf. [4, 5, 26]). Drawbacks are a significantly increased number of variables and the necessity to handle the equality constraints explicitly. However, the first and second derivatives of the discretized constraints and objective are typically readily available, sparse, and well-structured. Unfortunately, direct linear solvers applied to the step computation system can create a lot of fill-in. This is particularly pronounced for discretized optimal control problems with three-dimensional PDE constraints. Here, the memory requirements induced by the fill-in may practically rule out direct solvers for very large problems, in which case *preconditioned iterative* methods are the only viable alternative.

One important aspect in nonconvex optimization is the usage of second-order information. On the analytical side, sufficient and necessary optimality conditions require the Hessian of the Lagrangian function to be positive definite or positive semidefinite, respectively, on the nullspace of the linearized constraints. On the algorithmic side, positive definiteness of the Hessian on the constraints' nullspace is necessary for Newton type methods to compute downhill tangential steps. Thus, for both, verification of a candidate solution as well as computing good search directions, it is important to check whether the Hessian is positive definite on the nullspace of the linearized constraints. This information is given by the inertia of the Lagrange functional's second derivative. How to reliably obtain the inertia from iterative methods, however, is essentially an open research problem. In the current paper we propose an algebraic multilevel preconditioning technique (not to be confused with algebraic multigrid) based on an incomplete LBL^T factorization using maximum weighted matchings that aims to reveal the inertia of the iteration matrix and is used in a primal-dual interior point method. Although no theoretical guarantees are provided, we will show that the practical performance of this approach is very satisfactory.

For completeness, we point out that several other approaches have been proposed in the literature to handle nonconvexity in nonlinear optimization such as [13, 17, 23]. However, these require the direct factorization of a matrix that contains at least the Jacobian of the constraint matrix, i.e., the discretized PDE. A recent exception to this is [12] which allows the use of iterative linear solvers for nonconvex equality constrained optimization without specific preconditioner requirements, but practical performance has not yet been tested on large problems.

The remainder of the paper is structured as follows. Section 2 reviews a typical interior-point NLP optimization algorithm, introduces the matrix that commonly appears in linear systems for the search direction computation, and discusses how information about the inertia of this matrix can be used to promote convergence to minimizers of the problem. Section 3 describes the novel inertia-revealing preconditioning technique in detail. The practical performance of the preconditioner is assessed in Section 4, specifically with respect to the exactness of the inertia approximation and its usability for nonconvex optimization. Finally, Section 5 presents the application of the proposed approach in hyperthermia cancer treatment planning.

2 Saddle point matrices in nonlinear nonconvex optimization

In order to solve optimization problems (1) coming from the all-at-once approach for PDE-constrained optimal control, the optimization method needs to be able to handle a large number of variables and bound constraints. At current time, the most successful methods for NLP are active set Sequential Quadratic Programming (SQP) methods and Interior-Point Methods (IPM). Since SQP methods are currently not able to efficiently handle very large problems with millions of (bounded) variables due to the combinatorial complexity of identifying the active set, we will concentrate in this work on IPMs.

Interior point methods avoid the high complexity introduced by inequality constraints by replacing them by a barrier term which is added to the objective function. One then obtains the barrier problem

$$\min_{x\in\mathbb{R}^n} \varphi_\mu(x) = f(x) - \mu \sum_{i=0}^{n} \ln(x^{(i)}) \quad \text{subject to} \quad c(x) = 0, \tag{2}$$

where $\mu > 0$ is a barrier parameter, and $x^{(i)}$ denotes the i-th component of the vector x. For the sake of simplicity, we assume here that $x_L = 0$ and $x_U = \infty$, but the approach can easily be generalized.

The literature in the past 15 years has presented a number of different IPMs [18]. Most methods have in common that steps are computed from the linearization of some formulation of the optimality conditions of (2) and that μ is eventually driven to zero.

The particular method considered in this paper, implemented in the software package Ipopt, reduces the barrier parameter μ monotonically, where a decrease is done every time the corresponding barrier problem is solved to a convergence tolerance $O(\mu)$. At an iteration k with iterate $x_k > 0$ and approximation of the Lagrangian multipliers λ_k, the method computes search directions from a linearization of the optimality conditions for (2),

$$\begin{bmatrix} H_k & A_k \\ A_k^T & 0 \end{bmatrix} \begin{pmatrix} \Delta x_k \\ \Delta\lambda_k \end{pmatrix} = - \begin{pmatrix} \nabla\varphi_\mu(x_k) + A_k\lambda_k \\ c(x_k) \end{pmatrix} \tag{3}$$

Here, A_k^T denotes the Jacobian of the constraint functions at x_k, and H_k is a suitable (primal-dual) approximation of the Hessian of the Lagrangian function for (2). Once a search direction Δx_k has been determined, a line search is performed to find a step size $\alpha_k \in (0, 1]$, using a filter mechanism to test acceptability of trial step sizes. Finally, the new iterates are obtained from $x_{x+1} = x_k + \alpha_k \Delta x_k$ and $\lambda_{x+1} = \lambda_k + \alpha_k \Delta\lambda_k$

Details of the method implemented in Ipopt can be found in [40]. The following discussion is restricted to the features that matter when we want to use an iterative linear solver to solve the system (3).

First, under some circumstances no acceptable step size α_k can be determined for a search direction $(\Delta x_k, \Delta\lambda_k)$. In this case, the filter line search procedure switches to a *restoration phase*, which solves a related optimization problem, aiming to reduce the infeasibility. The search directions are then obtained by solving a linear system of the same dimension as (3) but with a matrix

$$\begin{bmatrix} H_k & A_k \\ A_k^T & -C_k \end{bmatrix} \tag{4}$$

where C_k is a diagonal matrix with positive entries.

Second, as mentioned in the Introduction, NLP solvers need to take special measures if they are solving nonconvex problems, to ensure that they generate descent directions for the globalization mechanism, and to attempt to avoid convergence to non-minimizers. In the considered IPM, this is done by requiring that the Hessian matrix H_k in (3) is positive definite in the null-space of A_k^T, or—equivalently—that the matrix in (3) has exactly n positive and m negative eigenvalues. In IPOPT, if the linear solver determines at an iteration that the number of negative eigenvalues is more than m, the matrix H_k in (3) or (4) is replaced by $H_k + \delta I$ for some $\delta > 0$; a sufficiently large value of δ is determined by a trial-and-error procedure.

In summary, the considered IPM requires solutions of linear systems

$$Kv = \begin{bmatrix} H & A \\ A^T & -C \end{bmatrix} v = b, \tag{5}$$

where the $n \times n$ matrix H is symmetric and potentially indefinite, C is a non-negative diagonal regularization matrix which is often zero, and the $n \times m$ matrix A has full column rank. Although we motivated this linear system in the context of the particular IPM implemented in IPOPT, we emphasize that many currently proposed IPMs compute steps by solving different variations of (5), including those implemented in popular software packages such as KNITRO [41] and LOQO [39]. An efficient solution methods for (5) that is able to provide inertia information could benefit these optimization algorithms as well.

For a detailed survey on solution techniques for large linear saddle-point systems (5), the interested reader should consult [3]. Most direct factorization methods for such indefinite symmetric systems (usually based on variations of the Bunch-Kaufman algorithm [11]) can very easily compute the inertia of the factorized matrix. However, iterative linear solvers do not compute such information on the fly. In the next section we discuss a preconditioner that aims to recover the inertia of the original matrix. Even though this approach is a heuristic, we show that it is working very well in practice.

3 An efficient heuristic to reveal the inertia using multilevel incomplete factorizations

The inertia of a symmetric matrix K—denoted by $\mathrm{In}(K)$—is the integer triple (n, m, z), where n, m and z denote the nonnegative numbers of positive, negative and zeros eigenvalues of K. In this section we make use of Sylvester's law of inertia [22], which states that if K is a symmetric matrix, and T is a nonsingular matrix of the same dimension as K, then $\mathrm{In}(K) = \mathrm{In}(TKT^T)$. For a symmetric Karush-Kuhn-Tucker matrix K, which may be partitioned with a permutation matrix P and scaled with a scaling matrix D into

$$PDKD^TP^T = \begin{pmatrix} B & F^T \\ F & C \end{pmatrix}$$

with B nonsingular, the Schur complement of B in K is defined as $S_C = C - FB^{-1}F^T$. The matrix $PDKD^TP^T$ can be decomposed into

$$PDKD^TP^T = \begin{pmatrix} B & F^T \\ F & C \end{pmatrix} = \begin{pmatrix} L_B & 0 \\ L_F & I \end{pmatrix} \begin{pmatrix} D_B & 0 \\ 0 & S_C \end{pmatrix} \begin{pmatrix} L_B^T & L_F^T \\ 0 & I \end{pmatrix}. \tag{6}$$

Sylvester's law of inertia applied to (6) gives the relation

$$\mathrm{In}(K) = \mathrm{In}(PDKD^TP^T) = \mathrm{In}(D_B) + \mathrm{In}(S_C) \tag{7}$$

When solving an equation with a decomposition approach involving a symmetric indefinite matrix K, which is large and sparse, the permutation matrix P must be chosen in order to maintain both sparsity and numerical accuracy. One can use a diagonal pivoting factorization of the form $PKP^T = LDL^T$, where L is a unit lower triangular matrix, P is a permutation matrix and D is a diagonal matrix; we refer to such a factorization as an LDL^T-factorization. However, for a general indefinite symmetric matrix K, one also has to allow pivots of dimension 1×1 and 2×2. This gives $PKP^T = LBL^T$, where B is a symmetric block-diagonal matrix whose diagonal blocks are of dimension 1×1 and 2×2. We refer to such a factorization as an LBL^T-factorization. Different pivoting strategies have been suggested, see e.g. [10, 11].

The key idea here is to define a series of permutation matrices P and nonsingular scaling matrices D without performing a complete factorization in such a way that the resulting incomplete LBL^T-factorization is as stable as possible and that most of the globally important numerical entries in K are permuted into the diagonal block D_B to serve as potential 1×1 and 2×2 pivots for the inertia computation.

The resulting incomplete factorization leads to a 1×1 and 2×2 diagonal block D_B and a Schur complement S_C. We now present the heuristic to reveal the inertia by an indefinite incomplete multilevel factorization that is mainly based on three parts, which are repeated in a multilevel framework in each subsystem. The components consist of (i) reordering of the system using weighted graph matchings to approximate Gaussian scaling and Gaussian 1×1 and 2×2 pivoting ordering as good as possible, (ii) approximate factorization using inverse-based pivoting and Eigenvalue decompositions of 1×1 and 2×2 diagonal blocks, and (iii) recursive application to the system of the pivot elements that are permuted into the Schur complement.

3.1 Symmetric 1×1 and 2×2 block weighted matchings

The key ingredient for turning this approach into an efficient inertia-revealing multilevel solver consists of the symmetric maximum weighted matching [16, 27, 36]. After the system is reordered into a representation

$$P_s D_s K D_s^T P_s^T = \hat{K}, \tag{8}$$

where $D_s \in \mathbb{R}^{n\times n}$ is a diagonal matrix and $P_s \in \mathbb{R}^{n\times n}$ is a permutation matrix, $\hat{K}$ is expected to have many diagonal blocks of size 1×1 or 2×2 that are well conditioned. Once the diagonal blocks of size 1×1 and 2×2 are built, the associated block graph of $\hat{K}$ is reordered by a symmetric reordering, such as Metis [28], to obtain

$$\Pi^T P_s^T D_s K D_s P_s \Pi = \tilde{K}, \tag{9}$$

where $\Pi \in \mathbb{R}^{n\times n}$ refers to the associated symmetric block permutation that preserves the 1×1 and 2×2 pivot blocks.

3.2 Inverse-based pivoting

Given $\tilde{K}$, we compute an incomplete factorization $LBL^T = \tilde{A} + E$ of $\tilde{K}$. To do this at step k of the algorithm, we have

$$\tilde{K} = \begin{pmatrix} B & F^T \\ F & C \end{pmatrix} = \begin{pmatrix} L_B & 0 \\ L_F & I \end{pmatrix} \begin{pmatrix} D_B & 0 \\ 0 & S_C \end{pmatrix} \begin{pmatrix} L_B^T & L_F^T \\ 0 & I \end{pmatrix}, \tag{10}$$

where $L_B \in \mathbb{R}^{k-1,k-1}$ is lower triangular with unit diagonal and $D_B \in \mathbb{R}^{k-1,k-1}$ is block diagonal with diagonal blocks of sizes 1×1 and 2×2. Also, $S_C = C - L_F D_B^{-1} L_F^T = (s_{ij})_{i,j}$ denotes the approximate Schur complement. To proceed with the incomplete factorization, we perform either a

1×1 update or a 2×2 block update. One possible choice could be to use Bunch's algorithm [9] and this approach has been used in [27]. Here we use a simple criterion based on block diagonal dominance of the leading block column. Depending on the values

$$d_1 = \sum_{j>1} \frac{|s_{j1}|}{|s_{11}|}, \quad d_2 = \sum_{j>2} \left\| (s_{j1}, s_{j2}) \begin{pmatrix} s_{11} & s_{12} \\ s_{12} & s_{22} \end{pmatrix}^{-1} \right\|, \tag{11}$$

we perform a 2×2 update only if $d_2 < d_1$. Here, d_1 and d_2 represent the diagonal dominance of the current row/column. The two leading columns of S_C can be efficiently computed using linked lists [29], and it is not required to have all entries of S_C available.

When applying the (incomplete) 1×1 and 2×2 block factorization LBL^T to $\tilde{K}$ we may still encounter a situation where at step k either $1/|s_{11}|$ or $\|[(s_{ij})_{i,j\leqslant 2}]^{-1}\|$ is large or even infinite. Since we are dealing with an incomplete factorization, we propose to use inverse-based pivoting [6]. Therefore, we require in every step that

$$\left\| \begin{pmatrix} L_B & 0 \\ L_F & I \end{pmatrix}^{-1} \right\| \leqslant \kappa_1 \tag{12}$$

for a prescribed bound κ_1. If after the update using a 1×1 pivot (or 2×2 pivot) the norm of the inverse lower triangular factor fails to be less than κ_1, the update is postponed and the leading rows/columns of L_F are permuted to the end of S_C. Otherwise, depending on whether a 1×1 or a 2×2 pivot has been selected, the entries

$$(s_{j1}/s_{11})_{j>1}, \qquad \left((s_{j1},\ s_{j2}) \begin{pmatrix} s_{11} & s_{12} \\ s_{12} & s_{22} \end{pmatrix}^{-1} \right)_{j>2} \tag{13}$$

become the next (block) column of L, and we drop these entries whenever their absolute value is less than ε/κ_1 for some threshold ε. For a detailed description see [6]. The norm of the inverse can be cheaply estimated using a refined strategy of [14].

3.3 Eigenvalue decomposition of 1×1 and 2×2 pivots

The inverse-based pivoting approach has been successfully used in [33, 34]. In addition to the inverse-based pivoting approach we will resort to a second heuristic to permute 1×1 and 2×2 diagonal blocks into the Schur complement if the absolute value of the eigenvalues α_i of these blocks are smaller than a prescribed bound κ_2:

$$|\alpha_1| < \kappa_2 \ \text{(for a } 1 \times 1 \text{ pivot)}; \quad \min(|\alpha_1|, |\alpha_2|) < \kappa_2 \ \text{(for a } 2 \times 2 \text{ pivot)}. \tag{14}$$

This strategy has the effect that diagonal blocks with potentially small eigenvalues are permuted into the Schur complement and, here, can be computed more exactly in the multilevel framework. This eigenvalue decomposition is another key component to turn the method into an effective inertia-revealing preconditioner.

3.4 Recursive application

After the incomplete matching-based factorization we have an approximate factorization

$$Q^T D_s^T \tilde{K} D_s Q = \begin{pmatrix} L_{11} & 0 \\ L_{21} & I \end{pmatrix} \begin{pmatrix} D_{11} & 0 \\ 0 & S_{22} \end{pmatrix} \begin{pmatrix} L_{11}^T & L_{21}^T \\ 0 & I \end{pmatrix}, \tag{15}$$

where Q represents the product of all permutation matrices that are necessary to maintain sparsity and accuracy, and D_s is the initial scaling matrix which was obtained as a by-product of the weighted matchings. It typically does not pay off to continue the factorization for the remaining matrix S_{22} which consists of the previously postponed updates. Thus, S_{22} is now explicitly computed and the strategies for reordering, scaling, and factorization are recursively applied to S_{22}, leading to a multilevel factorization.

We use a technique called aggressive dropping that sparsifies the triangular factor L a posteriori. Observe that when applying a perturbed triangular factor $\tilde{L}^{-1}$ for preconditioning, we have

$$\tilde{L}^{-1} = (I + E_L)L^{-1} \text{ with } E_L = \tilde{L}^{-1}(L - \tilde{L}),$$

instead of L^{-1}. We can expect that $\tilde{L}^{-1}$ serves as a good approximation to L^{-1} as long as $\|E_L\| \ll 1$. If we obtain $\tilde{L}$ from L by dropping some entry, say l_{ij}, from L, then we have to ensure that

$$\|\tilde{L}^{-1}e_i\| \cdot |l_{ij}| \leqslant \tau \ll 1$$

for some moderate constant $\tau < 1$, e.g., $\tau = 0.1$. This requires that a good estimate for $\nu_i \approx \|\tilde{L}^{-1}e_i\|$ is available for any $i = 1, \ldots, n$. It can be computed using $\tilde{L}^T$ instead of $\tilde{L}$ [6, 14]. Finally, knowing how many entries exist in column j, we drop any l_{ij} such that

$$|l_{ij}| \leqslant \tau/(\nu_i \cdot \#\{l_{kj} : \ l_{kj} \neq 0, k = j+1, \ldots, n\}).$$

3.5 Iterative solution

By construction, the computed incomplete multilevel factorization is symmetric but indefinite. For the iterative solution of linear systems using the multilevel factorization, different Krylov subspace solvers could be used, such as general methods that do not explicitly use symmetry (e.g., GMRES [32]) or methods like SYMMLQ [31], which preserve the symmetry of the original matrix but are devoted only to symmetric positive definite preconditioners. To fully exploit both symmetry and indefiniteness at the same time, we choose the simplified QMR method [20, 21].

4 Numerical results

In this section, we present comparative numerical results for the inertia-revealing multilevel preconditioner in large-scale nonconvex optimization. First, we will provide numerical results on the numerical linear algebra level by using a selection of Karush-Kuhn-Tucker systems. Then, we will evaluate the performance of the method within an interior-point optimization method on the popular CUTE [8] and COPS [7] collections of nonconvex optimization problems. Finally, we present numerical results on several large-scale examples of convex and nonconvex three-dimensional PDE-constrained optimizations.

The numerical test in Sections 4.1, 4.3 and 5 were performed on Intel Xeon servers (2.2 GHz) with either 24 or 32 GB of memory. The results in Section 4.2 were obtained on an AMD Opteron (2.2GHz, 8 GB memory). All codes were compiled by the GCC compiler suite (including GFortran) version 4.2 with the `-O3` optimization option and linked with the ACML library containing BLAS and LAPACK subroutines, optimized for AMD architectures.

The linear saddle-point systems (5) are solved by a preconditioned SQMR iteration, where the initial vector is always chosen as $v_0 = (0, \cdots, 0)^T$. We iterate until either the maximum of 200 iterations has been exceeded or until the residual has been reduced by a factor of either 10^{+4}, 10^{+7} or 10^{+14}. Furthermore, in Sections 4.2–5, where the iterative linear solver is used within the interior-point optimizer, the SQMR iteration is terminated if the last 5 residuals do not change within

4 digits of accuracy. In all experiments, we used the identical set-up for the incomplete multilevel factorization preconditioner with $\kappa_1 = 500$ (norm of the inverse), $\kappa_2 = 0.01$ (eigenvalue bound), and $\tau = 10^{-3}$ (threshold value for dropping). For completeness, let us recall the main software packages we used:

- PARDISO is a fast sparse direct solver developed at the Computer Science Department of the University of Basel [35, 36]. PARDISO/ML is the multilevel iterative solver package which contains all preconditioners used for the presented numerical results. Both packages are available at `http://www.pardiso-project.org`.

- IPOPT is a software package for large-scale nonlinear optimization [40]. This optimization package has been developed at the Department of Mathematical Sciences of the IBM Thomas Watson Research Center and the Carnegie Mellon University. It is available at `https://projects.coin-or.org/Ipopt`.

The performance of the multilevel incomplete factorization preconditioners are compared with each other and also compared with the exact factorization solver in PARDISO.

4.1 Performance comparison of the preconditioners for a selection of Karush-Kuhn-Tucker systems

Our first numerical results compare the computational requirements and inertia approximation of the preconditioners from Section 3 applied to typical matrices of the form (5) appearing in optimization problems.

The first Table 1 provides the set-up times (in seconds) for an exact sparse LBL^T factorization with PARDISO. We selected a representative set of matrices that arise in a nonconvex 3D PDE-constrained optimization process with IPOPT. We also provide the inertia information for these matrices. As mentioned in Section 2, it is important in nonconvex optimization that the current iteration matrix has a specific inertia in order to promote convergence to minimizers of the optimization problem. The IPOPT algorithm modifies the Hessian until the inertia of the full matrix is exactly n positive eigenvalues (number of Hessian equations) and m negative eigenvalues (number of Jacobian equations). For example, for the IPOPT series A, the inertia of the first two matrices (pde-a-01-01, pde-a-01-02) is not exactly $(n, m, 0)$ and a regularization based on [40] is performed until the inertia has exactly the correct number, such as in the third iteration for matrix pde-a-01-03.

The set-up times for the numerical factorization in Table 1 increase significantly with the problem sizes n and m, up to a fill-in factor of 215 in comparison to the numbers of nonzeros in the KKT matrix, resulting in an expensive factorization time of over $2,644$ seconds[1].

Table 2 shows the set-up information for these matrices for three different incomplete multilevel factorization preconditioners, using the parameters given in the introduction of Section 4. The first preconditioner (ILDLT) is an incomplete multilevel factorization solver with inverse-based pivoting and D is a diagonal matrix. The second preconditioner (ILDLT-MATCH) is additionally stabilized using symmetric weighted matchings described extensively in [16, 27, 36]. Our last preconditioner (ILBLT-MATCH) is based on the second preconditioner, but uses in addition a 1×1 and 2×2 diagonal matrix B based on Bunch and Kaufman pivoting. The set-up time for ILBLT-MATCH includes all processing steps that are necessary to build this preconditioner. Note that the high set-up time for the exact factorization in Table 1 can be reduced significantly by two orders of magnitude.

Comparing the inertia information of the three preconditioners in Table 2, we note that by using graph matchings and 1×1 and 2×2 pivoting, we can recover the correct inertia information by using

[1]Note that the performance of sparse direct solvers for symmetric indefinite matrices have been improved by two orders of magnitude during the last years, see, e.g., the results in [38].

a multilevel incomplete factorization method. Also importantly, the fill-in of the last preconditioner (ILBLT-MATCH) for these 3D PDE-constrained optimization matrices is only a factor of four to five, which makes this kind of multilevel preconditioner very attractive for large-scale optimization problems, which will be discussed in Section 4.3.

In Tables 3 and 4, we test the preconditioners against the exact sparse factorization for a set of publicly available symmetric indefinite saddle-point matrices described in [24]. Interestingly, the preconditioners behave very similar as in Table 2 and almost all inertia information can be revealed by using the third preconditioner. In order to demonstrate that our inertia revealing method is only a heuristic (that still performs very well in practise), we point out the results for matrix NCVXQP7. For this matrix the inertia computed by the preconditioner was not correct and we observed an overestimate of 15 positive eigenvalues compared to the exact inertia.

Finally, in Table 5 we present the convergence history of the iterative linear solver for a selection of matrices by using the ILBLT-MATCH preconditioner. It can be seen that it the linear systems can be solved very efficiently, with only a few numbers of SQMR iterations, even to a very tight tolerance of a residual reduction of 10^{-14}.

Based on these positive numerical results, we will use ILDLT-MATCH with SQMR in the interior-point optimization solver in the next sections and will refer to it as the PARDISO/ML method.

4.2 Numerical results for standard NLP test sets

In this section we assess the practical performance of the inertia revealing heuristic described earlier. The goal is to see whether the inertia information made available by the preconditioner ILDLT-MATCH is sufficiently reliable for a general-purpose optimization code to solve nonlinear nonconvex optimization problems contained in standard collections of test problems.

The first test set consists of 720 problems from the CUTE [8] collection, as provided by Benson [2] in the AMPL modeling language [19]. Here, we omitted problems which are unbounded, infeasible, or have too few degrees of freedom. The size of the problems varies between 1 – 50,000 variables (including slack variables for reformulated inequality constraints) and 0 – 14,000 constraints. The second test consists of 65 COPS [7] problems (Version 3.0)[2]. Those problems have 150 – 20,496 variables, and 0 – 20098, constraints. We note here that the AMPL preprocessor was disabled for the CUTE problems, and enabled for the COPS problems.

Tables 6 and 7 show the outcome of the optimization for the two test sets with different options to solve the linear systems. Comparing the iterative solver with the direct linear solver shows that the robustness is almost identical, and most cases with unsuccessful outcome for the iterative solver that have been solved with the direct solver are due to exceeding the time limit. Since these problems are not large, the iterative solve option usually requires more CPU time than the direct linear solver. Furthermore, ignoring the inertia leads to a significant reduction in robustness of the optimization code. In particular, in a number of problems the iterates diverge or the optimization algorithm fails because the generated steps do not have the descent properties that are required in order for IPOPT to generate a new iterate.

A comparison of the final objective function values in successful outcomes obtained for different options is presented in Table 8. As can be seen, ignoring the inertia leads to worse final objective function values in many cases, while the values are identical for the direct linear solver and the iterative solver options, in almost all cases. We note here that problems in the test sets have frequently several local minimizers, so that a convergence to different final solutions is not unexpected. However, the fact that the inertia-ignoring option leads to worse solutions in so many cases seems due to

[2]We excluded the first instance of the `tetra` example since the gradient of the objective function could not be evaluated at the starting point provided by AMPL.

convergence to maximizers or saddle-points since the search directions do not promote necessarily a decrease in the objective function.

These observations show that the inertia-revealing preconditioner is indeed able to provide estimates of the inertia of the linear systems encountered in the optimization algorithm that is sufficiently close to the exact inertia to allow robust optimization in most practical cases, at least measured in the standard and non-trivial test sets CUTE and COPS.

4.3 Numerical results for convex and nonconvex three-dimensional PDE constrained optimization problems

As a large-scale nonlinear programming example we choose a nonlinear PDE-constrained optimization problem with homogeneous Neumann boundary conditions. The domain is $\Omega = (0,1) \times (0,1) \times (0,1)$, and the goal is to compute the optimal boundary control $u(x)$ and state $y(x)$ with respect to $x = (x_1, x_2, x_3)$ that minimizes a convex

$$f(y,u) = \frac{1}{2} \int_{\Omega} (y(x) - y_t(x))^2 \; dx + \frac{\alpha}{2} \int_{\Omega} u(x)^2 \; dx \tag{16}$$

or nonconvex

$$f(y,u) = \frac{1}{2} \int_{\Omega} BT(y(x) - y_t(x)) \; dx + \frac{\alpha}{2} \int_{\Omega} u(x)^2 \; dx \tag{17}$$

objective function. Here, the parameter in the Tikhonov regularization is chosen as $\alpha = 10^{-4}$, and $BT(t)$ is a C^2 approximation of the nonconvex Beaton-Tukey function, given as

$$BT(t) = \begin{cases} (\frac{3}{2} - \frac{9C}{8B})t^2 + (\frac{7}{4}\frac{C}{B^3} - \frac{3}{2B^2})t^4 + (\frac{1}{2B^4} - \frac{5C}{8B^5})t^6 & \text{if } |t| \le B \\ \frac{B^2}{2} + C(|t| - B) & \text{if } |t| > B \end{cases}$$

with a cutoff value $B = 0.25$ and with $C = 0.01$. The original Beaton-Tukey function [1] has a discontinuous second derivative at $|t| = B$, and is constant for $|t| > B$. While the original function is a popular robust quality measure for the fit of observations $y(x)$ with respect to given data $y_t(x)$, our approximation provides the smoothness properties required by a second-order optimization method, and the slope $|BT'(t)| = C > 0$ informs the optimizer that smaller deviations are more desirable, even if $|t| > B$.

The PDE constraints in our example are defined by the elliptic operator

$$\text{in } \Omega : \quad -\Delta y(x) = 20, \quad y(x) \le 3.5, \quad y_t(x) = 3 + 5x_1(x_1 - 1)x_2(x_2 - 1), \tag{18}$$

and the boundary conditions are

$$\text{on } \partial\Omega : \quad y(x) = u(x), \quad 0 \le u(x) \le 10. \tag{19}$$

This problem is a variation of Example 5.1 in [30].

We use an equidistant Cartesian grid with N points in each space direction to discretize (16)–(19), where the usual 7-point stencil is used for discretizing the Laplace operator. The size of the nonlinear programming problem as a function of the discretization parameter N is shown in Table 9.

Tables 10 (convex) and 11 (nonconvex) shows the timing results for the complete interior-point optimization algorithm with IPOPT and the ILBLT-MATCH preconditioner.

In the convex PDE optimization example in Table 10, a residual reduction of already four orders of magnitude is sufficient to converge to the correct solution of the optimization problem. The iterative

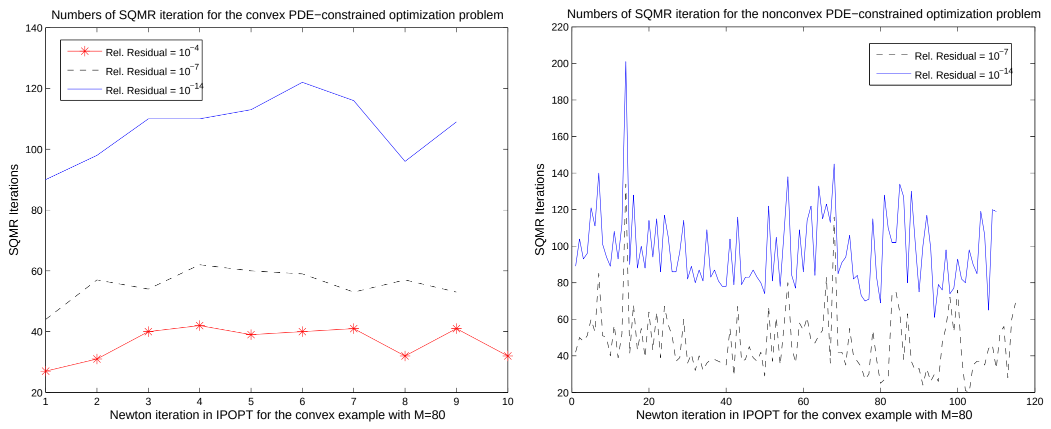

Figure 1: Number of Sqmr iterations for the convex (left) and nonconvex (right) PDE-constrained optimization example with $N = 80$ to reduce the residual by either 4, 7 or 14 order of magnitudes.

solver performs very well in this example and it is very memory efficient. For the larger example that have more than $N = 80$ discretization points in each direction we observed a 32-bit integer overflow in the direct solver, whereas with our preconditioner it was possible to solve a 3D PDE-constrained optimization problem with $N = 150$ in less than 3 hours on a single processor. This problem has more than 3 million state variables with upper bounds, over one hundred thousand control variables with both upper and lower bounds, and about 91 millions of nonzeros in the Jacobian matrix.

The nonconvex optimization example in Table 10 is computationally challenging and a higher residual reduction is necessary in order to converge to the optimal solution while not impeding the convergence of Newton's method. In general, the iterative method requires slightly more Newton iterations than the direct solver for convergence. The reason is that the preconditioner has not always computed the inertia correctly and a slight increase in Newton iterations is visible by comparing the results for the direct and iterative solver. However, the iterative method is much more memory efficient and faster even if a residual reduction of 14 orders of magnitude is requested. Note that all these nonconvex examples would fail if one would ignore the inertia in Ipopt during the Newton iteration.

Finally, in Figure 4.3, we plot the convergence histories of Sqmr iterations for both the convex and nonconvex example with $N = 80$. It is clearly shown that the number of iterations is in general rather small. For example, it is possible to reduce the residual by 14 orders of magnitude within typically 100 Sqmr iterations. Note that $N = 80$ represents already a large-scale saddle-point matrix with over 500,000 equations.

5 Numerical results for three-dimensional optimal hyperthermia treatment planning problems

Regional hyperthermia is a cancer therapy that aims at heating large and deeply seated tumors by means of radio wave absorption. Heating tumors above a temperature of about 41°C makes them more susceptible to an accompanying radio or chemo therapy. Modern hyperthermia applicators operating at around 100MHz provide 12 antennas for which the amplitude and phase can be controlled independently. The squared amplitude of the resulting superposed electrical field gives the energy density absorbed by the tissue. The generated heat is dissipated and transported by blood

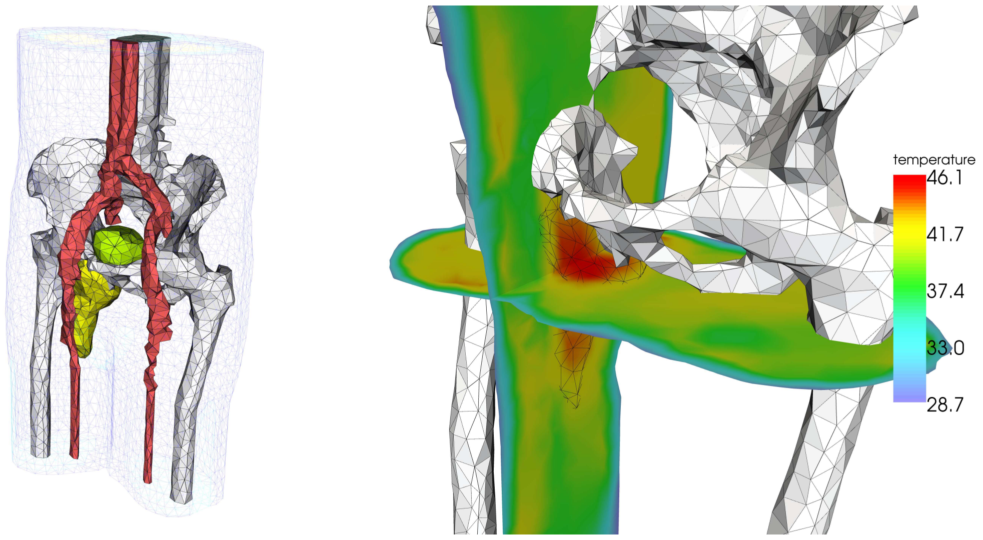

Figure 2: Left: Coarse finite element grid of the patient geometry. Shown are bones, large vessels, the bladder, and the tumor located in the lower right part of the pelvic region. Right: Optimal temperature distribution in the vicinity of the tumor.

flow, which finally results in a temperature profile inside the patient's body.

For designing an individually optimal therapy, amplitudes and phases have to be selected such that the tumor temperature is maximized. On the other hand, in order not to damage healthy tissue, certain temperature constraints have to be respected. A simple model leads to the following nonlinear nonconvex optimal control problem:

$$\min_{y\in H^1(\Omega),u\in\mathbb{C}^{12}} -\int_{\Omega_t} f(y)\,dx$$

subject to

$$\begin{aligned} -\operatorname{div}(\kappa\nabla y) + c_b w(y)\,(y-y_a) &= \frac{\sigma}{2}\left|\sum_i u_i E_i\right|^2 && \text{in } \Omega \qquad (20)\\ \gamma\partial_n y &= g - y && \text{on } \partial\Omega\\ y &\le y_{\max} && \text{in } \Omega\backslash\Omega_t. \end{aligned}$$

Here, Ω is the part of the patient's body that is affected, $\Omega_t \subset \Omega$ is the domain occupied by tumor tissue, κ is the heat diffusion coefficient, c_b the specific heat of blood, $w(y)$ the temperature-dependent perfusion, y_a the arterial blood temperature, g the exterior temperature, γ the heat transfer coefficient, σ the electrical conductivity, u_i the complex control of antenna i, and E_i the corresponding electrical field. The temperature constraint $y_{\max}$ is piecewise constant on each tissue type. For actual data we refer to [15]. Different types of cost functionals are in use. For simplicity, here we use a tracking type form with $f(y) = (y-50)^2$. As is apparent from (20), a simultaneous phase shift of all antennas has no effect on the temperature distribution. For this reason, one of the phases can been fixed, such that the control variables can be interpreted as a vector in $\mathbb{R}^{23}$.

The problem has been discretized by finite elements (FE) on a tetrahedral grid with 14334 vertices (see Fig. 2). For the simple case of constant perfusion, the FE discretization yields the following finite-dimensional NLP:

$$\min \frac{1}{2}(y-50)^T M_{ht}(y-50)$$

subject to

$$\begin{aligned} A_h y + M_{hw}(y-y_a) + D_h(y-g) - u^T E_h u &= 0 \\ y &\leq y_{\max} \end{aligned}$$

Here, A_h is the stiffness matrix, M_{ht} and M_{hw} denote the mass matrices scaled by χ_{Ω_t} and w, respectively, D_h discretizes the Robin boundary conditions, and E_h is the tensor mapping the control to the SAR. The associated KKT matrix is

$$\begin{pmatrix} \lambda_h^T E_h & & (E_h u)^T \\ & M_{ht} & A_h + M_{hw} + D_h \\ E_h u & A_h + M_{hw} + D_h & \end{pmatrix}.$$

For the simple case of linear finite elements, the discretized NLP data scales with the mesh size h as specified in Table 12. The size of the nonlinear programming problem for different FE discretizations is shown in Table 13. Table 14 shows the timing results for Ipopt combined with the ilblt-match preconditioner. For some discretizations, the iterative solver leads to much more Newton iterations than the direct solver, even with a requested residual reduction factor of 10^{-14}. This indicates that the inertia is not always computed correctly. However, the approximation of the inertia is still sufficiently accurate to allow a robust and reliable convergence of the NLP solver. Since on the one hand the evaluation of the NLP functions is quite expensive and on the other hand the problem structure leads to relatively low fill-in in the direct solver, the larger number of Newton iterations compensates the efficiency gained by the iterative solution. The remaining advantage of the incomplete factorization in this particular FE application is its lower memory requirement, which allows to solve larger problems than those that could be addressed by the direct solver.

6 Conclusion

We considered the use of preconditioned iterative linear solvers to solve large-scale nonlinear optimization problems. One way to handle nonconvexity and generate search directions that promote convergence to local minimizers is to ensure a specific inertia of the saddle-point matrix used in the computation of optimization steps. We examined the effectiveness of several new preconditioners for nonconvex problems. At the heart of these preconditioners lies the use of symmetric matchings [27] on each level, and in the preconditioning stage the use of the inertia-revealing, inverse-based and eigenvalue-bounded incomplete factorization preconditioner Pardiso/Ml [37]. The method is able to reveal the inertia of the original matrix sufficiently accurate. It is reliable and robust enough to allow a general-purpose interior-point optimization code to solve a large variety of nonlinear nonconvex optimization problems.

The resulting algorithm was shown to be able to solve very large-scale difficult discretized three-dimensional PDE-constrained optimization problems, which were intractable when a direct linear solver was used. The practical relevance was demonstrated on a hyperthermia cancer treatment application.

The encouraging results presented in this paper indicate that this method might also be successfully applied to other large-scale problems arising in convex and nonconvex large-scale PDE-constrained optimizations.

Acknowledgments

The authors would like to thank J. Gellermann (Charité) for providing clinical data.

References

[1] A. E. Beaton and J. W. Tukey, *The fitting of power series, meaning polynomials, illustrated on band-spectroscopic data*, Technometrics, 16 (1974), pp. 147–185.

[2] H. Y. Benson, *AMPL formulation of CUTE models.* See `http://www.sor.princeton.edu/~rvdb/ampl/nlmodels/cute/`.

[3] M. Benzi, G. H. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numerica, (2005), pp. 1–137.

[4] G. Biros and O. Ghattas, *Parallel Lagrange-Newton–Krylov-Schur methods for PDE-constrained optimization. II: The Lagrange-Newton solver and its application to optimal control of steady viscous flows.*, SIAM J. Scientific Computing, 27 (2005), pp. 714–739.

[5] ——, *Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization. I: The Krylov-Schur solver.*, SIAM J. Scientific Computing, 27 (2005), pp. 687–713.

[6] M. Bollhöfer and Y. Saad, *Multilevel preconditioners constructed from inverse–based ILUs*, SIAM J. Scientific Computing, 27 (2006), pp. 1627–1650.

[7] A. S. Bondarenko, D. M. Bortz, and J. J. Moré, *COPS: Large-scale nonlinearly constrained optimization problems*, Tech. Rep. ANL/MCS-TM-237, Argonne National Laboratory, Argonne, IL, USA, 1998. revised October 1999.

[8] I. Bongartz, A. R. Conn, N. I. M. Gould, and P. L. Toint, *CUTE: Constrained and unconstrained testing environment*, ACM Trans. Math. Softw., 21 (1995), pp. 123–160.

[9] J. R. Bunch, *Partial pivoting strategies for symmetric matrices*, SIAM J. Numerical Analysis, 11 (1974), pp. 521–528.

[10] J. R. Bunch and L. Kaufman, *Some stable methods for calculating inertia and solving symmetric linear systems*, Mathematics of Computation, (1977), pp. 63–179.

[11] J. R. Bunch, L. Kaufman, and B. Parlett, *Decomposition of a symmetric matrix*, Numerische Mathematik, 27 (1976), pp. 95–109.

[12] R. Byrd, F. Curtis, and J. Nocedal, *An inexact newton method for nonconvex equality constrained optimization*, tech. rep., Optimization Technology Center, Northwestern University, Evanston, IL, 2007.

[13] R. H. Byrd, M. E. Hribar, and J. Nocedal, *An interior point algorithm for large-scale nonlinear programming*, SIAM J. Optimization, 9 (1999), pp. 877–900.

[14] A. Cline, C. B. Moler, G. Stewart, and J. Wilkinson, *An estimate for the condition number of a matrix*, SIAM J. Numerical Analysis, 16 (1979), pp. 368–375.

[15] P. Deuflhard, M. Weiser, and M. Seebass, *A new nonlinear elliptic multilevel FEM applied to regional hyperthermia*, Comput. Vis. Sci., 3 (2000), pp. 115–120.

[16] I. S. Duff and S. Pralet, *Strategies for scaling and pivoting for sparse symmetric indefinite problems*, SIAM J. Matrix Analysis and Applications, 27 (2005), pp. 313–340.

[17] A. Forsgren, *Inertia-controlling factorizations for optimization algorithms*, Applied Numerical Mathematics, 43 (2002), pp. 91–107.

[18] A. Forsgren, P. E. Gill, and M. H. Wright, *Interior methods for nonlinear optimization*, SIAM Review, 44 (2002), pp. 525–597.

[19] R. Fourer, D. M. Gay, and B. W. Kernighan, *AMPL: A Modeling Language For Mathematical Programming*, Thomson Publishing Company, Danvers, MA, USA, 1993.

[20] R. Freund and F. Jarre, *A QMR–based interior–point algorithm for solving linear programs*, Mathematical Programming, Series B, 76 (1997), pp. 183–210.

[21] R. W. Freund, *Preconditioning of symmetric, but highly indefinite linear systems*, in 15th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics, Vol. 2, Numerical Mathematics, Wissenschaft und Technik Verlag, 1997, pp. 551–556.

[22] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, third ed., 1996.

[23] N. I. M. Gould, *On modified factorizations for large-scale linearly constrained optimization*, SIAM J. Optimization, 9 (1999), pp. 1041–1063.

[24] N. I. M. Gould, Y. Hu, and J. A. Scott, *A numerical evaluation of sparse direct solvers for the solution of large sparse, symmetric linear systems of equations*, Tech. Rep. RAL-TR-2005-005, Rutherford Appleton Laboratory, 2005. to appear.

[25] N. I. M. Gould, D. Orban, and P. L. Toint, *Numerical methods for large-scale nonlinear optimization*, Acta Numerica, (2005), pp. 299–361.

[26] E. Haber and U. Ascher, *Preconditioned all-at-once methods for large, sparse parameter estimation problems*, Inverse Problems, 17 (2001), pp. 1847–1864.

[27] M. Hagemann and O. Schenk, *Weighted matchings for preconditioning symmetric indefinite linear systems*, SIAM J. Scientific Computing, 28 (2006), pp. 403–420.

[28] G. Karypis and V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM J. Scientific Computing, 20 (1998), pp. 359–392.

[29] N. Li, Y. Saad, and E. Chow, *Crout versions of ILU for general sparse matrices*, SIAM J. Scientific Computing, 25 (2004), pp. 716–728.

[30] H. Maurer and H. D. Mittelmann, *Optimization techniques for solving elliptic control problems with control and state constraints. part 1: Boundary control*, Comp. Optim. Applic., 16 (2000), pp. 29–55.

[31] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numerical Analysis, 12 (1975), pp. 617–629.

[32] Y. Saad and M. H. Schultz, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Scientific and Statistical Computing, 7 (1986), pp. 856–869.

[33] O. Schenk, M. Bollhöfer, and R. A. Römer, *On large scale diagonalization techniques for the Anderson model of localization*, SIAM J. Scientific Computing, 28 (2006), pp. 963–983.

[34] ———, *On large scale diagonalization techniques for the Anderson model of localization*, SIAM Review, (2008). accepted, in press.

[35] O. Schenk and K. Gärtner, *Solving unsymmetric sparse systems of linear equations with PARDISO*, Journal of Future Generation Computer Systems, 20 (2004), pp. 475–487.

[36] ———, *On fast factorization pivoting methods for symmetric indefinite systems*, Elec. Trans. Numer. Anal., 23 (2006), pp. 158–179.

[37] O. Schenk and K. Gärtner, *PARDISO and PARDISO/ML — parallel sparse direct and algebraic multi-level solver packages for sparse linear matrices*, 2007. University of Basel, `http://www.pardiso-project.org`.

[38] O. Schenk, A. Wächter, and M. Hagemann, *Matching-based preprocessing algorithms to the solution of saddle-point problems in large-scale nonconvex interior-point optimization*, Comput. Optim. Appl., 36 (2007), pp. 321–341.

[39] R. J. Vanderbei and D. F. Shanno, *An interior-point algorithm for nonconvex nonlinear programming*, Computational Optimization and Applications, 13 (1999), pp. 231–252.

[40] A. Wächter and L. T. Biegler, *On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming*, Mathematical Programming, 106 (2006), pp. 25–57.

[41] R. A. Waltz, J. L. Morales, J. Nocedal, and D. Orban, *An interior algorithm for nonlinear optimization that combines line search and trust region steps*, Mathematical Programming, Series A, 107 (2006), pp. 391–408.

Table 1: Characteristics of sample matrices from 3D PDE-constrained optimization problems. Key: n = number of Hessian equations; m = number of Jacobian equations; $nnz(A)$,$nnz(H)$ = number of nonzeros in A and H; *inertia* = inertia of the KKT matrix; fill-in = fill-in of the complete factor during a sparse direct factorization in % compared to the original matrix; anal. and fact. = times for analysis and factorization to compute inertia information with sparse direct solver (in CPU seconds) .

Problem	inertia	sparse LBL^T		
		fill-in	anal.	fact.
IPOPT series A, n = 2,331, m =1,331 $nnz(A)$ = 37,791, $nnz(H)$ = 17,892				
pde-a-01-01	(2059, 1603, 0)	18.5	0.03	0.38
pde-a-01-02	(2067, 1595, 0)	18.5	0.03	0.38
pde-a-01-03	(2331, 1331, 0)	18.5	0.04	0.38
pde-a-02-01	(2073, 1589, 0)	18.5	0.05	0.38
pde-a-02-02	(2331, 1331, 0)	18.5	0.02	0.36
IPOPT series B, n = 17,261, m = 9,261 $nnz(A)$=290,981, $nnz(H)$ = 135,382				
pde-b-01-01	(15618, 10904, 0)	60.1	0.24	17.1
pde-b-01-02	(15623, 10899, 0)	60.2	0.34	16.4
pde-b-01-03	(17261, 9261, 0)	59.9	0.25	16.6
pde-b-02-01	(15633, 10889, 0)	60.4	0.26	16.0
pde-b-02-02	(17261, 9261, 0)	59.2	0.27	16.7
IPOPT series C, n = 132,921, m = 68,921 $nnz(A)$ = 2,283,561, $nnz(H)$ = 1,053,162				
pde-c-01-01	(132921, 68921, 0)	212.	4.3	2644.
pde-c-02-01	(68921, 132921, 0)	214.	4.4	2532.
pde-c-02-02	(132921, 68921, 0)	215.	4.2	2612.
pde-c-03-01	(68921, 132921, 0)	216.	4.4	2622.
pde-c-03-02	(132921, 68921, 0)	215.	4.3	2634.

Table 2: Inertia-revealing KKT preconditioner applied to matrices from Table 1. Key: fill-in = fill-in of the preconditioner in % compared to the original matrix; inertia = is the computed inertia correct? set-up= time to compute the preconditioner (in CPU seconds).

Problem	ILDLT			ILDLT-MATCH			ILBLT-MATCH		
	fill-in	inertia	set-up	fill-in	inertia	set-up	fill-in	inertia	set-up
IPOPT series A, fill-in sparse direct solver: ≈ 18.5									
pde-a-01-01	4.91	wrong	0.33	9.92	wrong	1.30	3.15	ok	0.12
pde-a-01-02	5.12	wrong	0.34	7.15	ok	0.58	3.17	ok	0.07
pde-a-01-03	5.51	wrong	0.37	8.39	ok	0.50	3.00	ok	0.07
pde-a-02-01	6.52	wrong	0.38	7.83	ok	0.67	3.29	ok	0.08
pde-a-02-02	5.19	wrong	0.34	7.31	ok	0.69	3.16	ok	0.07
IPOPT series B, fill-in sparse direct solver: ≈ 60.2									
pde-b-01-01	6.03	wrong	13.4	17.7	ok	7.38	4.23	ok	0.96
pde-b-01-02	5.83	wrong	12.4	67.7	wrong	1044	4.34	ok	1.00
pde-b-01-03	3.87	wrong	7.41	16.3	ok	4.48	4.16	ok	0.96
pde-b-02-01	6.03	wrong	13.4	18.2	ok	6.96	4.29	ok	0.99
pde-b-02-02	6.05	wrong	13.4	61.5	wrong	1019	4.37	ok	1.01
IPOPT series C, fill-in sparse direct solver: ≈ 212.2									
pde-c-01-01	18.4	wrong	2901	112.	ok	5915	4.53	ok	9.47
pde-c-02-01	21.4	wrong	4130	35.1	wrong	2915	4.57	ok	9.86
pde-c-02-02	29.4	wrong	4530	45.1	ok	3215	4.52	ok	9.56
pde-c-03-01	12.4	wrong	1031	15.1	ok	1015	4.56	ok	9.73
pde-c-03-02	11.4	wrong	1041	25.1	ok	2115	4.53	ok	9.66

Table 3: Characteristics of publicly available indefinite saddle-point matrices. Key: n = number of Hessian equations; m = number of Jacobian equations; $nnz(A)$,$nnz(H)$ = number of nonzeros in A and H; H spd.? = is the Hessian $H(x)$ positive definite?; fill-in = fill-in of the complete factor during a sparse direct factorization in % to the original matrix; anal. and fact.= times for analysis and factorization to compute inertia information with sparse direct solver (in CPU seconds)
.

Problem	n	m	H spd.?	$nnz(A)$	$nnz(H)$	sparse LBL^T		
						fill-in	anal.	fact.
A0NSDSIL	60,012	20,004	no	115,005	100,020	3.49	0.11	0.08
A2NNSNSL	60,012	20,004	no	111,099	100,020	3.31	0.23	0.07
A5ESINDL	45,006	15,002	no	15,002	75,010	4.89	0.07	0.04
AUG3DCQP	27,543	8,000	yes	50,286	27,543	43.0	0.41	1.58
BLOWEYA	20,002	10,002	no	50,003	40,003	1.75	0.19	0.02
BRAINPC2	13,807	13,800	yes	82,794	13,807	3.82	6.25	0.27
BRATU3D	15,625	12,167	no	15,625	85,169	49.8	0.30	2.35
c-55	19,121	13,659	yes	185,335	19,121	49.5	1.02	8.36
c-58	22,461	15,134	yes	257,481	22,461	30.1	1.18	6.51
c-59	23,813	17,469	yes	219,627	23,813	43.5	1.19	9.66
c-62	25,158	16,573	yes	258,806	25,158	67.3	1.41	20.9
c-63	25,505	18,729	yes	195,235	25,505	26.4	1.08	4.26
c-71	44,814	31,824	yes	391,458	44,814	89.2	2.48	72.7
DIXMAANL	234,128	155,746	yes	466,319	701,366	10.8	3.27	1.53
NCVXQP1	7,111	5,000	no	14,998	25,539	76.2	0.25	2.13
NCVXQP7	50,000	37,500	no	112,497	199,984	70.0	14.6	116.

Table 4: Inertia-revealing preconditioner applied to matrices from Table 3. Key: fill-in = fill-in of the preconditioner in % to the original matrix; inertial = is the computed inertia correct? (number of incorrect sign in parentheses); set-up = time to compute the preconditioner (in CPU seconds). † indicates that the process was terminated due to a significantly higher computation time compared to the direct solver
.

Problem	ILDLT			ILDLT-MATCH			ILBLT-MATCH		
	fill-in	inertia	set-up	fill-in	inertia	set-up	fill-in	inertia	set-up
A0NSDSIL	0.94	yes	0.88	0.92	no (9)	0.52	2.28	yes	3.79
A2NNSNSL	0.85	no (2)	0.82	0.94	no (4)	0.51	2.22	yes	0.56
A5ESINDL	0.85	yes	0.40	0.90	no (3)	0.38	2.98	yes	0.36
AUG3DCQP	0.49	no (8000)	0.49	0.99	yes	0.31	2.88	yes	0.30
BLOWEYA	0.78	no (100)	1.04	2.54	yes	0.56	0.79	yes	0.18
BRAINPC2	0.99	no (89)	0.87	2.50	no (1762)	3.97	0.68	yes	1.70
BRATU3D	†	†	†	49.1	yes	9.66	2.04	yes	0.22
c-55	†	†	†	1.62	yes	0.87	1.24	yes	0.82
c-58	†	†	†	0.89	yes	1.40	1.07	yes	0.97
c-59	†	†	†	1.52	yes	0.97	1.96	yes	1.04
c-62	†	†	†	1.16	yes	1.61	1.65	yes	1.24
c-63	†	†	†	1.46	yes	0.90	1.45	yes	0.89
c-71	†	†	†	1.77	yes	2.01	1.80	yes	2.01
DIXMAANL	1.86	no (2)	0.47	1.88	no (5)	0.54	1.86	yes	0.53
NCVXQP1	2.94	no (2712)	0.96	1.21	yes	0.28	1.00	yes	0.28
NCVXQP7	†	†	†	10.0	no (2)	20.8	5.5	no (15)	11.55

Table 5: Inertia-revealing KKT preconditioner ILBLT-MATCH. Numbers of SQMR iterations for different value of the initial residual reduction for several KKT matrices.

	Relative residual		
Problem	10^{-4}	10^{-7}	10^{-14}
pde-a-01-03	12	20	25
pde-a-02-02	14	20	27
pde-b-01-03	14	18	24
pde-b-01-02	17	21	27
pde-c-01-01	17	23	26
pde-c-02-02	16	23	29
pde-c-03-02	18	20	26
AUG3DCQP	16	41	59
BRAINPC2	19	23	27
c-55	6	15	21
c-58	27	39	54
c-59	14	18	23
c-62	13	19	28
c-63	10	18	21
c-71	10	17	22
DIXMAANL	73	95	131

Table 6: Optimization outcome for CUTE test set. Key: "Direct" uses the regular Pardiso code; "Direct w/o inertia" uses the regular Pardiso code, but the inertia of the linear system was ignored and no modifications for the Hessian block to treat negative curvature were done; "Iterative" uses the iterative linear solver with the inertia-revealing preconditioner PARDISO/ML (for two residual reduction tolerances for the iterative solver). The possible outcomes are: Successful termination due to satisfaction of the tolerances for the optimizer; exceeding maximal iteration count (3000) or CPU time limit (30min); convergence to points that satisfy the optimizer's tolerances to determine local infeasibility; divergence of the iterates; other failures (including IPOPT's termination in the restoration phase)

Outcome	Direct	Direct w/o inertia	Iterative (res=10^{-7})	Iterative (res=10^{-14})
Success	688	627	674	667
Max iter	14	40	15	16
Time limit	1	0	11	14
Local infeas	2	7	1	5
Diverging	0	7	0	0
Other failure	15	39	19	18

Table 7: Optimization outcome for COPS test set. Column and row labeling as in Table 6.

Outcome	Direct	Direct w/o inertia	Iterative (res=10^{-7})	Iterative (res=10^{-14})
Success	64	49	64	64
Max iter	0	4	0	0
Local infeas	0	1	0	0
Diverging	0	3	0	0
Other failure	1	8	1	1

Table 8: Comparison of final objective function values for CUTE and COPS test problems. For a given pair of linear solver options the entries in the table list the number of problems that were successfully solved by both options and for which the final objective function values are more than 1% for the option names in the row compared to the option named in the column. The two numbers in each entry correspond to the number of CUTE and COPS problems, respectively.

	Direct	Direct w/o inertia	Iterative (res=10^{-7})	Iterative (res=10^{-14})
Direct	—	95/1	5/0	0/0
Direct w/o inertia	0/0	—	5/0	6/0
Iter (res=10^{-7})	2/0	90/1	—	1/0
Iter (res=10^{-14})	0/0	88/1	0/0	—

Table 9: Size of nonlinear programming problems for the 3D PDE-constrained optimization problem with boundary control as a function of the discretization parameter N using second-order finite difference approximations.

Problem Size	Variables with upper bounds	Variables with lower/upper bounds	Number of equality constraints	Number of nonzeros in Jacobian matrix
$N = 20$	8,000	2,400	8,000	215,296
$N = 40$	64,000	9,600	64,000	1,726,576
$N = 60$	216,000	21,600	216,000	5,829,856
$N = 80$	512,000	38,400	512,000	13,821,136
$N = 150$	3,510,000	135,400	3,510,000	91,119,616

Table 10: Timing results for the convex 3D PDE-constrained optimization problem (with objective (16)) as a function of the discretization parameter N. Key: MByte = Memory consumption in MByte; it. = total number of Newton iterations in IPOPT; secs = CPU time in seconds. The numerical experiments for the iterative solver PARDISO/ML are performed with a relative residual (res.) of either 10^{-4}, 10^{-7} or 10^{-14}. † indicates a 32-bit integer overflow for the direct solver PARDISO.

Problem	IPOPT — PARDISO			IPOPT — PARDISO/ML						
					res.= 10^{-4}		res.= 10^{-7}		res.=10^{-14}	
	MByte	it.	secs	MByte	it.	secs	it.	secs	it.	secs
$N = 20$	56	9	13	11	9	7	9	8	9	10
$N = 40$	952	9	604	96	9	78	9	87	9	118
$N = 60$	4,484	9	6473	336	9	311	9	362	9	554
$N = 80$	15,784	10	35,710	808	10	935	9	1,016	9	1,536
$N = 100$	†	†	†	1,600	10	2,121	9	2,324	9	3,789
$N = 150$	†	†	†	5,400	10	9,588	9	10,607	9	16,491

Table 11: Timing results for the nonconvex 3D PDE-constrained optimization problem (with objective (17)) as a function of the discretization parameter N. Key: MByte = Memory consumption in MByte; it. = total number of Newton iterations in IPOPT; secs = CPU time in seconds. The numerical experiments for the iterative solver PARDISO/ML are performed with a relative residual (res.) of either 10^{-4}, 10^{-7} or 10^{-14}. † indicates a 32-bit integer overflow for the direct solver PARDISO and ‡ a convergence problem in IPOPT.

Problem	IPOPT — PARDISO			IPOPT — PARDISO/ML						
					res.= 10^{-4}		res.= 10^{-7}		res.=10^{-14}	
	MByte	it.	secs	MByte	it.	secs	it.	secs	it.	secs
$N = 20$	56	68	272	12	44	70	49	94	67	136
$N = 40$	952	49	8,157	96	69	1,083	56	947	54	1,068
$N = 60$	4,484	58	104,837	384	67	4,029	67	4,501	66	5,701
$N = 80$	15,784	102	1,234,345	832	‡	‡	115	21,772	109	28,524
$N = 100$	†	†	†	1,603	‡	‡	381	197,323	345	258,369

Table 12: Scaling of NLP data with mesh size h for linear elements. n is the number of grid points, k the number of boundary points, "spd" = symmetric positive definite, "sp semidef" = symmetric positive semidefinite

quantity	dimension	scaling	structure
A_h	$m \times m$	h	spd
M_{ht}	$m \times m$	h^3	sp semidef
M_{hw}	$m \times m$	h^3	spd
E_h	$m \times 23 \times 23$	h^3	
D_h	$m \times m$	h^2	sp semidef

Table 13: Characteristic of the hyperthermia treatment planning problem. Key: n = number of Hessian equations; m = number of Jacobian equations; $nnz(A)$,$nnz(H)$ = number of nonzeros in A and $H(x)$.

Discretization					
FE order	grid refinements	n	m	$nnz(A)$	$nnz(H)$
1	0	14,357	14,334	529,084	107,144
2	0	106,891	106,868	5,398,662	1,524,059
1	1	106,891	106,868	3,999,824	824,640
2	1	824,387	824,364	42,139,056	12,001,800

Table 14: Memory consumption in MByte, total number of Newton iterations in Ipopt, and CPU in seconds for the linear solvers in the nonconvex hyperthermia 3D PDE-constrained optimization problem. The numerical experiments for the iterative solver Pardiso/Ml are performed with a relative residual reduction (res.) of either 10^{-7} or 10^{-14}. † indicates a 32-bit integer overflow for the direct solver Pardiso.

Discretization		Ipopt — Pardiso			Ipopt — Pardiso/Ml				
FE	grid					res.= 10^{-7}		res.=10^{-14}	
order	refinements	MByte	it.	secs	MByte	it.	secs	it.	secs
1	0	72	93	580	32	90	469	87	476
2	0	1,352	118	35,379	288	165	44,461	194	50,836
1	1	1,120	83	20,151	392	374	34,251	223	28,118
2	1	†	†	†	7,005	—	—	113	482,786