Scalable Preconditioning of Block-Structured Linear Algebra Systems using ADMM

Jose S. Rodriguez‡, Carl D. Laird†‡, and Victor M. Zavala¶∗

‡Davidson School Of Chemical Engineering
Purdue University, 610 Purdue Mall, West Lafayette, IN 47907
†Center for Computing Research
Sandia National Laboratories, Albuquerque, NM 87185
¶Department of Chemical and Biological Engineering
University of Wisconsin-Madison, 1415 Engineering Dr, Madison, WI 53706, USA

Abstract

We study the solution of block-structured linear algebra systems arising in optimization by using iterative solution techniques. These systems are the core computational bottleneck of many problems of interest such as parameter estimation, optimal control, network optimization, and stochastic programming. Our approach uses a Krylov solver (GMRES) that is preconditioned with an alternating method of multipliers (ADMM). We show that this ADMM-GMRES approach overcomes well-known scalability issues of Schur complement decomposition in problems that exhibit a high degree of coupling. The effectiveness of the approach is demonstrated using linear systems that arise in stochastic optimal power flow problems and that contain up to 2 million total variables and 4,000 coupling variables. We find that ADMM-GMRES is nearly an order of magnitude faster than Schur complement decomposition. Moreover, we demonstrate that the approach is robust to the selection of the augmented Lagrangian penalty parameter, which is a key advantage over the direct use of ADMM.

Keywords: Schur complement decomposition; ADMM; iterative; linear algebra; large-scale

1 Introduction

The scalability of optimization solvers relies quite heavily on the solution of the underlying linear algebra systems. Advances in direct sparse linear algebra solvers have been instrumental in the widespread use of quadratic programming and nonlinear programming solvers such as Ipopt, OOQP, and Knitro [1, 2, 27]. Specialized direct solution techniques have also been developed to tackle large-scale and block-structured systems (using variants of Schur complement decomposition techniques) [10, 20, 21, 38, 39]. Block structures appear in many important applications such as parameter estimation, stochastic programming, network optimization, and optimal control. Schur

*Corresponding Author: victor.zavala@wisc.edu
decomposition techniques can also leverage parallel computing architectures and have enabled the solution of problems with millions to billions of variables and constraints. Unfortunately, many applications of interest still remain inaccessible due to fundamental scalability limitations of Schur complement techniques. Specifically, Schur complement decomposition does not scale well in problems that exhibit high degrees of block coupling. This is because high degrees of coupling require assembling and factorizing large Schur complement matrices (which are often highly dense).

Iterative solution techniques [5, 6, 12, 13, 31, 33] and associated preconditioning strategies [8, 17, 19, 32, 35, 44] have been proposed to address fundamental scalability issues of direct linear algebra strategies. In the context of block-structured problems, attempts have been made to solve the Schur complement system by using iterative solution techniques (to avoid assembling and factorizing the Schur complement). Preconditioners for Schur complements arising in special problem classes such as multi-commodity networks and stochastic programs have been developed [8]. Unfortunately, preconditioning strategies for general problem classes are still lacking. Another important issue that arises in this context is that the implementation of advanced linear algebra strategies is non-trivial (e.g., it requires intrusive modifications of optimization solvers).

Along a separate line of research, significant advances have been made in the development of problem-level decomposition techniques such as the alternating direction method of multipliers (ADMM) and Lagrangian dual decomposition [18, 22, 23, 25, 26, 34]. Such approaches are flexible and rather easy to implement but suffer from slow convergence. Recently, it has been proposed to use ADMM as a preconditioner for Krylov-based iterative solvers such as GMRES [41, 42]. In this work, we provide a detailed derivation of this ADMM-GMRES approach and test its performance in the context of block-structured linear algebra systems. We demonstrate that this approach overcomes the scalability issues of Schur complement decomposition. We also demonstrate that this approach is significantly more effective than using ADMM directly. Our tests are facilitated by the use of PyNumero, a recently-developed Python framework that enables the implementation and benchmarking of optimization algorithms. We use the proposed framework to tackle problems with hundreds of thousands to millions of variables generated from standard benchmark sets and power grid applications.

In their original work, Zhang and White [41, 42] present an excellent discussion on the use of ADMM as a GMRES preconditioner for solving one-block problems. Within their analysis they compare ADMM-GMRES with popular preconditioning techniques and with standard ADMM. Their work, however, does not discuss the scalability of the approach on large-scale problems and does not study the performance of ADMM-GMRES on problems with decomposable block-structure. We demonstrate the benefits of using ADMM-GMRES on problems with decomposable primal space and we focus on analyzing behavior in problems in which the number of coupling variables is large (i.e., stochastic problems with large dimensionality in the first stage). Our analysis also focuses on demonstrating that ADMM-GMRES overcomes limitations of Schur-complement decomposition.

The paper is structured as follows. In Section 2 we define the problem of interest and provide preliminary information on the use of Schur complement decomposition and ADMM approaches. In Section 3 we provide a detailed derivation of the ADMM-GMRES approach and in Section 4.1 we provide benchmark results.
2 Preliminaries

We study the solution of block-structured quadratic programs (QP) of the form:

\[
\begin{align*}
\min_{x_i, z} & \quad \sum_{i \in \mathcal{P}} \frac{1}{2} x_i^T D_i x_i + c_i^T x_i \\
\text{s.t.} & \quad J_i x_i = b_i, \quad (\lambda_i) \quad i \in \mathcal{P} \\
& \quad A_i x_i + B_i z = 0, \quad (y_i) \quad i \in \mathcal{P}.
\end{align*}
\]

(1a)

(1b)

(1c)

Here, \( \mathcal{P} := \{1, \ldots, P\} \) is a set of block variable partitions. Each partition contains a vector of primal variables \( x_i \in \mathbb{R}^{n_x} \) and the vector \( z \in \mathbb{R}^{n_z} \) contains the primal variables that the couple partitions. The total number of primal variables is \( n := n_z + \sum_{i \in \mathcal{P}} n_x \). Equation (1b) are the partition constraints with their respective dual variables \( \lambda_i \in \mathbb{R}^{m_i} \). Equation (1c) are the constraints that link partitions across set \( \mathcal{P} \) and have associated dual variables \( y_i \in \mathbb{R}^{l_i} \). We assume that the partition matrices \( J_i \in \mathbb{R}^{m_i \times n_x} \) have full row rank and that the right-hand-side coefficients \( b_i \in \mathbb{R}^{m_i} \) are in the column space of \( J_i \). The total number of partition constraints is \( m := \sum_{i \in \mathcal{P}} m_i \). We refer to \( A_i \in \mathbb{R}^{n_z \times n_x} \) and \( B_i \in \mathbb{R}^{n_z \times n_z} \) as linking matrices and we assume them to have full rank. The total number of linking constraints is \( l := \sum_{i \in \mathcal{P}} l_i \). The QP under study is the main computational kernel behind nonlinear programming strategies because it is used for computing primal-dual search steps.

We make the blanket assumption that the block-structured QP is strongly convex and that the combined Jacobian matrix (obtained by assembling partition and coupling constraints) has full row rank. Strong convexity can be obtained by ensuring that all block Hessian matrices \( D_i \) are positive definite. Strong convexity and full-rank conditions guarantee that the primal-dual solution of the QP exists and is unique. Moreover, these assumptions guarantee that the QP solution is a unique minimizer and that this can be found by solving the first-order stationarity conditions. Additional assumptions will also be needed on the nature of the building blocks of the QP (associated with each partition). Such assumptions are needed to ensure that proposed decomposition schemes are well-defined and will be stated as we proceed (in order to maintain clarity in the presentation).

The Lagrange function of (1) can be expressed as:

\[
\mathcal{L}(x, z, \lambda, y) = \sum_{i \in \mathcal{P}} \frac{1}{2} x_i^T D_i x_i + c_i^T x_i + y_i^T (A_i x_i + B_i z) + \lambda_i^T (J_i x_i - b_i),
\]

(2)

where \( x := (x_1, \ldots, x_P) \), \( \lambda := (\lambda_1, \ldots, \lambda_P) \) and \( y := (y_1, \ldots, y_P) \). The first-order optimality conditions of (1) are given by:

\[
\begin{align*}
\nabla_x \mathcal{L} &= 0 = D_i x_i + c_i + J_i^T \lambda_i + A_i^T y_i, \quad i \in \mathcal{P} \\
\nabla_\lambda \mathcal{L} &= 0 = J_i x_i - b_i, \quad i \in \mathcal{P} \\
\nabla_y \mathcal{L} &= 0 = A_i x_i - B_i z, \quad i \in \mathcal{P} \\
\nabla_z \mathcal{L} &= 0 = \sum_{i \in \mathcal{P}} B_i^T y_i.
\end{align*}
\]

(3)

These conditions form a block-structured linear system of the form shown in (6). For the sake of
compactness and ease of notation, we rewrite (6) as:

\[
\begin{bmatrix}
K & A^T
\end{bmatrix}
\begin{bmatrix}
A & B
\end{bmatrix}
\begin{bmatrix}
v \\
y
\end{bmatrix}
= \begin{bmatrix}
\gamma \\
0
\end{bmatrix}.
\]

(4)

where \( v = (v_1, \ldots, v_P) \), \( \nu = (x_i, \lambda_i) \), \( \gamma = (\gamma_1, \ldots, \gamma_P) \), \( \gamma_i = (-c_i, b_i) \), \( u = (v, z, y) \), \( r = (\gamma, 0, 0) \), and

\( K = \text{blkdiag}\{K_1, K_2, \ldots, K_P\} \). We also have \( A = \text{blkdiag}\{\tilde{A}_1, \tilde{A}_2, \ldots, \tilde{A}_P\} \), \( B = \text{rowstack}\{B_1, B_2, \ldots, B_P\} \)

with:

\[
K_i = \begin{bmatrix} D_i & J_i^T \\ J_i & A_i \end{bmatrix}, \quad \tilde{A}_i = \begin{bmatrix} A_i & 0 \end{bmatrix}, \quad i \in \mathcal{P}.
\]

(5)

\[
\begin{bmatrix}
D_1 & J_1^T & A_1^T \\
J_1 & \ddots & \ddots \\
& & \ddots & A_P^T \\
A_1 & & \ddots & \ddots \\
& \ddots & & \ddots & A_P \\
& & \ddots & & B_P \\
& & & \ddots & \ddots \\
& & & & \ddots & \ddots \\
& & & & & \ddots & \ddots \\
& & & & & & \ddots & \ddots \\
B_1^T & \cdots & B_P^T \\
\end{bmatrix}
\begin{bmatrix}
x_1 \\
\lambda_1 \\
x_P \\
\lambda_P \\
y_1 \\
y_P \\
z
\end{bmatrix}
= \begin{bmatrix}
-c_1 \\
b_1 \\
0 \\
b_P \\
0 \\
0 \\
0
\end{bmatrix}.
\]

(6)

2.1 Solution using Schur Decomposition

One can solve large instances of the block-structured QP by using a Schur-complement decomposition method (we refer to this approach simply as Schur decomposition) [40]. This approach decomposes (1) by using block Gaussian elimination on a permuted version of the linear system (6). The permuted system has the structure:

\[
\begin{bmatrix}
D_1 & J_1^T & A_1^T \\
J_1 & \ddots & \ddots \\
& & \ddots & A_P^T \\
A_1 & & \ddots & \ddots \\
& \ddots & & \ddots & A_P \\
& & \ddots & & B_P \\
& & & \ddots & \cdots & B_P^T \\
& & & \ddots & \ddots & \cdots & \ddots \\
& & & \ddots & \ddots & \ddots & \ddots \\
& & & \ddots & \ddots & \ddots & \ddots & \ddots \\
& & & \ddots & \ddots & \ddots & \ddots & \ddots \\
& & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
& & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
& & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
& & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
B_1^T & \cdots & B_P^T & \end{bmatrix}
\begin{bmatrix}
x_1 \\
\lambda_1 \\
x_P \\
\lambda_P \\
y_1 \\
y_P \\
z
\end{bmatrix}
= \begin{bmatrix}
-c_1 \\
b_1 \\
0 \\
b_P \\
0 \\
0 \\
0
\end{bmatrix}.
\]

(7)

This system can be expressed in compact form as:

\[
\begin{bmatrix}
K & B_s \\
B_s^T & z
\end{bmatrix}
\begin{bmatrix}
v_s \\
z
\end{bmatrix}
= \begin{bmatrix}
\gamma_s \\
0
\end{bmatrix}.
\]

(8)
where \( u_s = (u_{s_1}, \cdots, u_{s_P}) \), \( u_{s_i} = (x_i, \lambda_i, y_i) \), \( \gamma_s = (\gamma_{s_1}, \cdots, \gamma_{s_P}) \), and \( \gamma_i = (-c_i, b_i, 0) \). We also have \( K_s = \text{blkdiag}\{K_{s_1}, K_{s_2}, \cdots, K_{s_P}\} \) and \( B_s = \text{rowstack}\{B_{s_1}, B_{s_2}, \cdots, B_{s_P}\} \) with:

\[
K_{s_i} = \begin{bmatrix} D_i & J_i^T & A^T \\ J_i & A \end{bmatrix}, \quad B_{s_i} = \begin{bmatrix} 0 & 0 & B_i^T \end{bmatrix}^T, \quad i \in \mathcal{P}.
\]

(9)

Because we have assumed that \( D_i \) is positive definite and the combined constraint Jacobian \((J_i, A)\) has full row rank, we have that \( K_{s_i} \) is nonsingular. This also implies \( K_s \) is nonsingular and, as a result, we can form the Schur complement system:

\[
(B_s^T K_s^{-1} B_s) z = B_s^T K_s^{-1} \gamma_s.
\]

(10)

We refer to the coefficient matrix of (10) as the Schur complement. Since \( K_s \) is block-diagonal, one can assemble the Schur complement by factorizing the blocks \( K_{s_i} \) independently. By using this assembled Schur complement, one can then factorize the Schur complement matrix and solve system (10) to find a solution for the coupling variables \( z \). Having \( z \), one then proceeds to find solutions for the partition variables \( u_s \) by solving the following system:

\[
K_s u_s = \gamma_s - B_s z.
\]

(11)

Here, again, one can solve for each element \( u_{s_i} \) independently because \( K_s \) is block-diagonal. The Schur decomposition method is summarized in Algorithm 1. We refer the reader to [29] for details on the implementation of Schur decomposition approaches.

---

**Algorithm 1: Schur Decomposition for Block-Structured QP**

1. Let \( S = 0 \) and \( r_{sc} = 0 \)
2. Factorize \( K_s \) matrix:
3. 
   \hspace{1em} foreach \( i \in \mathcal{P} \) do
4. 
   \hspace{2em} Factorize \( K_{s_i} \)
5. Form Schur complement system:
6. 
   \hspace{1em} foreach \( i \in \mathcal{P} \) do
7. 
   \hspace{2em} \( S = S + B_i^T K_{s_i}^{-1} B_i \)
8. 
   \hspace{2em} \( r_{sc} = r_{sc} + B_i^T K_{s_i}^{-1} \gamma_{s_i} \)
9. Factorize \( S \) and compute coupling variables by solving:
10. \( S z = r_{sc} \)
11. Compute partition variables:
12. 
   \hspace{1em} foreach \( i \in \mathcal{P} \) do
13. 
   \hspace{2em} \( K_{s_i} u_{s_i} = \gamma_{s_i} - B_i z \)
Schur decomposition is a flexible approach that enables the solution of problems with many block partitions. A fundamental limitation of this approach, however, is that one needs to assemble and factorize the Schur complement matrix $S$ (which is often highly dense). As a result, Schur decomposition does not scale well with the number of coupling variables $z$. Iterative approaches can in principle be used to solve the Schur system but effective preconditioning strategies do not currently exist for general block-structured systems.

### 2.2 Solution using ADMM

The block-structured QP can also be decomposed and solved by using ADMM. This approach seeks to minimize the augmented Lagrangian function:

$$
\mathcal{L}_\rho(x, z, \lambda, y) = \sum_{i \in \mathcal{P}} x_i^T D_i x_i + c_i^T x_i + (A_i x_i + B_i z)^T y_i + \lambda_i^T (J_i x_i - b_i) + \frac{\rho}{2} \|A_i x_i + B_i z\|^2
$$

by performing alternating minimization with respect to the block variables $(x_i, y_i)$ and the coupling variables $z$. A standard implementation of ADMM for solving the QP of interest is presented in Algorithm 2.

#### Algorithm 2: ADMM for Block-Structured QP

**Input**: Starting point $u^0 = (\nu^0, y^0, z^0)$, maximum number of iterations $N_{\text{ADMM}}$, penalty parameter $\rho > 0$, and convergence tolerance $\epsilon > 0$

1. for $k = 0, 1, 2, \ldots, N$ do
2.   Update partition variables:
3.     foreach $i \in \mathcal{P}$ do
4.         $x_i^{k+1} = \arg \min_{x \in X_i} x_i^T D_i x_i + c_i^T x_i + (A_i x_i + B_i z)^T y_i + \frac{\rho}{2} \|A_i x_i + B_i z\|^2$
5.     Update coupling variables:
6.         $z^{k+1} = \arg \min_{z} \sum_{i \in \mathcal{P}} (A_i x_i^{k+1} + B_i z)^T y_i + \frac{\rho}{2} \|A_i x_i^{k+1} + B_i z\|^2$
7.     Update dual variables:
8.     foreach $i \in \mathcal{P}$ do
9.         $y_i^{k+1} = y_i^k + \rho (A_i x_i^{k+1} + B_i z^{k+1})$
10. if $\|y_i^{k+1} - y_i^k\| \leq \epsilon$ and $\|\rho^T B \cdot (z^{k+1} - z^k)\| \leq \epsilon$ then
11.     stop
12. Output: $u$

In the above algorithm, $X_i = \{x \mid J_i x - b_i = 0\}$ is used to denote the feasible set of each partition (the inner block constraints are satisfied exactly). The ADMM algorithm can be implemented by solving the first-order conditions of each subproblem directly. This is because each block subproblem is strongly convex and the block Jacobian has full row rank. This approach is sketched in Algorithm 3.
Algorithm 3: ADMM\((u^0, N, \rho)\)

**Input:** starting point \(u^0 = (\upsilon^0, y^0, z^0)\), maximum number of iterations \(N_{ADMM}\), penalty parameter \(\rho > 0\), and convergence tolerance \(\epsilon > 0\)

1. Factorize \(K\rho\) and \(B^TB\) matrix:
   
   \[
   \text{foreach } i \in \mathcal{P} \text{ do}
   \]
   
   2. Factorize partition matrices \(K\rho_i\) and \(B_i^TB_i\)
   
   3. for \(k = 0, 1, 2, \ldots, N\) do
   
   4. Update partition variables:
   
      \[
      K_{\rho_i} \upsilon^{k+1} = -\left( \gamma_i + \begin{bmatrix} \rho A_i^T B_i z^k & 0 \\ 0 & 0 \end{bmatrix} \right)
      \]
   
   5. Update coupling variables:
   
      \[
      z^{k+1} = -[B^TB]^{-1} \left( B^T A_i \upsilon^{k+1} + \frac{1}{\rho} B^T y^k \right)
      \]
   
   6. Update dual variables:
   
      \[
      \text{foreach } i \in \mathcal{P} \text{ do}
      \]
   
      \[
      y_i^{k+1} = y_i^k + \rho \left( A_i \upsilon_i^{k+1} + B_i z_i^{k+1} \right)
      \]
   
      \[
      \text{if } \|y_i^{k+1} - y_i^k\| \leq \epsilon \text{ and } \|\rho A_i B_i \left( z_i^{k+1} - z_i^k \right)\| \leq \epsilon \text{ then}
      \]
   
      \[
      \text{stop}
      \]

Output: \(u\)

In the above algorithm we have that \(K\rho = \text{blkdiag}\{K\rho_1, K\rho_2, \ldots, K\rho_P\}\) with

\[
K_{\rho_i} = \begin{bmatrix} D_i + \rho A_i^T A_i & J_i^T \\ J_i & 0 \end{bmatrix}.
\]

We note that the update of the coupling variables still requires forming and factorizing the matrix \(B^TB\). This operation only needs to be performed once and can be computed efficiently by exploiting the sparsity of \(B_i\). We first note that \(B^TB = \sum_{i \in \mathcal{P}} B_i^TB_i\) and block-structured problems typically have sparse \(B_i\) matrices of zeros and ones. In stochastic programming problems, for example, the \(B_i\) matrices are identity matrices of dimension \(n_z \times n_z\) resulting in \([B^TB]^{-1} = [\sum_{i \in \mathcal{P}} I]^{-1} = \frac{1}{n} I\). Thus, the update of coupling variables follows a simple averaging operator. For other structured \(B\) matrices, like those that arise in temporal decomposition, the calculation of \([B^TB]^{-1}\) also exploits the sparsity pattern on \(B_i\) (to compute specialized averaging operators over coupled blocks). We refer the reader to [34] for further details. As a result, the update step for the coupling variables in ADMM is far cheaper than that of Schur decomposition and can thus overcome the main computational bottleneck of the latter. Unfortunately, it is well-known that ADMM exhibits slow convergence and thus the ability to perform fast operations might be shadowed by the need to perform many iterations.

### 3 Solution using ADMM-GMRES

The key observation that motivates our work is that ADMM can be used as a *preconditioner* for iterative linear algebra techniques such as GMRES [41, 42]. To derive the ADMM preconditioning...
strategy, we consider the regularized QP (1):

$$\begin{align*}
\min_{x, z} & \quad \sum_{i \in P} \frac{1}{2} x_i^T D_i x_i + c_i^T x_i + \frac{\rho}{2} \| A x_i + B_i z \|^2 \\
\text{s.t.} & \quad J_i x_i = b_i, \quad (\lambda_i) \quad i \in P \\
& \quad A_i x_i + B_i z = 0, \quad (y_i) \quad i \in P.
\end{align*}$$

(14a)

The solution of this problem is also a solution of (1) (since the penalization term vanishes at the solution). The optimality conditions of the regularized QP are given by:

$$\begin{bmatrix}
K_{\rho} & \rho A^T B & A^T \\
\rho B^T A & \rho B^T B & B^T \\
A & B & \end{bmatrix}
\begin{bmatrix}
u \\
z \\
y
\end{bmatrix}
= \begin{bmatrix}
\gamma \\
0 \\
0
\end{bmatrix}.$$  

(15)

We refer to (15) as the KKT system and to $H_\rho$ as the KKT matrix. ADMM can be interpreted as a Gauss-Seidel (alternating) minimization of the block and coupling variables and the dual variables $[7, 11, 34]$. This induces a splitting operator $H_\rho = M_\rho - N_\rho$ satisfying:

$$\begin{bmatrix}
K_{\rho} & \rho A^T B & A^T \\
\rho B^T A & \rho B^T B & B^T \\
A & B & \end{bmatrix}
\begin{bmatrix}
u \\
z \\
y
\end{bmatrix}
= \begin{bmatrix}
\rho A^T B & -A^T \\
\rho B^T A & -B^T \\
A & B & -\frac{1}{\rho} I
\end{bmatrix}
\begin{bmatrix}
u \\
z \\
y
\end{bmatrix}.$$  

(16)

Applying splitting (16) to (15) gives the operator:

$$T_\rho(u) := G_\rho u + f_\rho$$

(17)

where $G_\rho = M_\rho^{-1} N_\rho$ and $f_\rho = M_\rho^{-1} r$. Note that any $u$ satisfying the fixed-point $T_\rho(u) = u$ also satisfies $(I - G_\rho) u = f_\rho$ and is a solution of the preconditioned KKT system:

$$M_\rho^{-1} H_\rho u = M_\rho^{-1} r.$$  

(18)

This follows from $M_\rho^{-1} H_\rho = M_\rho^{-1} (M_\rho - N_\rho) = I - G_\rho$. This motivates the development of a Richardson recursion of the form $u^{k+1} = G_\rho u^k + f_\rho$, which converges to a $u$ satisfying $(I - G_\rho) u = f_\rho$ and $M_\rho^{-1} H_\rho u = M_\rho^{-1} r$ (provided that the eigenvalues of $G_\rho$ are inside the unit circle). In Appendix A we show that the operator $T_\rho(u)$ can be computed by performing one ADMM iteration (using $u$ as starting point). In other words, we have that $T_\rho(u) = \text{ADMM}(u, N=1, \rho)$. This also implies that the Richardson recursion can be written as $u^{k+1} = \text{ADMM}(u^k, N=1, \rho)$. Consequently, the Richardson recursion (and thus ADMM) are consistent preconditioner choices.

The key idea behind ADMM-GMRES is to solve the system $M_\rho^{-1} H_\rho u = M_\rho^{-1} r$ by using the Krylov solver GMRES. This is equivalent to solving $(I - G_\rho) u = f_\rho$. The right-hand side of this system can be computed as $f_\rho = T_\rho(0)$. GMRES requires the computation of matrix-vector products with the
preconditioned coefficient matrix of the form $M^{-1}_\rho H\rho h = (I - G_\rho)h$. This can be done by using the operator (17) as:

$$M^{-1}_\rho H\rho h = h - [T_\rho(h) - T_\rho(0)], \quad (19)$$

This follows from the observation that:

$$M^{-1}_\rho H\rho h = h - [T_\rho(h) - T_\rho(0)] = h - [G_\rho h + f_\rho - f_\rho] = h - G_\rho h = (I - G_\rho)h. \quad (20)$$

From (20) we note that asking the ADMM oracle $\text{ADMM}(h, N, \rho)$ to iterate until reaching convergence will deliver $T_\rho(h) = h$ satisfying $M^{-1}_\rho H\rho h = f_\rho$. In such a case, the ADMM preconditioner is perfect (since it solves the actual preconditioned KKT system). Consequently, the quality of the ADMM preconditioner will improve as one increases $N$. For details on the properties of the preconditioner, we refer the reader to [41, 42]. The ADMM-GMRES strategy is summarized in Algorithm 4.

**Algorithm 4: ADMM-GMRES($N_{\text{GMRES}}, N_{\text{ADMM}}, \rho$)**

**Input:** maximum number of GMRES iterations $N_{\text{GMRES}}$, maximum number of ADMM iterations $N_{\text{ADMM}}$, penalty parameter $\rho > 0$, and tolerance $\epsilon > 0$

1. Compute right-hand-side vector:
   $$f_\rho = \text{ADMM}(0, N_{\text{ADMM}}, \rho)$$
2. Call GMRES solver:
   $$u = \text{GMRES}(I - G_\rho, f_\rho, N_{\text{GMRES}}, \epsilon)$$

**Output:** $u$

*Matrix-vector products are computed as $(I - G_\rho)h = h - (T_\rho(h) - f_\rho)$, where $T_\rho(h) = \text{ADMM}(h, N_{\text{ADMM}} = 1, \rho)$.

### 4 Numerical Results

In this section we discuss the implementation of ADMM-GMRES and present results for different benchmark problems. All numerical experiments were performed using PyNumero, which is an open-source framework written in Python and C++ that combines modeling capabilities of the algebraic modeling language Pyomo [24] with efficient libraries like the AMPL solver library [14], the Harwell Subroutine Library (HSL), and NumPy/SciPy [28]. It uses object-oriented principles that facilitate the implementation of algorithms and problem formulations that exploit block-structures via polymorphism and inheritance. All these features facilitate the implementation of ADMM, Schur decomposition, and ADMM-GMRES. The optimization models were implemented in Pyomo/PyNumero and all linear algebra operations were performed in compiled code. Within PyNumero, we used an MA27 interface to perform all direct linear algebra operations. We use the GMRES implementation available in Scipy to perform all iterative linear algebra operations. Iterative linear algebra routines available in KRYPY [15] were also used to validate results. To implement the power grid models we
used [1], a Pyomo-based package that facilitates the formulation if optimization problems that arise in power systems. The convergence criterion for GMRES and ADMM requires that the norm of the KKT system residual $Hu - r$ is smaller than $\epsilon = 1 \cdot 10^{-8}$. If the convergence criterion is not satisfied after 2,000 iterations, the algorithm was aborted and we report the final residual achieved. The linear solver MA27 was used with a pivoting tolerance of $1 \cdot 10^{-8}$. Readers may contact the corresponding author for getting access to the code used to produce the results in this section.

4.1 Standard Benchmark Problems

We first conducted tests with randomly generated instances to study qualitatively the performance of ADMM-GMRES on block-structured optimization problems. This section focuses on two-stage stochastic programs of the form:

$$
\begin{align*}
\min_{x_i, z} & \sum_{i \in \mathcal{P}} \frac{1}{2} x_i^T D x_i + e^T x_i \\
\text{s.t.} & \quad J x_i = b_i, \quad (\lambda_i) \quad i \in \mathcal{P} \\
& \quad A_i x_i - z = 0, \quad (y_i) \quad i \in \mathcal{P}
\end{align*}
$$

(21)

where $\mathcal{P}$ is the scenario set, $x_i$ are the second-stage (recourse) variables, and $z$ are first-stage (coupling) variables. We defined a nominal vector $b$ and create scenarios with right-hand-side vector $b_i$ using the nominal vector $b$ as the mean and a standard deviation $\sigma=0.5b$. We first demonstrate the scalability of Algorithm 4 when solving instances of problem (21) with high dimensionality in the coupling variables $z$. The stochastic problem was constructed in the following manner: $D$ was set to a $4,800 \times 4,800$ block diagonal matrix with 16 dense symmetric blocks. Each block was generated following Algorithm 14 from [42] with log-standard-deviation $s=0.5$ (see [42] for details). The random matrix $J$ has dimensions of $100 \times 4,800$. The number of scenarios was set to 50, giving an initial problem with 240,000 variables and 5,000 constraints. To investigate the scalability of Algorithm 4, the number of complicating variables was varied from 100 to 4,000. Note that, as $n_z$ increases, the number of constraints of (21) increases as $50n_z$. The largest problem solved contained 244,000 total variables and 205,000 total constraints.

We solved (21) using four different strategies. Figure 1 summarizes the results obtained with Schur decomposition, GMRES (without preconditioner), ADMM, and ADMM-GMRES. These results confirm the observations of Section 2. Specifically, Schur decomposition does not scale well as $n_z$ increases. The main reason for this behavior is that, as $n_z$ increases, the number of operations required to form the Schur-complement increase. In addition, because the Schur-complement matrix is a dense $n_z \times n_z$ matrix, the factorization time increases cubically as $n_z$ increases. We observe that GMRES takes the longest time to solve the problem. For ADMM and ADMM-GMRES we see more favorable scalability as $n_z$ increases. Specifically, we see that ADMM-GMRES converges faster and that the savings increase as the number of coupling variables increases. We also note that ADMM-GMRES mimics the trend in performance of ADMM but is significantly faster.

Figure 1: Scalability analysis of Schur decomposition, GMRES (without preconditioner), ADMM, and ADMM-GMRES.

Figure 2: Evolution of residuals for GMRES (without preconditioner), ADMM, and ADMM-GMRES.

Figure 2 shows the residuals for the iterative approaches for a problem with 1,000 complicating variables. We can see that all methods exhibit linear convergence but that ADMM-GMRES outper-
forms both ADMM and GMRES (unpreconditioned). Notably, ADMM-GMRES converges in just 35 iterations while ADMM and GMRES require over 300 iterations and 1,000 iterations, respectively. Moreover, we note that ADMM-GMRES can reach high accuracy levels (of $1 \cdot 10^{-8}$), which is a desirable feature of iterative solution strategies.

An important drawback of ADMM is the need to tune the penalty parameter $\rho$. The work in [16] shows that an optimal value for $\rho$ can be chosen based on the smallest and largest eigenvalues of the matrix $A^TD^{-1}A$. In principle, this selection of $\rho$ is optimal for ADMM-GMRES as well. However, for large-scale structured problems such as the ones considered here, computing the eigenvalues of $A^TD^{-1}A$ is expensive. Heuristic approaches have also been proposed to select $\rho$ at every ADMM iteration with the objective of accelerating convergence [37]. Unfortunately these heuristics do not provide guarantees and might incur additional overhead. In particular, for the QP problems considered here, varying $\rho$ at every ADMM iteration will require forming and factorizing the $K_{\rho}$ repetitively. Interestingly, we now proceed to demonstrate that ADMM-GMRES is fairly insensitive to the choice of $\rho$.

Figure 3 compares the performance of ADMM against that of ADMM-GMRES for a stochastic program with $n_z=1,000$ and different values of $\rho$ ranging from $10^{-3}$ to $10^3$. Here we measure performance in terms of the number of iterations. Our results on the block-structured problem are in agreement with those in [41], where a single block problem is solved. We see that ADMM-GMRES is remarkably robust to the choice of $\rho$ (the number of iterations remain below 30). ADMM, on the other hand, fails to converge within 2,000 iterations for small and large values of $\rho$. The superior performance of ADMM-GMRES is attributed to the fact that the selection of $\rho$ only has an effect on the preconditioner and not on the convergence properties of GMRES. Nevertheless, we do observe that an optimal selection of $\rho$ improves performance of both ADMM and ADMM-GMRES. The robustness of ADMM-GMRES is a desirable feature when using the solver within more advanced SQP-based solvers. In particular, recent developments of augmented lagrangian interior-point approaches
[9, 30] provide promising frameworks for ADMM-GMRES because the selection of $\rho$ is made based on information from the outer-iteration of the interior-point [3, 4]. Finally, recent developments of inertia-free methods for nonconvex nonlinear optimization enable the use of iterative linear solvers for the Newton subproblem.

### 4.2 Optimal Power Flow Problems

We now demonstrate the computational benefits of using ADMM-GMRES by solving stochastic optimal power flow problems. The optimal power flow problem is frequently used in power networks to determine an efficient dispatch of power generators that satisfy demand and maintains feasible operation conditions. This approach assumes that demand forecasts are accurate and determines a nominal operation point for power generation, power flow in transmission lines and voltage angle at each bus in the power grid. We solve the DC-power flow problem for 35 benchmark problems available in the PGLIB.OPF library [43] and determine nominal operation points for each of them.

The solution of each benchmark problem was also obtained using Ipopt.

To assess the computational performance of ADMM-GMRES, we formulated a set-point problem that uses the nominal solution of the DC-power flow problem but seeks to minimize the effect of potential uncertainty in electricity demand values. The optimization solved is the quadratic problem:

$$\min_{x_i, z} \sum_{s \in \mathcal{P}} \sum_{j \in \Omega_G} w_j (P_{Gj,s} - P^\dagger_{Gj})^2 + \sum_{i,j \in \Omega_L} w_{i,j} (P_{Fi,j,s} - P^\dagger_{Fi,j})^2 + \sum_{j \in \Omega_B} w_j (\theta_{j,s} - \theta^\dagger_{j})^2$$

(22a)

s.t. $\sum_{j \in \Omega_{Gi}} P_{Gj,s} - P_{Li,s} = \sum_{j \in \Omega_i} P_{Fi,j,s}, \ i \in \Omega_G, s \in \mathcal{P}$

(22b)

$$P_{Fi,j,s} = \frac{\theta_{i,s} - \theta_{j,s}}{X_{i,j}}, \ i, j \in \Omega_L, s \in \mathcal{P}$$

(22c)

$$P_{Gj,s} - z_j = 0, \ j \in \Omega_G, s \in \mathcal{P}$$

(22d)

$$\theta_{0,s} = 0, \ s \in \mathcal{P}.$$  

(22e)

Here, $\Omega_B$ and $\Omega_L$ denote the set of buses and transmission lines in the network, $\Omega_G$ the set of generators, $\Omega_{Gi}$ the set of generators at bus $i$, and $\Omega_i$ the set of buses connected to bus $i$. The variables in the model are the generator outputs $P_G$, the flow in the transmission lines $P_F$, and the voltage angles $\theta_j$. As parameters we have the reactance of the lines $X_{i,j}$, the loads $P_L$, and the set-point values $P^\dagger_G$, $P^\dagger_F$, and $\theta^\dagger$ obtained from the DC-power flow solution. We denote the reference bus as $\theta_0$ and define objective weight values $w$. The goal of formulation (22) is to find the closest feasible operation to the optimal DC-power flow solution while accounting for potential uncertainty in the demands. In this problem the first-stage variables are the output of the generators for which we use Equation (22d) to enforce the same power generation across the set of scenarios. The dimensionality of the first-stage of this problem is given by the number of generators in the power network. Hence, this number varies from three to 4,092 in the 35 different benchmark problems considered in our study. For each benchmark we generated 50 random scenarios with normal random distributed noise on the load $P_L$.

Tables 1 and 2 summarize the results for the 35 benchmarks. The problems were sorted according to their number of coupling variables. Table 1 presents the results for benchmarks with a first-stage
dimension greater than 100. Results for the smaller benchmarks are shown in Table 2. We see that, for the smaller problems, Schur decomposition is the best alternative as it can give exact solutions in about the same time as ADMM. However, for all benchmarks shown in Table 1, ADMM-GMRES finds an $\epsilon$-accurate solution in less time than ADMM and Schur decomposition. In particular, for case13659_pegase (with 4,091 first-stage variables), ADMM-GMRES solved the problem almost an order of magnitude faster than Schur decomposition. By observing the trend for the rest of the problems, we conclude that these favorable scalability results can be expected to hold as the number of coupling variables increases. We highlight that, for many of the problems shown in Table 1, ADMM does not converge after 2,000 iterations; ADMM-GMRES, on the other hand, consistently achieves $\epsilon$-accurate solutions in few iterations and regardless of the choice of $\rho$. In summary, our results demonstrate that ADMM-GMRES provides a plausible approach to overcome the limitations of Schur complement decomposition.
Table 1: Performance for ADMM-GMRES (ADGM), ADMM, and Schur decomposition (Schur) on problem (22) with \( n > 100 \).

<table>
<thead>
<tr>
<th>Case</th>
<th>P17</th>
<th>P16</th>
<th>P15</th>
<th>P14</th>
<th>P13</th>
<th>P12</th>
<th>P11</th>
<th>P10</th>
<th>P9</th>
<th>P8</th>
<th>P7</th>
<th>P6</th>
<th>P5</th>
<th>P4</th>
<th>P3</th>
<th>P2</th>
<th>P1</th>
</tr>
</thead>
<tbody>
<tr>
<td>n x z</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Solution Time (s)</td>
<td>311600</td>
<td>103</td>
<td>8455</td>
<td>103</td>
<td>88650</td>
<td>142</td>
<td>5262</td>
<td>3727</td>
<td>4728</td>
<td>64665</td>
<td>6278</td>
<td>112178</td>
<td>115185</td>
<td>95220</td>
<td>10259</td>
<td>722</td>
<td></td>
</tr>
<tr>
<td>Iteration Count</td>
<td>2000</td>
<td>722</td>
<td>4091</td>
<td>1670.692</td>
<td>723.158</td>
<td>183.298</td>
<td>2000</td>
<td>325</td>
<td>2.798E-09</td>
<td>3.466E-03</td>
<td>1.602E-08</td>
<td>367900</td>
<td>272</td>
<td>23728</td>
<td>173977</td>
<td>122372</td>
<td></td>
</tr>
<tr>
<td>( | \hat{H} - H | )</td>
<td>63</td>
<td>75</td>
<td>22</td>
<td>129</td>
<td>70</td>
<td>31</td>
<td>112</td>
<td>272</td>
<td>129</td>
<td>70</td>
<td>31</td>
<td>112</td>
<td>272</td>
<td>129</td>
<td>70</td>
<td>31</td>
<td>112</td>
</tr>
</tbody>
</table>

Table 1: Performance for ADMM-GMRES (ADGM), ADMM, and Schur decomposition (Schur) on problem (22) with \( n > 100 \).
Table 2: Performance for ADMM-GMRES (ADGM), ADMM and Schur decomposition (Schur) on problem (22) with \( n^2 \leq 100 \).

<table>
<thead>
<tr>
<th>Case</th>
<th>ADGM</th>
<th>ADMM</th>
<th>Schur</th>
<th>Schur</th>
<th>ADGM</th>
<th>ADMM</th>
<th>Schur</th>
<th>Schur</th>
<th>ADGM</th>
<th>ADMM</th>
<th>Schur</th>
<th>Schur</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>( |t_n - t_H| )</td>
<td>Iteration Count</td>
<td>Solution Time (s)</td>
<td>( z )</td>
<td>( x )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P33. case3libind</td>
<td>7.080E-09</td>
<td>1.35E-13</td>
<td>0.025</td>
<td>1.023</td>
<td>0.342</td>
<td>44</td>
<td>3</td>
<td>1.515E-13</td>
<td>0.025</td>
<td>1.023</td>
<td>0.342</td>
<td>44</td>
</tr>
<tr>
<td>P34. case3libind</td>
<td>9.659E-14</td>
<td>0.025</td>
<td>1.023</td>
<td>0.342</td>
<td>44</td>
<td>3</td>
<td>1.515E-13</td>
<td>0.025</td>
<td>1.023</td>
<td>0.342</td>
<td>44</td>
<td>3</td>
</tr>
<tr>
<td>P35. case3libind</td>
<td>2.79E-13</td>
<td>0.034</td>
<td>1.037</td>
<td>0.346</td>
<td>48</td>
<td>3</td>
<td>1.355E-13</td>
<td>0.034</td>
<td>1.037</td>
<td>0.346</td>
<td>48</td>
<td>3</td>
</tr>
<tr>
<td>P36. case3libind</td>
<td>2.17E-09</td>
<td>0.782</td>
<td>0.782</td>
<td>0.782</td>
<td>82</td>
<td>7</td>
<td>2.17E-09</td>
<td>0.782</td>
<td>0.782</td>
<td>0.782</td>
<td>82</td>
<td>7</td>
</tr>
<tr>
<td>P37. case3libind</td>
<td>6.50E-08</td>
<td>1.03E+01</td>
<td>0.702</td>
<td>2.17E-09</td>
<td>0.782</td>
<td>7</td>
<td>2.17E-09</td>
<td>0.782</td>
<td>0.782</td>
<td>0.782</td>
<td>82</td>
<td>7</td>
</tr>
<tr>
<td>P38. case3libind</td>
<td>1.97E-07</td>
<td>1.18E+01</td>
<td>1.18E+01</td>
<td>1.18E+01</td>
<td>18</td>
<td>11</td>
<td>1.97E-07</td>
<td>1.18E+01</td>
<td>1.18E+01</td>
<td>1.18E+01</td>
<td>18</td>
<td>11</td>
</tr>
<tr>
<td>P39. case3libind</td>
<td>8.34E-07</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>30</td>
<td>0</td>
<td>8.34E-07</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>P40. case3libind</td>
<td>9.0E-06</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>32</td>
<td>2</td>
<td>9.0E-06</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>32</td>
<td>2</td>
</tr>
<tr>
<td>P41. case3libind</td>
<td>2.10E+00</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>20</td>
<td>2</td>
<td>2.10E+00</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>P42. case3libind</td>
<td>3.00E+00</td>
<td>3.05E+02</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>32</td>
<td>2</td>
<td>3.00E+00</td>
<td>3.05E+02</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>32</td>
<td>2</td>
</tr>
<tr>
<td>P43. case3libind</td>
<td>5.00E+00</td>
<td>3.05E+02</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>32</td>
<td>2</td>
<td>5.00E+00</td>
<td>3.05E+02</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>32</td>
<td>2</td>
</tr>
<tr>
<td>P44. case3libind</td>
<td>4.0E+00</td>
<td>8.00E+01</td>
<td>8.00E+01</td>
<td>8.00E+01</td>
<td>9</td>
<td>0</td>
<td>4.0E+00</td>
<td>8.00E+01</td>
<td>8.00E+01</td>
<td>8.00E+01</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>P45. case3libind</td>
<td>3.00E+00</td>
<td>3.05E+02</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>32</td>
<td>2</td>
<td>3.00E+00</td>
<td>3.05E+02</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>32</td>
<td>2</td>
</tr>
<tr>
<td>P46. case3libind</td>
<td>5.00E+00</td>
<td>3.05E+02</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>32</td>
<td>2</td>
<td>5.00E+00</td>
<td>3.05E+02</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>32</td>
<td>2</td>
</tr>
<tr>
<td>P47. case3libind</td>
<td>4.0E+00</td>
<td>8.00E+01</td>
<td>8.00E+01</td>
<td>8.00E+01</td>
<td>9</td>
<td>0</td>
<td>4.0E+00</td>
<td>8.00E+01</td>
<td>8.00E+01</td>
<td>8.00E+01</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>P48. case3libind</td>
<td>3.00E+00</td>
<td>3.05E+02</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>32</td>
<td>2</td>
<td>3.00E+00</td>
<td>3.05E+02</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>32</td>
<td>2</td>
</tr>
<tr>
<td>P49. case3libind</td>
<td>5.00E+00</td>
<td>3.05E+02</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>32</td>
<td>2</td>
<td>5.00E+00</td>
<td>3.05E+02</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>32</td>
<td>2</td>
</tr>
<tr>
<td>P50. case3libind</td>
<td>4.0E+00</td>
<td>8.00E+01</td>
<td>8.00E+01</td>
<td>8.00E+01</td>
<td>9</td>
<td>0</td>
<td>4.0E+00</td>
<td>8.00E+01</td>
<td>8.00E+01</td>
<td>8.00E+01</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>P51. case3libind</td>
<td>3.00E+00</td>
<td>3.05E+02</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>32</td>
<td>2</td>
<td>3.00E+00</td>
<td>3.05E+02</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>32</td>
<td>2</td>
</tr>
<tr>
<td>P52. case3libind</td>
<td>5.00E+00</td>
<td>3.05E+02</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>32</td>
<td>2</td>
<td>5.00E+00</td>
<td>3.05E+02</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>32</td>
<td>2</td>
</tr>
<tr>
<td>P53. case3libind</td>
<td>4.0E+00</td>
<td>8.00E+01</td>
<td>8.00E+01</td>
<td>8.00E+01</td>
<td>9</td>
<td>0</td>
<td>4.0E+00</td>
<td>8.00E+01</td>
<td>8.00E+01</td>
<td>8.00E+01</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>P54. case3libind</td>
<td>3.00E+00</td>
<td>3.05E+02</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>32</td>
<td>2</td>
<td>3.00E+00</td>
<td>3.05E+02</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>32</td>
<td>2</td>
</tr>
<tr>
<td>P55. case3libind</td>
<td>5.00E+00</td>
<td>3.05E+02</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>32</td>
<td>2</td>
<td>5.00E+00</td>
<td>3.05E+02</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>32</td>
<td>2</td>
</tr>
<tr>
<td>P56. case3libind</td>
<td>4.0E+00</td>
<td>8.00E+01</td>
<td>8.00E+01</td>
<td>8.00E+01</td>
<td>9</td>
<td>0</td>
<td>4.0E+00</td>
<td>8.00E+01</td>
<td>8.00E+01</td>
<td>8.00E+01</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>P57. case3libind</td>
<td>3.00E+00</td>
<td>3.05E+02</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>32</td>
<td>2</td>
<td>3.00E+00</td>
<td>3.05E+02</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>32</td>
<td>2</td>
</tr>
<tr>
<td>P58. case3libind</td>
<td>5.00E+00</td>
<td>3.05E+02</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>32</td>
<td>2</td>
<td>5.00E+00</td>
<td>3.05E+02</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>32</td>
<td>2</td>
</tr>
<tr>
<td>P59. case3libind</td>
<td>4.0E+00</td>
<td>8.00E+01</td>
<td>8.00E+01</td>
<td>8.00E+01</td>
<td>9</td>
<td>0</td>
<td>4.0E+00</td>
<td>8.00E+01</td>
<td>8.00E+01</td>
<td>8.00E+01</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>P60. case3libind</td>
<td>3.00E+00</td>
<td>3.05E+02</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>32</td>
<td>2</td>
<td>3.00E+00</td>
<td>3.05E+02</td>
<td>1.8E+01</td>
<td>1.8E+01</td>
<td>32</td>
<td>2</td>
</tr>
</tbody>
</table>

16
Figure 4: Computational times for Schur decomposition, ADMM, and ADMM-GMRES for problems with $n_z \geq 100$.

Figure 5: Computational times for Schur decomposition, ADMM, and ADMM-GMRES for problems with $n_z < 100$. 
Figure 6: Residuals for Schur decomposition, ADMM, and ADMM-GMRES for problems with $n_z \geq 100$.

Figure 7: Residuals for Schur decomposition, ADMM, and ADMM-GMRES for problems with $n_z < 100$.

5 Conclusions and Future Work

We have demonstrated that ADMM provides an effective mechanism to precondition iterative linear solvers and with this overcome scalability limitations of Schur complement decomposition. Our
results also demonstrate that the approach is robust to the choice of the penalty parameter. As part of future work, we will investigate the performance of ADMM-GMRES within a nonlinear interior-point framework. Here, it will be necessary to relax our assumptions on strong convexity and on the full row rank of the Jacobian. Preliminary results reported in the literature indicate that different types of primal-dual regularized KKT systems can be used to compute search steps within interior-point methods under such relaxed conditions [9]. For instance, the primal-dual regularized system correspond to the optimality conditions of the QP problem:

\[
\min_{x, z, r} \frac{1}{2} x^T (D + \delta I)x + c^T x + \frac{1}{2\rho} \|r\|^2 + \frac{\rho}{2} \|Ax + Bz - \frac{1}{\rho} r\|^2 \\
\text{s.t.} \quad Ax + Bz - \frac{1}{\rho} r = 0, \quad (y)
\]

We will investigate ADMM variants to precondition such systems. The effectiveness of using ADMM as a preconditioner makes us wonder whether other approaches can be used for preconditioning as well. For instance, inexact dual Newton strategies can potentially be used to precondition structured KKT systems. This is an interesting direction of future work. We will also investigate advanced ADMM strategies that use second-order multiplier updates to accelerate the preconditioner.

We will also investigate using ADMM-GMRES for computing step directions within an interior-point framework to solve problems with bounds and/or inequality constraints. For this task we will need to compute the \( D \) matrix as \( \tilde{D} = D + \Sigma_x \) in every interior-point iteration, where the \( \Sigma_x \) matrix corrects for the presence of bounds (see [36]). This \( \Sigma_x \) matrix introduces ill-conditioning as the interior-point iterations progress and we plan to investigate the robustness of ADMM-GMRES to the presence of \( \Sigma_x \).

**Disclaimer**

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-NA-0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. This work was conducted as part of the Institute for the Design of Advanced Energy Systems (IDAES) with funding from the Office of Fossil Energy, Cross-Cutting Research, U.S. Department of Energy. V. M. Zavala acknowledges funding from the National Science Foundation under award NSF-EECS-1609183.

**References**


### A Computing Operator $T_\rho(u)$ using ADMM

Here we prove that the operator $T_\rho(u)$ can be computed by applying one ADMM iteration. Consider, without loss of generality (and in order to simplify the presentation), the case of a single block problem of the form:

$$
\min_{x,z} \frac{1}{2} x^T Dx + c^T x + \frac{\rho}{2} \| Ax + Bz \|^2
$$

s.t. $Ax + Bz = 0, (y).$

The results that we derive next can be extended to multiple blocks using induction. The KKT system for problem (24) is:

$$
\begin{bmatrix}
    K_\rho & \rho A^T B & A^T \\
    \rho B^T A & \rho B^T B & B^T \\
    A & B
\end{bmatrix}
\begin{bmatrix}
    x \\
    z \\
    y
\end{bmatrix}
= 
\begin{bmatrix}
    -c \\
    0 \\
    0
\end{bmatrix}
$$

where $K_\rho = D + \rho A^T A$. Applying a Gauss-Seidel splitting to this system at a point $u = (x, z, y)$ leads to the update $u^+ = T_\rho(u) = G_\rho u + f_\rho$, where $G_\rho = M_\rho^{-1} N_\rho$ and $f_\rho = M_\rho^{-1} r$. The explicit form of $M_\rho^{-1}$ is given by:

$$
M_\rho^{-1} = 
\begin{bmatrix}
    K_\rho^{-1} & 0 & 0 \\
    -\Sigma^{-1} B^T A K_\rho^{-1} & \frac{1}{\rho} \Sigma^{-1} & 0 \\
    \rho(I - B\Sigma^{-1} B^T) A K_\rho^{-1} & B\Sigma^{-1} & -\rho I
\end{bmatrix}
$$

where $\Sigma := B^T B$. Having $M_\rho^{-1}$ we construct:

$$
G_\rho = M_\rho^{-1} N_\rho = 
\begin{bmatrix}
    K_\rho^{-1} & 0 & 0 \\
    -\Sigma^{-1} B^T A K_\rho^{-1} & \frac{1}{\rho} \Sigma^{-1} & 0 \\
    \rho(I - B\Sigma^{-1} B^T) A K_\rho^{-1} & B\Sigma^{-1} & -\rho I
\end{bmatrix}
\begin{bmatrix}
    0 & -\rho A^T B & -A^T \\
    0 & 0 & -B^T \\
    0 & -\rho K_\rho^{-1} A^T B \\
    \rho^2(B\Sigma^{-1} B^T - I) A K_\rho^{-1} A^T B & \rho(B\Sigma^{-1} B^T - I) A K_\rho^{-1} A^T - B\Sigma^{-1} B^T + I
\end{bmatrix}
$$

and the right-hand-side-vector

$$
f_\rho = M_\rho^{-1} r = 
\begin{bmatrix}
    K_\rho^{-1} & 0 & 0 \\
    -\Sigma^{-1} B^T A K_\rho^{-1} & \frac{1}{\rho} \Sigma^{-1} & 0 \\
    \rho(I - B\Sigma^{-1} B^T) A K_\rho^{-1} & B\Sigma^{-1} & -\rho I
\end{bmatrix}
\begin{bmatrix}
    -c \\
    0 \\
    0
\end{bmatrix}
$$

By defining $Q := A K_\rho^{-1} A^T$ we can write the explicit form of the update $u^+$ as:
\[
\begin{bmatrix}
  x^+ \\
  z \\
  y
\end{bmatrix}
= \begin{bmatrix}
  0 & -\rho K^{-1}_\rho A^T B & -K^{-1}_\rho A^T \\
  0 & \rho \Sigma^{-1} B^T Q B & \Sigma^{-1} B^T (Q - \frac{1}{\rho} I) \\
  0 & \rho^2 (B \Sigma^{-1} B^T - I) Q B & \rho (B \Sigma^{-1} B^T - I) Q - B \Sigma^{-1} B^T + I
\end{bmatrix}
\begin{bmatrix}
  x \\
  z \\
  y
\end{bmatrix}
+ \begin{bmatrix}
  -K^{-1}_\rho c \\
  \Sigma^{-1} B^T A K^{-1}_\rho c \\
  \rho (B \Sigma^{-1} B^T - I) A K^{-1}_\rho c
\end{bmatrix}
\]

(29)

Upon expansion we obtain the update \( u^+ = (x^+, z^+, y^+) \):

\[
x^+ = -\rho K^{-1}_\rho A^T B z - K^{-1}_\rho A^T y - K^{-1}_\rho c \quad (30a)
\]

\[
z^+ = \rho \Sigma^{-1} B^T Q B z + \Sigma^{-1} B^T (Q - \frac{1}{\rho} I) y + \Sigma^{-1} B^T A K^{-1}_\rho c \quad (30b)
\]

\[
y^+ = \rho^2 (B \Sigma^{-1} B^T - I) Q B z + \left[ \rho (B \Sigma^{-1} B^T - I) Q - B \Sigma^{-1} B^T + I \right] y + \rho (B \Sigma^{-1} B^T - I) A K^{-1}_\rho c \quad (30c)
\]

We now show that ADMM delivers the same updates after one iteration. We use the augmented Lagrange function:

\[
\mathcal{L}(x, z, y) = x^T D x + c^T x + (A x + B z)^T y + \rho \frac{1}{2} \| A x + B z \|^2.
\]

(31)

Initializing at \( u = (x, z, y) \), the update \( x^+ \) is given by:

\[
x^+ = \arg \min_x \mathcal{L}_\rho(x, z, y) \quad (32)
\]

For which the optimality conditions are

\[
\nabla_x \mathcal{L}_\rho(x, z, y) = (D + \rho A^T A) x + \rho A^T B z + A^T y + c = 0 \quad (33)
\]

and thus,

\[
x^+ = -(D + \rho A^T A)^{-1} \left[ \rho A^T B z + A^T y + c \right] \\
= -K^{-1}_\rho \left[ \rho A^T B z + A^T y + c \right] \\
= -\rho K^{-1}_\rho A^T B z - K^{-1}_\rho A^T y - K^{-1}_\rho c \quad (34)
\]

We note that (34) and (30a) are equivalent. The update for the coupling variables \( z^+ \) is given by:

\[
z^+ = \arg \min_z \mathcal{L}_\rho(x^+, z, y) \quad (35)
\]

The optimality conditions are given by:

\[
\nabla_z \mathcal{L}_\rho(x^+, z, y) = B^T y + \rho B^T A x^+ + \rho B^T B z = 0 \quad (36)
\]
and thus,

\[ z^+ = - (B^T B)^{-1} \left[ \frac{1}{\rho} B^T y + B^T A x^+ \right] \]

\[ = - \Sigma^{-1} \left[ \frac{1}{\rho} B^T y + B^T A x^+ \right] \]

\[ = - \Sigma^{-1} \left[ \frac{1}{\rho} B^T y - \rho B^T A^{-1} K_{\rho} A^T B z - B^T A^{-1} K_{\rho} A^T y - B^T A K_{\rho}^{-1} c \right] \]

\[ = - \Sigma^{-1} \left[ \frac{1}{\rho} B^T y - \rho B^T Q B z - B^T Q y - B^T A K_{\rho}^{-1} c \right] \]

\[ = - \Sigma^{-1} \left[ B^T \left( \frac{1}{\rho} I - Q \right) y - \rho B^T Q B z - B^T A K_{\rho}^{-1} c \right] \]

\[ = \rho \Sigma^{-1} B^T Q B z + \Sigma^{-1} B^T \left( Q - \frac{1}{\rho} I \right) y + \Sigma^{-1} B^T A K_{\rho}^{-1} c \]  \hspace{1cm} (37)