12th IFAC International Symposium on
Advanced Control of Chemical Processes
July 14-17, 2024. Toronto, Canada

Evaluation of Direct and Iterative Approaches for the Parallel Solution of Structured Nonlinear Optimization Problems

Laurens R. Lueg * **Michael Bynum** ** **Carl D. Laird** *
Lorenz T. Biegler *

* *Chemical Engineering Department, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: lb01@andrew.cmu.edu)*
** *Sandia National Laboratories, Albuquerque, NM 87185 USA*

Abstract: Large-scale nonlinear optimization problems arise in a variety of applications and often exhibit some structure, which can be exploited by the use of parallel decompositions to speed up the solution. We present a general problem formulation for structured optimization problems and apply the interior point method, outlining different approaches to parallelize the step computations using the Schur complement decomposition. The use of an iterative linear solver can boost performance, given an appropriate preconditioner for the Schur complement. We present an approach to use sparse factorizations from previous solver iterations as a preconditioner, and compare it to both an L-BFGS preconditioner and direct solution.

Keywords: distributed optimization for large-scale systems, parallel computing, nonlinear optimization

1. INTRODUCTION

The solution of nonlinear programming problems (NLPs) is an essential task in a variety of applications, such as process design, parameter estimation or model-predictive control. Different problem features, such as discretized dynamics or spatial heterogeneity, induce a structure on the associated optimization problem, which may be exploited for parallelization. This is especially relevant in applications where the solution of the optimization problem is required in real-time.

The development of parallel methods for NLPs is an active area of research. The interior point method (IPM) is a widely-used algorithm for solving large-scale NLPs and it incurs most of the computational cost by factorizing a sparse indefinite KKT system at every iteration (Wächter and Biegler (2006)). For structured problems, the use of linear-algebra-level decompositions, such as the Schur complement, can be employed to speed up this operation (Zavala et al. (2008); Kang et al. (2014); Rodriguez et al. (2023)). The use of iterative linear solvers in combination with these methods has been identified as a promising avenue for performance improvements. However, the use of an appropriate preconditioner is imperative (Kang et al. (2014)). In this work, we take a closer look at this aspect for grid-structured parameter estimation problems. We compare the direct formation and factorization of the Schur complement with the implicit solution using an L-BFGS preconditioner proposed in Kang et al. (2014), as well as a preconditioner based on sparse factorizations of the Schur complement from previous solver iterations. We discuss the performance of the different approaches on a medium-sized problem and outline the trade-offs with respect to parallel performance.

2. PROBLEM STATEMENT

We begin by giving a general formulation for a structured optimization problem, which is characterized by n_P problem partitions, indexed below by k:

$$\min \sum_{k=1}^{n_P} \psi_k(x_k, y_k) \tag{1a}$$
$$\text{s. t.} \quad h_k(x_k, y_k) = 0 \qquad \forall_k \tag{1b}$$
$$g_k(x_k, y_k) \leq 0 \qquad \forall_k \tag{1c}$$
$$y_k = N_k^\top y \qquad \forall_k \tag{1d}$$
$$x_k \in \mathbb{R}^{n_k}, \quad y_k \in \mathbb{R}^{p_k} \qquad \forall_k \tag{1e}$$
$$y \in \mathbb{R}^p \tag{1f}$$

where $g_k : \mathbb{R}^{n_k+p_k} \mapsto \mathbb{R}^{r_k}$, $h_k : \mathbb{R}^{n_k+p_k} \mapsto \mathbb{R}^{l_k}$ and $\psi_k(x_k, y_k) : \mathbb{R}^{n_k+p_k} \mapsto \mathbb{R}$ are smooth and possibly non-convex functions. $N_k^\top \in \mathbb{R}^{p_k \times p}$ is a binary matrix with row sum equal to one and column sum at most one. This selects the appropriate complicating variables from $y \in \mathbb{R}^p$ ($p_k \leq p$) for each partition k. We can visualize the extent of interaction between problem partitions by defining a bipartite graph $\mathcal{G} = (U, V, E)$, where

$$U = \bigcup_{k=1}^{n_P} \{y_k^{(i)}\}_{\forall i=1\ldots p_k}, \quad V = \{y^{(i)}\}_{\forall i=1\ldots p} \tag{2}$$
$$E = \{(y_k^{(i)}, y^{(j)}) \in U \times V \mid N_k^{(j,i)} = 1\}, \tag{3}$$

where vertices U and V represent local and global complicating variables, respectively. An example of such a graph for a small problem is shown in Fig. 1 (U in white, V

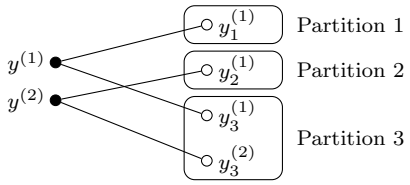

Fig. 1. Visualization of problem structure with respect to complicating variables. Global complicating variables on the left, local complicating variables on the right. Edges represent equality constraints.

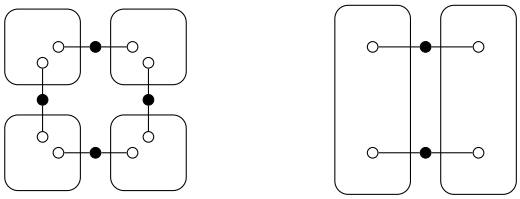

Fig. 2. Depending on partitioning, different structures can arise for identical problems.

in black). Edges E represent the mapping from global to local variables for each partition, according to N_k. Various types of optimization problems can be described using this framework, which shares some characteristics with previous works on describing structure in optimization problems, e.g. variable graphs in Allman et al. (2019) or a modeling paradigm based on hypergraphs (Jalving et al. (2022)). In considering a structured optimization problem as described in (1), a partitioning (and thus induced graph $\mathcal{G}$) is not unique. In Fig. 2, two alternatives for an example structure are shown. By reducing the number of partitions, the number of complicating variables is halved. Conversely, the size of the individual partitions in terms of n_k (sketched as size of the rounded boxes in Fig. 2) increases. This defines a trade-off between number of complicating variables and the number of partitions, as well as the balancing of partition sizes. This trade-off will become important for the parallel solution of structured problems using the Schur complement method.

3. INTERIOR POINT METHOD FOR STRUCTURED PROBLEMS

We apply the standard interior point method to Problem (1). After moving inequality constraints (1c) to the objective using a logarithmic barrier term with coefficient μ, the Lagrangian of the resulting augmented problem is given by:

$$\mathcal{L} = \sum_{k=1}^{n_P} \psi_k(x_k, N_k^\top y) + \sum_{k=1}^{n_P} \lambda_k^\top h_k(x_k, N_k^\top y) + \mu \sum_{k=1}^{n_P} \sum_{i=1}^{r_k} \ln(-g_k^{(i)}(x_k, N_k^\top y)), \tag{4}$$

where we substituted the local definition for y_k in terms of the global complicating variables. The KKT conditions for first-order optimality for the augmented problem are hence given by:

$$\nabla_{x_k}\mathcal{L} = \nabla_{x_k}\psi + \nabla_{x_k}h_k\lambda_k + \nabla_{x_k}g_k z_k = 0,\ \forall_{k=1\dots n_P} \tag{5a}$$

$$h_k = 0,\ \forall_{k=1\dots n_P}, \tag{5b}$$

$$G_k z_k = \mu \mathbf{1}_{r_k},\ \forall_{k=1\dots n_P} \tag{5c}$$

$$\nabla_y \mathcal{L} = \sum_{k=1}^{n_P} N_k \left[\nabla_{y_k}\psi_k + \nabla_{y_k}h_k\lambda_k + \nabla_{y_k}g_k z_k\right] = 0 \tag{5d}$$

where $G_k = \operatorname{diag}(\{-g_k^{(j)}\}_{\forall j=1\dots r_k}) \in \mathbb{R}^{r_k \times r_k}$. The evaluation points for all functions were dropped for readability, but they are analogous to (4). Applying a modified Newton's method to solve (5), the following linear system is solved at each iteration:

$$\begin{bmatrix} W_1 & & & A_1 \\ & W_2 & & A_2 \\ & & \ddots & \vdots \\ & & W_{n_P} & A_{n_P} \\ A_1^\top & A_2^\top & \cdots & A_{n_P}^\top \quad D \end{bmatrix} \begin{bmatrix} \Delta u_1 \\ \Delta u_2 \\ \vdots \\ \Delta u_{n_P} \\ \Delta y \end{bmatrix} = - \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_{n_P} \\ r_y \end{bmatrix}, \tag{6}$$

where

$$W_k = \begin{bmatrix} \nabla^2_{x_k x_k}\mathcal{L} + \nabla_{x_k}g_k G_k^{-1}\bar{Z}_{x_k} + \delta_H \mathbf{I} & \nabla_{x_k}h_k \\ (\nabla_{x_k}h_k)^\top & -\delta_C \mathbf{I} \end{bmatrix}, \tag{7}$$

$$\Delta u_k = \begin{bmatrix} \Delta x_k \\ \Delta \lambda_k \end{bmatrix}, \tag{8}$$

$$A_k^\top = N_k \begin{bmatrix} \nabla_{y_k x_k}\mathcal{L} + \bar{Z}_{y_k}^\top G_k^{-1}(\nabla_{x_k}g_k)^\top & \nabla_{y_k}h_k \end{bmatrix}, \tag{9}$$

$$r_k = \begin{bmatrix} \nabla_{x_k}\mathcal{L} + \nabla_{x_k}g_k(-z_k + \mu G_k^{-1}\mathbf{1}_{r_k}) \\ h_k \end{bmatrix}, \tag{10}$$

$$r_y = \sum_{k=1}^{n_P} N_k \left[\nabla_{y_k}\mathcal{L} + \nabla_{y_k}g_k(-z_k + \mu G_k^{-1}\mathbf{1}_{r_k})\right], \tag{11}$$

$$D = \delta_H \mathbf{I}_p + \sum_{k=1}^{n_P} N_k(\nabla^2_{y_k y_k}\mathcal{L} + \nabla_{y_k}g_k G_k^{-1}\bar{Z}_{y_k})N_k^\top. \tag{12}$$

The step direction in z_k was eliminated and can be recovered as

$$\Delta z_k = -z_k + G_k^{-1}[\bar{Z}_{x_k}\Delta x_k + \bar{Z}_{y_k}N_k^\top \Delta y + \mu \mathbf{1}_{r_k}], \tag{13}$$

where

$$\bar{Z}_{x_k} = (\nabla_{x_k}g_k Z_k)^\top \in \mathbb{R}^{r_k \times n_k}, \tag{14}$$

$$\bar{Z}_{y_k} = (\nabla_{y_k}g_k Z_k)^\top \in \mathbb{R}^{r_k \times p_k}. \tag{15}$$

The parameters δ_H and δ_C are chosen to ensure inertia requirements (Wächter and Biegler (2006)). The variables are updated by

$$x_k^+ = x_k + \alpha_p \Delta x_k, \tag{16}$$

$$\lambda_k^+ = \lambda_k + \alpha_d \Delta \lambda_k, \tag{17}$$

$$z_k^+ = z_k + \alpha_d \Delta z_k, \tag{18}$$

$$y^+ = y + \alpha_p \Delta y, \tag{19}$$

where the step lengths are chosen by the fraction-to-the-boundary rule (Wächter and Biegler (2006)). Once (5) is solved using this procedure, the barrier parameter μ is decreased, and the process is repeated until a solution tolerance for the overall problem (1) is met. In many applications where function and gradient evaluations are relatively cheap, the majority of the computational effort of solving (1) is spent on repeatedly solving systems described by (6).

3.1 Parallel Implementations of the Schur Complement Method

The solution of (6) can be decomposed as follows:

$$S = D - \sum_{k=1}^{n_P} A_k^\top W_k^{-1} A_k, \tag{20}$$

$$S\Delta y = -r_y + \sum_{k=1}^{n_P} A_k^\top W_k^{-1} r_k =: r_S, \tag{21}$$

$$\Delta u_k = -W_k^{-1}\left(A_k \Delta y + r_k\right), \tag{22}$$

where S is the *Schur complement*. For this approach, we identify the following high-level steps: forming the Schur complement (20), solving for the complicating variables (21) and solving for the local variables (22). This method has been termed the *explicit* Schur complement method (Kang et al. (2014)). A parallel implementation of this algorithm is given in Alg. 1. Note that we assume the number of available parallel processors to be equal to the number of partitions n_P. In this setting, factorizing the diagonal blocks (1.4), forming S and r_S (1.6-1.13) as well as computing Δu_k (1.18-1.21) can be performed in a distributed manner. In between, data needs to be shared between processors (1.14-1.15), where we used `allreduce` to denote a sum operation over all processors. Furthermore, Δy needs to be computed on all processors (1.16-1.17). The factorization and backsolve operations for W_k and S are performed using a sparse solver for indefinite systems, such as HSL MA27 (Duff and Reid (1983)).

Algorithm 1 Explicit Schur complement

1: **procedure** `xsc`$(\{W_k\}_{\forall k}, \{A_k\}_{\forall k}, D, \{r_k\}_{\forall k}, r_y)$
2: **Given:** sparse linear solver interface `lsi`
3: **for** $k = 1...n_P$ **do**
4: `lsi.factorize`(W_k) ▷ Factorize blocks
5: $V_k \leftarrow \mathbf{0} \in \mathbb{R}^{p \times p}$
6: **for** $i = 1...p_k$ **do** ▷ Form S
7: $j \leftarrow$ Index where $N_k^{(j,i)} = 1$
8: $m_k \leftarrow$ `lsi.backsolve`$(W_k, A_k^{\top(j)})$
9: $V_k^j \leftarrow V_k^j - A_k m_k$
10: **end for**
11: $v_k \leftarrow$ `lsi.backsolve`(W_k, r_k) ▷ Form r_S
12: $r_{Sk} \leftarrow A_k v_k$
13: **end for**
14: $S \leftarrow D +$ `allreduce`(V_k) ▷ Communicate
15: $r_S \leftarrow -r_y +$ `allreduce`(r_{Sk}) ▷ Communicate
16: `lsi.factorize`(S) ▷ Factorize S
17: $\Delta y \leftarrow$ `lsi.backsolve`(S, r_S) ▷ Solve for Δy
18: **for** $k = 1...n_P$ **do** ▷ Solve for Δu_k
19: $r_k \leftarrow r_k + A_k^\top \Delta y$
20: $\Delta u_k \leftarrow$ `lsi.backsolve`$(W_k, -(A_k^\top \Delta y + r_k))$
21: **end for**
22: **Return** $\{\Delta u_k\}_{\forall k}, \Delta y$
23: **end procedure**

An alternative method to utilize the Schur complement to solve (6) is to avoid forming S explicitly, and instead apply the preconditioned conjugate gradient (PCG) algorithm to solve

$$H\tilde{S}(\Delta y) = H r_S, \tag{23}$$

$$\text{where} \quad \tilde{S}(u) := \left(D - \sum_{k=1}^{n_P} A_k^\top W_k^{-1} A_k\right) u, \tag{24}$$

and H an appropriate preconditioner. This method has been termed the *implicit* Schur complement method (Kang et al. (2014)). In order to apply PCG, the Schur complement is required to be positive definite. This can be achieved by adjusting parameter δ_H whenever negative curvature is detected within a PCG step. See Kang et al. (2014) for a more detailed discussion on how to fulfill the IPM inertia requirements when applying the implicit Schur complement method. Note that PCG does not require the explicit formation of either S or H, but simply the ability to compute the associated matrix-vector product. The implicit Schur method is given in algorithmic form in Alg. 2. It is evident that the compute and communication costs incurred by the explicit variant to form and factorize S is avoided, at the expense of the use of the PCG algorithm.

Algorithm 2 Implicit Schur complement

1: **procedure** `isc`$(\{W_k\}_{\forall k}, \{A_k\}_{\forall k}, D, \{r_k\}_{\forall k}, r_y)$
2: **Given:** sparse linear solver interface `lsi`
3: **for** $k = 1...n_P$ **do**
4: `lsi.factorize`(W_k)
5: $v_k \leftarrow$ `lsi.backsolve`(W_k, r_k)
6: $r_{Sk} \leftarrow A_k v_k$
7: **end for**
8: $r_S \leftarrow -r_y +$ `allreduce`(r_{Sk})
9: Update preconditioner H
10: $\Delta y \leftarrow$ Solve (23) using PCG
11: **for** $k = 1...n_P$ **do**
12: $r_k \leftarrow r_k + A_k^\top \Delta y$
13: $\Delta u_k \leftarrow$ `lsi.backsolve`$(W_k, -(A_k^\top \Delta y + r_k))$
14: **end for**
15: **Return** $\{\Delta u_k\}_{\forall k}, \Delta y$
16: **end procedure**

Hence, in order to accurately compare the performance of the explicit and implicit Schur complement methods, the cost of using PCG needs to be evaluated. Indeed, PCG is an iterative scheme which requires the evaluation of the matrix-vector products $\tilde{S}(\cdot)$ (24) and $H(\cdot)$ (23) once per iteration. As the diagonal blocks W_k were already factorized in (2.4), the main computational cost of $\tilde{S}(\cdot)$ is the execution one backsolve operation per partition. PCG might still require many iterations to converge to a solution. To avoid this, a preconditioner $H(\cdot)$ is used to improve the conditioning of the linear system. However, defining a sensible preconditioner for the Schur complement system without forming S is not straightforward.

3.2 Preconditioning the Schur Complement System

The convergence rate of PCG for (21) depends on $\kappa(S)=\frac{\lambda_{\max}(S)}{\lambda_{\min}(S)}$, with improved convergence the closer κ is to 1 (Saad (2003)). The use of a preconditioner aims to improve the condition number of the preconditioned system (23), thus decreasing the number of PCG iterations needed to solve the system, at the expense of applying the preconditioner once per PCG iteration. An additional cost is incurred by forming the preconditioner. In the context

of interior point methods, where systems such as (6) are solved repeatedly, the use of an L-BFGS preconditioner based on PCG iterates from previous interior-point iterations has been proposed (Morales and Nocedal (2000), Kang et al. (2014)). This involves maintaining a memory of m past PCG iterates, and computing $H(\cdot)$ using formulations as described in Nocedal (1980) and Byrd et al. (1994). In the case of L-BFGS, the preconditioner requires storage of $O(m)$ dense vectors of size p, whereas applying it requires $O(mp)$ floating point operations.

Here, we also propose an alternative preconditioning scheme for Schur complement systems in the context of the interior-point method. Instead of maintaining a memory of CG iterates to compute $H(\cdot)$, the Schur complement is formed and factorized explicitly in the first interior-point iteration, and the associated backsolve operation is subsequently used as preconditioner. If the number of PCG iterations required to solve (23) exceeds a specified threshold τ_{cg}, the preconditioner is updated by re-forming and factorizing S in the next interior point iteration. The preconditioner is thus

$$H(u) = \texttt{backsolve}(S^-, u) \approx S^{-1}u, \tag{25}$$

where S^- is the most recently factorized Schur complement, and S the current one. To choose a threshold τ_{cg}, the cost of forming and factorizing S has to be weighed against the cost of additional PCG iterations at subsequent steps in the interior point method. A decision rule based on complexity estimates for backsolve and factorization operations is conceivable, however we omit this here and employ a heuristic that selects τ_{cg} proportional to the size of the Schur complement p. We call this approach adaptive Schur complement in the following.

4. COMPUTATIONAL EXPERIMENTS

The performance of the three approaches outlined above – explicit Schur complement (XSC), implicit Schur complement with L-BFGS preconditioner (ISC) and adaptive Schur complement (ASC) – is compared on a set of synthetic parameter estimation problems from the field of infectious disease modelling. The associated nonlinear optimization problem is given in (26). For each geographical region (e.g. counties, indexed by c in (26)), we define the differential variables S, E, I, R (denoting susceptible, exposed, infectious and recovered compartments of the county's population, respectively), the forcing term f, as well as an error term, which is decomposed into positive and negative components to allow for L1 regularization (cf. (26a)). The parameter to be estimated for each county c is the contact rate β, which is modeled as a piecewise constant function over time, with K time intervals. Fixed parameters of the model include the reporting fraction and reporting delay, ρ and κ, regularization coefficient λ, incubation rate σ and recovery rate γ. The model is fitted to simulated data of daily new cases Y, for n_C counties and T time steps. In our experiments, we use K=10 and T=200.

$$\min \quad \sum_{c=1}^{n_C}\sum_{t=1}^{T}(\rho\sigma E_{c,t-1} - Y_{c,t+\kappa})^2 + \lambda(\epsilon^+_{c,t} + \epsilon^-_{c,t}) \tag{26a}$$
$$S_{c,t+1} = (1 - f_{c,t})S_{c,t} - (\epsilon^+_{c,t} - \epsilon^-_{c,t}), \quad \forall_{c,t} \tag{26b}$$
$$E_{c,t+1} = (1 - \sigma)E_{c,t} + f_{c,t}S_{c,t} + (\epsilon^+_{c,t} - \epsilon^-_{c,t}), \quad \forall_{c,t} \tag{26c}$$
$$I_{c,t+1} = (1 - \gamma)I_{c,t} + \sigma E_{c,t}, \quad \forall_{c,t} \tag{26d}$$
$$R_{c,t+1} = R_{c,t} + \gamma I_{c,t}, \quad \forall_{c,t} \tag{26e}$$
$$f_{c,t} = \beta_{c,k(t)}I_{c,t} + \textstyle\sum_{j\neq c} C_{c,j,t}\beta_{j,k(t)}, \quad \forall_{c,t} \tag{26f}$$
$$S_{c,0} = 1, \;\; E_{c,0} = I_{c,0} = R_{c,0} = 0, \quad \forall_c \tag{26g}$$
$$\mathbf{S}, \mathbf{E}, \mathbf{I}, \mathbf{R}, \mathbf{f}, \boldsymbol{\epsilon}^+, \boldsymbol{\epsilon}^- \in \mathbb{R}^{n_C \times T}_{\geq 0}, \boldsymbol{\beta} \in \mathbb{R}^{n_C \times K}_{\geq 0}. \tag{26h}$$

Interaction between counties is modeled through the coefficient tensor C, where $C_{c,j,t}$ is an approximation for the likelihood of contact between individuals from county c and infected individuals from county j at time t (see Cummings et al. (2021) for more information on how to estimate this term). Here we assume the sparsity pattern of this tensor to be constant over time, so that a fixed structure of interaction between counties is defined. This is illustrated for the real-world example of Pennsylvania in Fig. 3, where interaction between counties is given as a graph.

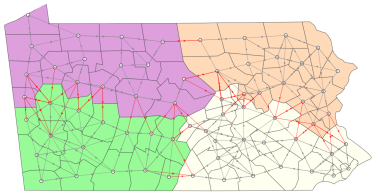

Fig. 3. Illustration of epidemic model structure for Pennsylvania, showing mobility patterns between counties (grey, red), as well as partitioning of counties with n_P=4 (colored).

In order to apply the methods discussed above to this problem, we must define a problem partitioning. Given that $n_P \leq n_C$, this requires the grouping of geographical regions into n_P partitions. In Fig. 3, we illustrate the partitioning of Pennsylvania into n_P=4 partitions. Given such a partitioning, the complicating variables of (26) are the contact rates of counties which are connected to counties from other problem partitions as defined by the sparsity pattern of C (highlighted in red in Fig. 3). Hence, the partitioning of geographical regions has direct impact on the number of complicating variables, partition size and thereby performance of the parallel decomposition schemes discussed in this report. This is discussed further in Sec. 4.2. In our tests, we only consider a simplified square grid of counties, where each county interacts with at most four neighbors. Lastly, we note that for the problems solved in our experiments, the only active inequalities at a local solution were those associated with the non-negativity of the decomposed error terms ϵ^+ and ϵ^- (26h). These can be avoided by using an L2 regularization, however this did not significantly affect the performance of the preconditioners in our tests. Generally, the relationship between the active set and the conditioning of the Schur complement of the KKT system could be a relevant topic for future work.

All tests were performed on a single node of the Bridges-2 machine at the Pittsburgh Supercomputing Center

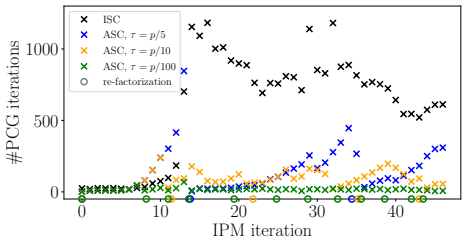

Fig. 4. Required PCG iterations using L-BFGS (ISC) and ASC preconditioners (for varying τ_{cg}) at each IPM iteration. Round markers indicate when re-factorizations where performed.

(Brown et al. (2021)), using at most 64 cores. For the L-BFGS preconditioner, m=50 PCG iterates were sampled uniformly from the previous interior point iteration to construct the preconditioner, as described in Morales and Nocedal (2001). For the ASC preconditioner, different values of τ_{cg} are investigated in Sec. 4.1, we chose $\tau_{\text{cg}}=p/10$ in Sec. 4.2. We used `parapint` [1], a Python package for parallel nonlinear optimization (Rodriguez et al. (2023)) to implement the algorithms outlined above.

4.1 Comparison of PCG Preconditioners

First, we will highlight the difference in performance between the ISC and ASC preconditioners. We applied the interior-point method to five instances (with different random initializations for data generation) of the test problem (26), with n_C=256 and n_P=64, resulting in p=2560 complicating variables. Using either an L-BFGS preconditioner or ASC for the Schur complement, we compare the average number of PCG iterations needed to solve (23) at each iteration of the interior point method. The results are shown in Fig. 4. The L-BFGS preconditioner performs well initially, however the performance deteriorates as the interior point method progresses. This is likely due to the ill-conditioning of the KKT system at later iterations, and the inability of the iterates from previous steps to accurately capture this development. For the ASC preconditioner, the number of PCG iterations is dependent on the threshold τ_{cg}. A lower threshold reduces the number of PCG steps, at the cost of more frequent re-factorizations of the Schur complement (as indicated by the round markers on the x-axis in Fig. 4).

4.2 Strong Scaling Tests

To investigate which of the proposed approaches achieves better overall parallel performance on the test problem, we solve the problem described above (n_C=256) repeatedly while increasing n_P and observing the required solution time, as well as the proportion of that time spent in the different subroutines of the respective methods. In this work, we use a naive partitioning of the 2D problem structure, the properties of which are summarized in Table 1. Namely, for each number of partitions/processors n_P considered here, we denote the total number of complicating variables p, the maximum number of local complicating variables across all partitions $\max_k p_k$, and the number of

[1] https://github.com/sandialabs/parapint

Table 1. Partitioning statistics.

n_P	2	4	8	16	32	64
p	320	960	2240	2560	2560	2560
$\max_k p_k$	320	640	640	480	250	140
n_k	153,600	76,800	38,400	19,200	9,600	4,800

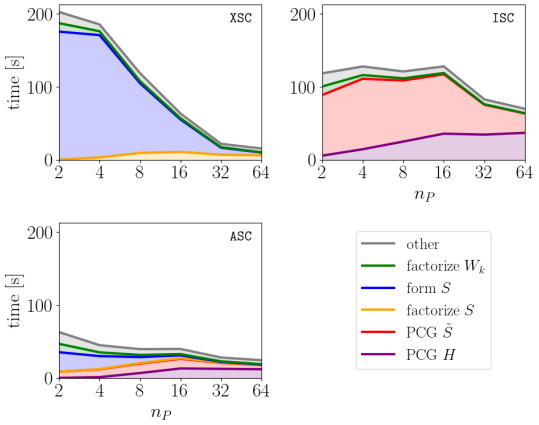

Fig. 5. Strong scaling profiles for XSC (top left), ISC (top right) and ASC (bottom left).

local variables n_k, which is equal for all partitions. These statistics will help us to understand the scaling results of the different methods. In Fig. 5, it is shown that although the overall solution time of XSC is high initially, forming the Schur complement scales well with n_P. This agrees with the relevant partitioning statistics from Table 1, as $\max_k p_k$ backsolves with W_k (size proportional to n_k) have to be performed in serial to form S – both eventually decrease as n_P increases. In the case of ISC, we observe virtually no performance improvements up to n_P=16, as the number of PCG iterations increases with p, yielding no performance gain even though the time required to apply $\tilde{S}(\cdot)$ decreases with n_k. As p stagnates for $n_P > 16$, we do observe the parallelization of $\tilde{S}(\cdot)$, as well as the serial nature of $H(\cdot)$. Finally, ASC achieves the lowest overall solution times up until n_P=64, combining some of the scaling qualities of both XSC and ISC. Again, the limiting factor for high n_P appears to be the serial preconditioner (a backsolve with the S^- in this case).

5. CONCLUSION

In this study, we introduced a general formulation for structured nonlinear optimization problems, and outlined a Schur complement decomposition approach to parallelize the application of the interior point method to solve such problems. Both explicit and implicit approaches to solve the Schur complement system are discussed, and for the latter, a preconditioner based on factorizations from previous interior point iterations was introduced. We compared these approaches on a parameter estimation problem defined over a 2D grid, and show that the explicit Schur complement approach spends the majority of time forming the Schur complement, but scales well with the number of problem partitions. The implicit approaches suffer from the fact that the number of complicating variables grows with the number of partitions, thus leading to limited parallel scaling of applying PCG. Still, the

proposed adaptive Schur complement approach is able to achieve lower absolute solution times than the explicit approach on the test problems considered here.

5.1 Future Work

Several factors which affect the parallel performance of the presented approaches, that were not discussed in this work, deserve further attention. The analysis performed here for problems defined over a 2D grid is likely to yield very different results for differently structured problems. The integration of methods to detect problem structure and determine an appropriate partitioning for different decomposition schemes automatically (Mitrai et al. (2022)) would improve the general applicability of the approaches discussed here. Furthermore, our analysis offered some indication that for the implicit schemes, the application of the preconditioner is likely to become a bottleneck for larger problems, as it cannot be applied in a distributed manner in its current form. For some problem structures, the use of preconditioners from the PDE literature, such as balanced domain decomposition (e.g. Mandel (1993)) appears to be a promising way to address this issue, especially in combination with more scalable methods to form the Schur complement for structured problems (Petra et al. (2014)).

ACKNOWLEDGEMENTS

Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC (NTESS), a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration (DOE/NNSA) under contract DE-NA0003525. This written work is authored by an employee of NTESS. The employee, not NTESS, owns the right, title and interest in and to the written work and is responsible for its contents. Any subjective views or opinions that might be expressed in the written work do not necessarily represent the views of the U.S. Government. The publisher acknowledges that the U.S. Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this written work or allow others to do so, for U.S. Government purposes. The DOE will provide public access to results of federally sponsored research in accordance with the DOE Public Access Plan.

This work used Bridges-2 at Pittsburgh Supercomputing Center through allocation MTH230029 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296.

REFERENCES

Allman, A., Tang, W., and Daoutidis, P. (2019). Decode: a community-based algorithm for generating high-quality decompositions of optimization problems. *Optimization and Engineering*, 20, 1067–1084.

Brown, S.T., Buitrago, P., Hanna, E., Sanielevici, S., Scibek, R., and Nystrom, N.A. (2021). Bridges-2: A platform for rapidly-evolving and data intensive research. In *Practice and Experience in Advanced Research Computing*, 1–4.

Byrd, R.H., Nocedal, J., and Schnabel, R.B. (1994). Representations of quasi-newton matrices and their use in limited memory methods. *Mathematical Programming*, 63(1-3), 129–156.

Cummings, D., Hart, W., Garcia-Carreras, B., Lanning, C., Lessler, J., and Staid, A. (2021). Spatio-temporal estimates of disease transmission parameters for covid-19 with a fully-coupled, county-level model of the united states. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States).

Duff, I.S. and Reid, J.K. (1983). The multifrontal solution of indefinite sparse symmetric linear. *ACM Transactions on Mathematical Software (TOMS)*, 9(3), 302–325.

Jalving, J., Shin, S., and Zavala, V.M. (2022). A graph-based modeling abstraction for optimization: Concepts and implementation in plasmo. jl. *Mathematical Programming Computation*, 14(4), 699–747.

Kang, J., Cao, Y., Word, D.P., and Laird, C.D. (2014). An interior-point method for efficient solution of block-structured nlp problems using an implicit schur-complement decomposition. *Computers & Chemical Engineering*, 71, 563–573.

Mandel, J. (1993). Balancing domain decomposition. *Communications in numerical methods in engineering*, 9(3), 233–241.

Mitrai, I., Tang, W., and Daoutidis, P. (2022). Stochastic blockmodeling for learning the structure of optimization problems. *AIChE Journal*, 68(6), e17415.

Morales, J.L. and Nocedal, J. (2000). Automatic preconditioning by limited memory quasi-newton updating. *SIAM Journal on Optimization*, 10(4), 1079–1096.

Morales, J.L. and Nocedal, J. (2001). Algorithm 809: Preqn: Fortran 77 subroutines for preconditioning the conjugate gradient method. *ACM Transactions on Mathematical Software (TOMS)*, 27(1), 83–91.

Nocedal, J. (1980). Updating quasi-newton matrices with limited storage. *Mathematics of computation*, 35(151), 773–782.

Petra, C.G., Schenk, O., Lubin, M., and Gärtner, K. (2014). An augmented incomplete factorization approach for computing the schur complement in stochastic optimization. *SIAM Journal on Scientific Computing*, 36(2), C139–C162.

Rodriguez, J.S., Parker, R.B., Laird, C.D., Nicholson, B.L., Siirola, J.D., and Bynum, M.L. (2023). Scalable parallel nonlinear optimization with pynumero and parapint. *INFORMS Journal on Computing*, 35(2), 509–517.

Saad, Y. (2003). *Iterative methods for sparse linear systems*. SIAM.

Wächter, A. and Biegler, L.T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106, 25–57.

Zavala, V.M., Laird, C.D., and Biegler, L.T. (2008). Interior-point decomposition approaches for parallel solution of large-scale nonlinear parameter estimation problems. *Chemical Engineering Science*, 63(19), 4834–4845.