HyKKT: A Hybrid Direct and Iterative Method for Solving KKT Linear Systems

Shaked Regev, Nai-Yuan Chiang, Eric Darve, Cosmin Petra, Michael A. Saunders, Kasia Świrydowicz, Slaven Peleš

Stanford University; Pacific Northwest National Laboratory; Lawrence Livermore National Laboratory

sregev@stanford.edu

August 3, 2022
Approved for unlimited distribution PNNL-SA-164807

Acknowledgements

This research was supported by the Exascale Computing Project (ECP), Project Number: 17-SC-20-SC, a collaborative effort of two DOE organizations—the Office of Science and the National Nuclear Security Administration—responsible for the planning and preparation of a capable exascale ecosystem—including software, applications, hardware, advanced system engineering, and early testbed platforms—to support the nation's exascale computing imperative.

ECP
EXASCALE COMPUTING PROJECT

General Sparse NLPs

Sparse NLP formulation supports sparse optimization problems, requires Hessians of objective and constraints in addition to gradients

$$
\begin{array}{lcll}
 & \min\limits_{x\in\mathbb{R}^n} & f(x) & \\
 & \text{s.t.} & c(x) = c_E & [y] \\
[y_{d,l}] & & d_l \le d(x) \le d_u & [y_{d,u}] \\
[z_l] & & x_l \le x \le x_u & [z_u]
\end{array}
$$

- Assume gradients and sparse Hessians are available
- Quantities insides brackets are Lagrange multipliers for the constraints
- For infinite bounds, multiplier is 0

Model Requirements

D1 objective and constraint functions $f(x)$, $c(x)$, $d(x)$

D2 first derivatives: $\nabla f(x)$, $J_c(x) = \nabla c(x)$, $J_d(x) = \nabla d(x)$

D3 Hessian of the Lagrangian

$$\nabla^2 L(x) = \nabla^2 f(x) + \sum_i y_{c,i} \nabla^2 c_i(x) + \sum_i y_{d,i} \nabla^2 d_i(x)$$

D4 simple bounds x_l and x_u, inequality bounds d_l and d_u, and rhs c_E of equality constraints

Motivation

- Out of the box GPU solvers do not work well on these problems
- KLU + cuSolver works but is proprietary, only works on NVIDIA GPUs
- Want a solver that allows substantial speedup on GPUs
- Using a Cholesky solver instead of LBL^T would allow parallelization and GPU utilization, but the problem is indefinite

Test case	Size	NNZ	MA57 reference CPU only	SuperLU (ECP – LBNL)		STRUMPACK (ECP – LBNL)		KLU + cuSolver (NVIDIA)		SSIDS (STFC, UK Gov.)	
				CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU
10k-bus	238,072	1,111,991	0.54s	4.06s	4.95s	2.82s	3.71s	0.81s	0.25s*	2.40s	4.76s
70k-bus	1,640,411	7,671,693	5.30s	30.46s	35.58s	24.4s	26.8s	13.26s	3.26s*	32.25s	197.66s

* *Total computational cost of symbolic factorization amortized over 5 solves, i.e. assumed 1 symbolic factorization can be reused over 5 optimization solver iterations.*

K. Świrydowicz et al. (2021) Linear solvers for power grid optimization problems: a review of GPU-accelerated linear solvers

Problem Setup

Interior method, used to solve KKT systems, generates series of linear systems $K_k \Delta x_k = r_k$:

$$\begin{bmatrix} H + D_x & 0 & J_c^T & J_d^T \\ 0 & D_s & 0 & -I \\ J_c & 0 & 0 & 0 \\ J_d & -I & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta s \\ \Delta y \\ \Delta y_d \end{bmatrix} = \begin{bmatrix} \tilde{r}_x \\ r_s \\ r_{yc} \\ r_{yd} \end{bmatrix}$$

- J_c and J_d - sparse Jacobians for equality and inequality constraints
- H - sparse Hessian matrix
- Diagonal D_x arises from primal variables x in log-barrier function
- Diagonal D_s arises from slack variables s in log-barrier function

Simplifying the Problem

Eliminating $\Delta s = J_d \Delta x - r_{yd}$ and $\Delta y_d = D_s \Delta s - r_d$ gives

$$\begin{bmatrix} \widetilde{H} & J_c^T \\ J_c & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} r_x \\ r_{yc} \end{bmatrix}, \qquad \widetilde{H} \equiv H + D_x + J_d^T D_s J_d,$$

where $r_x = \tilde{r}_x + J_d^T (D_s r_{yd} + r_d)$. Gaussian elimination with pivot $\begin{bmatrix} D_s & -I \\ -I & \end{bmatrix}$.

- We need to perform Ruiz Scaling on $\widetilde{H}$ and J_c so we can judge the sizes of the entries in the blocks

C. Petra et al. (2009) A computational study of the use of an optimization-based method for simulating large multibody systems
D. Ruiz (2001) A scaling algorithm to equilibrate both rows and columns norms in matrices

Schur Complement System

The KKT system is equivalent to

$$\begin{bmatrix} H_\gamma & J_c^T \\ J_c & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \widehat{r}_x \\ r_{yc} \end{bmatrix}, \qquad H_\gamma = \widetilde{H} + \gamma J_c^T J_c, \qquad \widehat{r}_x = r_x + \gamma J_c^T r_{yc}$$

- $\gamma > 0$ makes the system more SPD (increases the eigenvalues)
- If H_γ or whole system are poorly conditioned, only option may be to ignore the block structure and use an LBL^T factorization
- Sparse Cholesky on H_γ and CG on its Schur complement S:

$$S\Delta y = J_c H_\gamma^{-1} \widehat{r}_x - r_{yc}, \qquad S = J_c H_\gamma^{-1} J_c^T$$

$$H_\gamma \Delta x = \widehat{r}_x - J_c^T \Delta y$$

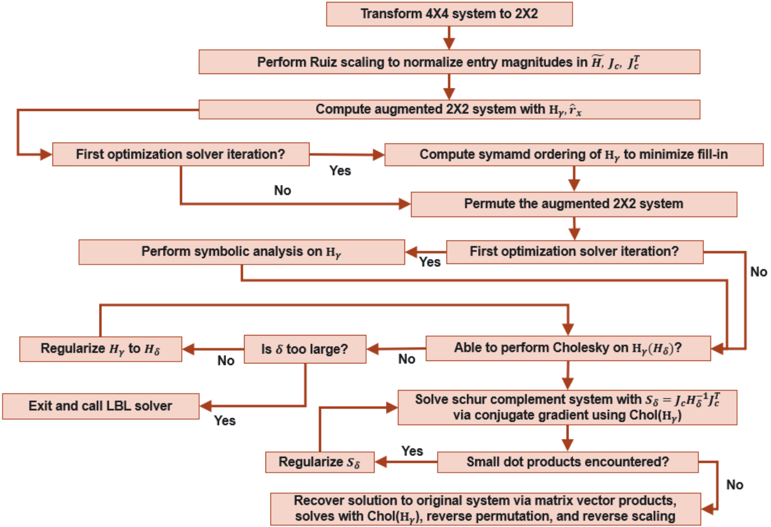
G. H. Golub and C. Greif (2003) On solving block-structured indefinite linear systems

Convergence Theorems

Reminder: $\widetilde{H} \equiv H + D_x + J_d^T D_s J_d, \qquad H_\gamma = \widetilde{H} + \gamma J_c^T J_c$

- For large γ and full-rank J_c, $\widetilde{H}$ is PD on null(J_c) iff H_γ is uniformly PD (required at optimization problem solution).
- HyKKT provides descent direction to interior method (for large γ)
- If $\widetilde{H}$ is positive definite on null$(J_c^T J_c)$, $\exists\gamma_{\text{max}}$ such that for $\gamma \geq \gamma_{\text{max}}$, $\kappa(H_\gamma)$ increases linearly with γ.
- For large enough γ and $C \equiv 1/\gamma \left(J_c \widetilde{H}^{-1} J_c^T\right)^{-1}$, $S_\gamma \equiv \gamma S = \gamma J_c H_\gamma^{-1} J_c^T = \sum_{k=0}^{\infty} (-1)^k C^k = I - C + O\left(\frac{1}{\gamma^2}\right) = I + O\left(\frac{1}{\gamma}\right)$.

HyKKT Workflow



Preliminary Results With Solver Prototype

- **RR** for $Ax = b$: $\frac{\|Ax-b\|}{\|b\|}$
- **BE**: $\frac{\|Ax-b\|}{\|A\|\|x\|+\|b\|}$, with $\|A\|_\infty \approx \|A\|$
- 5/6 matrix series (NLPs at different iterations of interior method) solved efficiently and accurately (other needed $O(1)$ regularization)
- BE consistently $< 10^{-8}$
- Average CG iterations < 20.
- δ_{min} in the range 10^{-8} down to 10^{-10} is reasonable for any $\gamma \leq 10^8$
- (1) refers to the 4×4 system, (2) refers to the 2×2 system

Jonathan Maack and Shrirang Abhyankar (2020) ACOPF sparse linear solver test suite, https://github.com/NREL/opf_matrices

US Eastern Interconnection Grid

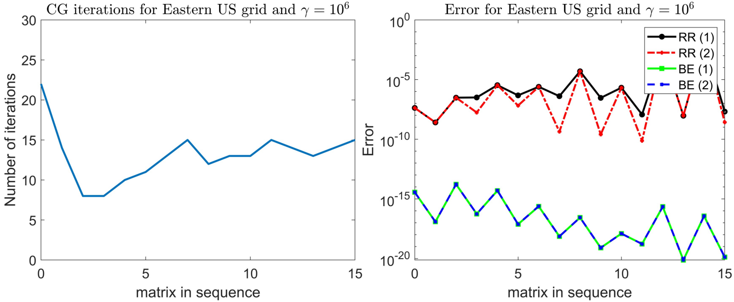


Figure: (Left) CG iterations on S with $\gamma = 10^6$. Mean number of iterations is 13.1. (Right) Various errors for $\gamma = 10^6$. BE $< 10^{-14}$.

Comparison with LBL^T: Factorization Density

Table: Dimensions, number of nonzeros, and factorization densities (average number of nonzeros in the factors per row) for solving full system directly with LBL^Tvia MA57 with pivot tolerance 0.01 (n_L, nnz_L, ρ_L) and solving systems with H_γ with Cholesky (n_C, nnz_C, ρ_C)

Abbreviation	n_L	nnz_L	ρ_L	n_C	nnz_C	ρ_C
Illinois	4.64K	94.7K	**20.4**	2.28K	34.9K	**15.3**
Texas	55.7K	2.95M	**52.9**	25.9K	645K	**24.9**
Western US	238K	10.7M	**44.8**	116K	2.23M	**19.2**
Eastern US	1.64M	85.4M	**52.1**	794K	17.7M	**22.3**

Comparison with LBL^T: Run Time (Preliminary)

Table: Times (in seconds) for solving full system directly on a CPU with LBL^T(via MA57) or HyKKT on a GPU. CG is solved to tolerance of 10^{-12}. All runs are on x86_64 CPUs and A100 GPUs. As the problems grow larger, HyKKT outperforms MA57 by an increasing factor.

Abbreviation	MA57	HyKKT	Relative size	**MA57/HyKKT**
Illinois	$6.24 \cdot 10^{-3}$	$1.01 \cdot 10^{-2}$	1	**0.62**
Texas	$1.00 \cdot 10^{-1}$	$1.04 \cdot 10^{-1}$	10	**1.04**
Western US	$3.38 \cdot 10^{-1}$	$1.46 \cdot 10^{-1}$	50	**2.32**
Eastern US	$3.48 \cdot 10^{0}$	$3.31 \cdot 10^{-1}$	350	**10.5**

Duff (2004)
Chen, Davis, Hager, and Rajamanickam (2008)

Summary

- We designed a linear solver strategy suitable for fine-grain parallelization and deployment on hardware accelerators
- We prove fast CG, holds in practice
- The iterative nature of the solver provides more flexibility to balance trade-offs between accuracy and performance
- Cholesky instead of LBL^T allows for better GPU utilization
- Non-optimized HyKKT outperforms `MA57` by $10x$ on largest problems tested