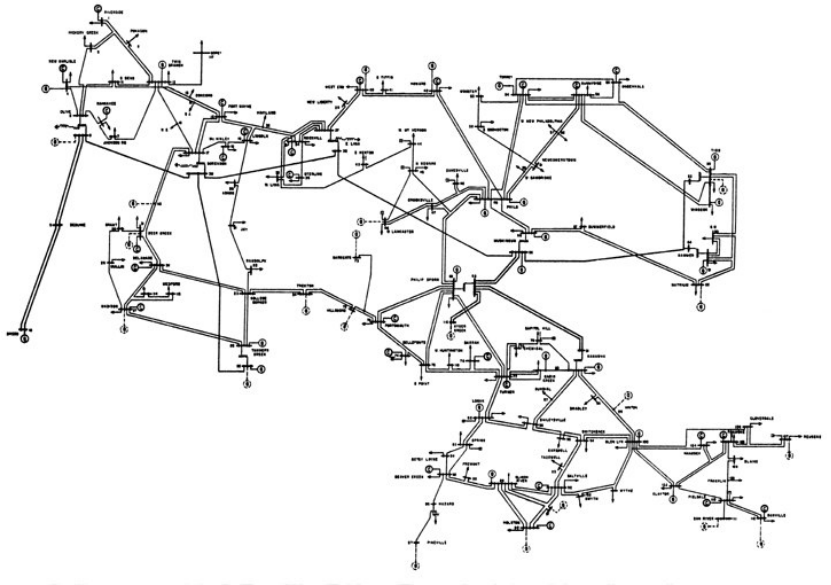
Reduced-space Interior-Point Method: A GPU accelerated Comeback

François Pacaud Sungho Shin Michel Schanen Daniel Adrian Maldonado
Mihai Anitescu

ICCOPT 2022

Motivation: solving optimal power flow problems on GPU architectures

Our research is funded by the Exascale Computing Project (ECP)



Exascale challenge

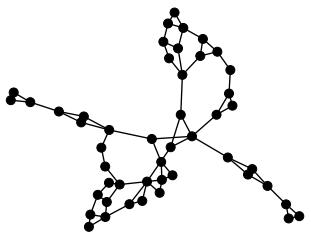
Handling unstructured sparsity on SIMD architectures is non trivial

Hardware

GPU centric (SIMD)

Physical model

unstructured



Why GPUs are hard for optimizers?

Observation

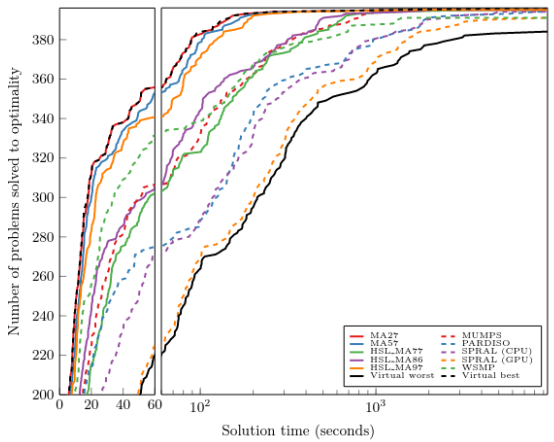
- GPUs are SIMD architecture (single instruction, multiple data)
- Excellent for *dense* and *batch* operations

On their hand, numerical optimization algorithms depend on two key routines

1. **Derivatives:** Explicit derivatives, Finite Differences, Automatic Differentiation
2. **Linear solve:** Solve KKT system to compute descent direction

$$(\nabla^2_{xx}\ell_k)d_k = -\nabla_x \ell_k$$

where $(\nabla^2_{xx}\ell_k)$ is sparse symmetric indefinite



(from (Tasseff et al., 2019))

- No good sparse symmetric indefinite solver on GPU
- Usual workarounds:
  1. Use decomposition algorithms (ADMM, Kim et al. (2021))
  2. Use iterative solver (CG-based) (Cao et al., 2016; Schubiger et al., 2020)

Our solution: densification

Idea: Exploit the available degrees of freedom

Densify the problem using the reduced Hessian

$$\hat{H}_{uu} = Z^\top H Z$$

- Approach widely used during the 1980s/1990s
  - Summarized in (Fletcher, 1987, Section 12.5): "Nonlinear elimination and feasible direction methods"
  - Also known as "Projected Hessian" (Nocedal and Overton, 1985; Gurwitz and Overton, 1989)
  - The reduced Hessian $\hat{H}_{uu}$ is often approximated (Biegler et al., 1995)
- The optimization community moved away from this technique in the 2000s:
  - "Many degrees of freedom" approaches (Poku et al., 2004)
  - Efficient resolution with interior-point combined with generic indefenite sparse linear solver (HSL (Duff and Reid, 1983), Pardiso (Schenk and Gärtner, 2004))
  - Lead to the development of mature NLP solvers (Wächter and Biegler, 2006; Waltz et al., 2006)

Take-home messages

1. We parallelize the evaluation of the reduced Hessian $\hat{H}_{uu}$ on the GPU
2. We exploit the reduced Hessian inside an interior-point method
3. Performance of the method depends on available degrees of freedom (the less, the better), but tractable overall

We applied this method to solve the OPF problem on GPU (Pacaud et al., 2022):

	IPM + Full-space Hessian (CPU)	IPM + Reduced Hessian (GPU)
9241pegase (many DOF)	**10.7s**	**23.7s**
9591goc (few DOF)	**11.7s**	**7.7s**

Magic happens when we exploit the structure

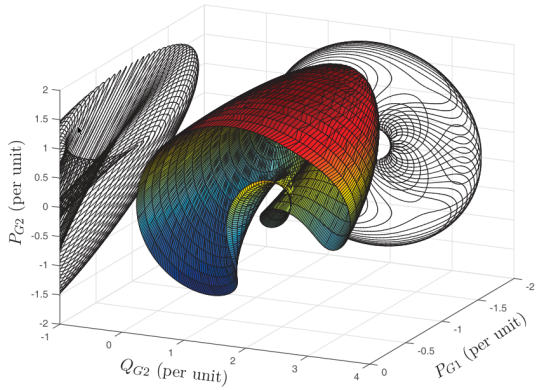


Figure: Nonlinear power flow (from Hiskens and Davy (2001))

Most real-life nonlinear problems encompasses a set of physical constraints

$$g(\boldsymbol{x}, \boldsymbol{u}) = 0$$

with $\boldsymbol{x}$ a state and $\boldsymbol{u}$ a control

Domain	g
Optimal control	Dynamics
PDE-constrained optimization	PDE
Optimal power flow	Power flow

Physically-constrained optimization problem

$$\min_{x,u} f(\boldsymbol{x}, \boldsymbol{u})$$

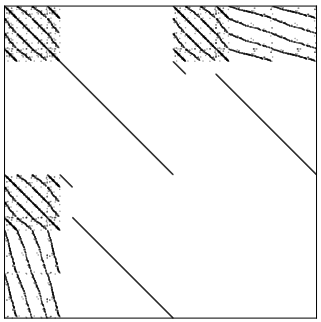
$$\text{s.t. } g(\boldsymbol{x}, \boldsymbol{u}) = 0 \,, \quad h(\boldsymbol{x}, \boldsymbol{u}) \leq 0$$

Well-known method (Cervantes et al., 2000; Biros and Ghattas, 2005)

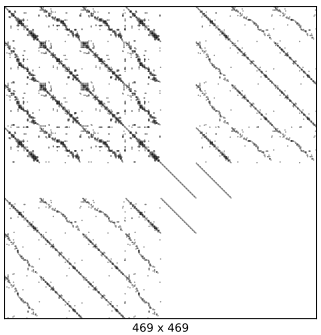
Condense step: we remove the inequalities

Notations

- W: Hessian of Lagrangian
- G: Jacobian of equality constraints (power flow)
- A: Jacobian of inequalities (operational constraints)



$\Downarrow$



In interior-point (IPM), the augmented KKT system (*symmetric*) writes

$$\begin{bmatrix} W + \Sigma_p & 0 & G^\top & A^\top \\ 0 & \Sigma_s & 0 & -I \\ G & 0 & 0 & 0 \\ A & -I & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_v \\ \boldsymbol{p}_s \\ \boldsymbol{p}_\lambda \\ \boldsymbol{p}_y \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \\ \boldsymbol{r}_3 \\ \boldsymbol{r}_4 \end{bmatrix}$$

We condense by taking the Schur-complement w.r.t. the inequalities block

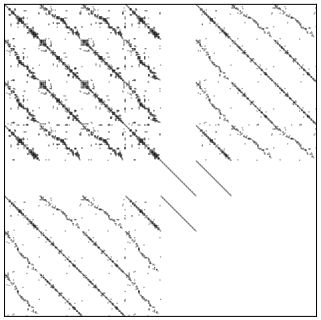
Condensed KKT

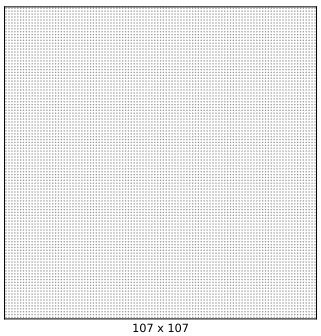
Let $K := W + A^\top \Sigma_s A$. The KKT system is equivalent to

$$\begin{bmatrix} K + \Sigma_p & G^\top \\ G & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_p \\ \boldsymbol{p}_\lambda \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_1 + A^\top (\Sigma_s \boldsymbol{r}_4 + \boldsymbol{r}_2) \\ \boldsymbol{r}_3 \end{bmatrix}$$

Usually discarded because of additional fill-in in left-hand-side matrix, but here we are densifying the KKT system

Reduce step: we remove the equalities





Exploiting the structure of $g(\boldsymbol{x}, \boldsymbol{u}) = 0$:

$$\begin{bmatrix} K_{xx} + \Sigma_x & K_{xu} & G_x^\top \\ K_{ux} & K_{uu} + \Sigma_u & G_u^\top \\ G_x & G_u & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_x \\ \boldsymbol{p}_u \\ \boldsymbol{p}_\lambda \end{bmatrix} = \begin{bmatrix} \widehat{\boldsymbol{r}}_1 \\ \widehat{\boldsymbol{r}}_2 \\ \widehat{\boldsymbol{r}}_3 \end{bmatrix}$$

Reduced KKT

If the Jacobian G_x is invertible, then the KKT system is equivalent to

$$\hat{K}_{uu}\, \boldsymbol{p}_u = \hat{\boldsymbol{r}}_2 - G_u G_x^{-1} \hat{\boldsymbol{r}}_1 - K_{ux} G_x^{-1} \hat{\boldsymbol{r}}_3 \quad \text{with} \quad \hat{K}_{uu} := Z^\top K Z$$

where we have defined the reduction operator

$$Z := \begin{bmatrix} -G_x^{-1} G_u \\ I \end{bmatrix}$$

- The matrix $\hat{K}_{uu}$, dense, can be factorized efficiently on the GPU with dense Cholesky (supposing regularization applied)
- Assembling $\hat{K}_{uu}$ requires only the factorization of the sparse Jacobian G_x

Implementing the reduction on the GPU

We suppose given the sparse matrix $K = W + A^\top \Sigma_s A$

$$\hat{K}_{uu} = \begin{bmatrix} -G_x^{-1} G_u \\ I \end{bmatrix}^\top \begin{bmatrix} K_{xx} & K_{xu} \\ K_{ux} & K_{uu} \end{bmatrix} \begin{bmatrix} -G_x^{-1} G_u \\ I \end{bmatrix}$$

We should avoid allocating the sensitivity matrix $S = -G_u G_x^{-1}$ (dense, size $n_x \times n_u$)!
Instead, use batched `HessMat` product $\hat{K}_{uu} V$

`HessMat` kernel: batch adjoint-adjoint reduction

Input: LU factorization, such that $P G_x Q = L\,U$ (2 `SpMM`, 2 `SpSM`)

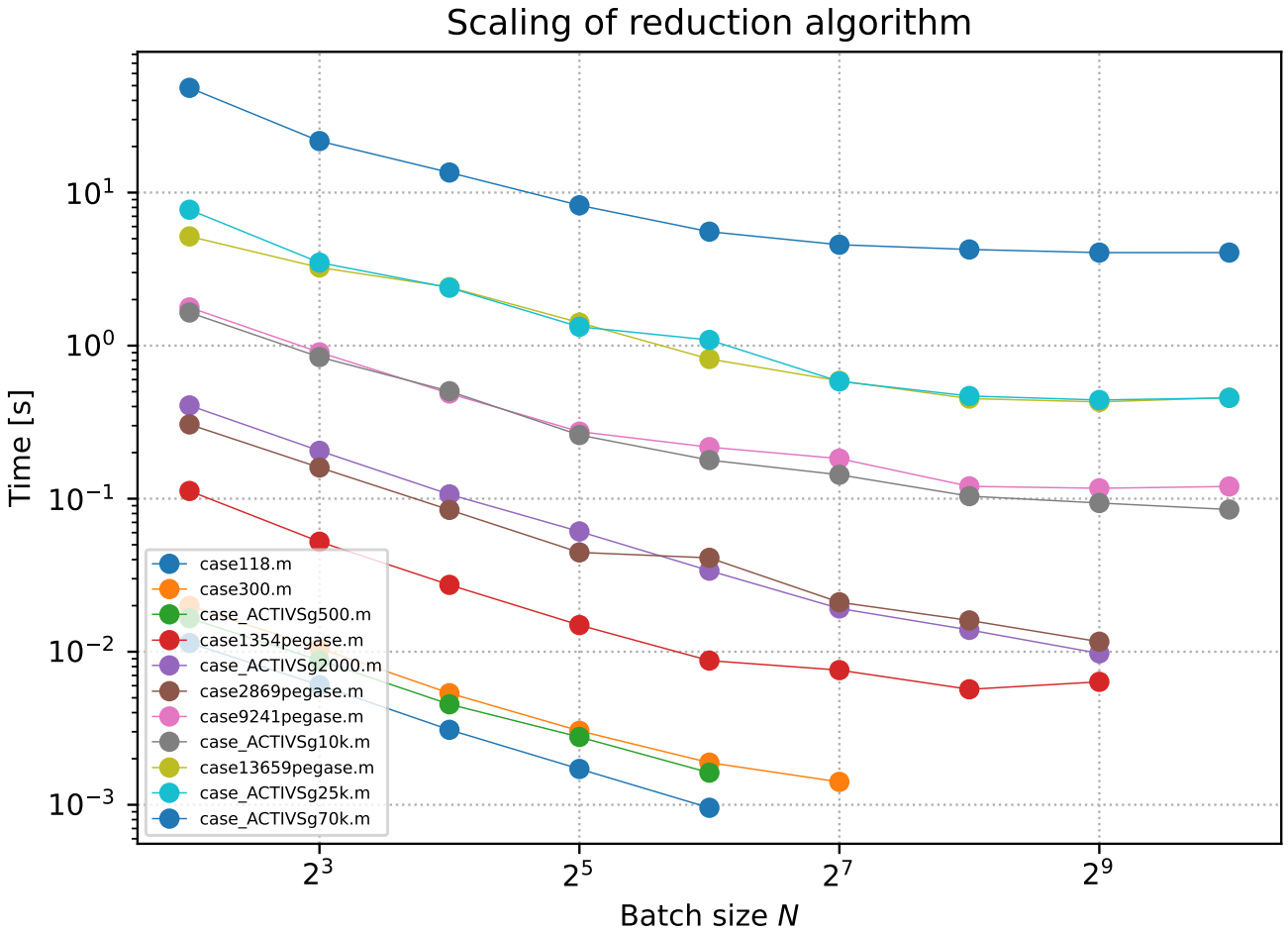
For every RHS $V \in \mathbb{R}^{n_u \times N}$

1. Solve $Z = G_x^{-1}(G_u V)$ (3 `SpMM`, 2 `SpSM`)
2. Evaluate $\begin{bmatrix} \Psi \\ H_u \end{bmatrix} = \begin{bmatrix} K_{xx} & K_{xu} \\ K_{ux} & K_{uu} \end{bmatrix} \begin{bmatrix} Z \\ V \end{bmatrix}$ (1 `SpMM`)
3. Solve $H_x = G_x^{-\top} \Psi$ (2 `SpMM`, 2 `SpSM`)
4. Output $\hat{K}_{uu} V = H_u - G_u H_x$ (1 `SpMM`)

- G_x first factorized on the CPU with KLU,
  then refactorized entirely on the GPU with `cusolverRF` (fast)
- $div(n_u, N) + 1$ `HessMat` products required to get full $\hat{K}_{uu}$

Performance of the reduction algorithm

Message: 7 seconds to reduce the matrix for the largest instance (ACTIVSg70k)



MadNLP: a GPU-ready IPM solver

MadNLP (Shin et al., 2020)

- Filter line-search interior-point method
- Implemented purely in Julia
- Open-source: `https://github.com/MadNLP/MadNLP.jl`

- **Derivatives:**
  - Custom automatic-differentation backend: ExaPF.jl
  - Derivatives evaluated in parallel on the GPU
- **Linear solve:** We compare two equivalent alternatives to solve the KKT system
  1. *The reference:* HSL MA27 running on the CPU
  2. *The contender:* our reduction algorithm, using `cusolver` to factorize the reduced matrix with dense Cholesky on the GPU

Coming back to the OPF problem

Observation

The smaller the number of degrees of freedom n_u, the more efficient is the reduction of the KKT system

		The reference MadNLP+MA27			*The contender* MadNLP+reduced KKT			
Case	DOF	#it	Time (s)	MA27 (s)	#it	Time (s)	Chol. (s)	Reduction (s)
		Many degrees of freedom						
9241pegase	0.14	69	**10.6**	6.1	69	**23.7**	1.2	16.2
ACTIVSg25k	0.10	86	**24.7**	16.9	86	**85.0**	4.3	68.1
ACTIVSg70k	0.08	90†	**89.8**	65.7	85†	**658.2**	21.5	606.5
		Few degrees of freedom						
9591goc	0.02	43	**11.7**	10.4	43	**7.7**	2.1	1.6
10480goc	0.03	50	**14.0**	12.0	50	**11.5**	3.9	3.3
19402goc	0.02	47	**30.8**	26.8	47	**19.5**	4.9	7.2

Legend:
†: algorithm runs into feasibility restoration

When is reduced better than full-space?

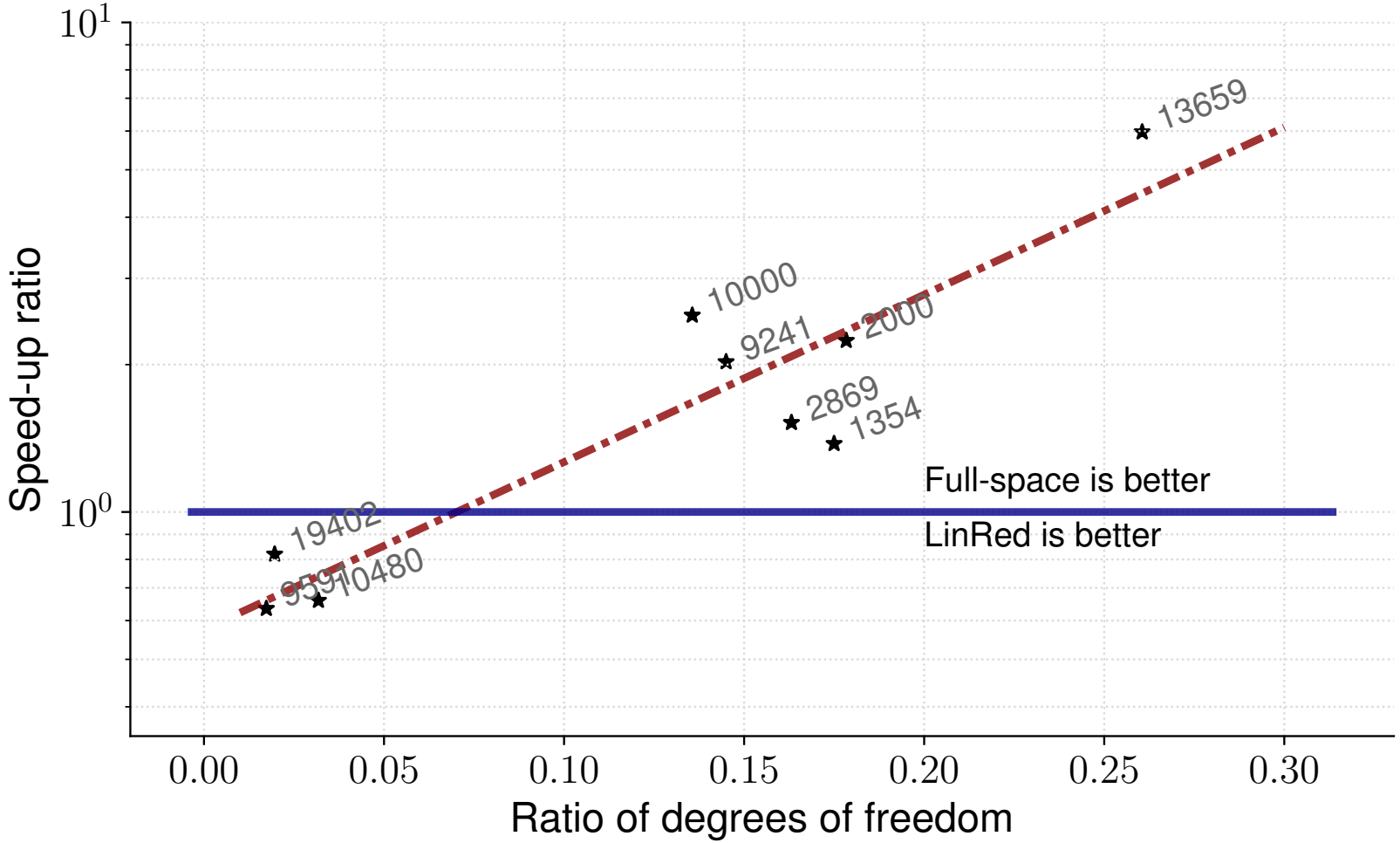


Figure: Breakaven point

Extension

Block-structured optimization problem

$$\min_{x_1,\cdots,x_N,u} \quad f(x_1,\cdots,x_N,u)$$

$$\text{s.t. } g(x_i,u)=0\,, \quad h(x_i,u)\leq 0 \qquad \forall i=1,\cdots,N$$

Stochastic optimization, structural design, ...

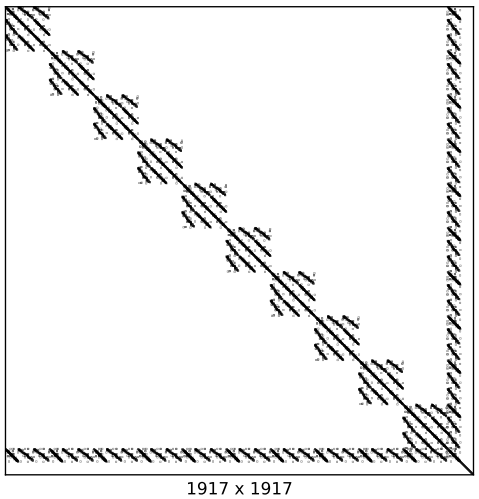


Figure: Block arrowhead Hessian

- Reduction is equivalent to PIPS-NLP's Schur-complement approach
- Possible resolution on multiple GPUs (Frontier, Aurora)

References I

Biegler, L. T., Nocedal, J., and Schmid, C. (1995). A reduced Hessian method for large-scale constrained optimization. SIAM Journal on Optimization, 5(2):314–347.

Biros, G. and Ghattas, O. (2005). Parallel Lagrange–Newton–Krylov–Schur Methods for PDE-Constrained Optimization. Part I: The Krylov–Schur Solver. SIAM Journal on Scientific Computing, 27(2):687–713.

Cao, Y., Seth, A., and Laird, C. D. (2016). An augmented Lagrangian interior-point approach for large-scale NLP problems on graphics processing units. Computers & Chemical Engineering, 85:76–83.

Cervantes, A. M., Wächter, A., Tütüncü, R. H., and Biegler, L. T. (2000). A reduced space interior point strategy for optimization of differential algebraic systems. Computers & Chemical Engineering, 24(1):39–51.

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.

Fletcher, R. (1987). Practical methods of optimization. John Wiley & Sons.

Gurwitz, C. B. and Overton, M. L. (1989). Sequential quadratic programming methods based on approximating a projected Hessian matrix. SIAM Journal on Scientific and Statistical Computing, 10(4):631–653.

Hiskens, I. A. and Davy, R. J. (2001). Exploring the power flow solution space boundary. IEEE transactions on power systems, 16(3):389–395.

Kim, Y., Pacaud, F., Kim, K., and Anitescu, M. (2021). Leveraging gpu batching for scalable nonlinear programming through massive lagrangian decomposition. arXiv preprint arXiv:2106.14995.

Nocedal, J. and Overton, M. L. (1985). Projected Hessian updating algorithms for nonlinearly constrained optimization. SIAM Journal on Numerical Analysis, 22(5):821–850.

Pacaud, F., Shin, S., Schanen, M., Maldonado, D. A., and Anitescu, M. (2022). Condensed interior-point methods: porting reduced-space approaches on gpu hardware. arXiv preprint arXiv:2203.11875.

Poku, M. Y. B., Biegler, L. T., and Kelly, J. D. (2004). Nonlinear optimization with many degrees of freedom in process engineering. Industrial & engineering chemistry research, 43(21):6803–6812.

Schenk, O. and Gärtner, K. (2004). Solving unsymmetric sparse systems of linear equations with pardiso. Future Generation Computer Systems, 20(3):475–487.

Schubiger, M., Banjac, G., and Lygeros, J. (2020). GPU acceleration of ADMM for large-scale quadratic programming. Journal of Parallel and Distributed Computing, 144:55–67.

References II

Shin, S., Coffrin, C., Sundar, K., and Zavala, V. M. (2020). Graph-based modeling and decomposition of energy infrastructures. arXiv preprint arXiv:2010.02404.

Tasseff, B., Coffrin, C., Wächter, A., and Laird, C. (2019). Exploring benefits of linear solver parallelism on modern nonlinear optimization applications. arXiv preprint arXiv:1909.08104.

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(1):25–57.

Waltz, R. A., Morales, J. L., Nocedal, J., and Orban, D. (2006). An interior algorithm for nonlinear optimization that combines line search and trust region steps. Mathematical Programming, 107(3):391–408.