Large-Scale Convex and Nonconvex Optimization in Data Science

IMS, CMAI and Math Department, CUHK

October 14, 2022

Yinyu Ye
Stanford University and CUHKSZ (Sabbatical Leave)
(Currently Visiting SEEM of CUHK ...)

Today’s Talk

- **New developments of ADMM-based interior point (ABIP) Method**
- **Optimal Diagonal Preconditioner and HDSDP**
- **A Dimension Reduced Second-Order Method**
- **SOLNP: a Derivative-Free Optimization Solver**

Introduction to ADMM

- Consider the following convex optimization problem

$$\min \ f(\boldsymbol{x})$$
$$\text{s.t.} \quad \boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$$
$$\boldsymbol{x} \in \boldsymbol{X}$$

- Where f is a convex function, and $\boldsymbol{X}$ the Cartesian product of possibly non-convex, real, closed, nonempty sets.

- The corresponding augmented Lagrangian function is

$$L(\boldsymbol{x}, \boldsymbol{\lambda})_{\boldsymbol{x} \in \boldsymbol{X}} = f(\boldsymbol{x}) - \boldsymbol{\lambda}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) + \frac{\rho}{2}||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}||_2^2$$

where $\boldsymbol{\lambda}$ is the Lagrangian multipliers or dual variables, and $\rho > 0$ is the step size.

Two-block ADMM with separable objectives

- Consider the following optimization problem

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) + g(\boldsymbol{s}) \\ \text{s.t.} \quad & \boldsymbol{A}\,\boldsymbol{x} + \boldsymbol{B}\boldsymbol{s} = \boldsymbol{b} \\ & \boldsymbol{x} \in \boldsymbol{X},\ \boldsymbol{s} \in \boldsymbol{S} \end{aligned}$$

- The corresponding augmented Lagrangian function is

$$L(\boldsymbol{x},\ \boldsymbol{s},\ \boldsymbol{\lambda})_{\boldsymbol{x}\in\boldsymbol{X},\,\boldsymbol{s}\in\boldsymbol{S}} = f(\boldsymbol{x}) + g(\boldsymbol{s}) - \boldsymbol{\lambda}^T(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{s} - \boldsymbol{b}) + \frac{\rho}{2}||\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{s} - \boldsymbol{b}||_2^2$$

$$\text{ADMM} = \begin{cases} x^{k+1} & = \arg\min_{x\in\mathcal{X}} \mathcal{L}_\beta(x, s^k; \lambda^k) \\ s^{k+1} & = \arg\min_{s\in\mathcal{X}} \mathcal{L}_\beta(x^{k+1}, s^{k+1}; \lambda^k) \\ \lambda^{k+1} & = \lambda^k - \beta(A[x; s]^{k+1} - b) \end{cases}$$

- The two-block ADMM with separable objective is guarantee to converge.

Multi-Block Cyclic ADMM algorithm

- We could also consider multiple blocks. Let $\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_b]$ if they have separable objectives

- Direct extension of multi-block (cyclic) ADMM updates as follows

$$\text{ADMM} = \begin{cases} x_1^{k+1} & = \arg\min_{x_1 \in \mathcal{X}} \mathcal{L}_\beta(x_1, \ldots, x_b^k; \lambda^k) \\ \ldots & \\ x_b^{k+1} & = \arg\min_{x_l \in \mathcal{X}} \mathcal{L}_\beta(x_1^{k+1}, \ldots, x_b; \lambda^k) \\ & \\ \lambda^{k+1} & = \lambda^k - \beta(Ax^{k+1} - b) \end{cases}$$

Direct extension of three-block ADMM does not converge

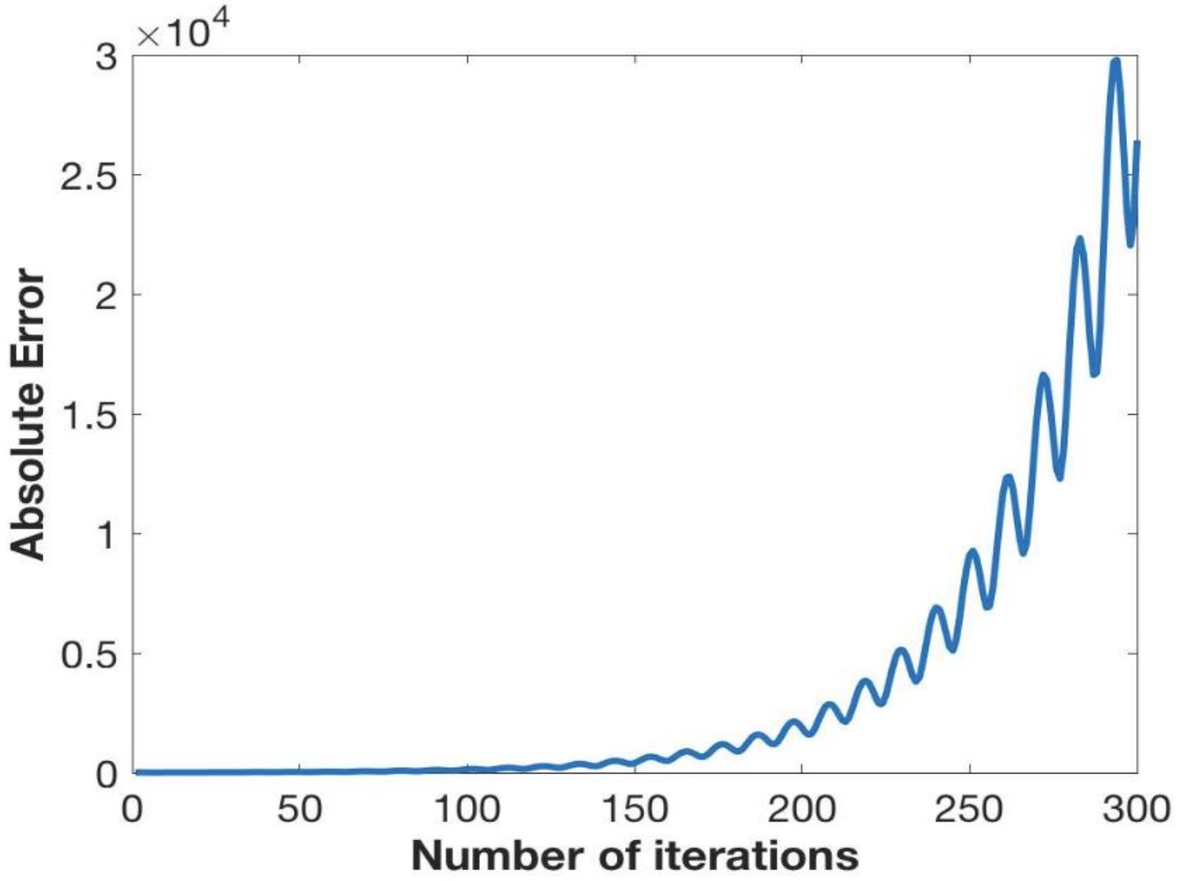


Figure 1: Non-singular system of square equations, three blocks (Chen et al. 2016)

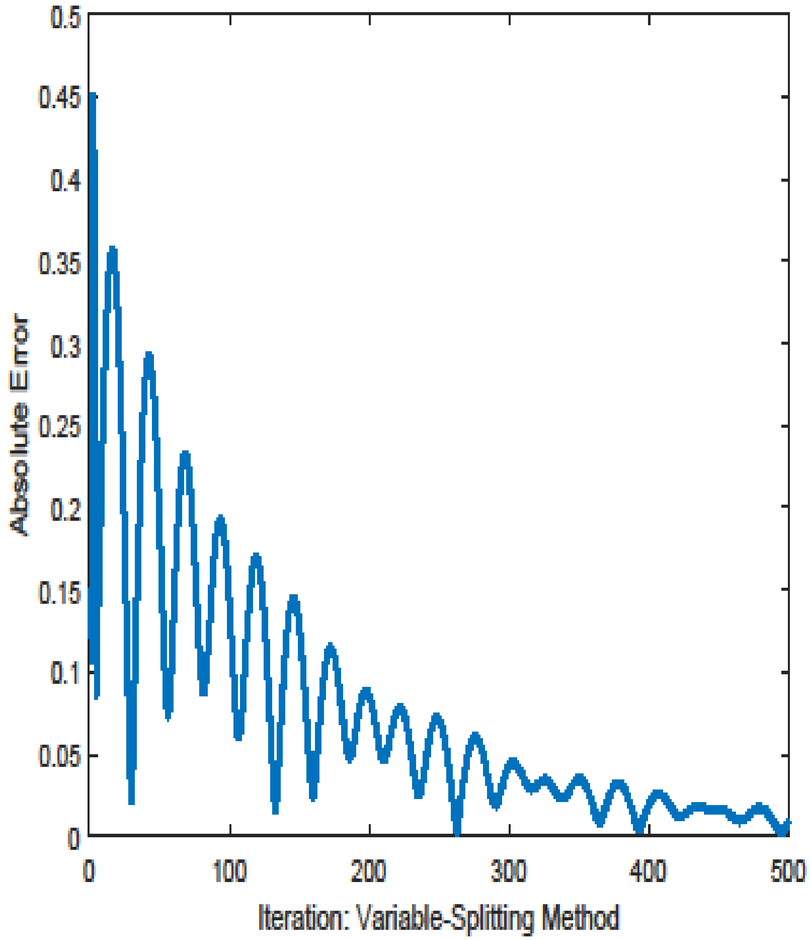
Randomly Permuted - ADMM (RP - ADMM) (Sun et al. 2016, Chen et al. 2019)

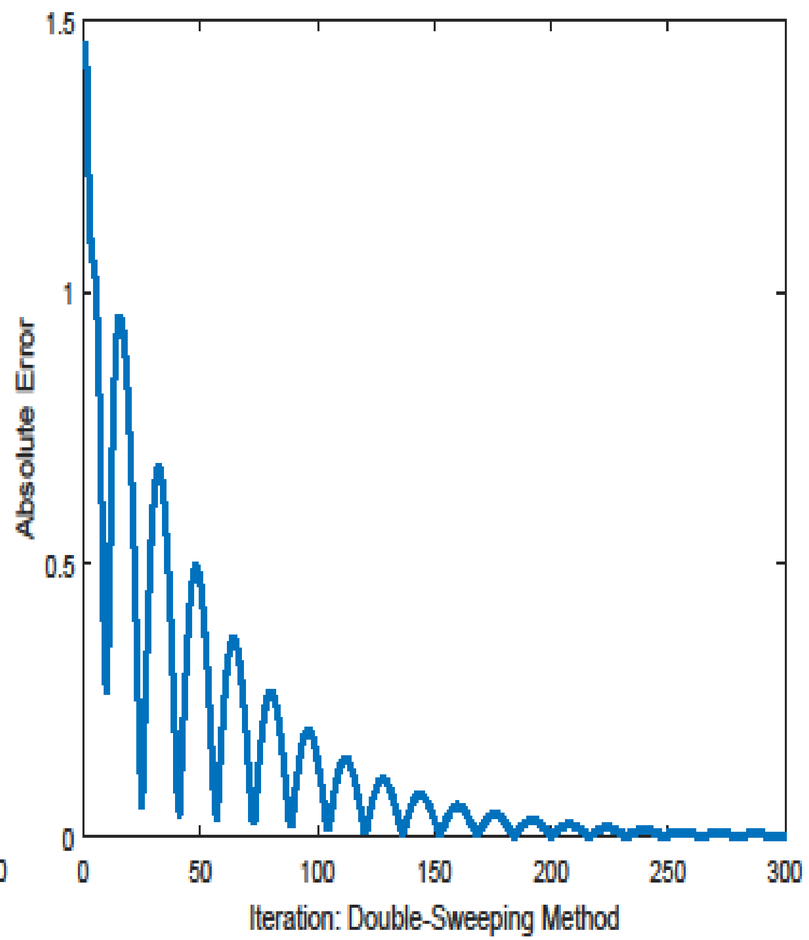
In each cycle of ADMM

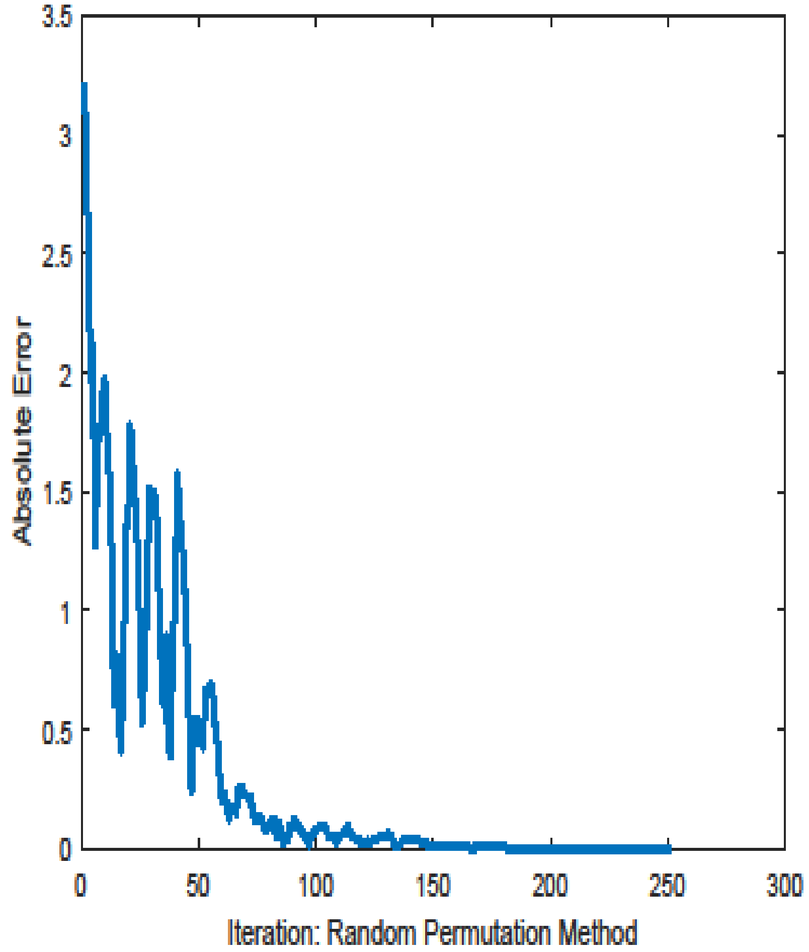
- **Randomly generate a permutation** of 1,…,b; and following the permutation order to update $\boldsymbol{x}_i$ sequentially.
- Update the multipliers the same way.
- RP - ADMM is guaranteed to converge in expectation.

Other methods need additional cost to overcome the divergence!

Performance on the diverging example







ADMM Based Interior-Point (ABIP) Method (Lin et al 2021, Deng et al. 2022)

- Different strategies/parameters may be significantly different among problems being solved
- An integration strategy based on decision tree is integrated into ABIP

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

Dimension

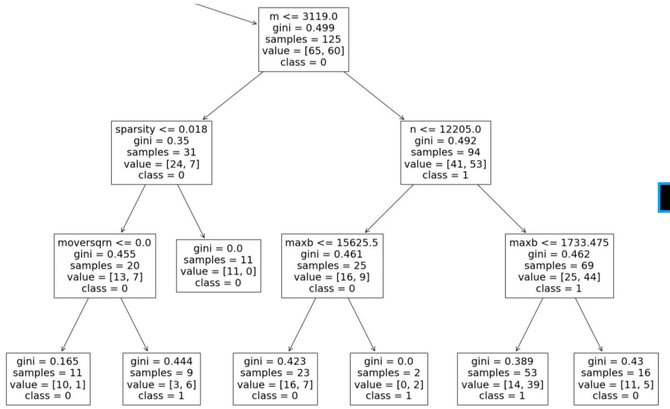
Sparsity

Constraint

Coefficient

Null Objective

…



Strategy 1 (restart)

Strategy 2 (scaling)

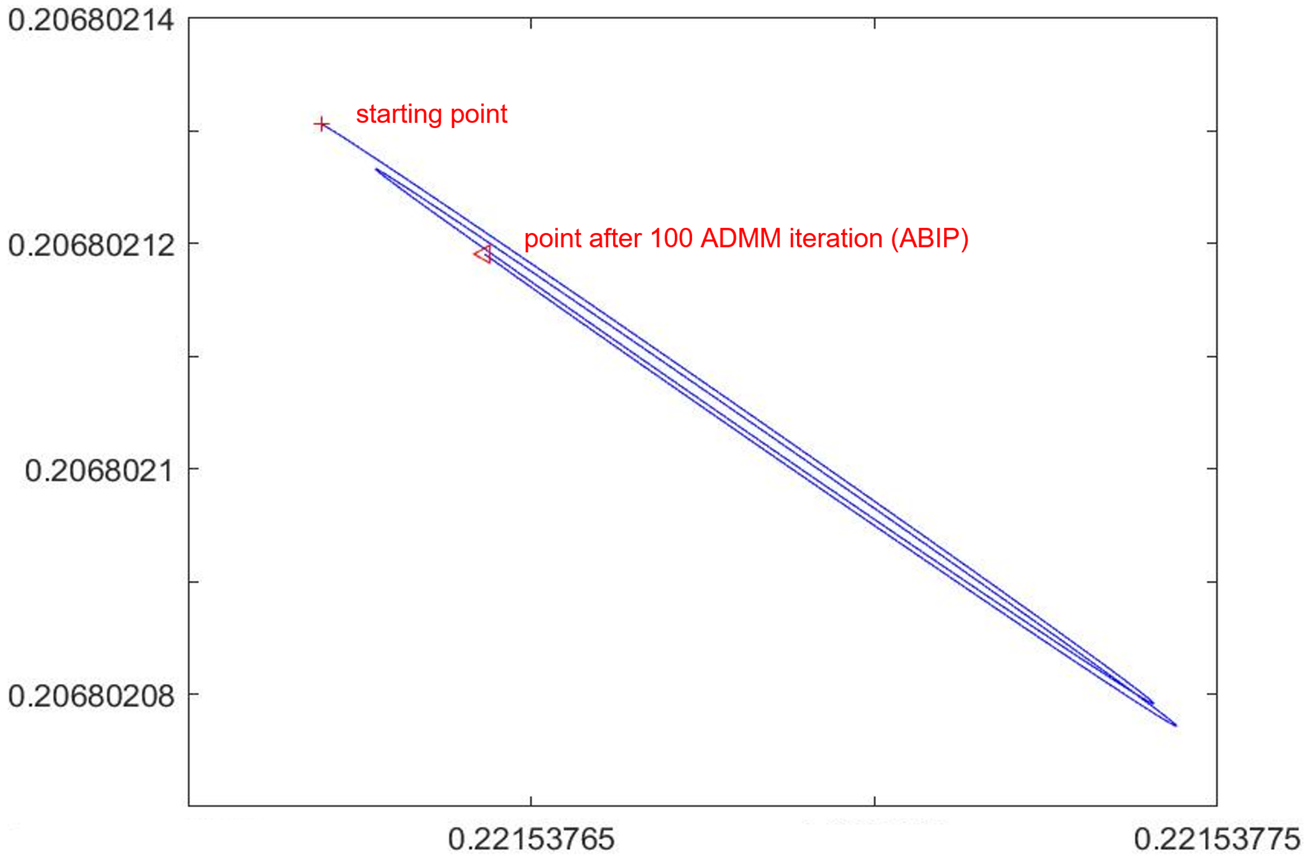
…

Strategy *k* (*μ reduction*)

- **A simple feature-to-strategy mapping is derived from a machine learning model**
- **For generalization limit the number of strategies (2 or 3 types)**

ABIP – Restart Strategy I

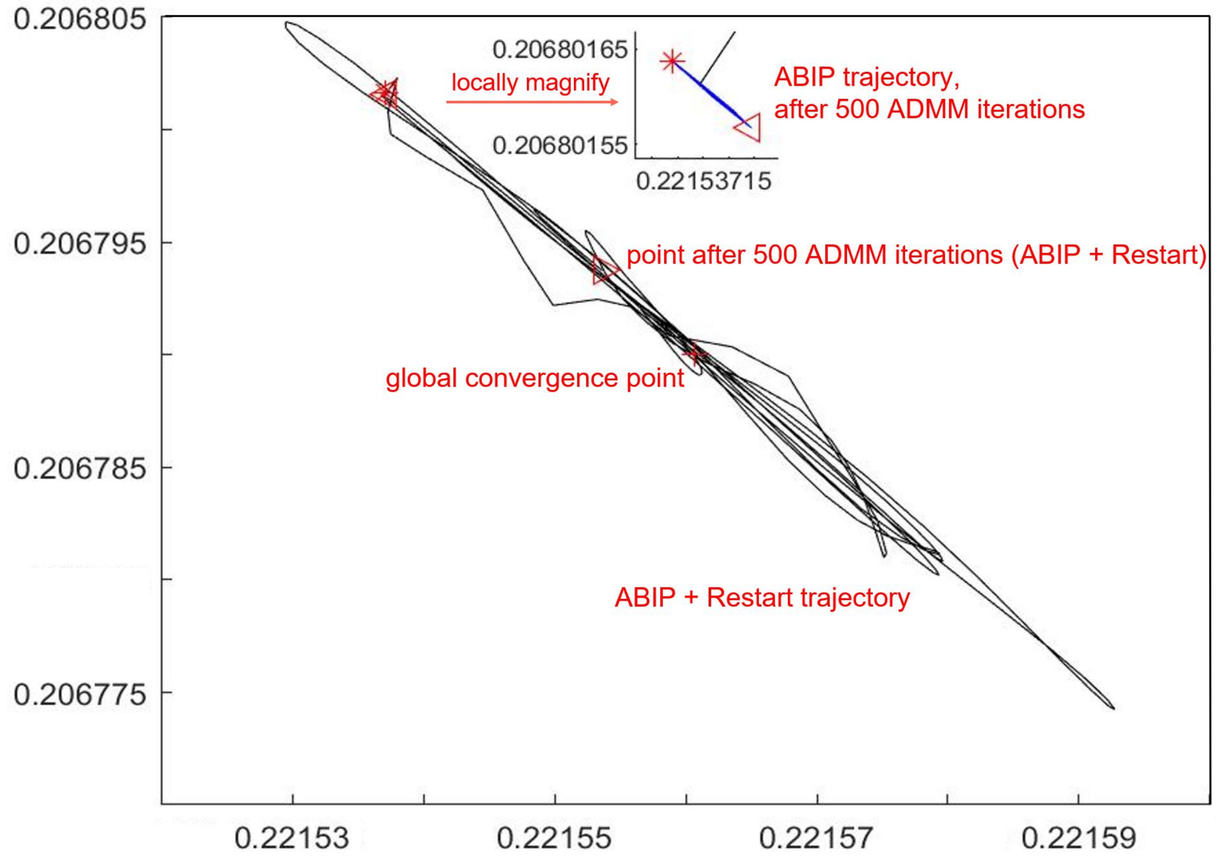
- ABIP tends to induce a spiral trajectory



Instance SC50B (only plot the first two dimension,)

ABIP – Restart Strategy II

- After restart, ABIP moves more aggressively and converges faster (reduce almost 70% ADMM iterations) !



Instance SC50B (only plot the first two dimension, after restart)

ABIP – Netlib

- Selected 105 Netlib instances
- $\epsilon = 10^{-6}$, use the direct method, 10^6 max ADMM iterations

Method	# Solved	# IPM	# ADMM	Avg.Time (s)
ABIP	65	74	265418	87.07
+ restart	68	74	88257	23.63
+ rescale	84	72	77925	20.44
+ hybrid μ (=ABIP+)	**86**	**22**	**73738**	**14.97**

- Hybrid μ : If $\mu > \epsilon$ use the aggressive strategy, otherwise use another strategy
- ABIP+ decreases both # IPM iterations and # ADMM iterations significantly

ABIP – MIP2017

- 240 MIP2017 instances
- $\epsilon = 10^{-4}$, presolved by PaPILO, use the direct method, 10^6 max ADMM iterations

Method	# Solved	SGM
COPT	**240**	**1**
PDLP(Julia)	202	17.4
ABIP	192	34.8
ABIP3+ Integration	**212**	**16.7**

- PDLP (Lu et al. 2021) is a practical first-order method (i.e., the primal-dual hybrid gradient (PDHG) method) for linear programming, and it enhences PDHG by a few implementation tricks.
- SGM stands for Shifted Geometric Mean, a standard measurement of solvers' performance

ABIP – PageRank

- 117 instances, generated from sparse matrix datasets: DIMACS10, Gleich, Newman and SNAP. Second order methods in commercial solver fail in most of these instances.

- $\epsilon = 10^{-4}$, use the indirect method, 5000 max ADMM iterations.

Method	# Solved	SGM
PDLP(Julia)	**122**	**1**
ABIP3+	119	1.31

- Examples:

Instance	# nodes	PDLP (Julia)	ABIP3+
amazon0601	403394	117.54	**71.15**
coAuthorsDBLP	299067	51.66	**24.70**
web-BerkStan	685230	447.68	**139.75**
web-Google	916428	293.01	**148.18**

ABIP – PageRank

- Generated by Google code

- When # nodes equals to # edges, the generated instance is a staircase matrix. For example,

-1.0000	0.1980	0	0	0	0	0	0	0	0
0.9900	-1.0000	0.4950	0.9900	0.4950	0.4950	0	0	0	0
0	0.1980	-1.0000	0	0	0	0.4950	0	0	0
0	0.1980	0	-1.0000	0	0	0	0	0	0
0	0.1980	0	0	-1.0000	0	0	0.9900	0	0
0	0.1980	0	0	0	-1.0000	0	0	0.9900	0
0	0	0.4950	0	0	0	-1.0000	0	0	0.9900
0	0	0	0	0.4950	0	0	-1.0000	0	0
0	0	0	0	0	0.4950	0	0	-1.0000	0
0	0	0	0	0	0	0.4950	0	0	-1.0000

10^4

Staircase matrix instance (# nodes = 10)

- In this case, ABIP+ is significantly faster than PDLP!

# nodes	PDLP (Julia)	ABIP+
10^4	8.60	**0.93**
10^5	135.67	**10.36**
10^6	2248.40	**60.32**

ABIP – Extension to Conice Linear Program

ABIP iteration remains valid for general conic linear program

$$\begin{aligned} \min\ & c^T x \\ \text{s.t. } & Ax = b \\ & x \in \mathcal{K} \end{aligned}$$

- ABIP-subproblem requires to solve a proximal mapping $x^+ = \operatorname{argmin} \lambda F(x) + \frac{1}{2}\|x - c\|^2$ with respect to the log-barrier functions $F(x)$ in $B(u, v, \mu^k)$

Positive orthant

- $F(x) = -\log(x)$
- $x = \operatorname{argmin} \lambda F(x) + \frac{1}{2}|x - c|^2 = \frac{c+\sqrt{c^2+4\lambda}}{2}$

Second-order cone

- $F(\boldsymbol{x}) = -\log(t^2 - \|x\|^2), \boldsymbol{x} = (t; x)$
- Can be solved by finding the root of quadratic functions

Positive semidefinite cone

- $F(\boldsymbol{x}) = -\log(\det x)$
- Equivalent to solve $-\lambda x^{-1} - c + x = 0$
- Can be solved by eigen decomposition

- The total IPM and ADMM iteration complexities of ABIP for conic linear program are respectively:

$$T_{IPM} = O\left(\log\left(\frac{1}{\varepsilon}\right)\right), \quad T_{ADMM} = O(\frac{1}{\varepsilon}\log\left(\frac{1}{\varepsilon}\right))$$

ABIP – Customization for ML

ABIP solves linear system:

$$\begin{pmatrix} I_m & A \\ A^T & -I_n \end{pmatrix} \begin{pmatrix} \hat{u}_1 \\ \hat{u}_2 \end{pmatrix} = \begin{pmatrix} \hat{\omega}_1 \\ -\hat{\omega}_2 \end{pmatrix}$$

elimination

$$\begin{pmatrix} I_m + AA^T & \\ A^T & -I_n \end{pmatrix} \begin{pmatrix} \hat{u}_1 \\ \hat{u}_2 \end{pmatrix} = \begin{pmatrix} \hat{\omega}_1 - A\hat{\omega}_2 \\ -\hat{\omega}_2 \end{pmatrix}$$

- if A is a general sparse matrix, we prefer augmented system, which is solved by sparse LDL decomposition

- If A is dense or it has highly different row and col dimensionalities, we prefer normal equation

For many QP problems in machine learning, we provide customized linear system solver by applying Sherman-Morrison-Woodbury formula and simplifying the normal equation

LASSO

- Data matrix $\tilde{A}$ of LASSO has n features, m samples
- The dimension of factorized matrix reduced from $2m + 2n + 3$ to $\min\{m, n\}$

SVM

- Data matrix $\tilde{A}$ of SVM has n features, m samples
- The dimension of factorized matrix reduced from $3m + 4n + 5$ to $n + 1$

ABIP – SVM

- For 6 large instances from LIBSVM, $\epsilon = 10^{-3}$, time limit = 2000s

	ABIP	OSQP	RACQP	SCS	GUROBI
solved	6	3	6	4	5
1st	1	2	2	0	1
2nd	2	1	2	1	0
3rd	3	0	0	1	1
4th	0	0	2	1	1
5th	0	0	0	1	2

Summary

ABIP is

- **a general purpose LP solver**
- **using ADMM to solve the subproblem**
- **developed with heuristics and intuitions from various strategies**
- **equipped with several new computational tricks**
- **Smart dual updates?**

Today’s Talk

- New developments of ADMM-based interior point (ABIP) Method
- Optimal Diagonal Preconditioner and HDSDP
- A Dimension Reduced Second-Order Method
- SOLNP: a Derivative-Free Optimization Solver

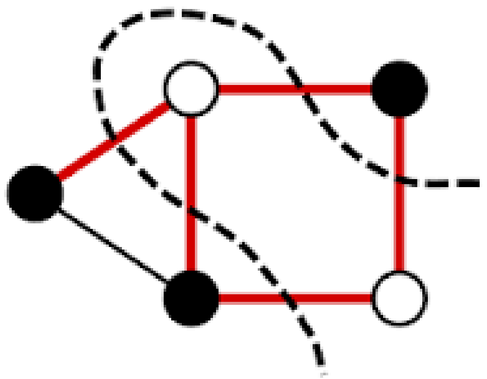
HDSDP: Homogeneous Dual-Scaling SDP solver

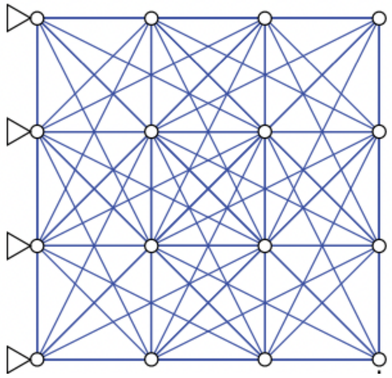
$$\begin{aligned} \min_{X} \quad & \langle C, X \rangle \\ \text{subject to} \quad & \mathcal{A}X = b \\ & X \in \mathbb{S}^n_+ \end{aligned}$$

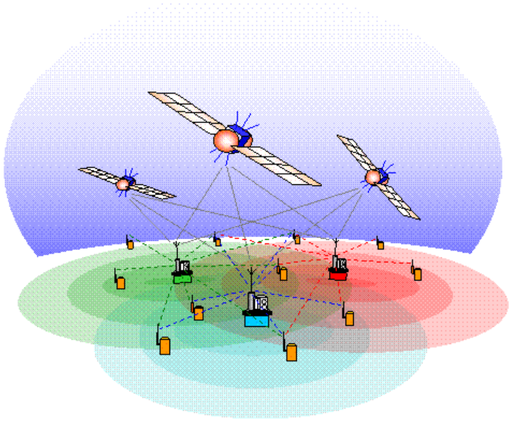
$$\begin{aligned} \mathcal{A}\mathbf{X} - \mathbf{b}\tau &= \mathbf{0} \\ -\mathcal{A}^*\mathbf{y} + \mathbf{C}\tau - \mathbf{S} &= \mathbf{0} \\ \mathbf{b}^\top\mathbf{y} - \langle \mathbf{C}, \mathbf{X} \rangle - \kappa &= 0 \\ \mathbf{X}, \mathbf{S} \succeq 0, \qquad & \kappa, \tau \geq 0 \end{aligned}$$

$$\begin{aligned} AP + PA + I &\preceq 0 \\ P &\succeq 0 \end{aligned}$$

$$\sum_{i=1}^{m} F_i y_i \preceq F_0$$







Interior point method for SDPs

SDP is solvable in polynomial time using the interior point methods

- **Take Newton step towards the perturbed KKT system**

$$\begin{aligned} \mathcal{A}X &= b \\ \mathcal{A}^*y + S &= C \\ XS &= 0 \end{aligned} \qquad \begin{aligned} \mathcal{A}X &= b \\ \mathcal{A}^*y + S &= C \\ XS &= \mu I \end{aligned} \qquad \begin{aligned} \mathcal{A}\Delta X &= -R_P \\ \mathcal{A}^*\Delta y + \Delta S &= -R_D \\ H_P(X\Delta S + \Delta X S) &= -R_\mu \end{aligned}$$

- **Efficient numerical solvers have been developed**

  COPT, Mosek, SDPT3, SDPA, DSDP…

- **Most IPM solvers adopt primal-dual path-following IPMs except DSDP**

  DSDP (Dual-scaling SDP) implements a dual potential reduction method

Homogeneous dual-scaling algorithm

From arbitrary starting dual solution $(y, S \succ 0, \tau > 0)$ with dual residual *R*

$$\begin{aligned} \mathcal{A}X - b\tau &= 0 \\ -\mathcal{A}^*y + C\tau - S &= 0 \\ b^\top y - \langle C, X\rangle - \kappa &= 0 \\ X = \mu S^{-1} \qquad & \kappa = \mu\tau^{-1} \end{aligned}$$

$$\begin{aligned} \mathcal{A}(X+\Delta X) - b(\tau+\Delta\tau) &= 0 \\ -\mathcal{A}^*(y+\Delta y) + C(\tau+\Delta\tau) - (S+\Delta S) &= 0 \\ \mu S^{-1}\Delta S S^{-1} + \Delta X &= \mu S^{-1} - X \\ \mu\tau^{-2}\Delta\tau + \Delta\kappa &= \mu\tau^{-1} - \kappa \end{aligned}$$

$$\begin{pmatrix} \mu M & -b - \mu\mathcal{A}S^{-1}CS^{-1} \\ -b + \mu\mathcal{A}S^{-1}CS^{-1} & -\mu(\langle C, S^{-1}CS^{-1}\rangle + \tau^{-2}) \end{pmatrix} \begin{pmatrix} \Delta y \\ \Delta\tau \end{pmatrix} = \begin{pmatrix} b\tau \\ b^\top y - \mu\tau^{-1} \end{pmatrix} - \mu \begin{pmatrix} \mathcal{A}S^{-1} \\ \langle C, S^{-1}\rangle \end{pmatrix} + \mu \begin{pmatrix} \mathcal{A}S^{-1}RS^{-1} \\ \langle C, S^{-1}RS^{-1}\rangle \end{pmatrix}$$

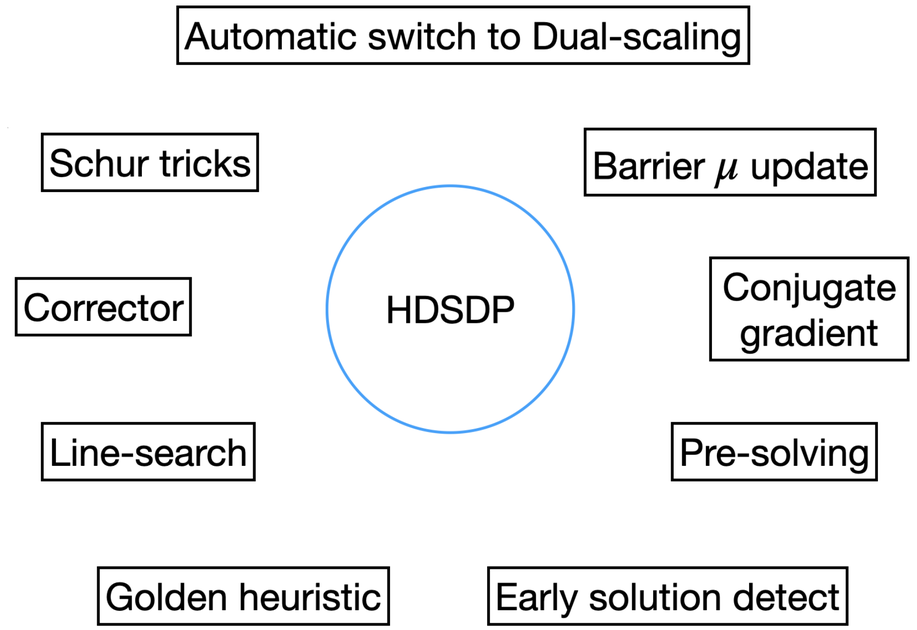
- **Primal iterations can still be fully eliminated**
- $S = -\mathcal{A}^*y + C\tau - R$ **inherits sparsity pattern of data**
  - **Less memory and since X is generally dense**
- **Infeasibility or an early feasible solution can be detected via the embedding**

New strategies are tailored for the method

Computational aspects for HDSDP Solver

To enhance performance, HDSDP (written in ANSI C) is equipped with

- **Pre-solving that detects special structure and dependency**
- **Line-searches over barrier to balance optimality & centrality**
- **Heuristics to update the barrier parameter μ**
- **Corrector strategy to reuse the Schur matrix**
- **A complete dual-scaling algorithm from DSDP5.8**
- **More delicate strategies for the Schur system**



Computational results

- **HDSDP is tuned and tested for many benchmark datasets**
- **Good performance on problems with both low-rank structure and sparsity**
- **Solve around 70/75 Mittelmann's benchmark problems**
- **Solve 90/92 SDPLIB problems**

Instance	DSDP5.8	HDSDP	Mosek v9	SDPT3	COPT v5
G40_mb	18	7	174	25	18
G48_mb	36	8	191	49	35
G48mc	11	2	71	24	18
G55mc	200	179	679	191	301
G59mc	347	246	646	256	442
G60_mb	700	213	7979	592	714
G60mc	712	212	8005	590	713

Instance	DSDP5.8	HDSDP	Mosek v9	SDPT3	COPT v5
checker1.5	87	41	72	71	81
foot	28	14	533	32	234
hand	4	2	76	8	40
ice_2.0	833	369	4584	484	1044
p_auss2	832	419	5948	640	721
r1_2000	17	8	333	20	187
torusg3-15	101	22	219	61	84

Selected Mittelmann's benchmark problems where HDSDP is fastest (all the constraints are rank-one)

(Results run on an intel i11700K machine)

Optimal Diagonal Pre-Conditioner [QGHYZ20]

Given matrix $M = X^\top X \succ 0$, iterative method (e.g., CG) is often applied to solve

$$Mx = b$$

- Convergence of iterative methods depends on the condition number $\kappa(M)$
- Good performance needs pre-conditioning and we solve $P^{-1/2}MP^{-1/2}x' = b$

  A good pre-conditioner reduces $\kappa(P^{-1/2}MP^{-1/2})$
- Diagonal $P = D$ is called diagonal pre-conditioner

More generally, we wish to find D (or E) such that $\kappa(D \cdot X \cdot E)$ is minimized ?

Is it possible to find optimal D^* and E^* ?

SDP works!

Application: Optimal Diagonal Pre-Conditioner

$$\min_{D \text{ diagonal}, D \succeq 0} \kappa(DMD)$$

$$\min_{D \text{ diagonal}, D \succeq 0} \kappa(X^T D X)$$

$$\begin{aligned} \min_{D,\kappa} \quad & \kappa \\ \text{subject to} \quad & I \preceq DMD \preceq \kappa I \end{aligned}$$

$$\begin{aligned} \min_{\kappa, D \succeq 0} \quad & \kappa \\ \text{subject to} \quad & \kappa X^T D X \succeq I \\ & I \succeq X^T D X \end{aligned}$$

$$\begin{aligned} \min_{D,\kappa} \quad & \kappa \\ \text{subject to} \quad & D \preceq M \\ & \kappa D \succeq M \end{aligned}$$

$$\begin{aligned} \max_{\tau, D \succeq 0} \quad & \tau \\ \text{subject to} \quad & X^T D X \succeq \tau \\ & I \succeq X^T D X \end{aligned}$$

- Finding the optimal diagonal pre-conditioner is an SDP
- Two SDP blocks and sparse coefficient matrices
- Trivial dual interior-feasible solution
- An ideal formulation for dual SDP methods $D = \sum d_i e_i e_i^T$

What about two-sided ?

Two-Sided Pre-Conditioner

$$\min_{D_1 \succeq 0, D_2 \succeq 0} \quad \kappa(D_1 X D_2)$$

- Common in practice and popular heuristics exist

  e.g. Ruiz-scaling, matrix equilibration & balancing
- Not directly solvable using SDP
- Can be solved by *iteratively* fixing $D_1 (D_2)$ and optimizing the other side

  Solving a sequence of SDPs
- Answer a question: how far can diagonal pre-conditioners go

Computational Results: Solving for the Optimal Pre-Conditioner

$$\begin{aligned} \min_{D,\kappa} \quad & \kappa \\ \text{subject to} \quad & D \preceq M \\ & \kappa D \succeq M \end{aligned}$$

$$\begin{aligned} \max_{\delta,d} \quad & \delta \\ \text{subject to} \quad & D - M \preceq 0 \\ & \delta M - D \preceq 0 \end{aligned}$$

SDP from optimal drag pre-conditioning problem

- Perfectly in the dual form
- Trivial dual feasible interior point solution
- 1 is an upper-bound for the optimal objective value

HDSDP

- A dual SDP algorithm (successor of DSDP5.8 by Benson)
- Support initial dual solution
- Customization for the diagonal pre-conditioner

n	Sparsity	HDSDP (start from $(-10^6, 0)$)	COPT	Mosek	SDPT3
500	0.05	7.1	6.8	9.1	18.0
1000	0.09	44.5	53.9	54.2	327.0
2000	0.002	34.3	307.1	374.7	572.3
5000	0.0002	64.3	>1200	>1200	>1200

Computational Results: Optimal Diagonal Pre-Conditioner

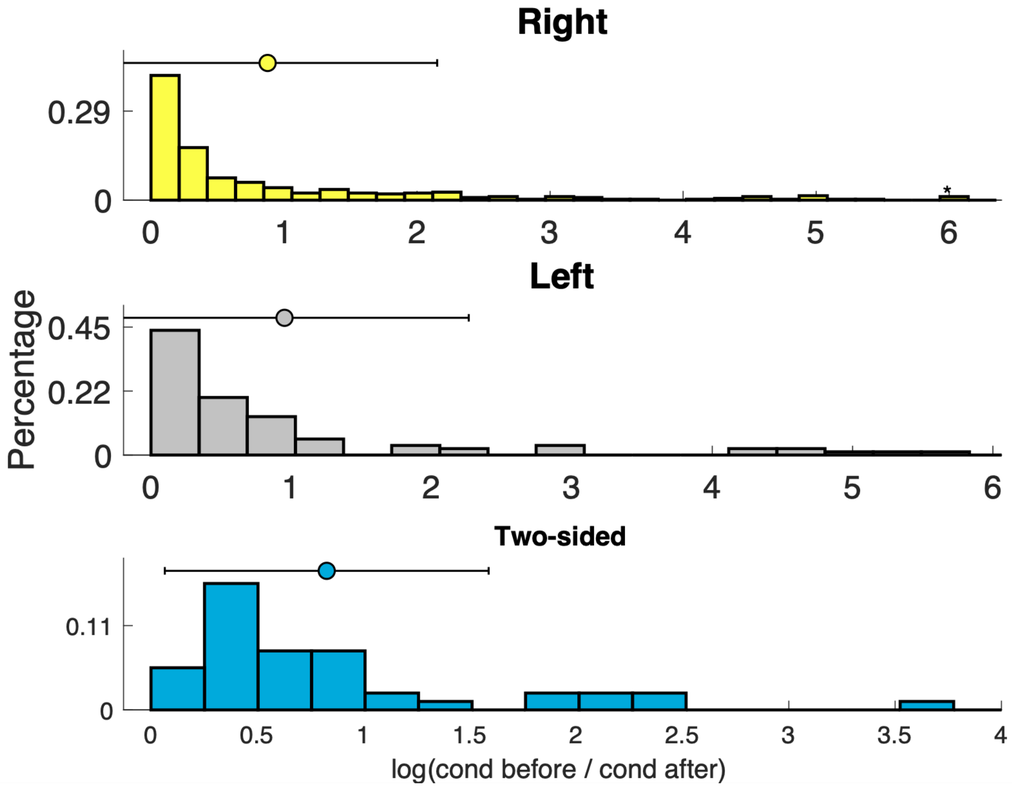
- Test over 491 Suite Sparse Matrices of fewer than 1000 columns

Reduction	Number
≥80%	121
≥50%	190
≥20%	261

Average reduction	49.7%
Better than diagonal	36.0%
Average time	1.29

- LIBSVM datasets

Mat	Size	Cbef	Caft	Reduce
YearPredictionMSD	90	5233000.00	470.20	0.999910
YearPredictionMSD.t	90	5521000.00	359900.00	0.934816
abalone_scale.txt	8	2419.00	2038.00	0.157291
bodyfat_scale.txt	14	1281.00	669.10	0.477475
cadata.txt	8	8982000.00	7632.00	0.999150
cpusmall_scale.txt	12	20000.00	6325.00	0.683813
eunite2001.t	16	52450000.00	8530.00	0.999837
eunite2001.txt	16	67300000.00	3591.00	0.999947
housing_scale.txt	13	153.90	83.22	0.459371
mg_scale.txt	6	10.67	10.03	0.059988
mpg_scale.txt	7	142.50	107.20	0.247842
pyrim_scale.txt	27	49100000.00	3307.00	0.999933
space_ga_scale.txt	6	1061.00	729.60	0.312041
triazines_scale.txt	60	24580000.00	15460000.00	0.371034



Distribution of condition number reduction
0.5 means 70% reduction in condition number
1 means 90% and 2 means 99% reduction in condition number

Summary

HDSDP is

- a general purpose SDP solver
- using dual-scaling and simplified HSD
- developed with heuristics and intuitions from DSDP
- equipped with several new computational tricks
- more iterative methods for solving subproblems?

Today’s Talk

- New developments of ADMM-based interior point (ABIP) Method
- Optimal Diagonal Preconditioner and HDSDP
- A Dimension Reduced Second-Order Method
- SOLNP: a Derivative-Free Optimization Solver

Early Complexity Analyses for Nonconvex Optimization

$$\min f(x), x \in X \; in \; \mathbb{R}^n,$$

- where f is nonconvex and twice-differentiable,

  $g_k = \nabla f(x_k), H_k = \nabla^2 f(x_k)$

- Goal: find x_k such that:

  $\| \nabla f(x_k) \| \leq \epsilon$ (primary, first-order condition)

  $\lambda_{min}(H_k) \geq -\sqrt{\epsilon}$ (in active subspace, secondary, second-order condition)

- For the ball-constrained nonconvex QP: $\min \; c^T x + 0.5 x T Q x \; s.t. \; \| x \|_2 \leq 1$

  O(loglog(ϵ^{-1})); see Y (1989,93), Vavasis&Zippel (1990)

- For nonconvex QP with polyhedral constraints: O(ϵ^{-1}); see Y (1998), Vavasis (2001)

Standard methods for general nonconvex optimization I

First-order Method (FOM): Gradient-Type Methods

- Assume f has L-Lipschitz cont. gradient
- Global convergence by, e.g., linear-search (LS)
- No guarantee for the second-order condition
- Worst-case complexity, $O(\epsilon^{-2})$; see the textbook by Nesterov (2004)

Each iteration requires $O(n^2)$ operations

Standard methods for general nonconvex optimization II

Second-order Method (SOM): Hessian-Type Methods

- Assume f has M-Lipschitz cont. Hessian
- Global convergence by, e.g., linear-search (LS), Trust-region (TR), or Cubic Regularization
- Convergence to second-order points
- No better than $O(\epsilon^{-2})$, for traditional methods (steepest descent and Newton); according to Cartis et al. (2010) .

Each iteration requires $O(n^3)$ operations

Analyses of SOM for general nonconvex optimization since 2000

Variants of SOM

- Trust-region with the fixed-radius strategy, $O(\epsilon^{-3/2})$, see the lecture notes by Y since 2005
- Cubic regularization, $O(\epsilon^{-3/2})$, see Nesterov and Polyak (2006), Cartis, Gould, and Toint (2011)
- A new trust-region framework, $O(\epsilon^{-3/2})$, Curtis, Robinson, and Samadi (2017)

With “slight” modification, complexity of SOM reduces from $O(\epsilon^{-2})$ to $O(\epsilon^{-3/2})$

Motivation from multi-directional FOM

- Two-directional FOM, with d_k being the momentum direction $(x_k - x_{k-1})$

$$x_{k+1} = x_k - \alpha_k^1 \nabla f(x_k) + \alpha_k^2 d_k \ = x_k + d_{k+1}$$

  where step-sizes are constructed; including CG, PT, AGD, Polyak, ADAM and many others.

- In SOM, a method typically minimizes a full dimensional quadratic Taylor expansion to obtain direction vector d_{k+1}. For example, one TR step solves for d_{k+1} from

$$\min_d \quad (g_k)^T d + 0.5 d T H_k d \quad s.t. \, ||\mathrm{d}||_2 \leq \Delta_k$$

  where Δ_k is the trust-region radius.

- DRSOM: Dimension Reduced Second-Order Method

  Motivation: using few directions in SOM

DRSOM I

- The DRSOM in general uses m-independent directions

$$d(\alpha) := D_k \alpha \,, D_k \in R^{nm},\ \alpha \in R^m$$

- Plug the expression into the full-dimension TR quadratic minimization problem,

  we minimize a m-dimension trust-region subproblem to decide “m stepsizes”:

$$\min\ m_k^{\alpha}(\alpha) := (c_k)^T \alpha + \frac{1}{2}\alpha^T Q_k \alpha$$
$$||\alpha||_{G_k} \leq \Delta_k$$

$$G_k = D_k^T D_k, Q_k = D_k^T H_k D_k, c_k = (g_k)^T D_k$$

How to choose D_k? How great would m be? Rank of H_k?
(Randomized) rank reduction of a symmetric matrix to log(n) (So et al. 08)?

DRSOM II

- In following, as an example, DRSOM adopts two FOM directions

$$d = -\alpha^1 \nabla f(x_k) + \alpha^2 d_k := d(\alpha)$$

where $g_k = \nabla f(x_k), H_k = \nabla^2 f\left(x^k\right), d_k = x_k - x_{k-1}$

- Then we minimize a 2-D trust-region problem to decide “two step-sizes”:

$$\min\ m_k^{\alpha}(\alpha) := f(x_k) + (c_k)^T \alpha + \frac{1}{2}\alpha^T Q_k \alpha$$

$$||\alpha||_{G_k} \leq \Delta_k$$

$$G_k = \begin{bmatrix} g_k^T g_k & -g_k^T d_k \\ -g_k^T d_k & d_k^T d_k \end{bmatrix}, Q_k = \begin{bmatrix} g_k^T H_k g_k & -g_k^T H_k d_k \\ -g_k^T H_k d_k & d_k^T H_k d_k \end{bmatrix}, c_k = \begin{bmatrix} -||g_k||^2 \\ g_k^T d_k \end{bmatrix}$$

DRSOM III

DRSOM can be seen as:

- "Adaptive" **Accelerated Gradient Method** (Polyak's momentum 60)
- A second-order method minimizing quadratic model in the reduced 2-D

  $$m_k(d) = f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T \nabla^2 f(x_k) d, d \in \text{span}\{-g_k, d_k\}$$

  compare to, e.g., Dogleg method, 2-D Newton **Trust-Region Method**

  $d \in \text{span}\{g_k, [H(x_k)]^{-1} g_k\}$ (e.g., Powell 70)
- A conjugate direction method for convex optimization exploring the **Krylov Subspace** (e.g., Yuan&Stoer 95)
- For convex quadratic programming with no radius limit, terminates in n steps

Computing Hessian-Vector Product in DRSOM is the Key

In the DRSOM with two directions:

$$Q_k = \begin{bmatrix} g_k^T H_k g_k & -g_k^T H_k d_k \\ -g_k^T H_k d_k & d_k^T H_k d_k \end{bmatrix}, c_k = \begin{bmatrix} -||g_k||^2 \\ g_k^T d_k \end{bmatrix}$$

How to cheaply obtain Q? Compute $H_k g_k, H_k d_k$ first.

- Finite difference:

  $$H_k \cdot v \approx \frac{1}{\epsilon}[g(x_k + \epsilon \cdot v) - g_k],$$

- Analytic approach to fit modern automatic differentiation,

  $$H_k g_k = \nabla(\frac{1}{2} g_k^T g_k), H_k d_k = \nabla(d_k^T g_k),$$

- or use Hessian if readily available !

DRSOM: key assumptions and theoretical results (Zhang at al. SHUFE)

Assumption. (a) f has Lipschitz continuous Hessian. (b) DRSOM iterates with a fixed-radius strategy: $\Delta_k = \epsilon/\beta$) c) **If the Lagrangian multiplier $\boldsymbol{\lambda_k < \sqrt{\epsilon}}$, assume $\| (\boldsymbol{H_k - \widetilde{H}_k})\boldsymbol{d_{k+1}} \| \leq \boldsymbol{C} \| \boldsymbol{d_{k+1}} \|^2$ (Cartis et al.),** where $\widetilde{H}_k$ is the projected Hessian in the subspace (commonly adopted for approximate Hessian)

Theorem 1. If we apply DRSOM to QP, then the algorithms terminates in at most n steps to find a first-order stationary point

Theorem 2. (Global convergence rate) For *f* with second-order Lipschitz condition, DRSOM terminates in $O(\epsilon^{-3/2})$ iterations. Furthermore, the iterate x_k satisfies the first-order condition, and the Hessian is positive semi-definite in the subspace spanned by the gradient and momentum.

Theorem 3. (Local convergence rate) If the iterate x_k converges to a strict local optimum x^* such that $H(x^*) \succ 0$, and if **<u>Assumption (c)</u>** is satisfied as soon as $\lambda_k \leq C_\lambda \| d_{k+1} \|$, then DRSOM has a local superlinear (quadratic) speed of convergence, namely: $\| x_{k+1} - x^* \| = O(\| x_k - x^* \|^2)$

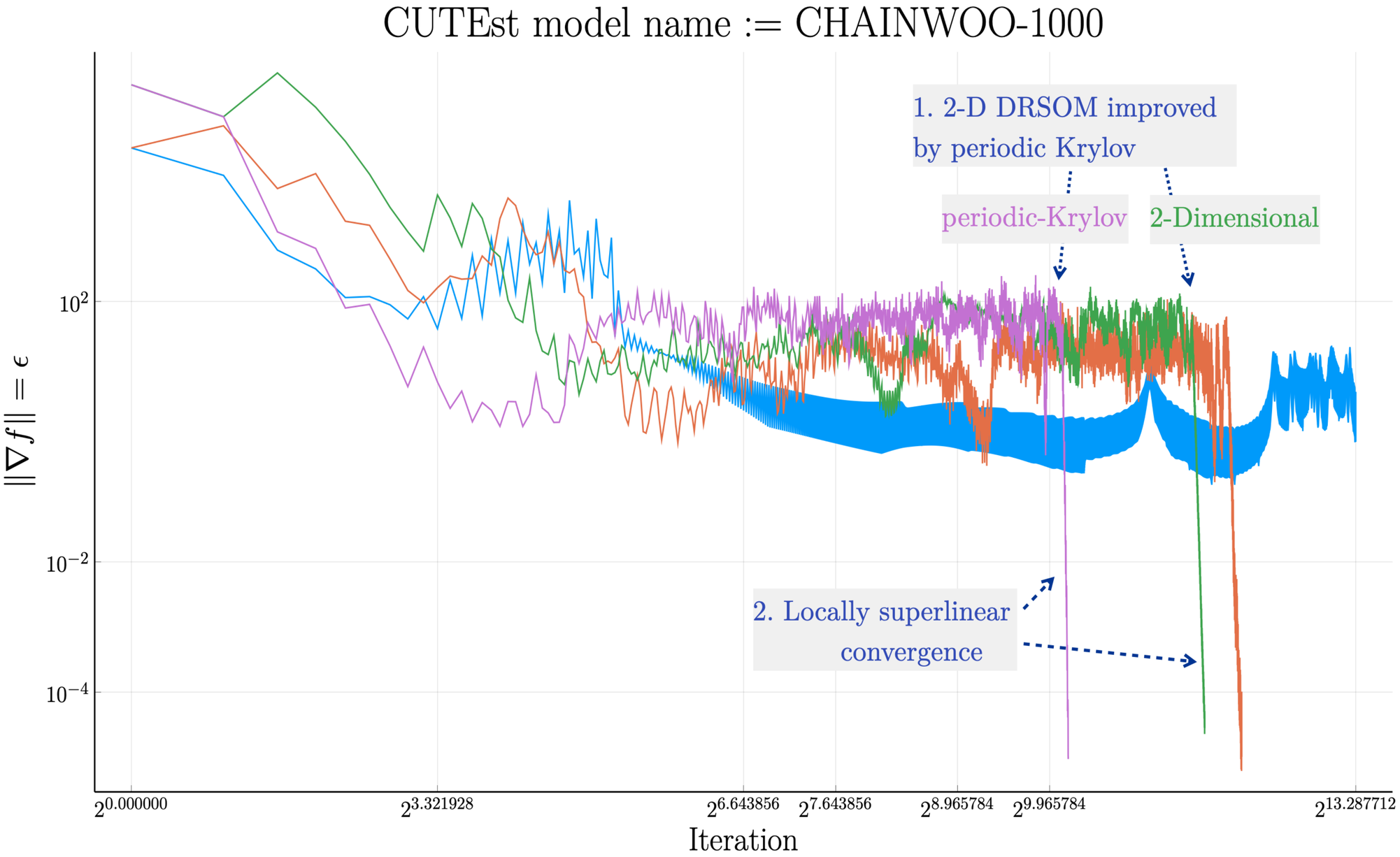
DRSOM: How to remove Assumption (c)?

- Global rate: ensure **Assumption (c)** holds ***periodically*** (whenever needed, e.g., switch to Krylov)
- Local rate: ensure **Assumption (c)** holds around $\boldsymbol{x}^*$,we have the desired results.

Specifically, expand subspace if Assumption (c) does not hold…

- Carmon et al. (2018) find the NC ($O(\epsilon^{-1/4})$ for each step) and proceed
- Run **Lanczos** (worst-case without sparsity $O(n^3)$)
- Trade-off between $O(\epsilon^{-7/4})$ (more dimension-free) and $O(\epsilon^{-3/2})$

DRSOM: convergence behavior, an example



Example from the CUTEst dataset

- *GD* and *LBFGS* both use a Line-search (Hager-Zhang)
- *DRSOM-F (2-D)*: original 2-dimensional version with g_k and d_k
- *DRSOM-F (periodic-Krylov)*, guarantees $\| (H_k - \widetilde{H}_k) d_{k+1} \| \leq C \| d_{k+1} \|^2$ periodically.

Sensor Network Location (SNL)

- Consider Sensor Network Location (SNL)

$$N_x = \{(i,j) : \|x_i - x_j\| = d_{ij} \leq r_d\}, N_a = \{(i,k) : \|x_i - a_k\| = d_{ik} \leq r_d\}$$

where r_d is a fixed parameter known as the radio range. The SNL problem considers the following QCQP feasibility problem,

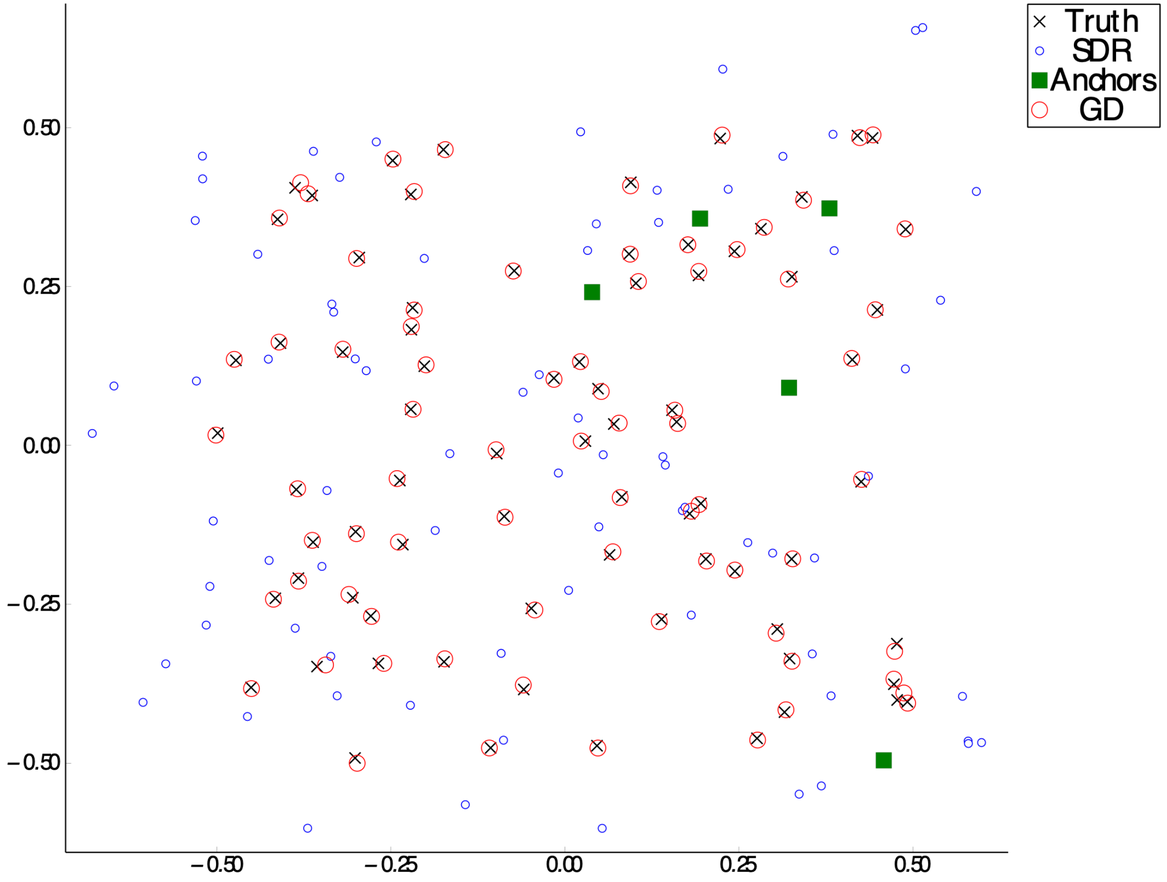
$$\|x_i - x_j\|^2 = d_{ij}^2, \forall (i,j) \in N_x$$
$$\|x_i - a_k\|^2 = \bar{d}_{ik}^2, \forall (i,k) \in N_a$$

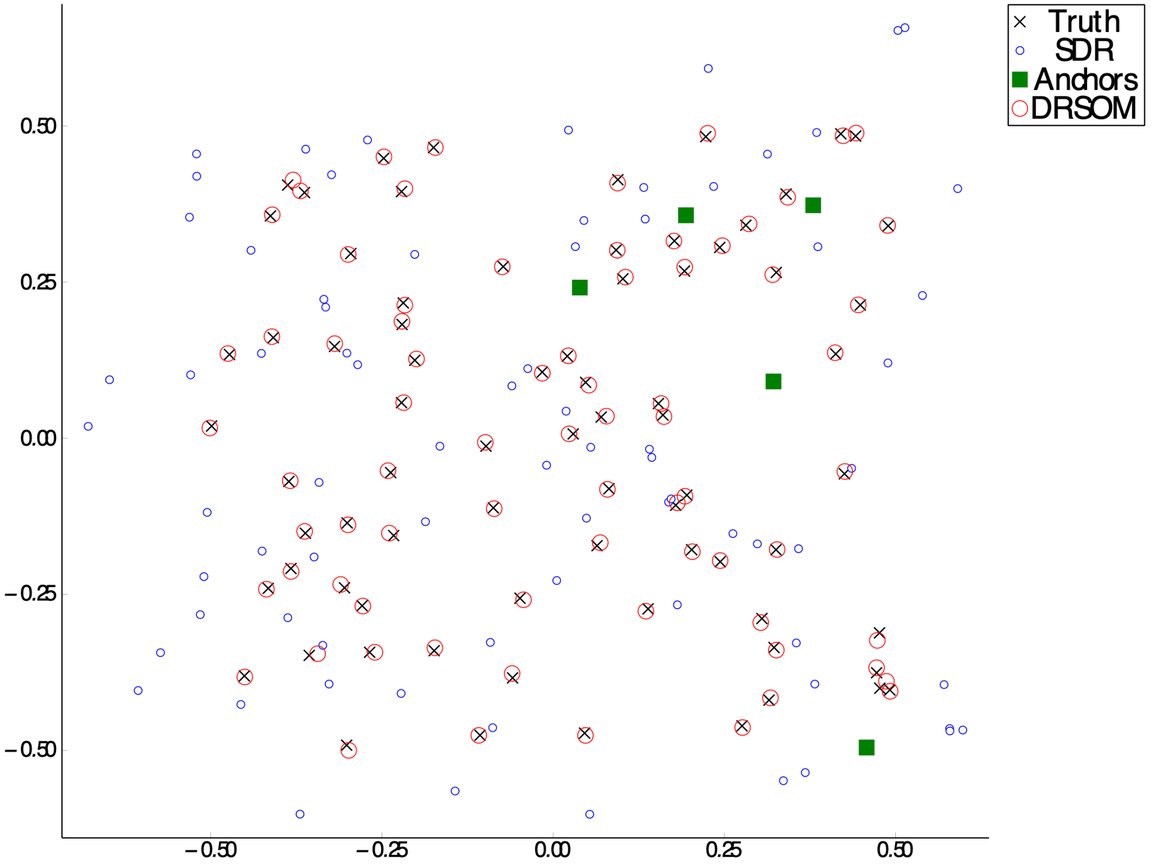
- We can solve SNL by the nonconvex nonlinear least square (NLS) problem

$$\min_X \sum_{(i<j,j)\in N_x} (\|x_i - x_j\|^2 - d_{ij}^2)^2 + \sum_{(k,j)\in N_a} (\|a_k - x_j\|^2 - \bar{d}_{kj}^2)^2.$$

Sensor Network Location (SNL)

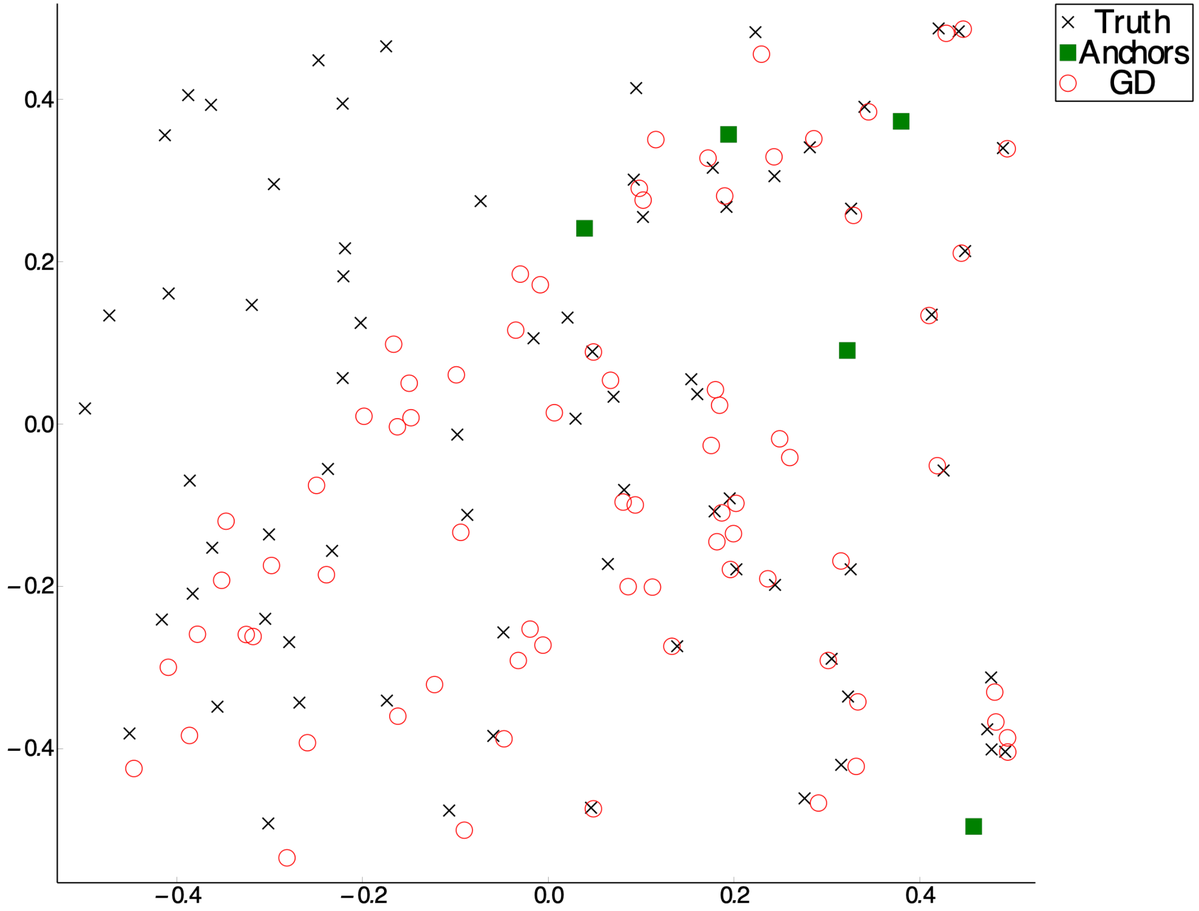
- Graphical results using SDP relaxation to initialize the NLS
- n = 80, m = 5 (anchors), radio range = 0.5, degree = 25, noise factor = 0.05
- Both Gradient Descent and DRSOM can find good solutions !

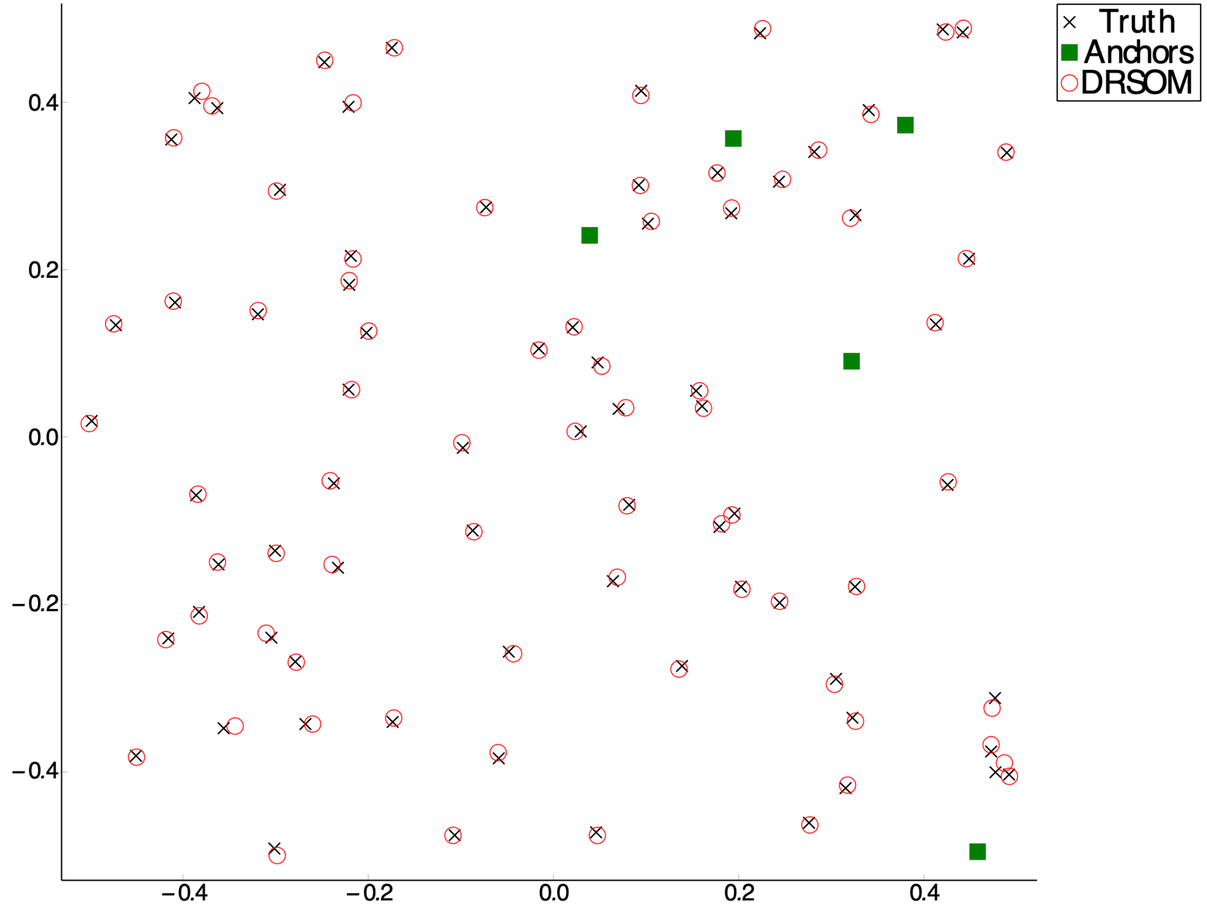




Sensor Network Location (SNL)

- Graphical results without SDP relaxation
- DRSOM can still converge to optimal solutions

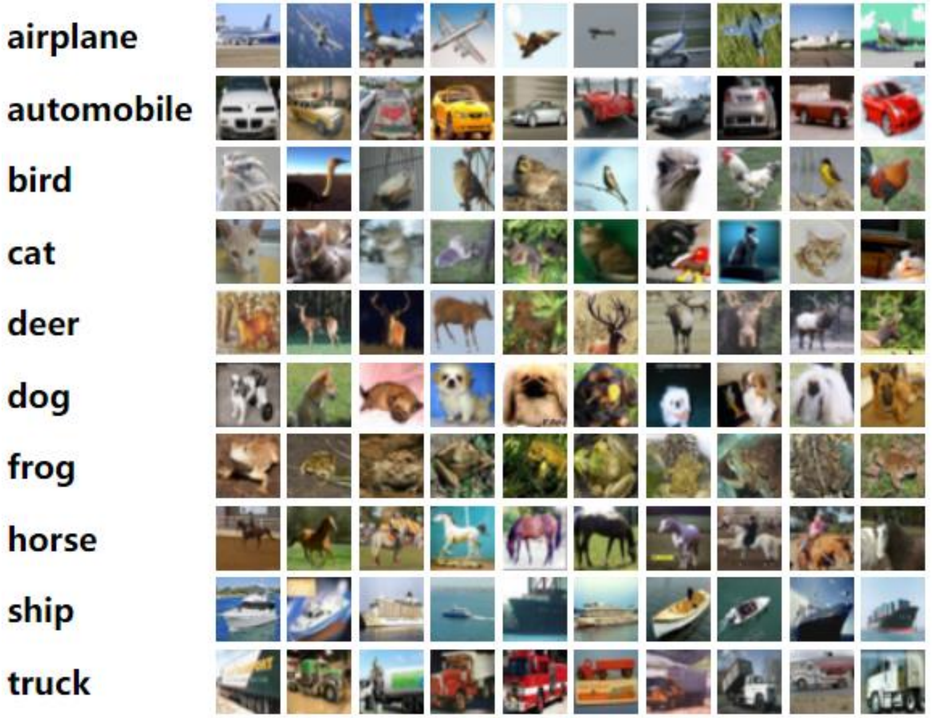




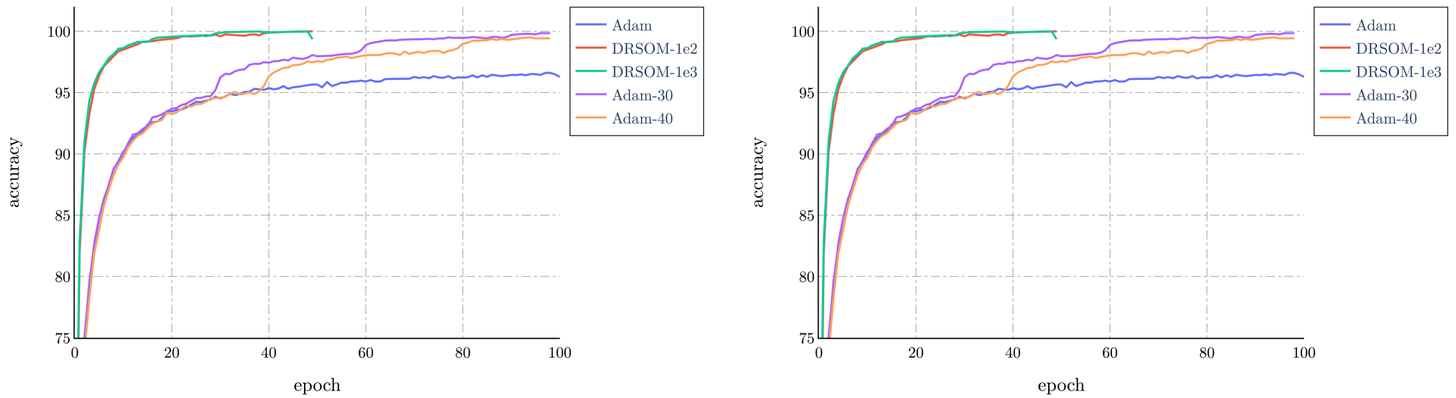
Neural Networks and Deep Learning

To use DRSOM in machine learning problems

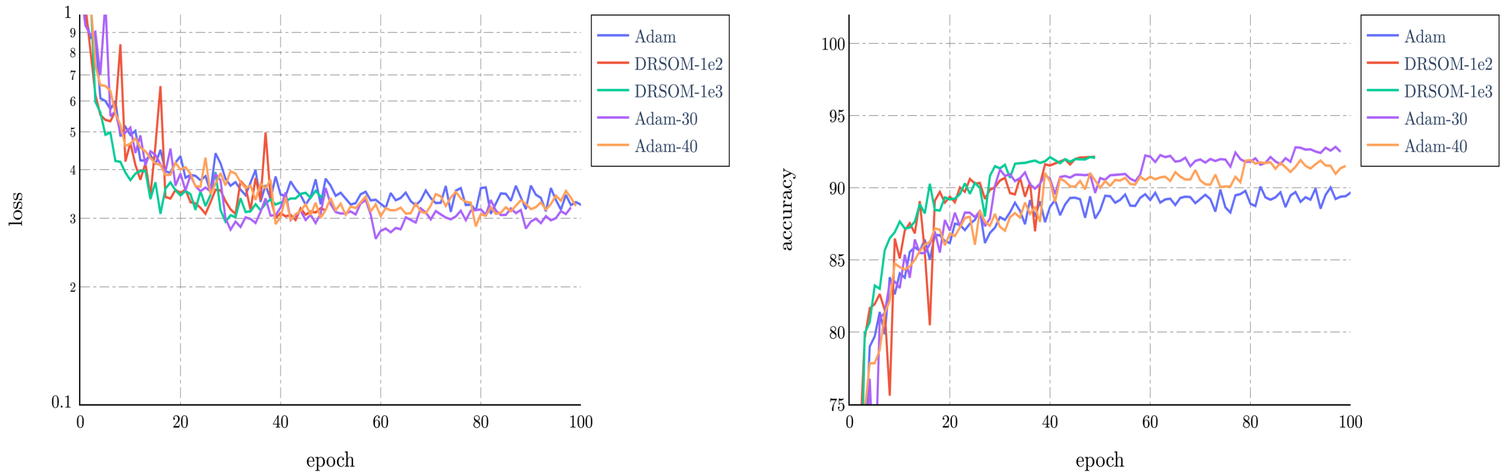
- We apply the mini-batch strategy to a vanilla DRSOM
- Use Automatic Differentiation to compute gradients
- Train ResNet18 Model with CIFAR 10
- Set Adam with initial learning rate 1e-3



Neural Networks and Deep Learning



Training results for ResNet18 with DRSOM and Adam



Test results for ResNet18 with DRSOM and Adam

Pros

- DRSOM has rapid convergence (30 epochs)
- DRSOM needs little tuning

Cons

- DRSOM may overfit the models
- Needs 4~5x time than Adam to run same number of epoch

Good potential to be a standard optimizer for deep learning!

DRSOM for Policy Gradient (PG) (Liu et al. SHUFE)

- As mentioned above, the goal is to maximize the expected discounted trajectory reward:

$$\max_{\theta \in \mathbb{R}^d} J(\theta) := \mathbb{E}_{\tau \sim p(\tau|\theta)}[\mathcal{R}(\tau)] = \int \mathcal{R}(\tau) p(\tau \mid \theta) d\tau$$

- The gradient can be estimated by:

$$\hat{\nabla} J(\theta) = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla \log p(\tau_i \mid \theta) \mathcal{R}(\tau_i)$$

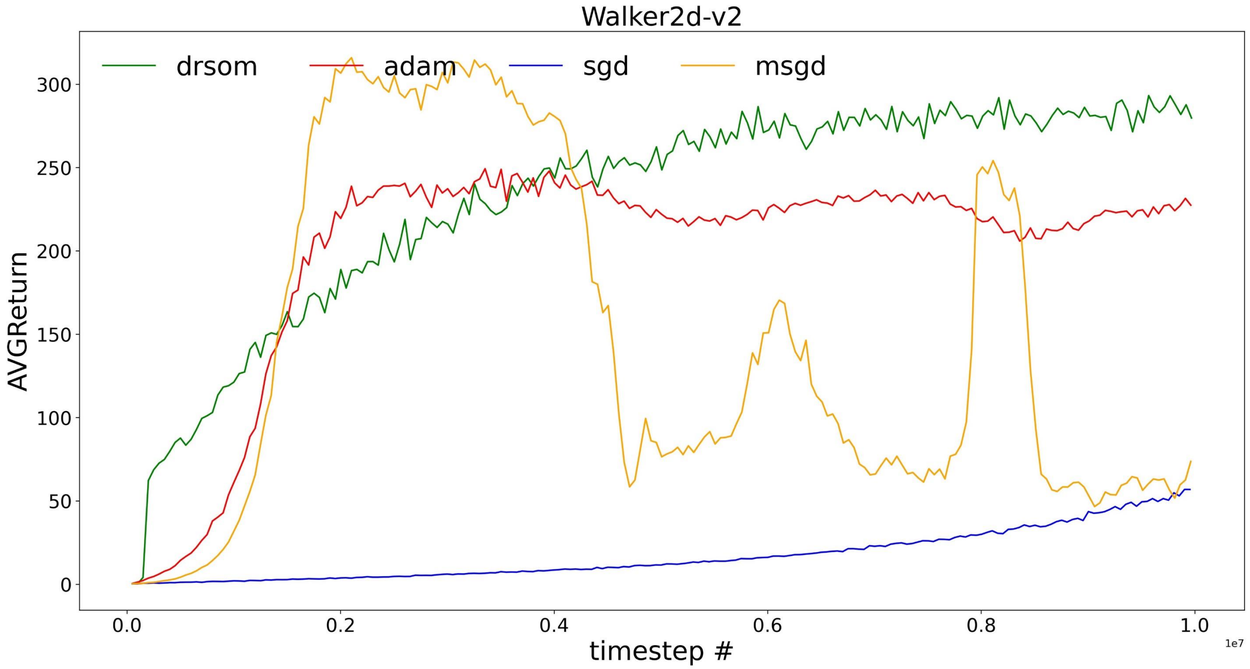
- With the estimated gradient, we can apply DRSOM to get the step size α, and update the parameter by:

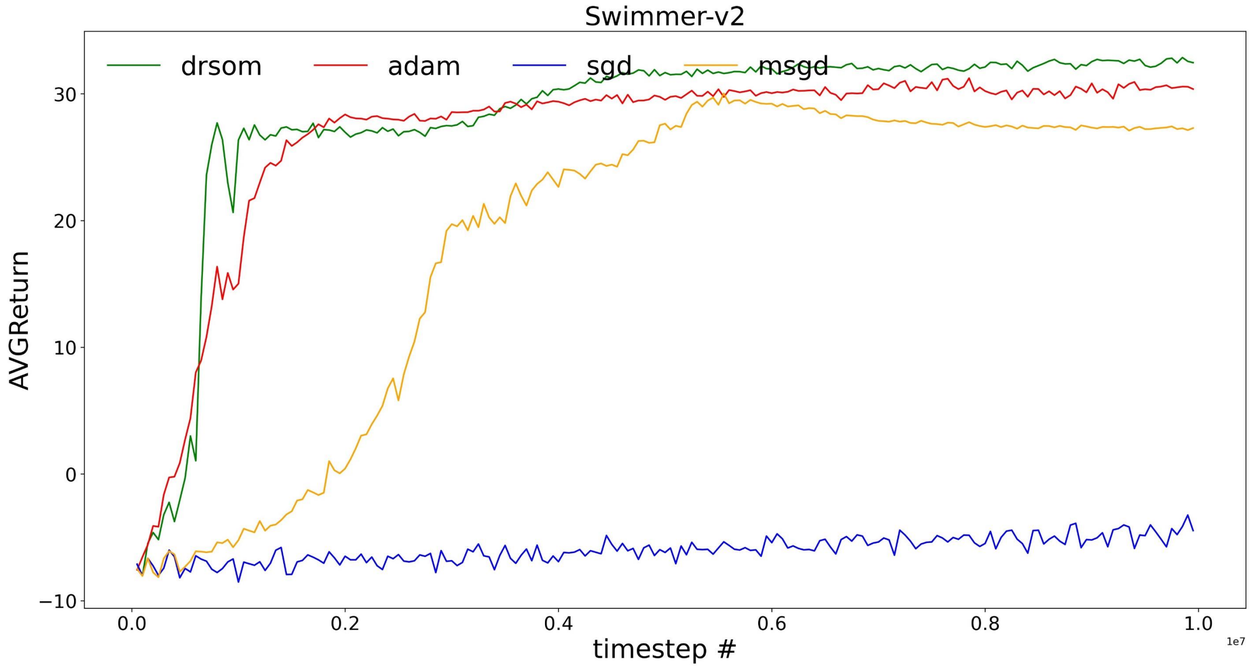
$$\theta_{t+1} = \theta_t + \alpha_t^1 \hat{\nabla} J(\theta_t) + \alpha_t^2 d_t$$

where d_t is the momentum direction.

DRSOM/ADAM/SGD Preliminary Results I

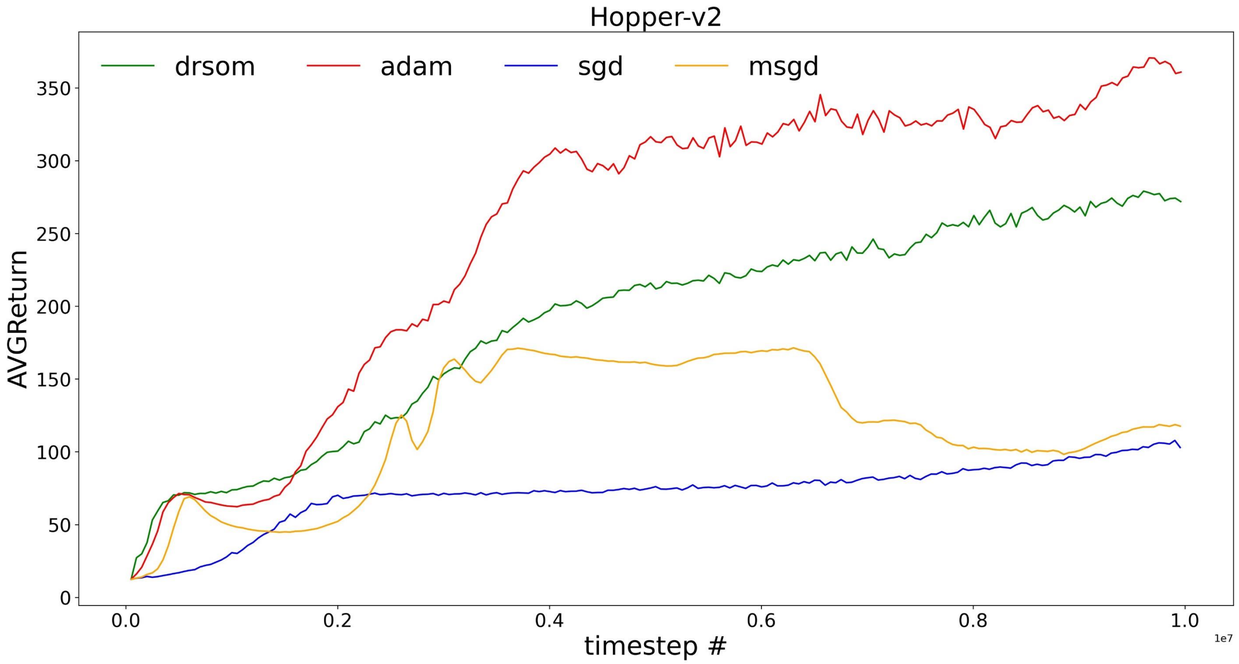
We compare the performance of DRSOM-based Reinforce with Adam-based reinforce and SGD-based reinforce(with(msgd) and without(sgd) momentum) on several GYM environments.
We set the learning rate of Adam and SGD both as 1e-3, and momentum of MSGD as 0.99

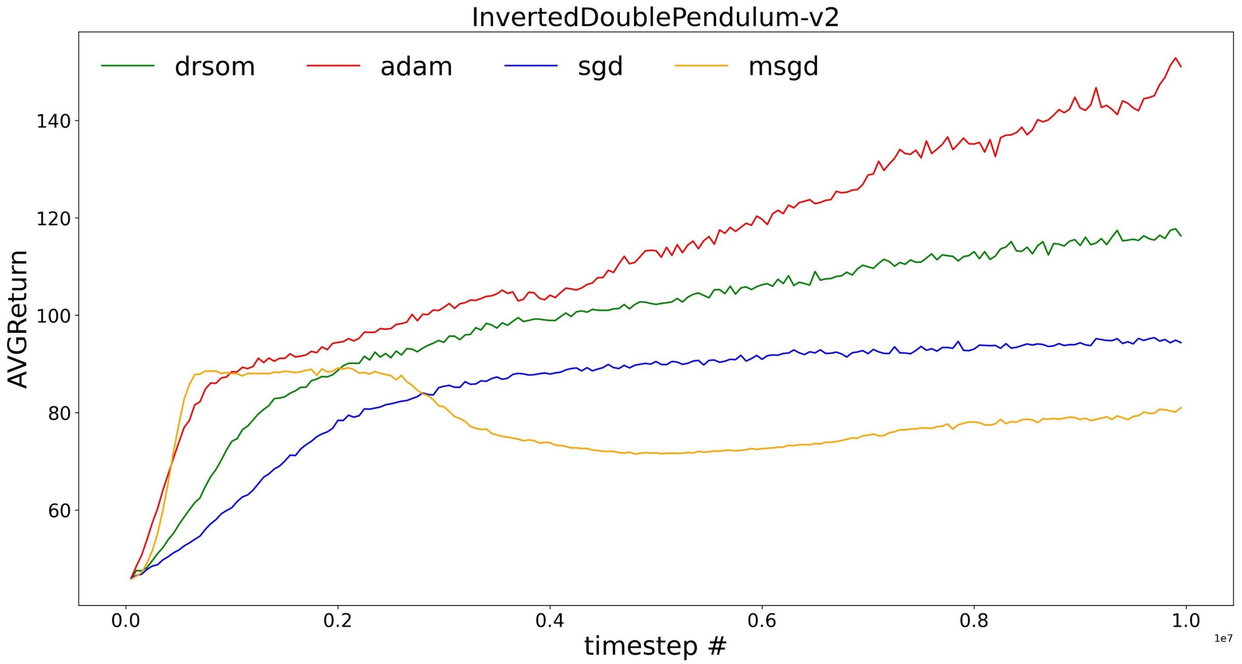




In these two cases, DRSOM converges faster and gain higher return than other algorithms. And also DRSOM seems to be more steady.

DRSOM/ADAM/SGD Preliminary Results II





In these two cases, DRSOM performs better than SGD but worse than ADAM.

DRSOM for TRPO I (Xue et al. SHUFE)

- **TRPO** attempts to optimize a surrogate function (based on the current iterate) of the objective function while keep a KL divergence constraint

$$\begin{aligned} \max_{\theta} \quad & L_{\theta_k}(\theta) \\ \text{s.t.} \quad & \mathrm{KL}\left(\mathrm{Pr}_{\mu}^{\pi_{\theta_k}} \| \mathrm{Pr}_{\mu}^{\pi_{\theta}}\right) \leq \delta \end{aligned}$$

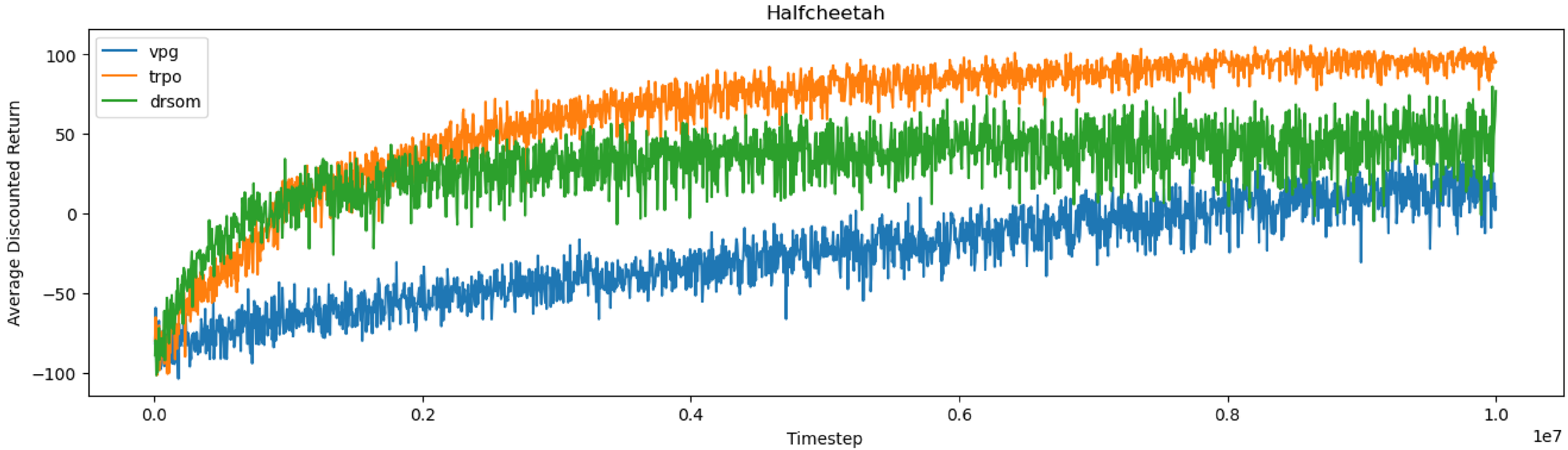
- In practice, it linearizes the surrogate function, quadratizes the KL constraint, and obtain

$$\begin{aligned} \max_{\theta} \quad & g_k^T(\theta - \theta_k) \\ \text{s.t.} \quad & \frac{1}{2}(\theta - \theta_k)^T F_k(\theta - \theta_k) \leq \delta \end{aligned}$$

where F_k is the Hessian of the KL divergence.

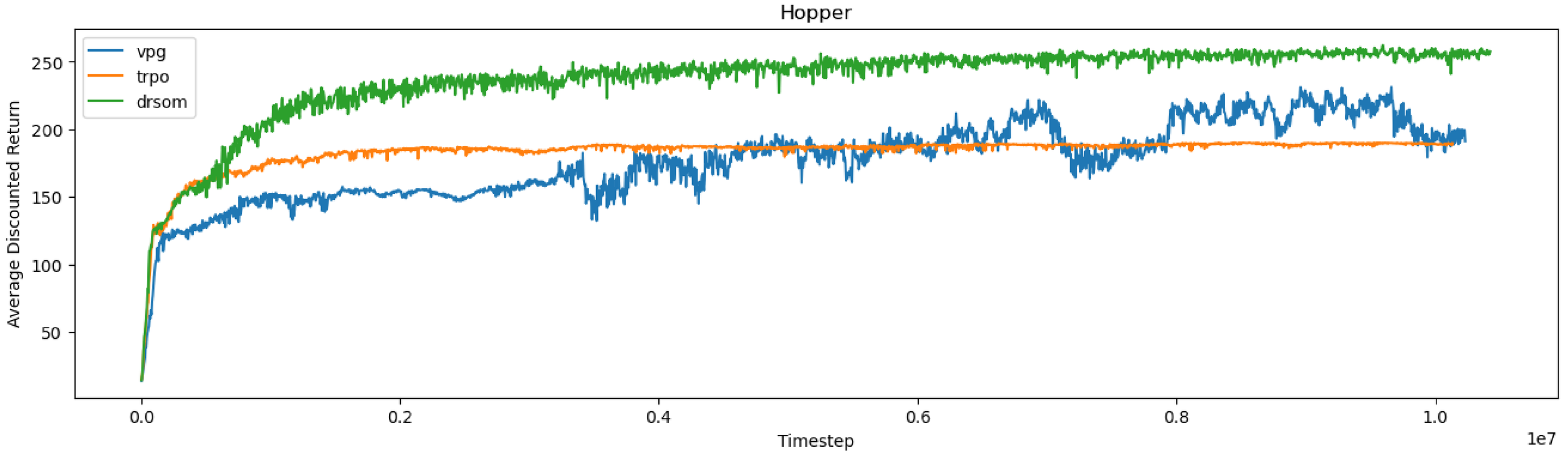
DRSOM/TRPO Preliminary Results I

- Although we only maintain the linear approximation of the surrogate function, surprisingly the algorithm works well in some RL environments

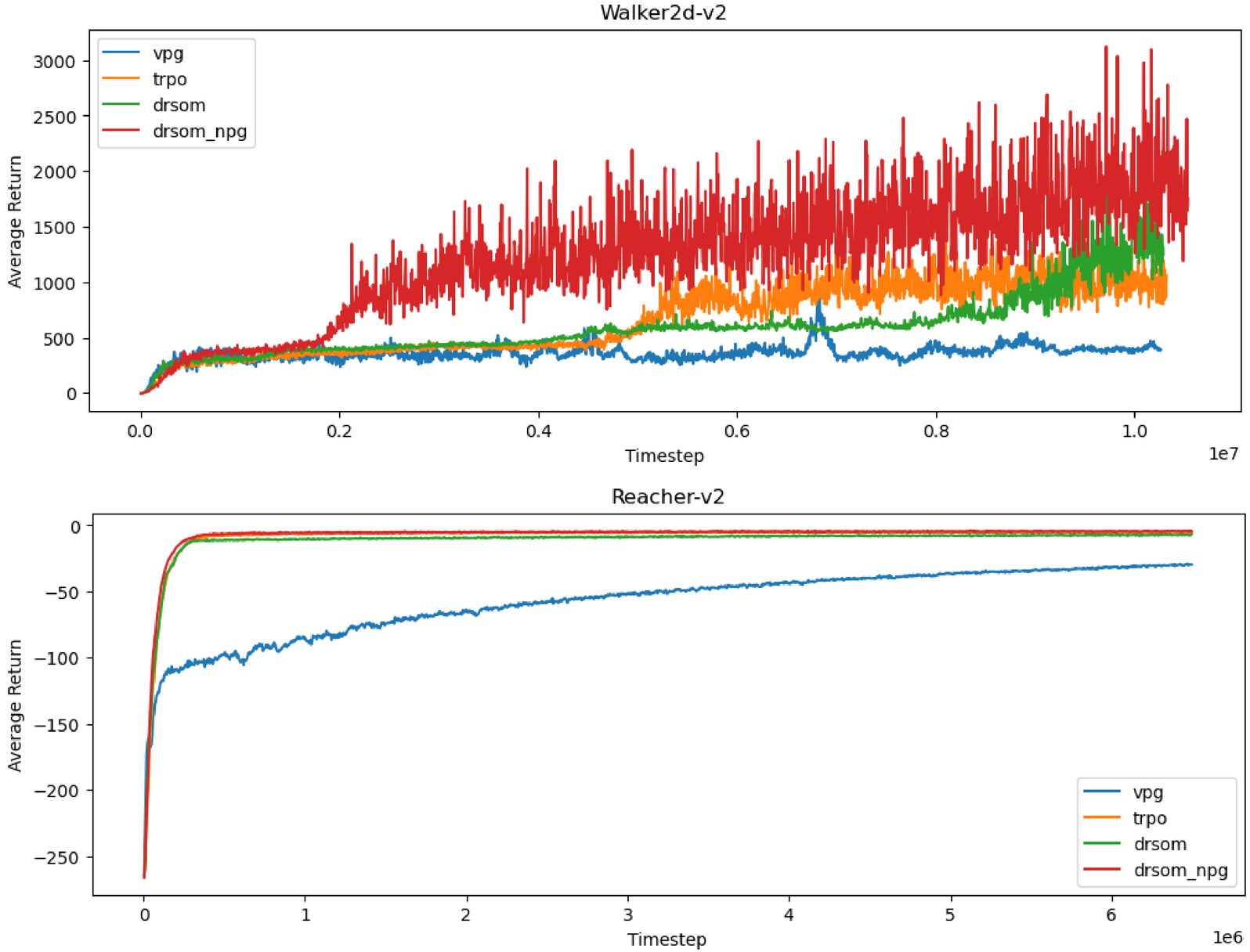


DRSOM/TRPO Preliminary Results II

- Sometimes even better than TRPO !



DRSOM/TRPO Preliminary Results III



DRSOM for Riemannian Optimization (Tang et al. NUS)

$$\min_{x\in\mathcal{M}} \quad f(x) \qquad \text{(ROP)}$$

- $\mathcal{M}$ is a Riemannian manifold embeded in Euclidean space $\mathbb{R}^n$.
- $f:\mathbb{R}^n\to\mathbb{R}$ is a second-order continuously differentiable function that is lower bounded in $\mathcal{M}$.

R-DRSOM: Choose an initial point $x_0\in\mathcal{M}$, set $k=0$, $p_{-1}=0$;
for $k=0,1,\ldots,T$ **do**
Step 1. Compute $g_k=\mathrm{grad}f(x_k)$, $d_k=\mathrm{T}_{x_k\leftarrow x_{k-1}}(p_{k-1})$, $H_kg_k=\mathrm{Hess}f(x_k)[g_k]$ and $H_kd_k=\mathrm{Hess}f(x_k)[d_k]$;

Step 2. Compute the vector $c_k=\begin{bmatrix}-\langle g_k,g_k\rangle_{x_k}\\ \langle g_k,d_k\rangle_{x_k}\end{bmatrix}$ and the following matrices

$$Q_k=\begin{bmatrix}\langle g_k,H_kg_k\rangle_{x_k} & \langle -d_k,H_kg_k\rangle_{k}\\ \langle -d_k,H_kg_k\rangle_{x_k} & \langle d_k,H_kd_k\rangle_{x_k}\end{bmatrix},\ ,\ G_k:=\begin{bmatrix}\langle g_k,g_k\rangle_{x_k} & -\langle d_k,g_k\rangle_{x_k}\\ -\langle d_k,g_k\rangle_{x_k} & \langle d_k,d_k\rangle_{x_k}\end{bmatrix}.$$

Step 3. Solve the following 2 by 2 trust region subproblem with radius $\triangle_k>0$

$$\alpha_k:=\arg\min_{\|\alpha_k\|_{G_k}\le\triangle_k} f(x_k)+c_k^\top\alpha+\frac{1}{2}\alpha^\top Q_k\alpha;$$

Step 4. $x_{k+1}:=\mathcal{R}_{x_k}\left(x_k-\alpha_k^1g_k+\alpha_k^2d_k\right)$;
end
Return x_k.

1D-Kohn-Sham Equation

$$\min \left\{ \frac{1}{2}\mathrm{tr}(R^\top L R) + \frac{\alpha}{4}\mathrm{diag}(RR^\top)^\top L^{-1}\mathrm{diag}(RR^\top) : \ R^\top R = I_p, \ R \in \mathbb{R}^{n\times r} \right\}, \qquad (3)$$

where L is a tri-diagonal matrix with 2 on its diagonal and -1 on its subdiagonal and $\alpha > 0$ is a parameter. We terminate algorithms when $\|\mathrm{grad} f(R)\| < 10^{-4}$.

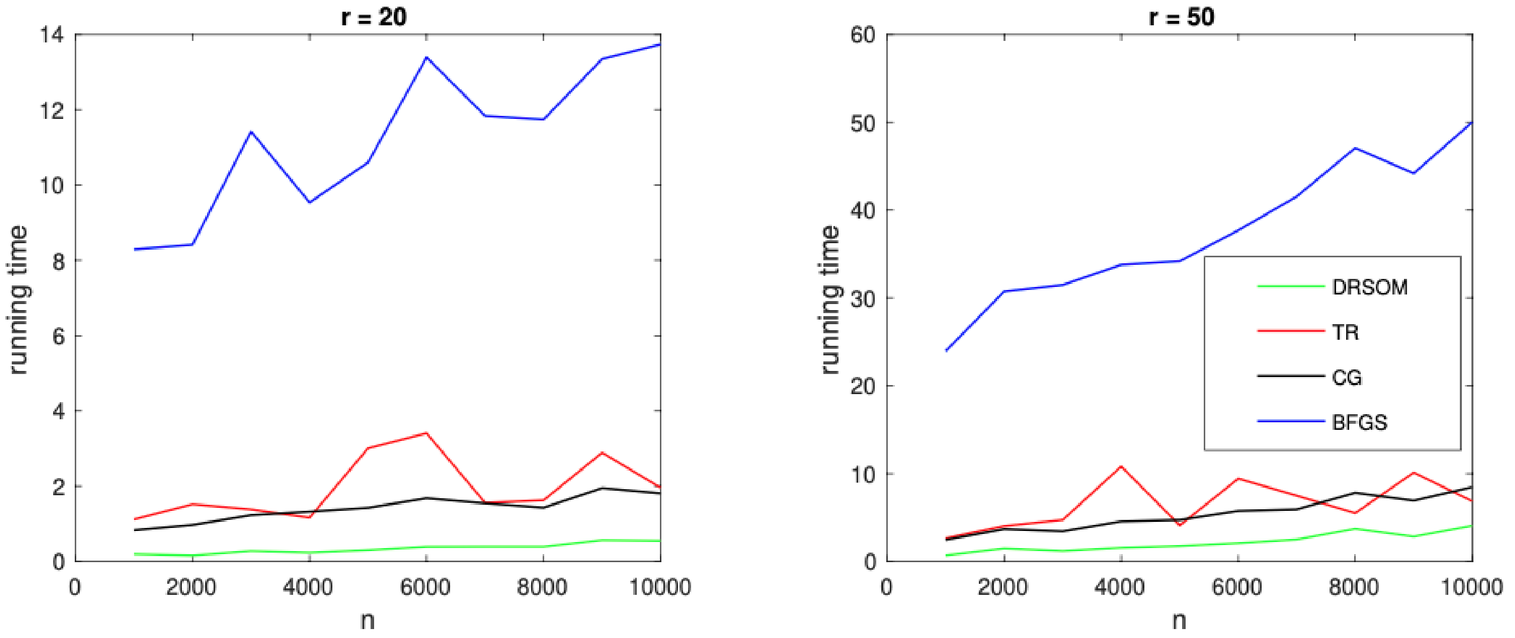


Figure 1: Results for Discretized 1D Kohn-Sham Equation. $\alpha = 1$.

Ongoing Research and Future Directions on DRSOM

- How to enforce or remove Assumption c) in algorithms/analyses
- How to design a more adaptive-radius mechanism with the same complexity bound, e.g., the trust-region framework of Curtis et al., 2017
- Incorporate the second-order steepest-descent direction, the eigenvector of the most negative Hessian eigenvalue
- Indefinite and Randomized Hessian rank-one updating vs BFGS
- Dimension Reduced Non-Smooth/Semi-Smooth Newton
- Dimension Reduced Second-Order Methods for optimization with more complicated constraints

Today’s Talk

- New developments of ADMM-based interior point (ABIP) Method
- Optimal Diagonal Preconditioner and HDSDP
- A Dimension Reduced Second-Order Method
- SOLNP+: a Derivative-Free Optimization Solver

Derivative-Free General Nonlinear Optimization

$$\begin{aligned} \min_{s\in\mathbb{R}^d} & f(s) \\ s.t. \quad & h_1(s) = 0, \\ & l_h \leqslant h_2(s) \leqslant u_h, \\ & l_s \leqslant s \leqslant u_s. \end{aligned}$$

Adding Slack Variables

$$\begin{aligned} \min_{x\in\mathbb{R}^d} & f(x) \\ s.t. \quad & g(x) = 0, \\ & l_x \leqslant x \leqslant u_x. \end{aligned}$$

- All functions are smooth functions.
- The solver only has access to zero-order information.
- Function evaluation may be expansive.
- There may be some noises in function evaluation.
- Many applications in real practice.

SOLNP+: Overview

- History
  - First proposed by Professor Ye in 1989.
  - Originally implemented (SOLNP) in Matlab, 1989.
  - R implementation (Rsolnp) by Alexios Ghalanos and Stefan Theussl, 2011.
  - New C implementation (SOLNP+) with improvements, 2022.
- Framework
  - Use finite difference to approximate the gradient.
  - Approximate the constraints by linear function.
  - Use Augmented Lagrangian Method (ALM) to solve the nonlinear constrained problem.
  - Use Sequential Quadratic Programming (SQP) and BFGS update to solve ALM subproblems.

SOLNP+ : Approximate Gradient and Constraints

- Use finite difference to calculate the approximated gradient.

$$[\nabla_\delta f(x)]_i = \frac{f(x+\delta e_i) - f(x)}{\delta}, \; e_i = [0, \; \cdots, 1, \; \cdots 0].$$

- Adaptively choose δ to increase robustness.
- Approximate the nonlinear constraints by linear function:

$$g(x) = 0 \quad \Rightarrow \quad g(x_k) + \nabla_{\delta_k} g(x_k)^T (x - x_k) = 0$$

SOLNP+ Outer Iteration: ALM Framework

- Modified Augmented Lagrangian function

$$L_k(x,y) = f(x) - y^T\left[g(x) - (g(x_k) + \nabla_{\delta_k} g(x_k)^T(x - x_k))\right] + \frac{\rho_k}{2}\left\|g(x) - (g(x_k) + \nabla_{\delta_k} g(x_k)^T(x - x_k))\right\|_2^2.$$

- Primal Update (Robinson, 1972):

$$\begin{aligned} \min\ & L_k(x, y_k) \\ \text{s.t. } & g(x_k) + \nabla_{\delta_k} g(x_k)^T(x - x_k) = 0, \\ & l_x \le x \le u_x, \end{aligned}$$

where y_k is the approximated Lagrange multiplier with respect to the linear constraints.

Solve ALM Subproblem: Find Feasible Solution

- The linearized problem may not be feasible.
- Find (approximated) feasible solution x_k^0 by solving the following LP.

$$
\begin{aligned}
&\min \ \tau \\
&\text{s.t. } g(x_k)(1-\tau) + \nabla_{\delta_k} g(x_k)^T (x - x_k) = 0, \\
&\quad l_x \le x \le u_x, \\
&\quad \tau \ge 0
\end{aligned}
$$

- When τ is small, we find a near feasible start point.
- Start from x_k^0, move along the direction that is in the null space of $\nabla_{\delta_k} g(x_k)^T$.

SOLNP+ Inner Iteration: SQP and BFGS Update

- SOLNP+ generates the following sequential quadratic programming (SQP) to solve the ALM subproblem.

$$\begin{aligned}\min\ & \frac{1}{2}(x-x_k^i)^T H_k^i (x-x_k^i) + \nabla_{\delta_k} L_k(x_k^i, y_k)^T (x-x_k^i) \\ \text{s.t. } & g(x_k) + \nabla_{\delta_k} g(x_k)^T (x-x_k) = b_k, \\ & l_x \le x \le u_x.\end{aligned}$$

where $b_k = g(x_k) + \nabla_{\delta_k} g(x^k)^T \left(x_k^0 - x_k\right)$ and BFGS update:

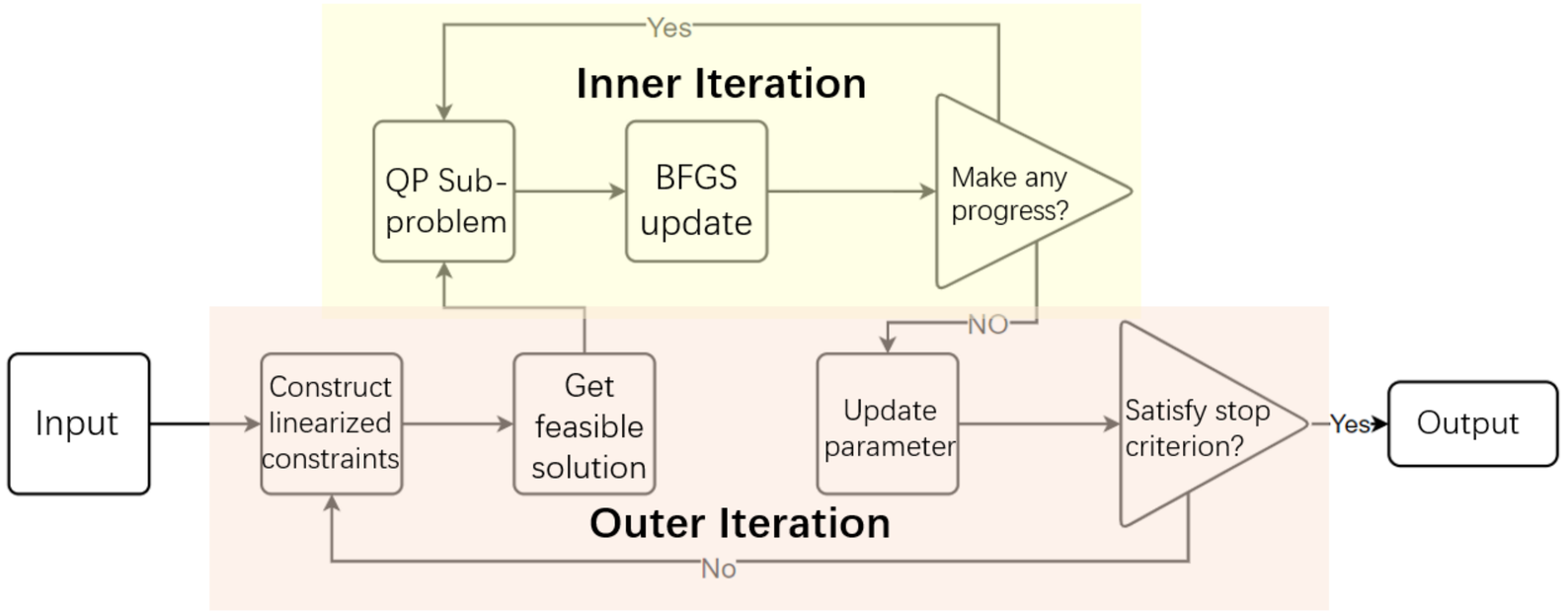
$$H_k^{i+1} = H_k^i + \frac{tt^T}{t^T s} - \frac{(H_k^i s)(H_k^i s)^T}{s^T H_k^i s}, \quad H_1^0 = I.$$

Where $t = \nabla_{\delta_k} L_k(x_k^{i+1}, y_k) - \nabla_{\delta_k} L_k(x_k^i, y_k)$ and $s = x_k^{i+1} - x_k^i$.

- Use Lagrange multiplier of linear constraints as an approximation to the real multiplier.

Computation Aspects for SOLNP+

- Heuristics to update the penalty parameter ρ_k.
- Restart when the algorithm cannot make any progress.
- Line search to improve quality of solution.
- Adaptively choose δ_k to increase robustness.



Computational Results: Functions without Noise

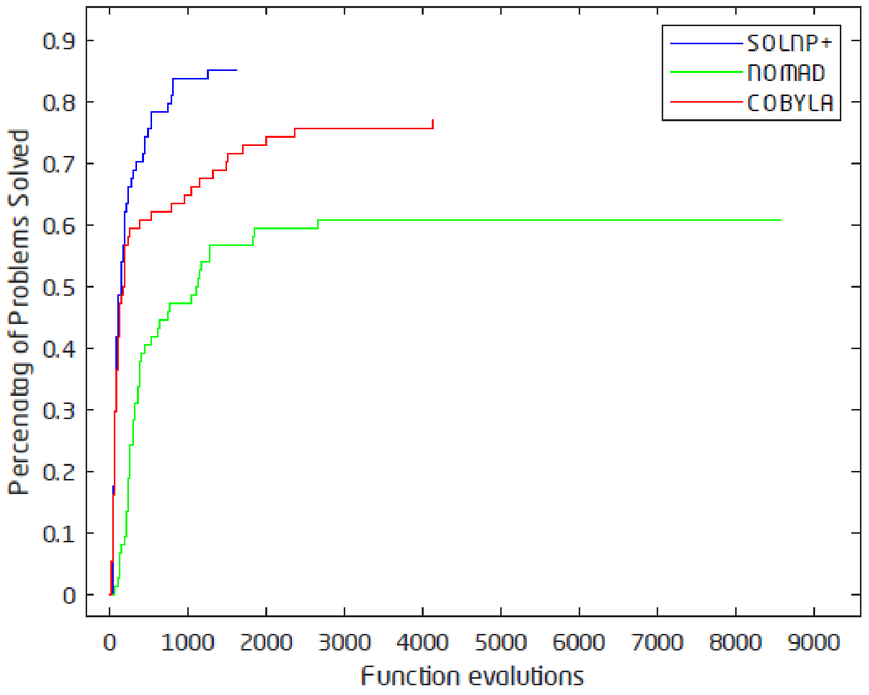


Figure 1: Test result of 74 problems in Hock and Schittkowski Hock and Schittkowski [1980] problems. Total running time of SOLNP+, NOMAD, COBYLA are 1.228696e+00s, 2.251209e+03s and 5.324220e+00s.

Prob.	Dim.	Number of Evaluations			Objective Function Value		
		SOLNP+	NOMAD	COBYLA	SOLNP+	NOMAD	COBYLA
hs11	2	41	312	53	-8.49787e+00	-8.49846e+00	-8.49846e+00
hs26	3	81	326	146	1.43427e-06	3.56000e+00	2.11600e+01
hs28	3	61	363	60	1.30568e-09	0.00000e+00	2.98619e-09
hs38	4	165	625	460	1.62759e-05	2.25010e-13	7.87702e+00
hs40	4	74	239	76	-2.50025e-01	-2.40655e-01	-2.50000e-01
hs46	5	272	252	537	4.30387e-09	3.33763e+00	9.24220e-06
hs56	7	158	383	263	-3.45603e+00	-1.00000e+00	-3.456161e+00
hs78	5	82	296	110	-2.91974e+00	2.73821e+00	-2.91970e+00
hs79	5	75	353	101	7.87804e-02	1.72669e-01	7.87768e-02
hs80	5	104	312	96	5.39484e-02	2.590245e-01	5.39499e-02
hs81	5	138	328	153	5.39470e-02	1.21224e-01	5.39499e-02
hs84	5	217	1818	108	-5.28034e+06	-5.28019e+06	-5.28033e+06
hs93	6	148	301	2367	1.35083e+02	1.36548e+02	1.35076e+02
hs106	8	530	2670	4000	7.08435e+03	7.66634e+03	8.94830e+03

Table 1 : Test results on selected Hock and Schittkowski problems. The blue color means that the solver returns an approximate optimal solution with better quality.

TM Ragonneau and Z Zhang. Pdfo: Cross-platform interfaces for powells derivative-free optimization solvers (version 1.1), 2021.

Le Digabel, Sébastien. "Algorithm 909: NOMAD: Nonlinear optimization with the MADS algorithm." *ACM Transactions on Mathematical Software (TOMS)* 37.4 (2011): 1-15.and Christophe Tribes.

Computational Results: Functions with Noise

- We consider the following problem,

$$\begin{aligned} \min_{x\in\mathbb{R}^d} \quad & f(x) \\ s.t. \quad & g(x) = 0, \\ & l_x \leqslant x \leqslant u_x. \end{aligned}$$

with observed value

$$\begin{aligned} \widehat{f}(x) &= f(x)(1+\sigma N_1(x)), \\ \widehat{g}(x) &= g(x)(1+\sigma N_2(x)) \end{aligned}$$

where $N_i(x) \sim N(0, I)\ i.i.d.$, $\sigma = 10^{-4}$.

- If the infeasibility of the point is less than 10^{-3}, we regard it as feasible point.

Prob.	Dim	Average Number of Evaluations				Average Objective Function Value			
		SOLNP	SOLNP+	NOMAD	COBYLA	SOLNP	SOLNP+	NOMAD	COBYLA
hs11	2	184.97	35.14	243.34	43.54	4.64631e+40(20/50)	-8.46861e+00	-8.49979e+00	-8.42549e+00
hs26	3	320.64	188.06	214.44	44.26	1.94628e+01	2.93551e-01(1/50)	3.36801e+00	2.11601e+01
hs28	3	37.76	58.54	318.52	60.68	9.46492e+00	4.26202e-07	2.72740e+00	1.67747e-04
hs38	4	42.44	224.44	770.76	261.38	6.87878e+03	8.45308e-01	2.039372e-10	7.93644e+00
hs40	4	388.33	45.58	183.88	67.14	-1.94623e-01(47/50)	-2.50324e-01	-2.39641e-01	-2.49996e-01
hs46	5	567.02	120.74	291.74	111.02	3.28695e+00	4.44609e-05	3.33753e+00	1.60205e+00
hs56	7	231.04	531.78	385.80	133.98	-1.00015e+00	-3.37944e+00	-9.99982e-01	-3.45015e+00(1/50)
hs78	5	–	118.60	211.52	73.58	(50/50)	-2.91860e+00	-2.74458e+00	-2.91955e+00
hs79	5	–	79.36	274.40	79.60	(50/50)	7.88079e-02	4.339176e+01	7.87789e-02
hs80	5	–	87.18	215.18	68.88	(50/50)	5.40269e-02	7.54378e-02	5.39545e-02
hs81	5	–	141.74	227.00	125.14	(50/50)	5.39633e-02	1.02641e-01	5.39499e-02
hs84	5	16.38	236.44	604.65	358.44	-2.49541e+06	-5.19516e+06	-5.26293e+06(30/50)	-5.24458e+06(41/50)
hs93	6	900.50	766.90	256.68	86.38	1.37050e+02	1.36190e+02	1.41290e+02	1.35922e+02
hs106	8	1036.24	581.98	1457.40	82.30	1.49873e+04(33/50)	1.50467e+04	7.79653e+03	1.49971e+04

Table 2: Test results with noise on selected Hock and Schittkowski Hock and Schittkowski [1980] problems. Each experiment is repeated 50 times. The blue color means that the solver returns a solution with better quality."(fail time/total time)" means the number of times for which the solvers return an infeasible solution. The average is taken for all the feasible solutions returned by the solver. Total test time of SOLNP, SOLNP+, NOMAD and COBYLA are 2.87135e-01, 1.59820e-01, 1.76231e+02 and 1.70019e-01 seconds.

Computational Results: Tumor Growth Problem

$$\min_{t_1,\cdots,t_n,a_1,\cdots,a_n} \quad P^* = P(t_{\text{end}}) + Q(t_{\text{end}}) + Q_P(t_{\text{end}})$$

$$\begin{aligned} s.t. \qquad & 0 \le t_i \le t_{\text{end}}, \qquad i = 1, \cdots n, \\ & 0 \le a_i \le 1, \qquad i = 1, \cdots n, \\ & 0 \le \max_{t\in[0,t_{\text{end}}]} C(t) \le v_{\max}, \\ & 0 \le \int_0^{t_{\text{end}}} C(t)\mathrm{d}t \le v_{\text{cum}}. \end{aligned}$$

At time t_i , we give drug of dosage a_i to the patient. P^* is the size of tumor at the end of the treatment. $C(t)$ is the drug concentration. They are calculated by solving an ODE.

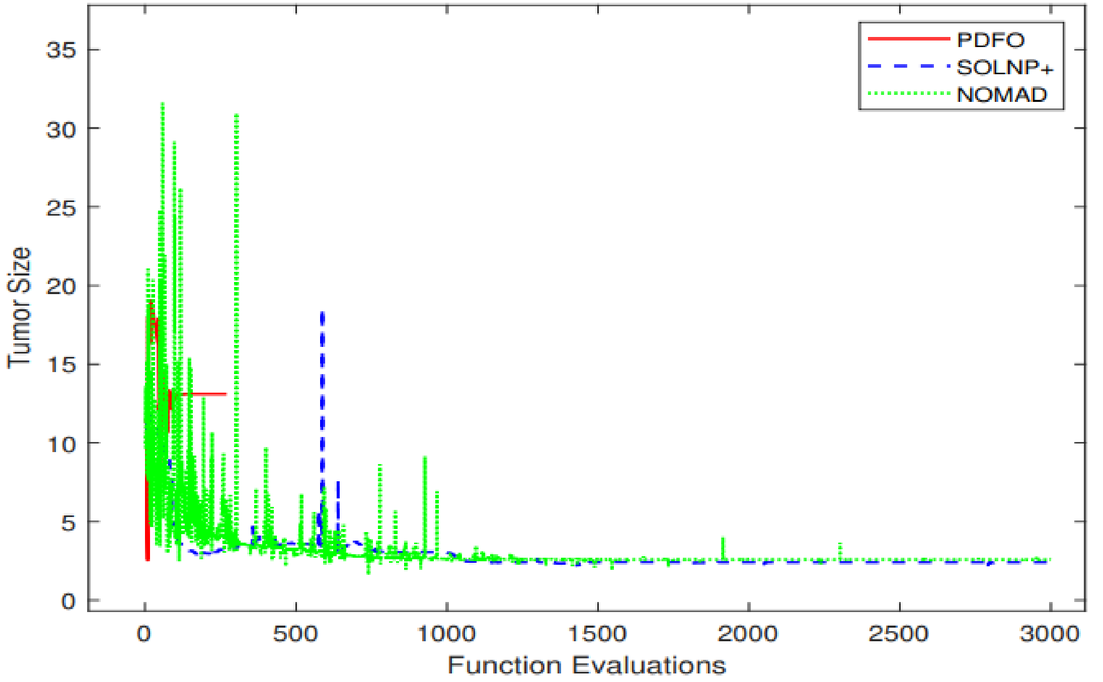


Figure 1: Convergence histories of the objective value.

Problem	Dim	Number of Evaluations			Objective Function Value		
		SOLNP+	NOMAD	COBYLA	SOLNP+	NOMAD	COBYLA
Tumor	8	3000	3000	270	2.41949e+00	2.57695e+00	1.31129e+01

Infeasibility			Running Time/s		
SOLNP+	NOMAD	COBYLA	SOLNP+	NOMAD	COBYLA
5.31037e-09	0.00000e+00	0.00000e+00	5.06375e+00	3.28349e+01	9.16734e-01

Table 3: Final output in the tumor problem of three solvers.

Summary of SOLNP+

- Able to make use of dual information.
- Provide estimation of both primal and dual solutions.
- Faster speed in small problems.
- Robust under noise.
- Nonconvex QP sub-problem solver

Takeaways

Algorithm customization is necessary
Second-order information matters
View optimization iterative process as an online learning process

- **THANK YOU**