

Trust-Region for ill-conditioned low-rank problems

Irène Waldspurger

CNRS et CEREMADE (Université Paris Dauphine)
Équipe MOKAPLAN (INRIA)

May 31, 2024

Journée thématique

Low-rank approximation and optimization

Institut de mathématiques de Marseille



Paul Caucheteux



Florentin Goyens



Clément Royer

(all at Paris Dauphine)

This is an ongoing work. Results are preliminary.

Main question

We consider a family of problems :

semidefinite problems, with a low-rank solution,
in Burer-Monteiro factorized form.

(All terms will be subsequently defined.)

These problems and **non-convex** and **ill-conditioned**.

Question : which algorithms best solve these problems ?

Scientific contribution

- ▶ We review and numerically compare several algorithms, on problems with various difficulties.
- ▶ We numerically find that the best-performing algorithm is a **second-order method**.
 - Interest : except in very small dimension, second-order methods are oftentimes criticized for being slow.
We exhibit medium-dimensional problems where they actually perform very well.
- ▶ In a simplified setting, we establish **convergence rates** to justify the good numerical behavior.

Roadmap

1. Definition of the considered problems.
2. Description of the main algorithms :
 - ▶ first-order methods ;
 - ▶ second-order methods (Trust Region).
3. Numerical results.
4. Theoretical convergence rates.

Semidefinite problems, with a low-rank solution,
in Burer-Monteiro factorized form.

Semidefinite problems, with a low-rank solution,
in Burer-Monteiro factorized form.

Let $f : \mathbb{R}^{n \times n} \rightarrow \mathbb{R}$ be convex.

A general **semidefinite problem** is

$$\begin{aligned} &\text{minimize } f(X), \\ &\text{over } X \in \mathbb{R}^{n \times n}, \\ &\quad X \succeq 0, \end{aligned} \tag{SDP}$$

Semidefinite problems, **with a low-rank solution**,
in Burer-Monteiro factorized form.

Let $f : \mathbb{R}^{n \times n} \rightarrow \mathbb{R}$ be convex.

A general semidefinite problem is

$$\begin{aligned} & \text{minimize } f(X), \\ & \text{over } X \in \mathbb{R}^{n \times n}, \\ & \quad X \succeq 0, \end{aligned} \tag{SDP}$$

“with a low-rank solution” : assume there exists a minimizer X_* , with $\text{rank } r_* \ll n$.

One motivation : phase retrieval

Reconstruct $x \in \mathbb{C}^n$
from $y = (|\langle x, v_i \rangle|)_{i \leq m}$

Here,

- ▶ v_1, \dots, v_m are known *measurement vectors* ;
- ▶ $|\cdot|$ is the complex modulus.

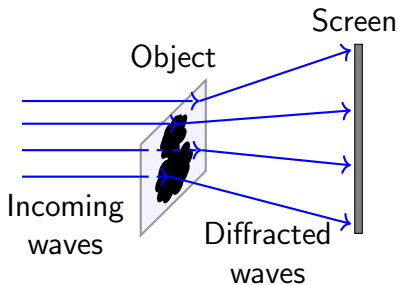


Figure – X-ray imaging

[Schechtman, Eldar, Cohen, Chapman, Miao, and Segev, 2015]

One motivation : phase retrieval

Phase retrieval problems are **non-convex**.

They are **difficult** (variants of them are NP-hard).

Several families of algorithms exist.

→ I am especially interested in **convex relaxations**.

One motivation : phase retrieval

Phase retrieval problems are **non-convex**.

They are **difficult** (variants of them are NP-hard).

Several families of algorithms exist.

→ I am especially interested in **convex relaxations**.

Principle of convex relaxations :

approximate the non-convex problem with a convex one,
which is a **semidefinite problem with a low-rank solution**.

- ▶ Advantage : avoid local minima due to non-convexity.
- ▶ Drawback : high-dimensional ; computationally costly.

Phase retrieval : convex relaxation

$$\begin{aligned} &\text{Find } x \in \mathbb{C}^n \\ &\text{s.t. } \forall i, y_i = |\langle x, v_i \rangle|. \end{aligned}$$

[Candès, Strohmer, and Voroninski, 2013; Chai, Moscoso, and Papanicolaou, 2011]

Phase retrieval : convex relaxation

$$\begin{array}{ll} \text{Find } x \in \mathbb{C}^n \\ \text{s.t. } \forall i, y_i = |\langle x, v_i \rangle|. \end{array}$$



$$\begin{array}{ll} \text{Find } xx^* \in \mathbb{C}^{n \times n} \\ \text{s.t. } \forall i, y_i^2 = |\langle x, v_i \rangle|^2 \\ \quad = \text{Tr}(xx^* v_i v_i^*). \end{array}$$

[Candès, Strohmer, and Voroninski, 2013; Chai, Moscoso, and Papanicolaou, 2011]

Phase retrieval : convex relaxation

$$\begin{array}{ll} \text{Find } x \in \mathbb{C}^n \\ \text{s.t. } \forall i, y_i = |\langle x, v_i \rangle|. \end{array}$$



$$\begin{array}{ll} \text{Find } xx^* \in \mathbb{C}^{n \times n} \\ \text{s.t. } \forall i, y_i^2 = |\langle x, v_i \rangle|^2 \\ \quad = \text{Tr}(xx^* v_i v_i^*). \end{array}$$



$$\begin{array}{ll} \text{Find } X \in \mathbb{C}^{n \times n} \\ \text{s.t. } \forall i, y_i^2 = \text{Tr}(X v_i v_i^*), \\ \quad X \succeq 0, \\ \quad \text{rank}(X) = 1. \end{array}$$

[Candès, Strohmer, and Voroninski, 2013; Chai, Moscoso, and Papanicolaou, 2011]

Phase retrieval : convex relaxation

$$\begin{aligned} &\text{Find } x \in \mathbb{C}^n \\ &\text{s.t. } \forall i, y_i = |\langle x, v_i \rangle|. \end{aligned}$$



$$\begin{aligned} &\text{Find } xx^* \in \mathbb{C}^{n \times n} \\ &\text{s.t. } \forall i, y_i^2 = |\langle x, v_i \rangle|^2 \\ &\quad = \text{Tr}(xx^* v_i v_i^*). \end{aligned}$$



$$\begin{aligned} &\text{Find } X \in \mathbb{C}^{n \times n} \\ &\text{s.t. } \forall i, y_i^2 = \text{Tr}(X v_i v_i^*), \\ &\quad X \succeq 0, \\ &\quad \cancel{\text{rank}(X) = 1.} \end{aligned}$$

Remove the rank constraint.

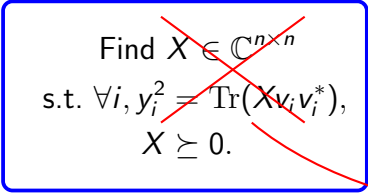
→ We obtain a convex
approximate problem.

[Candès, Strohmer, and Voroninski, 2013; Chai, Moscoso, and Papanicolaou, 2011]

Phase retrieval : convex relaxation

$$\begin{aligned} &\text{Find } X \in \mathbb{C}^{n \times n} \\ &\text{s.t. } \forall i, y_i^2 = \text{Tr}(X v_i v_i^*), \\ &\quad X \succeq 0. \end{aligned}$$

Phase retrieval : convex relaxation


$$\begin{array}{ll}\text{Find } X \in \mathbb{C}^{n \times n} \\ \text{s.t. } \forall i, y_i^2 = \text{Tr}(X v_i v_i^*), \\ X \succeq 0.\end{array}$$

$$\begin{array}{ll}\min & \sum_{i=1}^m (y_i^2 - \text{Tr}(X v_i v_i^*))^2 \\ \text{over } & X \in \mathbb{C}^{n \times n}\end{array}$$

equivalent to

Phase retrieval : convex relaxation

$$\begin{aligned} & \text{Minimize} \quad \sum_{i=1}^m (y_i^2 - \text{Tr}(X v_i v_i^*))^2 \\ & \quad \text{over } X \in \mathbb{C}^{n \times n} \\ & \text{such that } X \succeq 0. \end{aligned}$$

This is a (convex) **semidefinite problem**.

Phase retrieval : convex relaxation

$$\begin{aligned} & \text{Minimize } \sum_{i=1}^m (y_i^2 - \text{Tr}(X v_i v_i^*))^2 \\ & \text{over } X \in \mathbb{C}^{n \times n} \\ & \text{such that } X \succeq 0. \end{aligned}$$

This is a (convex) **semidefinite problem**.

Approximation of the non-convex problem, not equivalent, but often has the same solution :

$$X_* = x x^*.$$

Consequence : the minimizer **has low-rank**.

Back to SDP with low-rank solution

$$\begin{aligned} & \text{minimize } f(X), \\ & \text{over } X \in \mathbb{R}^{n \times n}, \\ & \quad X \succeq 0, \end{aligned}$$

We have described one specific motivation (phase retrieval).

- ▶ Advantage : avoid local minima due to non-convexity.
- ▶ Drawback : high-dimensional ; computationally costly.

Now, how do we numerically solve these problems ?

SDP with low-rank solution : algorithms

1. Generic SDP solvers, which **do not exploit low-rank**.
→ High per-iteration complexity.
2. Solvers **tailored to low-rank settings**
 - 2.1 Generic SDP solvers, where some parts are made much more efficient by the low-rank assumption ;
[Ding, Yurtsever, Cevher, Tropp, and Udell, 2019]
[Yurtsever, Tropp, Fercoq, Udell, and Cevher, 2021]
 - 2.2 Burer-Monteiro factorization.
[Burer and Monteiro, 2003]

SDP with low-rank solution : algorithms

1. Generic SDP solvers, which **do not exploit low-rank**.
→ High per-iteration complexity.
2. Solvers **tailored to low-rank settings**
 - 2.1 Generic SDP solvers, where some parts are made much more efficient by the low-rank assumption ;
[Ding, Yurtsever, Cevher, Tropp, and Udell, 2019]
[Yurtsever, Tropp, Fercoq, Udell, and Cevher, 2021]
 - 2.2 **Burer-Monteiro factorization**.
[Burer and Monteiro, 2003]

Burer-Monteiro factorization : principle

$$\begin{aligned} &\text{Minimize } f(X), \\ &\text{over } X \in \mathbb{R}^{n \times n}, \\ &\quad X \succeq 0. \end{aligned} \tag{SDP}$$

The solution X_* has low-rank r_* . For arbitrary $p \geq r_*$, write

$$X_* = V_* V_*^T \quad \text{for some } V_* \in \mathbb{R}^{n \times p}.$$

→

$$\begin{aligned} &\text{Minimize } f(VV^T), \\ &\text{over } V \in \mathbb{R}^{n \times p}. \end{aligned} \tag{BM}$$

Burer-Monteiro factorization : principle

$$\begin{array}{l} \text{Minimize } f(VV^T), \\ \text{over } V \in \mathbb{R}^{n \times p}. \end{array} \quad (\text{BM})$$

This factorized problem is equivalent to the SDP (i.e. the solution is the same, up to change of variable), but the dimension is **much lower**.

This is a **semidefinite problem**, with a **low-rank solution**, in Burer-Monteiro factorized form.

Question : which algorithm to solve it ?

What is difficult in Problem (BM)?

$$\begin{array}{l} \text{Minimize } f(VV^T), \\ \text{over } V \in \mathbb{R}^{n \times p}. \end{array}$$

(BM)

First obstacle : (BM) is non-convex (again).
In particular, it can have local minima.

What is difficult in Problem (BM) ?

$$\begin{array}{l} \text{Minimize } f(VV^T), \\ \text{over } V \in \mathbb{R}^{n \times p}. \end{array} \quad (\text{BM})$$

First obstacle : (BM) is non-convex (again).

In particular, it can have local minima.

Luckily, oftentimes, it does not : non-convexity is **benign**.

In our experiments, we do not seem to encounter this issue.

What is difficult in Problem (BM) ?

$$\begin{array}{l} \text{Minimize } f(VV^T), \\ \text{over } V \in \mathbb{R}^{n \times p}. \end{array}$$

(BM)

First obstacle : (BM) is non-convex (again).

In particular, it can have local minima.

Luckily, oftentimes, it does not : non-convexity is **benign**.

In our experiments, we do not seem to encounter this issue.

Second obstacle : (BM) is ill-conditioned.

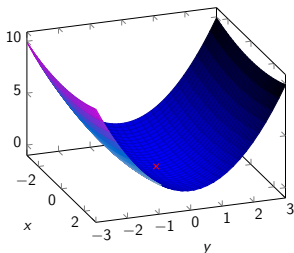
→ For our problems, this is the main issue.

Problem (BM) is ill-conditioned.

$$\begin{array}{l} \text{Minimize } f(VV^T), \\ \text{over } V \in \mathbb{R}^{n \times p}. \end{array}$$

(BM)

Ill-conditioning = in the neighborhood of the minimizer,
smallest Hessian eigenvalue \ll largest eigenvalue.



First source of ill-conditioning : f

$$\begin{array}{l} \text{Minimize } g(V) \stackrel{\text{def}}{=} f(VV^T), \\ \text{over } V \in \mathbb{R}^{n \times p}. \end{array} \quad (\text{BM})$$

At a minimizer V_* ,

$$\nabla^2 g_{V_*}(S, S) = \langle \nabla^2 f_{V_* V_*^T}(V_* S^T + S V_*^T), V_* S^T + S V_*^T \rangle.$$

If f is ill-conditioned, g is likely to be ill-conditioned.

Second source of ill-conditioning : $p > r_*$

$$\begin{array}{l} \text{Minimize } g(V) \stackrel{\text{def}}{=} f(VV^T), \\ \text{over } V \in \mathbb{R}^{n \times p}. \end{array} \quad (\text{BM})$$

At a minimizer V_* ,

$$\nabla^2 g_{V_*}(S, S) = \langle \nabla^2 f_{V_* V_*^T}(V_* S^T + S V_*^T), V_* S^T + S V_*^T \rangle.$$

Assume $p > r_* = \text{rank}(V_*)$ (**overparametrized** case).

There are $S \neq 0$ such that

$$V_* S^T + S V_*^T = 0.$$

\Rightarrow The Hessian is **singular**.

Algorithms for Problem (BM)

1. Methods tailored to particular f ,
e.g. the *Mixing Method* [Wang, Chang, and Kolter, 2017].
(Do not apply in phase retrieval.)
2. First-order methods :
 - 2.1 standard,
 - 2.2 with preconditioning.
3. Second-order methods.

First-order methods : standard

$$\begin{aligned} \text{Minimize } g(V) &\stackrel{\text{def}}{=} f(VV^T), \\ \text{over } V &\in \mathbb{R}^{n \times p}. \end{aligned}$$

These methods produce a sequence of iterates, computed using ∇f .

Simplest example : [gradient descent](#).

$$V_{k+1} = V_k - \tau \nabla g(V_k) = V_k - 2\tau \nabla f(V_k V_k^T) V_k$$

Many other possibilities : Nesterov, Augmented Lagrangian ...

[Chen and Goulart, 2023]

[Monteiro, Sujarani, and Cifuentes, 2024]

First-order methods : standard

$$\begin{aligned} \text{Minimize } g(V) &\stackrel{\text{def}}{=} f(VV^T), \\ \text{over } V &\in \mathbb{R}^{n \times p}. \end{aligned}$$

These methods are known to be very sensitive to ill-conditioning.

First-order methods : preconditioned

[Tong, Ma, and Chi, 2021]

[Zhang, Fattahi, and Zhang, 2021]

Assume ill-conditioning is **due to overparametrization only**.

Principle : there is a simple explicit description of the near-zero Hessian eigenvectors.

→ correct for these eigenvalues with a suitable preconditioner
 \approx change of metric.

Basic version :

$$V_{k+1} = V_k - 2\tau \nabla f(V_k V_k^T) \underbrace{V_k (V_k^T V_k)^{-1}}_{\text{preconditioner}}.$$

Second-order methods : general picture

These methods produce a sequence of iterates, computed using ∇f and $\nabla^2 f$.

- ▶ Advantage : explicitly using the Hessian allows to detect and correct ill-conditioning.
- ▶ Drawback : the Hessian is typically a big matrix.
→ high computational cost ?

In our case, we use the Hessian through a few matrix-vector computations only, and these can be efficiently computed.

→ not slower than first-order methods.

Second-order methods : Trust-Region

$$\begin{aligned} \text{Minimize } g(V) &\stackrel{\text{def}}{=} f(VV^T), \\ \text{over } V &\in \mathbb{R}^{n \times p}. \end{aligned}$$

Close to V_k , g can be approximated as

$$g(V_k + S) \approx g(V_k) + \langle \nabla g(V_k), S \rangle + \frac{1}{2} \langle \nabla^2 g_{V_k}(S), S \rangle.$$

Second-order methods : Trust-Region

$$\begin{aligned} \text{Minimize } g(V) &\stackrel{\text{def}}{=} f(VV^T), \\ \text{over } V &\in \mathbb{R}^{n \times p}. \end{aligned}$$

Close to V_k , g can be approximated as

$$g(V_k + S) \approx g(V_k) + \langle \nabla g(V_k), S \rangle + \frac{1}{2} \langle \nabla^2 g_{V_k}(S), S \rangle.$$

Principle of Trust-Region : minimize this second-order approximation .

$$\text{Set } V_{k+1} = V_k + S,$$

$$\text{with } S \in \operatorname{argmin} g(V_k) + \langle \nabla g(V_k), S \rangle + \frac{1}{2} \langle \nabla^2 g_{V_k}(S), S \rangle.$$

Second-order methods : Trust-Region

$$\begin{aligned} \text{Minimize } g(V) &\stackrel{\text{def}}{=} f(VV^T), \\ \text{over } V &\in \mathbb{R}^{n \times p}. \end{aligned}$$

Close to V_k , g can be approximated as

$$g(V_k + S) \approx g(V_k) + \langle \nabla g(V_k), S \rangle + \frac{1}{2} \langle \nabla^2 g_{V_k}(S), S \rangle.$$

Principle of Trust-Region : minimize this second-order approximation **over a small ball**.

$$\text{Set } V_{k+1} = V_k + S,$$

$$\text{with } S \in \underset{\|S\|_F \leq \Delta_k}{\operatorname{argmin}} g(V_k) + \langle \nabla g(V_k), S \rangle + \frac{1}{2} \langle \nabla^2 g_{V_k}(S), S \rangle.$$

Second-order methods : Trust-Region

At each iteration, we must solve

$$\operatorname{argmin}_{\|S\|_F \leq \Delta_k} g(V_k) + \langle \nabla g(V_k), S \rangle + \frac{1}{2} \langle \nabla^2 g_{V_k}(S), S \rangle.$$

- ▶ Exact solvers exist, but are costly.
- ▶ Inexact solvers are much faster.
→ We use **Truncated Conjugate Gradient**.

Typical complexity : $O(\underbrace{np \log(n)}_{\text{cost of a gradient computation}} \times \underbrace{\text{nb its}}_{\text{adapts to problem difficulty}})$

cost of a gradient computation

adapts to problem
difficulty

Algorithm 1 Trust-Region

- 1: **for** $k = 1, 2, \dots$ **do**
- 2: Using Truncated Conjugate Gradient, approximate

$$S \in \operatorname{argmin}_{\|S\|_F \leq \Delta_k} g(V_k) + \langle \nabla g(V_k), S \rangle + \frac{1}{2} \langle \nabla^2 g_{V_k}(S), S \rangle.$$

- 3: **if** $g(V_k + S)$ is small enough **then**
 - 4: Set $V_{k+1} = V_k + S$.
 - 5: **else**
 - 6: Set $V_{k+1} = V_k$.
 - 7: **end if**
 - 8: Increase or decrease Δ_k .
 - 9: **end for**
-

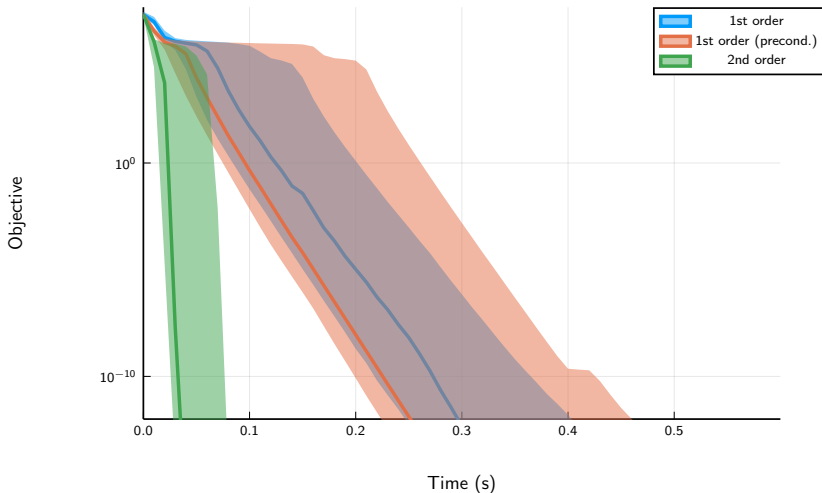
Overview of numerical results

We compare the algorithms on SDP coming from phase retrieval.

	Well-conditioned	Ill-conditioned (overparam.)	Ill-conditioned (overparam. + f)
Standard 1st order	fast	slow	slow
Precond. 1st order	fast	fast	slow
Trust-Region	fast	fast	faster

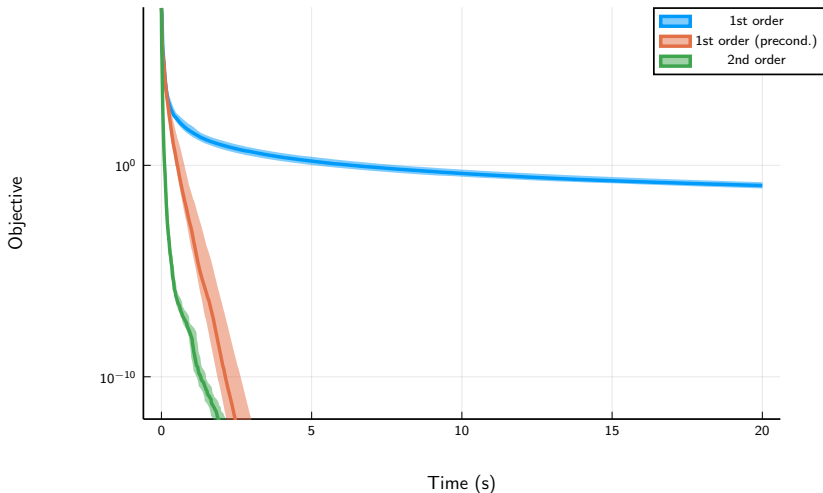
Graph 1 : well-conditioned problem

$n = 128, m = 768$, random measurements, *PhaseLift*, $p = 1$



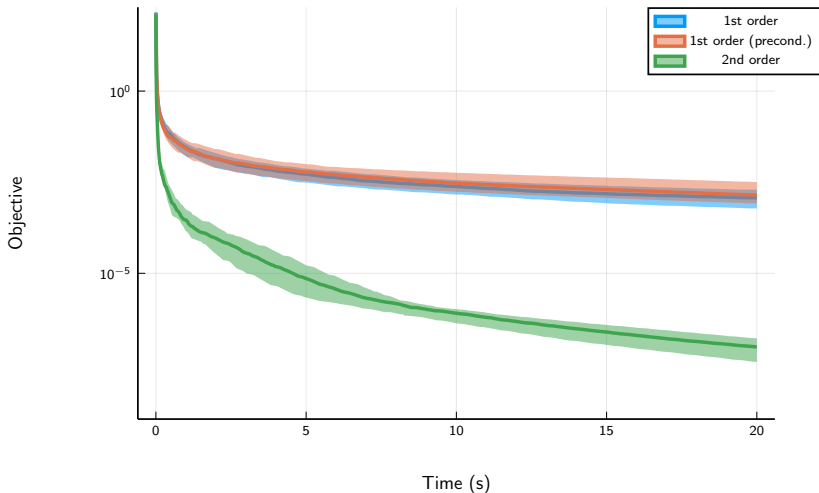
Graph 2 : ill-conditioned (overparametrization)

$n = 128, m = 768$, random measurements, *PhaseLift*, $p = 2$



Graph 3 : ill-conditioned (overparametrized and f)

$n = 128, m = 768$, wavelet measurements, *PhaseLift*, $p = 2$



Summary of the numerical experiments

- ▶ First-order methods struggle with ill-conditioning.
- ▶ Preconditioned first-order methods are immune to ill-conditioning from overparametrization.
- ▶ Second-order methods perform best when ill-conditioning comes from both overparametrization and f .

What's left to do on the numerical side

- ▶ Increase the dimension.
- ▶ Implement and test other available first-order methods.
- ▶ Investigate possible improvements for Trust-Region :
 - ▶ rank-adaptive strategies ;
 - ▶ mix with a more basic algorithm.

Theoretical guarantees ?

$$\begin{array}{l} \text{Minimize } f(VV^T), \\ \text{over } V \in \mathbb{R}^{n \times p}. \end{array} \quad (\text{BM})$$

Justify the good numerical behavior of Trust-Region through rigorous convergence rates ?

We focus on the case where ill-conditioning comes from overparametrization only.

→ Show that Trust-Region performs at least as well as preconditioned first-order methods ?
(i.e. local linear convergence rate, same iteration cost)

Prior work

Convergence rates for Trust-Region when minimizing a non-convex function g ?

$$\text{Minimize } g(x) \text{ for } x \in \mathbb{R}^n.$$

First result : Trust-Region converges locally superlinearly if

$$\nabla^2 g(x_{\min}) \succ 0.$$

This holds for an exact subproblem solver,
and for the inexact Truncated Conjugate Gradient.

[Absil, Baker, and Gallivan, 2007]

Prior work

Convergence rates for Trust-Region when minimizing a non-convex function g ?

$$\text{Minimize } g(x) \text{ for } x \in \mathbb{R}^n.$$

First result : Trust-Region converges locally superlinearly if

$$\nabla^2 g(x_{\min}) \succ 0.$$

$\Rightarrow g$ locally convex
(and precludes
non-isolated minima)

This holds for an exact subproblem solver,
and for the inexact Truncated Conjugate Gradient.

[Absil, Baker, and Gallivan, 2007]

Does not apply for us!

Prior work

Convergence rates for Trust-Region when minimizing a non-convex function g ?

$$\text{Minimize } g(x) \text{ for } x \in \mathbb{R}^n.$$

Second result : Trust-Region converges locally superlinearly if

g satisfies a *Polyak–Łojasiewicz condition* close to x_{\min} .

This holds for the Truncated Conjugate Gradient solver.

[Rebjock and Boumal, 2023]

Prior work

Convergence rates for Trust-Region when minimizing a non-convex function g ?

Minimize $g(x)$ for $x \in \mathbb{R}^n$.

Second result : Trust-Region converges locally superlinearly if

g satisfies a *Polyak–Łojasiewicz condition* close to x_{\min} .

This holds for the Truncated Conjugate Gradient solver.

[Rebjock and Boumal, 2023]

Does not apply for us
when $p > r_*$.

Our result

$$\begin{array}{l} \text{Minimize } f(VV^T), \\ \text{over } V \in \mathbb{R}^{n \times p}. \end{array} \quad (\text{BM})$$

For the moment, we consider the **simplest possible** function f which is **well-conditioned** and has a **low-rank minimizer** :

$$f : X \in \mathbb{R}^{n \times n} \rightarrow \frac{1}{4} \left\| X - \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & & & \\ \vdots & \ddots & & \\ 0 & & & 0 \end{pmatrix} \right\|_F^2.$$

We hope that the result generalizes to other well-conditioned functions.

Our result

Theorem : (with ongoing proof)

Close to the solution, Trust-Region converges linearly :
for some $C_0 > 0$, $\rho \in]0; 1[$ and all $k \in \mathbb{N}$,

$$f(V_k V_k^T) \leq C_0 \rho^k f(V_0 V_0^T),$$

This holds for an exact subproblem solver,
and for the inexact Truncated Conjugate Gradient,
capped at 3 iterations.
(\rightarrow same cost as first-order methods.)

Proof idea

We consider a specific function f , and the Trust-Region iterates are deterministic.

⇒ We should be able to **explicitly compute** V_{k+1} as a function of V_k .

Proof idea

We consider a specific function f , and the Trust-Region iterates are deterministic.

⇒ We should be able to **explicitly compute** V_{k+1} as a function of V_k .

Actually, the direct computation is horrible, and the expression we get for V_{k+1} is difficult to study.

Proof idea

In the exact solver case :

We introduce a suitable Lyapunov function.

We divide the space of V_k in different regions.

In each region, we exploit the optimality conditions of the subproblem to show that the Lyapunov function decays.

Proof idea

Using Truncated Conjugate Gradient :

We divide the space of V_k in different regions.

In each region, we either exploit the optimality conditions of the subproblem to show linear decay of f ,

or we estimate the result of the first three iterations of Conjugate Gradient.

Thank you very much for your attention !