Polynomial Preconditioning for Gradient Methods

Nikita Doikov [1] **Anton Rodomanov** [2]

Abstract

We study first-order methods with preconditioning for solving structured nonlinear convex optimization problems. We propose a new family of preconditioners generated by symmetric polynomials. They provide first-order optimization methods with a provable improvement of the condition number, cutting the gaps between highest eigenvalues, without explicit knowledge of the actual spectrum. We give a stochastic interpretation of this preconditioning in terms of coordinate volume sampling and compare it with other classical approaches, including the Chebyshev polynomials. We show how to incorporate a polynomial preconditioning into the Gradient and Fast Gradient Methods and establish the corresponding global complexity bounds. Finally, we propose a simple adaptive search procedure that automatically chooses the best possible polynomial preconditioning for the Gradient Method, minimizing the objective along a low-dimensional Krylov subspace. Numerical experiments confirm the efficiency of our preconditioning strategies for solving various machine learning problems.

1. Introduction

Motivation. Preconditioning is an important tool for improving the performance of numerical algorithms. The classical example is the preconditioned *Conjugate Gradient Method* (Hestenes & Stiefel, 1952) for solving a system of linear equations. It proposes to modify the initial system in a way to improve its eigenvalue distribution and thus to accelerate the convergence of the method. The question of choosing the right preconditioner heavily depends on the problem structure, and there exist many problem-specific recommendations which provide us with a good trade-off between computational cost and the spectrum properties of the new system. Some notable examples include *Jacobi* or the *diagonal* preconditioners, *symmetric successive over-relaxation*, the *incomplete Cholesky* factorization (Golub & Van Loan, 2013), *Laplacian* preconditioning for graph problems (Vaidya, 1991; Spielman & Teng, 2004), preconditioners for discretizations of system of *partial differential equations* (Mardal & Winther, 2011).

Another important class of numerical algorithms are the second-order methods or *Newton's Method* (see, e.g. (Nesterov, 2018)), that aims to solve difficult ill-conditioned problems by using local curvature information (the Hessian matrix) as a preconditioner at every step. However, being a powerful algorithm, each iteration of Newton's Method is very expensive. It requires to solve a system of linear equations with the Hessian matrix, and in case of quadratic objective it is equivalent to solving the original problem.

In this paper, our goal is to solve a general *nonlinear optimization problem* with a structured convex objective by the efficient first-order methods. Thus, in the case of unconstrained minimization of a smooth function: $\min_{\boldsymbol{x}} f(\boldsymbol{x})$, the simplest method that we study is as follows, for $k \geq 0$:

$$\boldsymbol{x}_{k+1} \quad = \quad \boldsymbol{x}_k - \alpha_k \boldsymbol{P} \nabla f(\boldsymbol{x}_k), \tag{1}$$

where $\alpha_k > 0$ is a stepsize and $\boldsymbol{P}$ is a fixed preconditioning matrix. $\boldsymbol{P} := \boldsymbol{I}$ corresponds to the classical gradient descent. Another natural choice is $\boldsymbol{P} := \boldsymbol{B}^{-1}$, where $\boldsymbol{B}$ is a *curvature matrix* of our problem[1], which is directly available for the algorithm. That resembles the Newton-type direction, and the method with this preconditioner tends to converge much faster in practice (see Figure 1). However, computing $\boldsymbol{B}^{-1}$ (or solving the corresponding linear system with $\boldsymbol{B}$) is a very expensive operation in the large scale setting.

Instead of using $\boldsymbol{B}^{-1}$, we propose a new family of *Symmetric Polynomial Preconditioners*, that provably improve the spectrum of the objective. The first member of our family is

$$\boldsymbol{P} \quad := \quad \operatorname{tr}(\boldsymbol{B})\boldsymbol{I} - \boldsymbol{B}. \tag{2}$$

We prove that using preconditioner (2) within method (1), makes the condition number *insensitive to the gap between*

[1]EPFL, Switzerland [2]UCLouvain, Belgium. Correspondence to: Nikita Doikov <nikita.doikov@epfl.ch>, Anton Rodomanov <anton.rodomanov@uclouvain.be>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

[1]See the definition of $\boldsymbol{B}$ in our Assumption 2.1 and the corresponding Examples 2.2, 2.3, 2.4 of different problems.

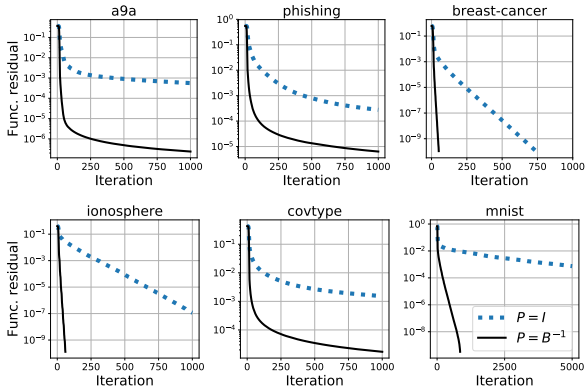


Figure 1: Training logistic regression with the standard gradient descent ($\boldsymbol{P} = \boldsymbol{I}$), and using the inverse of the curvature matrix ($\boldsymbol{P} = \boldsymbol{B}^{-1}$) as a preconditioner in (1). The latter method works much faster, while it can be very expensive to compute $\boldsymbol{B}^{-1}$ for large scale problems.

the top two eigenvalues of the curvature matrix. Since it is quite common for real data to have a highly nonuniform spectrum with several large gaps between the top eigenvalues (see Figure 2), our preconditioning can significantly improve the convergence of the first-order methods. At the same time, one step of the form (1),(2) is still cheap to compute. It involves just the standard matrix operations (trace and the matrix-vector product), without the need to solve linear systems with the curvature matrix as in Newton's Method.

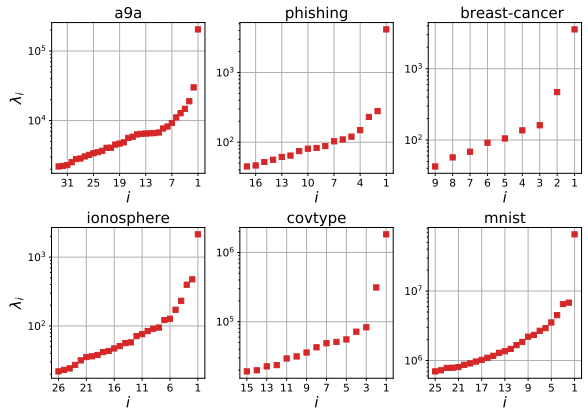


Figure 2: Leading eigenvalues (in the logarithmic scale) of the curvature matrix $\boldsymbol{B}$, for several typical datasets[2]. There are large gaps between the top eigenvalues.

This approach works for general structured nonlinear problems (*not necessarily quadratics*) and also for the problems with possible *composite* parts (e.g., constrained minimization or non-smooth regularization).

[2] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

Our new *family* of Symmetric Polynomial Preconditioners gradually interpolate between the first preconditioner (2) and $\boldsymbol{P} \propto \boldsymbol{B}^{-1}$ as the other extreme case. We show that increasing the order of the preconditioner, we are able to cut off *several* top eigenvalues of the curvature matrix, without knowing the actual spectrum. We can incorporate these preconditioners both into the Gradient Method, as well as into the accelerated Fast Gradient Method (Nesterov, 1983), with a further provable improvement of the condition number.

Finally, we address the common question of choosing the best possible preconditioner. We propose a new adaptive strategy for the basic nonlinear Gradient Method based on the *Krylov subspace* minimization. In this approach, preconditioner $\boldsymbol{P}$ is defined as a general polynomial of the curvature matrix $\boldsymbol{B}$ of a fixed (small) degree τ:

$$\boldsymbol{P} \;\; := \;\; a_0 \boldsymbol{I} + a_1 \boldsymbol{B} + \ldots + a_\tau \boldsymbol{B}^\tau,$$

where the vector of coefficients $\boldsymbol{a} \in \mathbb{R}^{\tau+1}$ is found by solving a certain linear system of size $\tau+1$ in each iteration of the method. It has a plain interpretation of projecting the direction $\boldsymbol{B}^{-1}\nabla f(\boldsymbol{x}_k)$ onto an affine set $\mathcal{K}^\tau_{\boldsymbol{x}_k}$, which is the Krylov subspace:

$$\mathcal{K}^\tau_{\boldsymbol{x}} \;\stackrel{\text{def}}{=}\; \operatorname{span}\left\{\nabla f(\boldsymbol{x}),\, \boldsymbol{B}\nabla f(\boldsymbol{x}), \ldots, \boldsymbol{B}^\tau \nabla f(\boldsymbol{x})\right\}. \tag{3}$$

In case of small τ, we can solve this linear system easily and obtain the best preconditioning guarantee for our method, which is *adaptive* for each iteration.

Related Work. It is widely known that the standard Conjugate Gradient Method is *optimal* in the class of the first-order algorithms for unconstrained minimization of convex *quadratic* functions (Nemirovski, 1995). The kth iteration of the Conjugate Gradients finds the full minimum of the objective over the k-dimensional Krylov subspace, and thus it provably solves the problem after $k = n$ iterations, where n is the dimension of the problem. Quadratic minimization is equivalent to solving a system of linear equations, therefore it is often referred as the *linear* case. Polynomial preconditioning for solving large linear systems has been extensively studied during the last several decades; see (Dubois et al., 1979; Johnson et al., 1983; Saad, 1985; Van Gijzen, 1995; Liu et al., 2015; Loe & Morgan, 2022) and references therein. See also Section 5.3 for the comparison of our preconditioning strategies with the linear Conjugate Gradient Method.

The situtation with *nonlinear* problems is more difficult. Along with the basic Gradient Method, the classical approaches include the Nonlinear Conjugate Gradients and Quasi-Newton Methods (see, e.g. (Nocedal & Wright, 2006)), which typically demonstrate a decent practical performance, while replicating the standard Conjugate Gradients in the linear case. However, these methods *lack* of

	Preconditioner	Condition number, β/α	Methods	Cost
Classical Gradient Method	$\boldsymbol{P} = \boldsymbol{I}$	λ_1/λ_n	GM, FGM	**cheap**
"Full Preconditioning"	$\boldsymbol{P} = \boldsymbol{B}^{-1}$	1	GM, FGM	**expensive**
Symmetric Polynomial Preconditioning **(ours)**	$\boldsymbol{P} = \boldsymbol{P}_\tau$ (11)	$\lambda_1/\lambda_n \cdot \xi_\tau(\boldsymbol{\lambda})$	GM, FGM	**cheap for small τ**
Krylov Subspace Minimization **(ours)**	optimal polynom.	$\lambda_{\tau+1}/\lambda_n$	GM	**cheap for small τ**

Table 1: The value β/α for different preconditioning strategies, $\boldsymbol{\lambda} = \boldsymbol{\lambda}(\boldsymbol{B})$. Note that $\xi_\tau(\boldsymbol{\lambda}) \leq 1$, and $\xi_\tau(\boldsymbol{\lambda}) \to 0$ in case of large spectral gaps, namely when $\lambda_1/\lambda_{\tau+1} \to \infty$ (see Section 4). For solving the problem with $\epsilon > 0$ accuracy, GM needs $k(\epsilon) = \mathcal{O}(\beta/\alpha \cdot 1/\epsilon)$ and $k(\epsilon) = \mathcal{O}(\beta/\alpha \cdot L/\mu \cdot \log 1/\epsilon)$ iterations for convex and strongly convex functions correspondingly. FGM needs only $\sqrt{k(\epsilon)}$ iterations (Theorems 3.1 and 3.2).

having any good *global complexity bounds*, and thus in the worst-case scenario they can actually perform even worse than the Gradient Method (Gupta et al., 2023). At the same time, the Fast Gradient Method developed by (Nesterov, 1983) is *optimal* for the class of nonlinear problems with a *uniformly bounded eigenvalues* of the Hessian (Nemirovski & Yudin, 1983). This assumption does not take into account the actual distribution of the spectrum. Hence, it can not distinguish the problems with large gaps between the top eigenvalues, as in Figure 2.

There have been several attempts to study more specific problem formulations, and so to gain a provable advantage for the optimization algorithms by leveraging the spectrum of the Hessian. Thus, the quadratic minimization problems were studied under the assumption of a particular *probability distribution* for the eigenvalues (Scieur & Pedregosa, 2020; Cunha et al., 2022), or assuming a certain fixed *spectral gap* (Goujaud et al., 2022), revealing the advantages of employing the Heavy-ball Method (Polyak, 1987) in these cases. Another example is the Stochastic Spectral Descent (Kovalev et al., 2018), which improves the condition number for quadratic problems if we know some of the eigenvectors.

In this work, we consider a refined smoothness characterization of the objective with the curvature matrix $\boldsymbol{B}$ (Assumption 2.1). It is similar in spirit to that one used in Stochastic Dual Newton Ascent (Qu et al., 2016). An important particular instance of this class of algorithms is the Randomized Coordinate Descent with *Volume Sampling* (Rodomanov & Kropotov, 2020). In the latter method, it was proposed to select subsets of variables of certain size m proportionally to the determinants of principal submatrices of $\boldsymbol{B}$. While this approach was practically implementable only for the subsets of size $m = 1$ or 2, it was shown that, in theory, the method is insensitive to the large spectral gap between the top $m - 1$ eigenvalues.

Surprisingly, our new family of the Symmetric Polynomial Preconditioners can be viewed as a *deterministic version* of the Volume Sampling technique (with $m = \tau + 1$ where τ is the degree of a preconditioning polynomial; preconditioner (2) corresponds to $\tau = 1$). Thus, we provide the Volume Sampling with a novel deterministic interpretation, which also leads to new accelerated and composite optimization algorithms (see Section 4.3 for a detailed comparison).

Contributions. We propose several polynomial preconditioning strategies for first-order methods for solving a general composite convex optimization problem, and prove their better global complexity guarantees, specifically:

- We study the convergence of the basic Gradient Method (GM, Algorithm 1) and the accelerated Fast Gradient Method (FGM, Algorithm 2) with a general (arbitrarily fixed) preconditioning matrix. We introduce *two* condition numbers, that are designated to the different parts of the objective (L/μ for nonlinearity and β/α for the curvature matrix), and show that they serve as main complexity factors.
- We develop a new family of Symmetric Polynomial Preconditioners (Section 4). Combining them with the preconditioned Gradient Methods, we establish a significant improvement of the curvature condition number β/α in case of *large gaps* between the top eigenvalues of the matrix (see Table 1).
- Then, we propose a new adaptive procedure based on the Krylov subspace minimization (Algorithm 3) that achieves the *best polynomial* preconditioning. We present the guarantees we can get, including cutting off the top eigenvalues directly and by employing the Chebyshev polynomials, and compare this approach with the Symmetric Polynomial Preconditioning.
- Numerical experiments are provided.

2. Notation and Assumptions

We consider the following optimization problem given in the *composite* form:

$$F^\star = \min_{\boldsymbol{x} \in \mathbb{R}^n} \Big\{ F(\boldsymbol{x}) \stackrel{\text{def}}{=} f(\boldsymbol{x}) + \psi(\boldsymbol{x}) \Big\}, \qquad (4)$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a differentiable convex function which is the *main* part of the problem, and $\psi : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is a proper closed convex function that can be nondifferentiable but has a *simple* structure. For example, it can be an indicator of a convex set, or ℓ_1-regularizer.

Additionally, we fix some symmetric positive-definite matrix $\boldsymbol{B} \in \mathbb{R}^{n\times n}$ (notation $\boldsymbol{B} = \boldsymbol{B}^\top \succ 0$). This matrix plays the key role in our characterization of the smoothness properties of f. Namely, we assume the following (considering for simplicity two-times differentiable functions):

Assumption 2.1. *The Hessian of f is uniformly bounded, for some constants $L \geq \mu \geq 0$:*

$$\mu \boldsymbol{B} \preceq \nabla^2 f(\boldsymbol{x}) \preceq L\boldsymbol{B}, \qquad \forall \boldsymbol{x} \in \mathbb{R}^n. \quad (5)$$

Having fixed the matrix $\boldsymbol{B}$, we define the corresponding induced norm by $\|\boldsymbol{x}\|_{\boldsymbol{B}} \stackrel{\text{def}}{=} \langle \boldsymbol{B}\boldsymbol{x}, \boldsymbol{x}\rangle^{1/2}$, $\boldsymbol{x} \in \mathbb{R}^n$. Thus, matrix $\boldsymbol{B}$ is responsible for fixing the coordinate system in the problem. Then, condition (5) can be rewritten in terms of the global lower and upper bound on the first-order approximation of f (Nesterov, 2018):

$$\begin{aligned} \tfrac{\mu}{2}\|\boldsymbol{y}-\boldsymbol{x}\|_{\boldsymbol{B}}^2 &\leq f(\boldsymbol{y}) - f(\boldsymbol{x}) - \langle \nabla f(\boldsymbol{x}), \boldsymbol{y}-\boldsymbol{x}\rangle \\ &\leq \tfrac{L}{2}\|\boldsymbol{y}-\boldsymbol{x}\|_{\boldsymbol{B}}^2, \qquad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n. \end{aligned} \quad (6)$$

In what follows, we denote by $\boldsymbol{\lambda} = \boldsymbol{\lambda}(\boldsymbol{B}) \in \mathbb{R}^n$ the vector of eigenvalues for the matrix $\boldsymbol{B}$, sorted in a nonincreasing order: $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$.

The classical example is $\boldsymbol{B} := \boldsymbol{I}$ (identity matrix). Then, condition (5) implies that the function f is (strongly) convex and has the Lipschitz continuous gradient. However, by choosing a specific $\boldsymbol{B}$, we tend to achieve a better granularity of the description of our problem class and thus to improve the convergence properties of the methods.

Example 2.2. *Let $\boldsymbol{a} \in \mathbb{R}^n$. Then, the quadratic function*

$$f(\boldsymbol{x}) = \tfrac{1}{2}\langle \boldsymbol{B}\boldsymbol{x}, \boldsymbol{x}\rangle - \langle \boldsymbol{a}, \boldsymbol{x}\rangle,$$

satisfies condition (5) with $L = \mu = 1$.

We see that in this case, the so-called *condition number* L/μ is just 1, which means that preconditioning the Gradient Method (1) with the matrix $\boldsymbol{P} := \boldsymbol{B}^{-1}$ would give an immediate convergence to the solution. However, inverting the matrix is prohibitively expensive for large scale problems. Our aim is to find a suitable trade-off between improving the condition number and the arithmetic cost of algorithm steps. Let us consider the following important example which can be met in many practical applications.

Example 2.3. *Let $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ be a given data matrix, and $\boldsymbol{b} \in \mathbb{R}^m$ be a given vector. Denote,*

$$f(\boldsymbol{x}) = g(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b})$$

Then, the derivatives are as follows: $\nabla f(\boldsymbol{x}) = \boldsymbol{A}^\top \nabla g(\boldsymbol{A}\boldsymbol{x}+\boldsymbol{b})$ and $\nabla^2 f(\boldsymbol{x}) = \boldsymbol{A}^\top \nabla^2 g(\boldsymbol{A}\boldsymbol{x}+\boldsymbol{b})\boldsymbol{A}$. Hence, assuming: $\mu \boldsymbol{I_m} \preceq \nabla^2 g(\boldsymbol{x}) \preceq L\boldsymbol{I_m}$, $\forall \boldsymbol{x}$, with some $L \geq \mu \geq 0$, condition (5) is satisfied [3] with

$$\boldsymbol{B} := \boldsymbol{A}^\top \boldsymbol{A}.$$

At the same time, for $\boldsymbol{B} := \boldsymbol{I_n}$ (the standard Euclidean norm), the Lipschitz constant increases by the factor $\lambda_1(\boldsymbol{A}^\top\boldsymbol{A})$, which makes the problem extremely ill-conditioned.

A particular case of this example is *separable optimization*, or *generalized linear models* (Bishop, 2006), which covers the classical regression and classification models.

Example 2.4. *Let*

$$f(\boldsymbol{x}) = \tfrac{1}{m}\sum_{i=1}^m \phi(\langle \boldsymbol{a}_i, \boldsymbol{x}\rangle), \qquad \boldsymbol{x} \in \mathbb{R}^n,$$

where $\phi : \mathbb{R} \to \mathbb{R}$ is a loss function satisfying: $\mu \leq \phi''(t) \leq L$, $\forall t \in \mathbb{R}$, with some $L \geq \mu \geq 0$. Then, forming the matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ whose rows are $\boldsymbol{a}_1^\top, \ldots, \boldsymbol{a}_m^\top$ and setting $\boldsymbol{B} := \boldsymbol{A}^\top\boldsymbol{A}$, condition (5) holds.

3. Preconditioned Gradient Methods

A natural intention would be to use the global upper bound (6) as a model for the smooth part of the objective. However, the direct minimization of such upper model requires to solve the linear system with the matrix $\boldsymbol{B}$, which can computationally unfeasible for large scale problems.

Instead, let us fix for our *preconditioner* some positive definite symmetric matrix $\boldsymbol{P} = \boldsymbol{P}^\top \succ 0$, which satisfies the following bound, for some $\alpha := \alpha(\boldsymbol{P})$ and $\beta := \beta(\boldsymbol{P}) \geq \alpha > 0$:

$$\boxed{\alpha \boldsymbol{B}^{-1} \preceq \boldsymbol{P} \preceq \beta \boldsymbol{B}^{-1}.} \quad (7)$$

We are going to use this matrix instead of $\boldsymbol{B}^{-1}$ in our methods. For a fixed symmetric positive definite matrix $\boldsymbol{P}$ and parameter $M > 0$, we denote the *gradient step* from a point $\boldsymbol{x} \in \operatorname{dom}\psi$ along a *gradient direction* $\boldsymbol{g} \in \mathbb{R}^n$ by

$$\begin{aligned} &\operatorname{GradStep}_{M,\boldsymbol{P}}(\boldsymbol{x}, \boldsymbol{g}) \\ &\quad \stackrel{\text{def}}{=} \operatorname*{argmin}_{\boldsymbol{y}\in\operatorname{dom}\psi} \Big\{ \langle \boldsymbol{g}, \boldsymbol{y}\rangle + \psi(\boldsymbol{y}) + \frac{M}{2}\|\boldsymbol{y}-\boldsymbol{x}\|_{\boldsymbol{P}^{-1}}^2 \Big\}. \end{aligned}$$

This operation is well-defined since the objective function in the above minimization problem is strongly convex. We assume that both ψ and $\boldsymbol{P}$ are reasonably simple so that the corresponding gradient step can be efficiently computed. An important case is $\psi = 0$ for which we have

[3] Here, we assume that $\boldsymbol{A}^\top\boldsymbol{A} \succ 0$ which is typically the case when $m \gg n$. Otherwise, we can reduce the dimensionality.

$\mathrm{GradStep}_{M,\boldsymbol{P}}(\boldsymbol{x},\boldsymbol{g}) = \boldsymbol{x} - \frac{1}{M}\boldsymbol{P}\boldsymbol{g}$. The latter expression can be efficiently computed whenever one can cheaply multiply the matrix $\boldsymbol{P}$ by any vector.

3.1. Preconditioned Basic Gradient Method

First, we consider the basic first-order scheme shown in Algorithm 1 for solving the composite problem (4). For simplicity, in this section, we only present a version of this method with a fixed step size and assume that all necessary constants are known. An adaptive version of Algorithm 1 which does not have these limitations and is more efficient in practice can be found in Appendix C.

Algorithm 1 Preconditioned Basic Gradient Method

Input: $\boldsymbol{x}_0 \in \operatorname{dom}\psi$, $\boldsymbol{P} = \boldsymbol{P}^\top \succ 0$, $M > 0$.
for $k = 0, 1, \ldots$ **do**
  Compute $\boldsymbol{x}_{k+1} = \mathrm{GradStep}_{M,\boldsymbol{P}}\big(\boldsymbol{x}_k, \nabla f(\boldsymbol{x}_k)\big)$.
end for

For Algorithm 1, we can prove the following results.

Theorem 3.1. *Consider Algorithm 1 with $M = \beta L$. Then, at each iteration $k \geq 1$, we have*

$$F(\boldsymbol{x}_k) - F^\star \;\leq\; \frac{\beta}{\alpha}\frac{L\|\boldsymbol{x}_0 - \boldsymbol{x}^\star\|^2_{\boldsymbol{B}}}{k}. \qquad (8)$$

When $\mu > 0$, the convergence is linear: for all $k \geq 1$,

$$F(\boldsymbol{x}_k) - F^\star \;\leq\; \Big(1 - \frac{1}{4}\frac{\alpha}{\beta}\frac{\mu}{L}\Big)^k [F(\boldsymbol{x}_0) - F^\star]. \qquad (9)$$

We see that one of the principal complexity factors in the above estimates is the condition number β/α which depends on the choice of our preconditioner $\boldsymbol{P}$ (see (7)). For the basic choice $\boldsymbol{P} = \boldsymbol{I}$, we have $\beta/\alpha = \lambda_1/\lambda_n$. However, as we show in the following sections, it is possible to use more efficient (and still quite cheap) preconditioners which improve this condition number.

3.2. Preconditioned Fast Gradient Method

Now let us consider an accelerated scheme shown in Algorithm 2. This algorithm is one of the standard variants of the Fast Gradient Method (FGM) known as the Method of Similar Triangles (see, e.g., Section 6.1.3 in (Nesterov, 2018)) but adapted to our assumptions (5) and (7).

As in other versions of FGM, to properly handle strongly convex problems, Algorithm 2 requires the knowledge of the strong convexity parameter α and μ (or, more precisely, their product $\rho = \alpha\mu$). For non-strongly convex problems, we can always choose $\alpha = \mu = 0$. See also Appendix C for a variant of Algorithm 2 which can automatically adjust the constant M in iterations.

Algorithm 2 Preconditioned Fast Gradient Method

Input: $\boldsymbol{x}_0 \in \operatorname{dom}\psi$, $\boldsymbol{P} = \boldsymbol{P}^\top \succ 0$, $M > 0$, $\rho \geq 0$.
Set $\boldsymbol{v}_0 = \boldsymbol{x}_0$, $A_0 = 0$.
for $k = 0, 1, \ldots$ **do**
  Find a_{k+1} from eq. $\frac{Ma_{k+1}^2}{A_k + a_{k+1}} = 1 + \rho(A_k + a_{k+1})$.
  Set $A_{k+1} = A_k + a_{k+1}$, $H_k = \frac{1+\rho A_{k+1}}{a_{k+1}}$.
  Set $\theta_k = \frac{a_{k+1}}{A_{k+1}}$, $\omega_k = \frac{\rho}{H_k}$, $\gamma_k = \frac{\omega_k(1-\theta_k)}{1-\omega_k\theta_k}$.
  Set $\hat{\boldsymbol{v}}_k = (1-\gamma_k)\boldsymbol{v}_k + \gamma_k\boldsymbol{x}_k$.
  Set $\boldsymbol{y}_k = (1-\theta_k)\boldsymbol{x}_k + \theta_k\hat{\boldsymbol{v}}_k$.
  Compute $\boldsymbol{v}_{k+1} = \mathrm{GradStep}_{H_k,\boldsymbol{P}}\big(\hat{\boldsymbol{v}}_k, \nabla f(\boldsymbol{y}_k)\big)$.
  Set $\boldsymbol{x}_{k+1} = (1-\theta_k)\boldsymbol{x}_k + \theta_k\boldsymbol{v}_{k+1}$.
end for

The convergence results for Algorithm 2 are as follows.

Theorem 3.2. *Consider Algorithm 2 with $M = \beta L$ and $\rho = \alpha\mu$. Then, at each iteration $k \geq 1$, we have*

$$F(\boldsymbol{x}_k) - F^\star \;\leq\; 2\frac{\beta}{\alpha}\frac{L\|\boldsymbol{x}_0 - \boldsymbol{x}^\star\|^2_{\boldsymbol{B}}}{k^2}. \qquad (10)$$

When $\mu > 0$, the convergence is linear: for all $k \geq 1$,

$$F(\boldsymbol{x}_k) - F^\star \;\leq\; \Big(1 - \sqrt{\frac{\alpha}{\beta}\frac{\mu}{L}}\Big)^{k-1}\frac{\beta}{\alpha}\frac{L}{2}\|\boldsymbol{x}_0 - \boldsymbol{x}^\star\|^2_{\boldsymbol{B}}.$$

Comparing these estimates with those from Theorem 3.1, we see that the accelerated scheme is much more efficient. For instance, to reach accuracy $\epsilon > 0$ in terms of the objective function in the non-strongly convex case, Algorithm 1 needs $k(\epsilon) = \frac{\beta}{\alpha}\frac{L\|\boldsymbol{x}_0 - \boldsymbol{x}^\star\|^2_{\boldsymbol{B}}}{\epsilon}$ iterations, while for Algorithm 2 this number is only $k_2(\epsilon) = \sqrt{2k(\epsilon)}$. Similar conclusions are valid in the strongly convex case.

Despite having much weaker dependency on the condition number β/α, Algorithm 2 is still quite sensitive to it. Thus, the proper choice of the preconditioner $\boldsymbol{P}$ is important for both our methods.

4. Symmetric Polynomial Preconditioning

We would like to have a *family* of preconditioners $\boldsymbol{P}_\tau$ for our problem indexed by some parameter τ. Varying τ should provide us with a trade off between the spectral quality of approximation (7) of the inverse matrix and the arithmetical cost of computing the preconditioner.

Surprisingly, such a family of preconditioners can be built by using *symmetric polynomials*, the classical objects of Algebra. We prove that our preconditioning improves the condition number $\frac{\beta}{\alpha}$ of the problem, by automatically cutting off the large gaps between the top eigenvalues.

4.1. Definition and Basic Properties

We define the family of symmetric matrices $\{\boldsymbol{P}_\tau\}_{0\le\tau\le n-1}$ recursively. We start with identity matrix: $\boldsymbol{P}_0 \stackrel{\text{def}}{=} \boldsymbol{I}$. Then,

$$\boxed{\boldsymbol{P}_\tau \stackrel{\text{def}}{=} \frac{1}{\tau}\sum_{i=1}^{\tau}(-1)^{i-1}\boldsymbol{P}_{\tau-i}\boldsymbol{U}_i,} \tag{11}$$

where $\boldsymbol{U}_\tau \stackrel{\text{def}}{=} \operatorname{tr}(\boldsymbol{B}^\tau)\boldsymbol{I} - \boldsymbol{B}^\tau$ are the auxiliary matrices. It turns out that matrices (11) serve as a good approximation of the inverse matrix: $\boldsymbol{P}_\tau \approx \boldsymbol{B}^{-1}$, up to some multiplicative constant, and the quality of such approximation is gradually improving when increasing parameter τ. Let us look at several first members. Clearly,

$$\boldsymbol{P}_1 \;=\; \operatorname{tr}(\boldsymbol{B})\boldsymbol{I} - \boldsymbol{B}, \tag{12}$$

which is very easy to handle, by having computed the trace of the curvature matrix. Then, multiplying $\boldsymbol{P}_1$ by any vector would require just one matrix-vector multiplication with our original $\boldsymbol{B}$. Further,

$$\begin{aligned}\boldsymbol{P}_2 &= \tfrac{1}{2}\operatorname{tr}(\boldsymbol{P}_1\boldsymbol{B})\boldsymbol{I} - \boldsymbol{P}_1\boldsymbol{B}\\ &= \tfrac{1}{2}\big[[\operatorname{tr}(\boldsymbol{B})]^2 - \operatorname{tr}(\boldsymbol{B}^2)\big]\boldsymbol{I} - \operatorname{tr}(\boldsymbol{B})\boldsymbol{B} + \boldsymbol{B}^2,\end{aligned} \tag{13}$$

thus its use would cost just *two* matrix-vector products with $\boldsymbol{B}$, having evaluated[4,5] the numbers $\operatorname{tr}(\boldsymbol{B})$ and $\operatorname{tr}(\boldsymbol{B}^2)$.

It is clear that in general $\boldsymbol{P}_\tau = p_\tau(\boldsymbol{B})$, where p_τ is a polynomial of a fixed degree τ with coefficients that can be found recursively from (11). Let us give a useful interpretation for our family of preconditioners, that also explains their name. For $\boldsymbol{a} \in \mathbb{R}^{n-1}$, we denote by $\sigma_0(\boldsymbol{a}), \dots, \sigma_{n-1}(\boldsymbol{a})$ the *elementary symmetric polynomials* in $n-1$ variables[6]. It is known that every symmetric polynomial (that is invariant to any permutation of the variables) can be represented as a weighed sum of elementary symmetric polynomials (Dummit & Foote, 2004). We establish the following important characterization.

Lemma 4.1. *Let $\boldsymbol{B} = \boldsymbol{Q}\operatorname{Diag}(\boldsymbol{\lambda})\boldsymbol{Q}^\top$ be the spectral decomposition. Then,*

$$\boldsymbol{P}_\tau \;=\; \boldsymbol{Q}\operatorname{Diag}(\sigma_\tau(\boldsymbol{\lambda}_{-1}), \dots, \sigma_\tau(\boldsymbol{\lambda}_{-n}))\boldsymbol{Q}^\top, \tag{14}$$

where $\boldsymbol{\lambda}_{-i} \in \mathbb{R}^{n-1}$ is the vector that contains all elements of $\boldsymbol{\lambda}$ except λ_i.

In particular, we justify $\boldsymbol{P}_\tau \succ 0$. For $\tau = n-1$, we get

$$\boldsymbol{P}_{n-1} \stackrel{(4.1)}{=} \det(\boldsymbol{B})\boldsymbol{B}^{-1} \stackrel{\text{def}}{=} \operatorname{Adj}(\boldsymbol{B}), \tag{15}$$

which gives us the true inverse matrix $\boldsymbol{B}^{-1}$ up to the constant factor $\det(\boldsymbol{B})$.

4.2. Approximation Quality

Now, let us show that the quality of approximation $\boldsymbol{P}_\tau \approx \boldsymbol{B}^{-1}$ and that the corresponding condition number $\frac{\beta}{\alpha}$ is improving when τ is increasing.

Theorem 4.2. *For any τ, we have*

$$\lambda_n\sigma_\tau(\boldsymbol{\lambda}_{-n})\boldsymbol{B}^{-1} \preceq \boldsymbol{P}_\tau \preceq \lambda_1\sigma_\tau(\boldsymbol{\lambda}_{-1})\boldsymbol{B}^{-1}. \tag{16}$$

Therefore, the condition number is bounded as

$$\frac{\beta}{\alpha} \stackrel{(16)}{=} \frac{\lambda_1}{\lambda_n}\cdot\xi_\tau(\boldsymbol{\lambda}), \quad \textit{where} \quad \xi_\tau(\boldsymbol{\lambda}) \stackrel{\text{def}}{=} \frac{\sigma_\tau(\boldsymbol{\lambda}_{-1})}{\sigma_\tau(\boldsymbol{\lambda}_{-n})}.$$

Note that $\xi_\tau(\boldsymbol{\lambda}) \le 1$. It is equal to 1 for $\tau = 0$ and can be much smaller for bigger values of τ. For example, for $\tau = 1$ (thus using preconditioner $\boldsymbol{P}_1$ given by (12)), we get

$$\xi_1(\boldsymbol{\lambda}) \;=\; \frac{\sum_{i=2}^n \lambda_i}{\sum_{i=1}^{n-1}\lambda_i} \;\le\; 1,$$

and it is much smaller than 1 when $\lambda_1 \gg \lambda_2$, which corresponds to the case when the highest eigenvalue is well separated from the others. Therefore, the methods with preconditioner $\boldsymbol{P}_1$ achieve a *provable acceleration* in the case of large gap between λ_1 and λ_2, *without explicit knowledge* of the spectrum of $\boldsymbol{B}$. The price of using $\boldsymbol{P}_1$ instead of $\boldsymbol{P}_0$ is just one extra matrix-vector product per iteration. Let us summarize the main properties of $\xi_\tau(\boldsymbol{\lambda})$ (see also Figure 3).

Lemma 4.3. *It holds that $\xi_0(\boldsymbol{\lambda}) = 1$, $\xi_{n-1}(\boldsymbol{\lambda}) = \frac{\lambda_n}{\lambda_1}$, and $\xi_\tau(\boldsymbol{\lambda})$ monotonically decreases with τ. Moreover,*

$$\xi_\tau(\boldsymbol{\lambda}) \;\to\; 0 \quad \textit{when} \quad \frac{\lambda_1}{\lambda_{\tau+1}} \;\to\; \infty.$$

A more explicit upper bound which implies the above limit is as follows.

Lemma 4.4. *For any $0 \le \tau \le n-1$, we have*

$$\xi_\tau(\boldsymbol{\lambda}) \le \frac{\sum_{i=\tau+1}^n \lambda_i}{\lambda_1 + \sum_{i=\tau+1}^{n-1}\lambda_i}.$$

We see that τ interpolates $\boldsymbol{P}_\tau$ between $\boldsymbol{I}$ and $\operatorname{Adj}(\boldsymbol{B})$, while the condition number $\frac{\beta}{\alpha}$ changes from $\frac{\lambda_1}{\lambda_n}$ to 1 Therefore, we obtain an extra degree of freedom in our methods for choosing an appropriate small value of τ, that improves the spectrum of the problem by cutting off large gaps between λ_1 and $\lambda_{\tau+1}$.

[4] Note that $\operatorname{tr}(\boldsymbol{B}^2) = \sum_{i=1}^n \|\boldsymbol{B}[:,i]\|_2^2$, where $\boldsymbol{B}[:,i] \in \mathbb{R}^n$ is the ith column of $\boldsymbol{B}$.

[5] For general τ, we can also use a stochastic estimate of the trace: $\xi_\tau \stackrel{\text{def}}{=} n\langle \boldsymbol{B}^\tau\boldsymbol{u}, \boldsymbol{u}\rangle$, where $\boldsymbol{u} \in \mathbb{R}^n$ is uniformly distributed on the unit sphere. It would give an unbiased estimate: $\mathbb{E}[\xi_\tau] = n\mathbb{E}[\operatorname{tr}(\boldsymbol{u}^\top\boldsymbol{B}^\tau\boldsymbol{u})] = n\operatorname{tr}(\mathbb{E}[\boldsymbol{u}\boldsymbol{u}^\top]\boldsymbol{B}^\tau) = \operatorname{tr}(\boldsymbol{B}^\tau)$.

[6] That is $\sigma_\tau(\boldsymbol{a}) \stackrel{\text{def}}{=} \sum_{1\le i_1<\dots<i_\tau\le n-1} a_{i_1}\dots a_{i_\tau}$.

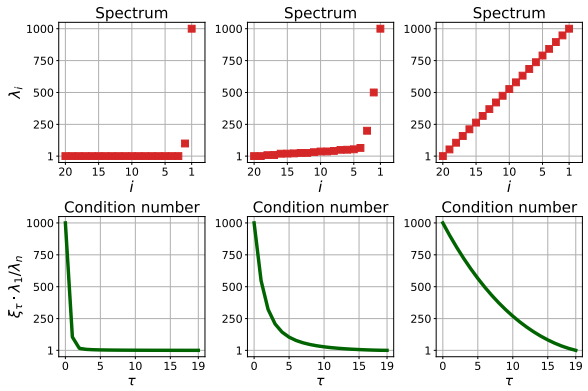


Figure 3: *Above:* different distributions of eigenvalues $\boldsymbol{\lambda}(\boldsymbol{B})$. *Below:* the corresponding improvement of the condition number $\frac{\beta}{\alpha} = \xi_\tau(\boldsymbol{\lambda}) \cdot \lambda_1/\lambda_n$ by using the preconditioner $\boldsymbol{P}_\tau$ of order τ.

4.3. Stochastic Representation

Let us provide another interesting interpretation for our family of preconditioners. One way of approximating the inverse matrix $\boldsymbol{B}^{-1}$ could be to extract from $\boldsymbol{B}$ a randomly selected principal submatrix of size $\tau+1$, compute its inverse, put it back into the original "big matrix" and zero out all other elements outside the submatrix. It turns out that, in expectation, the result of this operation is exactly proportional to our preconditioner $\boldsymbol{P}_\tau$ if we pick the submatrix from a special *volume sampling* distribution (Deshpande et al., 2006).

Theorem 4.5. *For any $0 \le \tau \le n-1$, it holds that*

$$\boldsymbol{P}_\tau \propto \mathbb{E}_{S \sim \mathrm{Vol}_{\tau+1}(\boldsymbol{B})}[\boldsymbol{I}_S(\boldsymbol{B}_{S\times S})^{-1}\boldsymbol{I}_S^\top], \qquad (17)$$

where $S \subseteq \{1,\dots,n\}$ is a random $(\tau+1)$-element subset of coordinates, $\boldsymbol{I}_S \in \mathbb{R}^{n\times(\tau+1)}$ is the matrix obtained from the identity matrix by retaining only the columns with indices from S, $\boldsymbol{B}_{S\times S} \overset{\text{def}}{=} \boldsymbol{I}_S^\top \boldsymbol{B} \boldsymbol{I}_S \in \mathbb{R}^{(\tau+1)\times(\tau+1)}$, and $\mathrm{Vol}_{\tau+1}(\boldsymbol{B})$ is the volume sampling distribution prescribing to pick S with probability $\propto \det(\boldsymbol{B}_{S\times S})$.

The idea of applying volume sampling in Optimization was first proposed in (Rodomanov & Kropotov, 2020) for accelerating coordinate descent methods. It was shown that using this particular nonuniform sampling of coordinates leads to a provable acceleration by a factor whose magnitude depends on gaps in the spectrum of the curvature matrix.

Thus, we can interpret our basic Gradient Method (Algorithm 1) with a fixed Symmetric Polynomial Preconditioner $\boldsymbol{P}_\tau$ as a deterministic counterpart of the randomized coordinate descent method from (Rodomanov & Kropotov, 2020) with $(\tau+1)$-element volume sampling of coordinates. Correspondingly, both methods have very similar convergence properties and theoretical efficiency estimates.

Nevertheless, this work offers several significant advantages over (Rodomanov & Kropotov, 2020). First, in addition to the basic method, we have an accelerated one (Algorithm 2), while the accelerated version of coordinate descent with volume sampling is an open question. Second, volume sampling is an expensive operation which is difficult to carry out already when $\tau = 2$. In contrast, the corresponding preconditioner $\boldsymbol{P}_2$ for our gradient methods is still computationally efficient (see Section 4.1). Finally, as we will show next, the basic Gradient Method can be improved to *automatically* choose the best possible polynomial preconditioner of degree τ (including the one we have been discussing in this section), and the resulting algorithm can easily handle much bigger values of τ.

5. Krylov Subspace Preconditioning

Our new symmetric polynomial preconditioners, introduced in the previous section, can be viewed as a certain family of polynomials that we apply to our curvature matrix $\boldsymbol{B}$. Thus, for a fixed degree $\tau > 0$, we use the matrix $\boldsymbol{P} = p_\tau(\boldsymbol{B})$ as a preconditioner, where p_τ is a specifically constructed polynomial of degree τ such that $\boldsymbol{P} \succ 0$.

A natural question is *how optimal is this choice of a polynomial?* Indeed, the problem of polynomial approximation has a long and rich history with an affirmative answer provided by the classical Chebyshev polynomials (Mason & Handscomb, 2002) for the uniform approximation bound. We present a new adaptive algorithm that automatically achieves the *best* polynomial preconditioning. Then, we study what are the complexity guarantees that we can get with this optimal approach. In this section, we focus on the non-composite case only, i.e. the problem of unconstrained minimization of a smooth function: $\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x})$.

5.1. Gradient Method with Krylov Preconditioning

We denote $\boldsymbol{P}_{\boldsymbol{a}} \overset{\text{def}}{=} a_0\boldsymbol{I} + a_1\boldsymbol{B} + \ldots + a_\tau\boldsymbol{B}^\tau$, where vector $\boldsymbol{a} = (a_0,\dots,a_\tau) \in \mathbb{R}^{\tau+1}$ is a parameter. In each iteration, it is found by solving the linear system:

$$\boldsymbol{a} \;=\; \boldsymbol{A}_\tau^{-1}\boldsymbol{g}_\tau \;\in\; \mathbb{R}^{\tau+1}, \qquad (18)$$

where $\boldsymbol{A}_\tau = \boldsymbol{A}_\tau(\boldsymbol{x}) \in \mathbb{R}^{(\tau+1)\times(\tau+1)}$ is the Gram matrix with the following structure $(0 \le i,j \le \tau)$:

$$\big[\boldsymbol{A}_\tau(\boldsymbol{x})\big]^{(i,j)} \;\overset{\text{def}}{=}\; L\cdot\langle\nabla f(\boldsymbol{x}), \boldsymbol{B}^{i+j+1}\nabla f(\boldsymbol{x})\rangle, \qquad (19)$$

and $\boldsymbol{g}_\tau = \boldsymbol{g}_\tau(\boldsymbol{x}) \in \mathbb{R}^{\tau+1}$ is defined by $(0 \le i \le \tau)$:

$$\big[\boldsymbol{g}_\tau(\boldsymbol{x})\big]^{(i)} \;\overset{\text{def}}{=}\; \langle\nabla f(\boldsymbol{x}), \boldsymbol{B}^{i}\nabla f(\boldsymbol{x})\rangle. \qquad (20)$$

Note that this operation is exactly the projection of the direction $\frac{1}{L}\boldsymbol{B}^{-1}\nabla f(\boldsymbol{x}_k)$ onto the *Krylov subspace* (3):

$$\boldsymbol{x}_{k+1} - \boldsymbol{x}_k \;:=\; \operatorname*{argmin}_{\boldsymbol{h}\in\mathcal{K}_{\boldsymbol{x}_k}^\tau} \|\boldsymbol{h} + \tfrac{1}{L}\boldsymbol{B}^{-1}\nabla f(\boldsymbol{x}_k)\|_{\boldsymbol{B}}^2.$$

Fortunately, for computing this projection we indeed do not need to invert the curvature matrix $\boldsymbol{B}$, but to solve only a small linear system (18) of size $\tau + 1$. We are ready to formulate our new adaptive method.

Algorithm 3 Gradient Method with Krylov Preconditioning

Initialization: $\boldsymbol{x}_0 \in \mathbb{R}^n$, $\tau \geq 0$, $L > 0$.
for $k = 0, 1, \ldots$ **do**
  Form matrix $\boldsymbol{A}_\tau(\boldsymbol{x}_k)$ and vector $\boldsymbol{g}_\tau(\boldsymbol{x}_k)$ by (19), (20).
  Compute $\boldsymbol{a}_k = \boldsymbol{A}_\tau(\boldsymbol{x}_k)^{-1}\boldsymbol{g}_\tau(\boldsymbol{x}_k) \in \mathbb{R}^{\tau+1}$.
  Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \boldsymbol{P}_{\boldsymbol{a}_k}\nabla f(\boldsymbol{x}_k)$.
end for

We prove the following optimality result.

Theorem 5.1. *Let $\boldsymbol{P} \succ 0$ be any preconditioner that is a polynomial of degree τ of the curvature matrix:*

$$\boldsymbol{P} \;=\; p_\tau(\boldsymbol{B}), \quad p_\tau \in \mathbb{R}[s], \quad \deg(p_\tau) = \tau.$$

Then, for the iteration of Algorithm 3 we have the global rates (8),(10) with the condition number that is attributed to $\boldsymbol{P}$ (7): $\frac{\beta}{\alpha} = \frac{\beta(\boldsymbol{P})}{\alpha(\boldsymbol{P})}$.

Hence, our method *automatically* chooses the best possible preconditioning matrix from the polynomial class. Let us understand what are the bounds for $\frac{\beta}{\alpha}$ that we can achieve in this case.

5.2. Bounds for the Condition Number

Let us assume that the top $\tau > 0$ eigenvalues of $\boldsymbol{B}$ are all *separated*. Then, we can easily cut them off with the following simple construction. Define

$$q_\tau(s) \;\stackrel{\text{def}}{=}\; \left(1 - \tfrac{s}{\lambda_1}\right)\left(1 - \tfrac{s}{\lambda_2}\right) \cdot \ldots \cdot \left(1 - \tfrac{s}{\lambda_\tau}\right). \qquad (21)$$

Proposition 5.2. *For any τ, taking $\boldsymbol{P} = p_\tau(\boldsymbol{B})$, where $p_\tau(s) := \frac{1+q_\tau(s)\cdot(\alpha s - 1)}{s}$ with q_τ defined by (21) and $\alpha = \frac{2}{\lambda_{\tau+1}+\lambda_n}$, the condition number is bounded by $\frac{\beta}{\alpha} \leq \frac{\lambda_{\tau+1}}{\lambda_n}$.*

The worst case instance for the cutting strategy is when all the eigenvalues except one share the same value. A better approach in such a situation would be to find a bound from the *uniform* polynomial approximation for the whole interval $[\lambda_n, \lambda_1]$, which is achieved with the Chebyshev polynomials (Nemirovski, 1995).

Proposition 5.3. *For a fixed $0 < \epsilon < 1$, let $\tau := \left\lfloor \sqrt{\frac{\lambda_1}{\lambda_n}} \ln \frac{8}{\epsilon} \right\rfloor$. Then, taking $\boldsymbol{P} = p_\tau(\boldsymbol{B})$, where $p_\tau(s) := \frac{1 - Q_\tau(s)}{s}$ with Q_τ is a normalized Chebyshev polynomial[7] of the first kind of degree $\tau + 1$, the condition number is bounded by $\frac{\beta}{\alpha} \leq 1 + \epsilon$.*

[7] See Appendix B.9 for the precise definition.

5.3. Discussion

We see that in the case of unconstrained smooth minimization, it is possible to achieve the guarantee of the *best polynomial* of a fixed degree τ, by computing a certain projection onto the corresponding Krylov subspace. Namely, we can achieve $\frac{\beta}{\alpha} \leq \frac{\lambda_{\tau+1}}{\lambda_n}$ (Proposition 5.2), which cuts off the top τ eigenvalues of the spectrum completely, if they are separated from the others. At the same time, by using the Chebyshev polynomials, we can contract a part of the spectrum *uniformly*, with an appropriate degree τ (Proposition 5.3). It remains to be an open question where we can incorporate adaptive Krylov preconditioning into the Fast Gradient Method, which would give us a further improvement of the condition number.

6. Experiments

Huber Loss. Let us present an illustrative experiment, with the regression model (Example 2.4) with the Huber loss function:

$$\phi(t) \;:=\; \begin{cases} \frac{t^2}{2\mu}, & \text{if} \quad |t| \leq \mu, \\ |t| - \frac{\mu}{2}, & \text{otherwise,} \end{cases}$$

where $\mu := 0.1$ is a parameter. The data is generated with a fixed distribution of eigenvalues: $\lambda_1 > \lambda_2 > \lambda_3 = \ldots \lambda_n = 1$. Thus, we have two gaps between the leading eigenvalues. We use the Gradient Method (Algorithm 1), with the adaptive search to fit the parameter M. The results are shown in Figure 4. Using the preconditioner $\boldsymbol{P}_1$ helps the method to deal with the large gap between λ_1 and λ_2, while $\boldsymbol{P}_2$ makes the method to be insensitive to the gap between λ_1 and λ_3, as predicted by our theory.

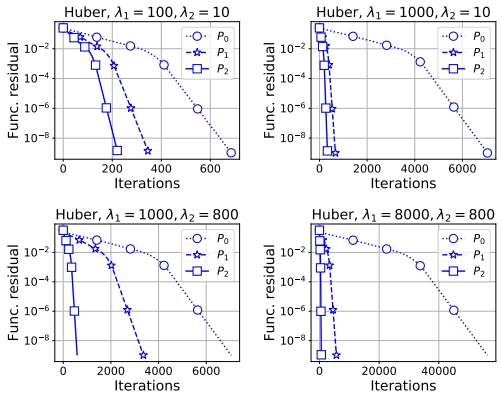


Figure 4: Minimizing the Huber loss.

Logistic Regression. We examine the training of logistic regression on real data. In Figure 5, we see that the best convergence is achieved by the Fast Gradient Method (FGM, Algorithm 2) with $\boldsymbol{P}_2$. Using Symmetric Polynomial Preconditioning makes the methods to converge much

better (*two times faster* for GM using $\boldsymbol{P}_2$ instead of $\boldsymbol{P}_0 \equiv \boldsymbol{I}$, and about 1.5 *times faster* for FGM). Among the versions of GM, the most encouraging performance belongs to the Krylov preconditioning, which is consistent with the theory.

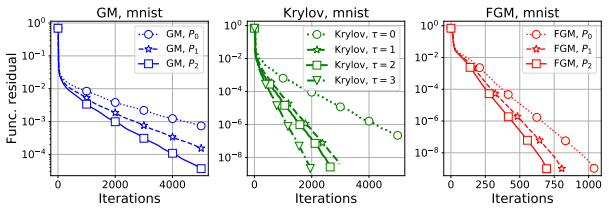


Figure 5: Training logistic regression.

Acknowledgements

The work of the first author was supported by the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number 22.00133. The work of the second author received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 788368).

References

Bishop, C. M. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Bullins, B. Highly smooth minimization of non-smooth problems. In *Conference on Learning Theory*, pp. 988–1030. PMLR, 2020.

Cunha, L., Gidel, G., Pedregosa, F., Scieur, D., and Paquette, C. Only tails matter: Average-case universality and robustness in the convex regime. In *International Conference on Machine Learning*, pp. 4474–4491. PMLR, 2022.

Deshpande, A., Rademacher, L., Vempala, S. S., and Wang, G. Matrix approximation and projective clustering via volume sampling. *Theory of Computing*, 2(12):225–247, 2006.

Dubois, P. F., Greenbaum, A., and Rodrigue, G. H. Approximating the inverse of a matrix for use in iterative algorithms on vector processors. *Computing*, 22(3):257–268, 1979.

Dummit, D. S. and Foote, R. M. *Abstract algebra*, volume 3. Wiley Hoboken, 2004.

Golub, G. H. and Van Loan, C. F. *Matrix computations*. JHU press, 2013.

Goujaud, B., Scieur, D., Dieuleveut, A., Taylor, A. B., and Pedregosa, F. Super-acceleration with cyclical step-sizes. In *International Conference on Artificial Intelligence and Statistics*, pp. 3028–3065. PMLR, 2022.

Gupta, S. D., Freund, R. M., Sun, X. A., and Taylor, A. Nonlinear conjugate gradient methods: worst-case convergence rates via computer-assisted analyses. *arXiv preprint arXiv:2301.01530*, 2023.

Hardy, G. H., Littlewood, J. E., and Pólya, G. *Inequalities*. Cambridge University Press, second edition, 1952.

Hestenes, M. R. and Stiefel, E. Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49(6):409, 1952.

Johnson, O. G., Micchelli, C. A., and Paul, G. Polynomial preconditioners for conjugate gradient calculations. *SIAM Journal on Numerical Analysis*, 20(2):362–376, 1983.

Kalman, D. A matrix proof of Newton's identities. *Mathematics Magazine*, 73(4):313–315, 2000.

Kovalev, D., Richtárik, P., Gorbunov, E., and Gasanov, E. Stochastic spectral and conjugate descent methods. *Advances in Neural Information Processing Systems*, 31, 2018.

Liu, Q., Morgan, R. B., and Wilcox, W. Polynomial preconditioned gmres and gmres-dr. *SIAM Journal on Scientific Computing*, 37(5):S407–S428, 2015.

Loe, J. A. and Morgan, R. B. Toward efficient polynomial preconditioning for gmres. *Numerical Linear Algebra with Applications*, 29(4):e2427, 2022.

Mardal, K.-A. and Winther, R. Preconditioning discretizations of systems of partial differential equations. *Numerical Linear Algebra with Applications*, 18(1):1–40, 2011.

Mason, J. C. and Handscomb, D. C. *Chebyshev polynomials*. Chapman and Hall/CRC, 2002.

Nemirovski, A. Information-based complexity of convex programming. *Lecture Notes*, 834, 1995.

Nemirovski, A. and Yudin, D. Problem complexity and method efficiency in optimization. 1983.

Nesterov, Y. A method for solving the convex programming problem with convergence rate O(1/k^2). In *Dokl. akad. nauk Sssr*, volume 269, pp. 543–547, 1983.

Nesterov, Y. Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152, 2005.

Nesterov, Y. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

Nesterov, Y. *Lectures on convex optimization*, volume 137. Springer, 2018.

Nocedal, J. and Wright, S. *Numerical optimization*. Springer Science & Business Media, 2006.

Polyak, B. T. *Introduction to optimization*. Optimization Software, 1987.

Qu, Z., Richtárik, P., Takác, M., and Fercoq, O. Sdna: stochastic dual newton ascent for empirical risk minimization. In *International Conference on Machine Learning*, pp. 1823–1832. PMLR, 2016.

Rodomanov, A. and Kropotov, D. A randomized coordinate descent method with volume sampling. *SIAM Journal on Optimization*, 30(3):1878–1904, 2020.

Saad, Y. Practical use of polynomial preconditionings for the conjugate gradient method. *SIAM Journal on Scientific and Statistical Computing*, 6(4):865–881, 1985.

Scieur, D. and Pedregosa, F. Universal average-case optimality of polyak momentum. In *International conference on machine learning*, pp. 8565–8572. PMLR, 2020.

Spielman, D. A. and Teng, S.-H. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pp. 81–90, 2004.

Vaidya, P. M. Solving linear equations with symmetric diagonally dominant matrices by constructing good preconditioners. *A talk based on this manuscript*, 2(3.4):2–4, 1991.

Van Gijzen, M. A polynomial preconditioner for the gmres algorithm. *Journal of Computational and Applied Mathematics*, 59(1):91–107, 1995.

Vishnoi, N. K. Lx=b. *Foundations and Trends® in Theoretical Computer Science*, 8(1–2):1–141, 2013.

Supplementary Material

A. Extra Experiments

Logistic Regression. Let us present experimental results for our preconditioning strategies, for the training of Logistic Regression with several real datasets. We investigate both the number of iterations and the number of matrix-vector products (the most difficult operation) required to reach a certain accuracy level in the functional residual. The results are shown in Figure 6.

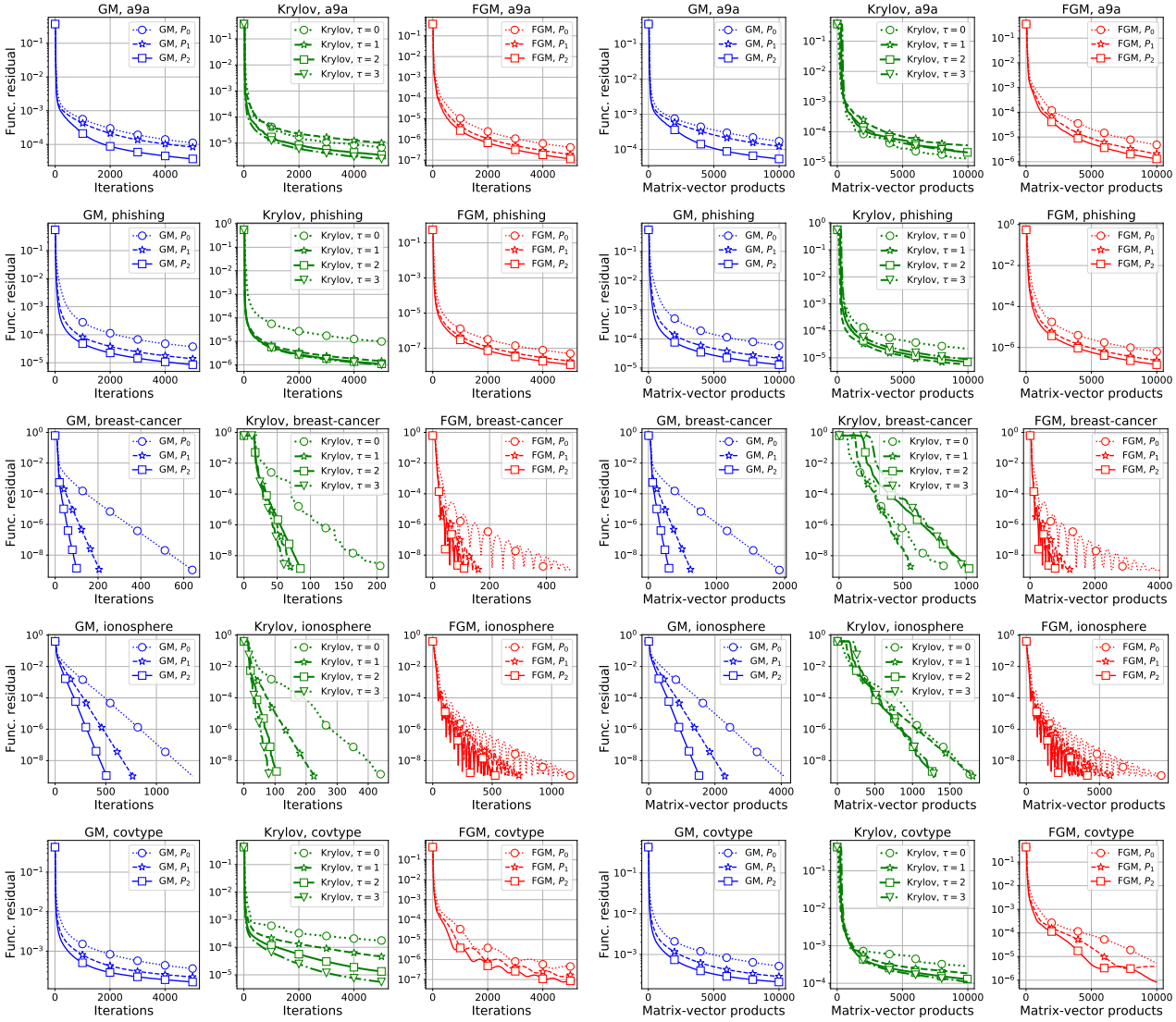


Figure 6: Training logistic regression with Algorithm 1 (GM) and Algorithm 2 (FGM) employing Symmetric Polynomial Preconditioning (11); and with Algorithm 3 (Krylov).

We see that using Symmetric Polynomial Preconditioning ($\boldsymbol{P}_1$ and $\boldsymbol{P}_2$) significantly accelerates both the Gradient Method (GM) and the Fast Gradient Method (GM), without extra arithmetic efforts during each iteration. Using the Krylov

preconditioning is more costly, while it equips GM with the best possible iteration rates.

Typical Distributions of the Data Spectrum. Let us provide the plots with the distributions of the leading eigenvalues (in the logarithmic scale) of the curvature matrix $\boldsymbol{B}$ for a selection of typical machine learning datesets[8].

We see (Figure 7) that it is quite common to have several top eigenvalues well separated from others. In these cases, our new Symmetric Polynomial Preconditioners provides the gradient methods with the best acceleration.

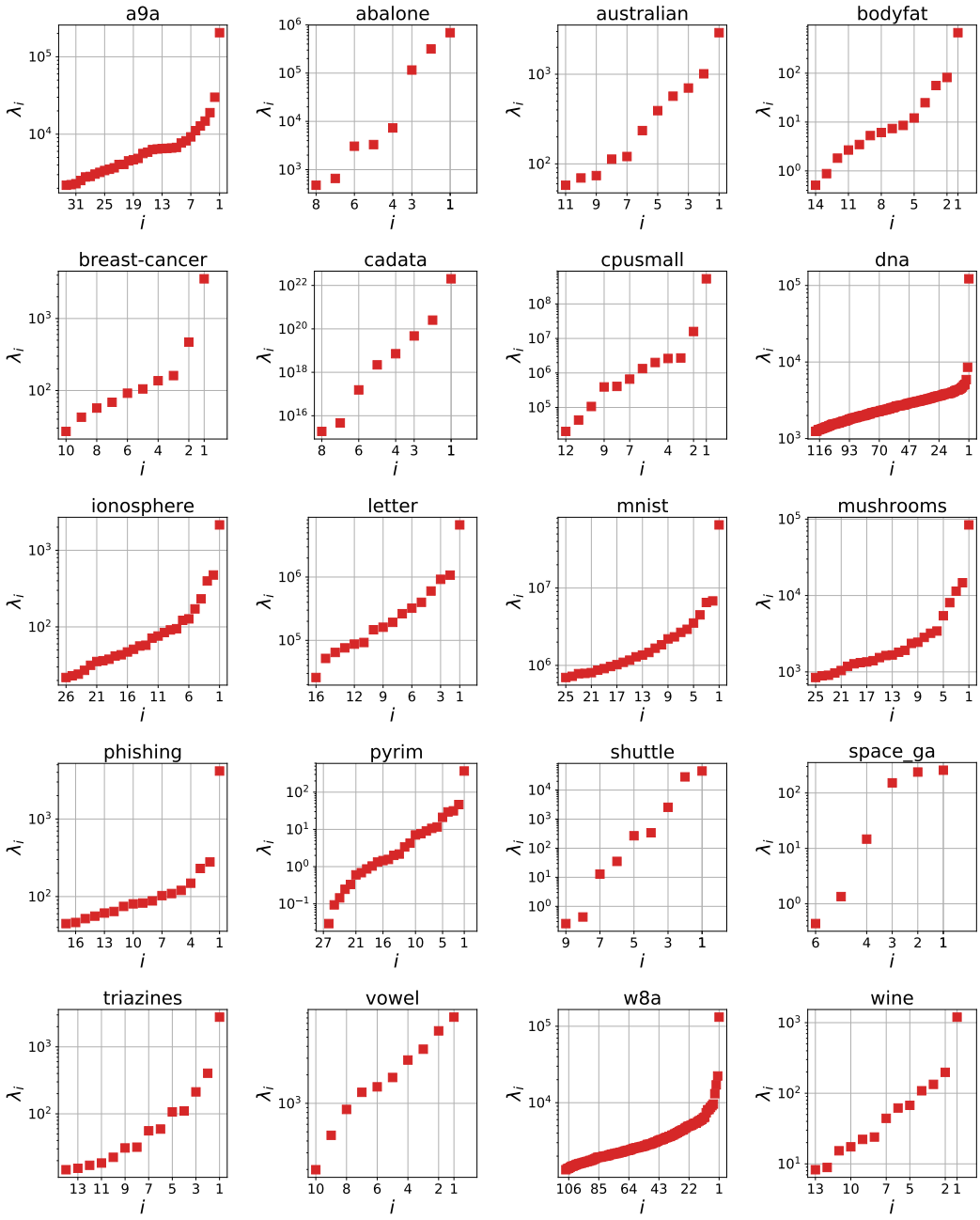


Figure 7: Leading eigenvalues (in the logarithmic scale) of the curvature matrix for several real datasets.

[8] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

Comparison with BFGS and L-BFGS. In the following experiment, we compare the performance of the Gradient Method (GM) and Fast Gradient Method (FGM) equipped with our Symmetric Polynomial Preconditioning, and GM with Krylov preconditioning, with the classical BFGS and L-BFGS optimization schemes (Nocedal & Wright, 2006).

The results are presented in Figure 8. We show both the number of matrix-vector products (the most expensive operation) and the total computational time[9] required to reach a given accuracy in terms of the functional residual.

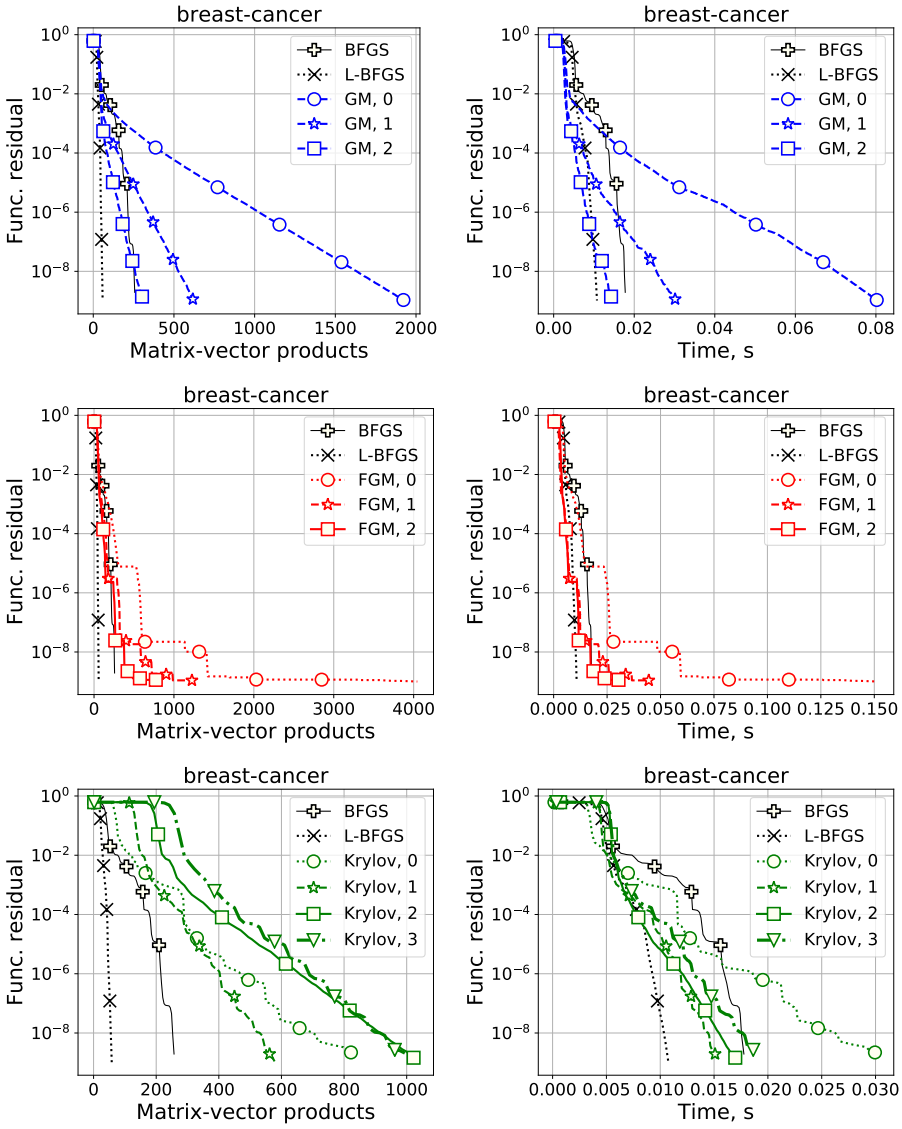


Figure 8: Comparison of our methods with Quasi-Newton methods: training Logistic Regression

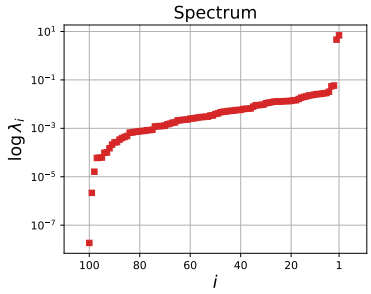
We see that the performance of the Gradient Method and Fast Gradient Method with Symmetric Polynomial preconditioner of order $\tau = 2$ (GM, 2 and FGM, 2 correspondingly) is comparable to that one of the BFGS and L-BFGS methods.

[9]Clock time was evaluated using the machine with Intel Core i5 CPU, 1.6GHz; 8 GB RAM. All methods were implemented in Python.

In the next experiment, we consider the problem of minimizing the Soft Maximum objective (log-sum-exp):

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) \quad := \quad \mu \ln\left(\sum_{i=1}^{m} \exp\left(\frac{\langle \boldsymbol{a}_i, \boldsymbol{x}\rangle - b_i}{\mu}\right)\right) \quad \approx \quad \max_{1\le i\le m}\left[\langle \boldsymbol{a}_i, \boldsymbol{x}\rangle - b_i\right], \tag{22}$$

where $\mu > 0$ is a sufficiently small number. The problems of this type are important in applications with minimax strategies for matrix games and for training ℓ_∞-regression (Nesterov, 2005; Bullins, 2020). The vectors $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m \in \mathbb{R}^n$ and $\boldsymbol{b} \in \mathbb{R}^n$ form our data, with $n = 100$, $m = 200$. That the structure of this objective satisfies Example 2.3 and the corresponding curvature matrix has the following distribution of the spectrum (in the double-logarithmic scale):



Note that the function (22) does not have a finite-sum structure, thus it is impossible to apply to this problem stochastic optimization methods. We use the value $\mu = 0.005$ for the smoothing parameter. The results are shown in Figure 9.

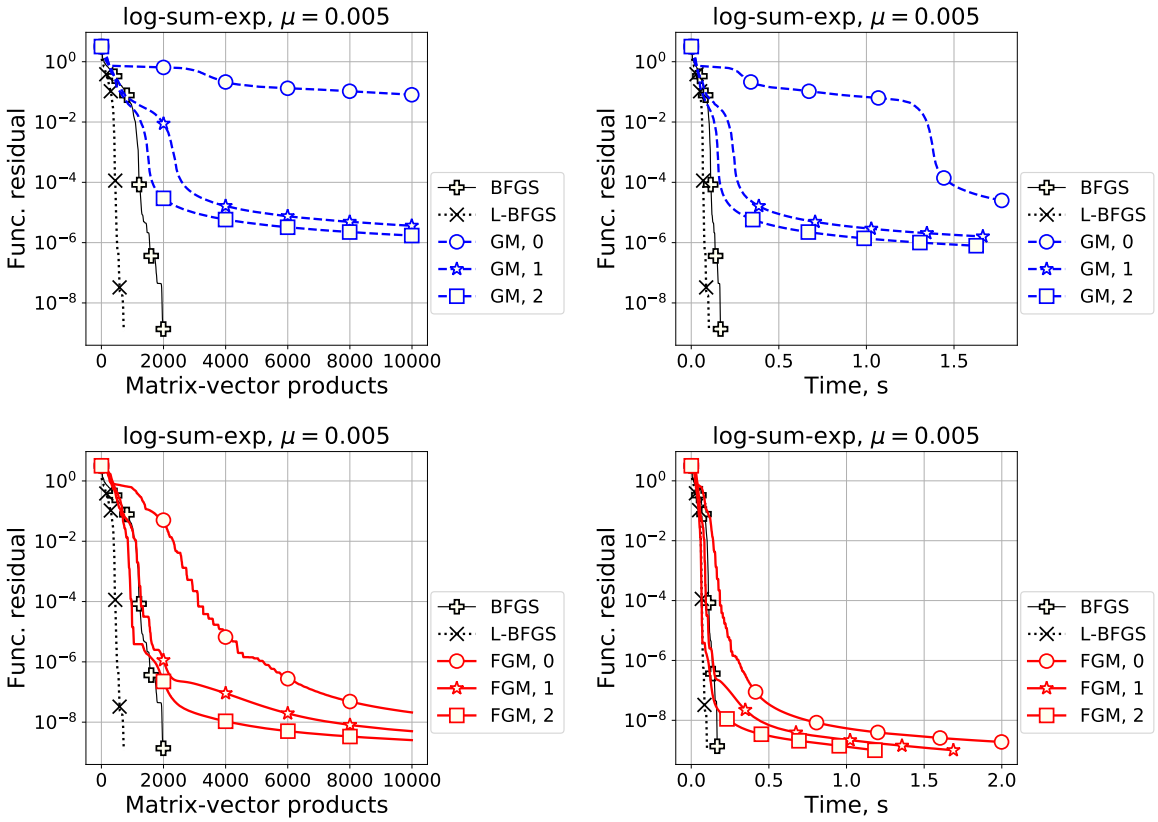


Figure 9: Comparison of our methods with Quasi-Newton methods: training Soft Maximum (log-sum-exp objective)

We see that using our Symmetric Polynomial Preconditioning significantly helps the Gradient Method (GM) and the Fast Gradient Method (FGM). The performance of FGM with preconditioner of order $\tau = 2$ is comparable to that one of the BFGS algorithm, both in terms of the matrix-vector products and total computational time.

B. Proofs

B.1. Proof of Theorem 3.1

Let us consider one iteration of the method, for some $k \geq 0$. By definition, $\boldsymbol{x}_{k+1} = \operatorname{argmin}_{\boldsymbol{y} \in \operatorname{dom} \psi} \{\Omega_k(\boldsymbol{y})\}$, where

$$\Omega_k(\boldsymbol{y}) \stackrel{\text{def}}{=} f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{y} - \boldsymbol{x}_k \rangle + \tfrac{M}{2}\|\boldsymbol{y} - \boldsymbol{x}_k\|^2_{\boldsymbol{P}^{-1}} + \psi(\boldsymbol{y})$$

is strongly convex with respect to $\boldsymbol{P}^{-1}$ norm with parameter $M := \beta L$. Thus, we have, for any $\boldsymbol{y} \in \operatorname{dom} \psi$:

$$\begin{aligned} &\tfrac{M}{2}\|\boldsymbol{y} - \boldsymbol{x}_k\|^2_{\boldsymbol{P}^{-1}} + F(\boldsymbol{y}) \;\geq\; \Omega_k(\boldsymbol{y}) \;\geq\; \Omega_k(\boldsymbol{x}_{k+1}) + \tfrac{M}{2}\|\boldsymbol{y} - \boldsymbol{x}_{k+1}\|^2_{\boldsymbol{P}^{-1}} \\ &\geq\; f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \rangle + \tfrac{L}{2}\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|^2_{\boldsymbol{B}} + \psi(\boldsymbol{x}_{k+1}) + \tfrac{M}{2}\|\boldsymbol{y} - \boldsymbol{x}_{k+1}\|^2_{\boldsymbol{P}^{-1}} \\ &\stackrel{(6)}{\geq}\; F(\boldsymbol{x}_{k+1}) + \tfrac{M}{2}\|\boldsymbol{y} - \boldsymbol{x}_{k+1}\|^2_{\boldsymbol{P}^{-1}}. \end{aligned} \tag{23}$$

Hence, substituting $\boldsymbol{y} := \boldsymbol{x}^\star$ (solution to the problem), we establish the boundness for all iterates:

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^\star\|_{\boldsymbol{P}^{-1}} \;\leq\; \|\boldsymbol{x}_k - \boldsymbol{x}^\star\|_{\boldsymbol{P}^{-1}}. \tag{24}$$

Further, let us take $\boldsymbol{y} := \gamma_k \boldsymbol{x}^\star + (1-\gamma_k)\boldsymbol{x}_k$, for some $\gamma_k \in [0,1]$. We obtain

$$\begin{aligned} F(\boldsymbol{x}_{k+1}) &\stackrel{(23)}{\leq} F(\gamma_k \boldsymbol{x}^\star + (1-\gamma_k)\boldsymbol{x}_k) + \tfrac{\gamma_k^2 M}{2}\|\boldsymbol{x}^\star - \boldsymbol{x}_k\|^2_{\boldsymbol{P}^{-1}} \\ &\leq \gamma_k F^\star + (1-\gamma_k)F(\boldsymbol{x}_k) + \tfrac{\gamma_k^2 M}{2}\|\boldsymbol{x}^\star - \boldsymbol{x}_k\|^2_{\boldsymbol{P}^{-1}}. \end{aligned} \tag{25}$$

Now, setting $A_k \stackrel{\text{def}}{=} k \cdot (k+1)$, $a_{k+1} \stackrel{\text{def}}{=} A_{k+1} - A_k = 2(k+1)$, and $\gamma_k := \frac{a_{k+1}}{A_{k+1}} = \frac{2}{k+2}$, we obtain

$$\begin{aligned} A_{k+1}\big(F(\boldsymbol{x}_{k+1}) - F^\star\big) &\stackrel{(25)}{\leq} A_k\big(F(\boldsymbol{x}_k) - F^\star\big) + \tfrac{a_{k+1}^2}{A_{k+1}} \cdot \tfrac{M}{2}\|\boldsymbol{x}^* - \boldsymbol{x}_k\|^2_{\boldsymbol{P}^{-1}} \\ &\stackrel{(24)}{\leq} A_k\big(F(\boldsymbol{x}_k) - F^\star\big) + \tfrac{a_{k+1}^2}{A_{k+1}} \cdot \tfrac{M}{2}\|\boldsymbol{x}^* - \boldsymbol{x}_0\|^2_{\boldsymbol{P}^{-1}} \\ &\stackrel{(7)}{\leq} A_k\big(F(\boldsymbol{x}_k) - F^\star\big) + \tfrac{a_{k+1}^2}{A_{k+1}} \cdot \tfrac{\beta}{\alpha} \cdot \tfrac{L}{2}\|\boldsymbol{x}^* - \boldsymbol{x}_0\|^2_{\boldsymbol{B}}. \end{aligned} \tag{26}$$

Telescoping this bound for the first k iterations, we get

$$F(\boldsymbol{x}_k) - F^\star \stackrel{(26)}{\leq} \tfrac{\beta}{\alpha} \cdot \tfrac{L}{2}\|\boldsymbol{x}^* - \boldsymbol{x}_0\|^2_{\boldsymbol{B}} \cdot \tfrac{1}{A_k}\sum_{i=1}^{k} \tfrac{a_i^2}{A_i} \;=\; \mathcal{O}\Big(\tfrac{\beta}{\alpha} \cdot \tfrac{L}{2k}\|\boldsymbol{x}^* - \boldsymbol{x}_0\|^2_{\boldsymbol{B}}\Big).$$

To prove the linear rate for the strongly convex case, we continue as follows

$$F(\boldsymbol{x}_{k+1}) \stackrel{(25),(6)}{\leq} \gamma_k F^\star + (1-\gamma_k)F(\boldsymbol{x}_k) + \gamma_k^2 \cdot \tfrac{\beta L}{\alpha \mu} \cdot \big(F(\boldsymbol{x}_k) - F^\star\big).$$

Choosing $\gamma_k := \frac{\alpha\mu}{2\beta L} < 1$, we get the exponential rate

$$F(\boldsymbol{x}_{k+1}) - F^\star \;\leq\; \Big(1 - \tfrac{\alpha\mu}{4\beta L}\Big)\big(F(\boldsymbol{x}_k) - F^\star\big),$$

which completes the proof. □

B.2. Proof of Theorem 3.2

Let $\boldsymbol{x} \in \operatorname{dom} \psi$ and $k \geq 0$ be arbitrary. From (6), (7), and the fact that $\rho = \alpha\mu$, it follows that

$$F(\boldsymbol{x}) = f(\boldsymbol{x}) + \psi(\boldsymbol{x}) \geq \ell_k(\boldsymbol{x}) + \frac{\rho}{2}\|\boldsymbol{x} - \boldsymbol{y}_k\|^2_{\boldsymbol{P}^{-1}}, \qquad \ell_k(\boldsymbol{x}) \stackrel{\text{def}}{=} f(\boldsymbol{y}_k) + \langle \nabla f(\boldsymbol{y}_k), \boldsymbol{x} - \boldsymbol{y}_k \rangle + \psi(\boldsymbol{x}).$$

Hence,

$$\begin{aligned} &A_k F(\boldsymbol{x}_k) + a_{k+1} F(\boldsymbol{x}) + \frac{1+\rho A_k}{2}\|\boldsymbol{x}-\boldsymbol{v}_k\|^2_{\boldsymbol{P}^{-1}} \\ &\quad \geq A_k \ell_k(\boldsymbol{x}_k) + a_{k+1}\ell_k(\boldsymbol{x}) + \frac{1+\rho A_k}{2}\|\boldsymbol{x}-\boldsymbol{v}_k\|^2_{\boldsymbol{P}^{-1}} + \frac{\rho a_{k+1}}{2}\|\boldsymbol{x}-\boldsymbol{y}_k\|^2_{\boldsymbol{P}^{-1}} \\ &\quad \geq A_k \ell_k(\boldsymbol{x}_k) + a_{k+1}\ell_k(\boldsymbol{x}) + \frac{1+\rho A_{k+1}}{2}\|\boldsymbol{x}-\hat{\boldsymbol{v}}_k\|^2_{\boldsymbol{P}^{-1}} \stackrel{\text{def}}{=} \zeta_k(\boldsymbol{x}), \end{aligned} \tag{27}$$

where the final inequality follows from the convexity of the squared norm and the fact that, according to our definitions,

$$\frac{(1+\rho A_k)\boldsymbol{v}_k + \rho a_{k+1}\boldsymbol{y}_k}{1+\rho A_{k+1}} = (1-\omega_k)\boldsymbol{v}_k + \omega_k \boldsymbol{y}_k = (1-\omega_k)\boldsymbol{v}_k + \omega_k[(1-\theta_k)\boldsymbol{x}_k + \theta_k \hat{\boldsymbol{v}}_k] = \hat{\boldsymbol{v}}_k.$$

Note that ζ_k is a $(1+\rho A_{k+1})$-strongly convex function w.r.t. $\|\cdot\|_{\boldsymbol{P}^{-1}}$, and $\boldsymbol{v}_{k+1}$ is precisely its minimizer. Therefore,

$$\zeta_k(\boldsymbol{x}) \geq \zeta_k(\boldsymbol{v}_{k+1}) + \frac{1+\rho A_{k+1}}{2}\|\boldsymbol{x}-\boldsymbol{v}_{k+1}\|^2_{\boldsymbol{P}^{-1}}. \tag{28}$$

Since ℓ_k is a convex function, we have, by our definition of $\boldsymbol{x}_{k+1}$,

$$A_k \ell_k(\boldsymbol{x}_k) + a_{k+1}\ell_k(\boldsymbol{v}_{k+1}) \geq A_{k+1}\ell_k(\boldsymbol{x}_{k+1}).$$

On the other hand, by the definition of $\boldsymbol{x}_{k+1}$ and $\boldsymbol{y}_k$,

$$\boldsymbol{x}_{k+1} - \boldsymbol{y}_k = \theta_k(\boldsymbol{v}_{k+1} - \hat{\boldsymbol{v}}_k) = \frac{a_{k+1}}{A_{k+1}}(\boldsymbol{v}_{k+1} - \hat{\boldsymbol{v}}_k).$$

Therefore,

$$\begin{aligned} \zeta_k(\boldsymbol{v}_{k+1}) &= A_k\ell_k(\boldsymbol{x}_k) + a_{k+1}\ell_k(\boldsymbol{v}_{k+1}) + \frac{1+\rho A_{k+1}}{2}\|\boldsymbol{v}_{k+1}-\hat{\boldsymbol{v}}_k\|^2_{\boldsymbol{P}^{-1}} \\ &\geq A_{k+1}\Big[\ell_k(\boldsymbol{x}_{k+1}) + \frac{A_{k+1}(1+\rho A_{k+1})}{2a^2_{k+1}}\|\boldsymbol{x}_{k+1}-\boldsymbol{y}_k\|^2_{\boldsymbol{P}^{-1}}\Big]. \end{aligned}$$

In view of our choice of a_{k+1}, we have the following identity:

$$\frac{M a^2_{k+1}}{A_{k+1}} = 1 + \rho A_{k+1}. \tag{29}$$

Combining this with the fact that $M = \beta L$ and using (7) and (6), we get

$$\begin{aligned} \zeta_k(\boldsymbol{v}_{k+1}) &\geq A_{k+1}\Big[\ell_k(\boldsymbol{x}_{k+1}) + \frac{M}{2}\|\boldsymbol{x}_{k+1}-\boldsymbol{y}_k\|^2_{\boldsymbol{P}^{-1}}\Big] \geq A_{k+1}\Big[\ell_k(\boldsymbol{x}_{k+1}) + \frac{L}{2}\|\boldsymbol{x}_{k+1}-\boldsymbol{y}_k\|^2_B\Big] \\ &= A_{k+1}\Big[f(\boldsymbol{y}_k) + \langle \nabla f(\boldsymbol{y}_k), \boldsymbol{x}_{k+1}-\boldsymbol{y}_k\rangle + \frac{L}{2}\|\boldsymbol{x}_{k+1}-\boldsymbol{y}_k\|^2_B + \psi(\boldsymbol{x}_{k+1})\Big] \geq A_{k+1}F(\boldsymbol{x}_{k+1}). \end{aligned}$$

Substituting the above bound into (28), and that one into (28), we thus obtain

$$A_k F(\boldsymbol{x}_k) + a_{k+1}F(\boldsymbol{x}) + \frac{1+\rho A_k}{2}\|\boldsymbol{x}-\boldsymbol{v}_k\|^2_{\boldsymbol{P}^{-1}} \geq A_{k+1}F(\boldsymbol{x}_{k+1}) + \frac{1+\rho A_{k+1}}{2}\|\boldsymbol{x}-\boldsymbol{v}_{k+1}\|^2_{\boldsymbol{P}^{-1}}.$$

This inequality is valid for any $k \geq 0$.

Fixing an arbitrary $k \geq 1$ and summing up the previous inequalities for all indices $k' = 0, \ldots, k-1$, we get

$$A_k F(\boldsymbol{x}_k) \leq A_k F(\boldsymbol{x}) + \frac{1+\rho A_0}{2}\|\boldsymbol{x}-\boldsymbol{v}_0\|^2_{\boldsymbol{P}^{-1}} = A_k F(\boldsymbol{x}) + \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{x}_0\|^2_{\boldsymbol{P}^{-1}}.$$

Substituting further $\boldsymbol{x} = \boldsymbol{x}^\star$ (an optimal solution) and using (7), gives us the following convergence rate estimate:

$$F(\boldsymbol{x}_k) - F^\star \leq \frac{\|\boldsymbol{x}^\star - \boldsymbol{x}_0\|^2_{\boldsymbol{P}^{-1}}}{2A_k} \leq \frac{\|\boldsymbol{x}^\star - \boldsymbol{x}_0\|^2_B}{2\alpha A_k}. \tag{30}$$

To complete the proof, it remains to use standard lower bounds on A_k (see (Nesterov, 2018)). Specifically, dropping the second term from the right-hand side of (29) and rearranging, we obtain, for any $k \geq 0$,

$$\sqrt{\frac{A_{k+1}}{M}} \leq a_{k+1} = A_{k+1} - A_k = (\sqrt{A_{k+1}} - \sqrt{A_k})(\sqrt{A_{k+1}} + \sqrt{A_k}) \leq 2(\sqrt{A_{k+1}} - \sqrt{A_k})\sqrt{A_{k+1}}.$$

Cancelling $\sqrt{A_{k+1}}$ on both sides and using the fact that $A_0 = 0$, we obtain, for any $k \geq 1$,

$$\sqrt{A_k} \geq \frac{k}{2\sqrt{M}}.$$

Squaring both sides, substituting the resulting inequality into (30) and replacing $M = \beta L$, we get (10).

When $\mu > 0$, we can drop the first term from the right-hand side of (29). This gives us

$$a_{k+1}^2 \geq \frac{\rho}{M} A_{k+1}^2.$$

Hence, for any $k \geq 0$,

$$A_{k+1} - A_k = a_{k+1} \geq q A_{k+1}, \qquad q \overset{\text{def}}{=} \sqrt{\frac{\rho}{M}} \leq 1,$$

or, equivalently,

$$A_{k+1} \geq \frac{A_k}{1-q}.$$

Consequently, for any $k \geq 1$,

$$A_k \geq \frac{A_1}{(1-q)^{k-1}} \geq \frac{1}{M(1-q)^{k-1}},$$

where the final inequality is due to (29) combined with the fact that $A_0 = 0$. Substituting this inequality into (30) and replacing $M = \beta L$, $\rho = \alpha\mu$, we get the second bound from Theorem 3.2. □

B.3. Proof of Lemma 4.1

Let us denote by $u_k(\boldsymbol{a})$ the k-th *power sum* of the variables:

$$u_k(\boldsymbol{a}) \overset{\text{def}}{=} \sum_{i=1}^{n-1} a_i^k, \qquad \forall \boldsymbol{a} \in \mathbb{R}^{n-1}.$$

Then, the classical Newton-Girard identities (see, e.g. (Kalman, 2000)) state the following relation between the elementary symmetric polynomials:

$$\sigma_\tau(\boldsymbol{a}) \equiv \tfrac{1}{\tau} \sum_{i=1}^{\tau} (-1)^{i-1} \sigma_{\tau-i}(\boldsymbol{a}) \cdot u_i(\boldsymbol{a}). \tag{31}$$

Note that for the matrix $\boldsymbol{U}_\tau \overset{\text{def}}{=} \operatorname{tr}(\boldsymbol{B}^\tau)\boldsymbol{I} - \boldsymbol{B}^\tau$, the following spectral decomposition holds:

$$\begin{aligned} \boldsymbol{U}_\tau &= \boldsymbol{Q}\operatorname{Diag}\Big(\sum_{i=1}^n \lambda_i^\tau - \lambda_1^\tau,\ \sum_{i=1}^n \lambda_i^\tau - \lambda_2^\tau, \ldots, \sum_{i=1}^n \lambda_i^\tau - \lambda_n^\tau\Big)\boldsymbol{Q}^\top \\ &= \boldsymbol{Q}\operatorname{Diag}\Big(u_\tau(\boldsymbol{\lambda}_{-1}), u_\tau(\boldsymbol{\lambda}_{-2}), \ldots, u_\tau(\boldsymbol{\lambda}_{-n})\Big)\boldsymbol{Q}^\top. \end{aligned} \tag{32}$$

Now, the identity that we need to prove is

$$\boldsymbol{P}_\tau = \boldsymbol{Q}\operatorname{Diag}\Big(\sigma_\tau(\boldsymbol{\lambda}_{-1}),\ \sigma_\tau(\boldsymbol{\lambda}_{-2}), \ldots, \sigma_\tau(\boldsymbol{\lambda}_{-n})\Big)\boldsymbol{Q}^\top. \tag{33}$$

We justify (33) by induction. By definition, $\boldsymbol{P}_0 \overset{\text{def}}{=} \boldsymbol{I}$ and $\sigma_0(\boldsymbol{a}) \equiv 1$, therefore (33) holds for $\tau = 0$, which is our base. Let us fix $\tau \geq 1$ and assume that (33) is true for all smaller indices. Then,

$$\begin{aligned}
\boldsymbol{P}_\tau &\overset{\text{def}}{=} \tfrac{1}{\tau}\sum_{i=1}^{\tau}(-1)^{i-1}\boldsymbol{P}_{\tau-i}\boldsymbol{U}_i \\
&\overset{(33),(32)}{=} \boldsymbol{Q}\mathrm{Diag}\Big(\sum_{i=1}^{\tau}(-1)^{i-1}\sigma_{\tau-i}(\boldsymbol{\lambda}_{-1})\cdot u_i(\boldsymbol{\lambda}_{-1}),\,\ldots,\,\sum_{i=1}^{\tau}(-1)^{i-1}\sigma_{\tau-i}(\boldsymbol{\lambda}_{-n})\cdot u_i(\boldsymbol{\lambda}_{-n})\Big)\boldsymbol{Q}^\top \\
&\overset{(31)}{=} \boldsymbol{Q}\mathrm{Diag}\Big(\sigma_\tau(\boldsymbol{\lambda}_{-1}),\,\ldots,\,\sigma_\tau(\boldsymbol{\lambda}_{-n})\Big)\boldsymbol{Q}^\top.
\end{aligned}$$

Hence, (33) is proven for all $0 \leq \tau \leq n-1$. □

B.4. Proof of Theorem 4.2

By Lemma 4.1, we have the following representation of our preconditioner:

$$\boldsymbol{P}_\tau \;=\; \boldsymbol{Q}\mathrm{Diag}\big(\sigma_\tau(\boldsymbol{\lambda}_{-1}),\sigma_\tau(\boldsymbol{\lambda}_{-2}),\ldots,\sigma_\tau(\boldsymbol{\lambda}_{-n})\big)\boldsymbol{Q}^\top.$$

Further, for the spectrum of the matrix

$$\boldsymbol{B}^{1/2}\boldsymbol{P}_\tau\boldsymbol{B}^{1/2} \;=\; \boldsymbol{Q}\mathrm{Diag}\big(\lambda_1\cdot\sigma_\tau(\boldsymbol{\lambda}_{-1}),\lambda_2\cdot\sigma_\tau(\boldsymbol{\lambda}_{-2}),\ldots,\lambda_n\cdot\sigma_\tau(\boldsymbol{\lambda}_{-n})\big)\boldsymbol{Q}^\top,$$

it holds, according to Lemma D.10, that

$$\lambda_1\cdot\sigma_\tau(\boldsymbol{\lambda}_{-1}) \;\geq\; \lambda_2\cdot\sigma_\tau(\boldsymbol{\lambda}_{-2}) \;\geq\; \ldots \;\geq\; \lambda_n\cdot\sigma_\tau(\boldsymbol{\lambda}_{-n}). \tag{34}$$

Consequently,

$$\lambda_n\cdot\sigma_\tau(\boldsymbol{\lambda}_{-n})\boldsymbol{I} \;\preceq\; \boldsymbol{B}^{1/2}\boldsymbol{P}_\tau\boldsymbol{B}^{1/2} \;\preceq\; \lambda_1\cdot\sigma_\tau(\boldsymbol{\lambda}_{-1})\boldsymbol{I},$$

which proves the required bound. □

B.5. Proof of Theorem 4.5

Let $\boldsymbol{B} = \boldsymbol{Q}\,\mathrm{Diag}(\boldsymbol{\lambda})\boldsymbol{Q}^\top$ be a spectral decomposition of $\boldsymbol{B}$, where $\boldsymbol{\lambda} = \boldsymbol{\lambda}(\boldsymbol{B})$ and $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ is an orthogonal matrix. Formula (3.5) in (Rodomanov & Kropotov, 2020) states that

$$\mathbb{E}_{S\sim\mathrm{Vol}_{\tau+1}(\boldsymbol{B})}[\boldsymbol{I}_S(\boldsymbol{B}_{S\times S})^{-1}\boldsymbol{I}_S^\top] = \frac{1}{\sigma_{\tau+1}(\boldsymbol{\lambda})}\boldsymbol{Q}\,\mathrm{Diag}\big(\sigma_\tau(\boldsymbol{\lambda}_{-1}),\ldots,\sigma_\tau(\boldsymbol{\lambda}_{-n})\big)\boldsymbol{Q}^\top.$$

(Their σ_τ is $\sigma_{\tau+1}$ in our notation.) But, according to Lemma 4.1,

$$\boldsymbol{Q}\,\mathrm{Diag}\big(\sigma_\tau(\boldsymbol{\lambda}_{-1}),\ldots,\sigma_\tau(\boldsymbol{\lambda}_{-n})\big)\boldsymbol{Q}^\top = \boldsymbol{P}_\tau,$$

and the claim follows. □

B.6. Proof of Lemma 4.3

Clearly, when $\tau = 0$, we have $\xi_0(\boldsymbol{\lambda}) \equiv 1$. For $\tau = n-1$, inequalities (16) are in fact identities, and $\xi_{n-1}(\boldsymbol{\lambda}) \overset{(15)}{=} \frac{\lambda_n}{\lambda_1}$.

To prove that $\xi_\tau(\boldsymbol{\lambda})$ is decreasing in τ, we need to justify that, for any $0 \leq \tau \leq n-2$, we have

$$\xi_\tau(\boldsymbol{\lambda}) \;\overset{\text{def}}{=}\; \tfrac{\sigma_\tau(\boldsymbol{\lambda}_{-1})}{\sigma_\tau(\boldsymbol{\lambda}_{-n})} \;\geq\; \xi_{\tau+1}(\boldsymbol{\lambda}) \;\overset{\text{def}}{=}\; \tfrac{\sigma_{\tau+1}(\boldsymbol{\lambda}_{-1})}{\sigma_{\tau+1}(\boldsymbol{\lambda}_{-n})}.$$

This follows from Lemma D.11 (recall that, by our assumptions, $\lambda_1 \geq \ldots \geq \lambda_n > 0$).

To prove the limit, let us divide the right hand side of

$$\xi_\tau(\boldsymbol{\lambda}) \;=\; \tfrac{\sigma_\tau(\boldsymbol{\lambda}_{-1})}{\sigma_\tau(\boldsymbol{\lambda}_{-n})} \;\leq\; \frac{\sum_{2\leq i_1<\ldots<i_\tau\leq n-1}\lambda_{i_1}\cdot\ldots\cdot\lambda_{i_\tau}}{\lambda_1\cdot\ldots\cdot\lambda_\tau}$$

by the biggest element from the sum, which is $\lambda_2\cdot\ldots\cdot\lambda_{\tau+1}$. Thus, we get

$$\xi_\tau(\boldsymbol{\lambda}) \;\leq\; \tfrac{1+E}{(\lambda_1/\lambda_{\tau+1})},$$

where E is a finite sum of numbers that are smaller than 1. Hence, $\xi_\tau(\boldsymbol{\lambda}) \to 0$ when $\frac{\lambda_1}{\lambda_{\tau+1}} \to \infty$. □

B.7. Proof of Lemma 4.4

We can assume that $\tau \geq 1$ since otherwise the inequality is trivial ($\xi_0(\boldsymbol{\lambda}) \leq 1$).

Using the definition of $\xi_\tau(\boldsymbol{\lambda})$ and applying Lemma D.12 (recalling that $\lambda_1 \geq \ldots \geq \lambda_n > 0$ by our assumption), we obtain

$$\xi_\tau(\boldsymbol{\lambda}) = \frac{\sigma_\tau(\boldsymbol{\lambda}_{-1})}{\sigma_\tau(\boldsymbol{\lambda}_{-n})} \leq \frac{\lambda_n + s_{\tau-1}}{\lambda_1 + s_{\tau-1}},$$

where $s_{\tau-1}$ is the sum of all but the $\tau - 1$ largest elements of $\boldsymbol{\lambda}_{-\{1,n\}} = (\lambda_2, \ldots, \lambda_{n-1})$:

$$s_{\tau-1} = \sum_{i=\tau+1}^{n-1} \lambda_i.$$

Substituting this into the previous display, we get

$$\xi_\tau(\boldsymbol{\lambda}) \leq \frac{\lambda_n + \sum_{i=\tau+1}^{n-1} \lambda_i}{\lambda_1 + \sum_{i=\tau+1}^{n-1} \lambda_i} = \frac{\sum_{i=\tau+1}^{n} \lambda_i}{\lambda_1 + \sum_{i=\tau+1}^{n-1} \lambda_i},$$

which is exactly the desired inequality.

B.8. Proof of Proposition 5.2

The problem of finding the best polynomial preconditioner can be reformulated as minimizing the norm of a symmetric matrix over the set of (positive) polynomials of a fixed degree $\tau \geq 0$:

$$\min_{p_\tau \in \mathcal{P}_\tau} \Big\{ \gamma(p_\tau) \quad \stackrel{\text{def}}{=} \quad \|\boldsymbol{B} p_\tau(\boldsymbol{B}) - \boldsymbol{I}\| \Big\},$$

where $\mathcal{P}_\tau \stackrel{\text{def}}{=} \big\{p_\tau \in \mathbb{R}[s] \,:\, \deg(p_\tau) = \tau,\ p_\tau(\boldsymbol{B}) \succ 0\big\}$. Here we use the spectral norm to measure the size of a symmetric matrix, and the objective can be rewritten as

$$\gamma(p_\tau) \quad = \quad \max_{s \in \mathrm{Spec}\,(\boldsymbol{B})} |s p_\tau(s) - 1|, \tag{35}$$

where $\mathrm{Spec}\,(\boldsymbol{B})$ is the discrete set of eigenvalues of the curvature matrix. For any value of $\gamma := \gamma(p_\tau)$, our original approximation guarantee (7) clearly satisfied with $\beta = 1 + \gamma$ and $\alpha = 1 - \gamma$, and the condition number becomes[10]

$$\tfrac{\beta}{\alpha} \quad = \quad \tfrac{1+\gamma}{1-\gamma}. \tag{36}$$

Now, we take

$$q_\tau(s) \quad := \quad \big(1 - \tfrac{s}{\lambda_1}\big)\big(1 - \tfrac{s}{\lambda_2}\big) \cdot \ldots \cdot \big(1 - \tfrac{s}{\lambda_\tau}\big). \tag{37}$$

First, note that $q_\tau(0) = 1$ and thus the polynomial $1 + q_\tau(s) \cdot (\alpha s - 1)$ is divisible by s. Hence the degree of the polynomial

$$p_\tau(s) \quad := \quad \tfrac{1 + q_\tau(s) \cdot (\alpha s - 1)}{s}$$

is exactly τ. Then, we obtain

$$\begin{aligned} \gamma \quad &= \quad \max_{s \in \mathrm{Spec}\,(\boldsymbol{B})} |s p_\tau(s) - 1| \quad = \quad \max_{s \in \mathrm{Spec}\,(\boldsymbol{B})} |q_\tau(s) \cdot (\alpha s - 1)| \\ &\leq \quad \max_{s \in \{\lambda_{\tau+1}, \ldots, \lambda_n\}} |\alpha s - 1| \quad = \quad \tfrac{\lambda_{\tau+1} - \lambda_n}{\lambda_{\tau+1} + \lambda_n}, \end{aligned} \tag{38}$$

and the optimal value is $\alpha = \frac{2}{\lambda_{\tau+1} + \lambda_n}$, where we put formally $\lambda_{n+1} \equiv \lambda_n$. It remains to substitute this bound into

$$\tfrac{\beta}{\alpha} \quad = \quad \tfrac{1+\gamma}{1-\gamma},$$

which is monotone in γ. □

[10] We are interested in $\gamma < 1$, since $\gamma = 1$ trivially holds for zero polynomial.

B.9. Proof of Proposition 5.3

Let us use an upper bound on γ from (35), which is the *uniform* polynomial approximation for the whole interval $[\lambda_n, \lambda_1]$:

$$\gamma(p_\tau) \quad \leq \quad \max_{s\in[\lambda_n,\lambda_1]} |sp_\tau(s) - 1|. \tag{39}$$

Then, we use

$$Q_\tau(s) \quad \stackrel{\text{def}}{=} \quad T_{\tau+1}\big(\tfrac{\lambda_1+\lambda_n-2s}{\lambda_1-\lambda_n}\big) \cdot T_{\tau+1}\big(\tfrac{\lambda_1+\lambda_n}{\lambda_1-\lambda_n}\big)^{-1}, \tag{40}$$

where $T_{\tau+1}(\cdot)$ is the standard Chebyshev polynomial of the first kind of degree $\tau+1$. Namely, we can define them recursively:

$$T_0(x) \quad \stackrel{\text{def}}{=} \quad 1, \qquad T_1(x) \quad \stackrel{\text{def}}{=} \quad x, \qquad T_{k+1}(x) \quad \stackrel{\text{def}}{=} \quad 2x \cdot T_k(x) - T_{k-1}(x), \qquad k \geq 1.$$

Note that $Q_\tau(0) = 1$, thus the polynomial $1 - Q_\tau(s)$ is divisible by s. Then, we take

$$p_\tau(s) \quad := \quad \tfrac{1-Q_\tau(s)}{s},$$

which is the polynomial of degree τ. This choice ensures that

$$\begin{aligned} \gamma \quad &\stackrel{(39),(40)}{\leq} \quad \max_{x\in[-1,1]} |T_{\tau+1}(x)| \cdot T_{\tau+1}\big(\tfrac{\lambda_1+\lambda_n}{\lambda_1-\lambda_n}\big)^{-1} \\ &= \quad T_{\tau+1}\big(\tfrac{\lambda_1+\lambda_n}{\lambda_1-\lambda_n}\big)^{-1} \quad \leq \quad 2\Big(\tfrac{\sqrt{\lambda_1}-\sqrt{\lambda_n}}{\sqrt{\lambda_1}+\sqrt{\lambda_n}}\Big)^{\tau+1}, \end{aligned}$$

where the last inequality is the classical bound for the Chebyshev polynomials (see, e.g. Section 16.4 in (Vishnoi, 2013)). Thus, the condition number

$$\tfrac{\beta}{\alpha} \quad = \quad \tfrac{1+\gamma}{1-\gamma}$$

decreases exponentially with τ. □

B.10. Proof of Theorem 5.1

Let us fix an arbitrary $\boldsymbol{P} = \boldsymbol{P}^\top \succ 0$ such that $\boldsymbol{P} = p_\tau(\boldsymbol{B})$, for some polynomial $p_\tau \in \mathbb{R}[s]$ and $\deg(p_\tau) = \tau$. We take $\beta := \beta(\boldsymbol{P})$ and $\alpha := \alpha(\boldsymbol{P})$ (from the definition (7)) and denote

$$\bar{\boldsymbol{P}} \quad := \quad \tfrac{1}{\beta L}\boldsymbol{P}.$$

Let us consider an arbitrary iteration $k \geq 0$, and denote the following step

$$\boldsymbol{T} \quad := \quad \boldsymbol{x}_k - \bar{\boldsymbol{P}}\nabla f(\boldsymbol{x}_k).$$

Recall also that

$$\boldsymbol{x}_{k+1} \quad := \quad \boldsymbol{x}_k - \boldsymbol{P}_{\boldsymbol{a}_k}\nabla f(\boldsymbol{x}_k).$$

By the optimality of $\boldsymbol{a}_k$ as the projection of $\boldsymbol{B}^{-1}\nabla f(\boldsymbol{x}_k)$ onto the Krylov subspace, we have:

$$\begin{aligned} &\langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_{k+1} - \boldsymbol{x}_k\rangle + \tfrac{L}{2}\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|^2_{\boldsymbol{B}} \quad \equiv \quad \tfrac{L}{2}\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k + \tfrac{1}{L}\boldsymbol{B}^{-1}\nabla f(\boldsymbol{x}_k)\|^2_{\boldsymbol{B}} - \tfrac{1}{2L}\|\nabla f(\boldsymbol{x}_k)\|^2_{\boldsymbol{B}^{-1}} \\ \leq \quad &\tfrac{L}{2}\|\boldsymbol{T} - \boldsymbol{x}_k + \tfrac{1}{L}\boldsymbol{B}^{-1}\nabla f(\boldsymbol{x}_k)\|^2_{\boldsymbol{B}} - \tfrac{1}{2L}\|\nabla f(\boldsymbol{x}_k)\|^2_{\boldsymbol{B}^{-1}} \quad \equiv \quad \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{T} - \boldsymbol{x}_k\rangle + \tfrac{L}{2}\|\boldsymbol{T} - \boldsymbol{x}_k\|^2_{\boldsymbol{B}}. \end{aligned} \tag{41}$$

Hence, we obtain, for any $\boldsymbol{y} \in \mathbb{R}^n$:

$$\begin{aligned} f(\boldsymbol{x}_{k+1}) \quad &\stackrel{(6)}{\leq} \quad f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_{k+1} - \boldsymbol{x}_k\rangle + \tfrac{L}{2}\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|^2_{\boldsymbol{B}} \\ &\stackrel{(41)}{\leq} \quad f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{T} - \boldsymbol{x}_k\rangle + \tfrac{L}{2}\|\boldsymbol{T} - \boldsymbol{x}_k\|^2_{\boldsymbol{B}} \\ &\leq \quad f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{T} - \boldsymbol{x}_k\rangle + \tfrac{\beta L}{2}\|\boldsymbol{T} - \boldsymbol{x}_k\|^2_{\boldsymbol{P}^{-1}} \\ &\leq \quad f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{y} - \boldsymbol{x}_k\rangle + \tfrac{\beta L}{2}\|\boldsymbol{y} - \boldsymbol{x}_k\|^2_{\boldsymbol{P}^{-1}}. \end{aligned}$$

where we used that $\boldsymbol{T}$ is the minimizer of the last upper bound in $\boldsymbol{y}$. Thus, using convexity, we get, for any $\boldsymbol{y} \in \mathbb{R}^n$:

$$f(\boldsymbol{x}_{k+1}) \quad \leq \quad f(\boldsymbol{y}) + \tfrac{\beta L}{2}\|\boldsymbol{y} - \boldsymbol{x}_k\|^2_{P^{-1}} \quad \leq \quad f(\boldsymbol{y}) + \tfrac{\beta}{\alpha} \cdot \tfrac{L}{2}\|\boldsymbol{y} - \boldsymbol{x}_k\|^2_{\boldsymbol{B}}. \tag{42}$$

In particular, substituting $\boldsymbol{y} := \boldsymbol{x}_k$ we justify that the method is *monotone*: $f(\boldsymbol{x}_{k+1}) \leq f(\boldsymbol{x}_k), \forall k \geq 0$. Therefore,

$$\boldsymbol{x}_k \quad \in \quad \mathcal{F}_0 \quad \stackrel{\text{def}}{=} \quad \left\{ \boldsymbol{x} \in \mathbb{R}^n \;:\; f(\boldsymbol{x}) \leq f(\boldsymbol{x}_0), \right\}$$

and we assume that the initial level set $\mathcal{F}_0$ is *bounded*, denoting

$$D_0 \quad \stackrel{\text{def}}{=} \quad \sup_{\boldsymbol{x} \in \mathcal{F}_0} \|\boldsymbol{x} - \boldsymbol{x}^\star\|_{\boldsymbol{B}} \quad < \quad +\infty.$$

Substituting $\boldsymbol{y} := \gamma_k \boldsymbol{x}^\star + (1-\gamma_k)\boldsymbol{x}_k$, $\gamma_k \in [0,1]$ into (42), we obtain

$$\begin{aligned} f(\boldsymbol{x}_{k+1}) \quad &\leq \quad \gamma_k f^\star + (1-\gamma_k) f(\boldsymbol{x}_k) + \gamma_k^2 \tfrac{\beta}{\alpha} \cdot \tfrac{L}{2}\|\boldsymbol{x}^\star - \boldsymbol{x}_k\|^2_{\boldsymbol{B}} \\ &\leq \quad \gamma_k f^\star + (1-\gamma_k) f(\boldsymbol{x}_k) + \gamma_k^2 \tfrac{\beta}{\alpha} \cdot \tfrac{L}{2} D_0^2. \end{aligned} \tag{43}$$

Substituting $\gamma_k := \frac{2}{k+1}$ and using the standard technique (see the proof of Theorem 3.1), we establish the global rate for the convex case:

$$f(\boldsymbol{x}_k) - f^\star \quad \leq \quad \mathcal{O}\Big(\tfrac{\beta}{\alpha} \cdot \tfrac{LD_0^2}{k} \Big).$$

For strongly convex functions ($\mu > 0$), we continue as

$$f(\boldsymbol{x}_{k+1}) \quad \stackrel{(43),(6)}{\leq} \quad \gamma_k f^\star + (1-\gamma_k) f(\boldsymbol{x}_k) + \gamma_k^2 \tfrac{\beta L}{\alpha \mu} \cdot \big(f(\boldsymbol{x}_k) - f^\star \big),$$

and choosing $\gamma_k := \frac{\alpha\mu}{2\beta L}$ we establish the exponential rate. □

C. Adaptive Search

In this section, we briefly present adaptive versions of Algorithms 1 and 2 which do not require the knowledge of the constant $M = \beta L$ and can automatically "tune" it in iterations yet preserving the original worst-case efficiency estimates. This is achieved by using a standard "backtracking line search" which can be found, e.g., in (Nesterov, 2013).

In what follows, for any $\boldsymbol{x}, \boldsymbol{y} \in \operatorname{dom}\psi$, $M > 0$ and $\boldsymbol{P} = \boldsymbol{P}^\top \succ 0$, we define the following predicate:

$$\text{QuadGrowth}_{M,\boldsymbol{P}}(\boldsymbol{x}, \boldsymbol{y}): \quad f(\boldsymbol{y}) \leq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \frac{M}{2}\|\boldsymbol{y} - \boldsymbol{x}\|^2_{\boldsymbol{P}^{-1}}.$$

According to our assumptions (6) and (7), we know that this predicate is surely satisfied for any pair of points once $M \geq \beta L$.

The adaptive version of Algorithm 1 is presented in Algorithm 4. This method starts with a certain initial guess $\tilde{M}_0$ for the constant βL and then, at every iteration, repeatedly increases the current guess in two times until the predicate becomes satisfied. This process is guaranteed to terminate (when M_k becomes bigger or equal to βL, or even sooner). After that, we accept the new point x_{k+1} and choose a new "optimistic" guess of the constant M for the next iteration by halving the value of M_k that we have accepted at the current iteration.

We assume that the preconditioner $\boldsymbol{P}$ is sufficiently simple so that we can efficiently check the predicate $\text{QuadGrowth}_{M_k,\boldsymbol{P}}(\boldsymbol{x}_k, \boldsymbol{x}_{k+1})$. For example, if $\psi = 0$, then $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \frac{1}{M_k}\boldsymbol{P}\nabla f(\boldsymbol{x}_k)$ and $M_k\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|^2_{\boldsymbol{P}^{-1}} = \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_k - \boldsymbol{x}_{k+1} \rangle$ can be efficiently computed.

For Algorithm 4, we can prove exactly the same rates as in Theorem 3.1 (up to absolute constants) provided that

$$\tilde{M}_0 \leq \beta L. \tag{44}$$

The proof is essentially the same as in Appendix B.1 with only two minor differences: 1) inequality (23) is now guaranteed by our predicate; 2) instead of using $M = \beta L$ in (26), we should use the bound $M_k \leq 2\beta L$ which follows from (44) and

Algorithm 4 Adaptive Preconditioned GM

Input: $\boldsymbol{x}_0 \in \operatorname{dom}\psi$, $\boldsymbol{P} = \boldsymbol{P}^\top \succ 0$, $\tilde{M}_0 > 0$.
for $k = 0, 1, \ldots$ **do**
Find smallest integer $i_k \geq 0$ such that

$$\boldsymbol{x}_{k+1} = \mathrm{GradStep}_{M_k,\boldsymbol{P}}\big(\boldsymbol{x}_k, \nabla f(\boldsymbol{x}_k)\big), \quad M_k = 2^{i_k}\tilde{M}_k$$

satisfies the predicate $\mathrm{QuadGrowth}_{M_k,\boldsymbol{P}}(\boldsymbol{x}_k, \boldsymbol{x}_{k+1})$.
Set $\tilde{M}_{k+1} = M_k/2$.
end for

the fact that any value of $M \geq \beta L$ is always acceptable in the line search. Using a classical argument from (Nesterov, 2013), it is not difficult to show that, on average, Algorithm 4 makes only ~ 2 gradient steps at each iteration.

In contrast to an upper estimate of the constant βL, an initial guess satisfying (44) can be easily generated. One simple recipe is to make a trial step $\boldsymbol{x}_1' = \mathrm{GradStep}_{M_0',\boldsymbol{P}}\big(\boldsymbol{x}_0, \nabla f(\boldsymbol{x}_0)\big)$ for an *arbitrarily chosen* $M_0' > 0$ and then compute

$$\tilde{M}_0 = \frac{f(\boldsymbol{x}_1') - f(\boldsymbol{x}_0) - \langle \nabla f(\boldsymbol{x}_0), \boldsymbol{x}_1' - \boldsymbol{x}_0\rangle}{\frac{1}{2}\|\boldsymbol{x}_1' - \boldsymbol{x}_0\|^2_{\boldsymbol{P}^{-1}}}.$$

Alternatively, we can find a suitable $\tilde{M}_0$ be choosing an arbitrary $M_0' > 0$ and then repeatedly halving it until the predicate $\mathrm{QuadGrowth}(\boldsymbol{x}_0, \boldsymbol{x}_1'(M))$ stops being satisfied for $\boldsymbol{x}_1'(M) = \mathrm{GradStep}_{M,\boldsymbol{P}}(\boldsymbol{x}_0, \nabla f(\boldsymbol{x}_0))$. This auxiliary procedure either terminates in a logarithmic number of steps, in which case we get a suitable $\tilde{M}_0$, or, otherwise, we quickly find an approximate solution of our problem.

Similar technique can be applied for the Fast Gradient Method. Specifically, let us introduce an auxiliary procedure shown in Algorithm 5 for computing one iteration of Algorithm 2 for a given value of M. Then, the adaptive FGM method can be constructed as shown in Algorithm 6. As in the basic method, we can show that the rates from Theorem 3.2 still remain valid (up to absolute constants) for Algorithm 6, provided that $\tilde{M}_0$ satisfies (44). For generating the initial guess $\tilde{M}_0$, we can use exactly the same techniques as before.

Algorithm 5 $(\boldsymbol{x}_+, \boldsymbol{v}_+, A_+; \boldsymbol{y}) = \mathrm{FastGradStep}_{M,\rho,\boldsymbol{P}}(\boldsymbol{x}, \boldsymbol{v}, A)$

Require: $M > 0$; $\rho \geq 0$; $\boldsymbol{P} = \boldsymbol{P}^\top \succ 0$; $\boldsymbol{x}, \boldsymbol{v} \in \operatorname{dom}\psi$; $A > 0$.
Find a_+ from eq. $\frac{Ma_+^2}{A+a_+} = 1 + \rho(A + a_+)$.
Set $A_+ = A + a_+$, $H = \frac{1+\rho A_+}{a_+}$, $\theta = \frac{a_+}{A_+}$, $\omega = \frac{\rho}{H}$, $\gamma = \frac{\omega(1-\theta)}{1-\omega\theta}$.
Set $\hat{\boldsymbol{v}} = (1-\gamma)\boldsymbol{v} + \gamma\boldsymbol{x}$, $\boldsymbol{y} = (1-\theta)\boldsymbol{x} + \theta\hat{\boldsymbol{v}}$.
Compute $\boldsymbol{v}_+ = \mathrm{GradStep}_{H,\boldsymbol{P}}\big(\hat{\boldsymbol{v}}, \nabla f(\boldsymbol{y})\big)$.
Set $\boldsymbol{x}_+ = (1-\theta)\boldsymbol{x} + \theta\boldsymbol{v}_+$.
Return $(\boldsymbol{x}_+, \boldsymbol{v}_+, A_+; \boldsymbol{y})$.

Algorithm 6 Adaptive Preconditioned FGM

Input: $\boldsymbol{x}_0 \in \operatorname{dom}\psi$, $\boldsymbol{P} = \boldsymbol{P}^\top \succ 0$, $\tilde{M}_0 > 0$.
Set $\boldsymbol{v}_0 = \boldsymbol{x}_0$, $A_0 = 0$.
for $k = 0, 1, \ldots$ **do**
Find smallest integer $i_k \geq 0$ such that

$$(\boldsymbol{x}_{k+1}, \boldsymbol{v}_{k+1}, A_{k+1}; \boldsymbol{y}_k) = \mathrm{FastGradStep}_{M_k,\boldsymbol{P}}\big(\boldsymbol{x}_k, \boldsymbol{v}_k, A_k\big), \qquad M_k = 2^{i_k}\tilde{M}_k$$

satisfies the predicate $\mathrm{QuadGrowth}_{M_k,\boldsymbol{P}}(\boldsymbol{y}_k, \boldsymbol{x}_{k+1})$.
Set $\tilde{M}_{k+1} = M_k/2$.
end for

D. Auxiliary Results

D.1. Elementary Symmetric Polynomials

The *elementary symmetric polynomial* in variables $\boldsymbol{x} \in \mathbb{R}^n$ of degree k (integer, $1 \leq k \leq n$) is defined as

$$\sigma_k(\boldsymbol{x}) \coloneqq \sum_{1 \leq i_1 < \ldots < i_k \leq n} x_{i_1} \ldots x_{i_k}.$$

It will be convenient to extend this definition to arbitrary integer degrees k and also to the case $n = 0$ (which corresponds to the empty vector $\boldsymbol{x}$). For this, we additionally define, for any $\boldsymbol{x} \in \mathbb{R}^n$ with $n \geq 0$,

$$\sigma_k(\boldsymbol{x}) \coloneqq \begin{cases} 1, & \text{if } k = 0, \\ 0, & \text{if } k < 0 \text{ or } k > n. \end{cases}$$

Thus, $\sigma_k(\boldsymbol{x})$ is defined for any any $\boldsymbol{x} \in \mathbb{R}^n$ with $n \geq 0$ and any integer k.

The following three properties are obvious from the definition.

Observation D.1 (symmetry)**.** *For any $x \in \mathbb{R}^n$ with $n \geq 0$, any integer k, and any permutation $\boldsymbol{\pi} \coloneqq (\pi_1, \ldots, \pi_n)$ of indices $\{1, \ldots, n\}$, we have $\sigma_k(\boldsymbol{x}) = \sigma_k(\boldsymbol{x}_{\boldsymbol{\pi}})$, where $\boldsymbol{x}_{\boldsymbol{\pi}} \coloneqq (x_{\pi_1}, \ldots, x_{\pi_n}) \in \mathbb{R}^n$ is the vector obtained from $\boldsymbol{x}$ by rearranging[11] its components according to $\boldsymbol{\pi}$.*

Observation D.2. *For any $\boldsymbol{x} \in \mathbb{R}^n_+$ with $n \geq 0$ and any integer k, we have $\sigma_k(\boldsymbol{x}) \geq 0$.*

Observation D.3. *For any $\boldsymbol{x} \in \mathbb{R}^n_+$ with at least $1 \leq k \leq n$ strictly positive elements, we have $\sigma_k(\boldsymbol{x}) > 0$.*

Let us now establish a number of other properties that will be useful in our analysis.

In what follows, given a vector $\boldsymbol{x}$ with $n \geq 1$ elements (indexed by $1, \ldots, n$), and an index $1 \leq i \leq n$, we use the notation $\boldsymbol{x}_{-i}$ to denote the $(n-1)$-dimensional vector obtained from $\boldsymbol{x}$ by removing its ith element. More generally, for a set of indices $I \subseteq \{1, \ldots, n\}$, we denote by $\boldsymbol{x}_{-I}$ the $(n - |I|)$-dimensional vector obtained from $\boldsymbol{x}$ by removing the elements with indices from I.

Also, for a vector $\boldsymbol{x}$ with $n \geq 0$ elements, we use the notation $\boldsymbol{x} \geq 0$ to express the fact that each element of $\boldsymbol{x}$ is nonnegative. For an empty vector $\boldsymbol{x}$ (i.e., when $n = 0$), this inequality is always assumed to be satisfied (vacuously).

We start with a simple but very useful recursive decomposition.

Lemma D.4. *For any $\boldsymbol{x} \in \mathbb{R}^n$, any index $1 \leq i \leq n$, and any integer k, we have*

$$\sigma_k(\boldsymbol{x}) = x_i \sigma_{k-1}(\boldsymbol{x}_{-i}) + \sigma_k(\boldsymbol{x}_{-i}).$$

Proof. In view of Observation D.1, it suffices to prove the identity only for $i = 1$.

If $k < 0$ or $k > n$, then $\sigma_k(\boldsymbol{x}) = \sigma_{k-1}(\boldsymbol{x}_{-1}) = \sigma_k(\boldsymbol{x}_{-1}) = 0$, and we get the identity $0 = 0$ which is indeed valid.

If $k = 0$, then $\sigma_{k-1}(\boldsymbol{x}_{-1}) = 0$, while $\sigma_k(\boldsymbol{x}) = \sigma_k(\boldsymbol{x}_{-1}) = 1$, and we get the identity $1 = 1$ which is also valid.

If $k = 1$, then

$$\sigma_k(\boldsymbol{x}) = \sum_{j=1}^{n} x_j = x_1 + \sum_{j=2}^{n} x_j = x_1 + \sigma_1(\boldsymbol{x}_{-1}),$$

so the claim is valid since $\sigma_0(\boldsymbol{x}_{-1}) = 1$ by definition.

Finally, in the general case when $2 \leq k \leq n$, we have

$$\begin{aligned} \sigma_k(\boldsymbol{x}) &= \sum_{1 \leq i_1 < \ldots < i_k \leq n} x_{i_1} \ldots x_{i_k} = x_1 \sum_{2 \leq i_2 < \ldots < i_k \leq n} x_{i_2} \ldots x_{i_k} + \sum_{2 \leq i_1 < \ldots < i_k \leq n} x_{i_1} \ldots x_{i_k} \\ &= x_1 \sigma_{k-1}(\boldsymbol{x}_{-1}) + \sigma_k(\boldsymbol{x}_{-1}). \end{aligned}$$

□

[11] By convention, the empty vector gets rearranged into the empty vector.

Next, we consider several inequalities between elementary symmetric polynomials of different degrees.

Lemma D.5 (Weak Newton's inequality)**.** *For any $\boldsymbol{x} \in \mathbb{R}^n_+$ with $n \geq 0$ and any integer k, we have*

$$\sigma_k^2(\boldsymbol{x}) \geq \sigma_{k-1}(\boldsymbol{x})\sigma_{k+1}(\boldsymbol{x}).$$

Remark D.6. In the only nontrivial case $1 \leq k \leq n-1$, the above is a weaker version of the classical *Newton's inequality*:

$$\hat{\sigma}_k^2(\boldsymbol{x}) \geq \hat{\sigma}_{k-1}(\boldsymbol{x})\hat{\sigma}_{k+1}(\boldsymbol{x}),$$

where, for any $1 \leq k \leq n$, $\hat{\sigma}_k(\boldsymbol{x}) \coloneqq \frac{\sigma_k(\boldsymbol{x})}{\binom{n}{k}}$ is the normalized elementary symmetric polynomial in variables $\boldsymbol{x}$, and $\binom{n}{k}$ is the binomial coefficient. For more information about the two inequalities, see, e.g., Section 2.22 in (Hardy et al., 1952). We present the proof of Lemma D.5 below for the reader's convenience.

Proof. Note that the claim is obvious if either $k < 1$ or $k \geq n$ since then either $\sigma_{k-1}(\boldsymbol{x}) = 0$ or $\sigma_{k+1}(\boldsymbol{x}) = 0$, and we get the trivial inequality $\sigma_k^2(\boldsymbol{x}) \geq 0$.

Let us prove the claim by induction on n.

For $n = 1$, the claim is obvious since, in this case, all values of k satisfy either $k < 1$ or $k \geq n$.

Assume that $n \geq 2$, and that we have already proved the claim for $n' = n - 1$. Let us prove it for $n' = n$.

According to the observation made at the beginning, we can assume that $1 \leq k \leq n-1$ since otherwise the claim is obvious. Since both sides of the claimed inequality are continuous in $\boldsymbol{x}$ (as certain polynomials), we can further assume w.l.o.g. that all the elements of $\boldsymbol{x}$ are strictly positive.

According to Lemma D.4, we can decompose

$$\sigma_k(\boldsymbol{x}) = x_1\sigma_{k-1} + \sigma_k, \qquad \sigma_{k-1}(\boldsymbol{x}) = x_1\sigma_{k-2} + \sigma_{k-1}, \qquad \sigma_{k+1}(\boldsymbol{x}) = x_1\sigma_k + \sigma_{k+1},$$

where $\sigma_{k'} \coloneqq \sigma_{k'}(\boldsymbol{x}_{-1}) \geq 0$ for any $k' \in \{k-2, k-1, k, k+1\}$ (see Observation D.2). Note that $\boldsymbol{x}_{-1}$ has exactly $n - 1 \geq 1$ elements, all of which are strictly positive (since we have assumed that all the elements of $\boldsymbol{x}$ are so). In particular, we can assume that $\sigma_k > 0$ (see Observation D.3 and recall that $1 \leq k \leq n-1$ by our assumption).

The inequality we need to prove is then

$$(x_1\sigma_{k-1} + \sigma_k)^2 \geq (x_1\sigma_{k-2} + \sigma_{k-1})(x_1\sigma_k + \sigma_{k+1}).$$

After the expansion of both sides, the above inequality becomes

$$x_1^2\sigma_{k-1}^2 + 2x_1\sigma_{k-1}\sigma_k + \sigma_k^2 \geq x_1^2\sigma_{k-2}\sigma_k + x_1(\sigma_{k-2}\sigma_{k+1} + \sigma_{k-1}\sigma_k) + \sigma_{k-1}\sigma_{k+1}.$$

Making cancellations and rearranging, we come to the following inequality we need to prove:

$$x_1^2(\sigma_{k-1}^2 - \sigma_{k-2}\sigma_k) + x_1(\sigma_{k-1}\sigma_k - \sigma_{k-2}\sigma_{k+1}) + (\sigma_k^2 - \sigma_{k-1}\sigma_{k+1}) \geq 0.$$

Since $x_1 \geq 0$, it suffices to prove the following three inequalities:

$$\sigma_{k-1}^2 \geq \sigma_{k-2}\sigma_k, \qquad \sigma_{k-1}\sigma_k \geq \sigma_{k-2}\sigma_{k+1}, \qquad \sigma_k^2 \geq \sigma_{k-1}\sigma_{k+1}.$$

But this is simple: the first and the third ones are valid in view of our inductive assumption, while the second one follows from the other two (indeed, $\sigma_{k-1}\sigma_k^2 \geq \sigma_{k-1}^2\sigma_{k+1} \geq \sigma_{k-2}\sigma_k\sigma_{k+1}$, and it remains to cancel σ_k which is assumed to be positive). □

Lemma D.7. *For any $\boldsymbol{x} \in \mathbb{R}^n$, any index $1 \leq i \leq n$, such that $\boldsymbol{x}_{-i} \geq 0$, and any integer k, we have*

$$\sigma_k(\boldsymbol{x})\sigma_k(\boldsymbol{x}_{-i}) \geq \sigma_{k+1}(\boldsymbol{x})\sigma_{k-1}(\boldsymbol{x}_{-i}).$$

Proof. Let us denote for brevity $\sigma_{k'} \coloneqq \sigma_{k'}(\boldsymbol{x}_{-i})$ for any $k' \in \{k-1, k, k+1\}$. Then, according to Lemma D.4,

$$\sigma_k(\boldsymbol{x}) = x_i\sigma_{k-1} + \sigma_k, \qquad \sigma_{k+1}(\boldsymbol{x}) = x_i\sigma_k + \sigma_{k+1}.$$

The inequality we need to justify is then

$$(x_i\sigma_{k-1} + \sigma_k)\sigma_k \geq (x_i\sigma_k + \sigma_{k+1})\sigma_{k-1},$$

or, equivalently,

$$\sigma_k^2 \geq \sigma_{k-1}\sigma_{k+1}.$$

But this is indeed true according to Lemma D.5 (and our assumption that $\boldsymbol{x}_{-i} \geq 0$). □

Lemma D.8. *For any $\boldsymbol{x} \in \mathbb{R}^n_+$ with $n \geq 0$ and any integer $0 \leq k \leq n$, we have*

$$\sigma_{k+1}(\boldsymbol{x}) \leq \bar{s}^{\uparrow}_k(\boldsymbol{x})\sigma_k(\boldsymbol{x}),$$

where $\bar{s}^{\uparrow}_k(\boldsymbol{x})$ is the sum of all but k largest elements of x (i.e., $\bar{s}^{\uparrow}_k(\boldsymbol{x}) = \sum_{i=k+1}^n x_{[i]}$, where $x_{[1]} \geq \ldots \geq x_{[n]}$ are the components of $\boldsymbol{x}$ sorted in nonincreasing order, and $\bar{s}^{\uparrow}_k(\boldsymbol{x}) \coloneqq 0$ for the empty vector $\boldsymbol{x}$).

Proof. We can assume that $n \geq 1$ since otherwise $k = n = 0$, the vector $\boldsymbol{x}$ is empty, $\sigma_{k+1}(\boldsymbol{x}) = \bar{s}^{\uparrow}_k(\boldsymbol{x}) = 0$, and we get a trivial inequality $0 \leq 0$.

Further, in view of Observation D.1, we can assume w.l.o.g. that $x_1 \geq \ldots \geq x_n$. Then, we need to prove that

$$\sigma_{k+1}(\boldsymbol{x}) \leq \Big(\sum_{i=k+1}^n x_i\Big)\sigma_k(\boldsymbol{x}).$$

Since both sides of the above inequality are continuous in $\boldsymbol{x}$ (as certain polynomials), we can assume w.l.o.g. that all the components of $\boldsymbol{x}$ are strictly positive.

Applying repeatedly Lemma D.7, we obtain

$$\frac{\sigma_{k+1}(\boldsymbol{x})}{\sigma_k(\boldsymbol{x})} \leq \frac{\sigma_k(\boldsymbol{x}_{-1})}{\sigma_{k-1}(\boldsymbol{x}_{-1})} \leq \frac{\sigma_{k-1}(\boldsymbol{x}_{-\{1,2\}})}{\sigma_{k-2}(\boldsymbol{x}_{-\{1,2\}})} \leq \ldots \leq \frac{\sigma_1(\boldsymbol{x}_{-\{1,\ldots,k\}})}{\sigma_0(\boldsymbol{x}_{-\{1,\ldots,k\}})} = \sum_{i=k+1}^n x_i,$$

where the final identity follows from the definitions of σ_1 and σ_0. (Note that each denominator in the above display is strictly positive, see Observation D.3 and recall that, by our assumptions, $\boldsymbol{x}$ has strictly positive components and $0 \leq k \leq n$.) This is exactly the desired inequality. □

Lemma D.9. *For any $\boldsymbol{x} \in \mathbb{R}^n$, any indices $1 \leq i, j \leq n$, any integer k, and any $a, b \in \mathbb{R}$, we have the implication*

$$x_i \geq x_j \quad \text{and} \quad a\sigma_k(\boldsymbol{x}_{-\{i,j\}}) \mathbin{\&} b\sigma_{k-1}(\boldsymbol{x}_{-\{i,j\}}) \qquad \Longrightarrow \qquad (ax_i + b)\sigma_k(\boldsymbol{x}_{-i}) \mathbin{\&} (ax_j + b)\sigma_k(\boldsymbol{x}_{-j}),$$

where "$\&$" is either "$\leq$" or "$\geq$". Furthermore, if $x_i > x_j$, then the reverse implication is also true.

Proof. We can assume that $i \neq j$ since otherwise the claim is trivial. In particular, we can assume that $n \geq 2$.

According to Lemma D.4, we can decompose

$$\sigma_k(\boldsymbol{x}_{-i}) = x_j\sigma_{k-1} + \sigma_k, \qquad \sigma_k(\boldsymbol{x}_{-j}) = x_i\sigma_{k-1} + \sigma_k,$$

where $\sigma_{k'} \coloneqq \sigma_{k'}(\boldsymbol{x}_{-\{i,j\}})$ for any $k' \in \{k-1, k\}$. The inequality after the implication sign is then

$$(ax_i + b)(x_j\sigma_{k-1} + \sigma_k) \mathbin{\&} (ax_j + b)(x_i\sigma_{k-1} + \sigma_k).$$

After the expansion of both sides, this inequality reads

$$x_ix_ja\sigma_{k-1} + x_ia\sigma_k + x_jb\sigma_{k-1} + b\sigma_k \mathbin{\&} x_ix_ja\sigma_{k-1} + x_ib\sigma_{k-1} + x_ja\sigma_k + b\sigma_k.$$

Making cancellations and rearranging, we obtain the following equivalent inequality:

$$(x_i - x_j)a\sigma_k \;\&\; (x_i - x_j)b\sigma_{k-1}.$$

This inequality is obviously true if $x_i \geq x_j$ and $a\sigma_k \;\&\; b\sigma_{k-1}$. On the other hand, if $x_i > x_j$, we can cancel $(x_i - x_j)$ on both sides and conclude that $a\sigma_k \;\&\; b\sigma_{k-1}$. □

Lemma D.10. *For any $\boldsymbol{x} \in \mathbb{R}^n$, any indices $1 \leq i, j \leq n$, such that $\boldsymbol{x}_{-\{i,j\}} \geq 0$, and any integer k, we have the implication*

$$x_i \geq x_j \qquad \Longrightarrow \qquad x_i\sigma_k(\boldsymbol{x}_{-i}) \geq x_j\sigma_k(\boldsymbol{x}_{-j}).$$

Proof. Follows from Lemma D.9 (applied to $a = 1$ and $b = 0$) since $\sigma_k(\boldsymbol{x}_{-\{i,j\}}) \geq 0$ in view of our assumption that $\boldsymbol{x}_{-\{i,j\}} \geq 0$ (see Observation D.2). □

Lemma D.11. *For any $\boldsymbol{x} \in \mathbb{R}^n$, any indices $1 \leq i, j \leq n$, such that $\boldsymbol{x}_{-\{i,j\}} \geq 0$, and any integer k, we have the implication*

$$x_i \geq x_j \qquad \Longrightarrow \qquad \sigma_k(\boldsymbol{x}_{-i})\sigma_{k+1}(\boldsymbol{x}_{-j}) \geq \sigma_{k+1}(\boldsymbol{x}_{-i})\sigma_k(\boldsymbol{x}_{-j}).$$

Proof. For brevity, let us denote $\sigma_{k'} \coloneqq \sigma_{k'}(\boldsymbol{x}_{-\{i,j\}})$ for any $k' \in \{k-1, k, k+1\}$. According to Lemma D.5 and our assumption that $\boldsymbol{x}_{-\{i,j\}} \geq 0$, we have $\sigma_k^2 \geq \sigma_{k-1}\sigma_{k+1}$. Since we also assume that $x_i \geq x_j$, we can therefore apply Lemma D.9 with $a = \sigma_k$ and $b = \sigma_{k+1}$ to get

$$(x_i\sigma_k + \sigma_{k+1})\sigma_k(\boldsymbol{x}_{-i}) \geq (x_j\sigma_k + \sigma_{k+1})\sigma_k(\boldsymbol{x}_{-j}).$$

This is exactly the desired inequality since, according to Lemma D.4,

$$\sigma_{k+1}(\boldsymbol{x}_{-i}) = x_j\sigma_k + \sigma_{k+1}, \qquad \sigma_{k+1}(\boldsymbol{x}_{-j}) = x_i\sigma_k + \sigma_{k+1}.$$ □

Lemma D.12. *For any $\boldsymbol{x} \in \mathbb{R}^n$, any indices $1 \leq i, j \leq n$, with $\boldsymbol{x}_{-\{i,j\}} \geq 0$, and any integer[12] $1 \leq k \leq \dim(\boldsymbol{x}_{-\{i,j\}}) + 1$, the following implication holds:*

$$x_i \geq x_j \qquad \Longrightarrow \qquad \sigma_k(\boldsymbol{x}_{-i})\big(x_i + \bar{s}^{\uparrow}_{k-1}(\boldsymbol{x}_{-\{i,j\}})\big) \leq \sigma_k(\boldsymbol{x}_{-j})\big(x_j + \bar{s}^{\uparrow}_{k-1}(\boldsymbol{x}_{-\{i,j\}})\big),$$

where $\bar{s}^{\uparrow}_{k-1}(\boldsymbol{x}_{-\{i,j\}})$ is the sum of all but $k-1$ largest elements of the vector $\boldsymbol{x}_{-\{i,j\}}$.

Proof. Follows from Lemma D.9 applied to $a = 1$ and $b = \bar{s}^{\uparrow}_{k-1}(\boldsymbol{x}_{-\{i,j\}})$ since $\sigma_k(\boldsymbol{x}_{-\{i,j\}}) \leq b\sigma_{k-1}(\boldsymbol{x}_{-\{i,j\}})$ according to Lemma D.8 (applied to $k' = k-1$ and $\boldsymbol{x}' = \boldsymbol{x}_{-\{i,j\}}$; note that $\dim(\boldsymbol{x}') = n - 2 \geq 0$ and $0 \leq k' \leq \dim(\boldsymbol{x}')$ by our assumption). □

[12] Here, $\dim(\boldsymbol{x})$ denotes the number of elements in the vector $\boldsymbol{x}$.