Performance of First- and Second-Order Methods for Large-Scale Optimization

Kimon Fountoulakis[1] and Jacek Gondzio[2]

[1]University of California Berkeley

[2]University of Edinburgh

ISMP, 13 July 2015

Motivation - need for controlled testing

Large-scale sparked the development of new optimization methods

Percentage of large-scale papers within convex optimization since 2012

- ArXiv: 53%
- Optimization Online: 43%
- Journal of Machine Learning Research: 60%
- Springer: 51%

Issue: frequently, the performance of new methods is tested on well-conditioned randomly generated problems

Need: controlled testing – a problem generator which can reveal weaknesses and strengths of new methods

Problem generator for ℓ_1 least-squares

$$\text{minimize } \tau\|x\|_1 + \frac{1}{2}\|Ax - b\|_2^2$$

$\tau > 0$, $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$, $m \leq n$ or $m > n$.

Optimality conditions

$$A^T(b - Ax^*) \in \tau\partial\|x^*\|_1.$$

Suppose $m > n$. Given τ, x^* and A (full column rank) we set:

$$b = \tau(A^T)^\dagger g + Ax^*, \quad g \in \partial\|x^*\|_1, \quad (A^T)^\dagger := A(A^TA)^{-1}$$

How to construct matrix A

- We need matrices for which $(A^\intercal A)^{-1}$ is trivial to compute.
- We fix the SVD form of A:

$$A = G_{left} \Sigma G_{right}^\intercal,$$

where G_{left} and G_{right} are products of Givens rotations:

$G(i, j, \theta) \in \mathbb{R}^{n \times n}$, which rotates plane i-j by an angle θ:

$$G(i,j,\theta) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & -s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix},$$

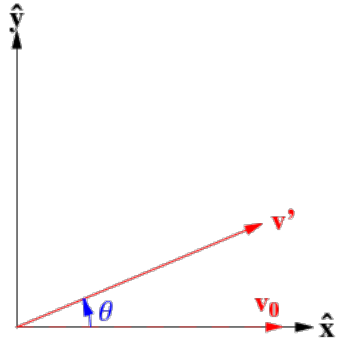
where $i, j \in \{1, 2, \cdots, n\}$, $c = \cos\theta$ and $s = \sin\theta$.

How to construct matrix A

- We need matrices for which $(A^\intercal A)^{-1}$ is trivial to compute.
- We fix the SVD form of A:

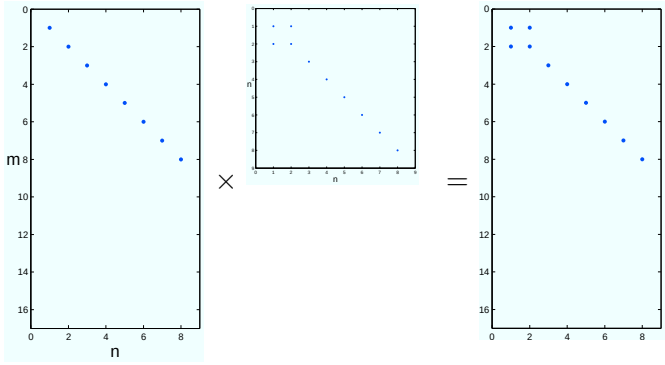
$$A = G_{left} \Sigma G_{right}^\intercal,$$

where G_{left} and G_{right} are products of Givens rotations:



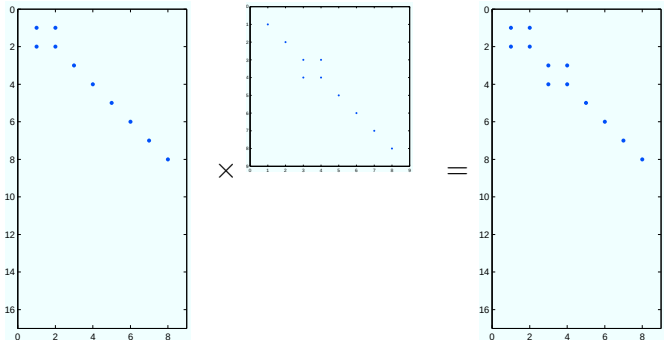
An example of matrix A

$$\Sigma G(1, 2, \theta)^{\mathsf{T}} = A_1$$



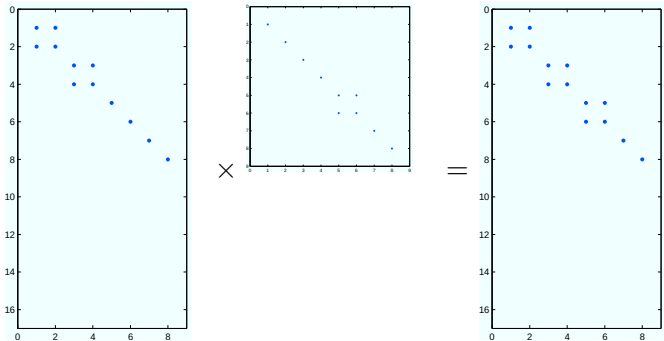
An example of matrix A

$$A_1 G(3, 4, \theta)^\mathsf{T} = A_2$$



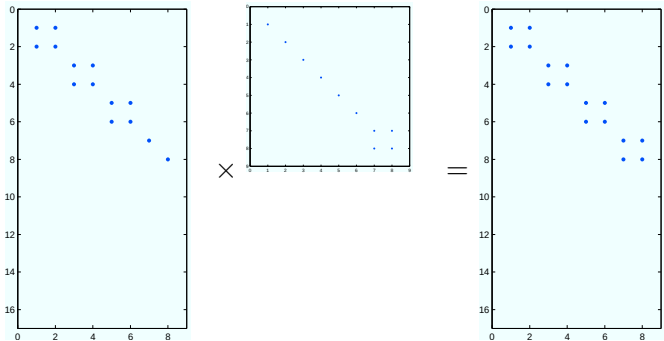
An example of matrix A

$$A_2 G(5, 6, \theta)^\mathsf{T} = A_3$$



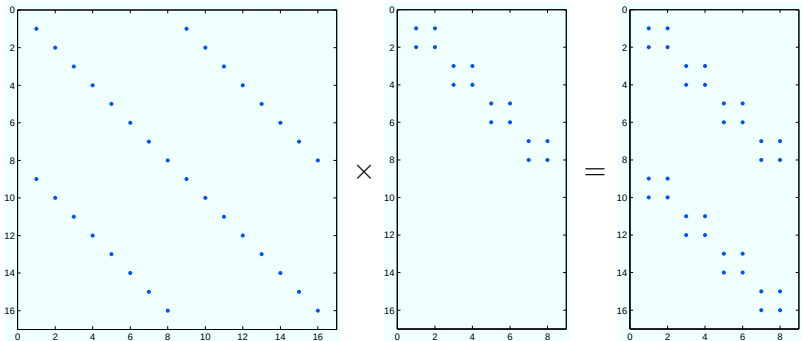
An example of matrix A

$$A_3 G(7, 8, \theta)^\mathsf{T} = A_4$$



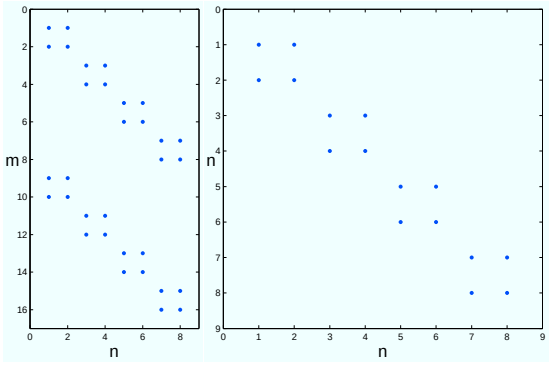
An example of matrix A

$$\tilde{G}_{5:12}(\theta)A_4 = A_{12}$$

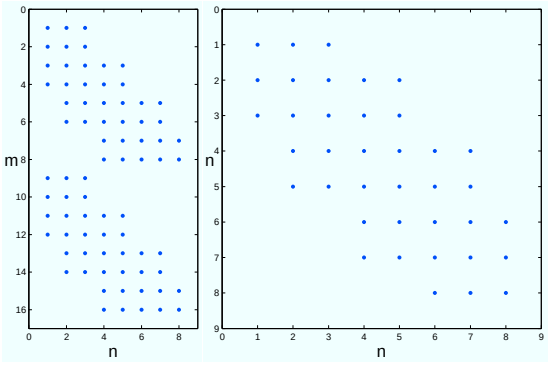


- $A_{12} = \tilde{G}_{5:12}(\theta)\Sigma(G(7,8,\theta)G(5,6,\theta)G(3,4,\theta)G(1,2,\theta))^{\mathsf{T}}$

Control of sparsity



A $A^\intercal A$



A $A^\intercal A$

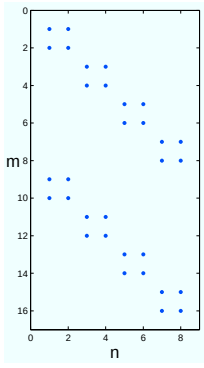
Mat-vecs

- $A_{12} = \tilde{G}_{5:12}(\theta)\Sigma(G(7,8,\theta)G(5,6,\theta)G(3,4,\theta)G(1,2,\theta))^\intercal$

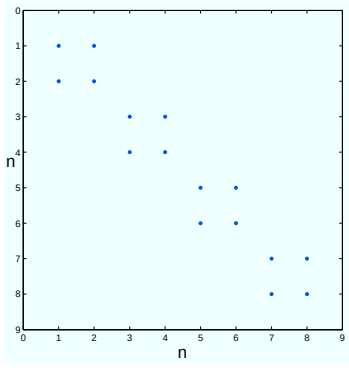
  $y := G(7,8,\theta)^\intercal x$
  $y := G(5,6,\theta)^\intercal y$
  $y := G(3,4,\theta)^\intercal y$
  $y := G(1,2,\theta)^\intercal y$
  $y := \Sigma y$
  $y := \tilde{G}_{5:12}(\theta) y$

- We can implement a for loop counter for pair of coordinates (i, j) in G
- Memory requirements: store y and two nonzeros for the rotation G

A trillion variable problem



A



$A^\intercal A$

- We do not exploit the separability of the problem.
- Columns of A are not nearly orthogonal, $\kappa(A^\intercal A) = 10^6$

A trillion variable problem

ARCHER

- 25th fastest supercomputer worldwide out of 500 supercomputers (based on TOP500 commercial supercomputers list)
- 118,080 cores, we used 65,536

n	processors	terabytes	seconds
$2^{36} \approx 68$ billion	4096	12.288	1970
2^{38}	16384	49.152	1990
$2^{40} \approx 1$ trillion	65536	196.608	2006

All problems have been solved to a relative error of order 10^{-4} using pdNCG

How to construct x^*: non-trivial example

Aim: construct sparse x^* that is difficult to recover by methods which do not capture curvature information.

Let $m \geq n$, $A = U\Sigma V^\top$, $\sigma_n \leq \sigma_{n-1} \leq \cdots \leq \sigma_1$, V_i: i^{th} column with singular value σ_i

- Curvature is related to: $A^\top A = V\Sigma^\top \Sigma V^\top$
- We can set: $x^* := \frac{1}{\sigma_1^2} V_1 + \frac{1}{\sigma_2^2} V_2 \ldots \frac{1}{\sigma_n^2} V_n$.

  Small coefficients for sing. vectors which correspond to large sing. values.
- Sparse version
  1: Choose number of nonzeros $s \leq n$
  2: Solve approximately

$$x^* := \begin{array}{ll} \underset{x \in \mathbb{R}^n}{\arg\min} & \|x - V(\Sigma^\top \Sigma)^{-1} 1_n\|^2 \\ \text{subject to:} & \|x\|_0 \leq s, \end{array}$$

Condition number for x^*

Let $\rho > 0$ and $\mathcal{I}_\rho := \{i \in [n] \mid \sigma_i^2 \geq \rho\}$, $V = [V_{\mathcal{I}_\rho}, V_{\mathcal{I}_\rho^c}]$ (without loss of gen.)

We define the projection $P_\rho = V_{\mathcal{I}_\rho} V_{\mathcal{I}_\rho}^\mathsf{T}$.

The condition number of x^* is defined as

$$\kappa_\rho(x^*) = \begin{cases} \frac{\|x^*\|_2}{\|P_\rho x^*\|_2}, & \text{if } P_\rho x^* \neq 0 \\ +\infty, & \text{otherwise} \end{cases}$$

Suppose $\sigma_n^2 \leq \rho \ll \sigma_1^2$.

- If $\kappa_\rho(x^*)$ is large, i.e., $\|P_\rho x^*\|_2$ is close to zero, then the majority of the mass of x^* is "hidden" in the span of the columns of $V_{\mathcal{I}_\rho^c}$.

Methods

- FISTA is an optimal first-order method. The Lipschitz constant is updated dynamically.
  A. Beck and M. Teboulle. SIAM J. Imaging Sci., 2(1):183-202,2009
- PCDM (Parallel Coordinate Descent Method) Parallel updates are performed asynchronously and the coordinates to be updated are chosen uniformly at random. Proximal parameter is set according to convergence theory.
  P. Richtárik and M. Takáč. Math Program. Ser. A, 2015, doi=10.1007/s10107-015-0901-6
- PSSgb (Projected Scaled Subgradient, Gafni-Bertsekas variant) is an active set second-order method.
  M. Schmidt. PhD Thesis, University British Columbia, 2010
- pdNCG (primal-dual Newton Conjugate Gradients) is a second-order method with a diagonal preconditioner for CG.
  K. Fountoulakis and J. Gondzio. Math. Program. Ser. A, 2015, doi=10.1007/s10107-015-0875-4

Setting

- FISTA, PSSgb, pdNCG are written in MATLAB, PCDM is C++ implementation.
- A 40 core computer is used.
- The default parameter tuning was used for all solvers.
- $m > n$, $n = 2^{22}$, $m = 2n$ (unless otherwise stated)
- We choose σ_n and σ_1, all other singular values are uniformly distributed in $[\sigma_n, \sigma_1]$.

Increasing condition number I

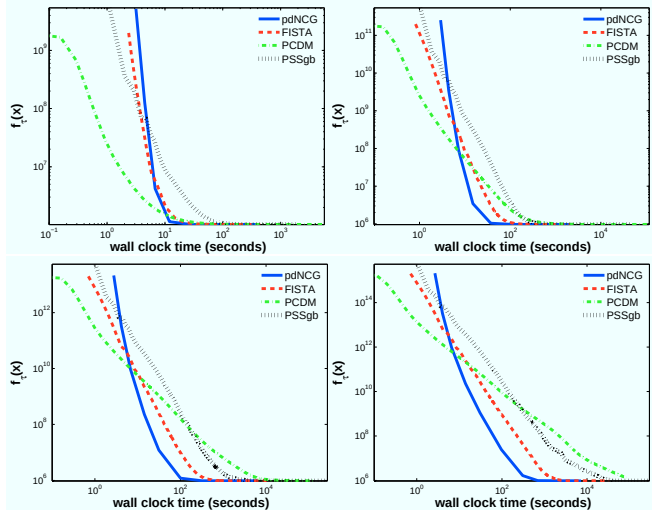
$\kappa(A^\intercal A) = 10^6,\ 10^8,\ 10^{10},\ 10^{12}$

$\sigma_n^2 = 10^{-2} \ll \sigma_1^2 = 10^4, 10^6, 10^8, 10^{10}$

$\kappa_{0.1}(x^*) \approx 1$

4 non-zeros per column of A

$$\frac{\text{mass of } diag(A^\intercal A)}{\text{mass of } A^\intercal A} \approx 0.69$$



Increasing condition number II

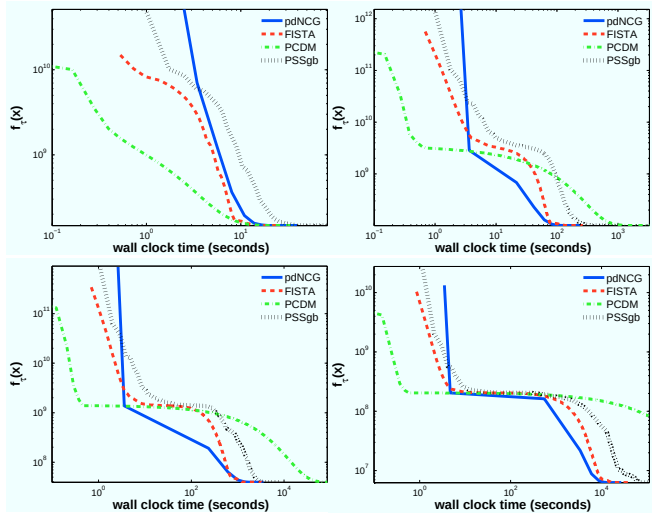
$\kappa(A^\intercal A) = 10^2,\ 10^4,\ 10^6,\ 10^8$

$\sigma_n^2 = 10^{-2} \ll \sigma_1^2 = 1, 10^2, 10^6, 10^8$

$\kappa_{0.1}(x^*) \approx 3,\ 3,\ 13,\ 6$

4 non-zeros per column of A

$$\frac{\text{mass of } diag(A^\intercal A)}{\text{mass of } A^\intercal A} \approx 0.69$$



Nearly orthogonal columns in A

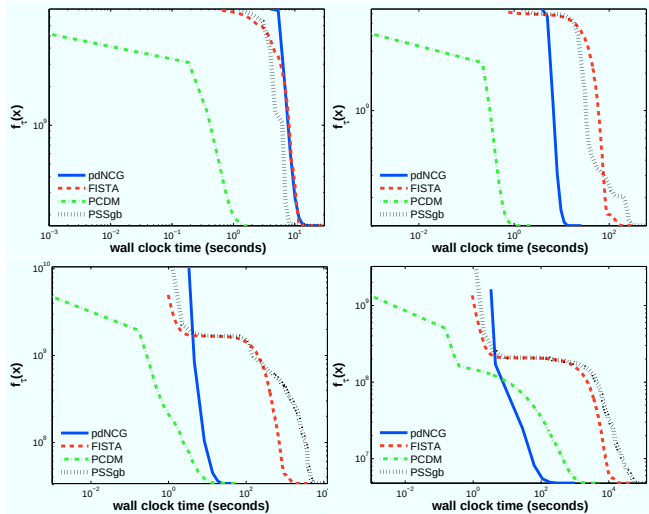
$\kappa(A^\top A) = 10^2,\ 10^4,\ 10^6,\ 10^8$

$\sigma_n^2 = 10^{-2} \ll \sigma_1^2 = 1, 10^2, 10^6, 10^8$

$\kappa_{0.1}(x^*) \approx 120, 120, 6, 6$

4 non-zeros per column of A

$$\frac{\text{mass of } diag(A^\top A)}{\text{mass of } A^\top A} \approx 0.99$$



Increasing density in A

$\kappa(A^\intercal A) = 10^4$

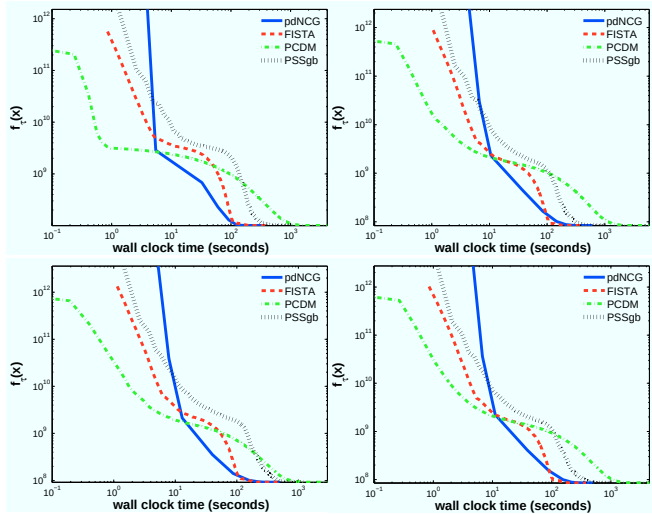
$\sigma_n^2 = 10^{-2} \ll \sigma_1^2 = 10^2$

$\kappa_{0.1}(x^*) \approx 3$

4,16,36,64 non-zeros per column of A

$$\frac{\text{mass of } diag(A^\intercal A)}{\text{mass of } A^\intercal A} \approx$$

0.69, 0.44, 0.39, 0.36



Increasing dimensions

$\kappa(A^\top A) = 10^4$

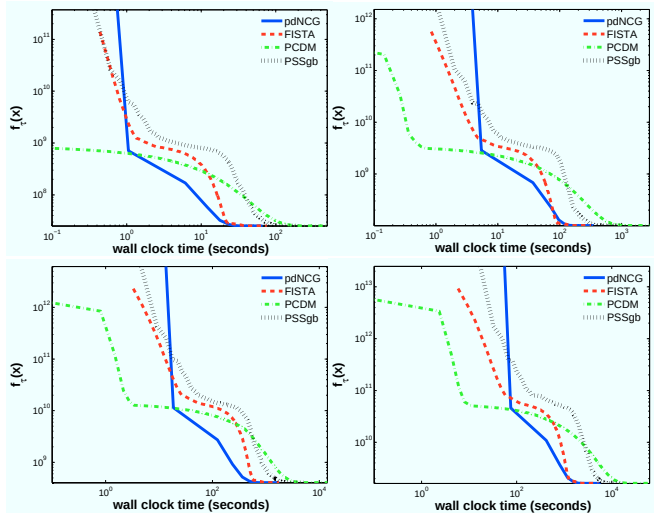
$\sigma_n^2 = 10^{-2} \ll \sigma_1^2 = 10^2$

$\kappa_{0.1}(x^*) \approx 3$

$n = 2^{20}, 2^{22}, 2^{24}, 2^{26}$

4 non-zeros per column of matrix A

$$\frac{\text{mass of } diag(A^\top A)}{\text{mass of } A^\top A} \approx 0.69$$



Many more experiments and details in the paper: Performance of first- and second-order methods for big data optimization. Technical report ERGO-15-005

Software: search on Google ERGO trillion

Thank you!

Methods

- FISTA is an optimal first-order method. At every iteration, FISTA minimizes:

$$Q_\tau(y;x) := \tau\|y\|_1 + \frac{1}{2}\|Ax-b\|^2 + (A^\intercal(Ax-b))^\intercal(y-x) + \frac{L}{2}\|y-x\|_2^2,$$

  Constant L us updated dynamically.

- PCDM (Parallel Coordinate Descent Method) Parallel updates are performed asynchronously and the coordinates to be updated are chosen uniformly at random. Let ϖ be the number of processors that are employed by PCDM. At every iteration PCDM minimizes

$$Q_\tau^i(y_i;x) := \tau|y_i| + \frac{1}{2}\|Ax-b\|^2 + (A_i^\intercal(Ax-b))(y_i-x_i) + \frac{\beta L_i}{2}(y_i-x_i)^2,$$

  $\forall i = 1, 2, \cdots, \varpi$, where A_i is the ith column of matrix A and $L_i = (A^\intercal A)_{ii}$ is the ith diagonal element of matrix $A^\intercal A$ and β is a positive constant.

Methods

PSSgb (Projected Scaled Subgradient, Gafni-Bertsekas variant) is a second-order method. Coordinates are separated into two sets, the working set $\mathcal{W}$ and the active set $\mathcal{A}$. The following local quadratic model is minimized:

$$Q_\tau(y; x) := f_\tau(x) + (\tilde{\nabla} f_\tau(x))^\intercal (y - x) + \frac{1}{2}(y - x)^\intercal H (y - x),$$

where $\tilde{\nabla} f_\tau(x)$ is a sub-gradient of f_τ at point x with the minimum Euclidean norm. Matrix H is defined as:

$$H = \begin{bmatrix} H_{\mathcal{W}} & 0 \\ 0 & H_{\mathcal{A}} \end{bmatrix},$$

where $H_{\mathcal{W}}$ is an L-BFGS Hessian approximation with respect to the coordinates $\mathcal{W}$ and $H_{\mathcal{A}}$ is a positive diagonal matrix. The diagonal matrix $H_{\mathcal{A}}$ is a scaled identity matrix, where the Shanno-Phua/Barzilai-Borwein scaling is used.

Methods

pdNCG (primal-dual Newton Conjugate Gradients) is a second-order method.

- Smooths the ℓ_1-norm
- At every iteration pdNCG constructs a convex smooth function Q_τ.
- The model Q is solved *inexactly* by reducing it to the linear system:

$$\nabla^2 f_\tau^\mu(x)(y - x) = -\nabla f_\tau^\mu(x),$$

  which is solved approximately using preconditioned Conjugate Gradients (PCG).

- A simple diagonal preconditioner is used for all experiments. The preconditioner is the inverse of the diagonal of matrix $\nabla^2 f_\tau^\mu(x)$.

Increasing τ

$\kappa(A^\top A) = 10^4$

$\sigma_n^2 = 10^{-2} \ll \sigma_1^2 = 10^2$

$\kappa_{0.1}(x^*) \approx 3$

4 non-zeros per column of matrix A

$\tau = 10^{-4}, 10^{-2}, 10^2, 10^4$

$$\frac{\text{mass of } diag(A^\top A)}{\text{mass of } A^\top A} \approx 0.69$$