Follow the flow: Proximal flow inspired multi-step methods

Yushen Huang YUSHEN.HUANG@STONYBROOK.EDU
Yifan Sun YIFAN.SUN@STONYBROOK.EDU
Stony Brook,NY,USA

Abstract

We investigate a family of Multi-Step Proximal Point Methods, the Backwards Differentiation Formulas, which are inspired by implicit linear discretization of gradient flow. The resulting methods are multi-step proximal point methods, with similar computational cost in each update as the proximal point method. We explore several optimization methods where applying an approximate multistep proximal points method results in improved convergence behavior. We argue that this is the result of the lowering of truncation error in approximating gradient flow.

1. Introduction

In this paper, we consider the following following optimization problem:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \; F(\mathbf{x}) = f(\mathbf{x}) + h(\mathbf{x}) \tag{1}$$

where $f(\mathbf{x})$ is a L- smooth function, $h(\mathbf{x})$ is a closed convex but not neccessary smooth function and $g(\mathbf{x})$ is bounded below. The problem with the following settings has been raised in many applications [3, 4, 6, 15, 16]. In this paper, we consider a family of multi-step proximal point updates. The algorithm is a generalization of proximal point method [8] where you use a linear combination of previous τ step as the point instead of just the last iterate.

$$\tilde{\mathbf{x}}_k = \sum_{i=1}^{\tau} \xi_i \mathbf{x}_{k-\tau+i}, \qquad \mathbf{x}_{k+1} = \mathcal{F}(\tilde{\mathbf{x}}_k) \tag{2}$$

Here, $\mathcal{F}$ is an approximate proximal point step. When $\tau = 1$, $\xi_1 = 1$ and (2) reduces to the "vanilla" approximate proximal point method, of which there are many works [1, 2, 8, 10]. In this paper, we investigate improvements garnered by higher order $\tau > 1$. Note that unlike nonlinear explicit discretization methods (like Runge-Kutta), there is very little overhead in increasing τ, as the averaging is done in an online manner.

However, there are two questions that could arise naturely. First, suppose that we are given τ, how do we choose ξ_i optimally? Second, can increasing τ always improve performance?

For the first question, we link the MulstiStep Proximal Methods to the discretization of Gradient Flow. Using dynamical systems to interpret optimization methods has garnered considerable interest [11, 12, 14, 17]. In order to optimize the peformance in practice, the coefficient should be chosed to maximize the order of truncation error which leads to so called BDF Scheme.

For the second question, the answer is under two aspects. The multistep proximal point methods do not speeed up the convergence rate because their convergence rate is the same as the gradient flow. However, the methods give significant better results in many optimization problems such as

proximal gradient with 1 norm over compressed sensing,proximal gradient with LSP penalty over compressed sensing,alternating projections over random linear subspaces and alternating minimization for matrix factorization.

2. Numerical Experiments

In this section, we empirically validate our proposed methods by considering several optimization problems: proximal gradient with L-1 norm, proximal gradient with LSP penalty, and alternating minimization for matrix factorization. For those experiments, we calculate the equation (2) based on the following two approaches:

Approach 1: The idea of the first approach is based on approximating

$$\text{prox}_{\alpha f+\alpha h}(\tilde{\mathbf{x}}^k) = \underset{\mathbf{x}}{\text{argmin}} \overbrace{\underbrace{\alpha f(\mathbf{x}) + \frac{1}{2}\|\mathbf{x} - \tilde{\mathbf{x}}^{(k,1)}\|_2^2}_{\text{smooth term}} + \underbrace{\alpha h(\mathbf{x})}_{\text{prox term}}}^{F_k(\mathbf{x})} \tag{3}$$

by the following.

Initialize $\tilde{\mathbf{x}}^{k,1} = \tilde{\mathbf{x}}^k$ based on left side of (2) and perform m iterations of the following update

$$\tilde{\mathbf{x}}^{(k,j+1)} \quad = \quad \text{prox}_{\beta\alpha h}(\tilde{\mathbf{x}}^{(k,j)} - \alpha\beta\nabla f(\tilde{\mathbf{x}}^{(k,j)}) - \beta(\tilde{\mathbf{x}}^{(k,j)} - \tilde{\mathbf{x}}^{(k,1)})).$$

Then update $\mathbf{x}^{k+1} = \tilde{\mathbf{x}}^{(k,m+1)}$.

Approach 2: The second approach is based on alternating minimization which is efficient in large-scale optimization [9, 13] and can be computed in parallel [5, 7]. The idea is approximating (3) by the following alternating descent.

Initialize $\tilde{\mathbf{x}}_1^{k,1}, \tilde{\mathbf{x}}_2^{k,2} = \tilde{\mathbf{x}}^k$ based on left side of (2) and perform m iterations of the following update:

$$\tilde{\mathbf{x}}_1^{k,j+1} = \arg\min_{\mathbf{x}_1} f(\mathbf{x}_1, \tilde{\mathbf{x}}_2^{k,j}) + h(\mathbf{x}_1, \tilde{\mathbf{x}}_2^{k,j}) + \frac{1}{2\alpha}\|\mathbf{x}_1 - \tilde{\mathbf{x}}_1^{k,j}\|^2$$
$$\tilde{\mathbf{x}}_2^{k,j+1} = \arg\min_{\mathbf{x}_2} f(\tilde{\mathbf{x}}_1^{k,j}, \mathbf{x}_2) + h(\tilde{\mathbf{x}}_1^{k,j}, \mathbf{x}_2) + \frac{1}{2\alpha}\|\mathbf{x}_2 - \tilde{\mathbf{x}}_2^{k,j}\|^2.$$

Then update $\mathbf{x}^{k+1} = \tilde{\mathbf{x}}^{(k,m+1)}$.

2.1. Proximal Gradient with ℓ_1 norm

The Proximal Gradient with ℓ_1 norm over compressed sensing problem is formulated as:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 + \lambda\|\mathbf{x}\|_1$$

where $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{b} \in \mathbb{R}^m$, with $m \ll n$ (underdetermined system) and the ℓ_1 norm is to make the solution of the system to be as sparse as possible. We choose $m = 100$ and $n = 500$

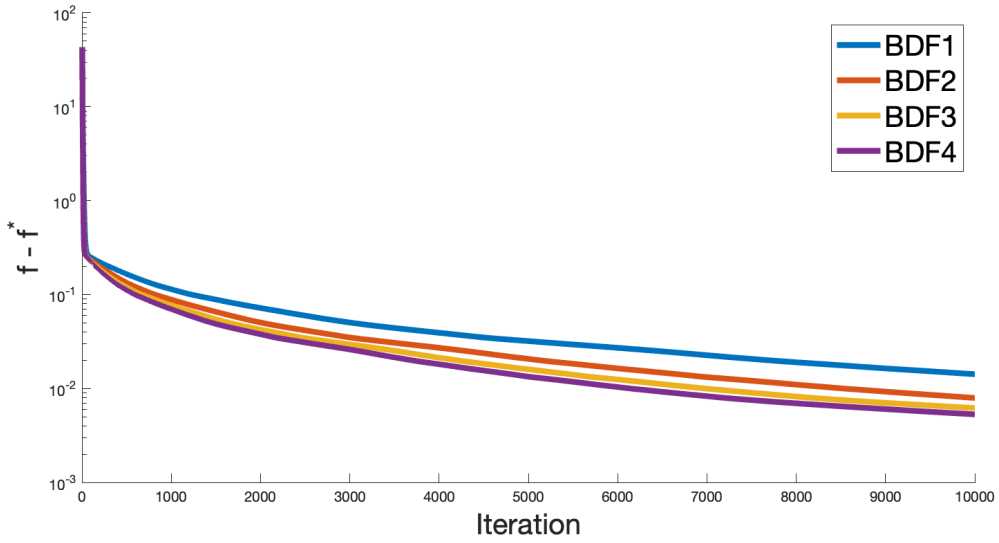

Figure 1: Comparsion of different BDF schemes for Proximal Gradient with $L-1$ norm over compressed sensing. We use **Approach** 1 to estimate the proximal mapping approximation with

and $\lambda = 0.1$. The entries of $\mathbf{A}_{i,j}, \mathbf{b}_i \sim \mathcal{N}(0,1)$. We use **approach** 1 to approximate the proximal mapping with the number of inner approximations to be $1, 5, 10$ and choose the maximum outer iteration to be 1000. The results are given in figure 1 In the above experiment, we can see that the higher order BDF scheme outperforms the lower BDF scheme.

2.2. Proximal Gradient with LSP penalty over compressed sensing

The Proximal Gradient with LSP penalty over compressed sensing is formulated as follows:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|^2 + h(\mathbf{x})$$

where $h(\mathbf{x}) = \sum_i \log(1+|x_i|/\lambda_i)$, $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{b} \in \mathbb{R}^m$. Similarly, $m \ll n$ since the system is underdertmine system. Compared to ℓ-1 norm settings, which gives each entry of $\mathbf{x}$ equal threshold to be sparse. The LSP penalty enjoys the benefit that you can have different thresholds for different entry.

For the experiment settings, we choose $m = 100$ and $n = 500$. We choose λ_i uniformly random from 0 to 1. For matrix $\mathbf{A}$, we choose each entry of $\mathbf{A}_{i,j}$ randomly by gaussian distribution with mean 0 and standard deviation 1. For matrix $\mathbf{b}$, we choose each entry $\mathbf{b}_i$ by Gaussian distribution with mean 0 and standard deviation 1. We use **Approach** 1 to approximate the proximal mapping with the number of inner approximations to be 1 and choose the maximum outer iteration to be 10000. The results are given in figure 3 . Similar to ℓ_1 norm setting, the higher-order BDF performs better than lower order BDF scheme.

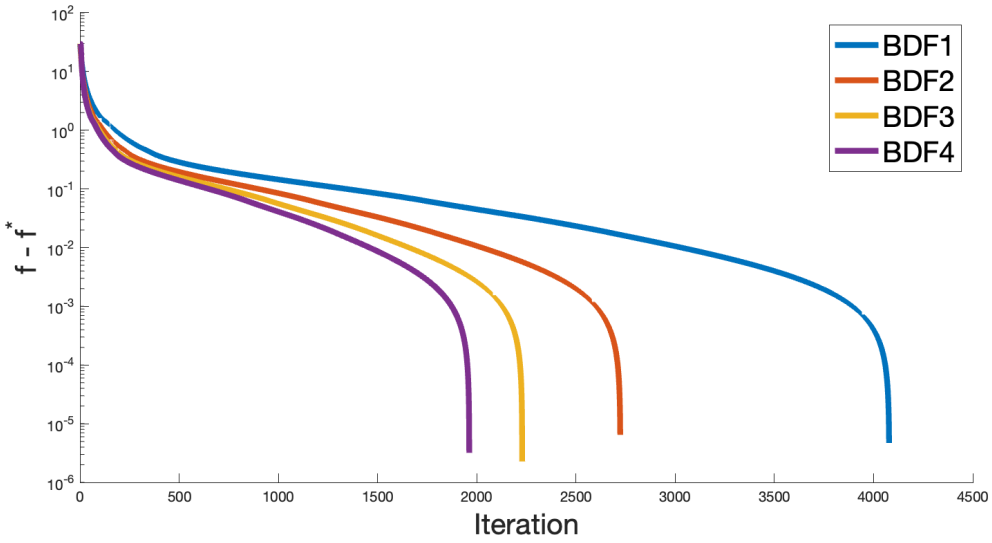

Figure 2: Comparsion of different BDF schemes for Proximal Gradient with LSP penalty over compressed sensing. The figure uses **Algorithm Approach** 1 to make the proximal mapping approximation. The figure uses 1 inner iteration

2.3. Alternating Minimization For Matrix Factorization

The Matrix Factorization problem can be formulated as the following:

$$\min_{\mathbf{U},\mathbf{V}} \frac{1}{2}\|\mathbf{U}\mathbf{V}^T - \mathbf{R}\|_F^2$$

The objective function is a classical ill-conditioned non-convex function. For the experiment settings, we choose $\mathbf{U}$ as a 100×50 matrix and $\mathbf{V}$ as a 50×100 matrix. We choose $\mathbf{R} = \mathbf{U}_{\text{true}}\mathbf{V}_{\text{true}}$ where $\mathbf{U}_{\text{true}}$ and $\mathbf{V}_{\text{true}}$ have the same dimension as $\mathbf{U}$ and $\mathbf{V}$. We use **Approach** 2 to approximate the proximal mapping with the number of inner approximations 1 and choose the maximum outer iteration to be 1000.

3. Conclusion and Future Work

The goal of this work is to investigate the use of approximate implicit discretizations of (**Proximal Flow**) in badly conditioned non-smooth problem settings. In this work, we figure out that the higher-order approximate implicit discretization helps in many optimization problems. However, it is worth pointing out that it is also important to find an efficient manner of choosing the approximate methods. In our work, we provide two approaches that approximate the implicit updates, and both work well in practice. For future work, it will be interesting to investigate applying higher-order discretization to other flows, such as rescaled proximal or accelerated proximal flow.

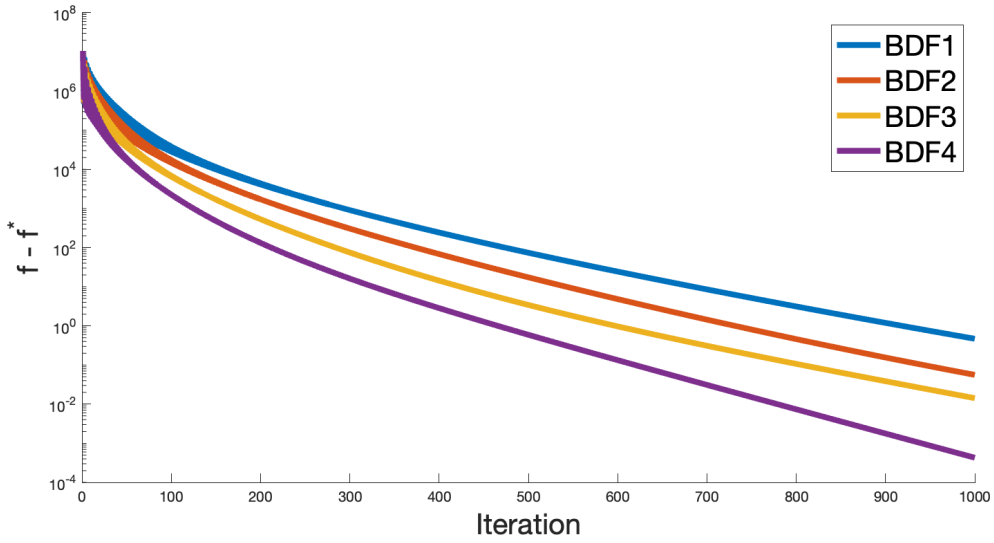

Figure 3: Comparsion of different BDF schemes for matrix factorization. The figure uses using **Approach** 2 to do the proximal mapping approximation. The top figure has an inner iteration 1.

References

[1] Hilal Asi and John C Duchi. Stochastic (approximate) proximal point methods: Convergence, optimality, and adaptivity. *SIAM Journal on Optimization*, 29(3):2257–2290, 2019.

[2] Hilal Asi, Karan Chadha, Gary Cheng, and John C Duchi. Minibatch stochastic approximate proximal point methods. *Advances in neural information processing systems*, 33:21958–21968, 2020.

[3] Emmanuel Candes and Benjamin Recht. Exact matrix completion via convex optimization. *Communications of the ACM*, 55(6):111–119, 2012.

[4] Theodoros Evgeniou, Charles A Micchelli, Massimiliano Pontil, and John Shawe-Taylor. Learning multiple tasks with kernel methods. *Journal of machine learning research*, 6(4), 2005.

[5] Olivier Fercoq and Peter Richtárik. Accelerated, parallel, and proximal coordinate descent. *SIAM Journal on Optimization*, 25(4):1997–2023, 2015.

[6] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[7] Ji Liu, Steve Wright, Christopher Ré, Victor Bittorf, and Srikrishna Sridhar. An asynchronous parallel stochastic coordinate descent algorithm. In *International Conference on Machine Learning*, pages 469–477. PMLR, 2014.

[8] Jean-Jacques Moreau. Proximité et dualité dans un espace hilbertien. *Bulletin de la Société mathématique de France*, 93:273–299, 1965.

[9] Yu Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

[10] Yurii Nesterov. Inexact high-order proximal-point methods with auxiliary search procedure. *SIAM Journal on Optimization*, 31(4):2807–2828, 2021.

[11] Orlando Romero and Mouhacine Benosman. Finite-time convergence in continuous-time optimization. In *International Conference on Machine Learning*, pages 8200–8209. PMLR, 2020.

[12] Bin Shi, Simon S Du, Weijie Su, and Michael I Jordan. Acceleration via symplectic discretization of high-resolution differential equations. *Advances in Neural Information Processing Systems*, 32, 2019.

[13] Hao-Jun Michael Shi, Shenyinying Tu, Yangyang Xu, and Wotao Yin. A primer on coordinate descent algorithms. *arXiv preprint arXiv:1610.00040*, 2016.

[14] Weijie Su, Stephen Boyd, and Emmanuel J Candes. A differential equation for modeling nesterov's accelerated gradient method: theory and insights. *arXiv preprint arXiv:1503.01243*, 2015.

[15] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, 1996.

[16] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68(1):49–67, 2006.

[17] Jingzhao Zhang, Aryan Mokhtari, Suvrit Sra, and Ali Jadbabaie. Direct runge-kutta discretization achieves acceleration. *Advances in neural information processing systems*, 31, 2018.