Primal-Dual Splitting: Recent Improvements and Variants

[1]**Thomas Pock and** [2]**Antonin Chambolle**

[1]Institute for Computer Graphics and Vision, TU Graz, Austria
[2]CMAP & CNRS École Polytechnique, France

SIAM Imaging 2012

The proximal point algorithm

- Consider the problem of finding a x in a Hilbert space such that $T(x) \ni 0$, where T is a maximal monotone operator.
- The proximal point algorithm: [Martinet '70], [Rockafellar '76]
- Choose a x^0 and generate the sequence $\{x^n\}$ for each $n \geq 0$ via

$$0 \in T(x^{n+1}) + \lambda_n^{-1}(x^{n+1} - x^n)$$

The proximal point algorithm

- Consider the problem of finding a x in a Hilbert space such that $T(x) \ni 0$, where T is a maximal monotone operator.
- The proximal point algorithm: [Martinet '70], [Rockafellar '76]
- Choose a x^0 and generate the sequence $\{x^n\}$ for each $n \geq 0$ via

$$0 \in T(x^{n+1}) + \lambda_n^{-1}(x^{n+1} - x^n)$$

- The proximal point algorithm can be written as [Minty '62]

$$x^{n+1} = (I + \lambda_n T)^{-1}(x^n)\,,$$

where $(I + \lambda_n T)^{-1}$ is the resolvent operator.

The proximal point algorithm

- Consider the problem of finding a x in a Hilbert space such that $T(x) \ni 0$, where T is a maximal monotone operator.
- The proximal point algorithm: [Martinet '70], [Rockafellar '76]
- Choose a x^0 and generate the sequence $\{x^n\}$ for each $n \geq 0$ via

$$0 \in T(x^{n+1}) + \lambda_n^{-1}(x^{n+1} - x^n)$$

- The proximal point algorithm can be written as [Minty '62]

$$x^{n+1} = (I + \lambda_n T)^{-1}(x^n)\,,$$

where $(I + \lambda_n T)^{-1}$ is the resolvent operator.

- Let $F(x)$ be a proper, l.s.c. convex function and $T(x) = \partial F(x)$, then the algorithm reads

$$x^{n+1} = \arg\min_x F(x) + \frac{1}{2\lambda_n}\|x - x^n\|_2^2$$

The proximal point is $\mathcal{O}(1/n)$

- Convergence of the proximal point algorithm in case of summable errors have been proven [Rockafellar '76] in and in case of non-summable errors recently in [Zaslavski '11]

The proximal point is $\mathcal{O}(1/n)$

- Convergence of the proximal point algorithm in case of summable errors have been proven [Rockafellar '76] in and in case of non-summable errors recently in [Zaslavski '11]
- Moreover, if $\lambda_{n+1} \leq \lambda_n$, one has [Dong '12]

$$\|x^{n+1} - (I + \lambda_{n+1} T)^{-1}(x^{n+1})\|_2^2 \leq \|x^n - (I + \lambda_n T)^{-1}(x^n)\|_2^2$$

from which follows that

$$\|x^n - (I + \lambda_n T)^{-1}(x^n)\|_2^2 \leq \mathcal{O}(1/n)$$

The proximal point is $\mathcal{O}(1/n)$

- Convergence of the proximal point algorithm in case of summable errors have been proven [Rockafellar '76] in and in case of non-summable errors recently in [Zaslavski '11]
- Moreover, if $\lambda_{n+1} \leq \lambda_n$, one has [Dong '12]

$$\|x^{n+1} - (I + \lambda_{n+1} T)^{-1}(x^{n+1})\|_2^2 \leq \|x^n - (I + \lambda_n T)^{-1}(x^n)\|_2^2$$

  from which follows that

$$\|x^n - (I + \lambda_n T)^{-1}(x^n)\|_2^2 \leq \mathcal{O}(1/n)$$

- Unfortunately, computing the resolvent is often as difficult as solving the problem
- However, in case $T(x)$ can be written as sum of two operators each with simple to compute resolvents, things will change...

A class of problems

Let us consider the following class of structured convex optimization problems

$$\min_{x \in X} F(Kx) + G(x) \ ,$$

- $K : X \to Y$ is a linear and continuous operator from a Hilbert space X to a Hilbert space Y and F, G are convex, proper, l.s.c. functions.
- Note that the subdifferential of this problem is the sum of two operators

$$T(x) = K^* \partial F(Kx) + \partial G(x)$$

A class of problems

Let us consider the following class of structured convex optimization problems

$$\min_{x \in X} F(Kx) + G(x) \ ,$$

- $K : X \to Y$ is a linear and continuous operator from a Hilbert space X to a Hilbert space Y and F, G are convex, proper, l.s.c. functions.
- Note that the subdifferential of this problem is the sum of two operators

$$T(x) = K^* \partial F(Kx) + \partial G(x)$$

- **Main assumption:** F, G are "simple" in the sense that they have easy to compute resolvent operators:

$$(I + \partial F)^{-1}(\hat{p}) = \arg\min_p \frac{\|p - \hat{p}\|^2}{2\lambda} + F(p)$$

$$(I + \partial G)^{-1}(\hat{x}) = \arg\min_x \frac{\|x - \hat{x}\|^2}{2\lambda} + G(x)$$

A class of problems

Let us consider the following class of structured convex optimization problems

$$\min_{x \in X} F(Kx) + G(x) \ ,$$

- $K : X \to Y$ is a linear and continuous operator from a Hilbert space X to a Hilbert space Y and F, G are convex, proper, l.s.c. functions.
- Note that the subdifferential of this problem is the sum of two operators

$$T(x) = K^* \partial F(Kx) + \partial G(x)$$

- **Main assumption:** F, G are "simple" in the sense that they have easy to compute resolvent operators:

$$(I + \partial F)^{-1}(\hat{p}) = \arg\min_p \frac{\|p - \hat{p}\|^2}{2\lambda} + F(p)$$

$$(I + \partial G)^{-1}(\hat{x}) = \arg\min_x \frac{\|x - \hat{x}\|^2}{2\lambda} + G(x)$$

- It turns out that many standard problems can be cast in this framework.

Some examples

- The ROF model

$$\min_u \|\nabla u\|_{2,1} + \frac{\lambda}{2}\|u - f\|_2^2 \,,$$

Some examples

- The ROF model

$$\min_u \|\nabla u\|_{2,1} + \frac{\lambda}{2}\|u - f\|_2^2 \,,$$

- Basis pursuit problem (LASSO)

$$\min_x \|x\|_1 + \frac{\lambda}{2}\|Ax - b\|_2^2$$

Some examples

- The ROF model

$$\min_u \|\nabla u\|_{2,1} + \frac{\lambda}{2}\|u - f\|_2^2 \,,$$

- Basis pursuit problem (LASSO)

$$\min_x \|x\|_1 + \frac{\lambda}{2}\|Ax - b\|_2^2$$

- Linear support vector machine

$$\min_{w,b} \frac{\lambda}{2}\|w\|_2^2 + \sum_{i=1}^{n} \max\left(0, 1 - y_i\left(\langle w, x_i\rangle + b\right)\right)$$

Some examples

- The ROF model

$$\min_u \|\nabla u\|_{2,1} + \frac{\lambda}{2}\|u - f\|_2^2 \,,$$

- Basis pursuit problem (LASSO)

$$\min_x \|x\|_1 + \frac{\lambda}{2}\|Ax - b\|_2^2$$

- Linear support vector machine

$$\min_{w,b} \frac{\lambda}{2}\|w\|_2^2 + \sum_{i=1}^{n} \max\left(0, 1 - y_i\left(\langle w, x_i\rangle + b\right)\right)$$

- General linear programming problems

$$\min_x \langle c, x\rangle \,, \text{ s.t. } \begin{cases} Ax &= b \\ x &\geq 0 \end{cases}$$

Primal, dual, primal-dual

The real power of convex optimization comes through duality

Recall the convex conjugate:

$$F^*(y) = \max_{x \in X} \langle x, y \rangle - F(x) \ ,$$

we can transform our initial problem

$$\min_{x \in X} F(Kx) + G(x) \quad \text{(Primal)}$$

Primal, dual, primal-dual

The real power of convex optimization comes through duality

Recall the convex conjugate:

$$F^*(y) = \max_{x \in X} \langle x, y \rangle - F(x) \ ,$$

we can transform our initial problem

$$\min_{x \in X} F(Kx) + G(x) \quad \text{(Primal)}$$

$$\min_{x \in X} \max_{y \in Y} \langle Kx, y \rangle + G(x) - F^*(y) \quad \text{(Primal-Dual)}$$

Primal, dual, primal-dual

The real power of convex optimization comes through duality

Recall the convex conjugate:

$$F^*(y) = \max_{x \in X} \langle x, y \rangle - F(x) \,,$$

we can transform our initial problem

$$\min_{x \in X} F(Kx) + G(x) \quad \text{(Primal)}$$

$$\min_{x \in X} \max_{y \in Y} \langle Kx, y \rangle + G(x) - F^*(y) \quad \text{(Primal-Dual)}$$

$$\max_{y \in Y} -(F^*(y) + G^*(-K^* y)) \quad \text{(Dual)}$$

There is a primal-dual gap

$$\mathcal{G}(x, y) = F(Kx) + G(x) + (F^*(y) + G^*(-K^* y))$$

that vanishes if and only if (x, y) is optimal

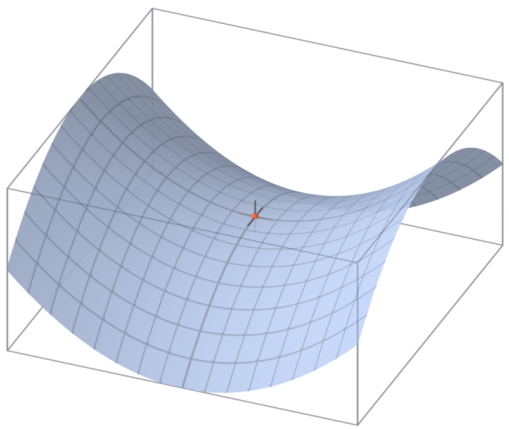
Optimality conditions

We focus on the primal-dual formulation:

$$\min_{x \in X} \max_{y \in Y} \langle Kx, y \rangle + G(x) - F^*(y)$$

We assume, there exists a saddle-point $(\hat{x}, \hat{y}) \in X \times Y$ which satisfies the Euler-Lagrange equations

$$\begin{cases} K\hat{x} - \partial F^*(\hat{y}) \ni 0 \\ K^*\hat{y} + \partial G(\hat{x}) \ni 0 \end{cases}$$



Example for a saddle-point of a convex-concave function

Known algorithms

Many standard algorithms to solve the considered class of problems:

- Classical Arrow-Hurwicz method [Arrow-Hurwicz, '58]
- Extragradient-methods [Korpelevich '76, Popov '80]
- Douglas-Rachford Splitting [Mercier-Lions '79]
- Alternating direction method of multipliers Globinski, Marroco '75] [Gabay, Mercier '76] [Eckstein, Bertsekas '89], [Goldstein, Osher '09]
- Many more algorithms for special cases: [Nesterov '03], [Daubechies, Defrise, De Mol '04], [Combettes, Pesquet, '08], [Beck, Teboulle '09], [Raguet, Fadili, Peyré, '11]

A first-order primal-dual algorithm

Proposed in a series of papers: [P., Cremers, Bischof, Chambolle, '09], [Chambolle, P., '10], [P., Chambolle, '11]

- Initialization: Choose $\mathrm{T}, \Sigma \in S_{++}$, $\theta \in [0,1]$, $(x^0, y^0) \in X \times Y$.
- Iterations ($n \geq 0$): Update x^n, y^n as follows:

$$\begin{cases} x^{n+1} &= (I + \mathrm{T}\partial G)^{-1}(x^n - \mathrm{T}K^* y^n) \\ y^{n+1} &= (I + \Sigma\partial F^*)^{-1}(y^n + \Sigma K(x^{n+1} + \theta(x^{n+1} - x^n))) \end{cases}$$

- Alternates gradient descend in x and gradient ascend in y
- Linear extrapolation of iterates of x in the y step
- T, Σ can be seen as preconditioning matrices

A first-order primal-dual algorithm

Proposed in a series of papers: [P., Cremers, Bischof, Chambolle, '09], [Chambolle, P., '10], [P., Chambolle, '11]

- Initialization: Choose $\mathrm{T}, \Sigma \in S_{++}$, $\theta \in [0,1]$, $(x^0, y^0) \in X \times Y$.
- Iterations ($n \geq 0$): Update x^n, y^n as follows:

$$\begin{cases} x^{n+1} &= (I + \mathrm{T}\partial G)^{-1}(x^n - \mathrm{T}K^* y^n) \\ y^{n+1} &= (I + \Sigma\partial F^*)^{-1}(y^n + \Sigma K(x^{n+1} + \theta(x^{n+1} - x^n))) \end{cases}$$

- Alternates gradient descend in x and gradient ascend in y
- Linear extrapolation of iterates of x in the y step
- T, Σ can be seen as preconditioning matrices

- Can be derived from a pre-conditioned ADMM algorithm
- Can be seen as a relaxed Arrow-Hurwicz scheme
- Can be seen as an approximate extragradient scheme

Relations to the proximal point algorithm

- Recall the proximal point algorithm

$$0 \in T(x^{n+1}) + \lambda_n^{-1}(x^{n+1} - x^n), \; n \geq 0$$

Relations to the proximal point algorithm

- Recall the proximal point algorithm

$$0 \in T(x^{n+1}) + \lambda_n^{-1}(x^{n+1} - x^n), \; n \geq 0$$

- The iterations of the primal-dual algorithm can be rewritten as

$$0 \in \begin{pmatrix} \partial G(x^{n+1}) + K^* y^{n+1} \\ \partial F^*(y^{n+1}) - K x^{n+1} \end{pmatrix} + M \begin{pmatrix} x^{n+1} - x^n \\ y^{n+1} - y^n \end{pmatrix}, \; n \geq 0$$

$$M = \begin{bmatrix} \mathrm{T}^{-1} & -K^* \\ -\theta K & \Sigma^{-1} \end{bmatrix}$$

Relations to the proximal point algorithm

- Recall the proximal point algorithm

$$0 \in T(x^{n+1}) + \lambda_n^{-1}(x^{n+1} - x^n), \; n \geq 0$$

- The iterations of the primal-dual algorithm can be rewritten as

$$0 \in \begin{pmatrix} \partial G(x^{n+1}) + K^* y^{n+1} \\ \partial F^*(y^{n+1}) - K x^{n+1} \end{pmatrix} + M \begin{pmatrix} x^{n+1} - x^n \\ y^{n+1} - y^n \end{pmatrix}, \; n \geq 0$$

$$M = \begin{bmatrix} \mathrm{T}^{-1} & -K^* \\ -\theta K & \Sigma^{-1} \end{bmatrix}$$

- This is exactly the proximal point algorithm, but with a norm in M.

Relations to the proximal point algorithm

- Recall the proximal point algorithm

$$0 \in T(x^{n+1}) + \lambda_n^{-1}(x^{n+1} - x^n), \; n \geq 0$$

- The iterations of the primal-dual algorithm can be rewritten as

$$0 \in \begin{pmatrix} \partial G(x^{n+1}) + K^* y^{n+1} \\ \partial F^*(y^{n+1}) - K x^{n+1} \end{pmatrix} + M \begin{pmatrix} x^{n+1} - x^n \\ y^{n+1} - y^n \end{pmatrix}, \; n \geq 0$$

$$M = \begin{bmatrix} \mathrm{T}^{-1} & -K^* \\ -\theta K & \Sigma^{-1} \end{bmatrix}$$

- This is exactly the proximal point algorithm, but with a norm in M.
- Convergence of the proximal point algorithm is ensured if M is symmetric and positive definite

Convergence

Theorem

Let $\theta = 1$, T and Σ symmetric positive definite maps satisfying

$$\|\Sigma^{\frac{1}{2}} K \mathrm{T}^{\frac{1}{2}}\|^2 < 1 \,,$$

then the primal-dual algorithm converges to a saddle-point.

Convergence

Theorem

Let $\theta = 1$, T and Σ symmetric positive definite maps satisfying

$$\|\Sigma^{\frac{1}{2}} K \mathrm{T}^{\frac{1}{2}}\|^2 < 1 \,,$$

then the primal-dual algorithm converges to a saddle-point.

The algorithm gives different convergence rates on different problem classes [Chambolle, P., '10]

- F^* and G nonsmooth: $O(1/n)$

Convergence

Theorem

Let $\theta = 1$, T and Σ symmetric positive definite maps satisfying

$$\|\Sigma^{\frac{1}{2}} K \mathrm{T}^{\frac{1}{2}}\|^2 < 1 \,,$$

then the primal-dual algorithm converges to a saddle-point.

The algorithm gives different convergence rates on different problem classes [Chambolle, P., '10]

- F^* and G nonsmooth: $O(1/n)$
- F^* or G uniformly convex: $O(1/n^2)$

Convergence

Theorem

Let $\theta = 1$, T and Σ symmetric positive definite maps satisfying

$$\|\Sigma^{\frac{1}{2}} K \mathrm{T}^{\frac{1}{2}}\|^2 < 1 \,,$$

then the primal-dual algorithm converges to a saddle-point.

The algorithm gives different convergence rates on different problem classes [Chambolle, P., '10]

- F^* and G nonsmooth: $O(1/n)$
- F^* or G uniformly convex: $O(1/n^2)$
- F^* and G uniformly convex: $O(\omega^n)$, $\omega < 1$

Convergence

Theorem

Let $\theta = 1$, T and Σ symmetric positive definite maps satisfying

$$\|\Sigma^{\frac{1}{2}} K \mathrm{T}^{\frac{1}{2}}\|^2 < 1 \,,$$

then the primal-dual algorithm converges to a saddle-point.

The algorithm gives different convergence rates on different problem classes [Chambolle, P., '10]

- F^* and G nonsmooth: $O(1/n)$
- F^* or G uniformly convex: $O(1/n^2)$
- F^* and G uniformly convex: $O(\omega^n)$, $\omega < 1$

- Coincides with so far best known rates of first-order methods

Extensions

- Since the primal-dual algorithm is in principle a proximal point algorithm, we can perform an additional overrelaxation [Gol'shtein, Tret'yakov '79]

$$\begin{cases} x^{n+\frac{1}{2}} = (I + \mathrm{T}\partial G)^{-1}\left(x^n - \mathrm{T}\,\mathcal{K}^T y^n\right) \\ y^{n+\frac{1}{2}} = (I + \Sigma\partial F^*)^{-1}\left(y^n + \Sigma\,\mathcal{K}(2x^{n+\frac{1}{2}} - x^n)\right) \\ (x^{n+1}, y^{n+1}) = (x^{n+\frac{1}{2}}, y^{n+\frac{1}{2}}) + \gamma(x^{n+\frac{1}{2}} - x^n, y^{n+\frac{1}{2}} - y^n)\,. \end{cases}$$

  where $\gamma \in [0, 1[$
- Speeds up the convergence in many cases.

Extensions

- Since the primal-dual algorithm is in principle a proximal point algorithm, we can perform an additional overrelaxation [Gol'shtein, Tret'yakov '79]

$$\begin{cases} x^{n+\frac{1}{2}} = (I + \mathrm{T}\partial G)^{-1}\left(x^n - \mathrm{T}\,\mathcal{K}^T y^n\right) \\ y^{n+\frac{1}{2}} = (I + \Sigma\partial F^*)^{-1}\left(y^n + \Sigma\,\mathcal{K}(2x^{n+\frac{1}{2}} - x^n)\right) \\ (x^{n+1}, y^{n+1}) = (x^{n+\frac{1}{2}}, y^{n+\frac{1}{2}}) + \gamma(x^{n+\frac{1}{2}} - x^n, y^{n+\frac{1}{2}} - y^n)\,. \end{cases}$$

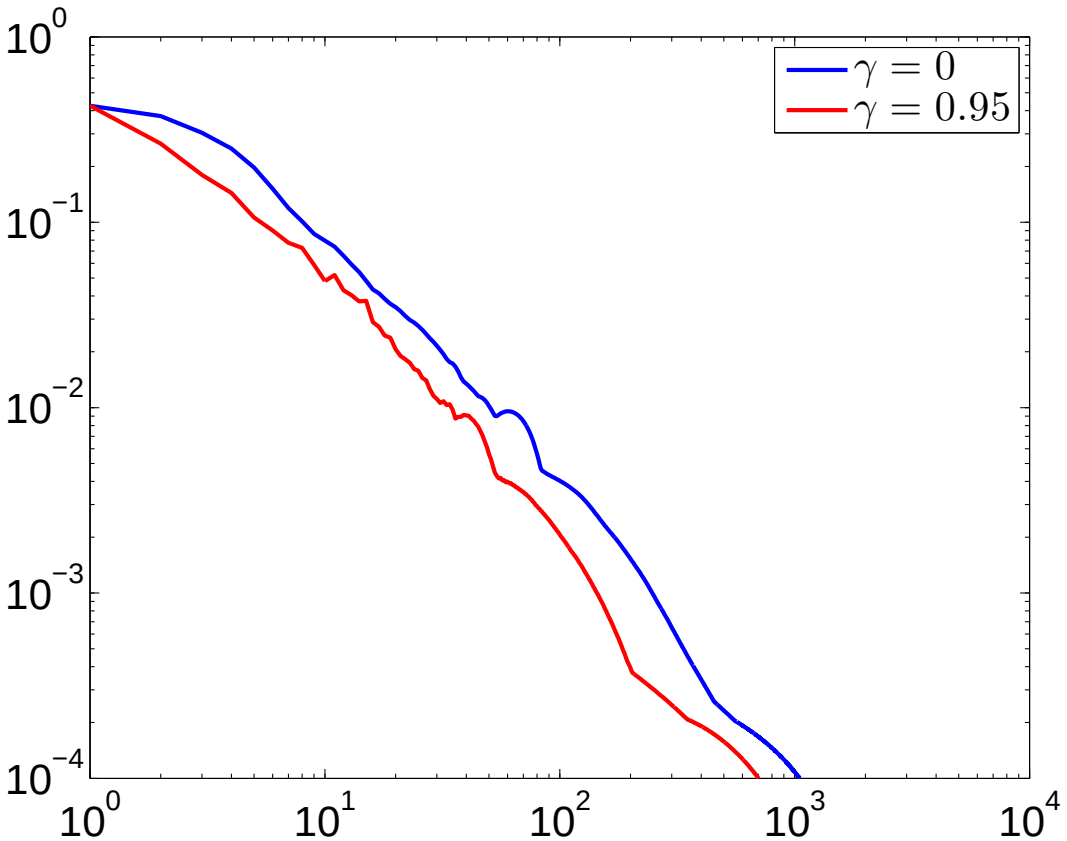
  where $\gamma \in [0, 1[$
- Speeds up the convergence in many cases.
- The algorithm has recently been generalized by Laurent Condat in case the function $G(x)$ can be written as

$$G(x) = G_1(x) + G_2(x)$$

  where $G_1(x)$ is convex, proper, l.s.c. but $G_2(x)$ is differentiable with Lipschitz continuous gradient
- Has advantages in case there is some smoothness in the problem.

Effect of the overrelaxation

Application to the ROF model using the accelerated $\mathcal{O}(1/n^2)$ algorithm



- No overrelaxation: $\gamma = 0$, 1050 iterations
- Overrelaxation: $\gamma = 0.95$, 700 iterations

α-preconditioning

- It is important to choose the preconditioner such that the prox-operators are still easy to compute
- Restrict the preconditioning matrices to diagonal matrices

α-preconditioning

- It is important to choose the preconditioner such that the prox-operators are still easy to compute
- Restrict the preconditioning matrices to diagonal matrices

Lemma

Let $\mathrm{T} = \mathrm{diag}(\tau_1, \ldots \tau_n)$ *and* $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_m)$.

$$\tau_j = \frac{1}{\sum_{i=1}^{m} |K_{i,j}|^{2-\alpha}}, \quad \sigma_i = \frac{1}{\sum_{j=1}^{n} |K_{i,j}|^{\alpha}}$$

then for any $\alpha \in [0, 2]$

$$\|\Sigma^{\frac{1}{2}} K \mathrm{T}^{\frac{1}{2}}\|^2 = \sup_{x \in X,\, x \neq 0} \frac{\|\Sigma^{\frac{1}{2}} K \mathrm{T}^{\frac{1}{2}} x\|^2}{\|x\|^2} \leq 1 \,.$$

α-preconditioning

- It is important to choose the preconditioner such that the prox-operators are still easy to compute
- Restrict the preconditioning matrices to diagonal matrices

Lemma

Let $\mathrm{T} = \mathrm{diag}(\tau_1, \ldots \tau_n)$ *and* $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_m)$.

$$\tau_j = \frac{1}{\sum_{i=1}^{m} |K_{i,j}|^{2-\alpha}}, \quad \sigma_i = \frac{1}{\sum_{j=1}^{n} |K_{i,j}|^{\alpha}}$$

then for any $\alpha \in [0, 2]$

$$\|\Sigma^{\frac{1}{2}} K \mathrm{T}^{\frac{1}{2}}\|^2 = \sup_{x \in X,\, x \neq 0} \frac{\|\Sigma^{\frac{1}{2}} K \mathrm{T}^{\frac{1}{2}} x\|^2}{\|x\|^2} \leq 1 \,.$$

- The parameter α can be used to vary between pure primal ($\alpha = 0$) and pure dual ($\alpha = 2$) preconditioning

α-preconditioning

- It is important to choose the preconditioner such that the prox-operators are still easy to compute
- Restrict the preconditioning matrices to diagonal matrices

Lemma

Let $\mathrm{T} = \mathrm{diag}(\tau_1, \ldots \tau_n)$ *and* $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_m)$.

$$\tau_j = \frac{1}{\sum_{i=1}^{m} |K_{i,j}|^{2-\alpha}}, \quad \sigma_i = \frac{1}{\sum_{j=1}^{n} |K_{i,j}|^{\alpha}}$$

then for any $\alpha \in [0, 2]$

$$\|\Sigma^{\frac{1}{2}} K \mathrm{T}^{\frac{1}{2}}\|^2 = \sup_{x \in X,\, x \neq 0} \frac{\|\Sigma^{\frac{1}{2}} K \mathrm{T}^{\frac{1}{2}} x\|^2}{\|x\|^2} \leq 1 \,.$$

- The parameter α can be used to vary between pure primal ($\alpha = 0$) and pure dual ($\alpha = 2$) preconditioning
- It turns out that for $\alpha = 0$, the primal-dual algorithm is equivalent to the alternating step method [Eckstein '89]

Relations to matrix scaling

- The α-preconditioner tries to normalize the row- and column norms of the matrix K.

Relations to matrix scaling

- The α-preconditioner tries to normalize the row- and column norms of the matrix K.
- This technique is known as matrix scaling or matrix binormalization [Livne, Golub, '04], [Bradeley '10]
- Given a symmetric $n \times n$ matrix A, find a diagonal scaling matrix $D = \text{diag}(d_1, ..., d_n)$ such that $D^{\frac{1}{2}} A D^{\frac{1}{2}}$ has row and column 2-norms equal to one

Relations to matrix scaling

- The α-preconditioner tries to normalize the row- and column norms of the matrix K.
- This technique is known as matrix scaling or matrix binormalization [Livne, Golub, '04], [Bradeley '10]
- Given a symmetric $n \times n$ matrix A, find a diagonal scaling matrix $D = \mathrm{diag}(d_1, ..., d_n)$ such that $D^{\frac{1}{2}} A D^{\frac{1}{2}}$ has row and column 2-norms equal to one
- This is equivalent finding a positive solution to the equations

$$\sum_{j=1}^{n} d_i A_{i,j}^2 d_j = 1\,, \;\; i = 1...n$$

Relations to matrix scaling

- The α-preconditioner tries to normalize the row- and column norms of the matrix K.
- This technique is known as matrix scaling or matrix binormalization [Livne, Golub, '04], [Bradeley '10]
- Given a symmetric $n \times n$ matrix A, find a diagonal scaling matrix $D = \mathrm{diag}(d_1, ..., d_n)$ such that $D^{\frac{1}{2}} A D^{\frac{1}{2}}$ has row and column 2-norms equal to one
- This is equivalent finding a positive solution to the equations

$$\sum_{j=1}^{n} d_i A_{i,j}^2 d_j = 1\,, \;\; i = 1...n$$

- Extensions to general matrices are discussed but most theoretical results hold only in the symmetric case

Left-right-preconditioning

- For the rectangular case we require that $\Sigma^{\frac{1}{2}} K T^{\frac{1}{2}}$ should have row- and column 2-norms as close as possible to one

Left-right-preconditioning

- For the rectangular case we require that $\Sigma^{\frac{1}{2}} K T^{\frac{1}{2}}$ should have row- and column 2-norms as close as possible to one
- Optimized diagonal left-right preconditioners can be computed via

$$\min_{\sigma_i>0,\tau_j>0} \sum_{i=1}^{m} \left(\sum_{j=1}^{n} \sigma_i (K_{i,j})^2 \tau_j - 1 \right)^2 + \sum_{j=1}^{n} \left(\sum_{i=1}^{m} \sigma_i (K_{i,j})^2 \tau_j - 1 \right)^2$$

- Can be solved via alternating minimization (slow)
- Finally, we have to rescale T and Σ such that

$$\| \Sigma^{\frac{1}{2}} K T^{\frac{1}{2}} \|^2 \leq 1$$

Linear programming

- Many applications of LP relaxations in computer vision and machine learning
- LP program in inequality form

$$\min c^T x \quad \text{s.t.} \quad Ax \leq b\,,\; x \geq 0\,,$$

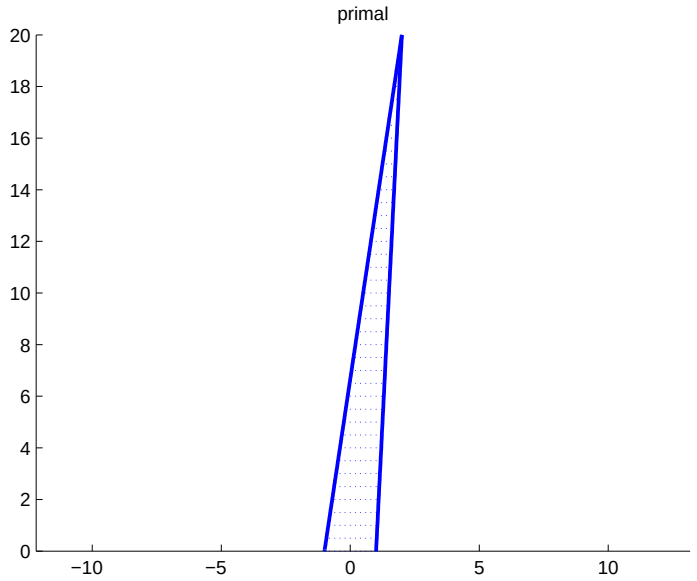
- Preconditioned primal-dual algorithm

$$\begin{cases} x^{k+1} = \mathrm{proj}_{[0,\infty)}\left(x^k - \mathrm{T}(A^T y^k + c)\right) \\ y^{k+1} = \mathrm{proj}_{[0,\infty)}\left(y^k + \Sigma(A(2x^{k+1} - x^k) - b)\right) \end{cases},$$

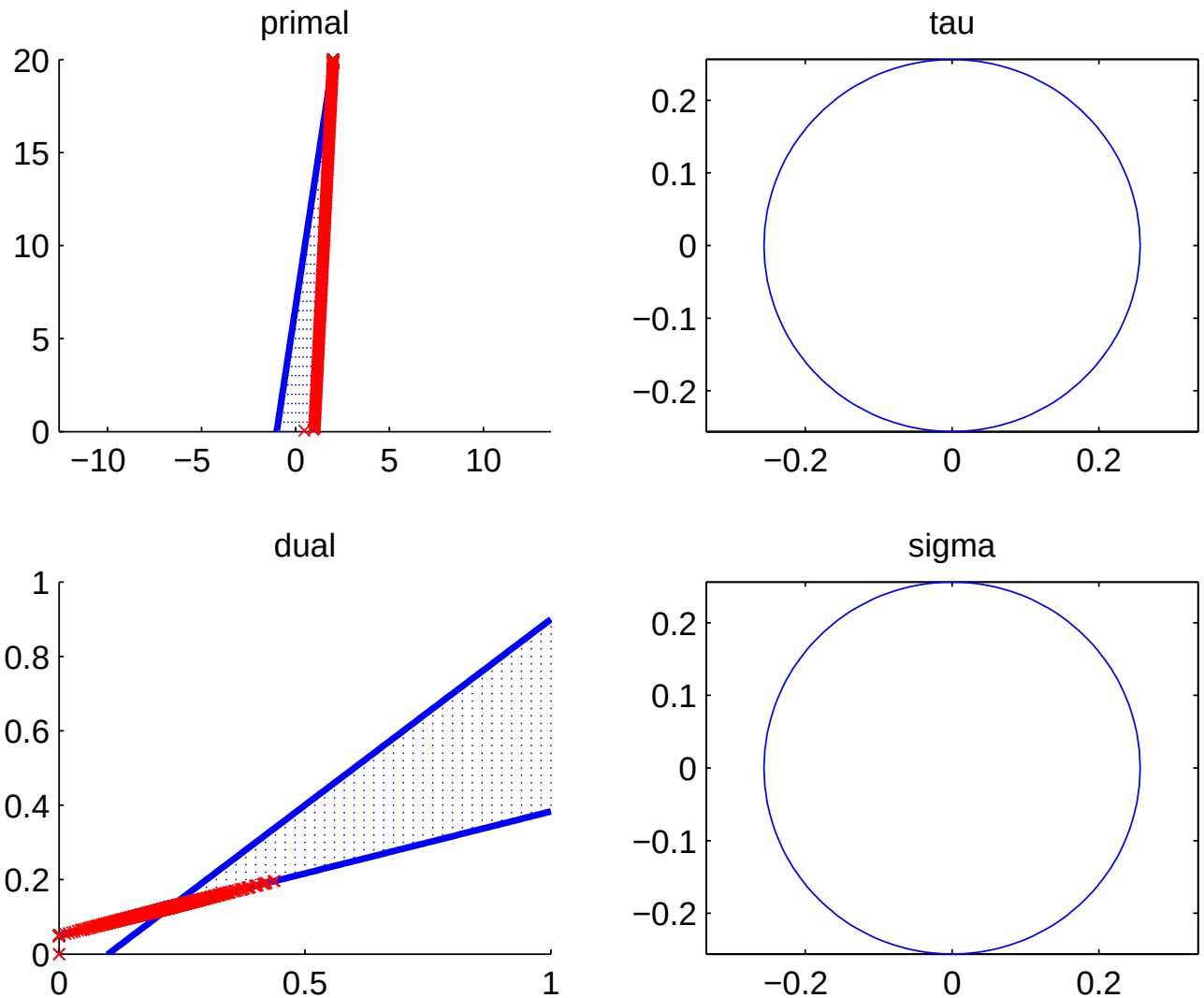
A (simple) 2D example

Consider the following 2D linear program

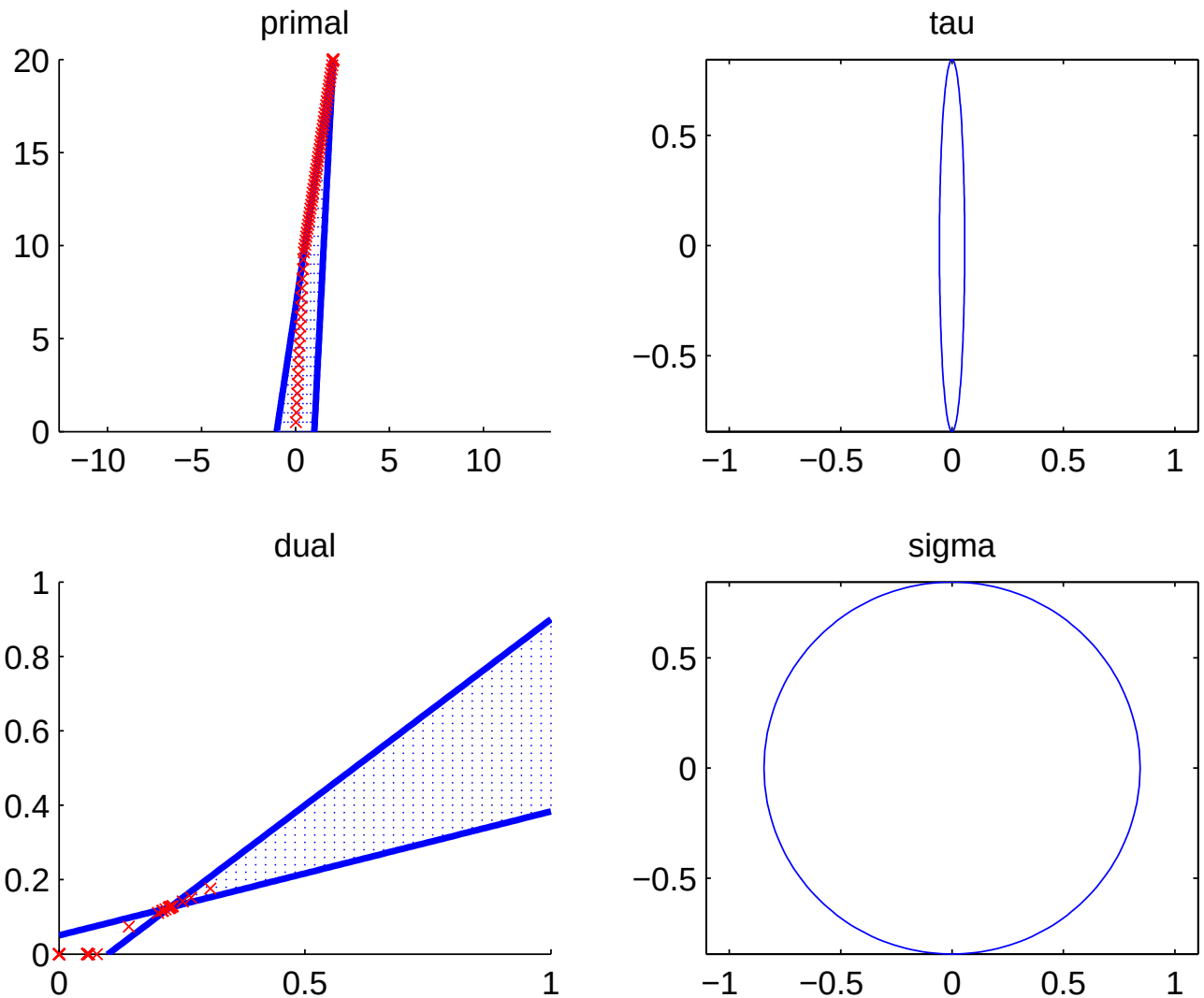
$$c = (-1, -1)^T,\ A = \begin{pmatrix} -20/3 & 1 \\ 20 & -1 \end{pmatrix},\ b = (20/3, 20)^T$$



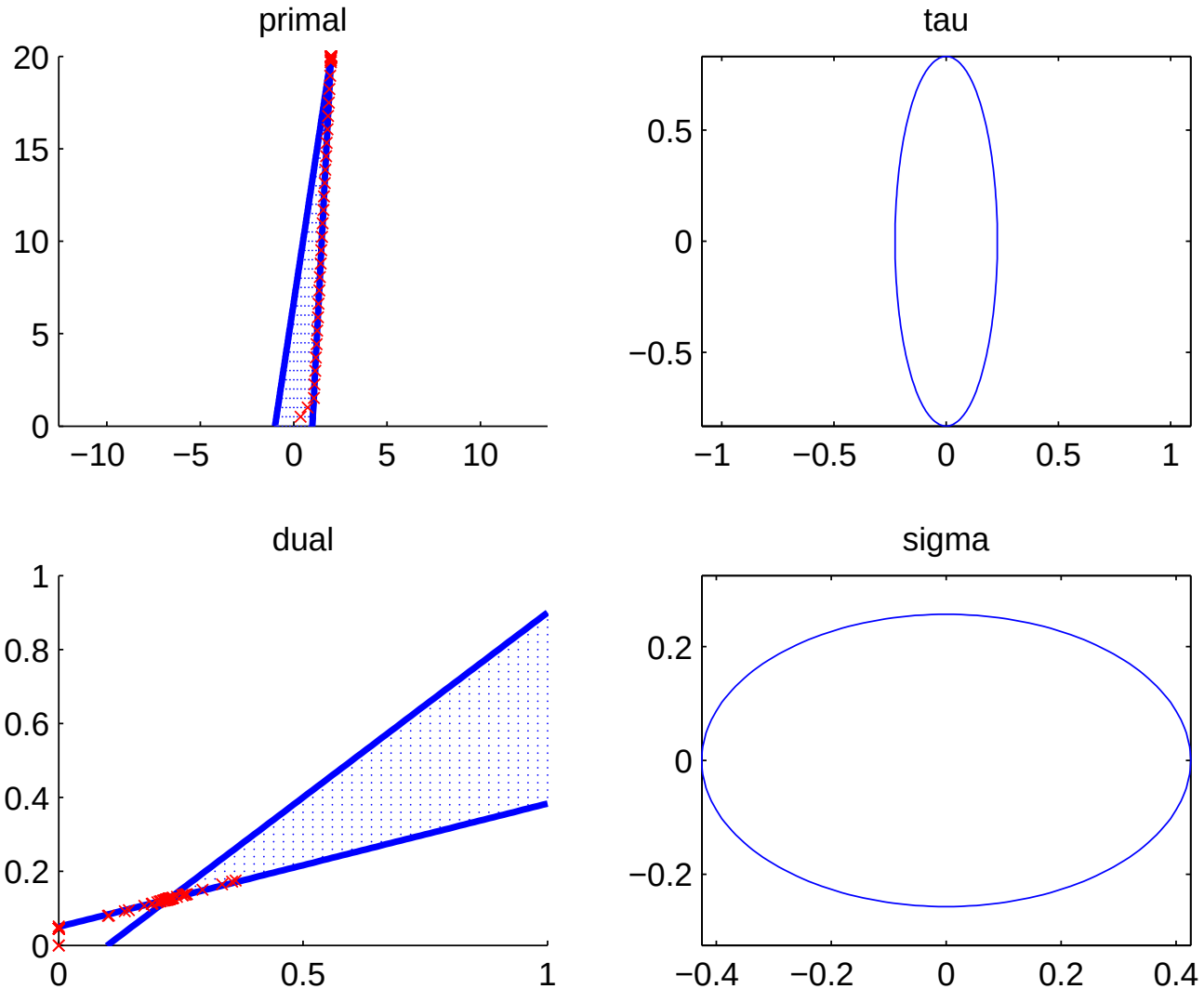
No preconditioning (14030 iterations)



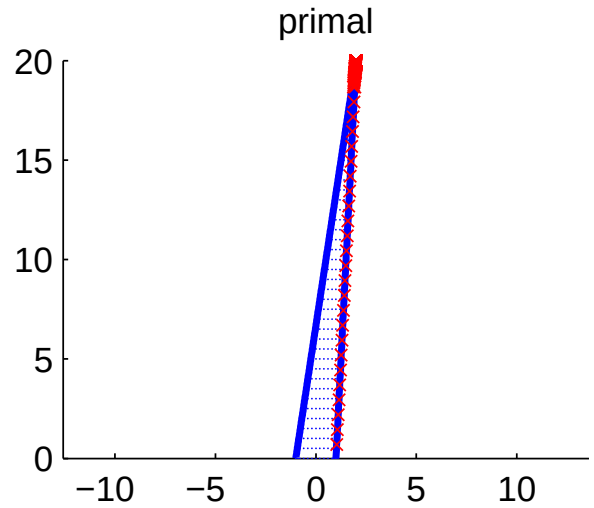
α-preconditioning ($\alpha = 0$) (910 iterations)

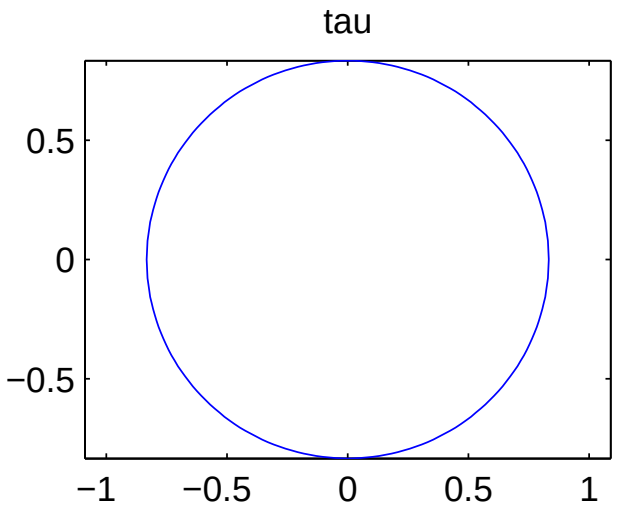


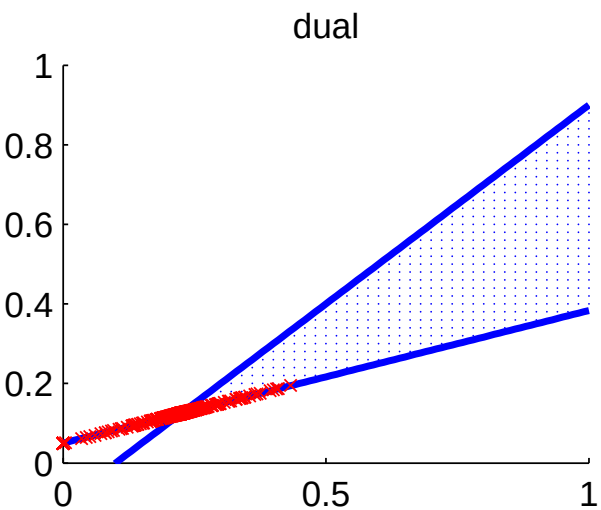
α-preconditioning ($\alpha = 1$) (880 iterations)

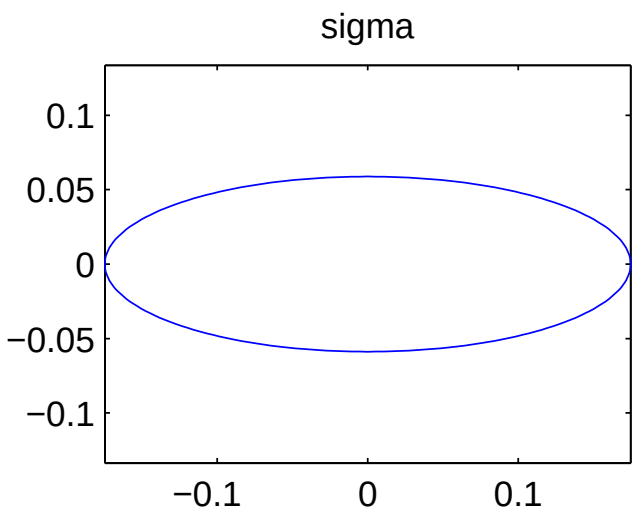


α-preconditioning ($\alpha = 2$) (4350 iterations)

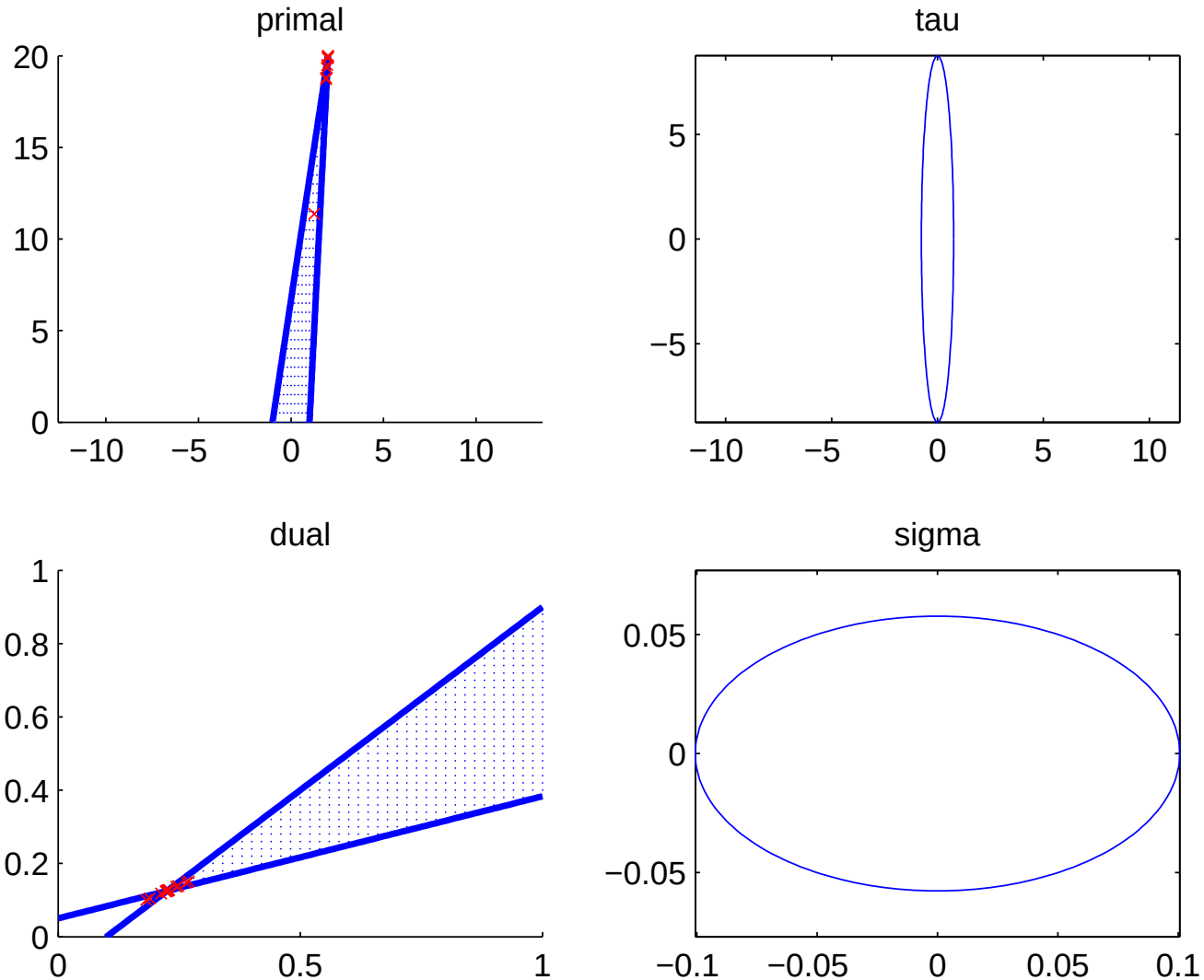








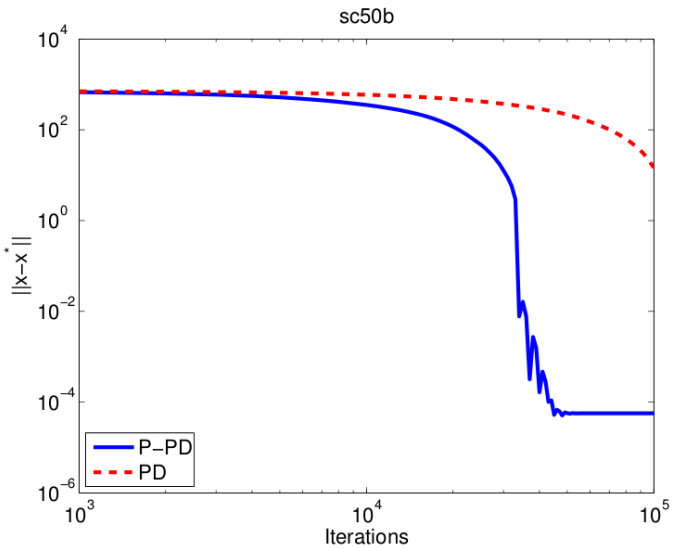
Left-right-preconditioning (190 iterations)

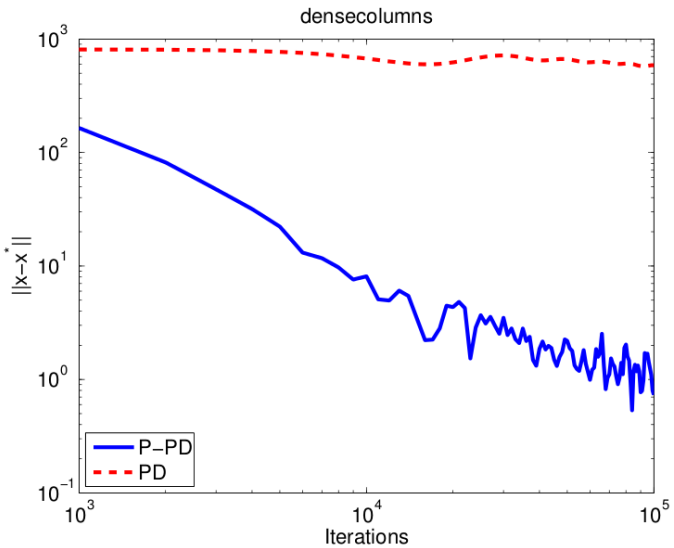


Linear programming, further examples

	IP	PD	P-PD
`sc50b`	0.01s	1.75s	0.49s
`densecolumn`	0.17s	268.51s	0.61s

Table: Comparison of of IP, PD and P-PD on two standard LP test problems.





Graph cuts

- Graph cuts are widely used in computer vision
- Can be written as a weighted total variation energy [Chambolle, '05]

$$\min_u \|D_w u\|_{\ell_1} + \langle u, w^u \rangle \ , \ \text{s.t. } 0 \leq u \leq 1 \ ,$$

- Preconditioned primal-dual algorithm

$$\begin{cases} u^{k+1} = \text{proj}_{[0,1]} \left(u^k + \text{T}(D_w^T y^k - w^u) \right) \\ y^{k+1} = \text{proj}_{[-1,1]} \left(y^k + \Sigma(D_w(2u^{k+1} - u^k)) \right) \ , \end{cases}$$

- Comparison

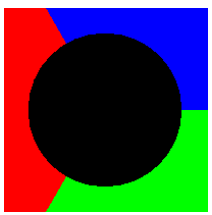
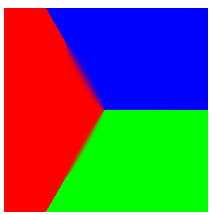
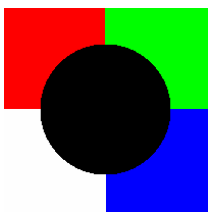
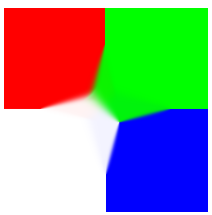
MAXFLOW	PD	P-PD	P-PD-GPU
0.160s	15.75s	8.56s	0.045s

Continuous Potts model

- The Continuous Potts model [Chambolle, Cremers, P. '05] with k phases can be written as a convex problem

$$\min_{u \in S} \max_{q \in B} \sum_{l=1}^{k} \langle Du_l, q_l \rangle + \langle u_l, f_l \rangle \ ,$$

  where S is the simplex constraint and B is an interection of non-local ℓ_2 balls.
- Comparison

	PD	P-PD	Speedup
Synthetic (3 phases)	221.71s	75.65s	2.9
Synthetic (4 phases)	1392.02s	538.83s	2.6
Natural (8 phases)	592.85s	113.76s	5.2

Conclusion

- Preconditioned primal-dual algorithm for convex saddle point problems with known structure
- Equivalent to the proximal point algorithm in a particular norm
- Different choices for diagonal preconditioning
- Applications to non-smooth problems (LP, graph cuts, Potts model, ...)

Conclusion

- Preconditioned primal-dual algorithm for convex saddle point problems with known structure
- Equivalent to the proximal point algorithm in a particular norm
- Different choices for diagonal preconditioning
- Applications to non-smooth problems (LP, graph cuts, Potts model, ...)

- Iterative update of the preconditioners involving also information from F and G
- Accelerated algorithm performs some kind of scalar preconditioning - relations need to be understood better
- Can we improve the convergence rate factor?

Conclusion

- Preconditioned primal-dual algorithm for convex saddle point problems with known structure
- Equivalent to the proximal point algorithm in a particular norm
- Different choices for diagonal preconditioning
- Applications to non-smooth problems (LP, graph cuts, Potts model, ...)

- Iterative update of the preconditioners involving also information from F and G
- Accelerated algorithm performs some kind of scalar preconditioning - relations need to be understood better
- Can we improve the convergence rate factor?

Thank you for your attention!