Grouped Coordinate Descent Algorithms for Robust Edge-Preserving Image Restoration

Jeffrey A. Fessler

EECS Department
The University of Michigan

SPIE'97: Image Reconstruction and Restoration

July 29, 1997

OUTLINE

- Problem Description
- Huber Algorithm
- Optimization Transfer
- Convex Algorithm (ala Lange / De Pierro)
- Grouped Coordinate Descent (GCD) Algorithm
- Anecdotal Results
- Summary

"LINEAR" INVERSE PROBLEM

$$\underline{y} = \boldsymbol{A}\underline{x} + \text{noise}$$

- $\underline{y}$: noisy measurements (blurred image or sinogram)
- $\underline{x}$: unknown object (true image)
- $\boldsymbol{A}$: known system model
  (each column is a point response function)
- Errors in $\boldsymbol{A}$ partially motivate robust methods

Goal: recover an estimate $\underline{\hat{x}}$ of $\underline{x}$ from $\underline{y}$.

DATA-FIT COST FUNCTION

Want $\underline{\hat{x}}$ to "fit the data," i.e. $\underline{y} \approx \boldsymbol{A}\underline{\hat{x}}$

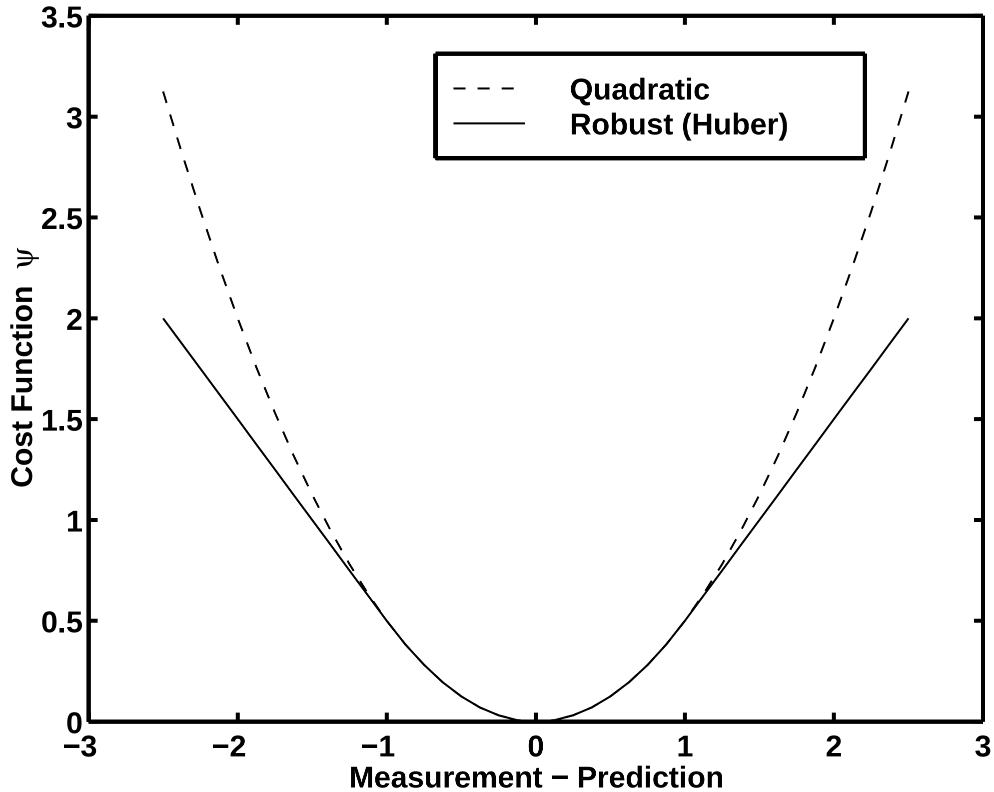
Natural cost function for independent measurement errors:

$$\Phi^{\text{data}}(\underline{x}) = \sum_{i=1}^{m_1} \psi_i^{\text{data}}([\underline{y} - \boldsymbol{A}\underline{x}]_i)$$

- $[\underline{y} - \boldsymbol{A}\underline{x}]_i = y_i - \sum_{j=1}^{p} a_{ij} x_j$
- m_1: length of $\underline{y}$
- ψ_i: convex function.

Traditional choice: $\psi_i(t) = t^2/2$, which is appropriate for Gaussian noise, but is not robust to noise with heavy-tailed distributions.

ROBUST DATA-FIT COST FUNCTION



Example - Huber function:

$$\psi(t) = \begin{cases} t^2/2, & |t| \le \delta, \\ \delta|t| - \delta^2/2, & |t| > \delta \end{cases}$$

ROBUST ESTIMATORS

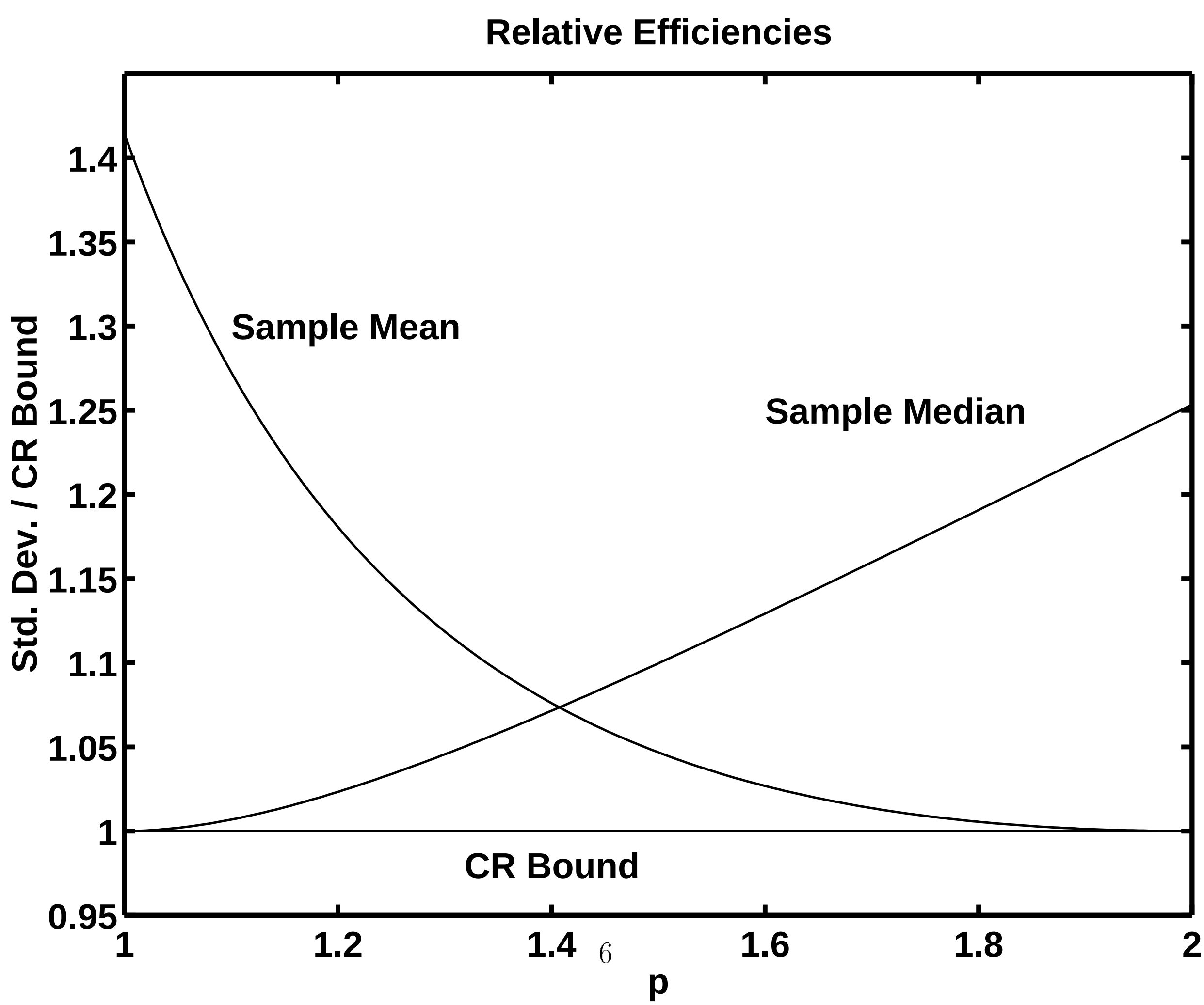
Generalized-Gaussian family of pdfs with unit variance:

$$f_X(x;\mu,p) = \frac{p}{2}\frac{1}{\Gamma(1/p)}\sqrt{r_p}\exp\Big(-|x-\mu|^p r_p^{p/2}\Big) \quad \text{where } r_p = \frac{\Gamma(3/p)}{\Gamma(1/p)}.$$

Asymptotic variance of the sample median estimator for μ is:

$$\frac{1}{4nf^2(\mu)} = \frac{1}{n}\frac{\Gamma^2(1/p)}{p^2 r_p} \quad \text{(cf } 1/n \text{ for the sample mean).}$$

CR bound for estimating μ: $\sigma^2_{\hat{\mu}} \geq \frac{1}{n}\frac{1}{p^2 r_p}\frac{\Gamma(1/p)}{\Gamma(2-1/p)}$.



REGULARIZATION

Minimizing Φ^{data} is inadequate for ill-conditioned inverse problems.

Prior "knowledge" of piece-wise smoothness:

- $x_j - x_{j-1} \approx 0$ (piece-wise constant)
- $x_{j-1} - 2x_j + x_{j+1} \approx 0$ (piece-wise linear)
- $x_j \approx 0$ (support constraints)
- $\ldots$ Combining: $\boldsymbol{C}\underline{x} \approx \underline{z}$

Regularized cost function: $\Phi(\underline{x}) = \Phi^{\text{data}}(\underline{x}) + \Phi^{\text{penalty}}(\underline{x}),$

$$\Phi^{\text{penalty}}(\underline{x}) = \sum_{i=1}^{m_2} \psi_i^{\text{penalty}}([\boldsymbol{C}\underline{x} - \underline{z}]_i)$$

EXAMPLE: ROUGHNESS PENALTY (AKA GIBBS PRIOR)

$$\boldsymbol{D}_n = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ & & \ddots & \ddots & \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix} \qquad \boldsymbol{C} = \begin{bmatrix} \boldsymbol{I}_{n_y} \otimes \boldsymbol{D}_{n_x} \\ \boldsymbol{D}_{n_y} \otimes \boldsymbol{I}_{n_x} \end{bmatrix}$$

where $\otimes$ denotes the Kronecker matrix product.

If $\underline{z} = \underline{0}$ and $\mathcal{N}_j$ is the four pixel neighborhood of pixel j, then

$$\Phi^{\text{penalty}}(\underline{x}) = \sum_j \sum_{k \in \mathcal{N}_j} \psi_{j,k}(x_j - x_k)$$

Conventional (Tikhonov-Miller) regularization: $\psi(t) = t^2/2$.
(Gaussian prior)

For edge-preserving image recovery, need non-quadratic $\psi(\cdot)$, such as Huber function.

UNIFIED COST FUNCTION

$$\boxed{\Phi(\underline{x}) = \sum_{i=1}^{m} \psi_i([\boldsymbol{B}\underline{x} - \underline{c}]_i)}$$

Regularized edge-preserving cost function is a special case:

$$\Phi(\underline{x}) = \Phi^{\text{data}}(\underline{x}) + \Phi^{\text{penalty}}(\underline{x}), \quad \boldsymbol{B} = \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{C} \end{bmatrix}, \quad \underline{c} = \begin{bmatrix} \underline{y} \\ \underline{z} \end{bmatrix}$$

Optimization problem:

$$\underline{\hat{x}} = \arg\min_{\underline{x}} \Phi(\underline{x}) \quad \text{or} \quad \underline{\hat{x}} = \arg\min_{\underline{x} \geq \underline{0}} \Phi(\underline{x}).$$

OPTIMIZATION

Simple in quadratic case where $\psi_i(t) = t^2/2 \; \forall i$

$$\underline{\hat{x}} = (\boldsymbol{B}'\boldsymbol{B})^{-1}\boldsymbol{B}'\underline{c}$$

Good algorithms:

- Preconditioned conjugate gradients
- Coordinate descent (Gauss-Siedel)

Challenging for non-quadratic ψ_i's
Very challenging for non-convex ψ_i's

Proposition: algorithms tailored to structure of Φ can outperform general purpose optimization methods.

but cannot solve it all...

ASSUMPTIONS

$\boldsymbol{B}$ has full column rank, so $\boldsymbol{M} > \boldsymbol{0} \;\Rightarrow\; \boldsymbol{B}'\boldsymbol{M}\boldsymbol{B} > \boldsymbol{0}$
(Easily achieved with sensible regularization design)

- ψ is symmetric
- ψ is everywhere differentiable (and therefore continuous)
- $\dot{\psi}(t) = d/dt\, \psi(t)$ is non-decreasing (and hence ψ is convex)
- $\boxed{\omega_\psi(t) = \dot{\psi}(t)/t}$ is non-increasing for $t \geq 0$
- $\omega_\psi(0) = \lim_{t\to 0} \dot{\psi}(t)/t$ is finite and nonzero, i.e. $0 < \omega_\psi(0) < \infty$

Φ has a unique minimizer
(Easily ensured with perturbation of regularizer)

rules out entropy, $|t|^p$

to understand ω, look at...

UNCONSTRAINED SOLUTION

$$\boxed{\Phi(\underline{x}) = \sum_{i=1}^{m} \psi_i([\boldsymbol{B}\underline{x} - \underline{c}]_i)}$$

Column gradient:

$$\nabla\Phi(\underline{x}) = \boldsymbol{B}'\boldsymbol{\Omega}(\underline{x})(\boldsymbol{B}\underline{x} - \underline{c}), \qquad \nabla\Phi(\underline{x})\big|_{\underline{x}=\hat{\underline{x}}} = \mathbf{0}$$

$$\text{where} \quad \boldsymbol{\Omega}(\underline{x}) = \mathrm{diag}\{\omega_{\psi_i}([\boldsymbol{B}\underline{x} - \underline{c}]_i)\}$$

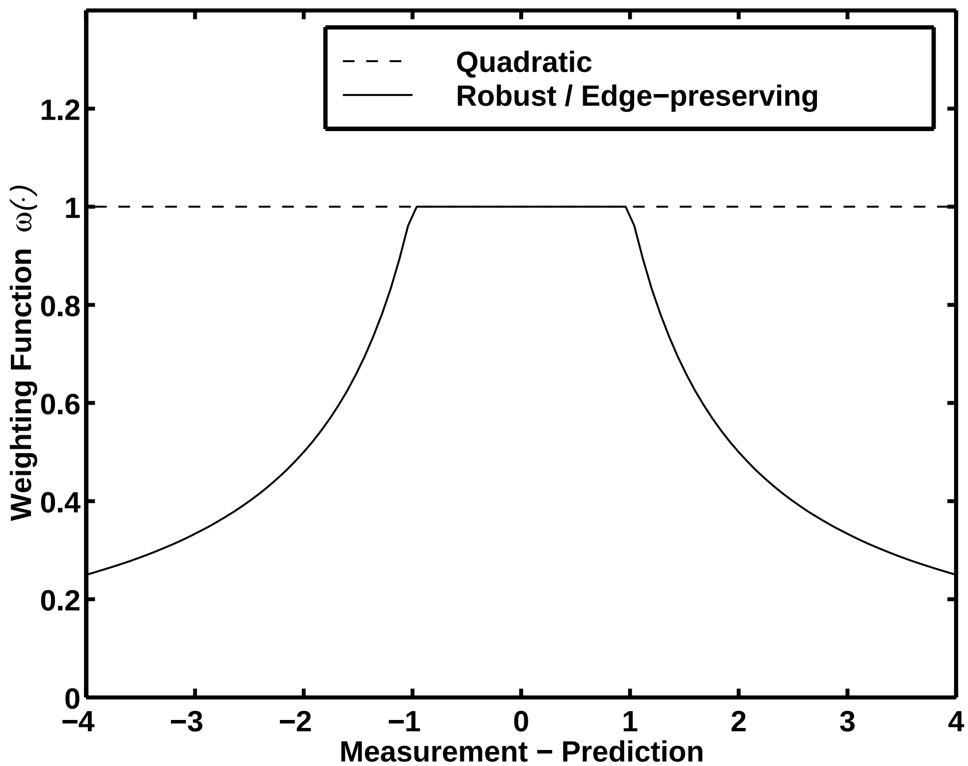
Unconstrained solution:

$$\begin{aligned}
\hat{\underline{x}} &= [\boldsymbol{B}'\boldsymbol{\Omega}(\hat{\underline{x}})\boldsymbol{B}]^{-1}\boldsymbol{B}'\boldsymbol{\Omega}(\hat{\underline{x}})\underline{c} \\
&= \arg\min_{\underline{x}} \frac{1}{2}(\underline{c} - \boldsymbol{B}\underline{x})'\boldsymbol{\Omega}(\hat{\underline{x}})(\underline{c} - \boldsymbol{B}\underline{x})
\end{aligned}$$

(ala WLS, but <u>weights</u> depend on estimate $\hat{\underline{x}}$, hence nonlinear)

Therefore need iterative algorithm...

WEIGHTING FUNCTIONS ω_ψ



NEWTON-RAPHSON ALGORITHM

$$\boxed{\underline{x}^{n+1} = \underline{x}^n - [\boldsymbol{B}'\boldsymbol{\Lambda}(\underline{x}^n)\boldsymbol{B}]^{-1}\nabla\Phi(\underline{x}^n)}$$

where

$$\boldsymbol{\Lambda}(\underline{x}^n) = \operatorname{diag}\left\{\ddot{\psi}_i([\boldsymbol{B}\underline{x} - \underline{c}]_i)\right\}$$

Advantage:

- Super-linear convergence rate (if convergent)

Disadvantages:

- Requires twice-differentiable ψ_i's
- Not guaranteed to converge
- Not guaranteed to monotonically decrease Φ
- Does not enforce nonnegativity constraint
- Impractical for image recovery due to matrix inverse

General purpose remedy: bound-constrained Quasi-Newton algorithms

HUBER ALGORITHM (1981)

Recall $\quad \hat{\underline{x}} = [\boldsymbol{B}'\boldsymbol{\Omega}(\hat{\underline{x}})\boldsymbol{B}]^{-1}\boldsymbol{B}'\boldsymbol{\Omega}(\hat{\underline{x}})\underline{c} = \hat{\underline{x}} - [\boldsymbol{B}'\boldsymbol{\Omega}(\hat{\underline{x}})\boldsymbol{B}]^{-1}\nabla\Phi(\hat{\underline{x}})$

Successive Substitutions:

$$\boxed{\underline{x}^{n+1} = \underline{x}^n - [\boldsymbol{B}'\boldsymbol{\Omega}(\underline{x}^n)\boldsymbol{B}]^{-1}\nabla\Phi(\underline{x}^n)}$$

Advantages:

- Monotonically decreases Φ
- Converges globally to unique minimizer (not shown by Huber)

Disadvantages:

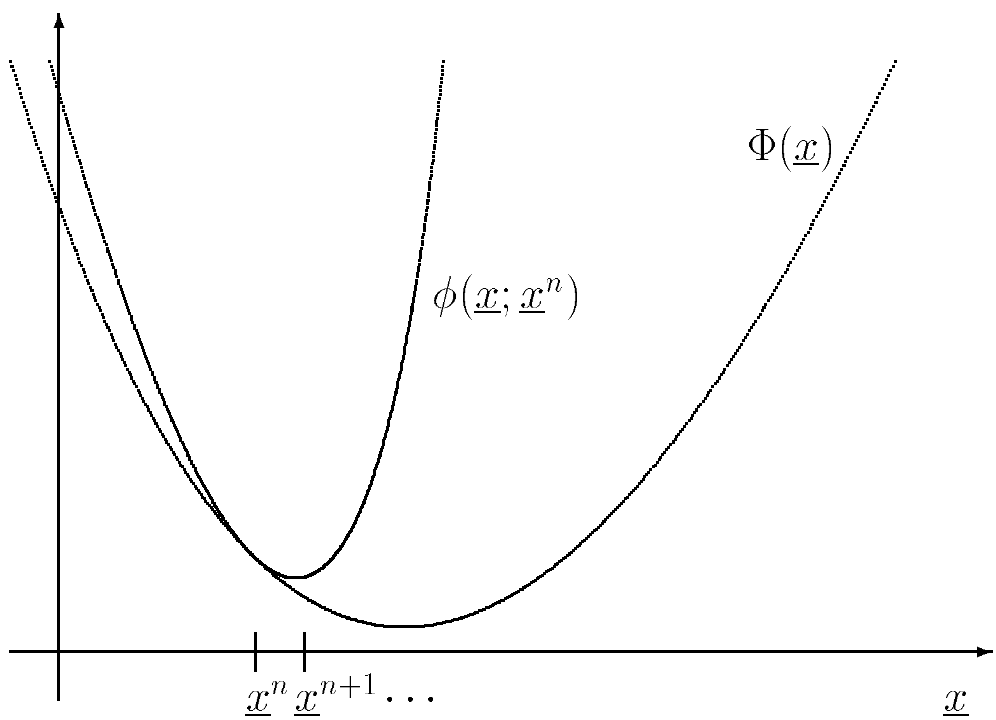
- Does not enforce nonnegativity constraint
- Impractical for image recovery due to matrix inverse

Successive substitutions is often not convergent. Why here?

OPTIMIZATION TRANSFER

$$\underline{x}^{n+1} = \arg\min_{\underline{x}} \phi^{\text{Huber}}(\underline{x}; \underline{x}^n)$$

$$\phi^{\text{Huber}}(\underline{x}; \underline{x}^n) = \frac{1}{2}(\underline{c} - \boldsymbol{B}\underline{x})' \boldsymbol{\Omega}(\underline{x}^n)(\underline{c} - \boldsymbol{B}\underline{x})$$



Minimizing surrogate function ϕ ensures a monotone decrease in Φ if:

- $\phi(\underline{x}^n; \underline{x}^n) = \Phi(\underline{x}^n)$
- $\nabla_{\underline{x}} \phi(\underline{x}; \underline{x}^n)\big|_{\underline{x}=\underline{x}^n} = \nabla \Phi(\underline{x})\big|_{\underline{x}=\underline{x}^n}$
- $\Phi(\underline{x}) \leq \phi(\underline{x}; \underline{x}^n)$.

These 3 (sufficient) conditions are satisfied by ϕ^{Huber}

OPTIMIZATION TRANSFER IN 2D

GENERALIZED HUBER ALGORITHM

$$\boxed{\underline{x}^{n+1} = \underline{x}^n - \boldsymbol{M}_n^{-1} \nabla \Phi(\underline{x}^n)}$$

where

$$\boldsymbol{M}_n \geq \boldsymbol{B}' \boldsymbol{\Omega}(\underline{x}^n) \boldsymbol{B}$$

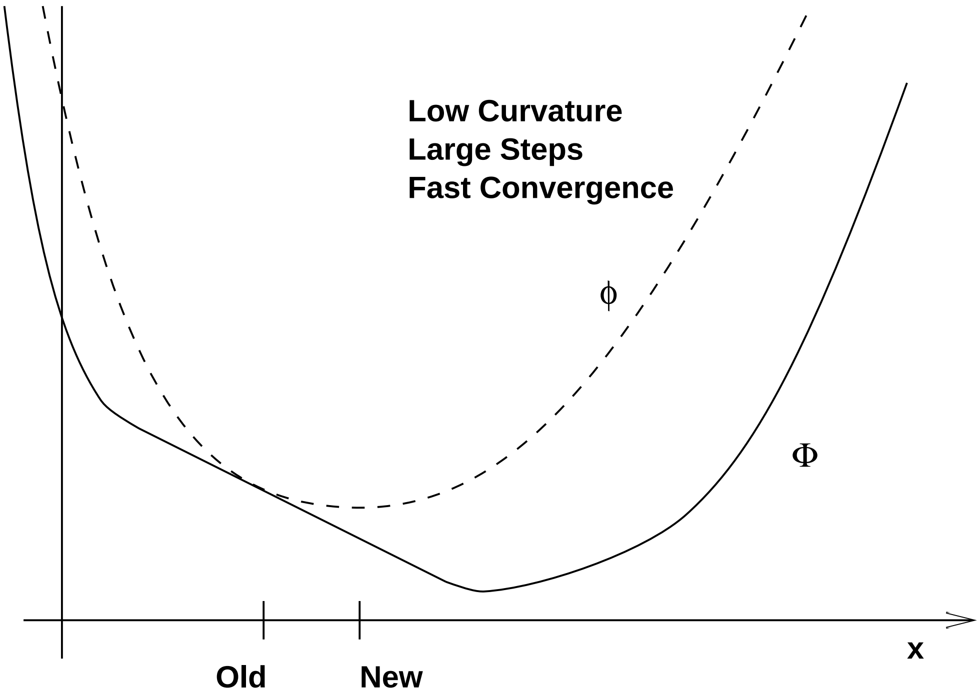
Advantages:

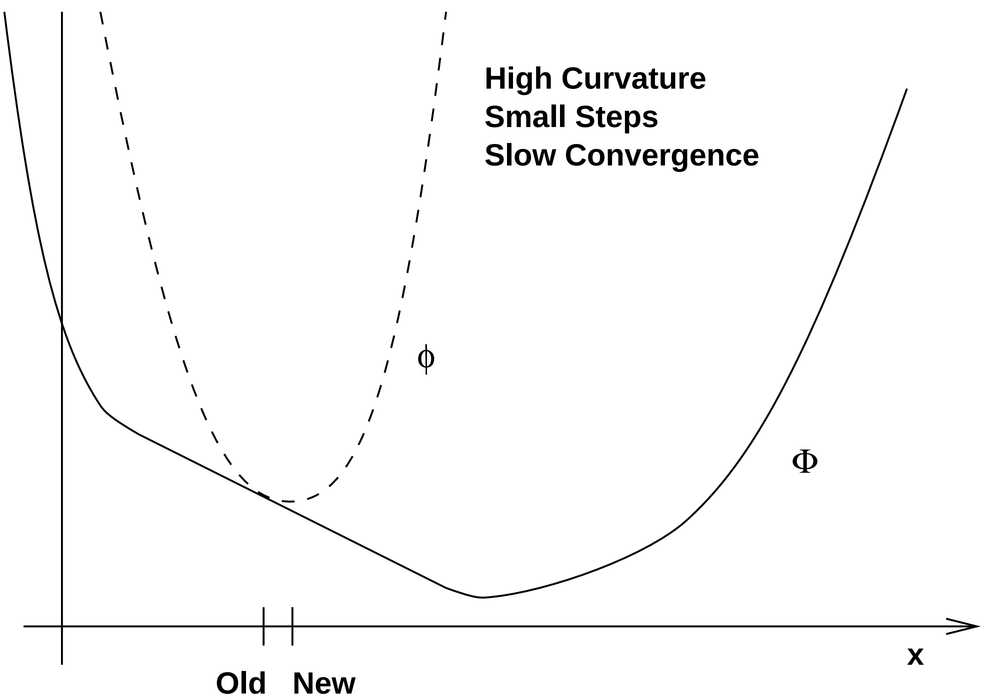
- Monotonically decreases Φ
- Converges globally to unique minimizer
- Can choose $\boldsymbol{M}_n$ to be easily invertible, e.g. diagonal.
  (Or splitting matrices more generally)

Disadvantages:

- Does not enforce nonnegativity constraint
- Converges slower than Huber algorithm

CONVERGENCE RATE





can we beat this tradeoff?

USING THE STRUCTURE OF Φ

De Pierro's decomposition (uses form of argument of ψ_i):

$$\boldsymbol{B}\underline{x} - \underline{c} = \sum_{j=1}^{p} \alpha_{ij} \left[\frac{b_{ij}}{\alpha_{ij}} (x_j - x_j^n) + \boldsymbol{B}\underline{x}^n - \underline{c} \right]$$

provided $\alpha_{ij} \geq 0$ and $\sum_{j=1}^{p} \alpha_{ij} = 1, \ \forall i$.

The α_{ij}'s are algorithm design factors.
Natural choice is $\alpha_{ij} = |b_{ij}| / \sum_{j=1}^{p} |b_{ik}|$.

By <u>convexity</u> of ψ_i:

$$\psi_i([\boldsymbol{B}\underline{x} - \underline{c}]_i) \leq \sum_{j=1}^{p} \alpha_{ij} \psi_i \left(\frac{b_{ij}}{\alpha_{ij}} (x_j - x_j^n) + \boldsymbol{B}\underline{x}^n - \underline{c} \right)$$

Construct surrogate function:

$$\Phi(\underline{x}) = \sum_{i=1}^{m} \psi_i([\boldsymbol{B}\underline{x} - \underline{c}]_i) \leq \phi^{\mathrm{LDC}}(\underline{x}; \underline{x}^n)$$

$$\phi^{\mathrm{LDC}}(\underline{x}; \underline{x}^n) = \sum_{j=1}^{p} \phi_j(x_j; \underline{x}^n),$$

$$\phi_j(x_j; \underline{x}^n) = \sum_{i=1}^{m} \alpha_{ij} \psi_i \left(\frac{b_{ij}}{\alpha_{ij}} (x_j - x_j^n) + \boldsymbol{B}\underline{x}^n - \underline{c} \right)$$

ϕ^{LDC} satisfies the 3 conditions for monotonicity

LANGE / DE PIERRO CONVEX ALGORITHM

$$\boxed{\underline{x}^{n+1} = \arg\min_{\underline{x}} \phi^{\mathrm{LDC}}(\underline{x}; \underline{x}^n)}$$

$$\begin{aligned} x_j^{n+1} &= \arg\min_{x_j \geq 0} \phi_j(x_j; \underline{x}^n) \\ &= \arg\min_{x_j \geq 0} \sum_{i=1}^{m} \alpha_{ij} \psi_i \left(\frac{b_{ij}}{\alpha_{ij}} (x_j - x_j^n) + \boldsymbol{B}\underline{x}^n - \underline{c} \right) \end{aligned}$$

Advantages:

- Monotonically decreases Φ
- Converges globally to unique minimizer
- No matrix inversion required
- Can enforce nonnegativity constraint
- Parallelizable (all pixels updated simultaneously)

Disadvantages:

- Requires subiteration for minimization
  Solution: use 1-D Huber algorithm
- Very slow convergence (ala EM algorithm)
  Solution: update only a subset of the pixels simultaneously

GROUPED COORDINATE DESCENT ALGORITHM

Construct surrogate function using Lange / De Pierro convexity method but for only a (large) subset of the pixels.

Pixel Groups (2x3)

1	5	3	1	5	3	1	5
4	2	6	4	2	6	4	2
1	5	3	1	5	3	1	5
4	2	6	4	2	6	4	2
1	5	3	1	5	3	1	5
4	2	6	4	2	6	4	2

Pixels separated => decoupled => fast convergence

Many pixels per subiteration => parallelizable

Retains advantages of Convex Algorithm, but converges faster.

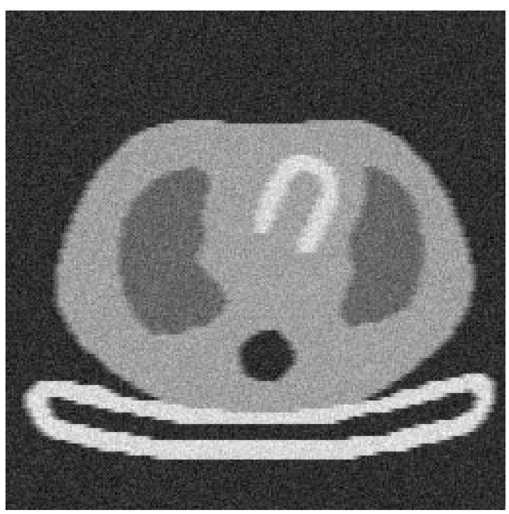
Disadvantages:

- Slightly less parallelizable.
- Slightly more complicated implementation
- Difficult to exploit structure of $\boldsymbol{B}$
  (e.g. FFTs for shift-invariant PSF, separable blur in PET)

SIMULATION EXAMPLE

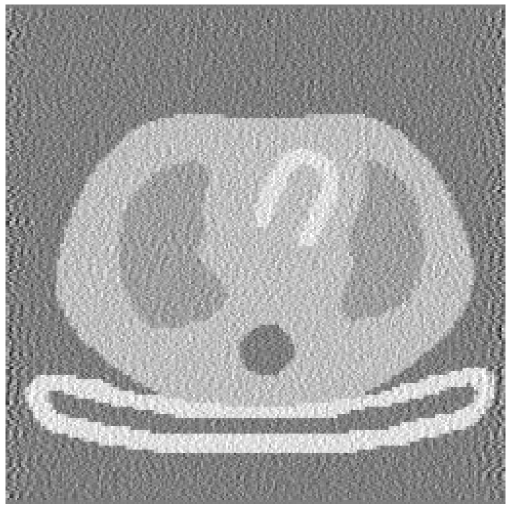
True object $\underline{x}$:

With 5 pixel horizontal motion blur and Gaussian noise, $\underline{y}$ is:



RESTORED IMAGE

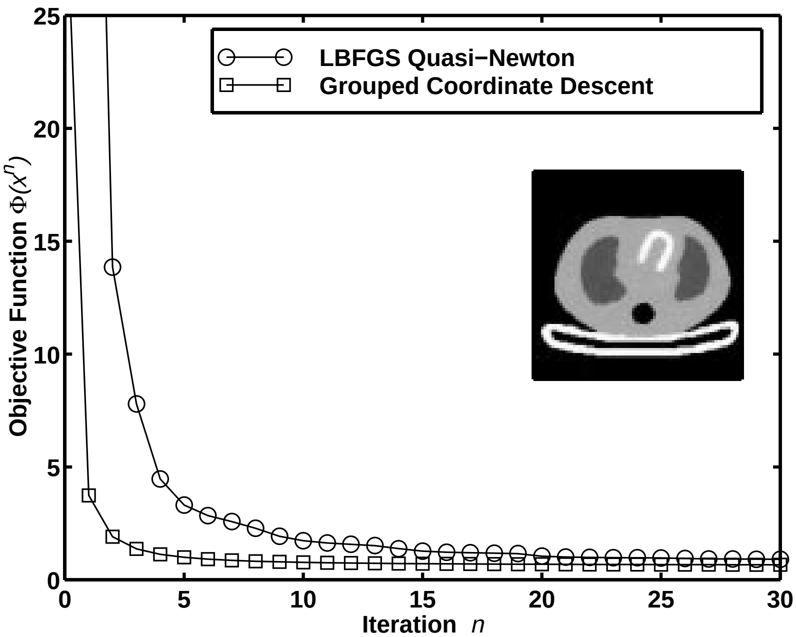
Wiener filter:

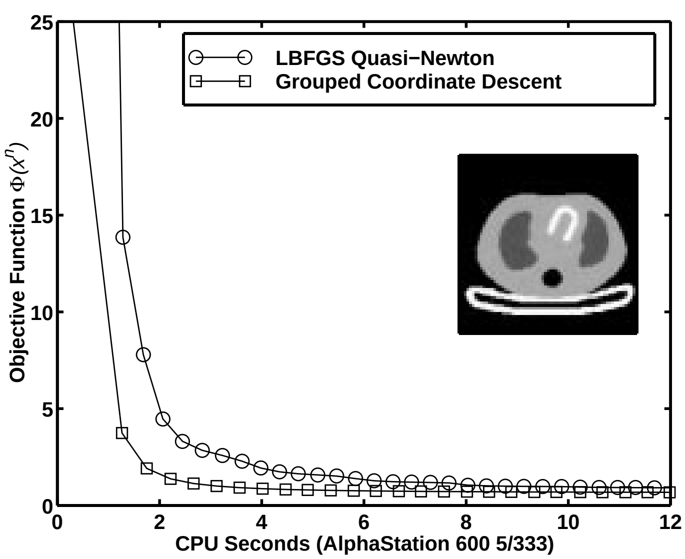


Edge-preserving restoration $\underline{\hat{x}}$:

Huber function used for ψ_i's for piece-wise smoothness.
15 iterations of Grouped Coordinate Descent.

CONVERGENCE RATES

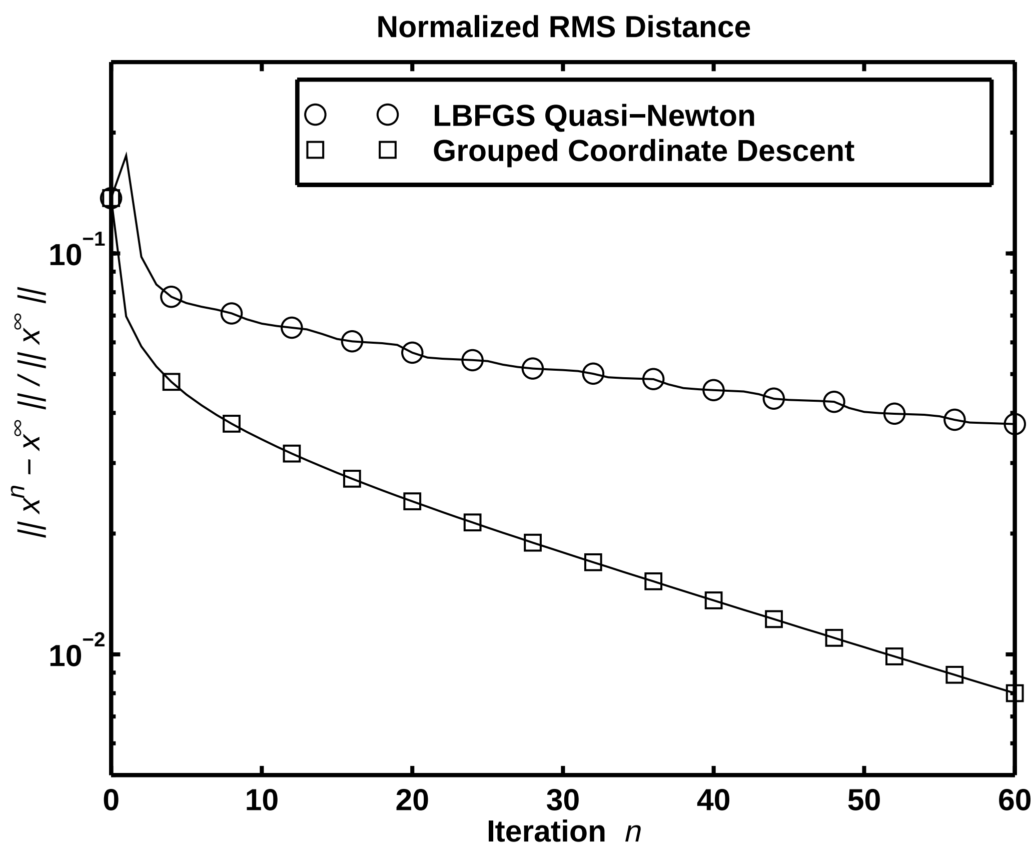




LBFGS: Limited Memory Bound Constrained Quasi-Newton Method
(R. Byrd, P. Lu, J. Nocedal, R. Schnabel, C. Zhu)

NORMALIZED RMS DISTANCE

$$\frac{\|\underline{x}^n - \underline{x}^\infty\|}{\|\underline{x}^\infty\|}$$



$\underline{x}^\infty$: 400 iterations of single-coordinate descent

(Thanks to Web Stayman for interfacing LBFGS with ASPIRE.)

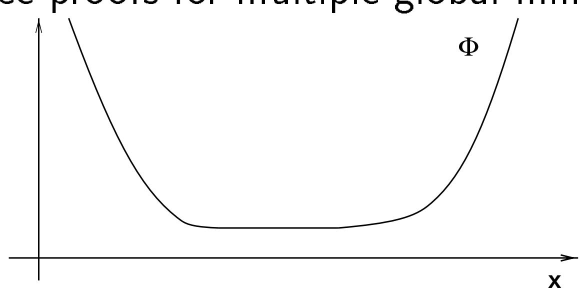
SUMMARY

Grouped Coordinate Descent Algorithm

- Accommodates non-quadratic cost function
  (for noise robustness and preserving edges)
- Monotonically decreases Φ
- Converges globally to unique minimizer
- Easily accommodates nonnegativity constraint
- Parallelizable
- Converges faster than a general-purpose optimization method

---

Future Work:
Extend convergence proofs for multiple global minimizers:



Slides and paper available from:

`http://www.eecs.umich.edu/~fessler/`