Parallelizable algorithms for image recovery problems

Jeffrey A. Fessler and Saowapak Sotthivirat

EECS Department
The University of Michigan

Workshop on Mathematics in Image Processing
WMIP2000, The University of Hong Kong
Dec. 16, 2000

Executive Summary

- In a wide variety of estimation problems, one estimates an unknown parameter vector $\mathbf{x}^{\text{true}}$ by minimizing a *partially separable* cost function:

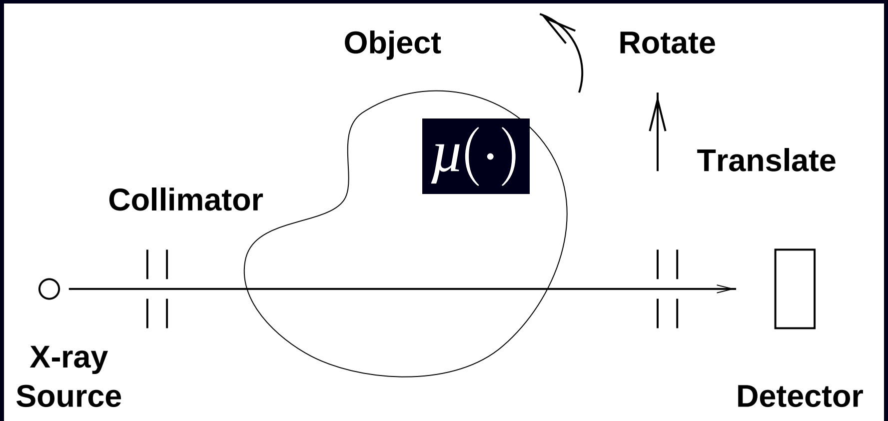
$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x} \geq \mathbf{0}} \Phi(\mathbf{x}), \qquad \Phi(\mathbf{x}) = \sum_i \psi_i([\mathbf{B}\mathbf{x} - \mathbf{c}]_i), \qquad [\mathbf{B}\mathbf{x} - \mathbf{c}]_i = \sum_j b_{ij} x_j - c_i.$$

- Fast methods for estimating $\mathbf{x}^{\text{true}}$ by minimizing $\Phi(\mathbf{x})$ are essential for successful routine use in applications such as medical tomography.
- We have developed fast converging algorithms for minimizing $\Phi(\mathbf{x})$.
- One algorithm has the fast convergence of coordinate descent, yet is parallelizable.
- The new algorithms converge faster than general-purpose minimization methods.

Outline

- Motivating applications and cost functions
- Edge-preserving regularization
- Unified cost function
- Minimization algorithms
  - Optimization transfer
  - Separable paraboloidal surrogates (SPS) algorithm
  - Paraboloidal surrogate coordinate descent (PSCD) algorithm
  - Parallelizable coordinate descent algorithm
- Representative results
- Summary and future work

Application: X-ray Computed Tomography



Statistical model: $$Y_i \sim \text{Poisson}\{b_i \exp(-[\mathbf{Ax}]_i) + r_i\}$$

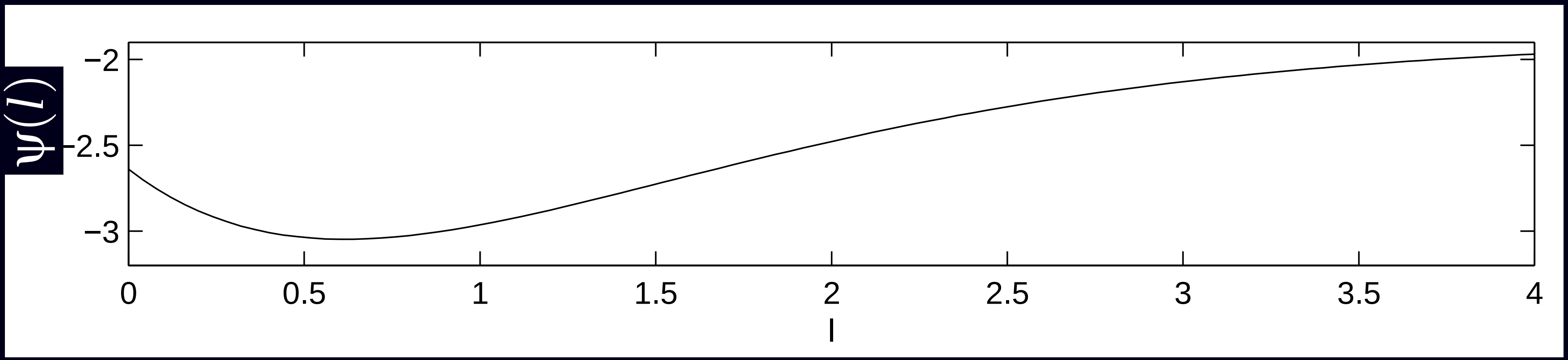
- Y_i: measurement along ith ray (statistically independent), $i = 1, \ldots, n_d$
- x_j: unknown attenuation coefficient in the jth voxel
- b_i: mean number of transmitted photons along ith ray
- a_{ij}: Radon projection matrix
- r_i: random coincidences and scatter
- Beer's Law for photon survival probability: $e^{-\int \mu(\cdot)\, dl}$
- $[\mathbf{Ax}]_i$: discrete approximation to line integral along ith ray

X-ray CT Statistical Image Reconstruction

It is natural to estimate the attenuation image $\mathbf{x}$ by finding the "best fit" to the sinogram data, as measured by the log-likelihood:

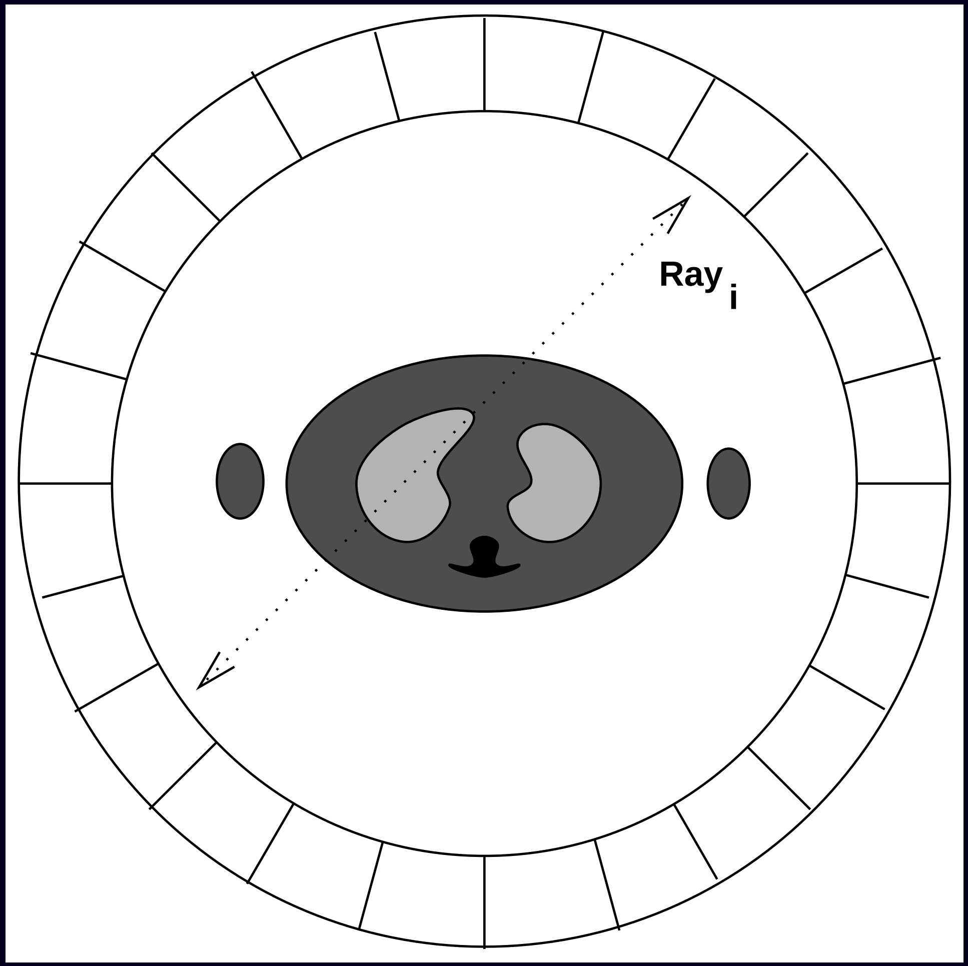
$$\hat{\mathbf{x}}_{\mathrm{ML}} \triangleq \arg\min_{\mathbf{x} \geq \mathbf{0}} \Phi^{\mathrm{data}}(\mathbf{x}) \quad \text{where} \quad \Phi^{\mathrm{data}}(\mathbf{x}) = \sum_{i=1}^{n_d} \psi_i([\mathbf{A}\mathbf{x}]_i)$$

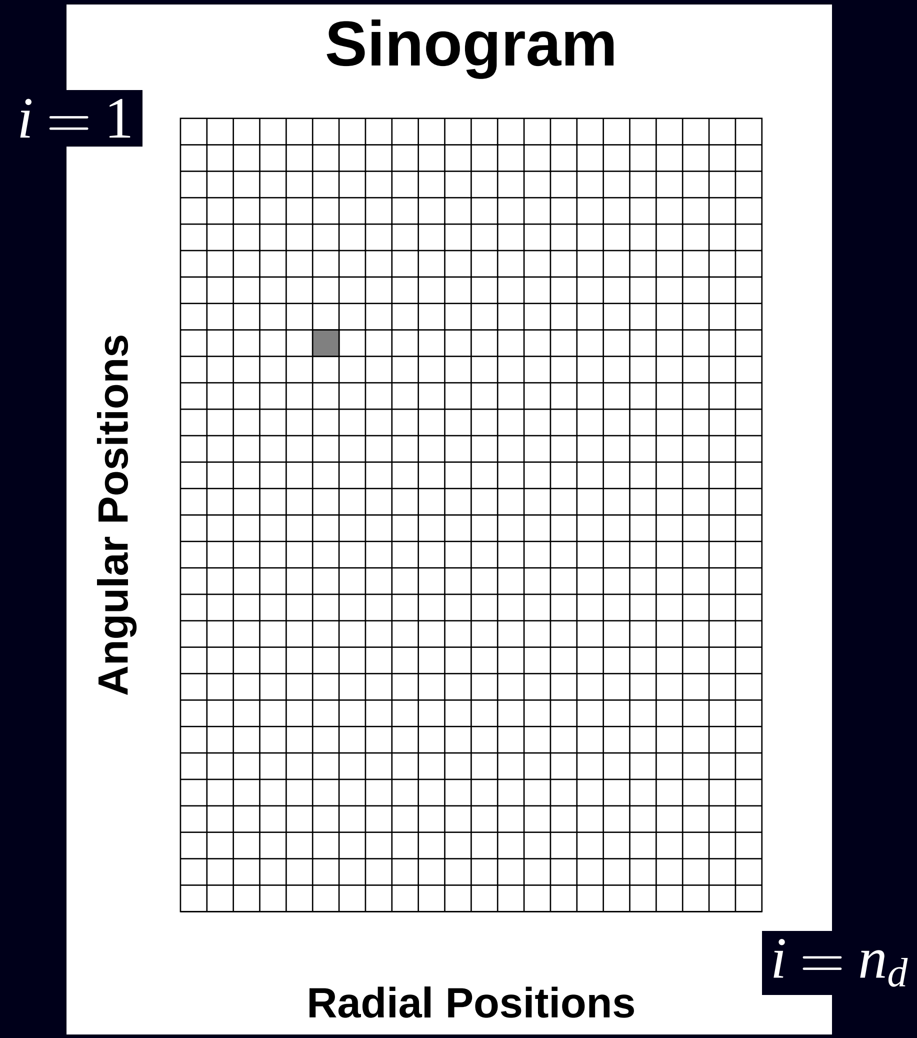
$$\psi_i(l) \triangleq \left(b_i e^{-l} + r_i\right) - Y_i \log\left(b_i e^{-l} + r_i\right).$$



- Summation form due to independence of recorded photon counts.
- Inner products $[\mathbf{A}\mathbf{x}]_i$ due to Beer's law and line integrals
- ψ_i's determined by Poisson negative log-likelihood

Application: PET Image Reconstruction





$$n_d \approx (n_{\text{crystals}})^2$$

PET Image Reconstruction

Most statistical methods for PET image reconstruction are based on the following Poisson statistical model.

$$Y_i \sim \text{Poisson}\left\{ \varepsilon_i s_i \sum_j g_{ij} x_j + r_i \right\}, \; i = 1, \ldots, n_d.$$

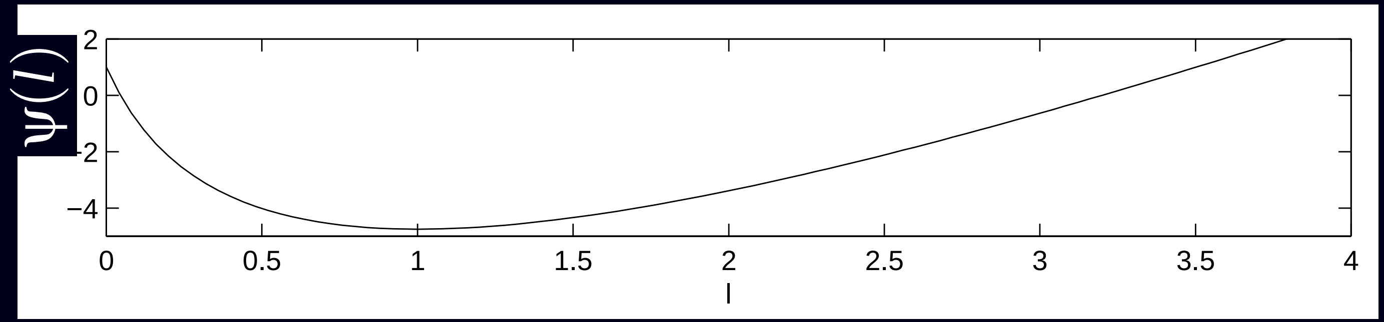
- Y_i: measured counts in sinogram bins (statistically independent)
- x_j: unknown radiotracer concentration in the jth voxel
- ε_i: ith detector efficiency
- s_i: photon survival probability along ith ray (attenuation)
- g_{ij}: projection matrix
- r_i: random coincidences and scatter
- n_d: number of detector pairs

Maximum-Likelihood PET Image Reconstruction

If the Poisson model is valid, it is natural to estimate the emission image $\mathbf{x}$ by finding the "best fit" to the sinogram data, as measured by the log-likelihood:

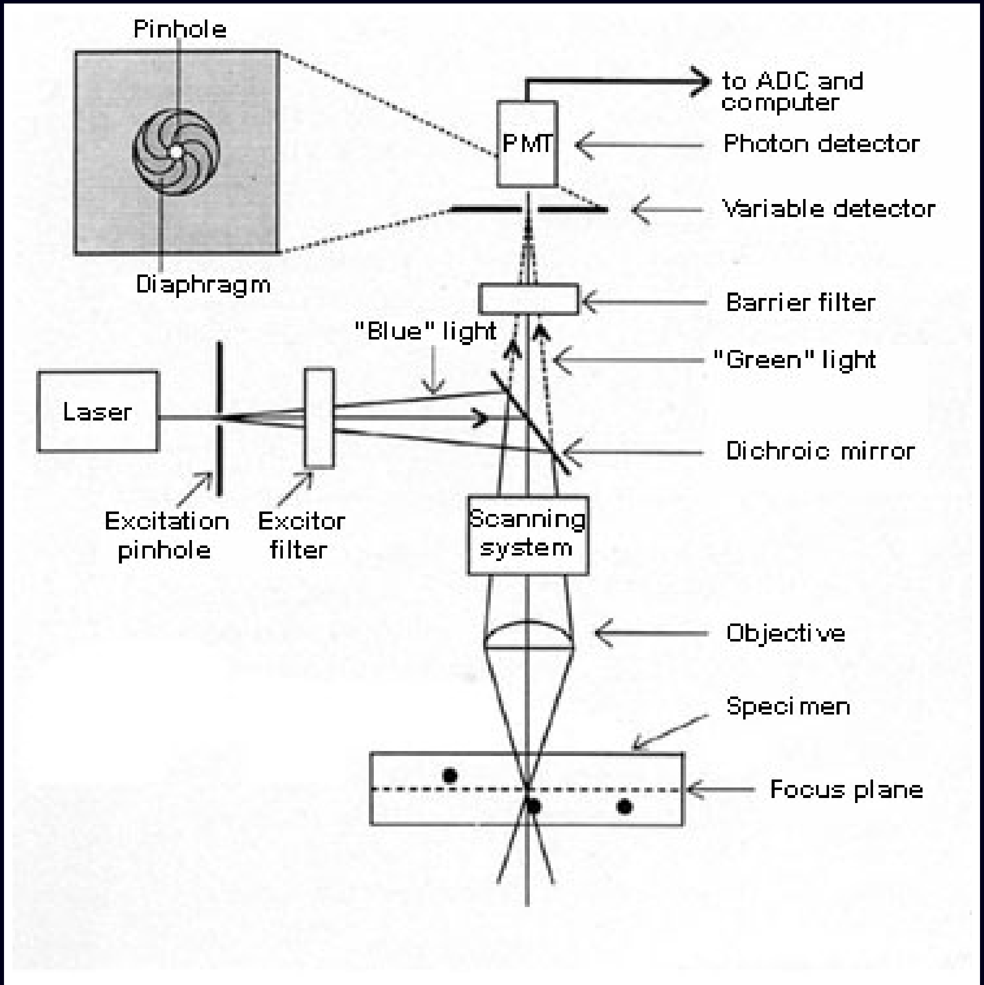
$$\hat{\mathbf{x}}_{\mathrm{ML}} \triangleq \arg\min_{\mathbf{x}\geq\mathbf{0}} \Phi^{\mathrm{data}}(\mathbf{x}) \quad \text{where} \quad \Phi^{\mathrm{data}}(\mathbf{x}) = \sum_{i=1}^{n_d} \psi_i([\mathbf{A}\mathbf{x}]_i)$$

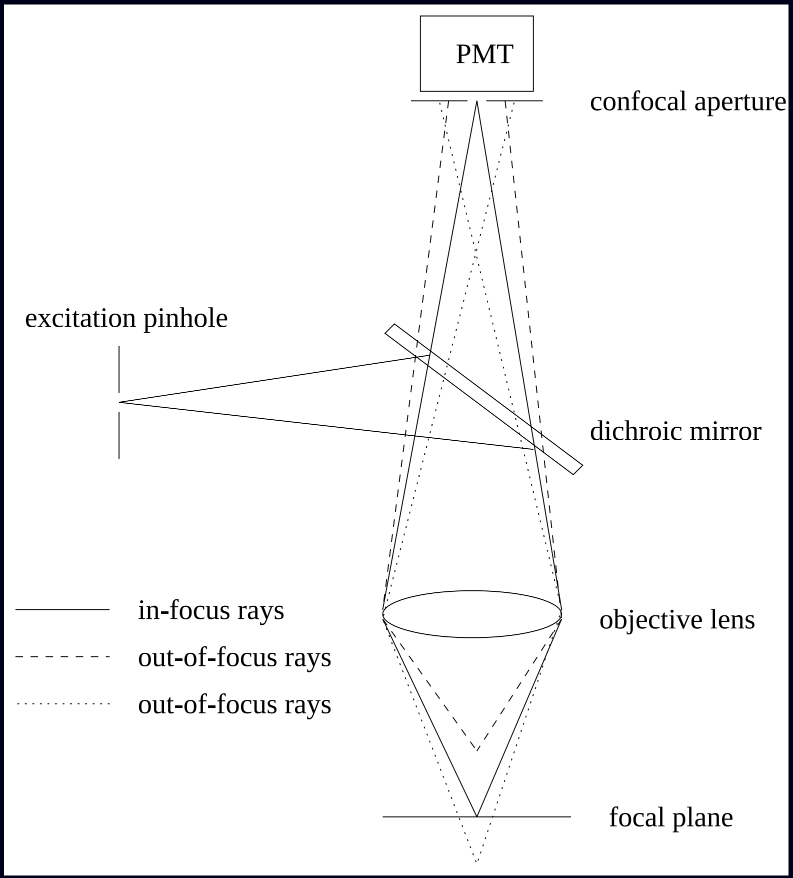
$$\psi_i(l) \triangleq (l + r_i) - Y_i \log(l + r_i), \quad a_{ij} \triangleq \varepsilon_i s_i g_{ij}.$$



- Summation form due to independence of recorded photon counts.
- Inner products $[\mathbf{A}\mathbf{x}]_i$ due to Radon tomographic projection
- ψ_i's determined by Poisson negative log-likelihood

Application: Confocal Microscopy 3D Image Restoration





Cost function is comparable to that of PET / SPECT.

Application: Robust Multiuser Detection

Wang and Poor, Feb. 1999 IEEE Tr. Sig. Proc.
"Robust multi-user detection in non-Gaussian channels"

Model for direct-sequence code-division multiple access (CDMA):

$$Y_i = \sum_{j=1}^{K} a_{ij} x_j + N_i, \; i = 1, \ldots, n_d$$

- Y_i: sampled output of chip-matched filter
- x_j: jth information bit scaled by received amplitude
- N_i: possibly non-Gaussian noise
- a_{ij}: signature sequence of jth user

Robust bit estimator (using, *e.g.*, the Huber function):

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \Phi(\mathbf{x}), \qquad \Phi(\mathbf{x}) = \sum_{i=1}^{n_d} \psi(Y_i - [\mathbf{A}\mathbf{x}]_i)$$

Application: Physics-based MR image reconstruction

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \text{noise}$$

- $\mathbf{y}$: samples in spatial frequency space
- $\mathbf{x}$: object transverse magnetization
- $\mathbf{A}$: Fourier transform modified by magnetic field inhomogeneity

$$Y_i = \sum_{j=1}^{n_p} x_j \exp\left(\sqrt{-1}2\pi\left[\underline{k}_i \cdot \underline{r}_j + \Delta_j t_i\right]\right)$$

- $\underline{k}_i$: frequency space location of ith sample
- $\underline{r}_j$: coordinates of jth voxel
- Δ_j: field inhomogeneity induced off-resonance frequency for jth voxel
- t_i: time of ith sample

Gaussian noise, so $\psi_i(t) = t^2/2$ (least squares estimation)

Edge-preserving Regularization

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

Generic 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)
- $x_j \approx 0, \; j \in J \subset \{1, \ldots, n_p\}$ (support constraints)
- $\ldots$

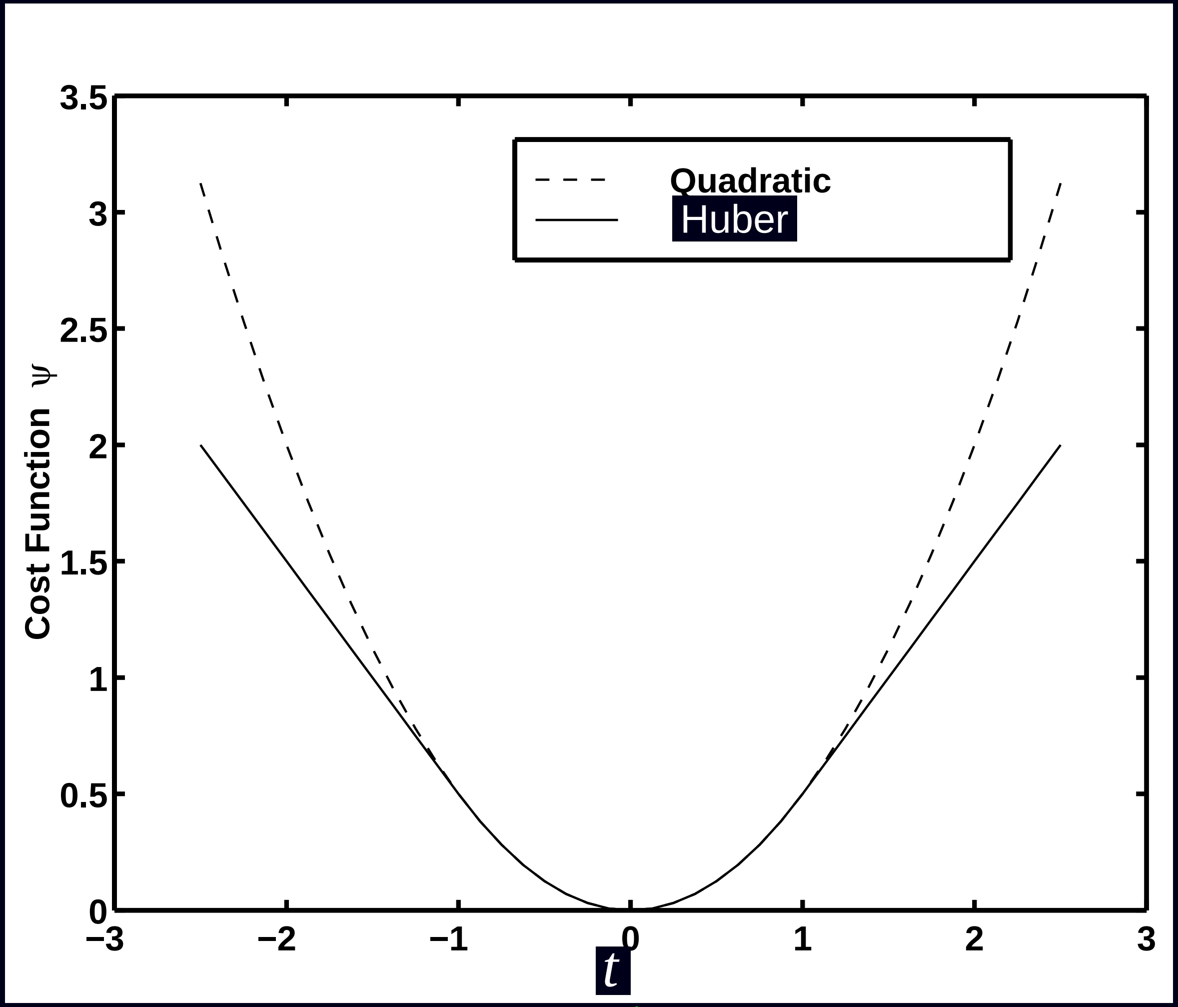
Combining: $\mathbf{C}\mathbf{x} \approx \mathbf{z}$ (where typically $\mathbf{z} = \mathbf{0}$).

Expressed as penalty function:

$$\Phi^{\text{penalty}}(\mathbf{x}) = \sum_k \psi_k^{\text{penalty}}?([\mathbf{C}\mathbf{x} - \mathbf{z}]_k).$$

To "preserve" edges, ψ_k^{penalty} should be nonquadratic.

Example of edge-preserving potential function

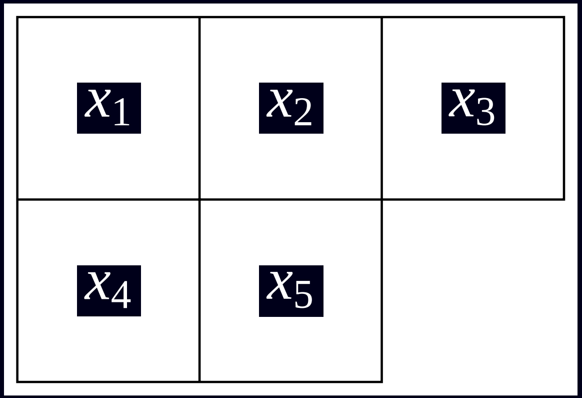


Huber function: $\psi(t) = \begin{cases} t^2/2, & |t| \leq \delta, \\ \delta|t| - \delta^2/2, & |t| > \delta \end{cases}$

Penalty Function: General Form

$$\Phi^{\text{penalty}}(\mathbf{x}) = \sum_k \psi_k([\mathbf{C}\mathbf{x}]_k), \quad \text{where} \quad [\mathbf{C}\mathbf{x}]_k = \sum_j c_{kj} x_j$$

Example:



$$\mathbf{C}\mathbf{x} = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 \\ -1 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} x_2 - x_1 \\ x_3 - x_2 \\ x_5 - x_4 \\ x_4 - x_1 \\ x_5 - x_2 \end{bmatrix}$$

Unified Cost Function

$$\boxed{\Phi(\mathbf{x}) = \sum_{i=1}^{N} \psi_i([\mathbf{B}\mathbf{x} - \mathbf{c}]_i)}$$ "partially separable"

Regularized edge-preserving cost function is a special case:

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

Optimization problem:

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

This formulation encompasses a wide variety of inverse problems.

Challenges: nonnegativity constraint, nonquadratic ψ_i's, size of $\mathbf{B}$.

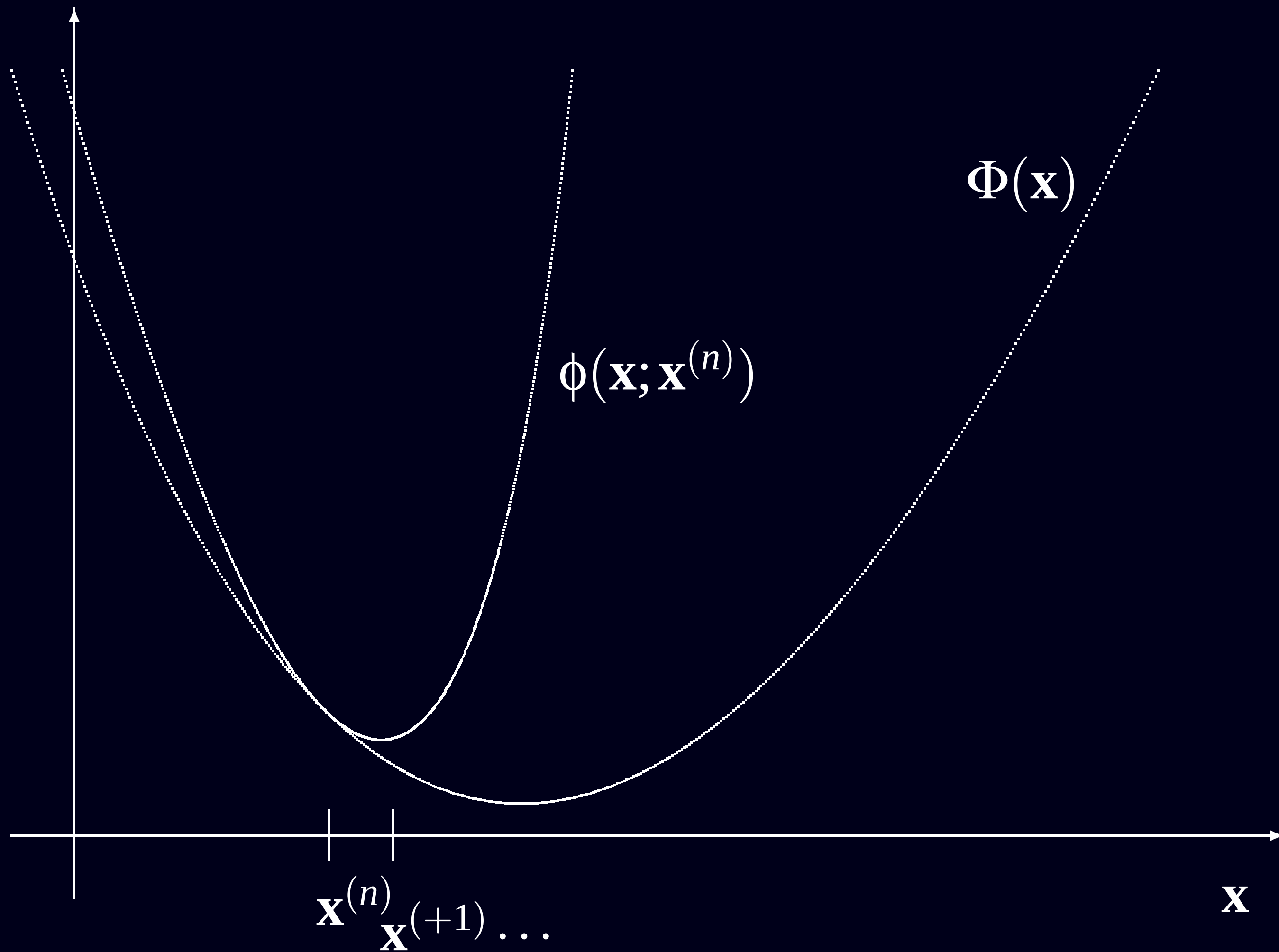
Ideal Algorithm

$$\hat{\mathbf{x}} \triangleq \arg\min_{\mathbf{x} \geq \mathbf{0}} \Phi(\mathbf{x}) \quad \text{(global minimizer)}$$

stable and convergent	$\{\mathbf{x}^{(n)}\}$ converges to $\hat{\mathbf{x}}$ if run indefinitely
converges quickly	$\{\mathbf{x}^{(n)}\}$ gets "close" to $\hat{\mathbf{x}}$ in just a few iterations
globally convergent	$\lim_n \mathbf{x}^{(n)}$ independent of starting image
fast	requires minimal computation per iteration
robust	insensitive to finite numerical precision
user friendly	nothing to adjust (*e.g.* acceleration factors)
monotonic	$\Phi(\mathbf{x}^{(n)})$ increases every iteration
parallelizable	(when necessary)
simple	easy to program and debug
flexible	accommodates any type of system model (matrix stored by row or column or projector/backprojector)

Choices: forgo one or more of the above

Optimization Transfer (1D illustration)



Optimization Transfer

(cf EM Algorithm)

- E-step: choose surrogate function $\phi(\mathbf{x};\mathbf{x}^{(n)})$
- M-step: minimize surrogate function

$$\mathbf{x}^{(n+1)} = \arg\min_{\mathbf{x} \geq \mathbf{0}} \phi(\mathbf{x};\mathbf{x}^{(n)})$$

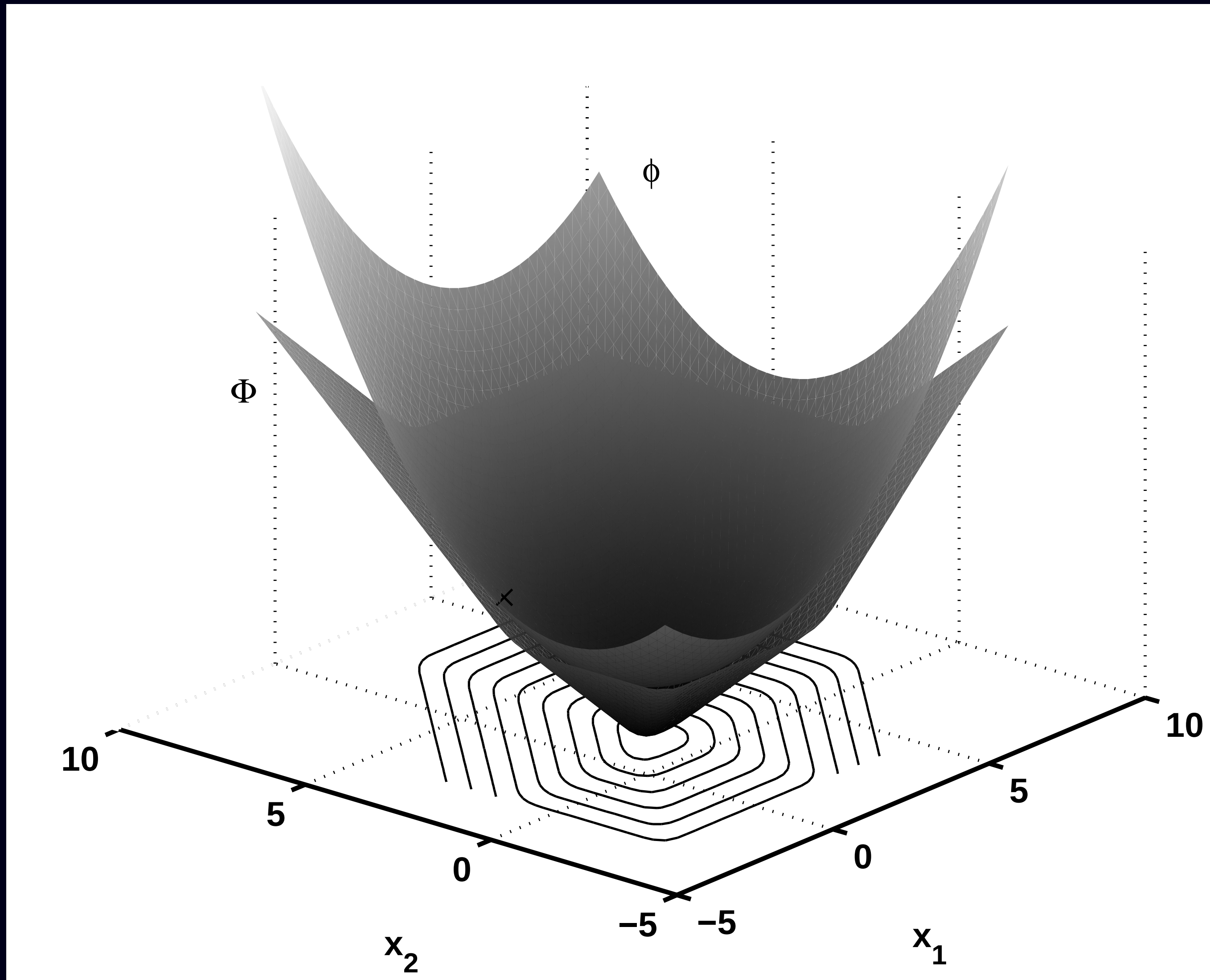
Surrogate design goals:

- Easy to "compute"
- Easy to minimize
- Fast convergence rate
- Monotone convergence

$$\Phi(\mathbf{x}^{(n)}) - \Phi(\mathbf{x}) \geq \phi(\mathbf{x}^{(n)};\mathbf{x}^{(n)}) - \phi(\mathbf{x};\mathbf{x}^{(n)})$$

Often mutually incompatible goals $\therefore$ compromises

Optimization Transfer in 2D



Exploiting Partial Separability (E-step)

Cost Function — Paraboloidal Surrogate Function

$$\Phi(\mathbf{x}) = \sum_{i=1}^{N} \psi_i([\mathbf{B}\mathbf{x} - \mathbf{c}]_i) \;\leq\; \phi(\mathbf{x}; \mathbf{x}^{(n)}) \triangleq \sum_{i=1}^{N} q_i([\mathbf{B}\mathbf{x} - \mathbf{c}]_i; t_i^{(n)}),$$

where $t_i^{(n)} \triangleq [\mathbf{B}\mathbf{x}^{(n)} - \mathbf{c}]_i)$.

1-D tangent parabola surrogate:

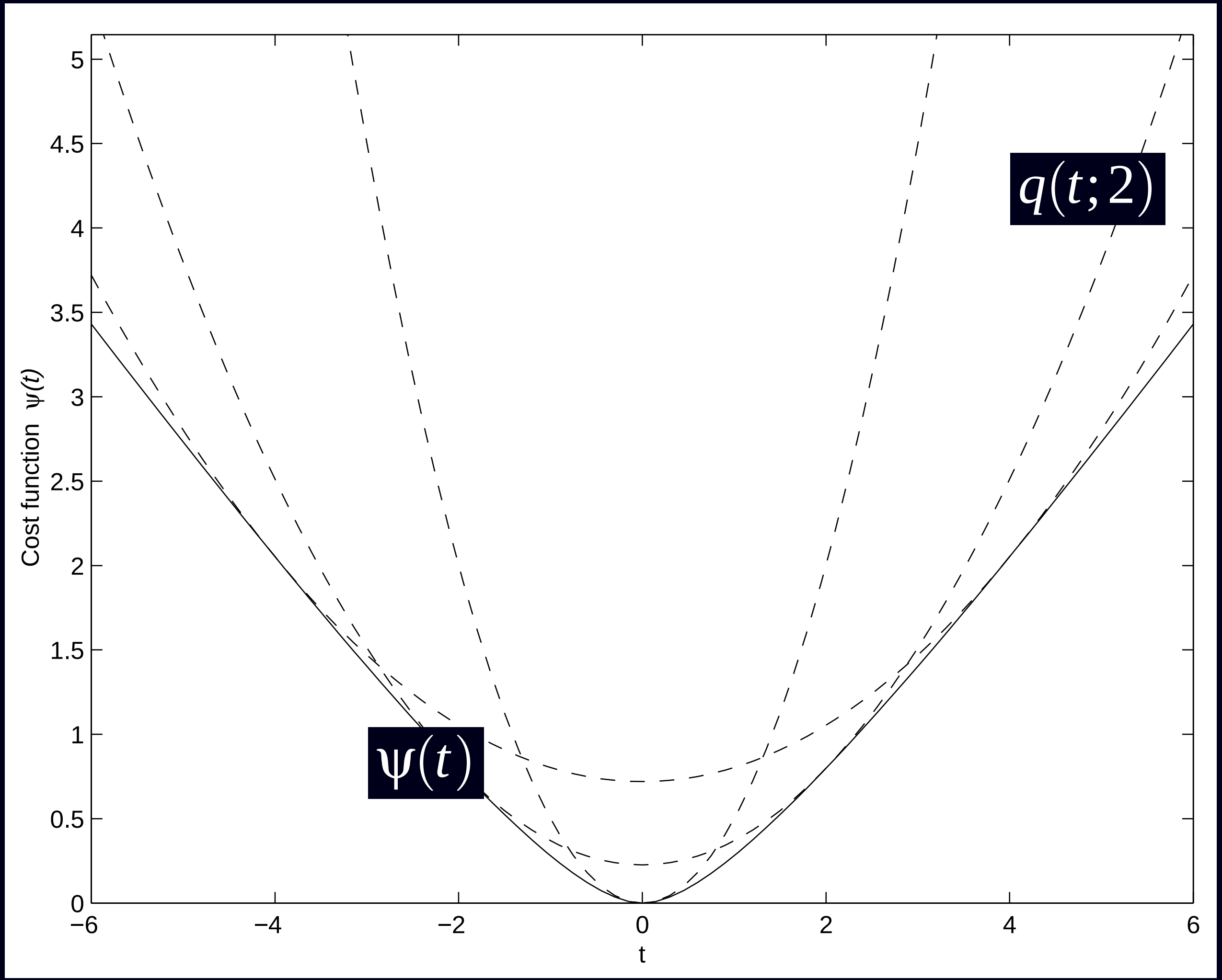
$$\psi_i(t) \leq q_i(t; t_i^{(n)}), \quad q_i(t; s) \triangleq \psi_i(s) + \dot{\psi}_i(s)(t - s) + \kappa_i(s)\frac{(t-s)^2}{2}.$$

Optimal parabola curvature (for fastest convergence rate):

$$\kappa_i(s) \triangleq \min\{\kappa \geq 0 \,:\, q_i(t; s) \geq \psi_i(t) \;\forall t\}.$$

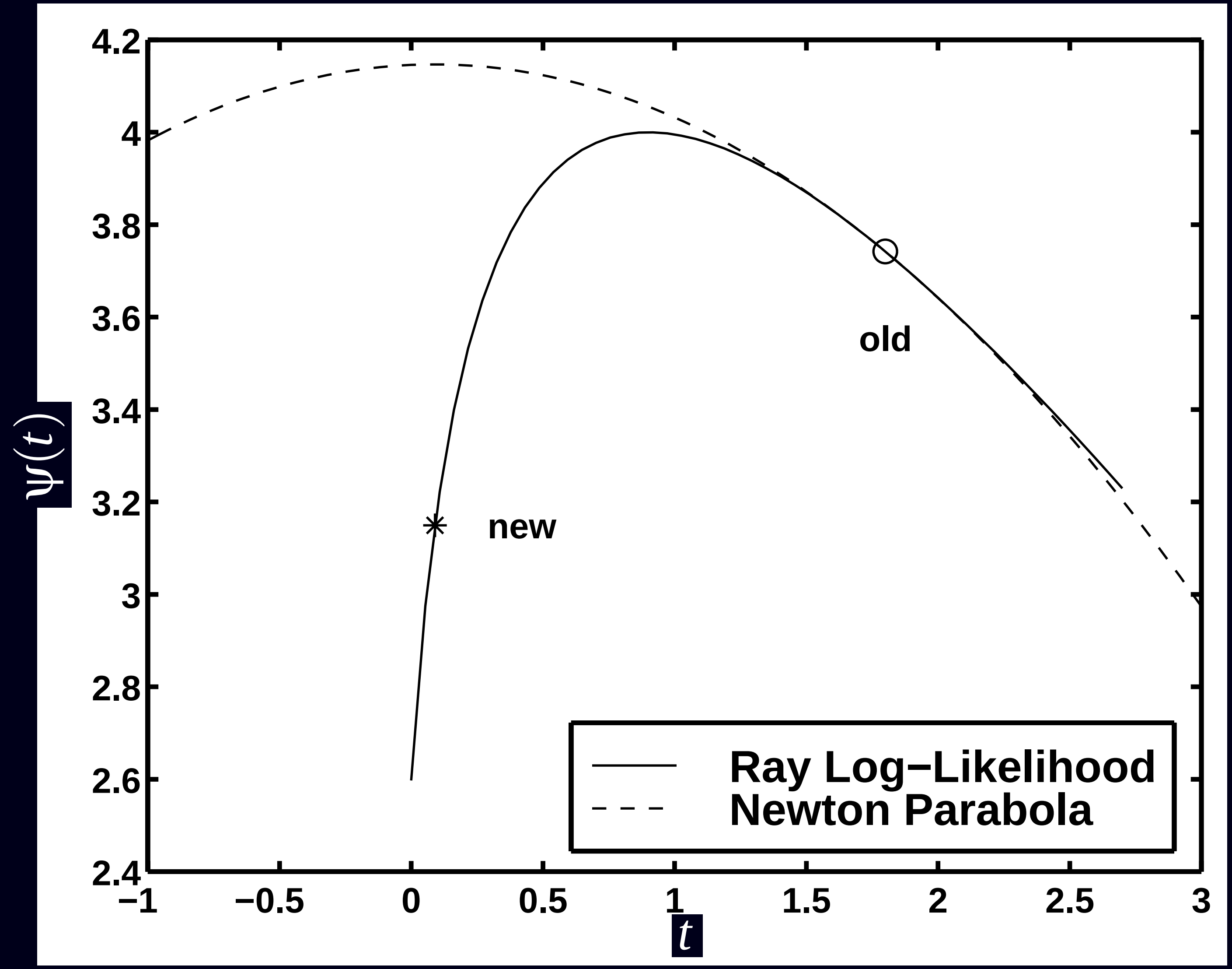
For Huber-like functions: $\kappa_i(s) = \dot{\psi}_i(s)/s \triangleq \omega_i(s)$.
For emission and transmission tomography, optimal κ_i derived by Erdoğan (Tr. Med. Im., 1999)

Tangent Parabolas



$\omega_\psi(t_0)$ is the curvature of the parabola that is tangent at t_0

Nonmonotonicity of Newton-Raphson



Minimizing the Paraboloidal Surrogate (M-step)

$$\phi(\mathbf{x};\mathbf{x}^{(n)}) = c_0 + \nabla\Phi(\mathbf{x}^{(n)})(\mathbf{x}-\mathbf{x}^{(n)}) - \frac{1}{2}(\mathbf{x}-\mathbf{x}^{(n)})'\mathbf{B}'\,\mathrm{diag}\left\{\kappa_i^{(n)}\right\}\mathbf{B}(\mathbf{x}-\mathbf{x}^{(n)}),$$

where the tangent parabola curvatures are:

$$\kappa_i^{(n)} = \kappa_i(t_i^{(n)}) = \kappa_i([\mathbf{B}\mathbf{x}^{(n)} - \mathbf{c}]_i).$$

M-step: Minimize $\phi(\mathbf{x};\mathbf{x}^{(n)})$ using any iterative “least squares” algorithm that accommodates nonnegativity constraints.

Reasonable choices of algorithms

- PSCD Paraboloidal surrogates coordinate descent:
  fast converging, but non-parallelizable
- SPS (separable paraboloidal surrogates):
  slow converging, but fully parallelizable
- PPCD (partitioned-separable paraboloidal surrogate coordinate descent)
  best of both worlds?

Paraboloidal surrogates coordinate descent (PSCD)

- Update one pixel at a time, w.r.t. the surrogate, holding other pixels fixed:

$$x_j^{(n+1)} = \arg\min_{x_j \geq 0} \phi(x_1^{(n+1)}, \ldots, x_{j-1}^{(n+1)}, x_j, x_{j+1}^{(n)}, \ldots, x_{n_p}^{(n)}; \mathbf{x}^{(n)}).$$

- Cycle through all pixels, then update the paraboloidal surrogate ($\kappa_i^{(n)}$'s).

Advantages:

- Intrinsically monotonic, global convergence (for a broad family of ψ_i's)
- Fast converging (from good initial image)
- Nonnegativity constraint trivial

Disadvantages:

- Requires column access of system matrix
- Poorly parallelizable

Separable paraboloid surrogate

One can use the convexity of the paraboloidal surrogate ϕ to define a second surrogate function that is **separable**:

$$\phi(\mathbf{x};\mathbf{x}^{(n)}) \leq \phi^{SP}(\mathbf{x};\mathbf{x}^{(n)}) \triangleq \sum_{j=1}^{n_p} \phi_j(x_j - x_j^{(n)};\mathbf{x}^{(n)})$$

where

$$\phi_j(t;\mathbf{x}^{(n)}) \triangleq \sum_{i=1}^{N} \pi_{ij}\kappa_i^{(n)} \frac{1}{2}\left(\frac{b_{ij}}{\pi_{ij}}t + [\mathbf{B}\mathbf{x}^{(n)} - \mathbf{c}]_i\right)^2,$$

$$\pi_{ij} = \frac{|b_{ij}|}{\sum_{j=1}^{n_p} |b_{ik}|}.$$

Minimizing the separable paraboloid ϕ^{SP} is trivial, especially compared to minimizing a paraboloid.

$$\mathbf{x}^{(n+1)} = \arg\min_{\mathbf{x}\geq\mathbf{0}} \phi(\mathbf{x};\mathbf{x}^{(n)}) \quad \Rightarrow \quad x_j^{(n+1)} = \arg\min_{x_j\geq\mathbf{0}} \phi_j(x_j - x_j^{(n)};\mathbf{x}^{(n)}), \; j = 1,\ldots,n_p.$$

Separable paraboloid surrogate (SPS) algorithm

$$x_j^{(n+1)} = \left[x_j^{(n)} - \frac{\dot{q}_j(0; \mathbf{x}^{(n)})}{\ddot{q}_j(0; \mathbf{x}^{(n)})} \right]_+ \quad \Rightarrow \quad \mathbf{x}^{(n+1)} = \left[\mathbf{x}^{(n)} - \mathrm{diag}\left\{ \frac{1}{\ddot{q}_j(0; \mathbf{x}^{(n)})} \right\} \nabla \Phi(\mathbf{x}^{(n)}) \right]_+$$

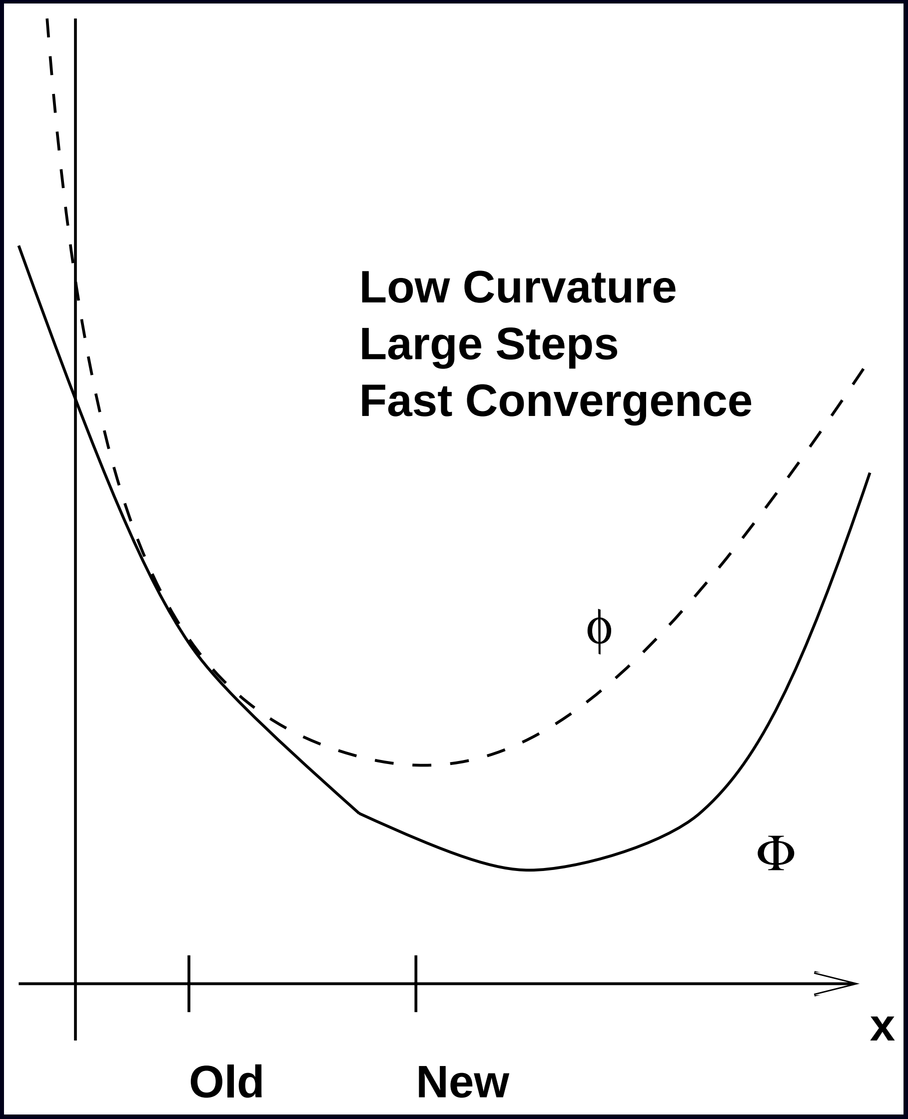
Advantages:

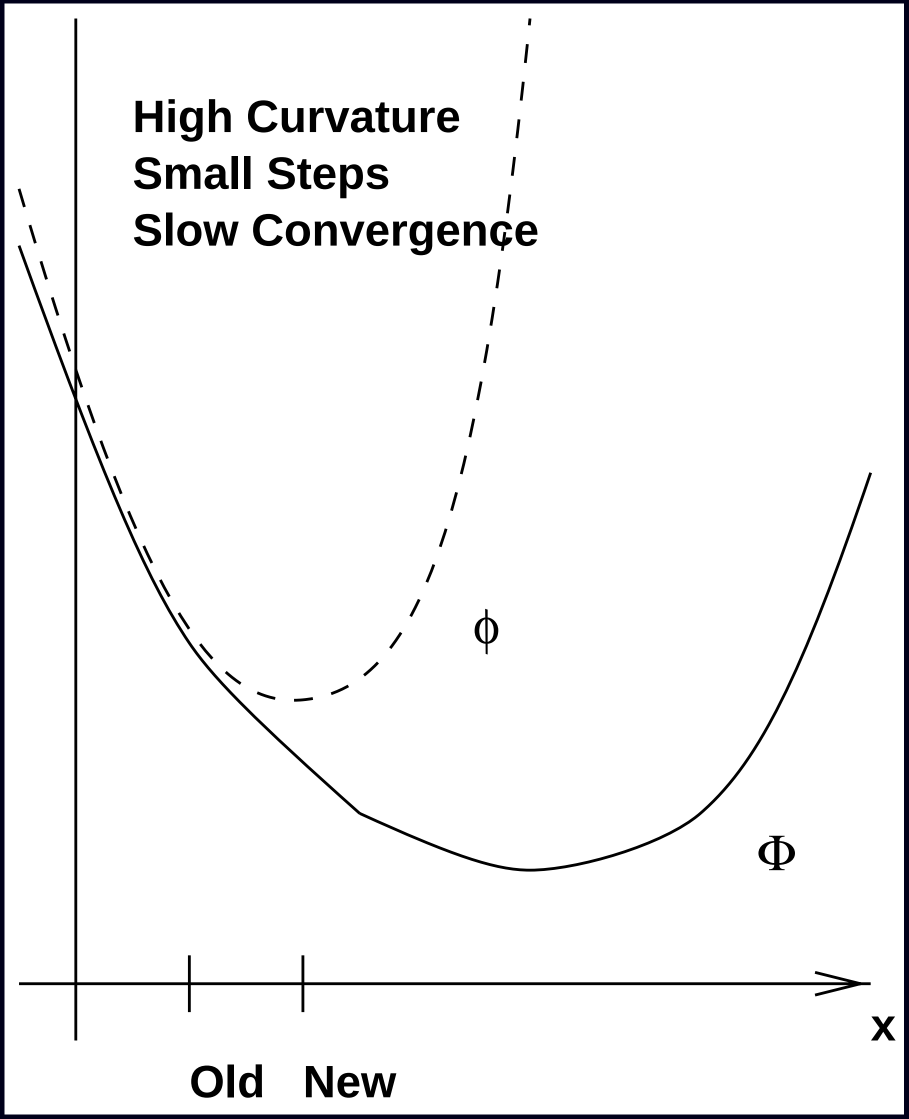
- Monotonically decreases Φ
- Converges globally to unique minimizer (for broad family of convex ψ_i's)
- No matrix inversion required
- Easily enforces nonnegativity constraint
- Completely parallelizable (all pixels updated simultaneously)

Disadvantages:

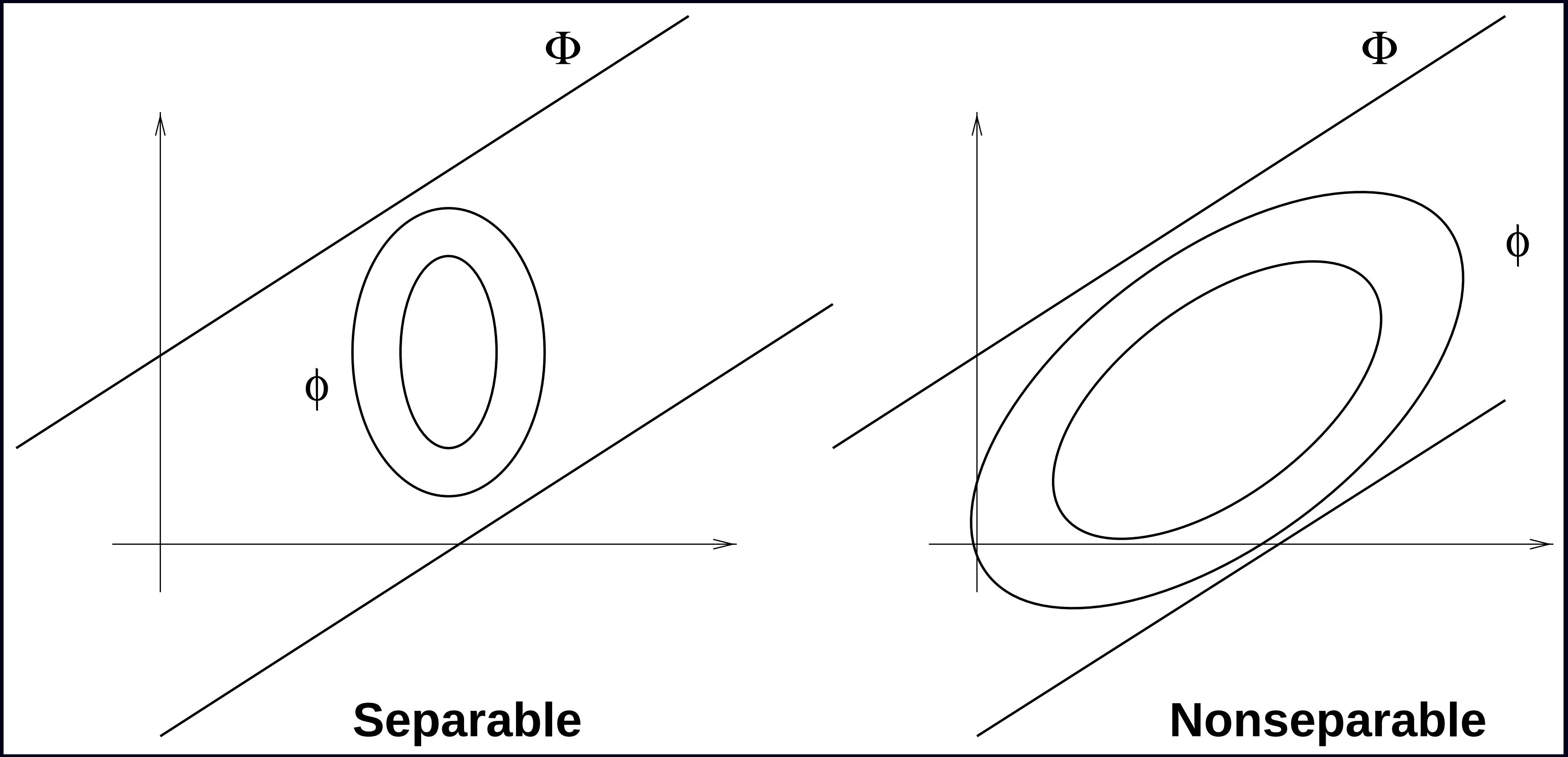
- Very slow convergence (ala EM algorithm)

Convergence Rate / Surrogate Curvature



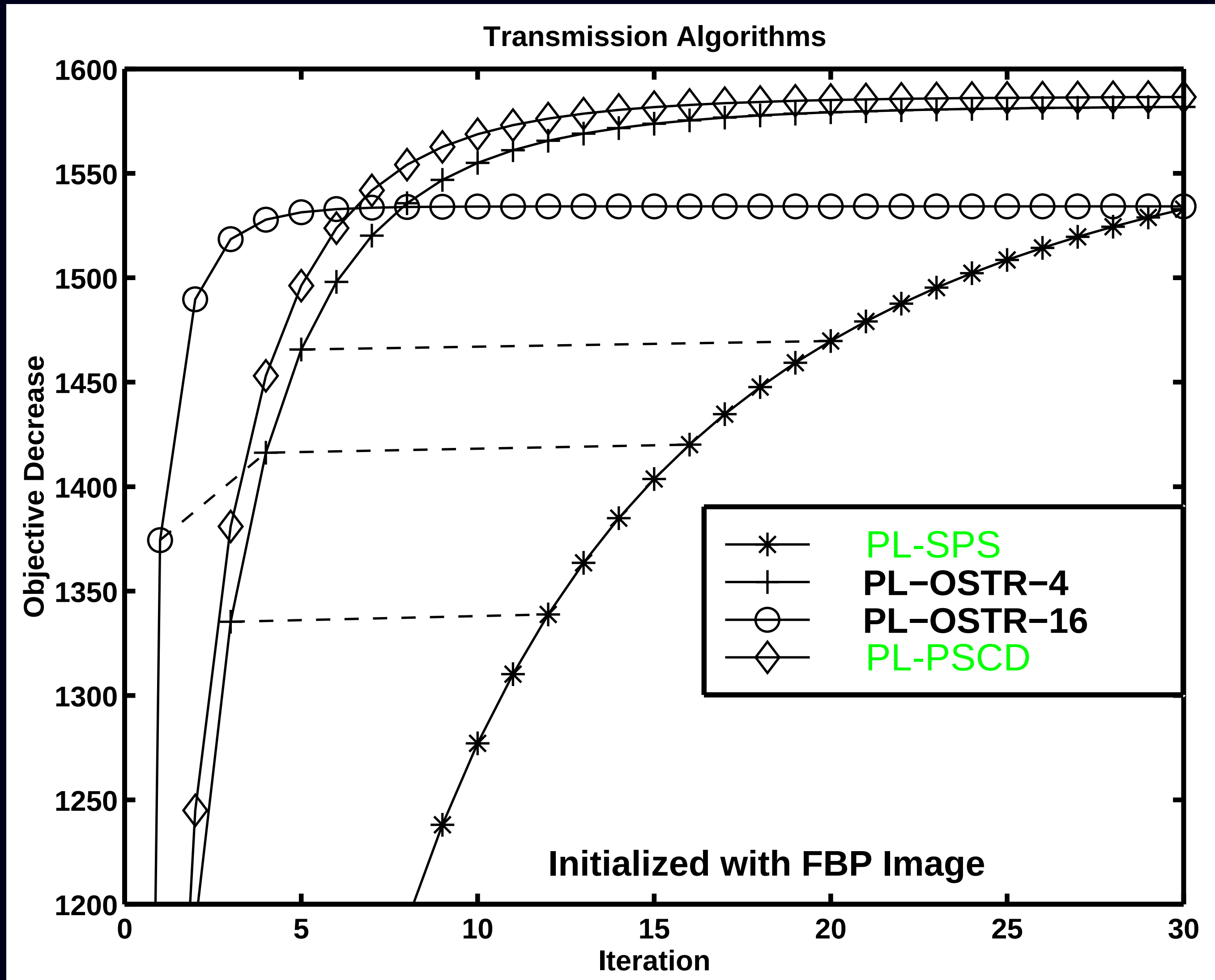


Separable vs Nonseparable Surrogates



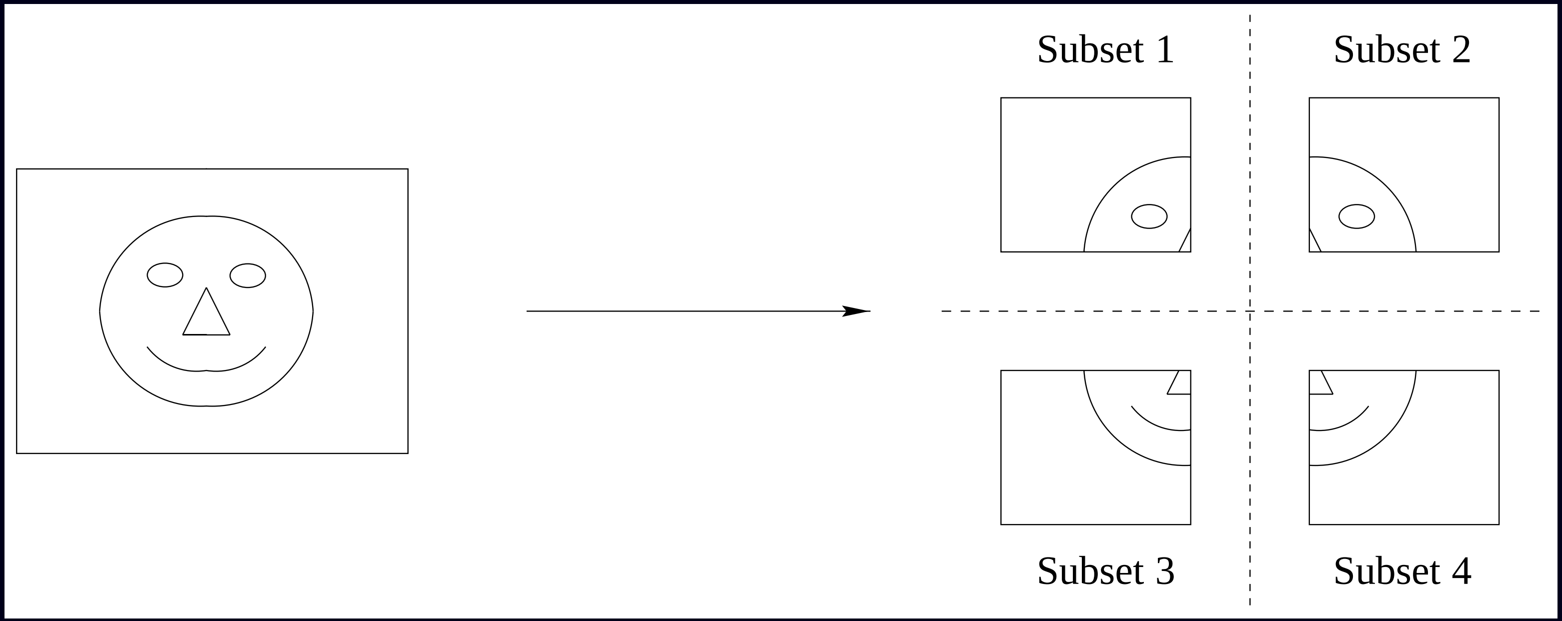
Separable surrogates (*e.g.* EM) have high curvature ∴ slow convergence.
Nonseparable surrogates can have lower curvature ∴ faster convergence.
Harder to minimize? Use paraboloids (quadratic surrogates).

PSCD vs SPS Algorithm



Naive Parallelizable Coordinate Descent

- Goal: fast convergence of coordinate descent, yet parallelizable
- Suitable for coarse-grain parallelization



- Each processor applies coordinate descent independently to its block
- Not guaranteed to be monotonic!

Partitioned-separable paraboloidal surrogate

Partion pixels into K subsets indexed by J_k, where $\bigcup_{k=1}^{K} J_k = \{1, 2, \ldots, n_p\}$.

E-step: Form a surrogate function that is separable *between blocks*:

$$\Phi(\mathbf{x}) \leq \phi(\mathbf{x}; \mathbf{x}^{(n)}) \leq Q(\mathbf{x}; \mathbf{x}^{(n)}) = \sum_{k=1}^{K} Q_k(\mathbf{x}_{J_k}; \mathbf{x}^{(n)}).$$

By construction, decreasing this new surrogate $Q(\mathbf{x}; \mathbf{x}^{(n)})$ will monotonically decrease the cost function Φ:

$$\Phi(\mathbf{x}^{(n)}) - \Phi(\mathbf{x}) \geq Q(\mathbf{x}^{(n)}; \mathbf{x}^{(n)}) - Q(\mathbf{x}; \mathbf{x}^{(n)}).$$

M-step: Partitioned-separable form allows processor parallelization:

$$\mathbf{x}^{(n+1)} = \arg\min_{\mathbf{x} \geq \mathbf{0}} Q(\mathbf{x}; \mathbf{x}^{(n)}) \quad \Rightarrow \quad \mathbf{x}_{J_k}^{(n+1)} = \arg\min_{\mathbf{x}_{J_k} \geq \mathbf{0}} Q_k(\mathbf{x}_{J_k}; \mathbf{x}^{(n)}), \; k = 1, \ldots, K.$$

PPCD Derivation

Adaptation of De Pierro's convexity trick for modified EM algorithm:

$$[\mathbf{B}\mathbf{x}-\mathbf{c}]_i = \sum_{j=1}^{n_p} b_{ij}x_j - c_i = \sum_{k=1}^{K} \pi_{ik}\left(\frac{s_{ik}^{(n)}(\mathbf{x}_{J_k})}{\pi_{ik}} + t_i^{(n)}\right)$$

$$\text{where } \pi_{ik} = \frac{\sum_{j\in J_k}|b_{ij}|}{\sum_{j=1}^{n_p}|b_{ij}|} \geq 0 \text{ and } \sum_{k=1}^{K}\pi_{ik} = 1,$$

$$s_{ik}^{(n)} = [\mathbf{B}_{J_k}(\mathbf{x}_{J_k} - \mathbf{x}_{J_k}^{(n)}) - \mathbf{c}]_i = \sum_{j\in J_k} b_{ij}(x_j - x_j^{(n)}) - c_i.$$

When q is a quadratic (and hence convex function):

$$q([\mathbf{B}\mathbf{x}-\mathbf{c}]_i) = q\left(\sum_{k=1}^{K}\pi_{ik}\left(\frac{s_{ik}^{(n)}(\mathbf{x}_{J_k})}{\pi_{ik}} + t_i^{(n)}\right)\right) \leq \sum_{k=1}^{K}\pi_{ik}\, q\left(\frac{s_{ik}^{(n)}(\mathbf{x}_{J_k})}{\pi_{ik}} + t_i^{(n)}\right).$$

The latter term is the foundation for Q_k, being partitioned separable.

Partitioned separable Paraboloidal-surrogate Coordinate Descent (PPCD) Algorithm

E-step

- Form paraboloidal surrogate $\phi(\mathbf{x};\mathbf{x}^{(n)})$ from cost function $\Phi(\mathbf{x})$
- Form partitioned separable surrogate $Q(\mathbf{x};\mathbf{x}^{(n)}) = \sum_{k=1}^{K} Q_k(\mathbf{x}_{J_k};\mathbf{x}^{(n)})$

M-step

- K processors independently "minimize" (or decrease) $\{Q_k(\mathbf{x}_{J_k};\mathbf{x}^{(n)})\}$, using any nonnegative least-squares method such as coordinate descent

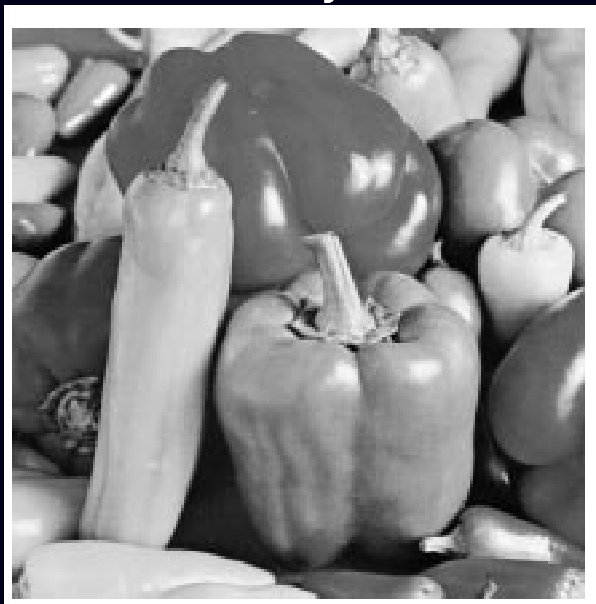
$$\mathbf{x}_{J_k}^{(n+1)} = \arg\min_{\mathbf{x}_{J_k} \geq \mathbf{0}} Q_k(\mathbf{x}_{J_k};\mathbf{x}^{(n)}), \; k = 1,\ldots,K.$$

- Broadcast $\mathbf{x}_{J_k}^{(n+1)}$ to other processors

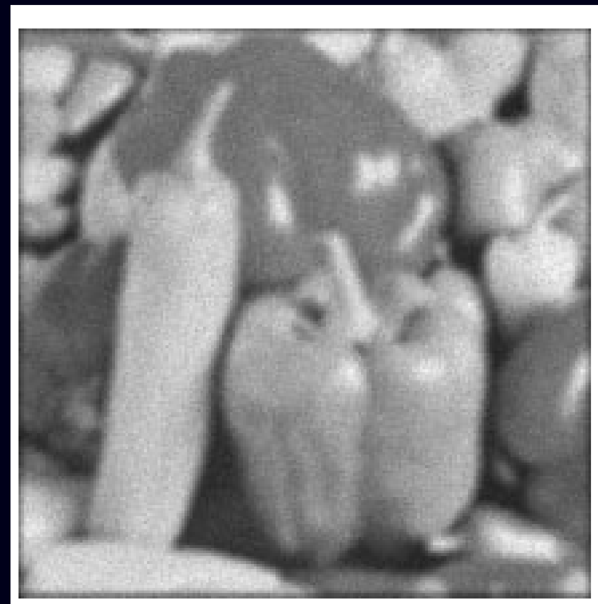
For a broad class of convex ψ_i's, global convergence follows from SAGE convergence proof, Fessler and Hero, 1995 (IEEE T-SP).

Representative 2D Simulation Results

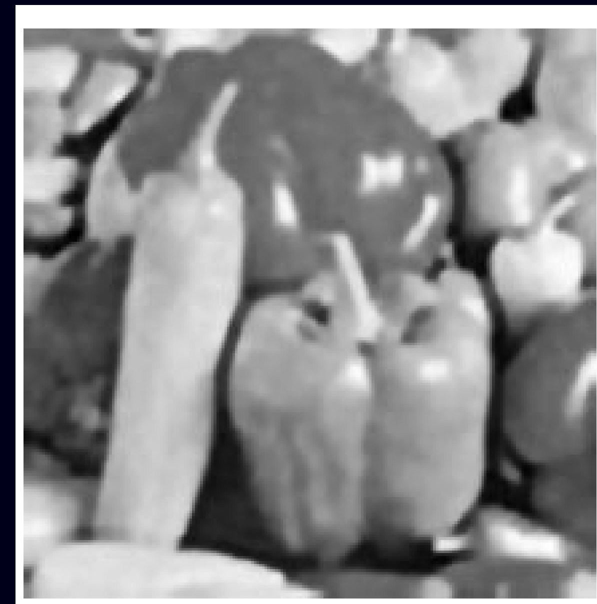
Object



Measurement



Restoration

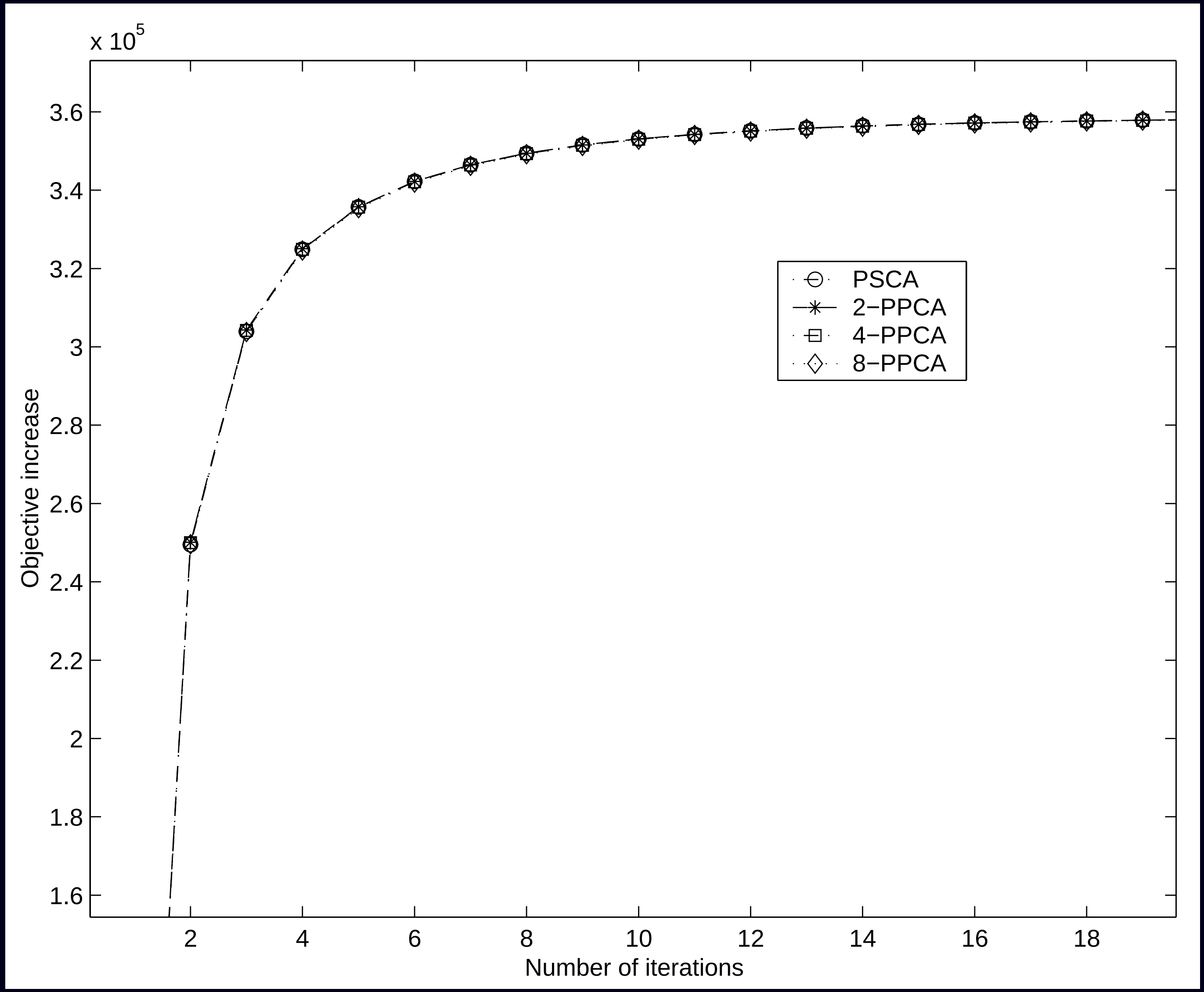


Poisson measurement noise with PSNR = 25 dB

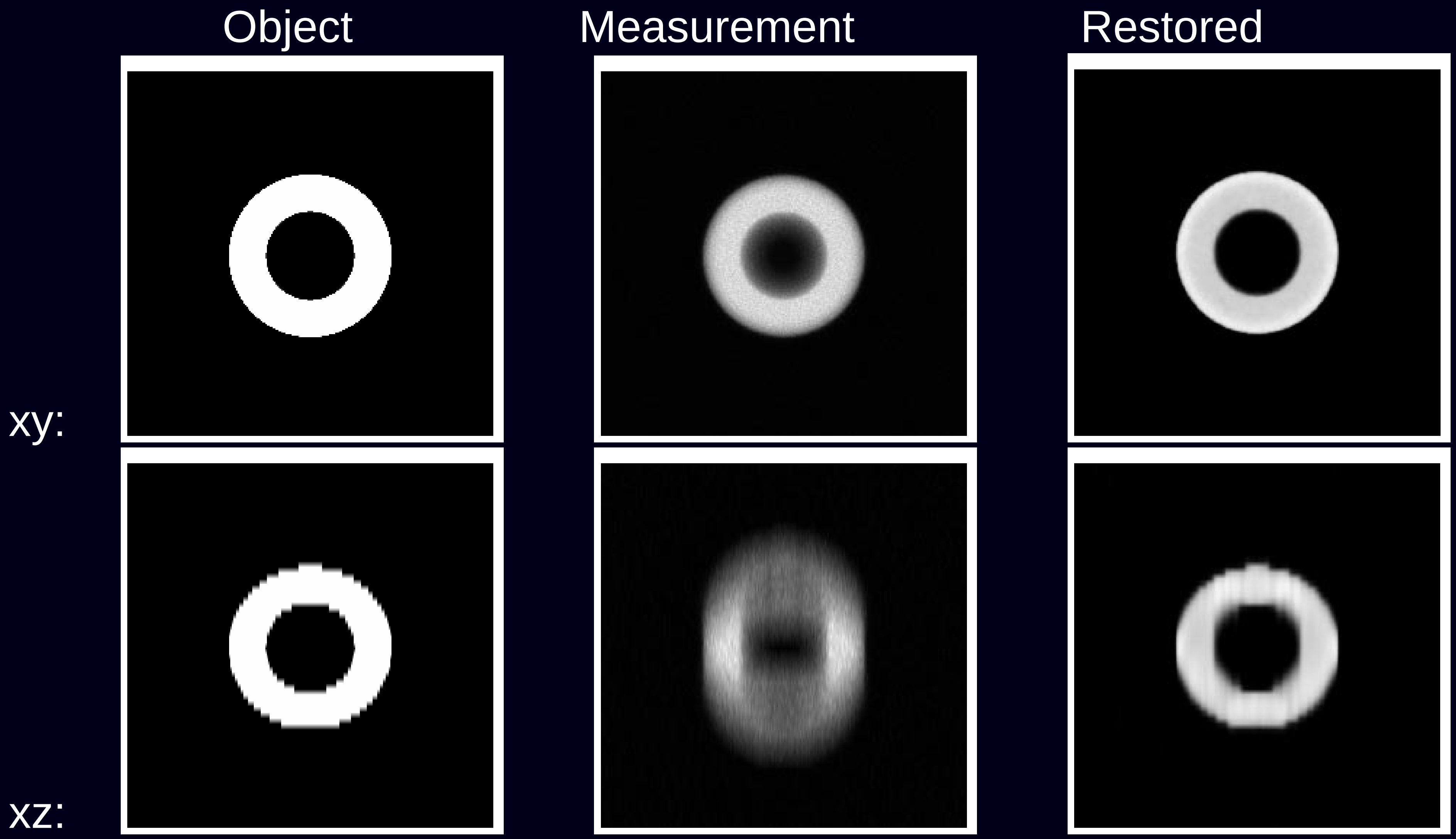
Restored by maximum penalized likelihood
using nonquadratic edge-preserving penalty function:

$$\psi(t) = \delta^2 \left[\left| \frac{t}{\delta} \right| - \log\left(1 + \left| \frac{t}{\delta} \right| \right) \right], \quad \text{with } \delta = 1.5.$$

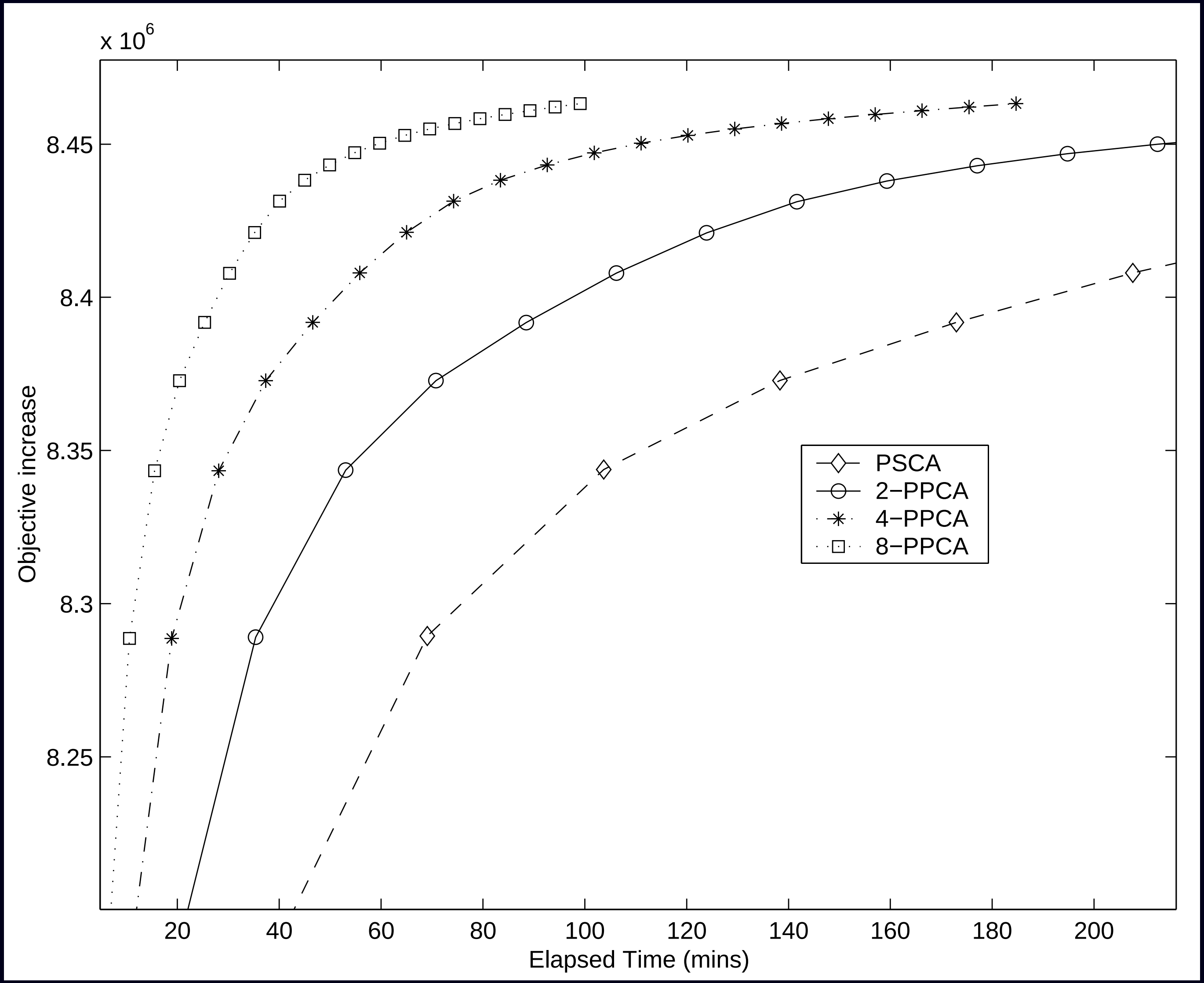
Convergence rate vs number of processors (2D)



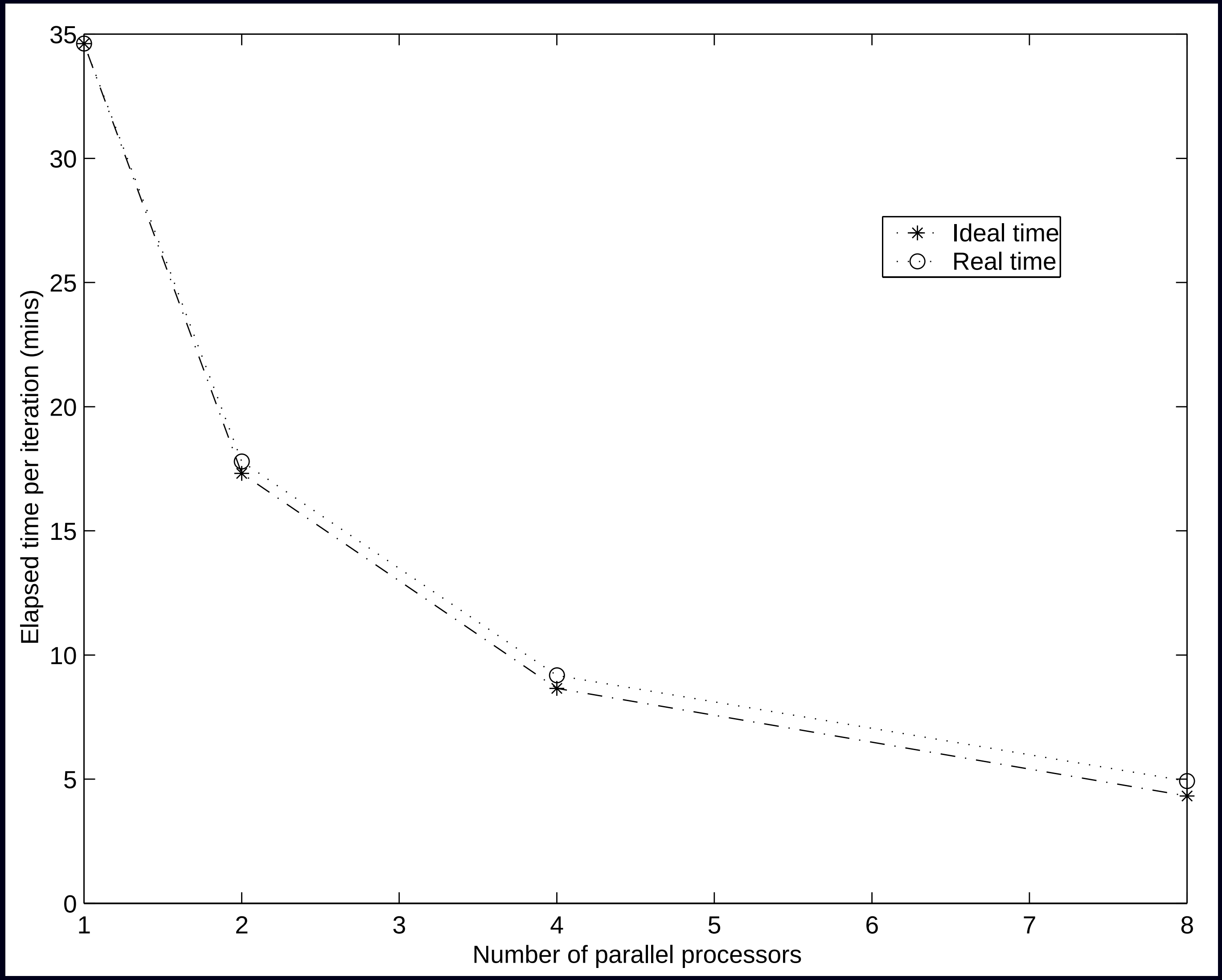
Confocal microscopy simulation (3D)



Cost function decrease vs Elapsed time



Elapsed time per iteration vs # of Processors



Summary

Optimization transfer

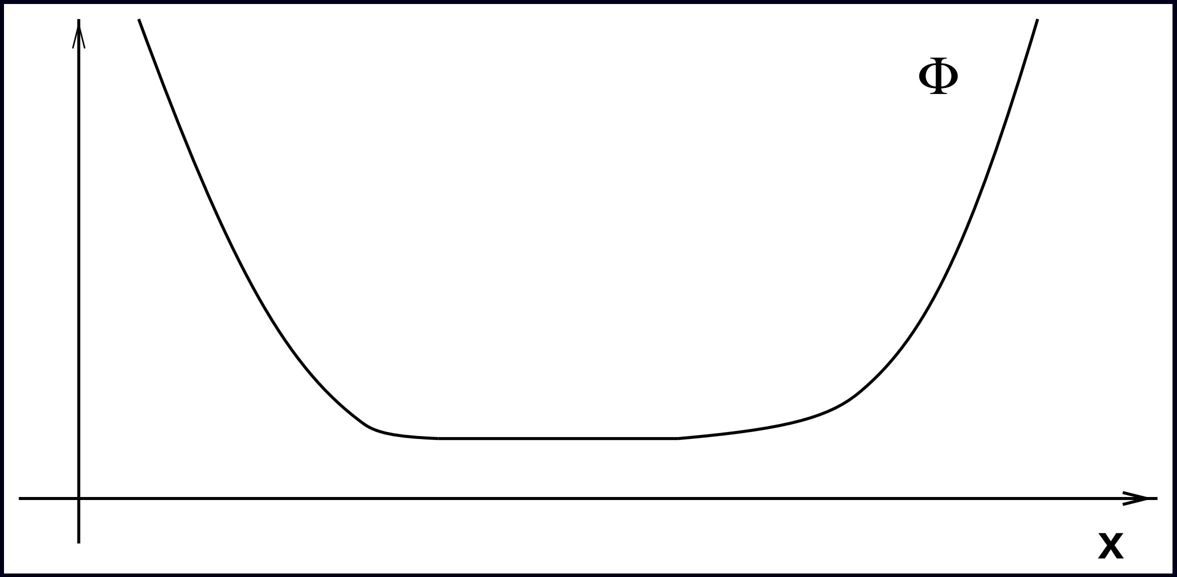
- Natural framework for algorithm development
- Exploits structure of “partially separable” cost functions

Partitioned separable paraboloidal surrogate coordinate descent algorithm

- Accommodates non-quadratic cost functions
- Monotonically decreases Φ
- Converges globally to unique minimizer (for broad class of ψ_i’s)
- Easily accommodates nonnegativity constraint
- Parallelizable
- Converges faster than a general-purpose optimization method

Future Work

Convergence proof for multiple minima:



Slides and paper available from:

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

Monotone Convergence

From R. Meyer "Sufficient conditions for the convergence of monotonic mathematical programming algorithms," J. Comput. System. Sci., 1976.

Let M be a point to set mapping such that, on G,

- M is uniformly compact,
- M is upper semi-continuous,
- M is strictly monotonic with respect to the function Φ.

If $\{\mathbf{x}^{(n)}\}$ is any sequence generated by the algorithm $\mathbf{x}^{(n+1)} \in M(\mathbf{x}^{(n)})$:

- all accumulation points of $\{\mathbf{x}^{(n)}\}$ will be fixed points,
- $\Phi(\mathbf{x}^{(n)}) \to \Phi(\mathbf{x}^{\star})$ where $\mathbf{x}^{\star}$ is a fixed point,
- $\|\mathbf{x}^{(n+1)} - \mathbf{x}^{(n)}\| \to 0$, and
- either $\{\mathbf{x}^{(n)}\}$ converges or the accumulation points of $\{\mathbf{x}^{(n)}\}$ form a continuum.