Part 3. Algorithms

Method = Cost Function + Algorithm

Outline

- Ideal algorithm
- Classical general-purpose algorithms
- Considerations:
  - nonnegativity
  - parallelization
  - convergence rate
  - monotonicity
- Algorithms tailored to *cost functions* for imaging
  - Optimization transfer
  - EM-type methods
  - Poisson emission problem
  - Poisson transmission problem
- Ordered-subsets / block-iterative algorithms

Why iterative algorithms?

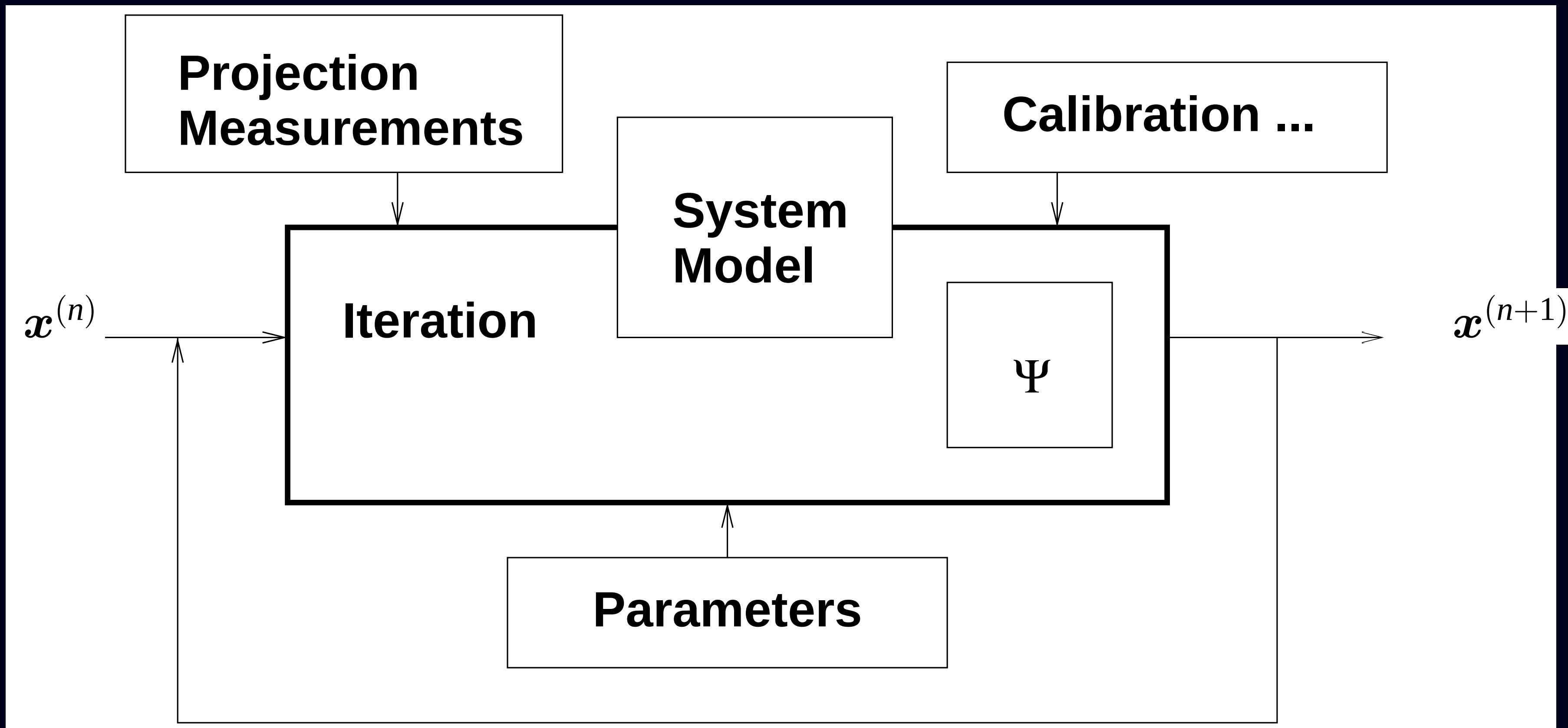
- For nonquadratic Ψ, no closed-form solution for minimizer.
- For quadratic Ψ with nonnegativity constraints, no closed-form solution.
- For quadratic Ψ without constraints, closed-form solutions:

$$\text{PWLS:} \quad \hat{\boldsymbol{x}} = [\boldsymbol{A}'\boldsymbol{W}\boldsymbol{A} + \boldsymbol{R}]^{-1}\boldsymbol{A}'\boldsymbol{W}\boldsymbol{y}$$
$$\text{OLS:} \quad \hat{\boldsymbol{x}} = [\boldsymbol{A}'\boldsymbol{A}]^{-1}\boldsymbol{A}'\boldsymbol{y}$$

Impractical (memory and computation) for realistic problem sizes.
$\boldsymbol{A}$ is sparse, but $\boldsymbol{A}'\boldsymbol{A}$ is not.

All algorithms are imperfect. No single best solution.

General Iteration



Deterministic iterative mapping: $\boldsymbol{x}^{(n+1)} = M(\boldsymbol{x}^{(n)})$

Ideal Algorithm

$$\boldsymbol{x}^\star \stackrel{\triangle}{=} \arg\min_{\boldsymbol{x} \geq \boldsymbol{0}} \Psi(\boldsymbol{x}) \qquad \text{(global minimizer)}$$

Properties

stable and convergent	$\{\boldsymbol{x}^{(n)}\}$ converges to $\boldsymbol{x}^\star$ if run indefinitely
converges quickly	$\{\boldsymbol{x}^{(n)}\}$ gets "close" to $\boldsymbol{x}^\star$ in just a few iterations
globally convergent	$\lim_n \boldsymbol{x}^{(n)}$ independent of starting image $\boldsymbol{x}^{(0)}$
fast	requires minimal computation per iteration
robust	insensitive to finite numerical precision
user friendly	nothing to adjust (*e.g.*, acceleration factors)
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

Classic Algorithms

Non-gradient based

- Exhaustive search
- Nelder-Mead simplex (amoeba)

Converge very slowly, but work with nondifferentiable *cost functions*.

Gradient based

- Gradient descent

$$\boldsymbol{x}^{(n+1)} \triangleq \boldsymbol{x}^{(n)} - \alpha \nabla \Psi(\boldsymbol{x}^{(n)})$$

Choosing α to ensure convergence is nontrivial.

- Steepest descent

$$\boldsymbol{x}^{(n+1)} \triangleq \boldsymbol{x}^{(n)} - \alpha_n \nabla \Psi(\boldsymbol{x}^{(n)}) \quad \text{where} \quad \alpha_n \triangleq \arg\min_{\alpha} \Psi\left(\boldsymbol{x}^{(n)} - \alpha \nabla \Psi(\boldsymbol{x}^{(n)})\right)$$

Computing α_n can be expensive.

Limitations

- Converge slowly.
- Do not easily accommodate nonnegativity constraint.

Gradients & Nonnegativity - A Mixed Blessing

Unconstrained optimization of differentiable *cost functions*:

$$\nabla\Psi(\boldsymbol{x}) = 0 \quad \text{when} \quad \boldsymbol{x} = \boldsymbol{x}^\star$$

- A necessary condition always.
- A sufficient condition for strictly convex *cost functions*.
- Iterations search for zero of gradient.

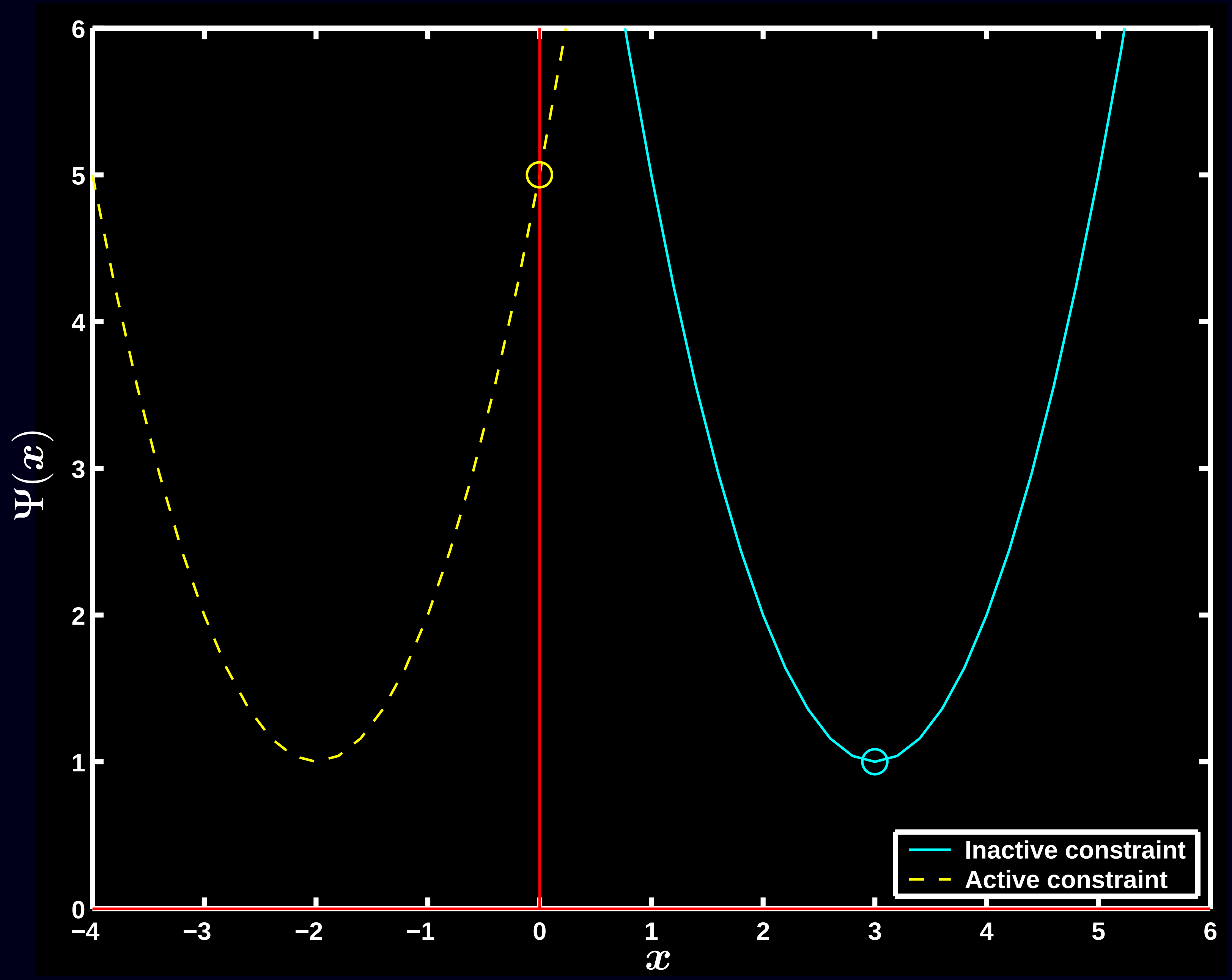
Nonnegativity-constrained minimization:

Karush-Kuhn-Tucker conditions

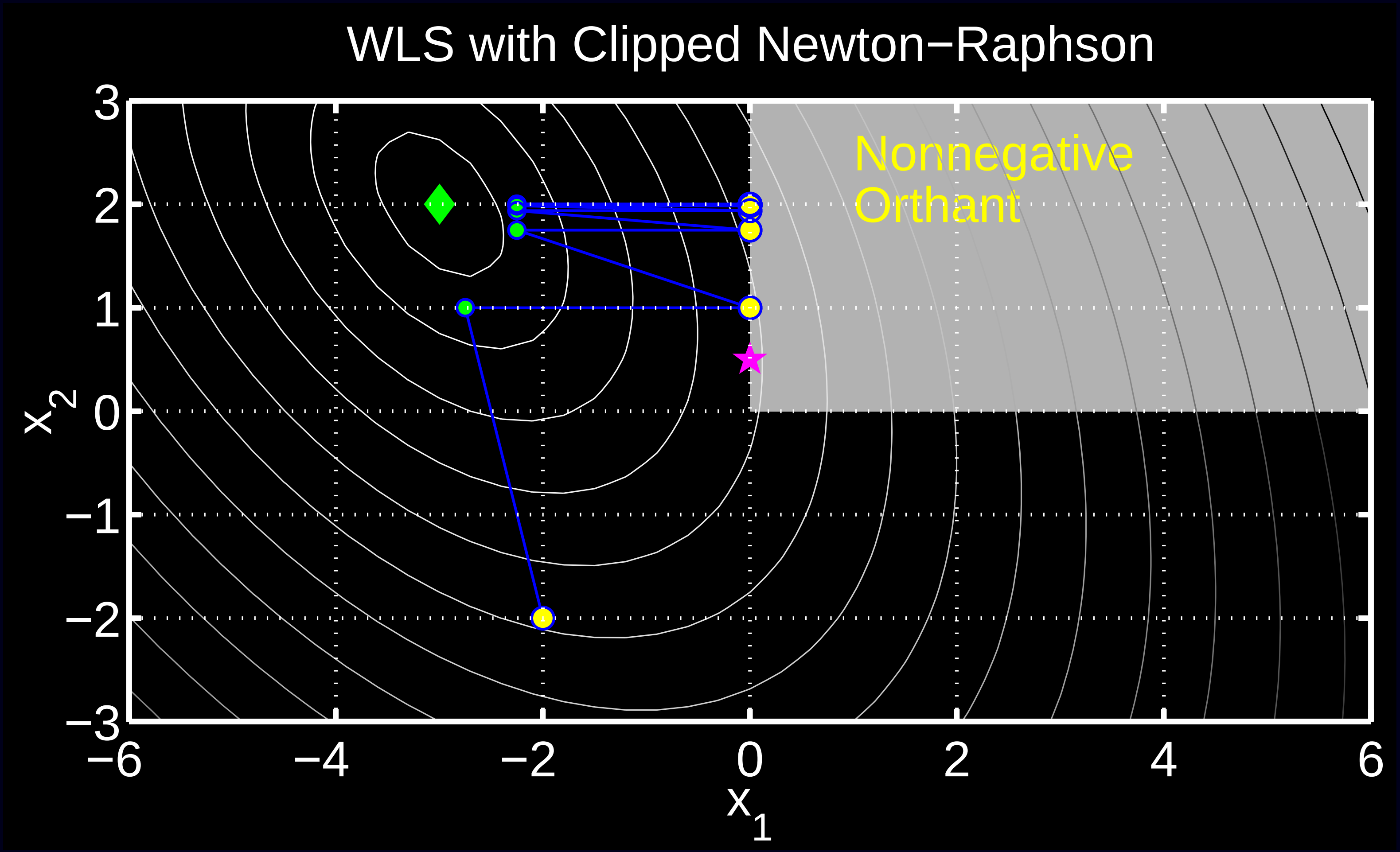
$$\left.\frac{\partial}{\partial x_j}\Psi(\boldsymbol{x})\right|_{\boldsymbol{x}=\boldsymbol{x}^\star} \quad \text{is} \quad \begin{cases} = 0, & x_j^\star > 0 \\ \geq 0, & x_j^\star = 0 \end{cases}$$

- A necessary condition always.
- A sufficient condition for strictly convex *cost functions*.
- Iterations search for ???
- $0 = x_j^\star \frac{\partial}{\partial x_j}\Psi(\boldsymbol{x}^\star)$ is a necessary condition, but never sufficient condition.

Karush-Kuhn-Tucker Illustrated



Why Not Clip Negatives?



Newton-Raphson with negatives set to zero each iteration.
Fixed-point of iteration is not the constrained minimizer!

Newton-Raphson Algorithm

$$\boxed{\boldsymbol{x}^{(n+1)} = \boldsymbol{x}^{(n)} - [\nabla^2 \Psi(\boldsymbol{x}^{(n)})]^{-1} \nabla \Psi(\boldsymbol{x}^{(n)})}$$

Advantage:
- Super-linear convergence rate (if convergent)

Disadvantages:
- Requires twice-differentiable Ψ
- Not guaranteed to converge
- Not guaranteed to monotonically decrease Ψ
- Does not enforce nonnegativity constraint
- Impractical for image recovery due to matrix inverse

General purpose remedy: bound-constrained Quasi-Newton algorithms

Newton's Quadratic Approximation

2nd-order Taylor series:

$$\Psi(\boldsymbol{x}) \approx \phi(\boldsymbol{x}; \boldsymbol{x}^{(n)}) \triangleq \Psi(\boldsymbol{x}^{(n)}) + \nabla\Psi(\boldsymbol{x}^{(n)})(\boldsymbol{x} - \boldsymbol{x}^{(n)}) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^{(n)})^T \nabla^2\Psi(\boldsymbol{x}^{(n)})(\boldsymbol{x} - \boldsymbol{x}^{(n)})$$

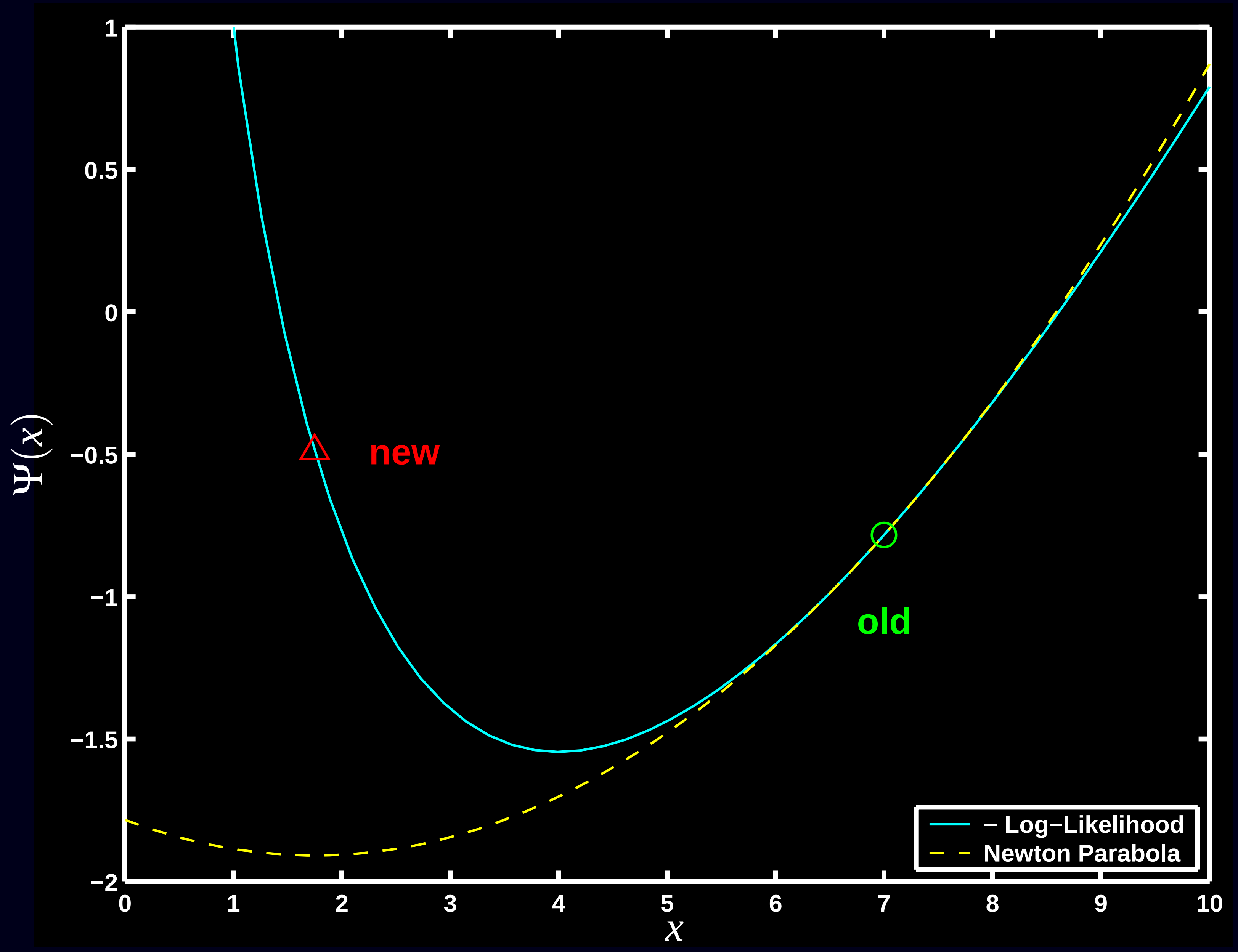
Set $\boldsymbol{x}^{(n+1)}$ to the ("easily" found) minimizer of this quadratic approximation:

$$\begin{aligned} \boldsymbol{x}^{(n+1)} &\triangleq \arg\min_{\boldsymbol{x}} \phi(\boldsymbol{x}; \boldsymbol{x}^{(n)}) \\ &= \boldsymbol{x}^{(n)} - [\nabla^2\Psi(\boldsymbol{x}^{(n)})]^{-1} \nabla\Psi(\boldsymbol{x}^{(n)}) \end{aligned}$$

Can be nonmonotone for Poisson emission tomography log-likelihood, even for a single pixel and single ray:

$$\Psi(x) = (x + r) - y \log(x + r)$$

Nonmonotonicity of Newton-Raphson



Consideration: Monotonicity

An algorithm is monotonic if

$$\Psi(\boldsymbol{x}^{(n+1)}) \leq \Psi(\boldsymbol{x}^{(n)}), \quad \forall \boldsymbol{x}^{(n)}.$$

Three categories of algorithms:

- Nonmonotonic (or unknown)
- Forced monotonic (*e.g.*, by line search)
- Intrinsically monotonic (by design, simplest to implement)

Forced monotonicity

Most nonmonotonic algorithms can be converted to forced monotonic algorithms by adding a line-search step:

$$\boldsymbol{x}^{\text{temp}} \triangleq M(\boldsymbol{x}^{(n)}), \quad \boldsymbol{d} = \boldsymbol{x}^{\text{temp}} - \boldsymbol{x}^{(n)}$$

$$\boldsymbol{x}^{(n+1)} \triangleq \boldsymbol{x}^{(n)} - \alpha_n \boldsymbol{d}^{(n)} \text{ where } \alpha_n \triangleq \arg\min_{\alpha} \Psi\left(\boldsymbol{x}^{(n)} - \alpha \boldsymbol{d}^{(n)}\right)$$

Inconvenient, sometimes expensive, nonnegativity problematic.

Conjugate Gradient Algorithm

Advantages:
- Fast converging (if suitably preconditioned) (in unconstrained case)
- Monotonic (forced by line search in nonquadratic case)
- Global convergence (unconstrained case)
- Flexible use of system matrix $\boldsymbol{A}$ and tricks
- Easy to implement in unconstrained quadratic case
- Highly parallelizable

Disadvantages:
- Nonnegativity constraint awkward (slows convergence?)
- Line-search awkward in nonquadratic cases

Highly recommended for unconstrained quadratic problems (*e.g.*, PWLS without nonnegativity). Useful (but perhaps not ideal) for Poisson case too.

Consideration: Parallelization

Simultaneous (fully parallelizable)
update all pixels simultaneously using all data
EM, Conjugate gradient, ISRA, OSL, SIRT, MART, ...

Block iterative (ordered subsets)
update (nearly) all pixels using one subset of the data at a time
OSEM, RBBI, ...

Row action
update many pixels using a single ray at a time
ART, RAMLA

Pixel grouped (multiple column action)
update some (but not all) pixels simultaneously a time, using all data
Grouped coordinate descent, multi-pixel SAGE
(Perhaps the most nontrivial to implement)

Sequential (column action)
update one pixel at a time, using all (relevant) data
Coordinate descent, SAGE

Coordinate Descent Algorithm

aka Gauss-Siedel, successive over-relaxation (SOR), iterated conditional modes (ICM)

Update one pixel at a time, holding others fixed to their most recent values:

$$x_j^{\text{new}} = \arg\min_{x_j \geq 0} \Psi(x_1^{\text{new}}, \ldots, x_{j-1}^{\text{new}}, x_j, x_{j+1}^{\text{old}}, \ldots, x_{n_p}^{\text{old}}), \qquad j = 1, \ldots, n_p$$

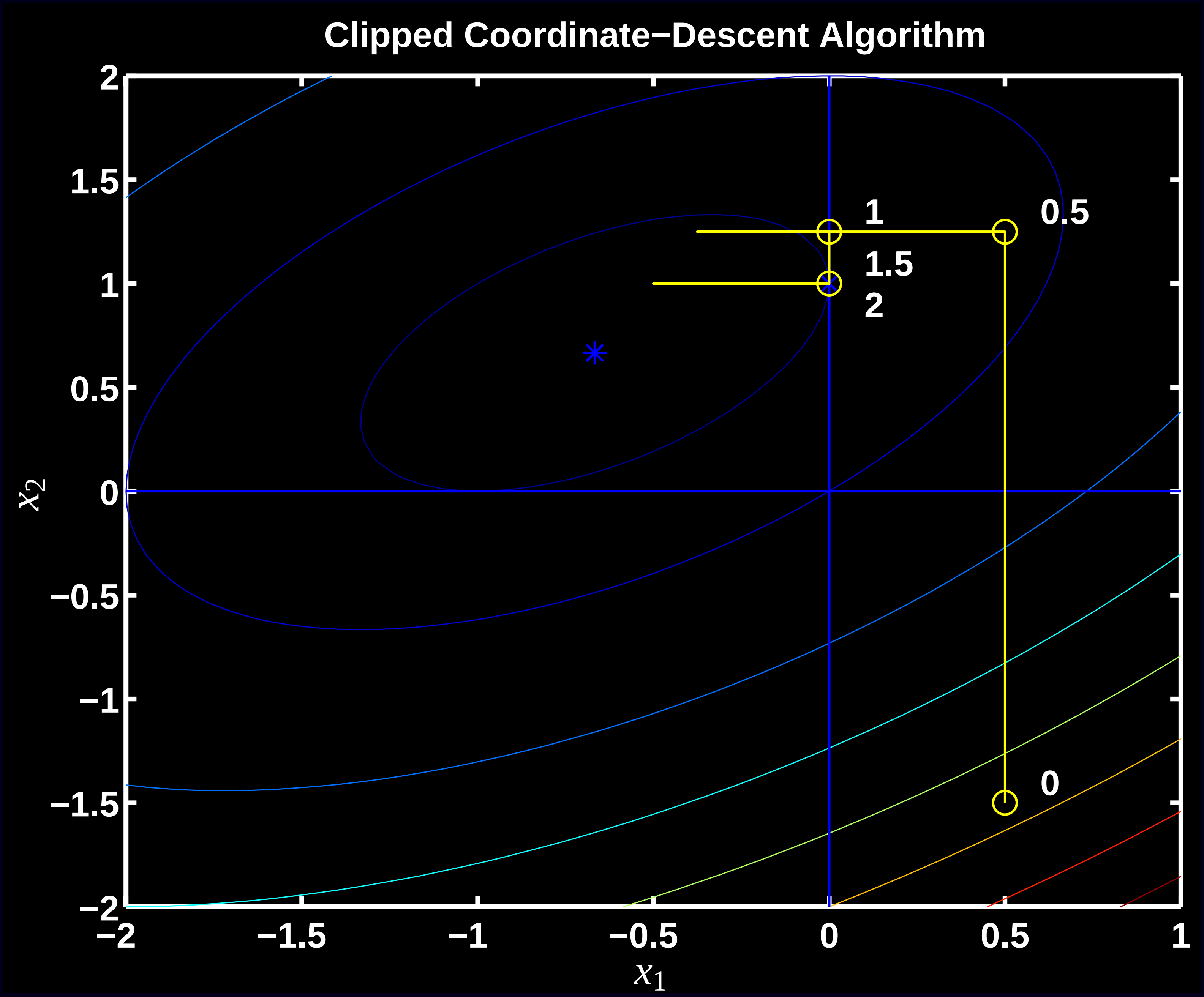
Advantages:

- Intrinsically monotonic
- Fast converging (from good initial image)
- Global convergence
- Nonnegativity constraint trivial

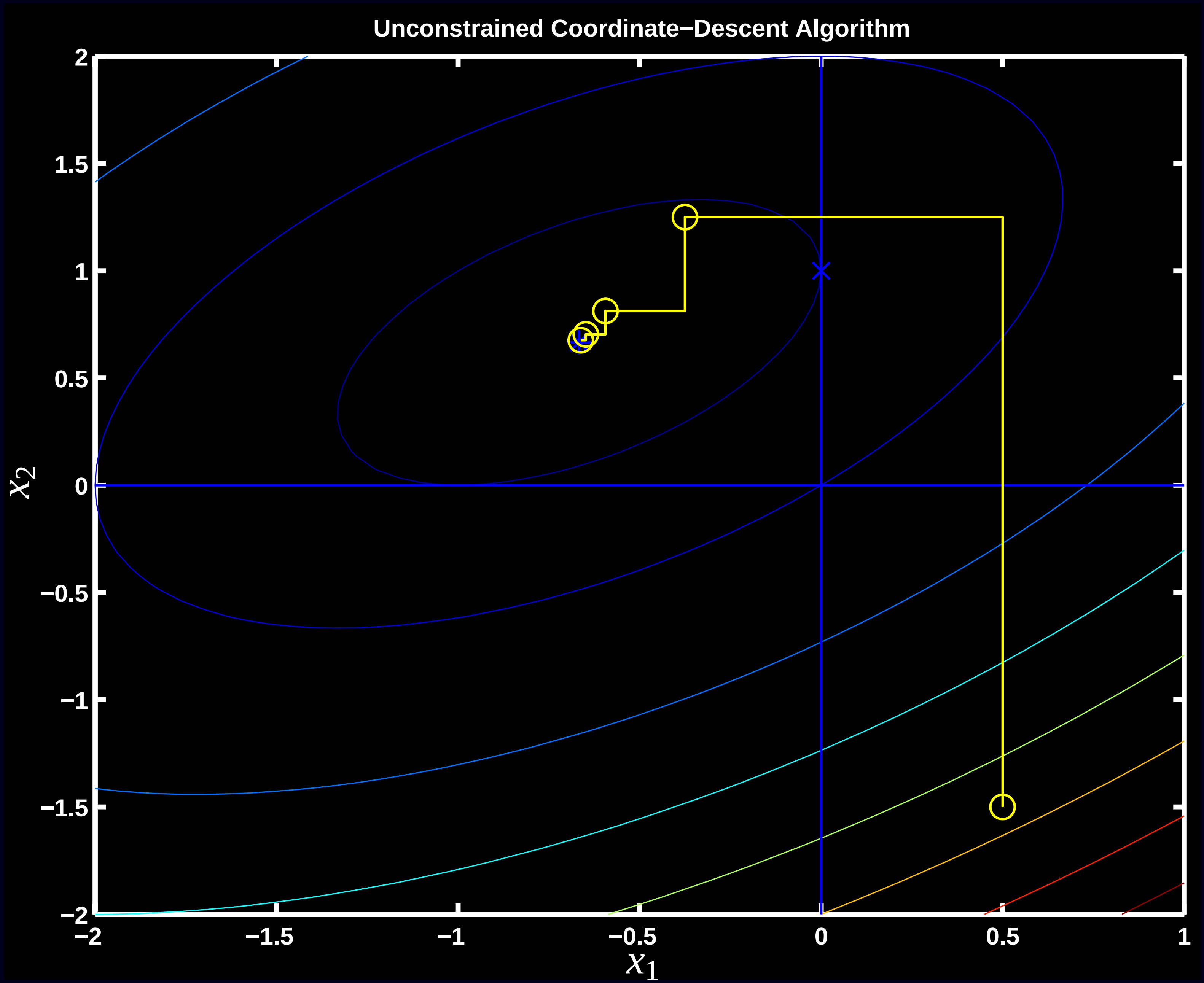
Disadvantages:

- Requires column access of system matrix $\boldsymbol{A}$
- Cannot exploit some "tricks" for $\boldsymbol{A}$
- Expensive "arg min" for nonquadratic problems
- Poorly parallelizable

Constrained Coordinate Descent Illustrated



Coordinate Descent - Unconstrained



Coordinate-Descent Algorithm Summary

Recommended when all of the following apply:

- quadratic or nearly-quadratic convex *cost function*
- nonnegativity constraint desired
- precomputed and stored system matrix $\boldsymbol{A}$ with column access
- parallelization not needed (standard workstation)

Cautions:

- Good initialization (*e.g.*, properly scaled FBP) essential.
  (Uniform image or zero image cause slow initial convergence.)
- Must be programmed carefully to be efficient.
  (Standard Gauss-Siedel implementation is suboptimal.)
- Updates high-frequencies fastest $\Rightarrow$ poorly suited to unregularized case

Used daily in UM clinic for 2D SPECT / PWLS / nonuniform attenuation

Summary of General-Purpose Algorithms

Gradient-based

- Fully parallelizable
- Inconvenient line-searches for nonquadratic *cost functions*
- Fast converging in unconstrained case
- Nonnegativity constraint inconvenient

Coordinate-descent

- Very fast converging
- Nonnegativity constraint trivial
- Poorly parallelizable
- Requires precomputed/stored system matrix

CD is well-suited to moderate-sized 2D problem (*e.g.*, 2D PET),
but poorly suited to large 2D problems (X-ray CT) and fully 3D problems

Neither is ideal.

$\therefore$ need *special-purpose algorithms* for image reconstruction!

Data-Mismatch Functions Revisited

For fast converging, intrinsically monotone algorithms, consider the form of Ψ.

WLS:

$$-L(\boldsymbol{x}) = \sum_{i=1}^{n_d} \frac{1}{2} w_i \left(y_i - [\boldsymbol{A}\boldsymbol{x}]_i\right)^2 = \sum_{i=1}^{n_d} h_i([\boldsymbol{A}\boldsymbol{x}]_i), \qquad \text{where } h_i(l) \triangleq \frac{1}{2} w_i (y_i - l)^2.$$

Emission Poisson log-likelihood:

$$-L(\boldsymbol{x}) = \sum_{i=1}^{n_d} ([\boldsymbol{A}\boldsymbol{x}]_i + r_i) - y_i \log([\boldsymbol{A}\boldsymbol{x}]_i + r_i) = \sum_{i=1}^{n_d} h_i([\boldsymbol{A}\boldsymbol{x}]_i)$$

$$\text{where } h_i(l) \triangleq (l + r_i) - y_i \log(l + r_i).$$

Transmission Poisson log-likelihood:

$$-L(\boldsymbol{x}) = \sum_{i=1}^{n_d} \left(b_i e^{-[\boldsymbol{A}\boldsymbol{x}]_i} + r_i\right) - y_i \log\left(b_i e^{-[\boldsymbol{A}\boldsymbol{x}]_i} + r_i\right) = \sum_{i=1}^{n_d} h_i([\boldsymbol{A}\boldsymbol{x}]_i)$$

$$\text{where } h_i(l) \triangleq (b_i e^{-l} + r_i) - y_i \log\left(b_i e^{-l} + r_i\right).$$

MRI, polyenergetic X-ray CT, confocal microscopy, image restoration, ...
All have same *partially separable* form.

General Imaging Cost Function

General form for data-mismatch function:

$$-L(\boldsymbol{x}) = \sum_{i=1}^{n_d} h_i([\boldsymbol{A}\boldsymbol{x}]_i)$$

General form for regularizing penalty function:

$$R(\boldsymbol{x}) = \sum_k \psi_k([\boldsymbol{C}\boldsymbol{x}]_k)$$

General form for *cost function*:

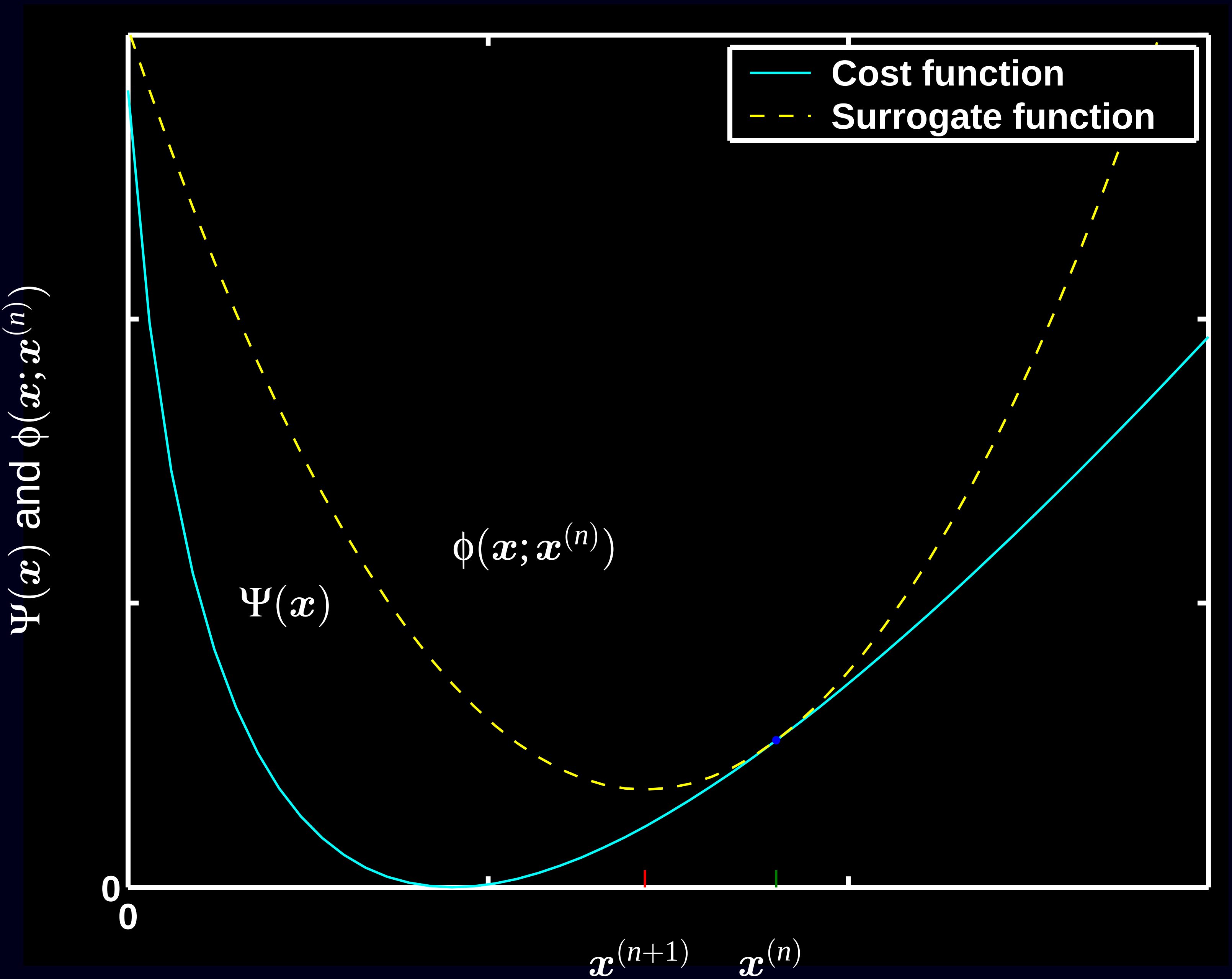
$$\boxed{\Psi(\boldsymbol{x}) = -L(\boldsymbol{x}) + \beta R(\boldsymbol{x}) = \sum_{i=1}^{n_d} h_i([\boldsymbol{A}\boldsymbol{x}]_i) + \beta \sum_k \psi_k([\boldsymbol{C}\boldsymbol{x}]_k)}$$

Properties of Ψ we can exploit:

- summation form (due to independence of measurements)
- convexity of h_i functions (usually)
- summation argument (inner product of $\boldsymbol{x}$ with ith row of $\boldsymbol{A}$)

Most methods that use these properties are forms of *optimization transfer*.

Optimization Transfer Illustrated



Optimization Transfer

General iteration:

$$\boxed{\boldsymbol{x}^{(n+1)} = \arg\min_{\boldsymbol{x} \geq \boldsymbol{0}} \phi\left(\boldsymbol{x}; \boldsymbol{x}^{(n)}\right)}$$

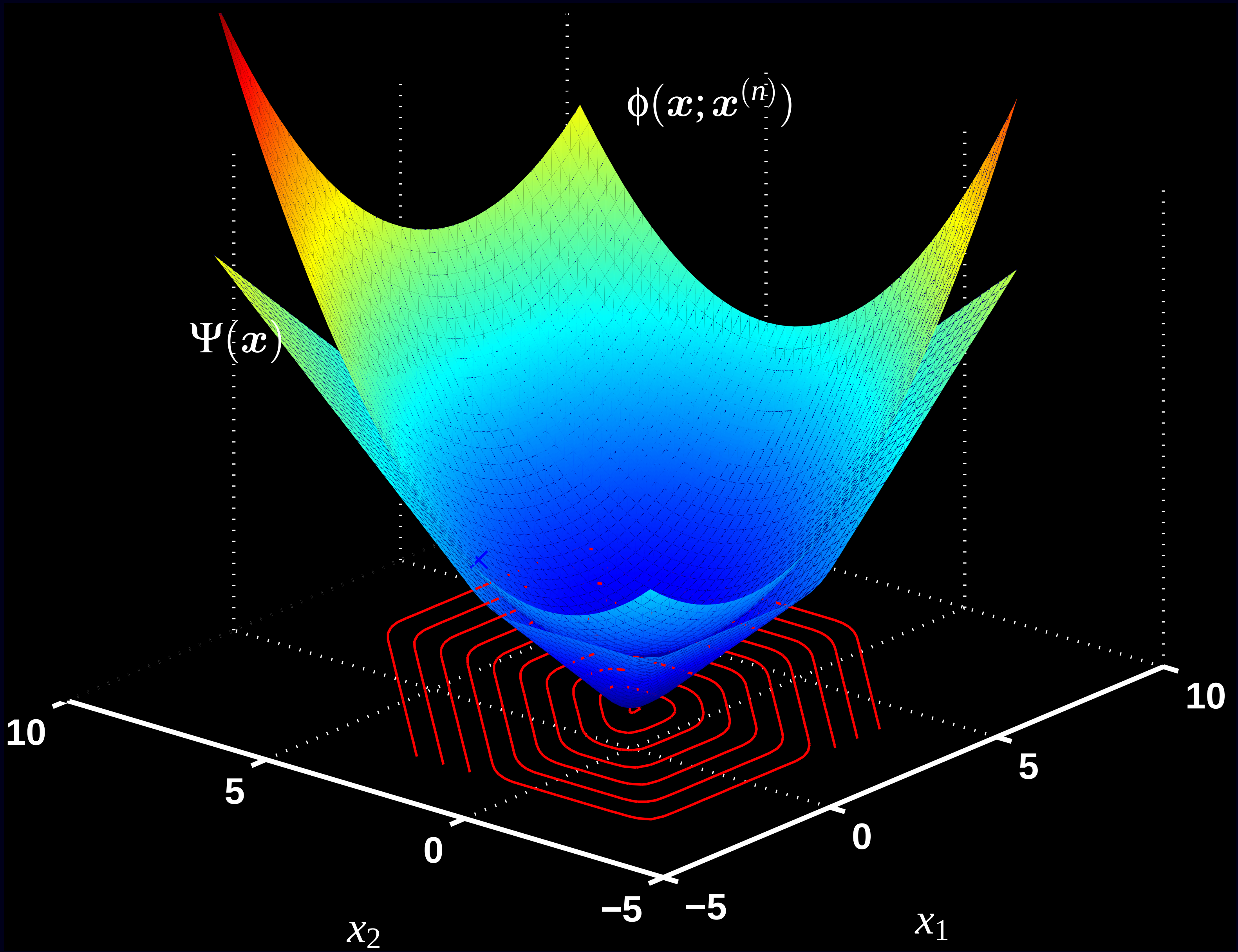
Monotonicity conditions (Ψ decreases provided these hold):

- $\phi(\boldsymbol{x}^{(n)}; \boldsymbol{x}^{(n)}) = \Psi(\boldsymbol{x}^{(n)})$ (matched current value)
- $\nabla_{\boldsymbol{x}} \phi(\boldsymbol{x}; \boldsymbol{x}^{(n)}) \Big|_{\boldsymbol{x}=\boldsymbol{x}^{(n)}} = \nabla \Psi(\boldsymbol{x}) \Big|_{\boldsymbol{x}=\boldsymbol{x}^{(n)}}$ (matched gradient)
- $\phi(\boldsymbol{x}; \boldsymbol{x}^{(n)}) \geq \Psi(\boldsymbol{x}) \quad \forall \boldsymbol{x} \geq \boldsymbol{0}$ (lies above)

These 3 (sufficient) conditions are satisfied by the *Q* function of the EM algorithm (and SAGE).

The 3rd condition is *not* satisfied by the Newton-Raphson quadratic approximation, which leads to its nonmonotonicity.

Optimization Transfer in 2d



Optimization Transfer cf EM Algorithm

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

M-step: minimize surrogate function

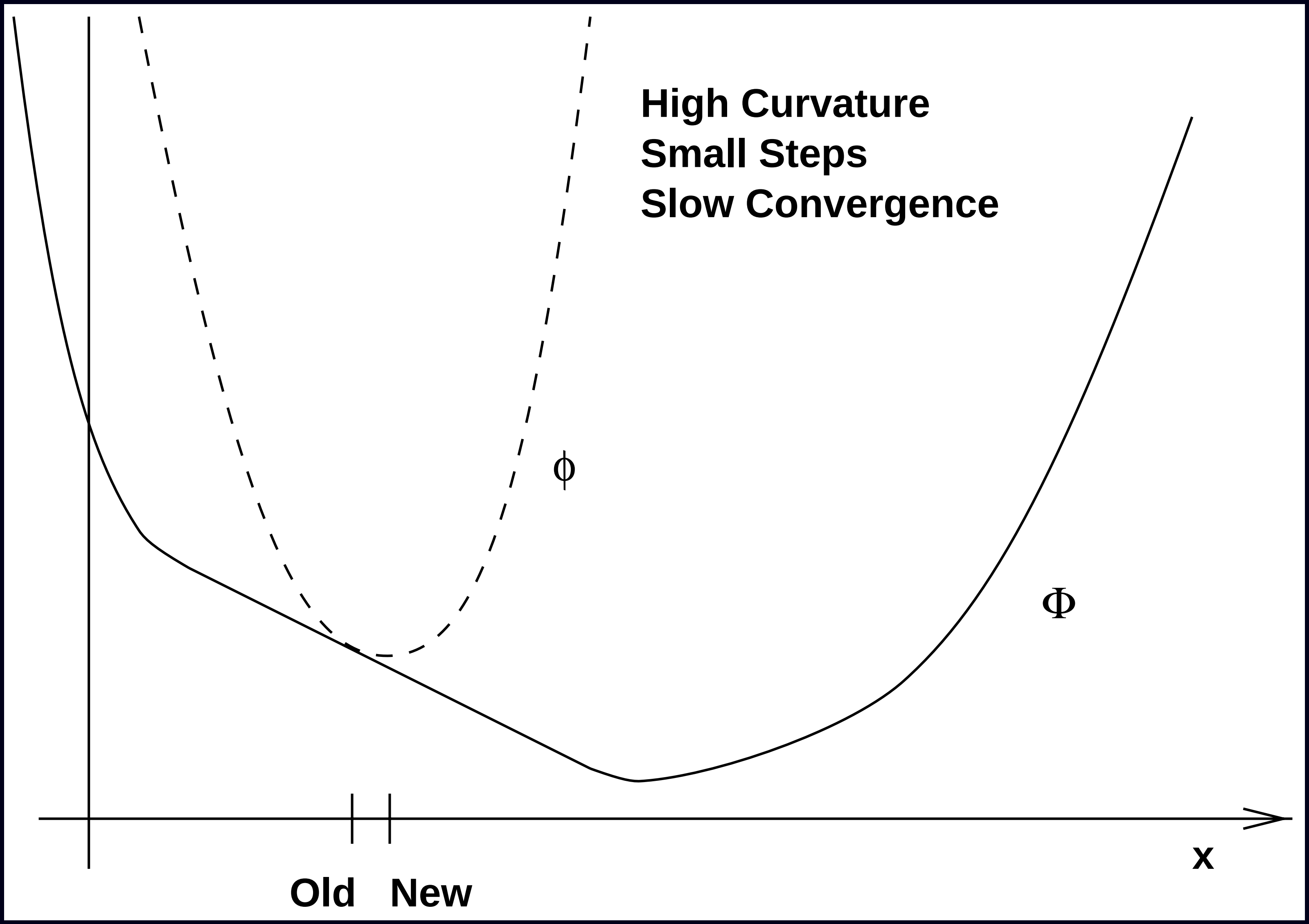
$$\boldsymbol{x}^{(n+1)} = \arg\min_{\boldsymbol{x} \geq \boldsymbol{0}} \phi(\boldsymbol{x}; \boldsymbol{x}^{(n)})$$

Designing surrogate functions

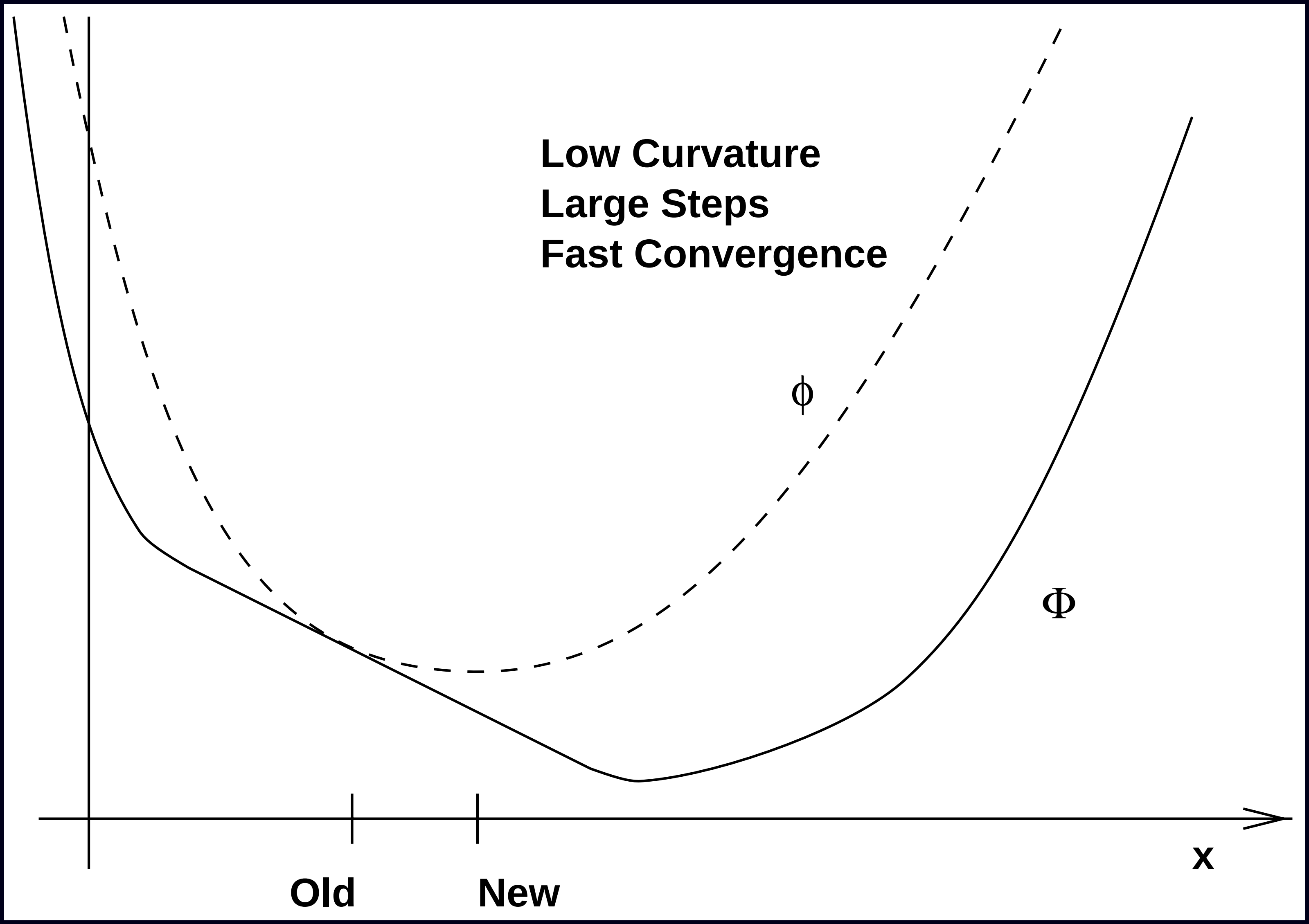
- Easy to “compute”
- Easy to minimize
- Fast convergence rate

Often mutually incompatible goals $\therefore$ compromises

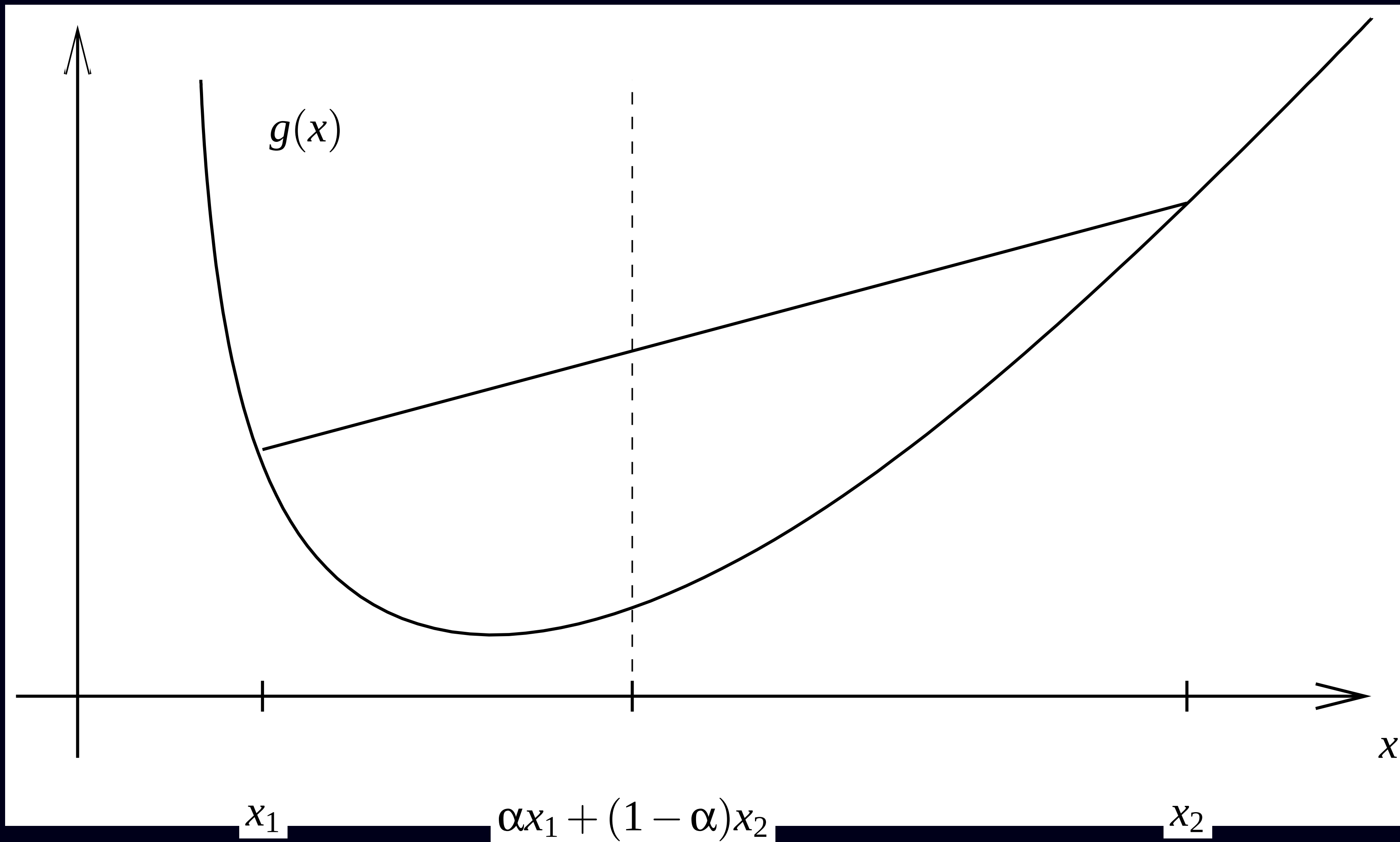
Convergence Rate: Slow



Convergence Rate: Fast



Tool: Convexity Inequality



g convex $\Rightarrow g(\alpha x_1 + (1-\alpha)x_2) \le \alpha g(x_1) + (1-\alpha) g(x_2)$ for $\alpha \in [0,1]$

More generally: $\alpha_k \ge 0$ and $\sum_k \alpha_k = 1 \Rightarrow g(\sum_k \alpha_k x_k) \le \sum_k \alpha_k g(x_k)$. Sum outside!

Example 1: Classical ML-EM Algorithm

Negative Poisson log-likelihood *cost function* (unregularized):

$$\Psi(\boldsymbol{x}) = \sum_{i=1}^{n_d} h_i([\boldsymbol{A}\boldsymbol{x}]_i), \qquad h_i(l) = (l + r_i) - y_i \log(l + r_i).$$

Intractable to minimize directly due to summation within logarithm.

Clever trick due to De Pierro (let $\bar{y}_i^{(n)} = [\boldsymbol{A}\boldsymbol{x}^{(n)}]_i + r_i$):

$$[\boldsymbol{A}\boldsymbol{x}]_i = \sum_{j=1}^{n_p} a_{ij} x_j = \sum_{j=1}^{n_p} \left[\frac{a_{ij} x_j^{(n)}}{\bar{y}_i^{(n)}}\right] \left(\frac{x_j}{x_j^{(n)}} \bar{y}_i^{(n)}\right).$$

Since the h_i's are *convex* in Poisson emission model:

$$h_i([\boldsymbol{A}\boldsymbol{x}]_i) = h_i\left(\sum_{j=1}^{n_p} \left[\frac{a_{ij} x_j^{(n)}}{\bar{y}_i^{(n)}}\right] \left(\frac{x_j}{x_j^{(n)}} \bar{y}_i^{(n)}\right)\right) \leq \sum_{j=1}^{n_p} \left[\frac{a_{ij} x_j^{(n)}}{\bar{y}_i^{(n)}}\right] h_i\left(\frac{x_j}{x_j^{(n)}} \bar{y}_i^{(n)}\right)$$

$$\Psi(\boldsymbol{x}) = \sum_{i=1}^{n_d} h_i([\boldsymbol{A}\boldsymbol{x}]_i) \leq \phi(\boldsymbol{x}; \boldsymbol{x}^{(n)}) \triangleq \sum_{i=1}^{n_d} \sum_{j=1}^{n_p} \left[\frac{a_{ij} x_j^{(n)}}{\bar{y}_i^{(n)}}\right] h_i\left(\frac{x_j}{x_j^{(n)}} \bar{y}_i^{(n)}\right)$$

Replace convex *cost function* $\Psi(\boldsymbol{x})$ with *separable* surrogate function $\phi(\boldsymbol{x}; \boldsymbol{x}^{(n)})$.

"ML-EM Algorithm" M-step

E-step gave separable surrogate function:

$$\phi(\boldsymbol{x};\boldsymbol{x}^{(n)}) = \sum_{j=1}^{n_p} \phi_j(x_j;\boldsymbol{x}^{(n)}), \quad \text{where} \quad \phi_j(x_j;\boldsymbol{x}^{(n)}) \triangleq \sum_{i=1}^{n_d} \left[\frac{a_{ij} x_j^{(n)}}{\bar{y}_i^{(n)}}\right] h_i\left(\frac{x_j}{x_j^{(n)}} \bar{y}_i^{(n)}\right).$$

M-step separates:

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

Minimizing:

$$\frac{\partial}{\partial x_j} \phi_j(x_j;\boldsymbol{x}^{(n)}) = \sum_{i=1}^{n_d} a_{ij} \dot{h}_i\left(\bar{y}_i^{(n)} x_j / x_j^{(n)}\right) = \sum_{i=1}^{n_d} a_{ij} \left[1 - \frac{y_i}{\bar{y}_i^{(n)} x_j / x_j^{(n)}}\right]\Bigg|_{x_j = x_j^{(n+1)}} = 0.$$

Solving (in case $r_i = 0$):

$$\boxed{x_j^{(n+1)} = x_j^{(n)} \left[\sum_{i=1}^{n_d} a_{ij} \frac{y_i}{[\boldsymbol{A}\boldsymbol{x}^{(n)}]_i}\right] \Big/ \left(\sum_{i=1}^{n_d} a_{ij}\right), \qquad j = 1,\ldots,n_p}$$

- Derived without any statistical considerations, unlike classical EM formulation.
- Uses only convexity and algebra.
- Guaranteed monotonic: surrogate function ϕ satisfies the 3 required properties.
- M-step trivial due to *separable surrogate*.

ML-EM is Scaled Gradient Descent

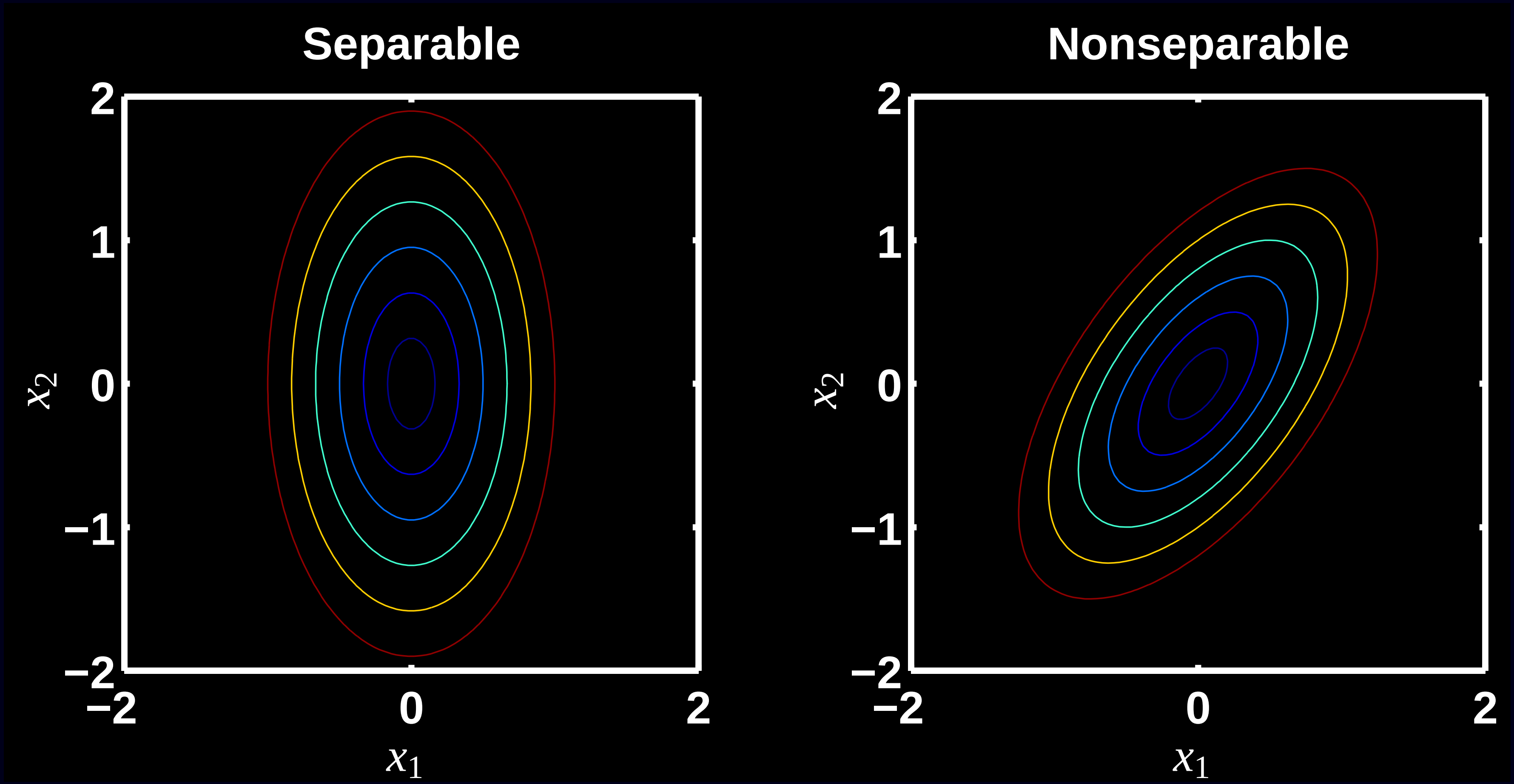
$$
\begin{aligned}
x_j^{(n+1)} &= x_j^{(n)} \left[\sum_{i=1}^{n_d} a_{ij} \frac{y_i}{\bar{y}_i^{(n)}} \right] \Big/ \left(\sum_{i=1}^{n_d} a_{ij} \right) \\
&= x_j^{(n)} + x_j^{(n)} \left[\sum_{i=1}^{n_d} a_{ij} \left(\frac{y_i}{\bar{y}_i^{(n)}} - 1 \right) \right] \Big/ \left(\sum_{i=1}^{n_d} a_{ij} \right) \\
&= \boxed{x_j^{(n)} - \left(\frac{x_j^{(n)}}{\sum_{i=1}^{n_d} a_{ij}} \right) \frac{\partial}{\partial x_j} \Psi(\boldsymbol{x}^{(n)}),} \qquad j = 1, \ldots, n_p
\end{aligned}
$$

$$
\boldsymbol{x}^{(n+1)} = \boldsymbol{x}^{(n)} + \boldsymbol{D}(\boldsymbol{x}^{(n)}) \nabla \Psi(\boldsymbol{x}^{(n)})
$$

This particular diagonal scaling matrix remarkably

- ensures monotonicity,
- ensures nonnegativity.

Consideration: Separable vs Nonseparable



Contour plots: loci of equal function values.

Uncoupled vs coupled minimization.

Separable Surrogate Functions (Easy M-step)

The preceding EM derivation structure applies to *any cost function* of the form

$$\Psi(\boldsymbol{x}) = \sum_{i=1}^{n_d} h_i([\boldsymbol{A}\boldsymbol{x}]_i).$$

cf ISRA (for nonnegative LS), “convex algorithm” for transmission reconstruction

Derivation yields a separable surrogate function

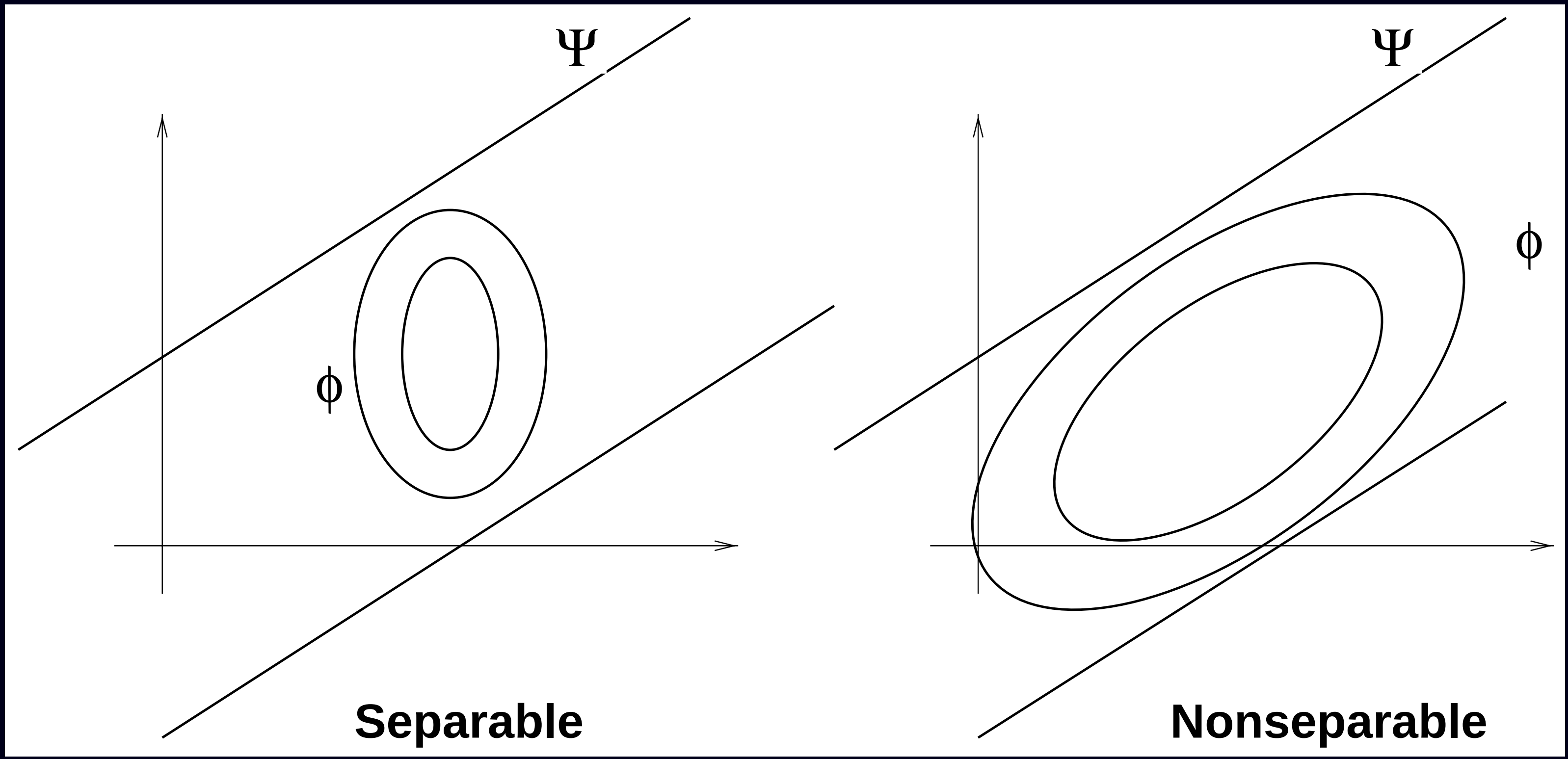
$$\Psi(\boldsymbol{x}) \leq \phi(\boldsymbol{x}; \boldsymbol{x}^{(n)}), \quad \text{where} \quad \phi(\boldsymbol{x}; \boldsymbol{x}^{(n)}) = \sum_{j=1}^{n_p} \phi_j(x_j; \boldsymbol{x}^{(n)})$$

M-step separates into 1D minimization problems (fully parallelizable):

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

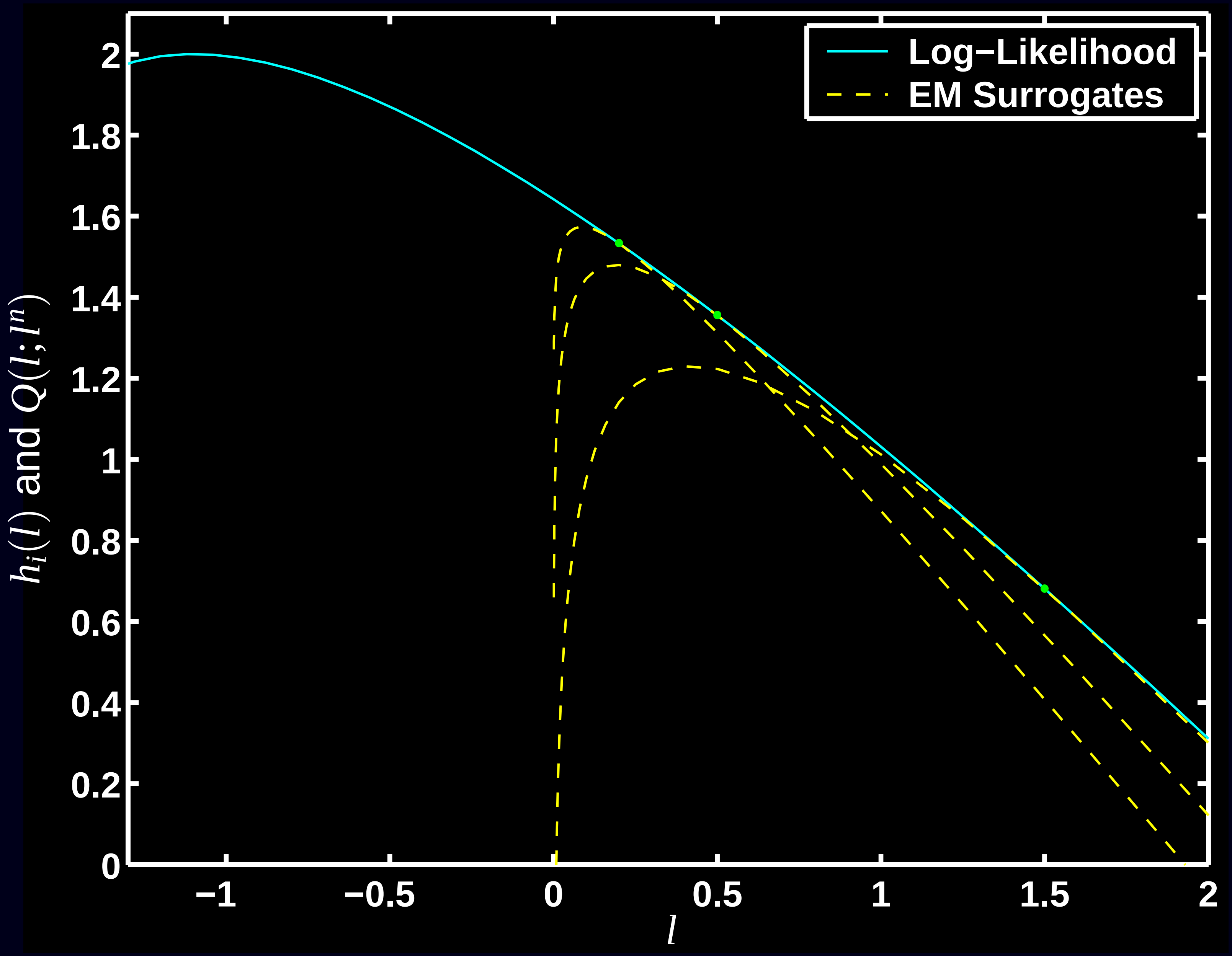
Why do EM / ISRA / convex-algorithm / etc. converge so slowly?

Separable vs Nonseparable



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).

High Curvature of EM Surrogate



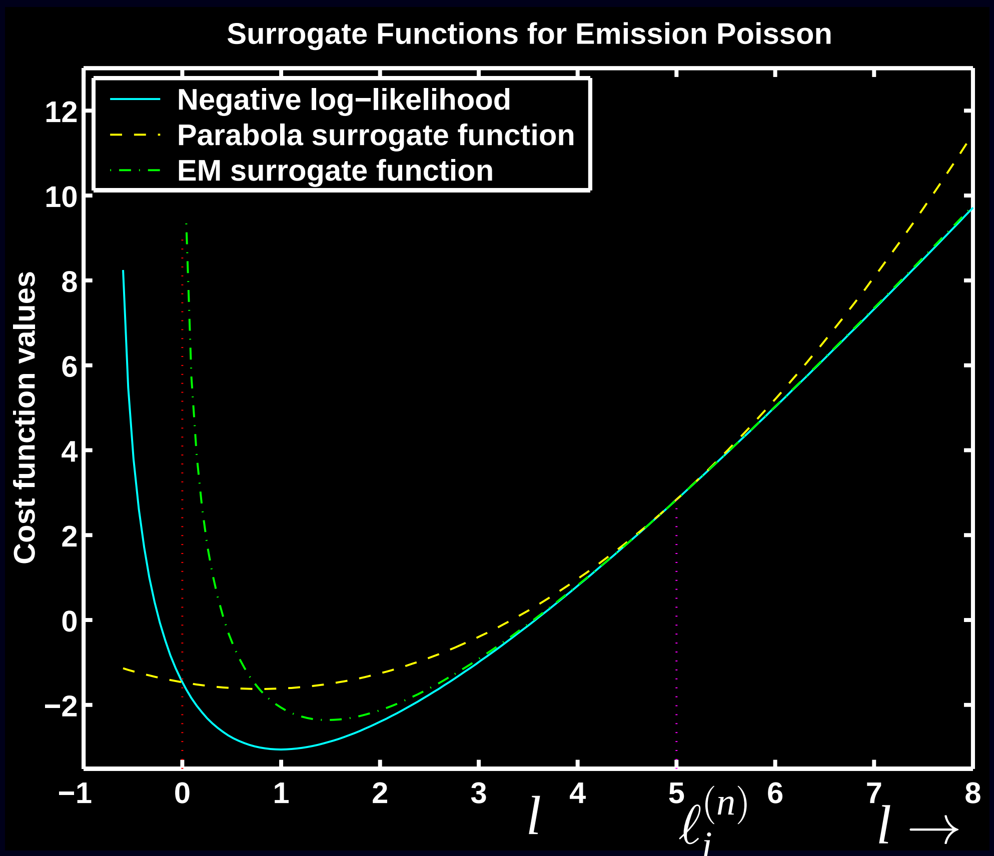
1D Parabola Surrogate Function

Find parabola $q_i^{(n)}(l)$ of the form:

$$q_i^{(n)}(l) = h_i(\ell_i^{(n)}) + \dot{h}_i(\ell_i^{(n)})(l - \ell_i^{(n)}) + c_i^{(n)} \frac{1}{2}(l - \ell_i^{(n)})^2, \text{ where } \ell_i^{(n)} \triangleq [\boldsymbol{A}\boldsymbol{x}^{(n)}]_i$$

Satisfies tangent condition. Choose curvature to ensure “lies above” condition:

$$c_i^{(n)} \triangleq \min\left\{c \geq 0 : q_i^{(n)}(l) \geq h_i(l), \quad \forall l \geq 0\right\}.$$



Lower curvature!

Paraboloidal Surrogate

Combining 1D parabola surrogates yields *paraboloidal surrogate*:

$$\Psi(\boldsymbol{x}) = \sum_{i=1}^{n_d} h_i([\boldsymbol{A}\boldsymbol{x}]_i) \le \phi(\boldsymbol{x}; \boldsymbol{x}^{(n)}) = \sum_{i=1}^{n_d} q_i^{(n)}([\boldsymbol{A}\boldsymbol{x}]_i)$$

Rewriting: $\phi(\boldsymbol{\delta} + \boldsymbol{x}^{(n)}; \boldsymbol{x}^{(n)}) = \Psi(\boldsymbol{x}^{(n)}) + \nabla\Psi(\boldsymbol{x}^{(n)})\boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}'\boldsymbol{A}' \operatorname{diag}\left\{c_i^{(n)}\right\} \boldsymbol{A}\boldsymbol{\delta}$

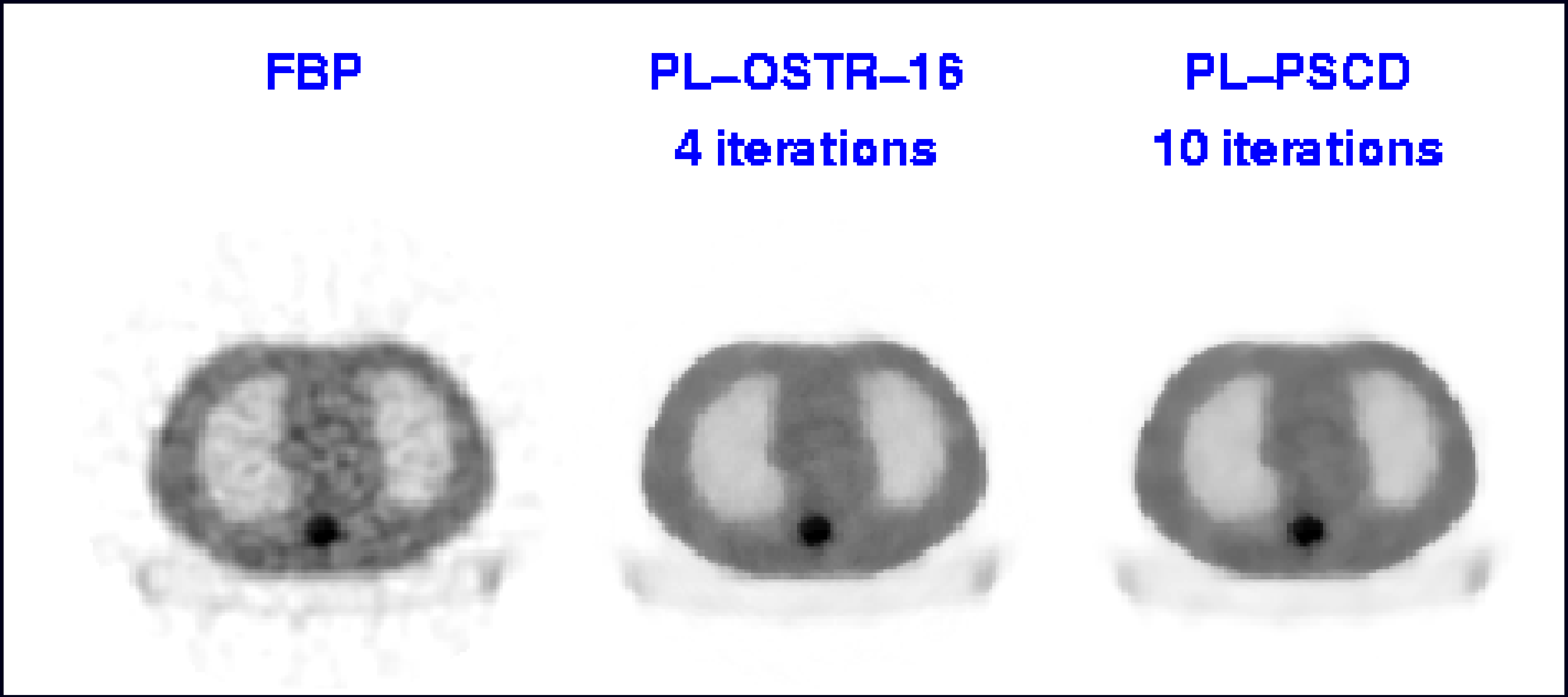
Advantages

- Surrogate $\phi(\boldsymbol{x}; \boldsymbol{x}^{(n)})$ is *quadratic*, unlike Poisson log-likelihood
  $\Rightarrow$ easier to minimize
- Not separable (unlike EM surrogate)
- Not self-similar (unlike EM surrogate)
- Small curvatures $\Rightarrow$ fast convergence
- Instrinsically monotone global convergence
- Fairly simple to derive / implement

Quadratic minimization

- Coordinate descent
  - \+ fast converging
  - \+ Nonnegativity easy
  - \- precomputed column-stored system matrix
- Gradient-based quadratic minimization methods
  - \- Nonnegativity inconvenient

Example: PSCD for PET Transmission Scans



- square-pixel basis
- strip-integral system model
- shifted-Poisson statistical model
- edge-preserving convex regularization (Huber)
- nonnegativity constraint
- inscribed circle support constraint
- paraboloidal surrogate coordinate descent (PSCD) algorithm

Separable Paraboloidal Surrogate

To derive a parallelizable algorithm apply another De Pierro trick:

$$[\boldsymbol{A}\boldsymbol{x}]_i = \sum_{j=1}^{n_p} \pi_{ij} \left[\frac{a_{ij}}{\pi_{ij}} (x_j - x_j^{(n)}) + \ell_i^{(n)} \right], \qquad \ell_i^{(n)} = [\boldsymbol{A}\boldsymbol{x}^{(n)}]_i.$$

Provided $\pi_{ij} \geq 0$ and $\sum_{j=1}^{n_p} \pi_{ij} = 1$, since parabola q_i is convex:

$$q_i^{(n)}([\boldsymbol{A}\boldsymbol{x}]_i) = q_i^{(n)} \left(\sum_{j=1}^{n_p} \pi_{ij} \left[\frac{a_{ij}}{\pi_{ij}} (x_j - x_j^{(n)}) + \ell_i^{(n)} \right] \right) \leq \sum_{j=1}^{n_p} \pi_{ij} q_i^{(n)} \left(\frac{a_{ij}}{\pi_{ij}} (x_j - x_j^{(n)}) + \ell_i^{(n)} \right)$$

$$\therefore \phi(\boldsymbol{x}; \boldsymbol{x}^{(n)}) = \sum_{i=1}^{n_d} q_i^{(n)}([\boldsymbol{A}\boldsymbol{x}]_i) \leq \tilde{\phi}(\boldsymbol{x}; \boldsymbol{x}^{(n)}) \triangleq \sum_{i=1}^{n_d} \sum_{j=1}^{n_p} \pi_{ij} q_i^{(n)} \left(\frac{a_{ij}}{\pi_{ij}} (x_j - x_j^{(n)}) + \ell_i^{(n)} \right)$$

Separable Paraboloidal Surrogate:

$$\tilde{\phi}(\boldsymbol{x}; \boldsymbol{x}^{(n)}) = \sum_{j=1}^{n_p} \phi_j(x_j; \boldsymbol{x}^{(n)}), \qquad \phi_j(x_j; \boldsymbol{x}^{(n)}) \triangleq \sum_{i=1}^{n_d} \pi_{ij} q_i^{(n)} \left(\frac{a_{ij}}{\pi_{ij}} (x_j - x_j^{(n)}) + \ell_i^{(n)} \right)$$

Parallelizable M-step (cf gradient descent!):

$$x_j^{(n+1)} = \arg\min_{x_j \geq 0} \phi_j(x_j; \boldsymbol{x}^{(n)}) = \left[x_j^{(n)} - \frac{1}{d_j^{(n)}} \frac{\partial}{\partial x_j} \Psi(\boldsymbol{x}^{(n)}) \right]_+, \qquad d_j^{(n)} = \sum_{i=1}^{n_d} \frac{a_{ij}^2}{\pi_{ij}} c_i^{(n)}$$

Natural choice is $\pi_{ij} = |a_{ij}| / |a|_i$, $|a|_i = \sum_{j=1}^{n_p} |a_{ij}|$

Example: Poisson ML Transmission Problem

Transmission negative log-likelihood (for ith ray):

$$h_i(l) = (b_i e^{-l} + r_i) - y_i \log\left(b_i e^{-l} + r_i\right).$$

Optimal (smallest) parabola surrogate curvature (Erdoğan, T-MI, Sep. 1999):

$$c_i^{(n)} = c(\ell_i^{(n)}, h_i), \qquad c(l,h) = \begin{cases} \left[2\dfrac{h(0) - h(l) + \dot{h}(l)l}{l^2}\right]_+, & l > 0 \\ \left[\ddot{h}(l)\right]_+, & l = 0. \end{cases}$$

Separable Paraboloidal Surrogate Algorithm:

Precompute $|a|_i = \sum_{j=1}^{n_p} a_{ij}, \qquad i = 1, \ldots, n_d$

$$\begin{aligned} \ell_i^{(n)} &= [\boldsymbol{A}\boldsymbol{x}^{(n)}]_i, && \text{(forward projection)} \\ \bar{y}_i^{(n)} &= b_i e^{-\ell_i^{(n)}} + r_i && \text{(predicted means)} \\ \dot{h}_i^{(n)} &= 1 - y_i/\bar{y}_i^{(n)} && \text{(slopes)} \\ c_i^{(n)} &= c(\ell_i^{(n)}, h_i) && \text{(curvatures)} \end{aligned}$$

$$x_j^{(n+1)} = \left[x_j^{(n)} - \frac{1}{d_j^{(n)}} \frac{\partial}{\partial x_j} \Psi(\boldsymbol{x}^{(n)})\right]_+ = \left[x_j^{(n)} - \frac{\sum_{i=1}^{n_d} a_{ij} \dot{h}_i^{(n)}}{\sum_{i=1}^{n_d} a_{ij} |a|_i c_i^{(n)}}\right]_+, \qquad j = 1, \ldots, n_p$$

Monotonically decreases *cost function* each iteration. No logarithm!

The MAP-EM M-step “Problem”

Add a penalty function to our surrogate for the negative log-likelihood:

$$\begin{aligned}\Psi(\boldsymbol{x}) &= -L(\boldsymbol{x}) + \beta R(\boldsymbol{x}) \\ \phi(\boldsymbol{x};\boldsymbol{x}^{(n)}) &= \sum_{j=1}^{n_p} \phi_j(x_j;\boldsymbol{x}^{(n)}) + \beta R(\boldsymbol{x})\end{aligned}$$

M-step: $\boldsymbol{x}^{(n+1)} = \arg\min_{\boldsymbol{x}\geq \boldsymbol{0}} \phi(\boldsymbol{x};\boldsymbol{x}^{(n)}) = \arg\min_{\boldsymbol{x}\geq \boldsymbol{0}} \sum_{j=1}^{n_p} \phi_j(x_j;\boldsymbol{x}^{(n)}) + \beta R(\boldsymbol{x}) = ?$

For nonseparable penalty functions, the M-step is coupled $\therefore$ difficult.

Suboptimal solutions

- Generalized EM (GEM) algorithm (coordinate descent on ϕ)
  Monotonic, but inherits slow convergence of EM.
- One-step late (OSL) algorithm (use outdated gradients) (Green, T-MI, 1990)

$$\frac{\partial}{\partial x_j}\phi(\boldsymbol{x};\boldsymbol{x}^{(n)}) = \frac{\partial}{\partial x_j}\phi_j(x_j;\boldsymbol{x}^{(n)}) + \beta \frac{\partial}{\partial x_j} R(\boldsymbol{x}) \overset{?}{\approx} \frac{\partial}{\partial x_j}\phi_j(x_j;\boldsymbol{x}^{(n)}) + \beta \frac{\partial}{\partial x_j} R(\boldsymbol{x}^{(n)})$$

  Nonmonotonic. Known to diverge, depending on β.
  Temptingly simple, but *avoid!*

Contemporary solution

- Use separable surrogate for penalty function too (De Pierro, T-MI, Dec. 1995)
  Ensures monotonicity. Obviates all reasons for using OSL!

De Pierro's MAP-EM Algorithm

Apply separable paraboloidal surrogates to penalty function:

$$R(\boldsymbol{x}) \leq R_{\text{SPS}}(\boldsymbol{x};\boldsymbol{x}^{(n)}) = \sum_{j=1}^{n_p} R_j(x_j;\boldsymbol{x}^{(n)})$$

Overall separable surrogate: $\phi(\boldsymbol{x};\boldsymbol{x}^{(n)}) = \sum_{j=1}^{n_p} \phi_j(x_j;\boldsymbol{x}^{(n)}) + \beta \sum_{j=1}^{n_p} R_j(x_j;\boldsymbol{x}^{(n)})$

The M-step becomes fully parallelizable:

$$x_j^{(n+1)} = \arg\min_{x_j \geq 0} \phi_j(x_j;\boldsymbol{x}^{(n)}) - \beta R_j(x_j;\boldsymbol{x}^{(n)}), \qquad j = 1,\ldots,n_p.$$

Consider quadratic penalty $R(\boldsymbol{x}) = \sum_k \psi([\boldsymbol{C}\boldsymbol{x}]_k)$, where $\psi(t) = t^2/2$.
If $\gamma_{kj} \geq 0$ and $\sum_{j=1}^{n_p} \gamma_{kj} = 1$ then

$$[\boldsymbol{C}\boldsymbol{x}]_k = \sum_{j=1}^{n_p} \gamma_{kj} \left[\frac{c_{kj}}{\gamma_{kj}} (x_j - x_j^{(n)}) + [\boldsymbol{C}\boldsymbol{x}^{(n)}]_k \right].$$

Since ψ is convex:

$$\begin{aligned} \psi([\boldsymbol{C}\boldsymbol{x}]_k) &= \psi\left(\sum_{j=1}^{n_p} \gamma_{kj} \left[\frac{c_{kj}}{\gamma_{kj}} (x_j - x_j^{(n)}) + [\boldsymbol{C}\boldsymbol{x}^{(n)}]_k \right] \right) \\ &\leq \sum_{j=1}^{n_p} \gamma_{kj} \psi\left(\frac{c_{kj}}{\gamma_{kj}} (x_j - x_j^{(n)}) + [\boldsymbol{C}\boldsymbol{x}^{(n)}]_k \right) \end{aligned}$$

De Pierro's Algorithm Continued

So $R(\boldsymbol{x}) \leq R(\boldsymbol{x};\boldsymbol{x}^{(n)}) \triangleq \sum_{j=1}^{n_p} R_j(x_j;\boldsymbol{x}^{(n)})$ where

$$R_j(x_j;\boldsymbol{x}^{(n)}) \triangleq \sum_k \gamma_{kj} \psi\left(\frac{c_{kj}}{\gamma_{kj}}(x_j - x_j^{(n)}) + [\boldsymbol{C}\boldsymbol{x}^{(n)}]_k\right)$$

M-step: Minimizing $\phi_j(x_j;\boldsymbol{x}^{(n)}) + \beta R_j(x_j;\boldsymbol{x}^{(n)})$ yields the iteration:

$$x_j^{(n+1)} = \frac{x_j^{(n)} \sum_{i=1}^{n_d} a_{ij} y_i / \bar{y}_i^{(n)}}{B_j + \sqrt{B_j^2 + \left(x_j^{(n)} \sum_{i=1}^{n_d} a_{ij} y_i / \bar{y}_i^{(n)}\right)\left(\beta \sum_k c_{kj}^2 / \gamma_{kj}\right)}}$$

where $$B_j \triangleq \frac{1}{2}\left[\sum_{i=1}^{n_d} a_{ij} + \beta \sum_k \left(c_{kj}[\boldsymbol{C}\boldsymbol{x}^{(n)}]_k - \frac{c_{kj}^2}{\gamma_{kj}} x_j^{(n)}\right)\right], \qquad j = 1,\ldots,n_p$$

and $\bar{y}_i^{(n)} = [\boldsymbol{A}\boldsymbol{x}^{(n)}]_i + r_i$.

Advantages: Intrinsically monotone, nonnegativity, fully parallelizable.
Requires only a couple % more computation per iteration than ML-EM

Disadvantages: Slow convergence (like EM) due to separable surrogate

Ordered Subsets Algorithms

aka *block iterative* or *incremental gradient* algorithms

The gradient appears in essentially every algorithm:

$$\frac{\partial}{\partial x_j} \Psi(\boldsymbol{x}) = \sum_{i=1}^{n_d} a_{ij} \dot{h}_i([\boldsymbol{A}\boldsymbol{x}]_i).$$

This is a *backprojection* of a sinogram of the derivatives $\left\{ \dot{h}_i([\boldsymbol{A}\boldsymbol{x}]_i) \right\}$.

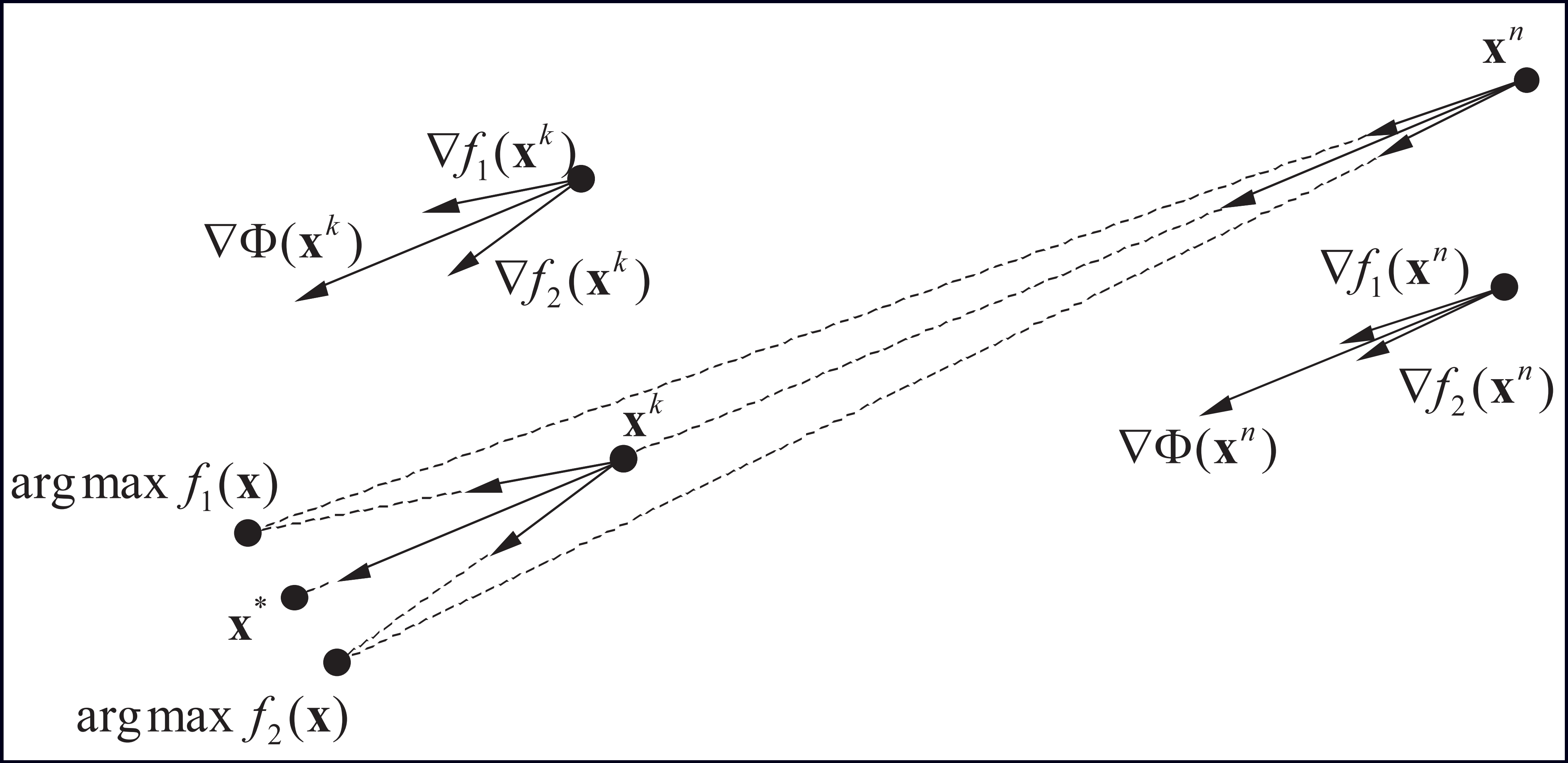
Intuition: with half the angular sampling, this backprojection would be fairly similar

$$\frac{1}{n_d} \sum_{i=1}^{n_d} a_{ij} \dot{h}_i(\cdot) \approx \frac{1}{|S|} \sum_{i \in S} a_{ij} \dot{h}_i(\cdot),$$

where S is a subset of the rays.

To "OS-ize" an algorithm, replace all backprojections with partial sums.

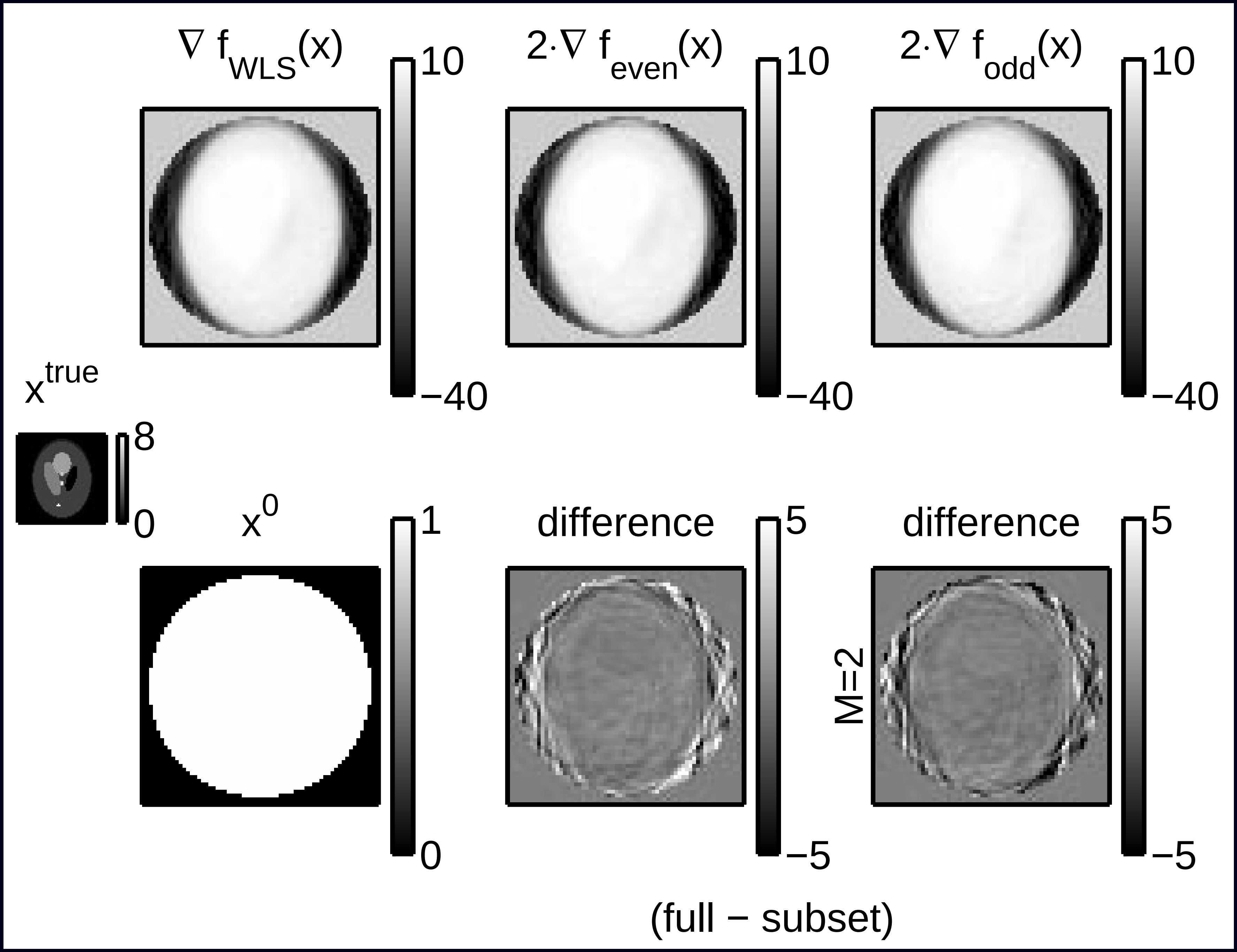
Geometric View of Ordered Subsets



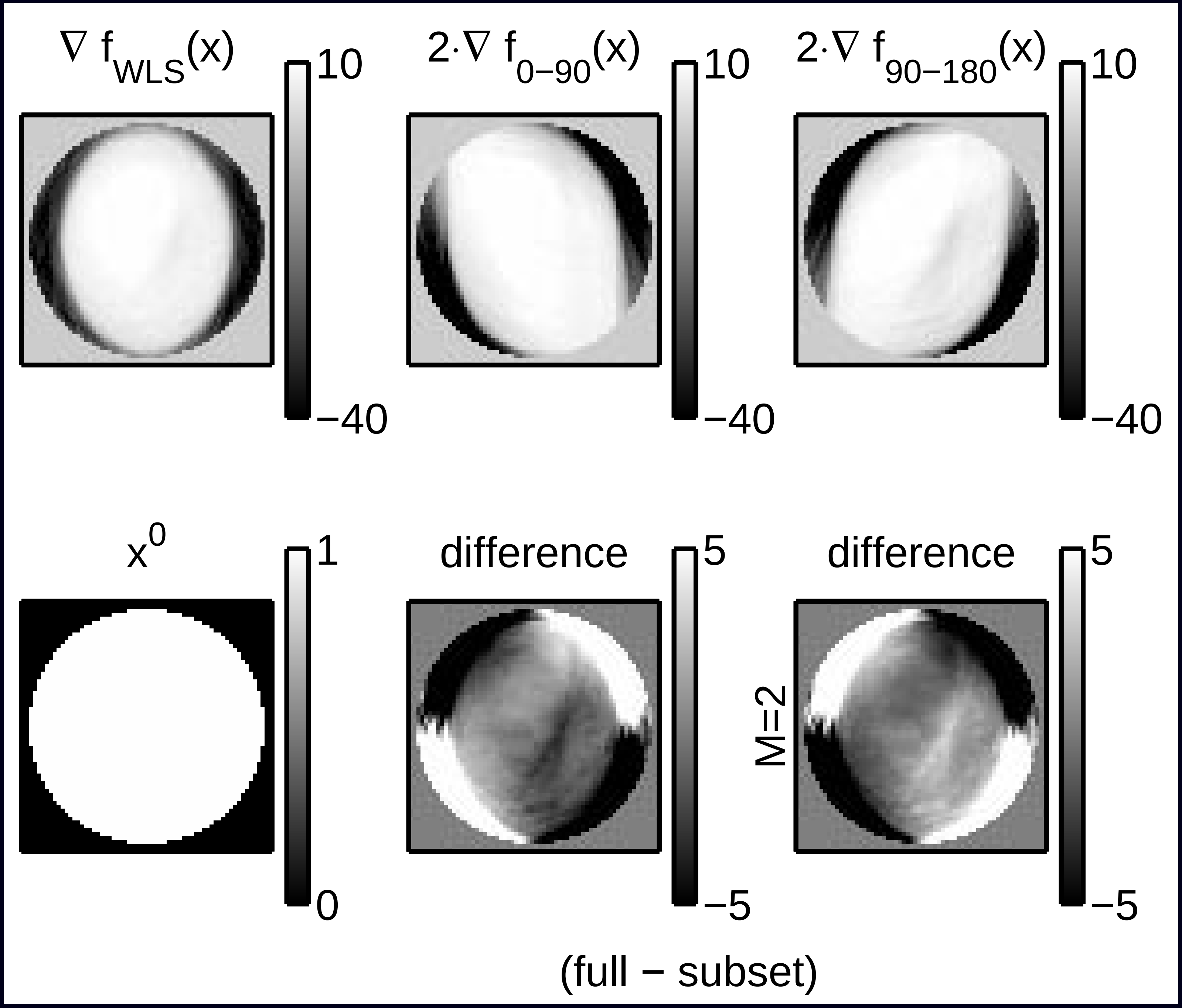
Two subset case: $\Psi(\boldsymbol{x}) = f_1(\boldsymbol{x}) + f_2(\boldsymbol{x})$ (*e.g.*, odd and even projection views).

For $\boldsymbol{x}^{(n)}$ far from $\boldsymbol{x}^\star$, even partial gradients should point roughly towards $\boldsymbol{x}^\star$.
For $\boldsymbol{x}^{(n)}$ near $\boldsymbol{x}^\star$, however, $\nabla\Psi(\boldsymbol{x}) \approx \mathbf{0}$, so $\nabla f_1(\boldsymbol{x}) \approx -\nabla f_2(\boldsymbol{x}) \Rightarrow$ cycles!
Issues. Subset balance: $\nabla\Psi(\boldsymbol{x}) \approx M \nabla f_k(\boldsymbol{x})$. Choice of ordering.

Incremental Gradients (WLS, 2 Subsets)



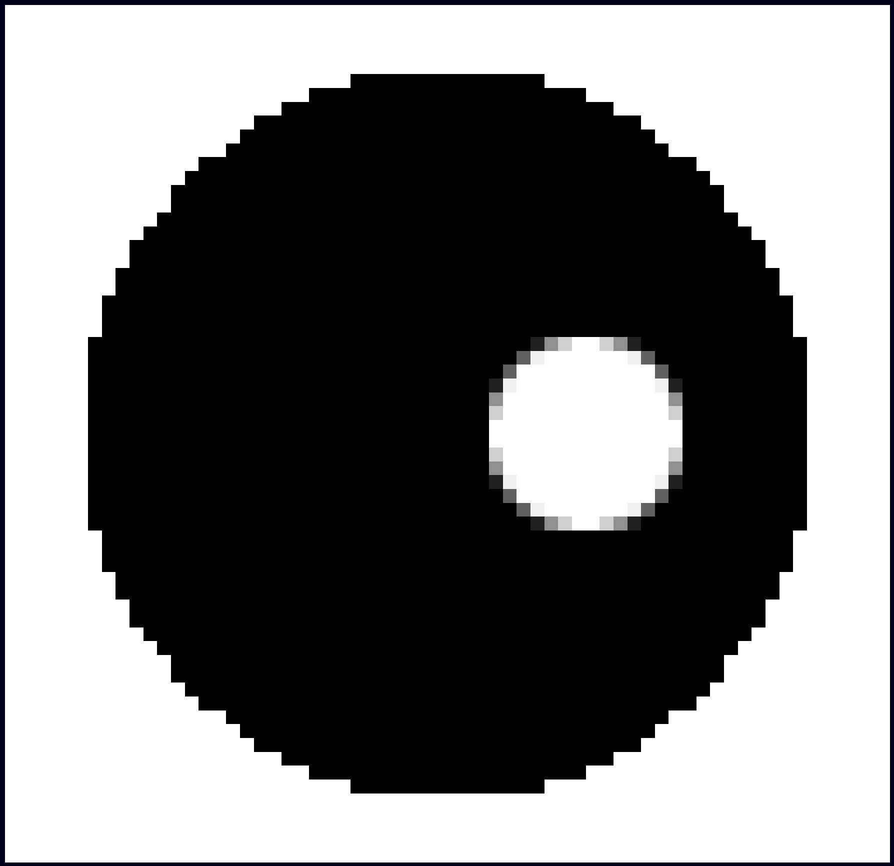
Subset Imbalance



Problems with OS-EM

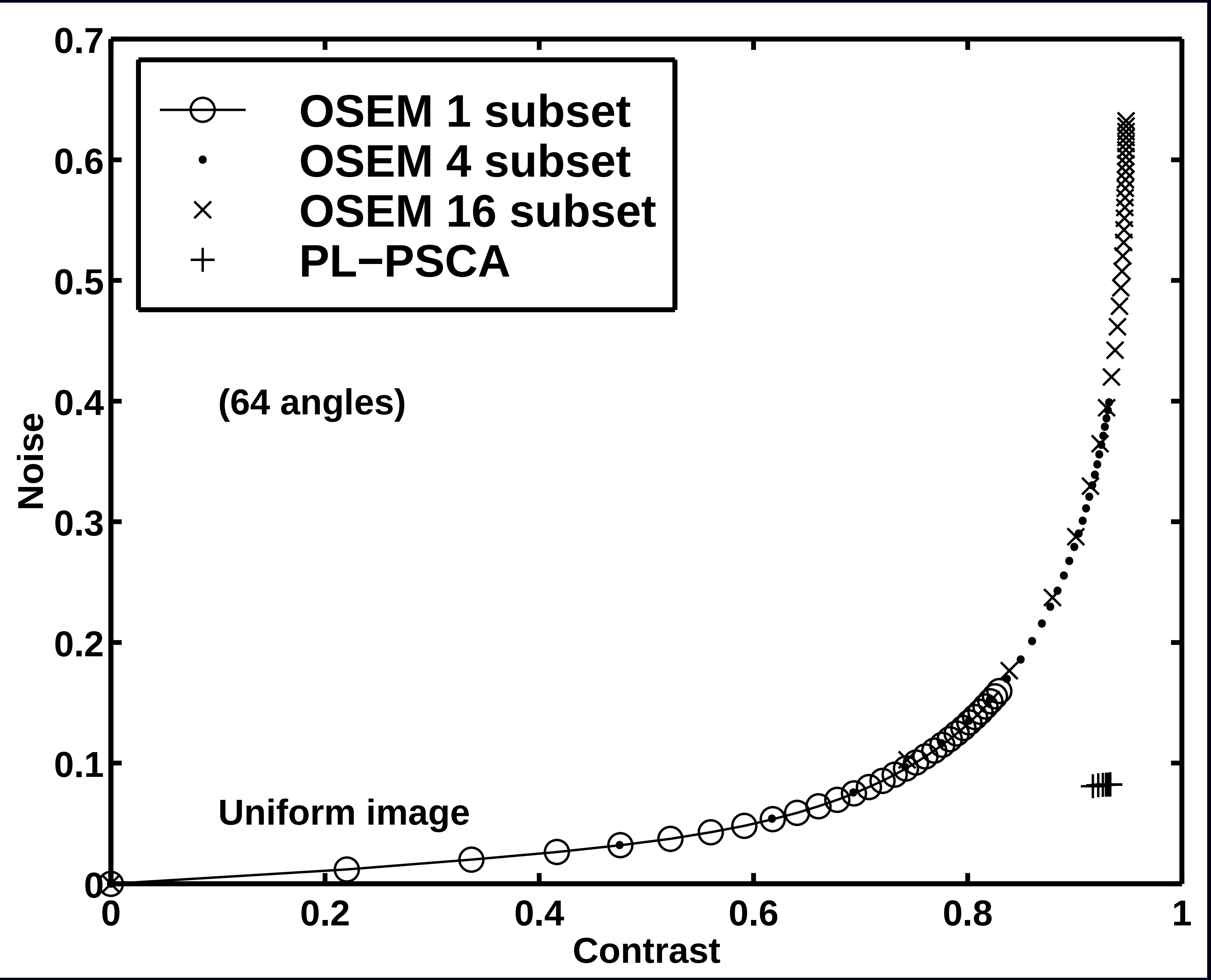
- Non-monotone
- Does not converge (may cycle)
- Byrne's RBBI approach only converges for consistent (noiseless) data
- $\therefore$ unpredictable
  - What resolution after n iterations?
    Object-dependent, spatially nonuniform
  - What variance after n iterations?
  - ROI variance? (*e.g.*, for Huesman's WLS kinetics)

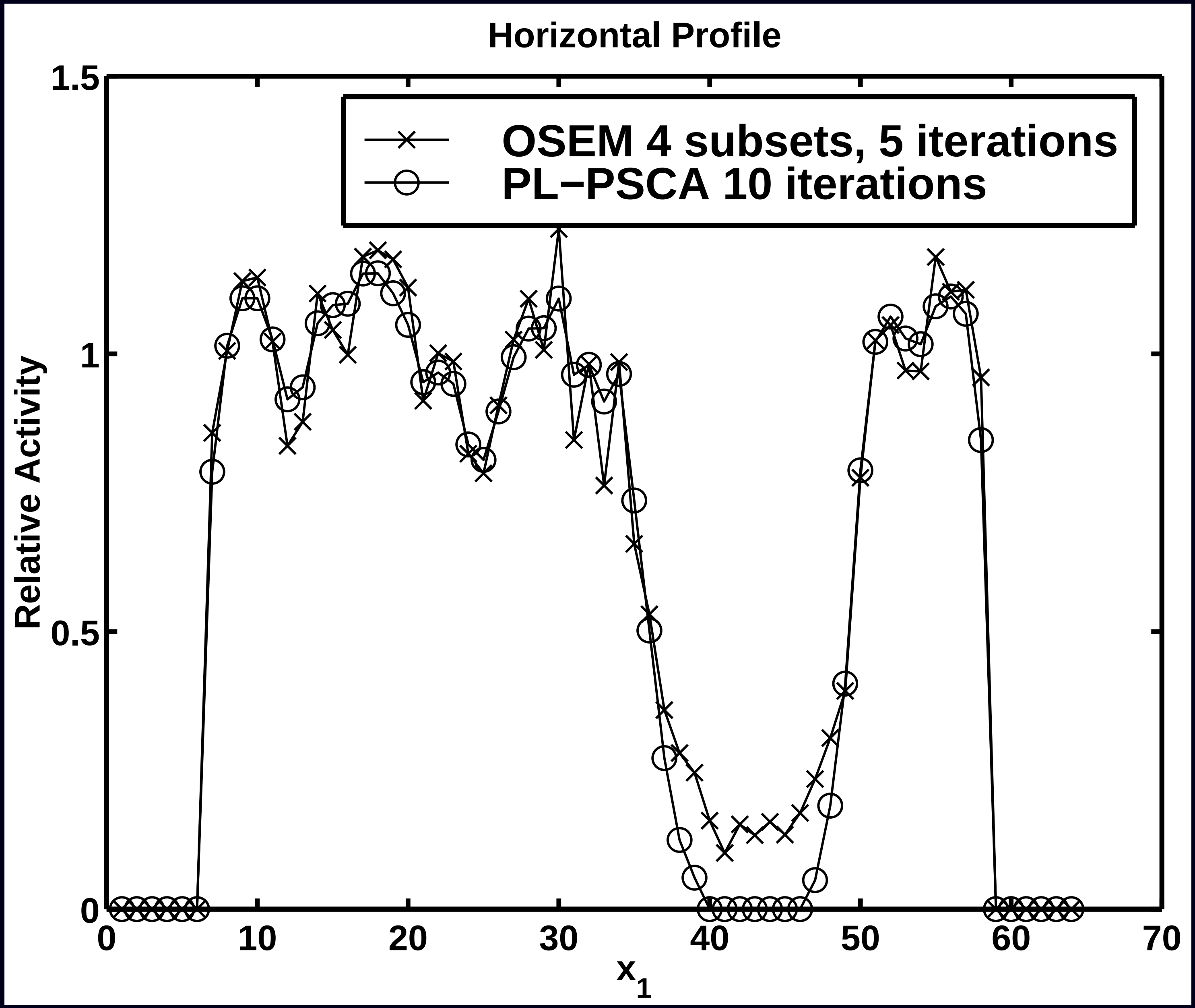
OSEM vs Penalized Likelihood



- 64×62 image
- 66×60 sinogram
- 10^6 counts
- 15% randoms/scatter
- uniform attenuation
- contrast in cold region
- within-region σ opposite side

Contrast-Noise Results





Horizontal Profile
OSEM 4 subsets, 5 iterations
PL−PSCA 10 iterations
Relative Activity
1.5
1
0.5
0
0
10
20
30
40
50
60
70
x1

An Open Problem

Still no algorithm with all of the following properties:

- Nonnegativity easy
- Fast converging
- Intrinsically monotone global convergence
- Accepts any type of system matrix
- Parallelizable

Relaxed block-iterative methods

$$\Psi(\boldsymbol{x}) = \sum_{k=1}^{K} \Psi_k(\boldsymbol{x})$$

$$\boldsymbol{x}^{(n+(k+1)/K)} = \boldsymbol{x}^{(n+k/K)} - \alpha_n D(\boldsymbol{x}^{(n+k/K)}) \nabla \Psi_k(\boldsymbol{x}^{(n+k/K)}), \qquad k = 0, \ldots, K-1$$

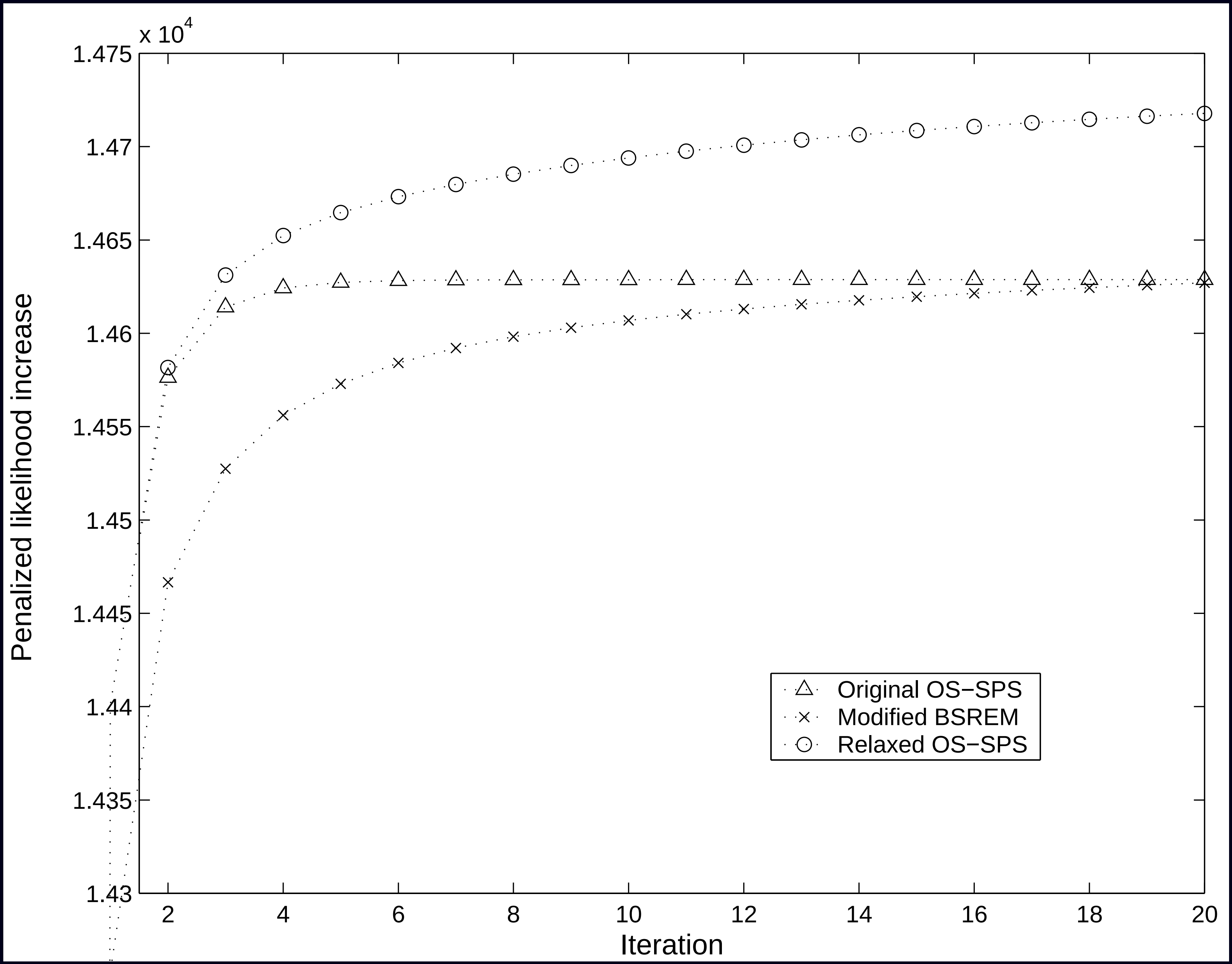
Relaxation of step sizes:

$$\alpha_n \to 0 \text{ as } n \to \infty, \qquad \sum_n \alpha_n = \infty, \qquad \sum_n \alpha_n^2 < \infty$$

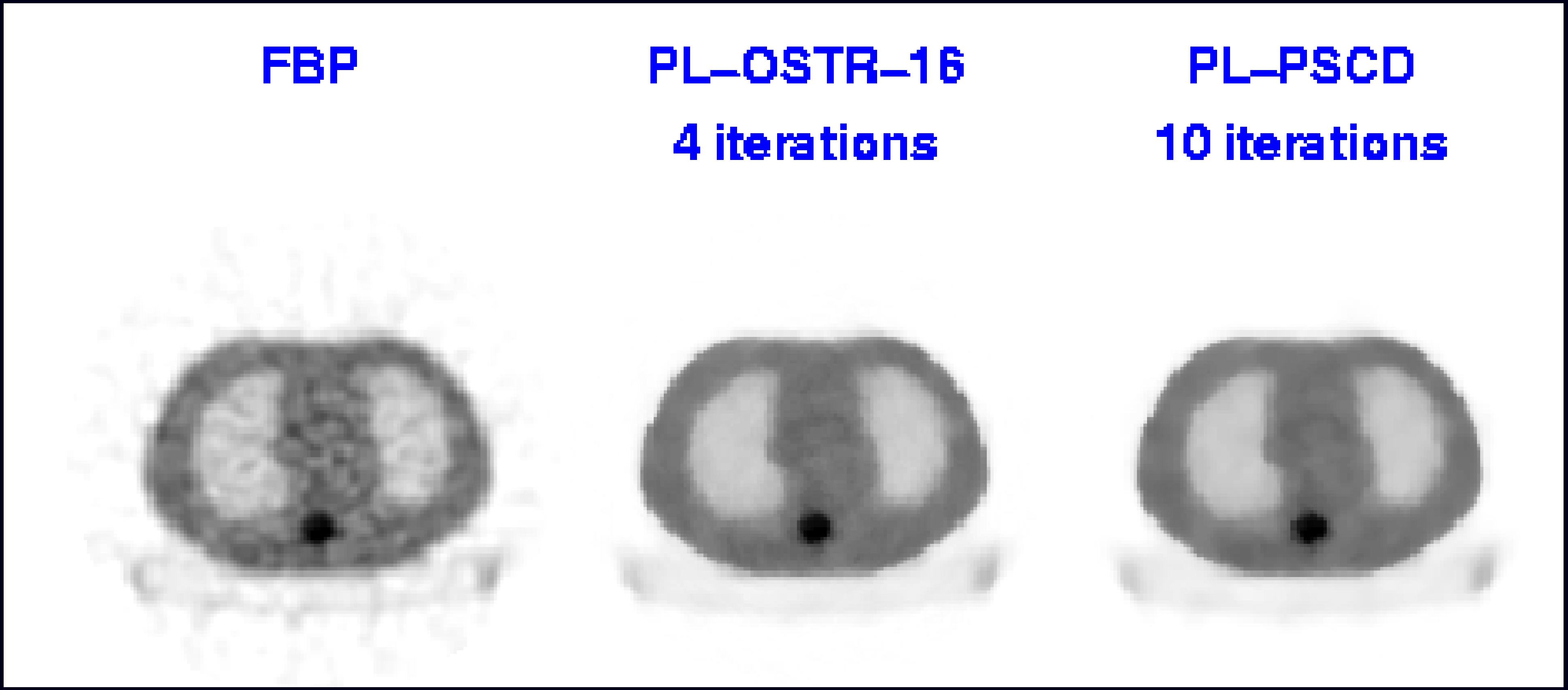
- ART
- RAMLA, BSREM (De Pierro, T-MI, 1997, 2001)
- Ahn and Fessler, NSS/MIC 2001

Proper relaxation can induce convergence, *but* still lacks monotonicity.
Choice of relaxation schedule requires experimentation.

Relaxed OS-SPS



OSTR



Ordered subsets version of separable paraboloidal surrogates
for PET transmission problem with nonquadratic convex *regularization*

Matlab m-file `http://www.eecs.umich.edu/~fessler`
`/code/transmission/tpl_osps.m`

Precomputed curvatures for OS-SPS

Separable Paraboloidal Surrogate (SPS) Algorithm:

$$x_j^{(n+1)} = \left[x_j^{(n)} - \frac{\sum_{i=1}^{n_d} a_{ij} \dot{h}_i([\boldsymbol{A}\boldsymbol{x}^{(n)}]_i)}{\sum_{i=1}^{n_d} a_{ij} |a|_i c_i^{(n)}} \right]_+, \qquad j = 1, \ldots, n_p$$

Ordered-subsets abandons monotonicity, so why use optimal curvatures $c_i^{(n)}$?

Precomputed curvature:

$$c_i = \ddot{h}_i(\hat{l}_i), \qquad \hat{l}_i = \arg\min_l h_i(l)$$

Precomputed denominator (saves one backprojection each iteration!):

$$d_j = \sum_{i=1}^{n_d} a_{ij} |a|_i c_i, \qquad j = 1, \ldots, n_p.$$

OS-SPS algorithm with M subsets:

$$x_j^{(n+1)} = \left[x_j^{(n)} - \frac{\sum_{i \in S^{(n)}} a_{ij} \dot{h}_i([\boldsymbol{A}\boldsymbol{x}^{(n)}]_i)}{d_j / M} \right]_+, \qquad j = 1, \ldots, n_p$$

Summary of Algorithms

- General-purpose optimization algorithms
- Optimization transfer for image reconstruction algorithms
- Separable surrogates $\Rightarrow$ high curvatures $\Rightarrow$ slow convergence
- Ordered subsets accelerate *initial* convergence
  require relaxation for true convergence
- Principles apply to emission and transmission reconstruction
- Still work to be done...