Object Motion and Image Reconstruction

James G. Nagy
Mathematics and Computer Science
Emory University
Atlanta, GA, USA

Collaborators

Julianne Chung	Tracy Faber
Piotr Wendykier	Nivedita Raghunath
James Herring	John Votaw
Lars Ruthotto	Per Christian Hansen

We usually assume relative

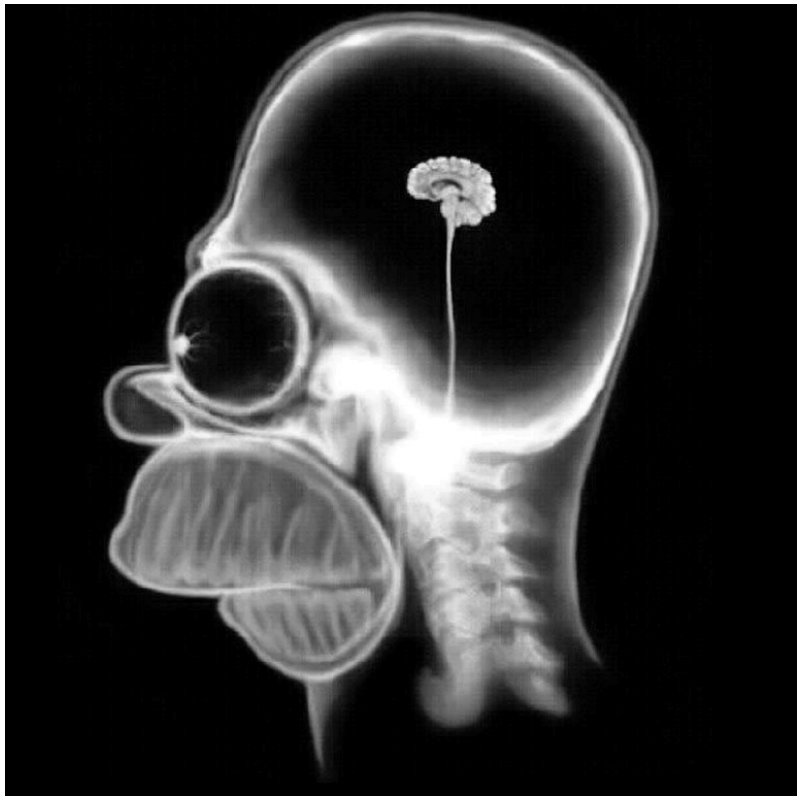
motion

between object and imaging device

is bad.

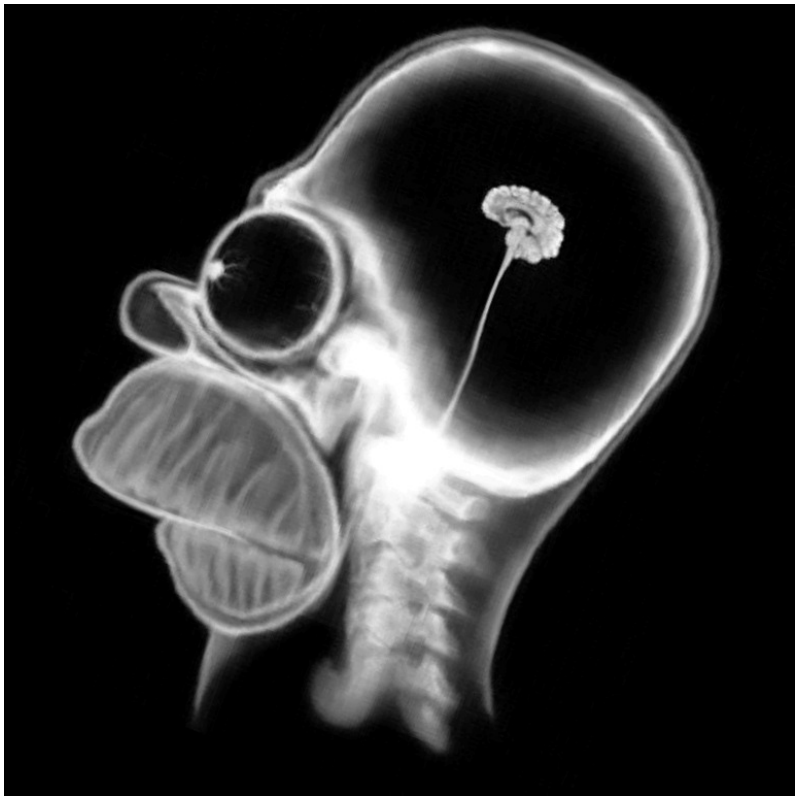
Motion Blur

- $\mathbf{x}_1 = \mathbf{A}(\mathbf{y}_1)\,\mathbf{x}$
  (object after movement)



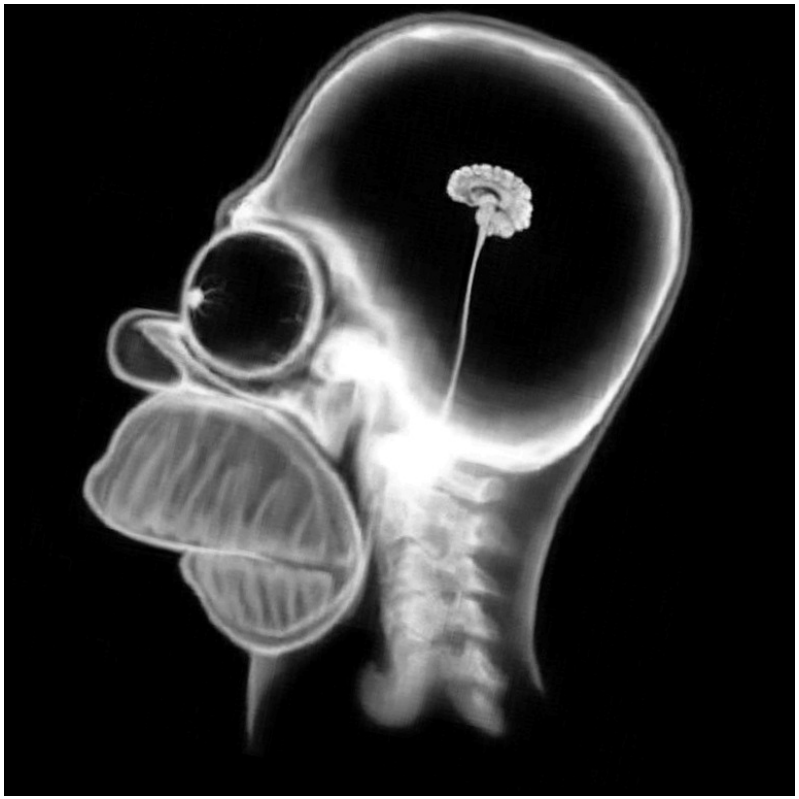
Motion Blur

- $\mathbf{x}_2 = \mathbf{A}(\mathbf{y}_2)\,\mathbf{x}$
  (object after movement)



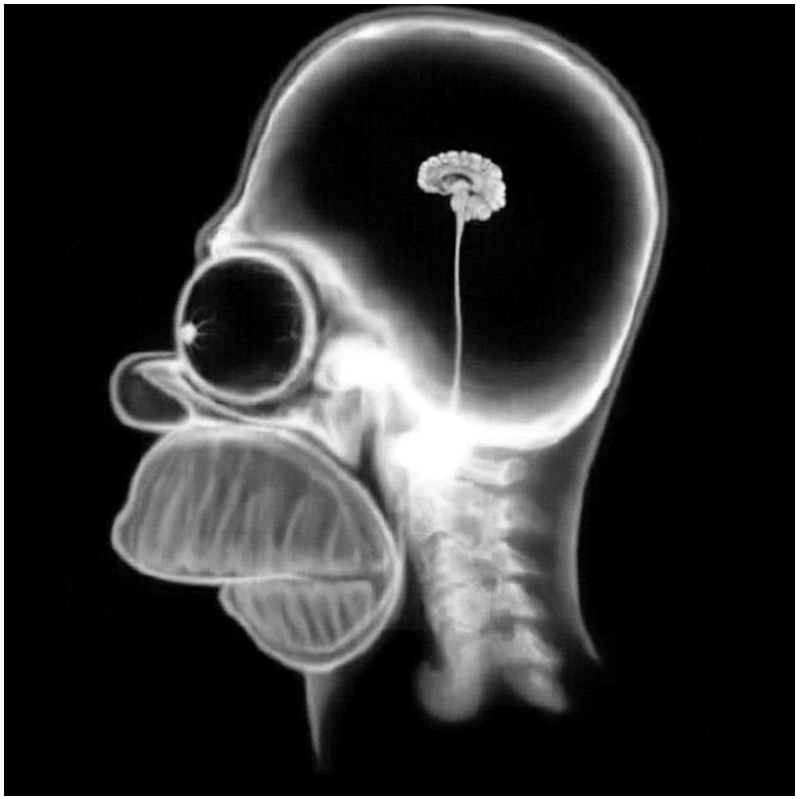
Motion Blur

- $\mathbf{x}_3 = \mathbf{A}(\mathbf{y}_3)\,\mathbf{x}$
  (object after movement)



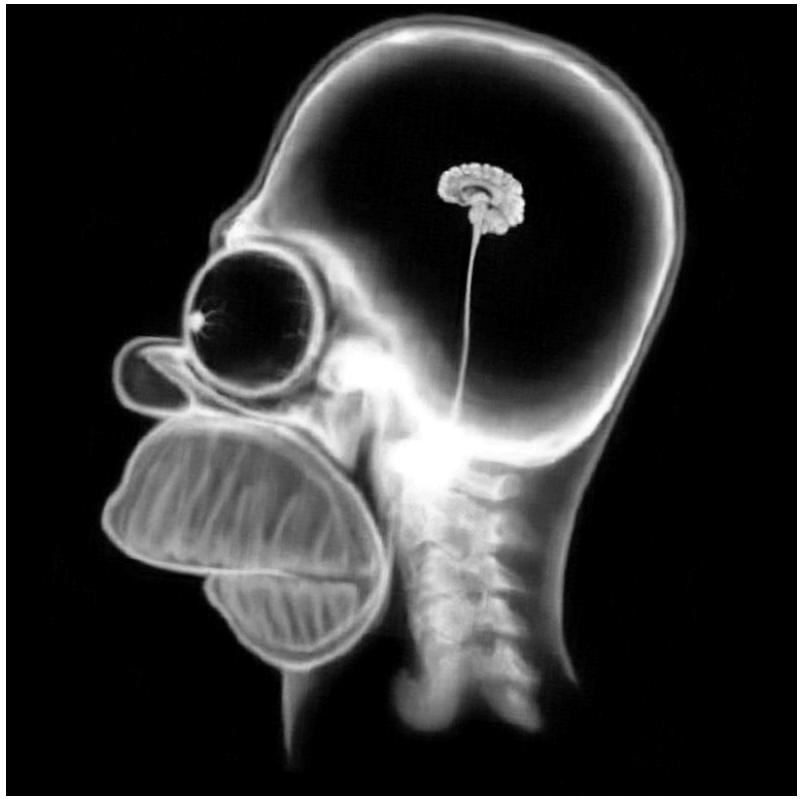
Motion Blur

- $\mathbf{x}_4 = \mathbf{A}(\mathbf{y}_4)\,\mathbf{x}$
  (object after movement)



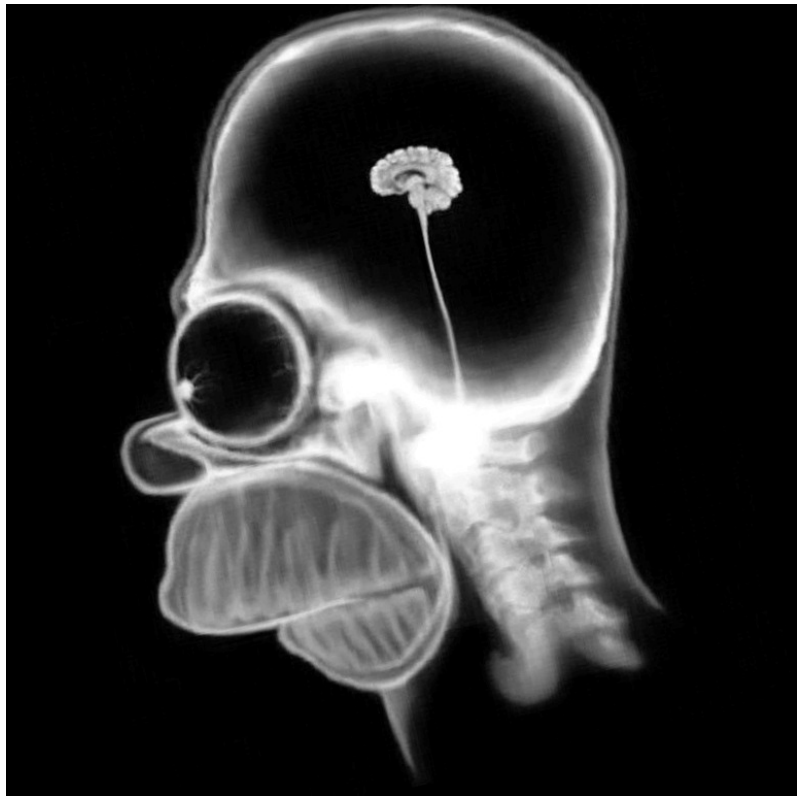
Motion Blur

- $\mathbf{x}_5 = \mathbf{A}(\mathbf{y}_5)\,\mathbf{x}$
  (object after movement)



Motion Blur

- $\mathbf{x}_6 = \mathbf{A}(\mathbf{y}_6)\,\mathbf{x}$
  (object after movement)



Motion Blur

- $\mathbf{x}_i = \mathbf{A}(\mathbf{y}_i)\,\mathbf{x}$
  (object after movement)

$$\mathbf{b}_{\text{true}} = \sum_{i=1}^{m} w_i \mathbf{x}_i \quad \text{(noise free)}$$

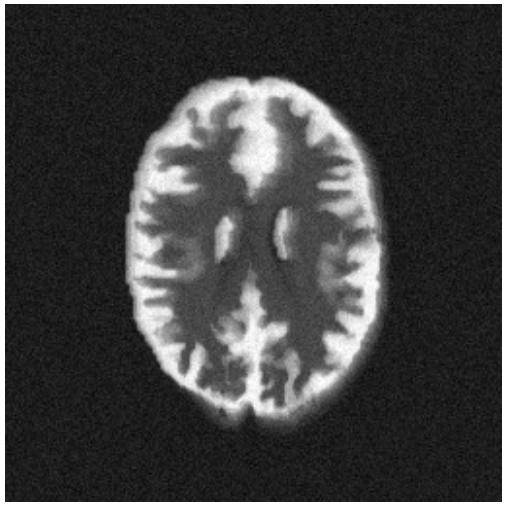
$$= \underbrace{\sum_{i=1}^{m} w_i \mathbf{A}(\mathbf{y}_i)}_{\mathbf{A}(\mathbf{y})}\,\mathbf{x}$$

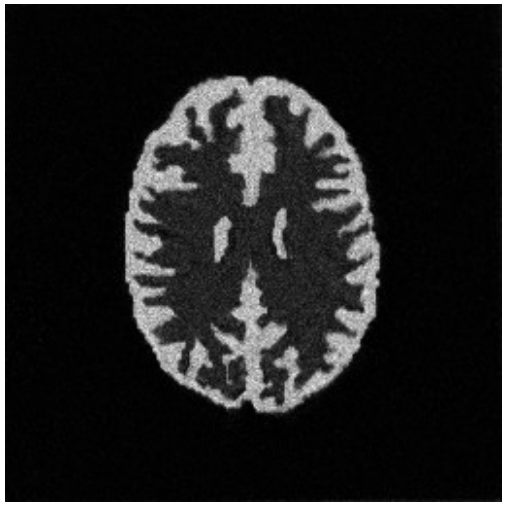
$$\mathbf{b} = \mathbf{A}(\mathbf{y})\,\mathbf{x} + \text{noise}$$

- $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m)$
  $=$ registration parameters
- Goal: Improve parameters $\mathbf{y}$ and compute $\mathbf{x}$

Application: Patient Motion in Brain Imaging (PET, MRI)

Head movements during long scan time cause blur.





Patient Motion in Brain Imaging

To correct for motion artifacts, we need:

- Motion information, e.g., registration parameters

$$\mathbf{y}$$

- Mathematical function that relates motion distorted data to pristine data, e.g.

$$\mathbf{b} = \mathbf{A}(\mathbf{y})\mathbf{x} + \mathbf{e}$$

- Optimization method to "undo" the motion distortion

  Often need to include constraints and/or regularization

But sometimes relative

motion

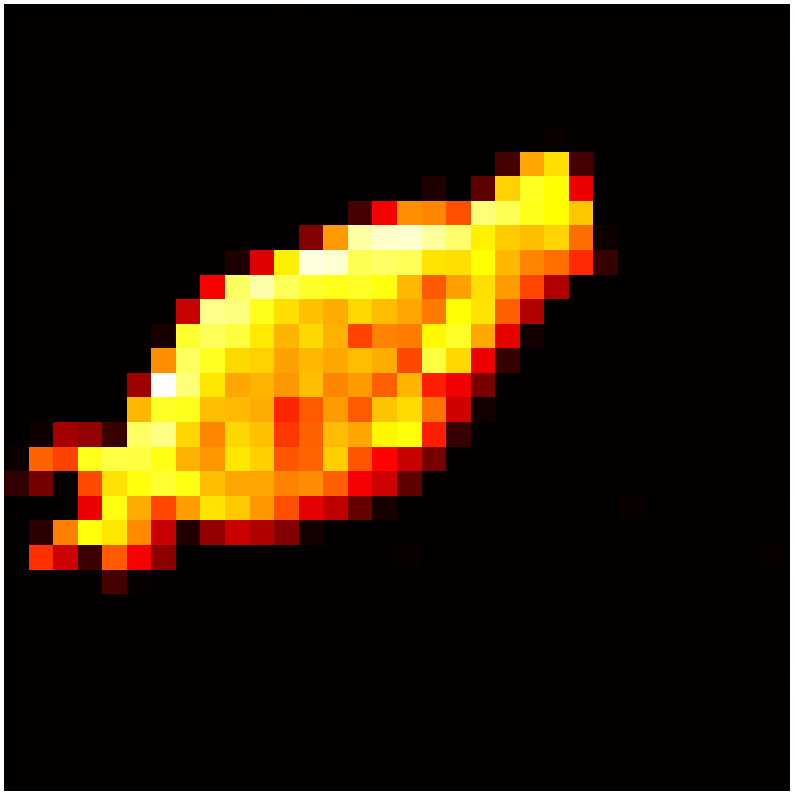
between object and imaging device

is good.

Application: Image Digital Super-Resolution

- $\mathbf{b}_1 = \mathbf{A}(\mathbf{y}_1)\,\mathbf{x} + \mathbf{e}_1$
  (collected low resolution images)

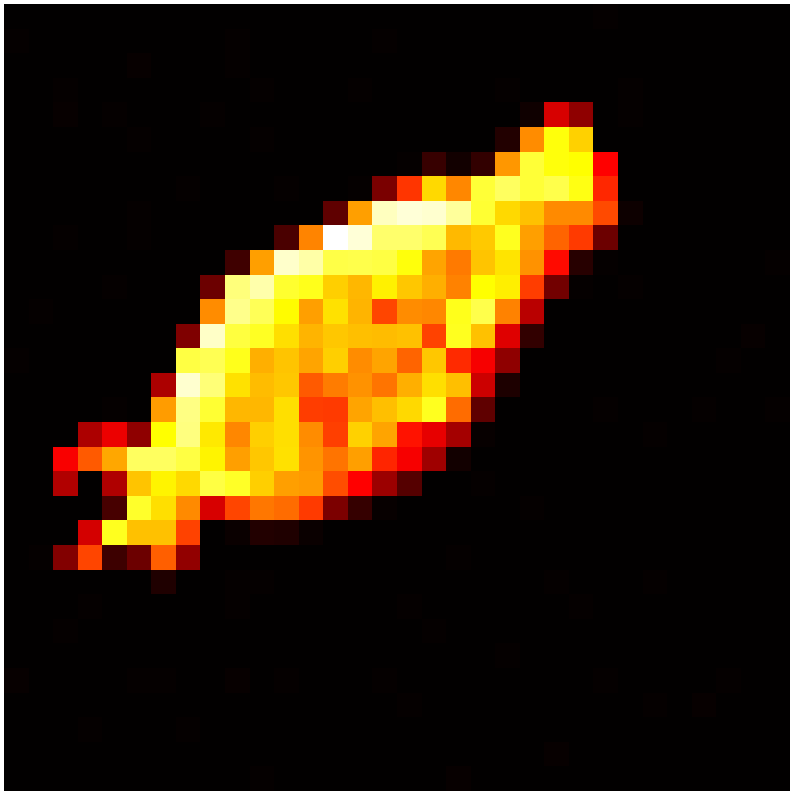
1−th low resolution image



Application: Image Digital Super-Resolution

- $\mathbf{b}_8 = \mathbf{A}(\mathbf{y}_8)\,\mathbf{x} + \mathbf{e}_8$
  (collected low resolution images)

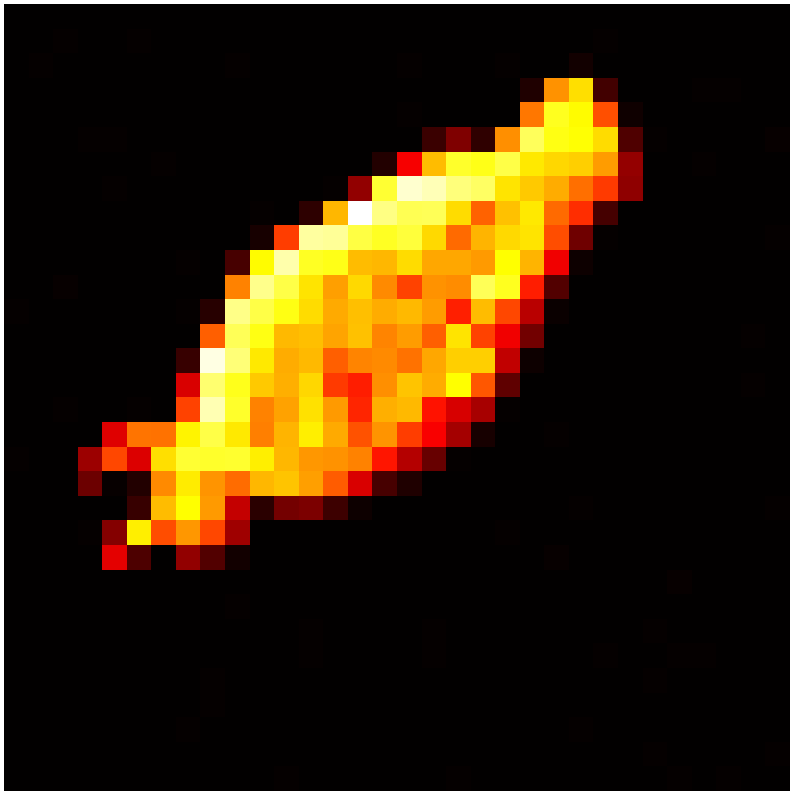
8−th low resolution image



Application: Image Digital Super-Resolution

- $\mathbf{b}_{15} = \mathbf{A}(\mathbf{y}_{15})\,\mathbf{x} + \mathbf{e}_{15}$
  (collected low resolution images)

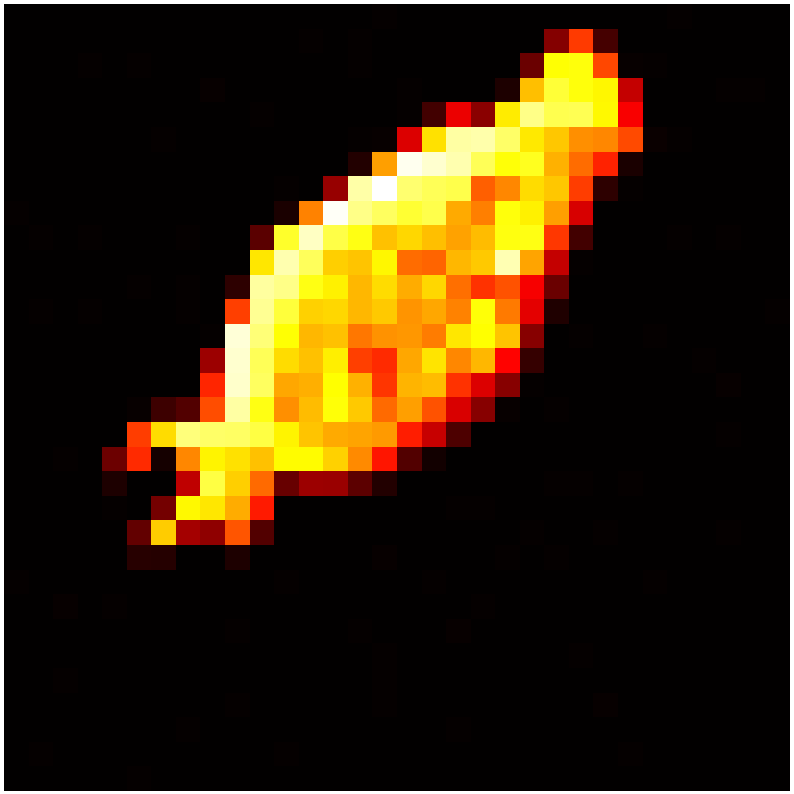
15−th low resolution image



Application: Image Digital Super-Resolution

- $\mathbf{b}_{22} = \mathbf{A}(\mathbf{y}_{22})\,\mathbf{x} + \mathbf{e}_{22}$
  (collected low resolution images)

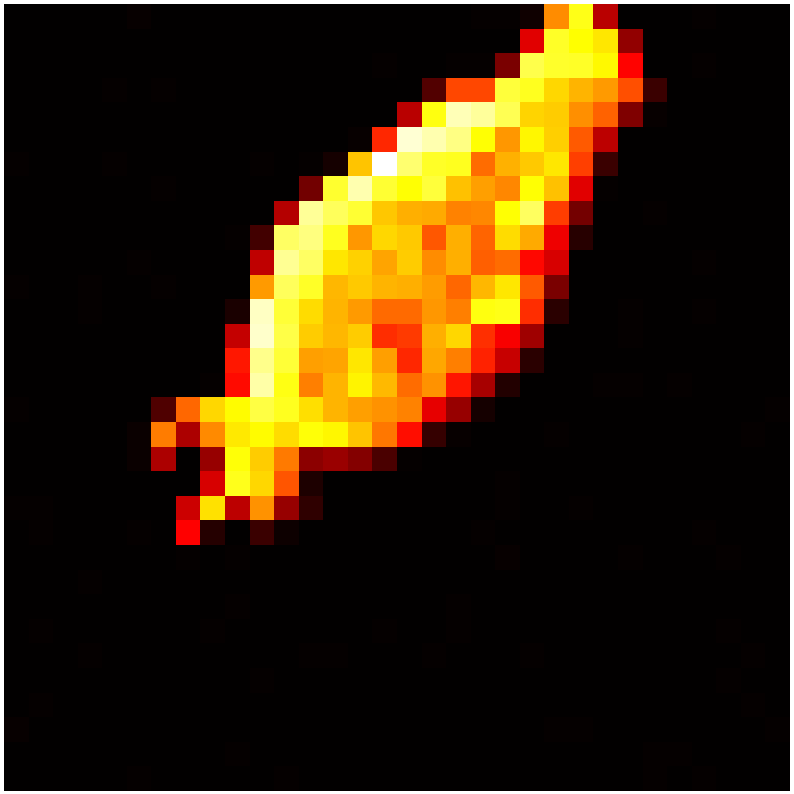
22−th low resolution image



Application: Image Digital Super-Resolution

- $\mathbf{b}_{29} = \mathbf{A}(\mathbf{y}_{29})\,\mathbf{x} + \mathbf{e}_{29}$
  (collected low resolution images)

29−th low resolution image

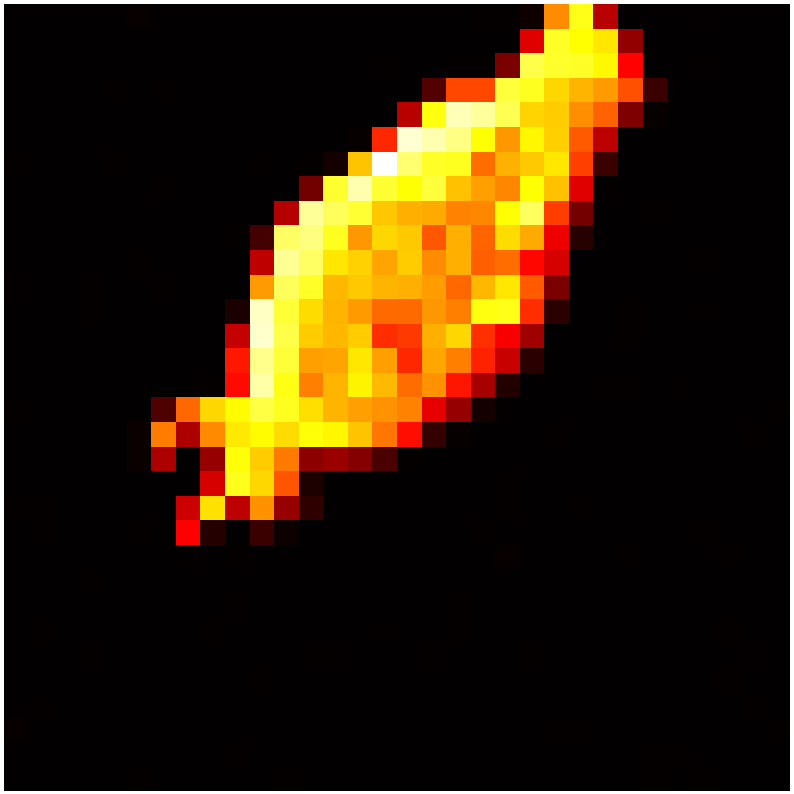


Application: Image Digital Super-Resolution

- $\mathbf{b}_j = \mathbf{A}(\mathbf{y}_j)\,\mathbf{x} + \mathbf{e}_j$
  (collected low resolution images)

$$\underbrace{\begin{bmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_m \end{bmatrix}}_{\mathbf{b}} = \underbrace{\begin{bmatrix} \mathbf{A}(\mathbf{y}_1) \\ \vdots \\ \mathbf{A}(\mathbf{y}_m) \end{bmatrix}}_{\mathbf{A}(\mathbf{y})} \mathbf{x} + \underbrace{\begin{bmatrix} \mathbf{e}_1 \\ \vdots \\ \mathbf{e}_m \end{bmatrix}}_{\mathbf{e}}$$

29−th low resolution image



- $\mathbf{y}$ = registration parameters
- Goal: Improve parameters $\mathbf{y}$ and compute $\mathbf{x}$

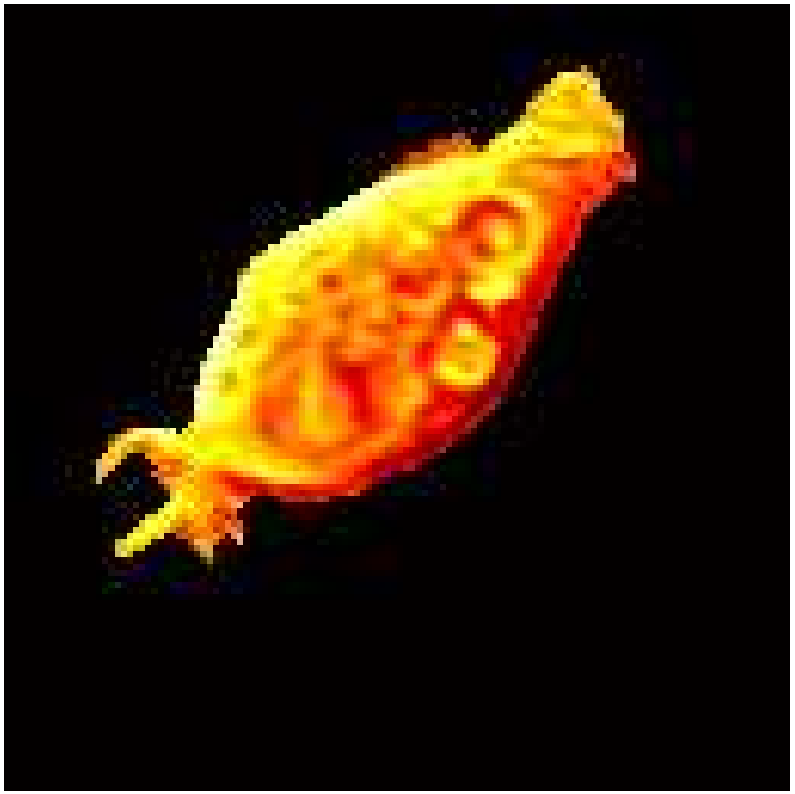
Application: Image Digital Super-Resolution

- $\mathbf{b}_j = \mathbf{A}(\mathbf{y}_j)\,\mathbf{x} + \mathbf{e}_j$
  (collected low resolution images)

$$\underbrace{\begin{bmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_m \end{bmatrix}}_{\mathbf{b}} = \underbrace{\begin{bmatrix} \mathbf{A}(\mathbf{y}_1) \\ \vdots \\ \mathbf{A}(\mathbf{y}_m) \end{bmatrix}}_{\mathbf{A}(\mathbf{y})} \mathbf{x} + \underbrace{\begin{bmatrix} \mathbf{e}_1 \\ \vdots \\ \mathbf{e}_m \end{bmatrix}}_{\mathbf{e}}$$

- $\mathbf{y}$ = registration parameters
- Goal: Improve parameters $\mathbf{y}$ and compute $\mathbf{x}$

Reconstructed high resolution image



Outline

1. The Linear Problem: $\mathbf{b} = \mathbf{A}\mathbf{x} + \mathbf{e}$
2. Application: Motion Correction in Brain Imaging
3. The Nonlinear Problem: $\mathbf{b} = \mathbf{A}(\mathbf{y})\,\mathbf{x} + \mathbf{e}$
4. Examples
5. Concluding Remarks

The Linear Problem

Assume $\mathbf{A} = \mathbf{A}(\mathbf{y})$ is known exactly.

- We are given $\mathbf{A}$ and $\mathbf{b}$, where

$$\mathbf{b} = \mathbf{A}\mathbf{x} + \mathbf{e}$$

- $\mathbf{A}$ is an ill-conditioned matrix, and we do not know $\mathbf{e}$.
- We want to compute an approximation of $\mathbf{x}$.

- The approximation $\mathbf{x}_{\text{inv}} \approx \mathbf{A}^{-1}\mathbf{b}$ is usually very bad.
- Need regularization.

Regularization

Basic Idea: Instead of computing $\mathbf{x}_{\text{inv}} = \mathbf{A}^{-1}\mathbf{b}$, use:

$$\mathbf{x}_{\text{reg}} = \mathbf{A}_{\text{reg}}^{-1}\mathbf{b}$$

so that

$$\begin{aligned} \hat{\mathbf{x}} &= \mathbf{A}_{\text{reg}}^{-1}\mathbf{b} \\ &= \mathbf{A}_{\text{reg}}^{-1}(\mathbf{A}\mathbf{x} + \mathbf{e}) \\ &= \mathbf{A}_{\text{reg}}^{-1}\mathbf{A}\mathbf{x} + \mathbf{A}_{\text{reg}}^{-1}\mathbf{e} \end{aligned}$$

where

$$\mathbf{A}_{\text{reg}}^{-1}\mathbf{A}\mathbf{x} \approx \mathbf{x} \qquad \text{and} \qquad \mathbf{A}_{\text{reg}}^{-1}\mathbf{e} \quad \text{is not too large}$$

Regularization by Filtering

Some examples:

- TSVD: If we can compute the SVD, $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$,

$$\mathbf{x}_{\text{reg}} = \mathbf{V}\mathbf{\Sigma}_k^{\dagger}\mathbf{U}^T\mathbf{b} = \sum_{i=1}^{k} \frac{\mathbf{u}_i^T\mathbf{b}}{\sigma_i}\mathbf{v}_i \,,$$

- Tikhonov

$$\min_{\mathbf{x}} \left\{ \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda^2 \|\mathbf{x}\|_2^2 \right\} \quad \Leftrightarrow \quad \min_{\mathbf{x}} \left\| \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix} - \begin{bmatrix} \mathbf{A} \\ \lambda\mathbf{I} \end{bmatrix} \mathbf{x} \right\|_2^2$$

- Iterative (e.g., LSQR)
  - Apply LSQR to the (unregularized) problem, $\min \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2$
  - Stop iteration early, when solution is good.

Choosing Regularization Parameters

In each case need to choose regularization parameter:

- TSVD – choose truncation index.
- Tikhonov – choose λ
- Iterative – choose stopping iteration.

Lots of choices: Generalized Cross Validation (GCV), L-curve, discrepancy principle, ...

Choosing Regularization Parameters

In each case need to choose regularization parameter:

- TSVD – choose truncation index.
- Tikhonov – choose λ
- Iterative – choose stopping iteration.

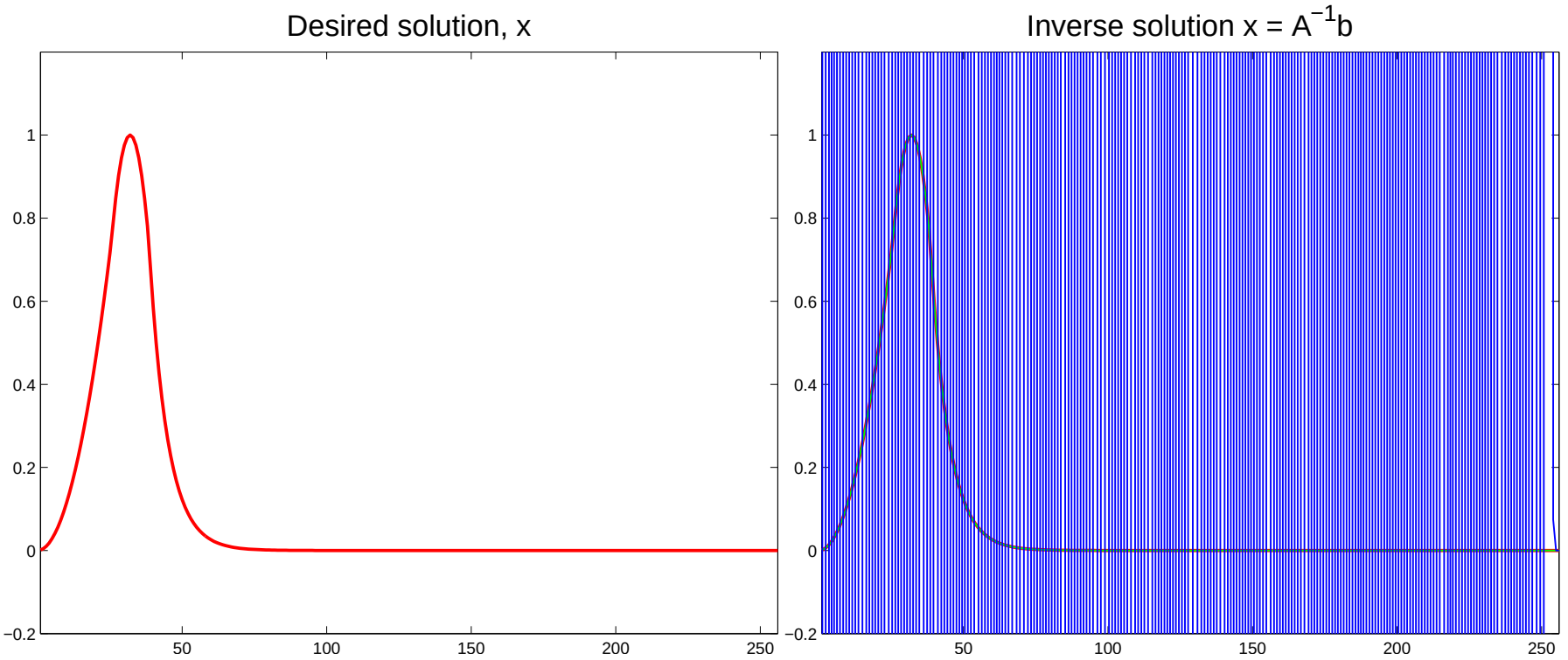
Lots of choices: Generalized Cross Validation (GCV), L-curve, discrepancy principle, ...

GCV and Tikhonov: Choose λ to minimize

$$\mathrm{GCV}(\lambda) = \frac{n \displaystyle\sum_{i=1}^{n} \left(\frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i^2 + \lambda^2} \right)^2}{\left(\displaystyle\sum_{i=1}^{n} \frac{1}{\sigma_i^2 + \lambda^2} \right)^2}$$

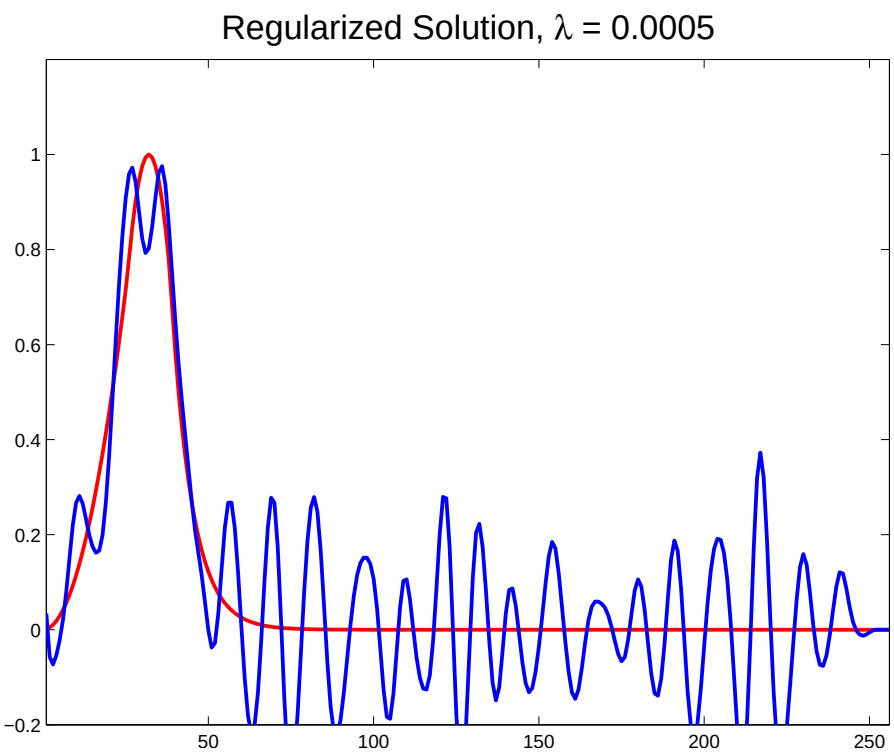
Example: Inverse Heat Equation

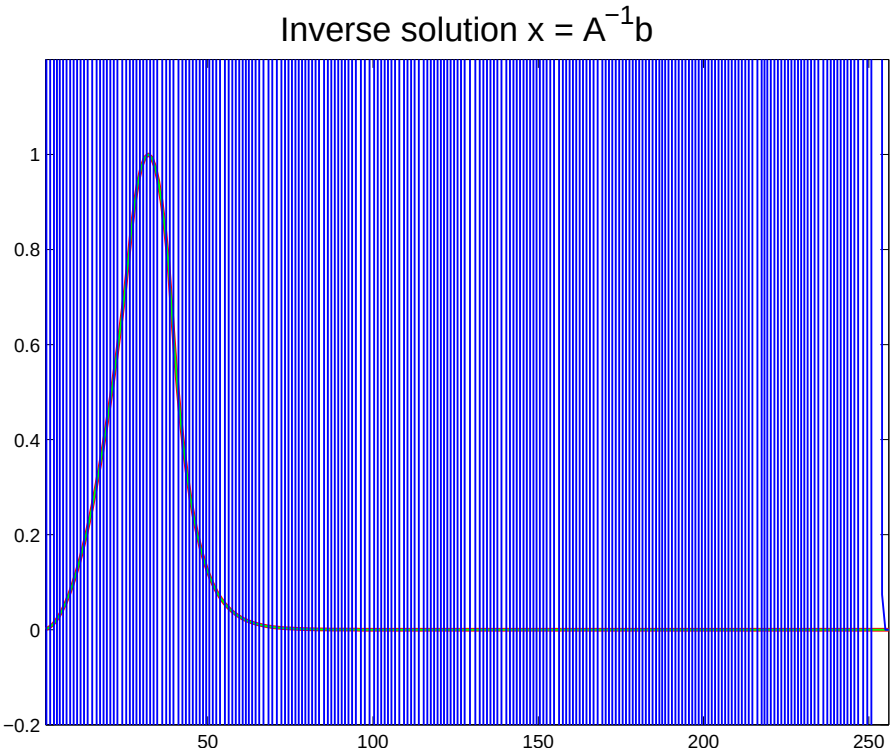
Reconstruction using Tikhonov reg. can be better than $\mathbf{x}_{\text{inv}}$.
Quality of reconstruction depends on λ.
But λ depends on $\mathbf{A}$ and $\mathbf{b}$.



Example: Inverse Heat Equation

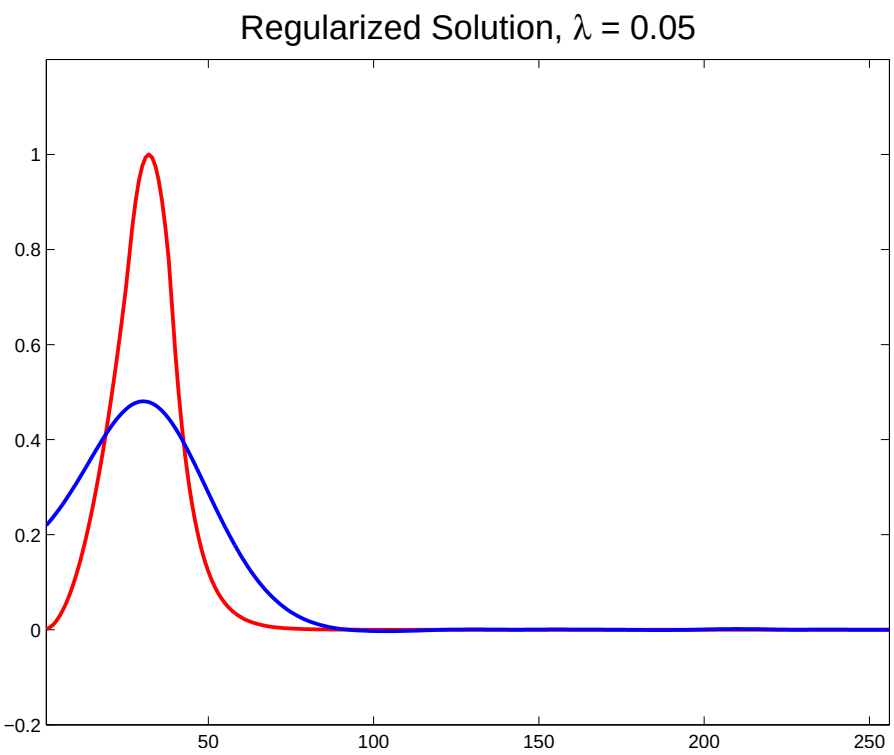
Reconstruction using Tikhonov reg. can be better than $\mathbf{x}_{\text{inv}}$.
Quality of reconstruction depends on λ.
But λ depends on $\mathbf{A}$ and $\mathbf{b}$.

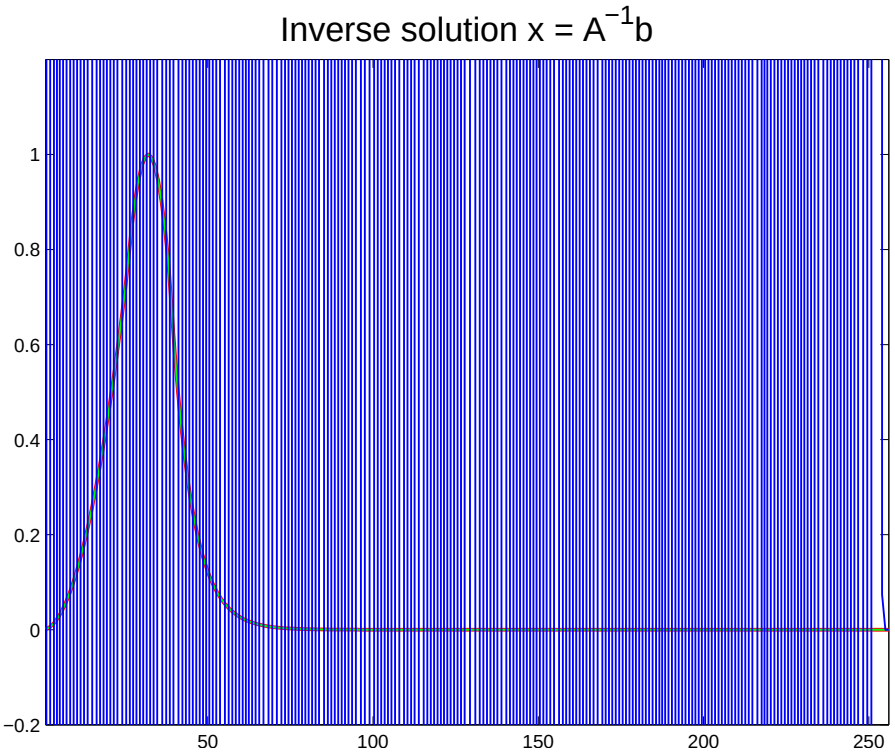




Example: Inverse Heat Equation

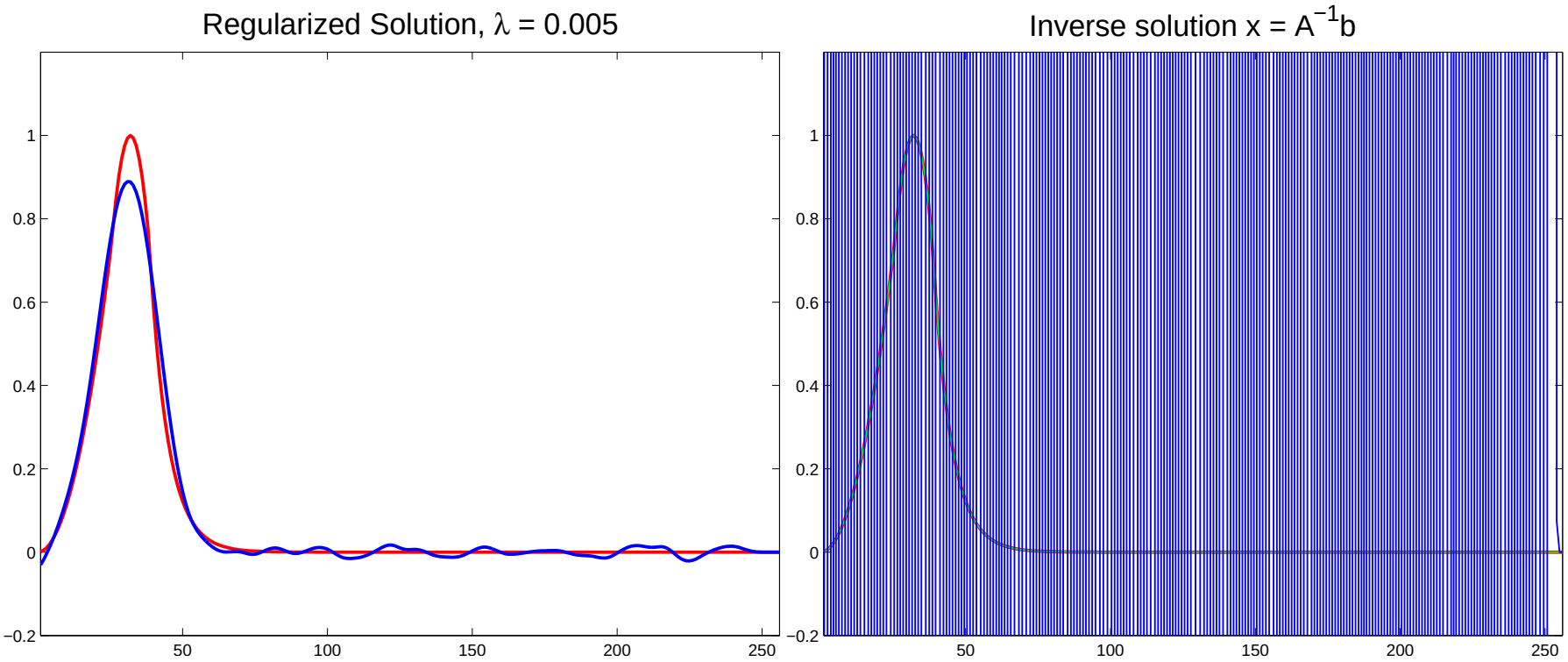
Reconstruction using Tikhonov reg. can be better than $\mathbf{x}_{\text{inv}}$.
Quality of reconstruction depends on λ.
But λ depends on $\mathbf{A}$ and $\mathbf{b}$.





Example: Inverse Heat Equation

Reconstruction using Tikhonov reg. can be better than $\mathbf{x}_{\text{inv}}$.
Quality of reconstruction depends on λ.
But λ depends on $\mathbf{A}$ and $\mathbf{b}$.



Filtering for Large Scale Problems

Some remarks:

- For large matrices, computing SVD is expensive.
- SVD algorithms do not readily simplify for structured or sparse matrices.
- Alternative for large scale problems: LSQR iteration
  (Paige and Saunders, ACM TOMS, 1982)

Lanczos Bidiagonalization (LBD)

Given $\mathbf{A}$ and $\mathbf{b}$, for $k = 1, 2, ...,$ compute

- $\mathbf{W}_k = \begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 & \cdots & \mathbf{w}_k & \mathbf{w}_{k+1} \end{bmatrix}, \quad \mathbf{w}_1 = \mathbf{b}/||\mathbf{b}||$
- $\mathbf{Z}_k = \begin{bmatrix} \mathbf{z}_1 & \mathbf{z}_2 & \cdots & \mathbf{z}_k \end{bmatrix}$
- $\mathbf{B}_k = \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_k & \alpha_k \\ & & & \beta_{k+1} \end{bmatrix}$

where $\mathbf{W}_k$ and $\mathbf{Z}_k$ have orthonormal columns, and

$$\begin{aligned} \mathbf{A}^T\mathbf{W}_k &= \mathbf{Z}_k\mathbf{B}_k^T + \alpha_{k+1}\mathbf{z}_{k+1}\mathbf{e}_{k+1}^T \\ \mathbf{A}\mathbf{Z}_k &= \mathbf{W}_k\mathbf{B}_k \end{aligned}$$

LBD and LSQR

At kth LBD iteration, use QR to solve *projected* LS problem:

$$\min_{\mathbf{x}\in R(\mathbf{Z}_k)} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 = \min_{\mathbf{f}} \|\mathbf{W}_k^T\mathbf{b} - \mathbf{B}_k\mathbf{f}\|_2^2 = \min_{\mathbf{f}} \|\beta\mathbf{e}_1 - \mathbf{B}_k\mathbf{f}\|_2^2$$

where $\mathbf{x}_k = \mathbf{Z}_k\mathbf{f}$

For our ill-posed inverse problems:

- Singular values of $\mathbf{B}_k$ converge to k largest sing. values of $\mathbf{A}$.
- Thus, $\mathbf{x}_k$ is in a subspace that approximates a subspace spanned by the large singular components of $\mathbf{A}$.
  - For $k < n$, $\mathbf{x}_k$ is a regularized solution.
  - $\mathbf{x}_n = \mathbf{x}_{\text{inv}} = \mathbf{A}^{-1}\mathbf{b}$ (bad approximation)

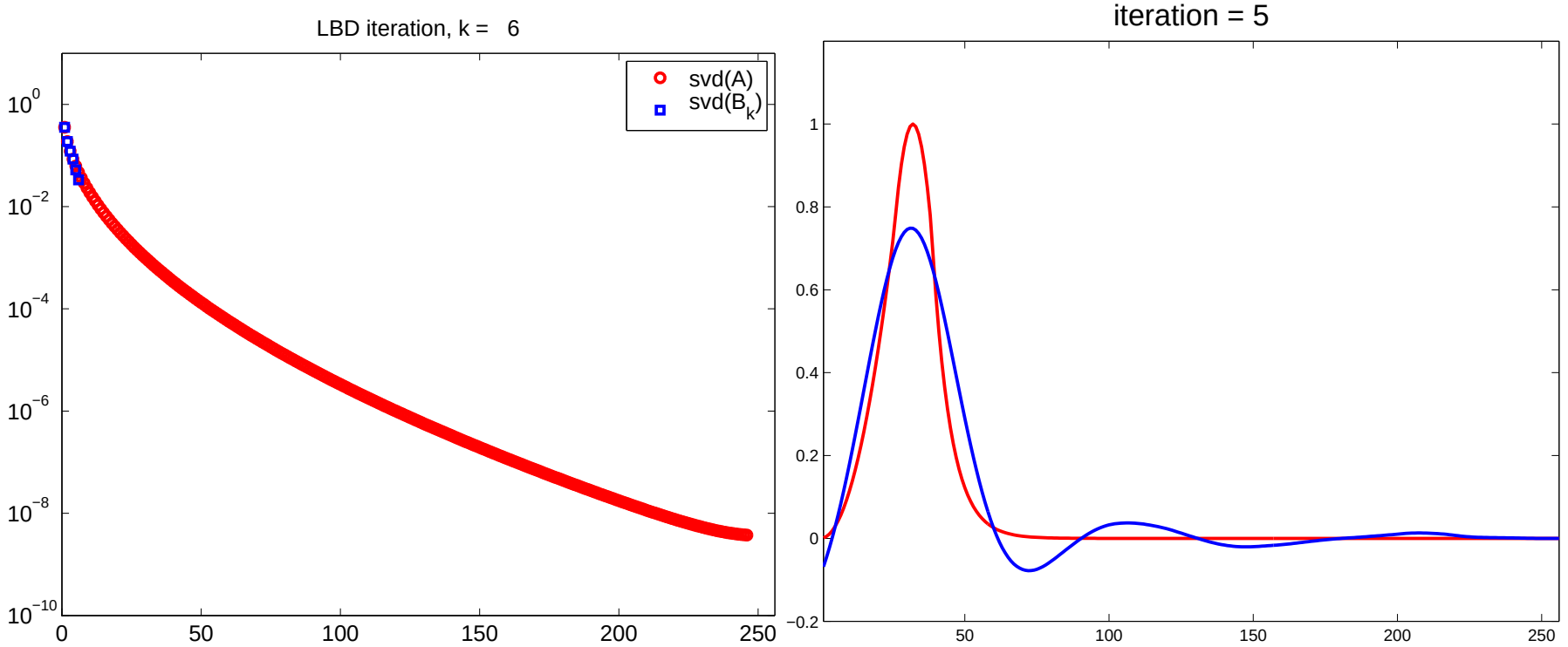
Example: Inverse Heat Equation

Singular values of $\mathbf{B}_k$ converge to large singular values of $\mathbf{A}$.

Thus, for early iterations k: $\quad \mathbf{f} = \mathbf{B}_k \setminus \mathbf{W}_k \mathbf{b}$

$$\mathbf{x}_k = \mathbf{Z}_k \mathbf{f}$$

is a regularized reconstruction.



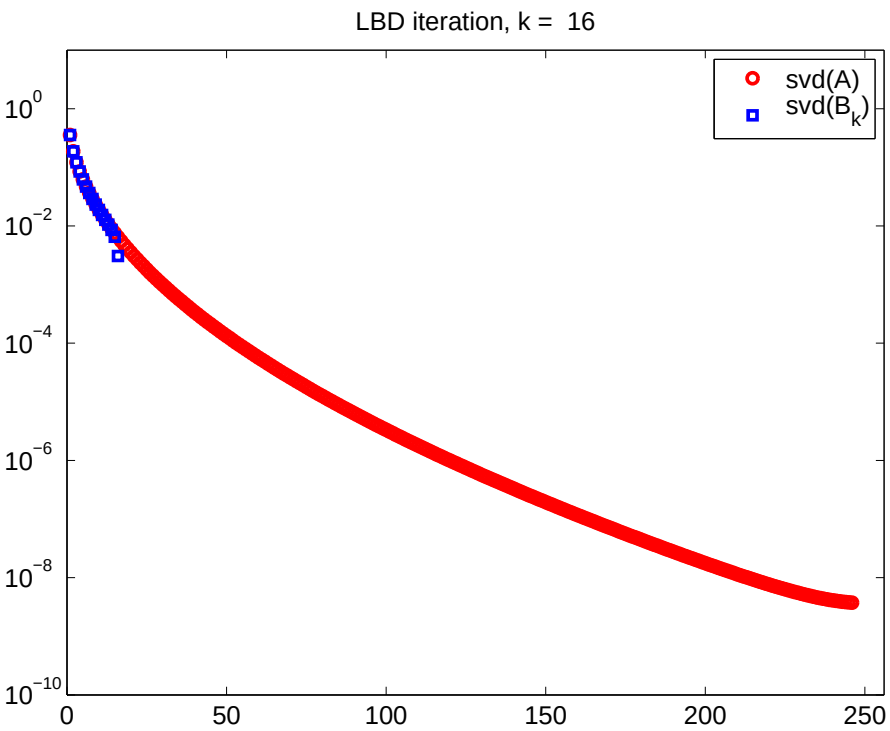
Example: Inverse Heat Equation

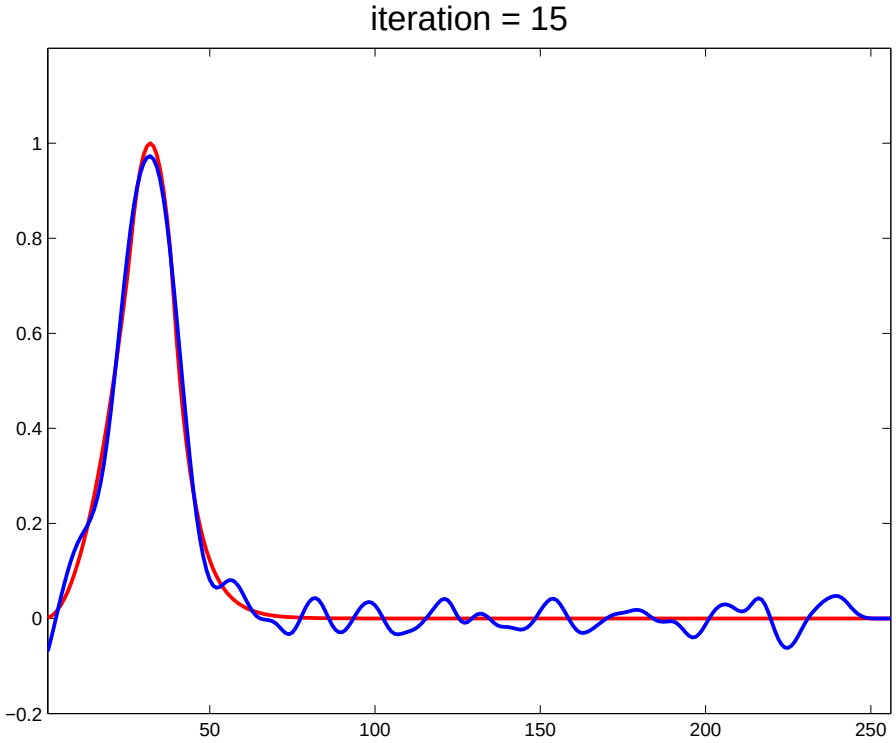
Singular values of $\mathbf{B}_k$ converge to large singular values of $\mathbf{A}$.

Thus, for early iterations k: $\mathbf{f} = \mathbf{B}_k \setminus \mathbf{W}_k\mathbf{b}$

$\mathbf{x}_k = \mathbf{Z}_k\mathbf{f}$

is a regularized reconstruction.





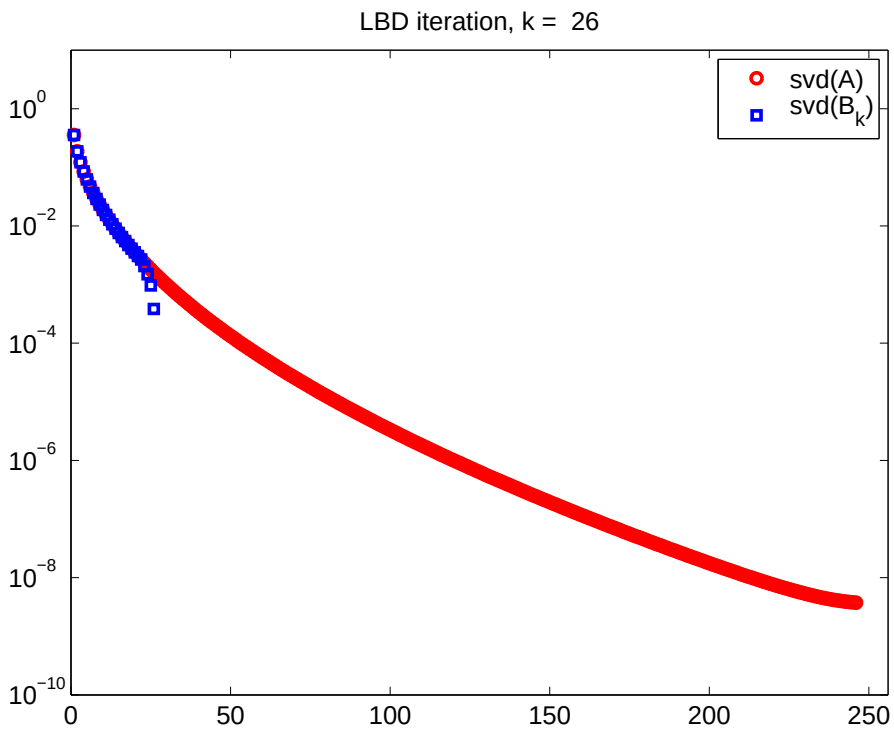
Example: Inverse Heat Equation

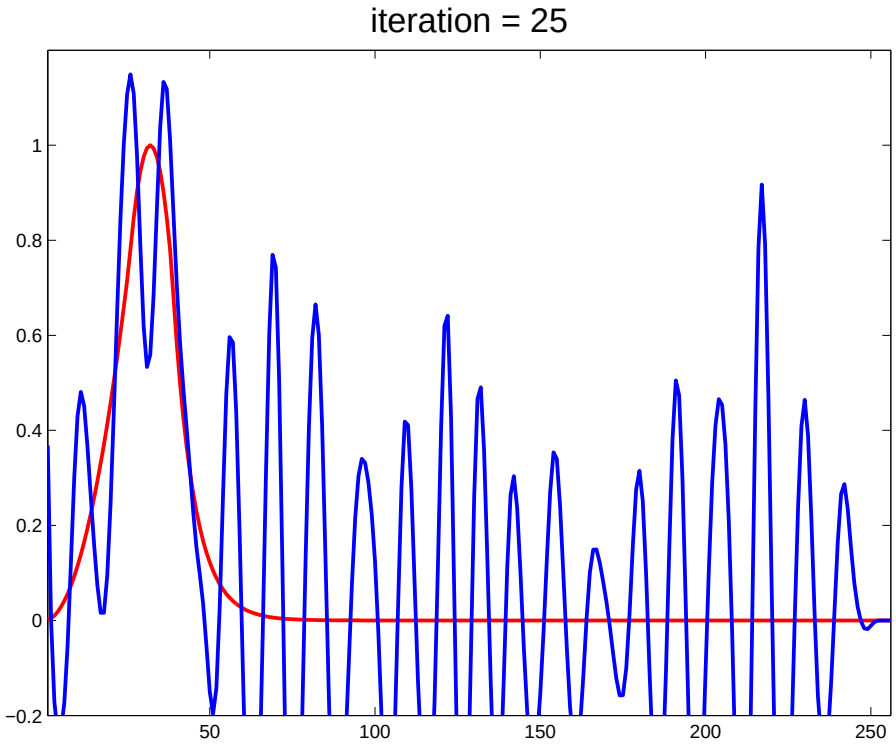
Singular values of $\mathbf{B}_k$ converge to large singular values of $\mathbf{A}$.

Thus, for later iterations k: $\mathbf{f} = \mathbf{B}_k \setminus \mathbf{W}_k\mathbf{b}$

$$\mathbf{x}_k = \mathbf{Z}_k\mathbf{f}$$

is a noisy reconstruction.





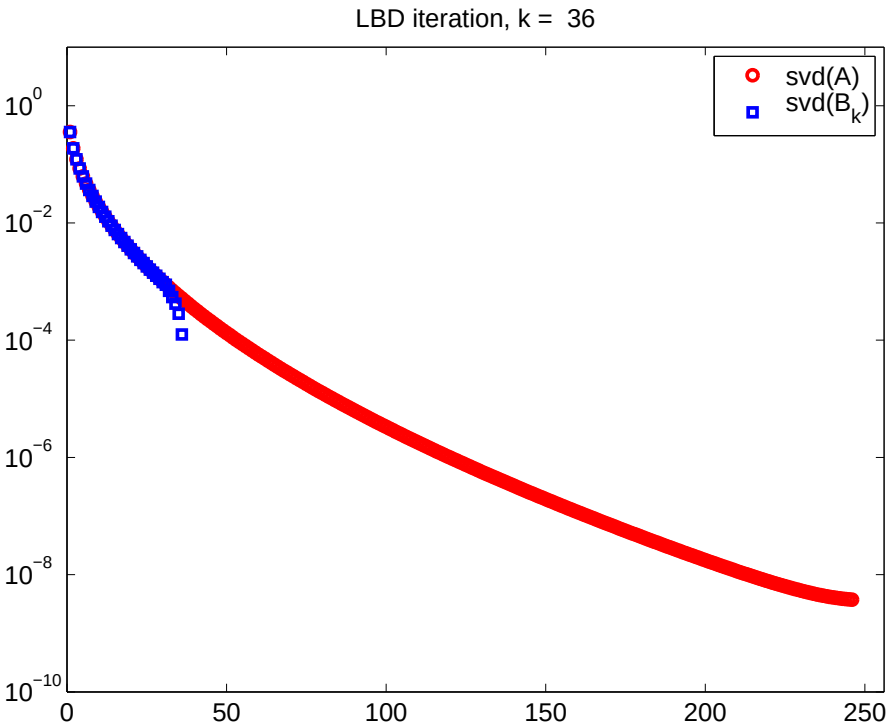
Example: Inverse Heat Equation

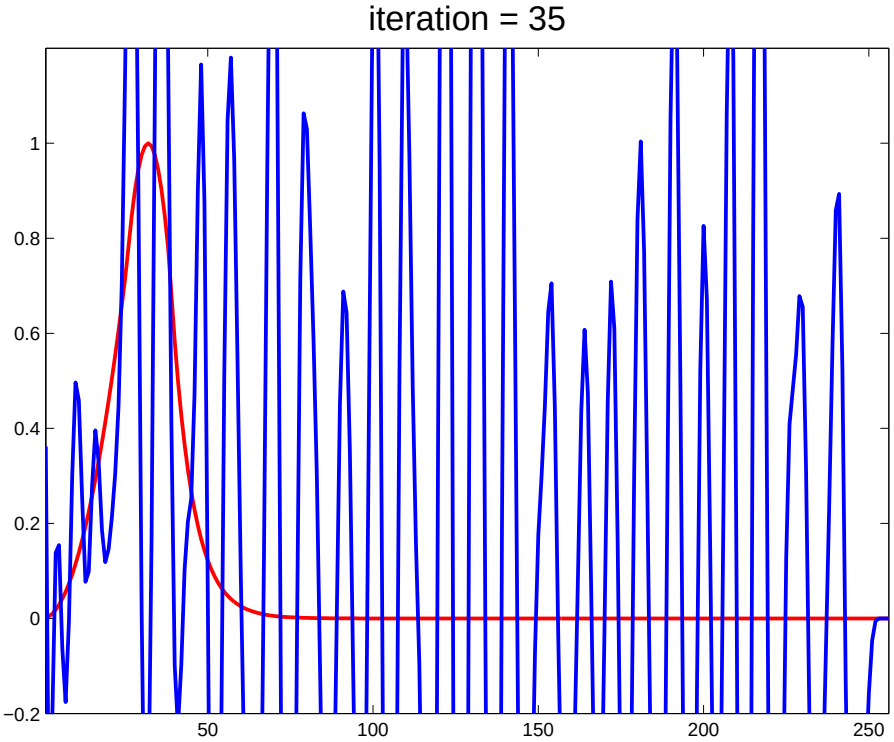
Singular values of $\mathbf{B}_k$ converge to large singular values of $\mathbf{A}$.

Thus, for later iterations k: $\mathbf{f} = \mathbf{B}_k \setminus \mathbf{W}_k\mathbf{b}$

$$\mathbf{x}_k = \mathbf{Z}_k\mathbf{f}$$

is a noisy reconstruction.





Lanczos Based Hybrid Methods

To avoid noisy reconstructions, embed regularization in LBD:

- O'Leary and Simmons, SISSC, 1981.
- Björck, BIT 1988.
- Björck, Grimme, and Van Dooren, BIT, 1994.
- Larsen, PhD Thesis, 1998.
- Hanke, BIT 2001.
- Kilmer and O'Leary, SIMAX, 2001.
- Kilmer, Hansen, Español, SISC 2007.
- Chung, N, O'Leary, ETNA 2007
  (HyBR Implementation)

Regularize the Projected Least Squares Problem

To stabilize convergence, regularize the projected problem:

$$\min_{\mathbf{f}} \left\| \begin{bmatrix} \beta \mathbf{e}_1 \\ \mathbf{0} \end{bmatrix} - \begin{bmatrix} \mathbf{B}_k \\ \lambda \mathbf{I} \end{bmatrix} \mathbf{f} \right\|_2^2$$

Note: $\mathbf{B}_k$ is very small compared to $\mathbf{A}$, so

- Can use "expensive" methods to choose λ (e.g., GCV)
- Can also use GCV information to estimate stopping iteration (Björck, Grimme, and Van Dooren, BIT, 1994).

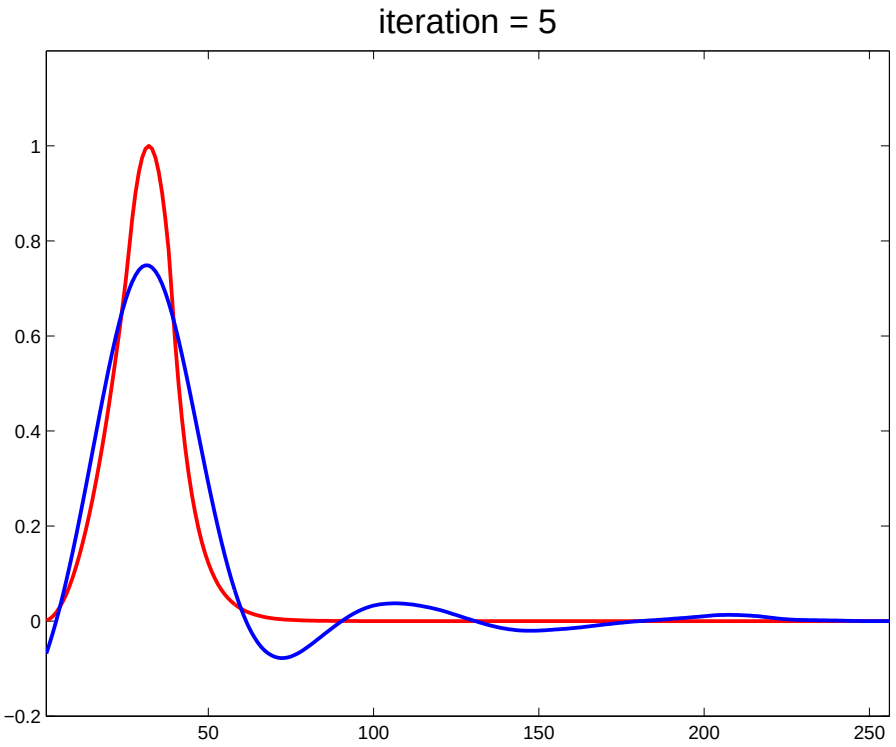
Example: Inverse Heat Equation

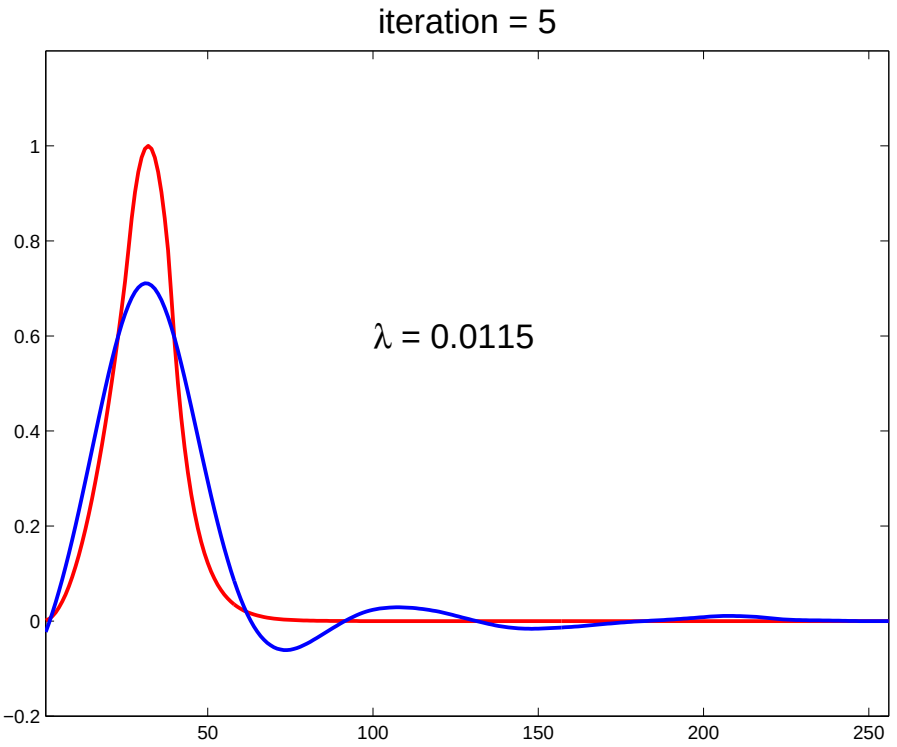
LSQR (no regularization)

$$\mathbf{f} = \mathbf{B}_k \setminus \mathbf{W}_k \mathbf{b}$$
$$\mathbf{x}_k = \mathbf{Z}_k \mathbf{f}$$

HyBR (Tikhonov regularization)

$$\mathbf{f} = \begin{bmatrix} \mathbf{B}_k \\ \lambda_k \mathbf{I} \end{bmatrix} \setminus \begin{bmatrix} \mathbf{W}_k \mathbf{b} \\ \mathbf{0} \end{bmatrix}$$
$$\mathbf{x}_k = \mathbf{Z}_k \mathbf{f}$$





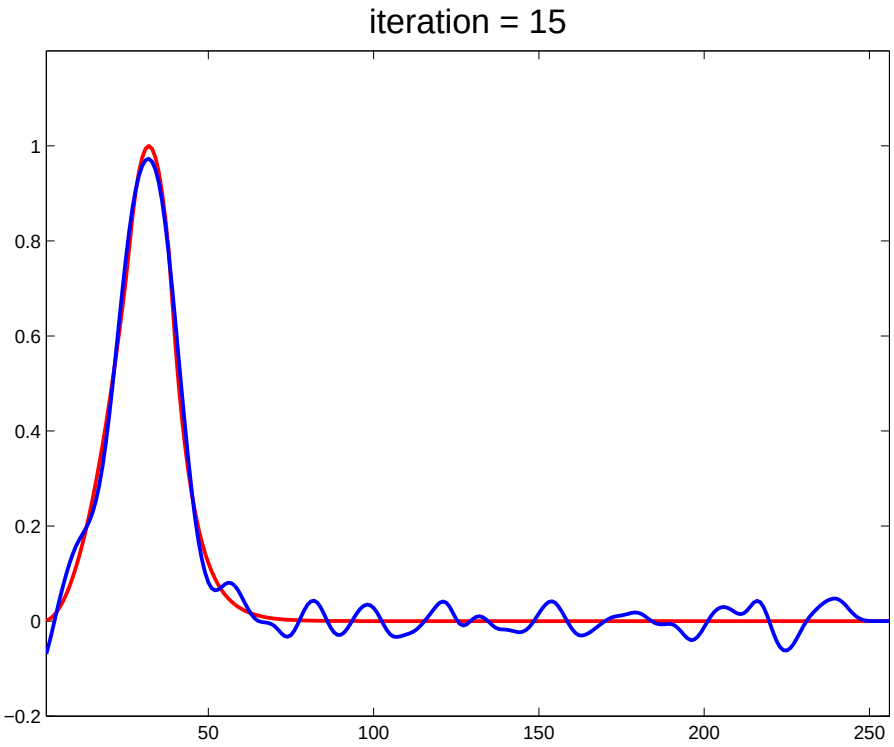
Example: Inverse Heat Equation

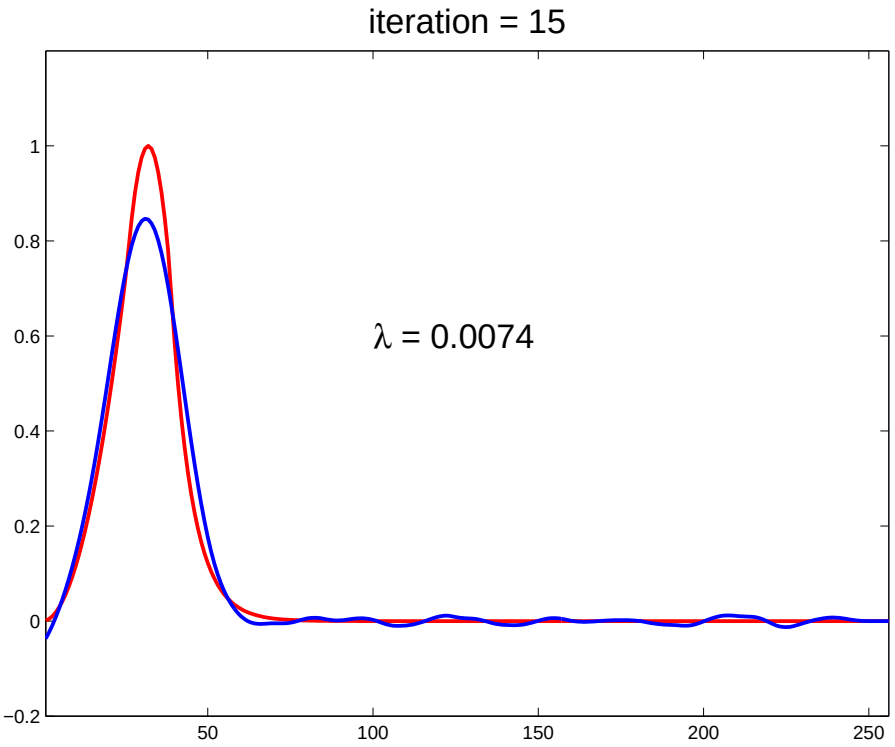
LSQR (no regularization)

$$\mathbf{f} = \mathbf{B}_k \setminus \mathbf{W}_k \mathbf{b}$$
$$\mathbf{x}_k = \mathbf{Z}_k \mathbf{f}$$

HyBR (Tikhonov regularization)

$$\mathbf{f} = \begin{bmatrix} \mathbf{B}_k \\ \lambda_k \mathbf{I} \end{bmatrix} \setminus \begin{bmatrix} \mathbf{W}_k \mathbf{b} \\ \mathbf{0} \end{bmatrix}$$
$$\mathbf{x}_k = \mathbf{Z}_k \mathbf{f}$$





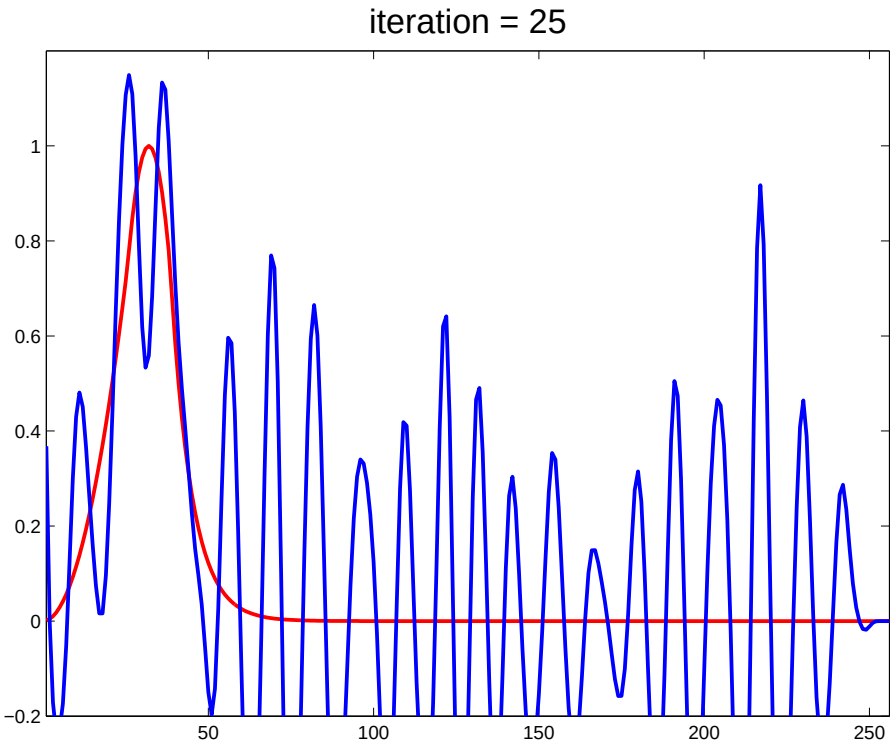
Example: Inverse Heat Equation

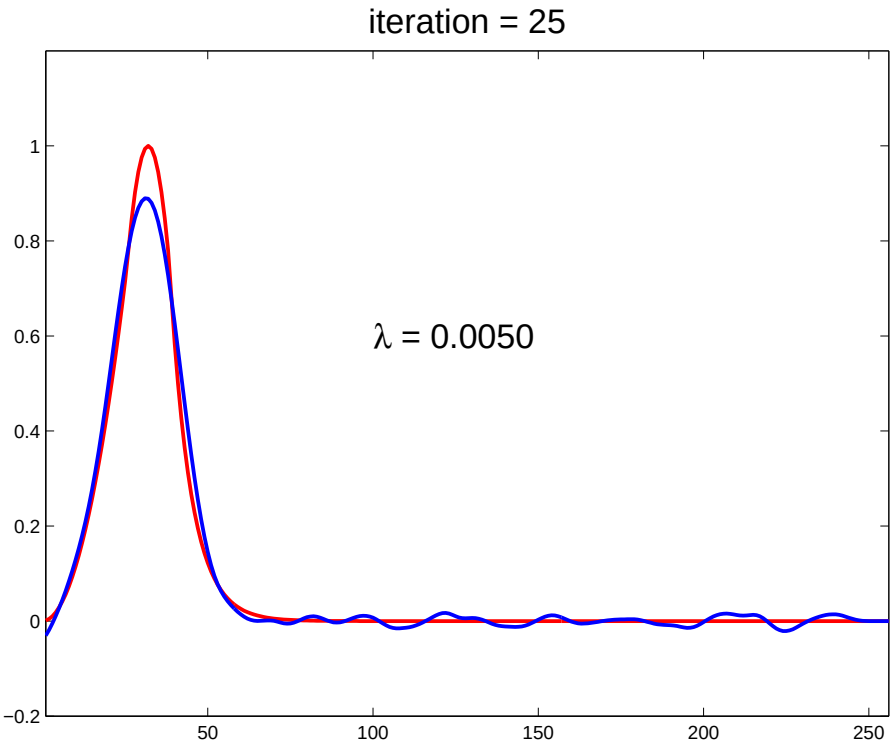
LSQR (no regularization)

$$\mathbf{f} = \mathbf{B}_k \setminus \mathbf{W}_k \mathbf{b}$$
$$\mathbf{x}_k = \mathbf{Z}_k \mathbf{f}$$

HyBR (Tikhonov regularization)

$$\mathbf{f} = \begin{bmatrix} \mathbf{B}_k \\ \lambda_k \mathbf{I} \end{bmatrix} \setminus \begin{bmatrix} \mathbf{W}_k \mathbf{b} \\ \mathbf{0} \end{bmatrix}$$
$$\mathbf{x}_k = \mathbf{Z}_k \mathbf{f}$$





Example: Inverse Heat Equation

LSQR (no regularization)

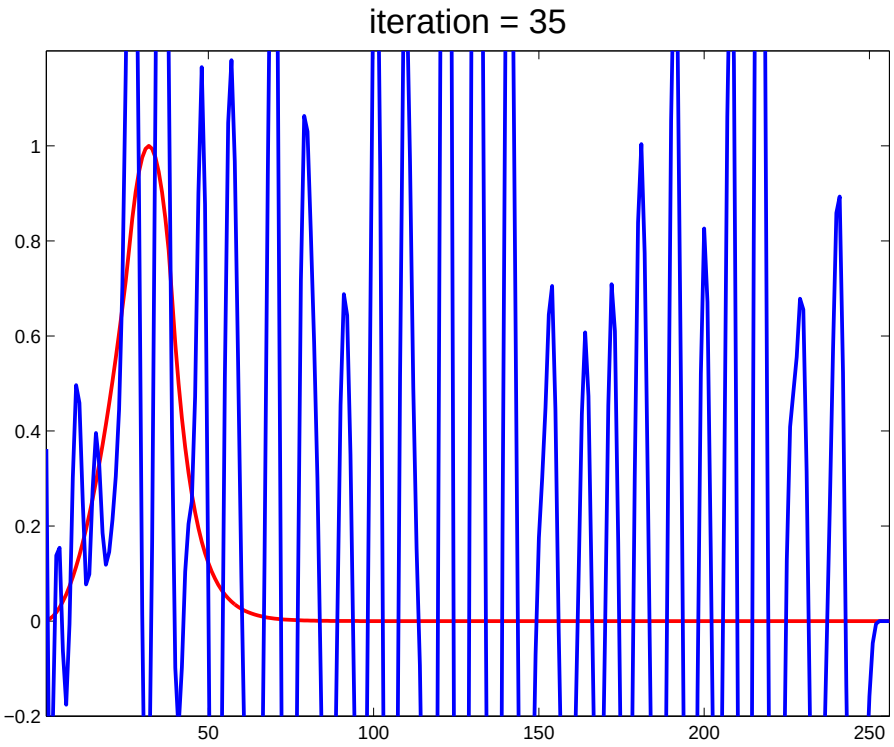
$$\mathbf{f} = \mathbf{B}_k \setminus \mathbf{W}_k \mathbf{b}$$

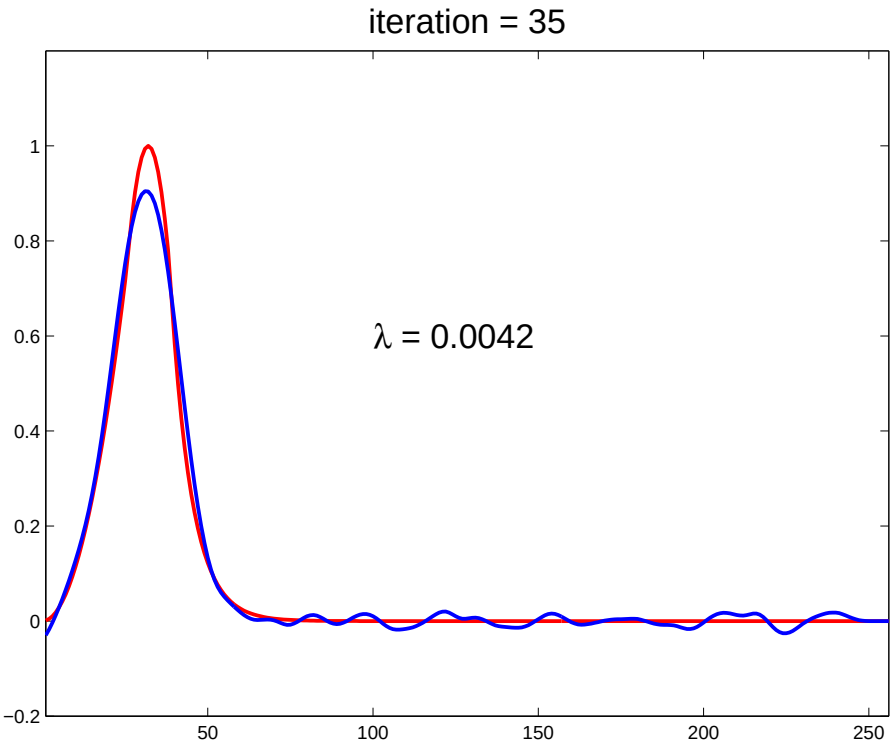
$$\mathbf{x}_k = \mathbf{Z}_k \mathbf{f}$$

HyBR (Tikhonov regularization)

$$\mathbf{f} = \begin{bmatrix} \mathbf{B}_k \\ \lambda_k \mathbf{I} \end{bmatrix} \setminus \begin{bmatrix} \mathbf{W}_k \mathbf{b} \\ \mathbf{0} \end{bmatrix}$$

$$\mathbf{x}_k = \mathbf{Z}_k \mathbf{f}$$





Regularize the Projected Least Squares Problem

To stabilize convergence, regularize the projected problem:

$$\min_{\mathbf{f}} \left\| \begin{bmatrix} \beta \mathbf{e}_1 \\ \mathbf{0} \end{bmatrix} - \begin{bmatrix} \mathbf{B}_k \\ \lambda \mathbf{I} \end{bmatrix} \mathbf{f} \right\|_2^2$$

Problems choosing regularization parameters:

- Very little regularization is needed in early iterations.
- GCV tends to choose too large λ for bidiagonal system.
  Our remedy: Use a weighted GCV (Chung, N, O'Leary, 2007)
- Can also use WGCV information to estimate stopping iteration
  (approach similar to Björck, Grimme, and Van Dooren, BIT, 1994).

Weighted GCV

If GCV tends to over or under smooth for class of problems, use:

$$GCV(\lambda) = \frac{n||(I - AA_\lambda^\dagger)\mathbf{b}||^2}{\left[\text{trace}(I - \omega AA_\lambda^\dagger)\right]^2}$$

- $\omega = 1 \Rightarrow$ standard GCV
- $\omega > 1 \Rightarrow$ smoother solutions
- $\omega < 1 \Rightarrow$ less smooth solutions

Weighted GCV used in:

Friedman, Silverman (Technometrics, 1989)

Nychka, et al. (FUNFITS statistical toolbox, 1998)

Cummins, Filloon, Nychka (J. Am. Stat. Assoc., 2001)

Kim, Gu (Royal Stat. Soc. B, 2004)

Interpretations of Modified GCV

- Weighted "leave-one-out" prediction method.

- $\text{trace}\left(I - \omega A A_\lambda^\dagger\right) = \sum_{i=1}^{n}(1-\phi_i) + (1-\omega)\sum_{i=1}^{n}\phi_i,$

  where $\phi_i = \dfrac{\sigma_i^2}{\sigma_i^2 + \lambda^2}$ (Tikhonov SVD filter factors)

- If $\omega > 1$, modified GCV function has poles when $\sum_{i=1}^{n}\phi_i = \dfrac{n}{\omega}$

How to choose ω?

- GCV chooses too large λ_k at each iteration.
- If we know $\lambda_{k,opt}$, find ω by solving

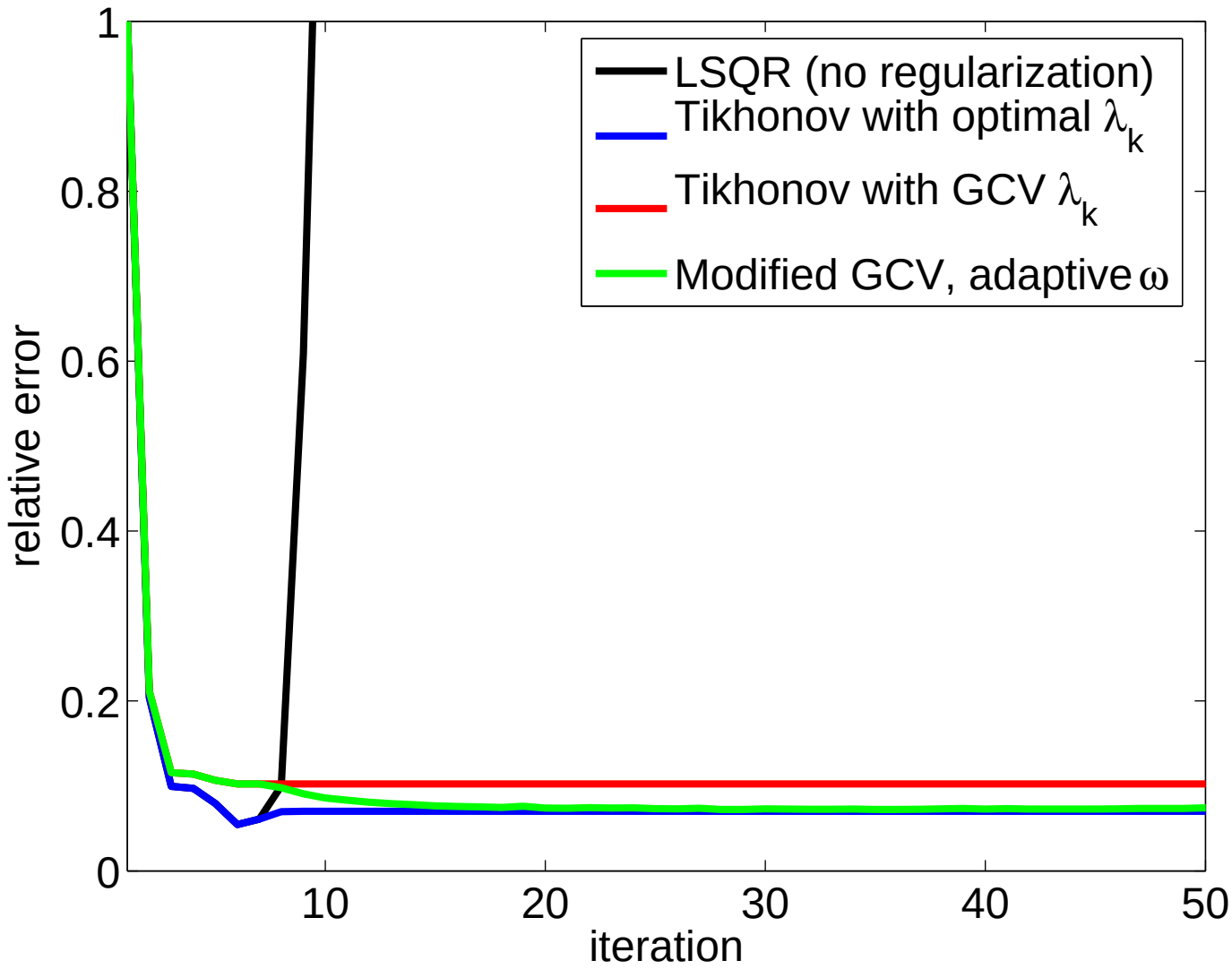
$$\frac{\partial}{\partial \lambda} \left[G(\omega, \lambda) \right] \Big|_{\lambda = \lambda_{k,opt}} = 0$$

- At early iterations, we need little or no regularization, so

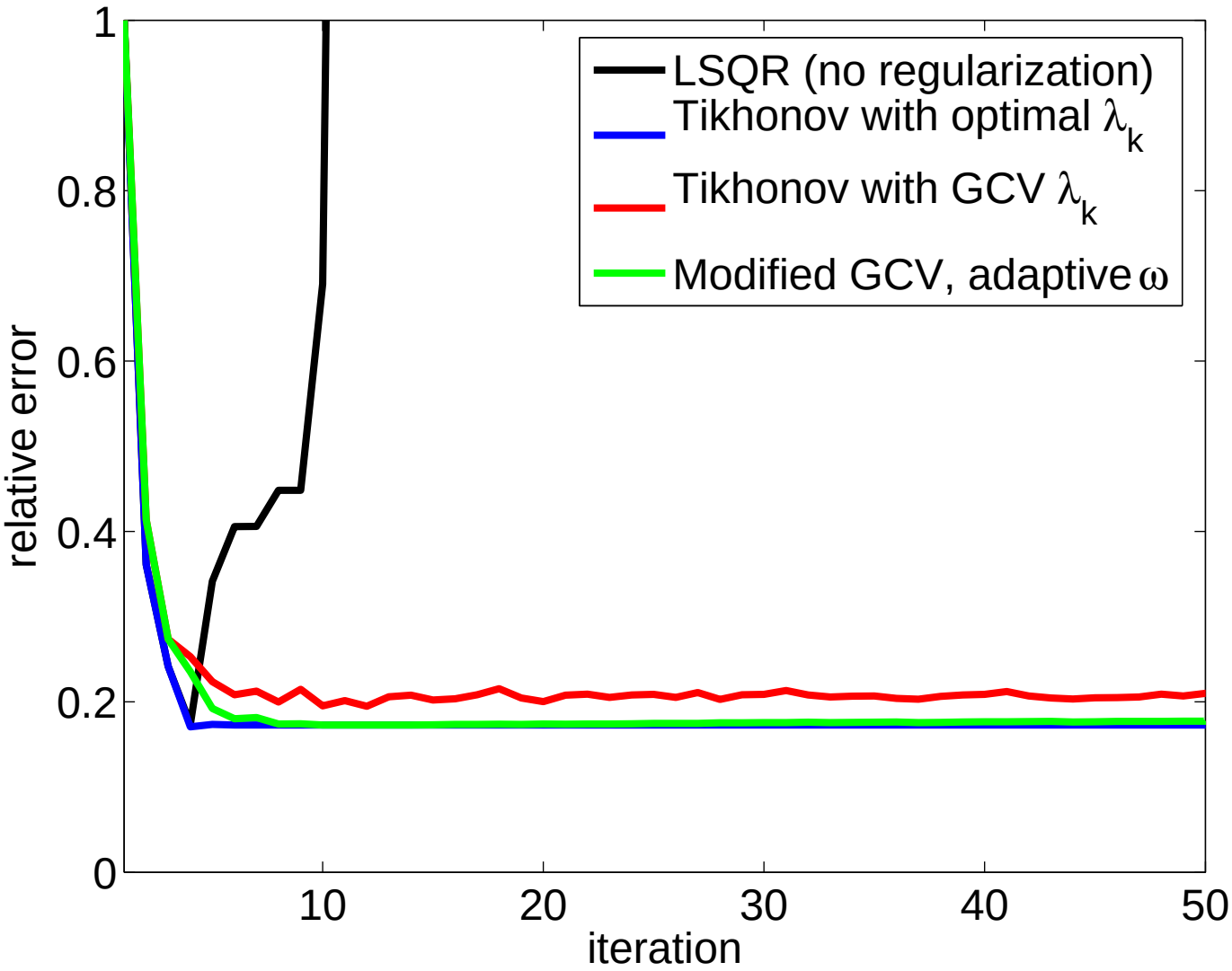
$$0 \leq \lambda_{k,opt} \leq \sigma_{min} \left(\mathbf{B}_k \right)$$

- Adaptive approach:
  - Find $\hat{\omega}_k$ corresponding to $\lambda_{k,opt} = \sigma_{min} \left(\mathbf{B}_k \right)$
  - Use $\omega_k =$mean$\{\hat{\omega}_1, \hat{\omega}_2, \ldots, \hat{\omega}_k\}$

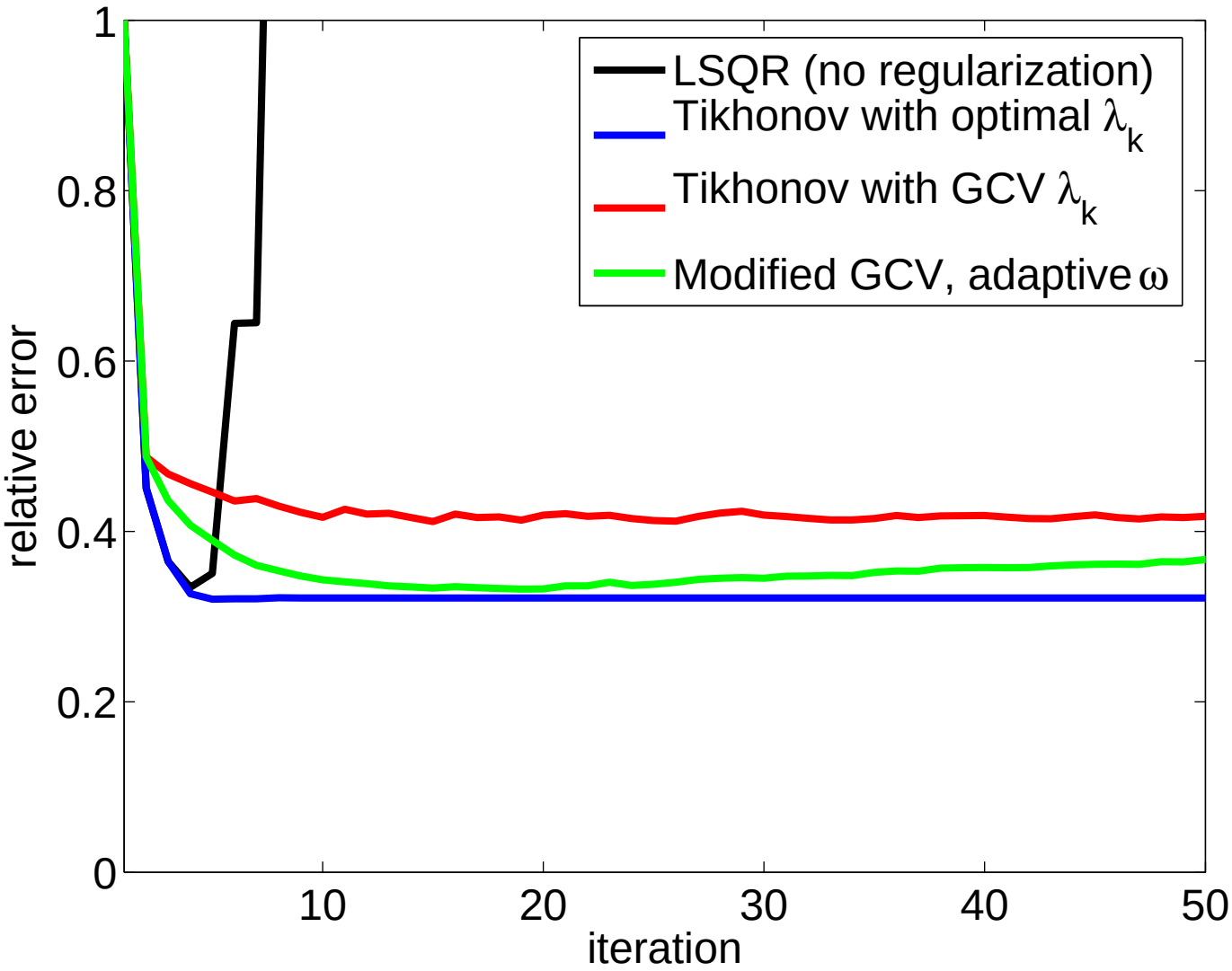
Examples: Regularization Tools, `phillips`



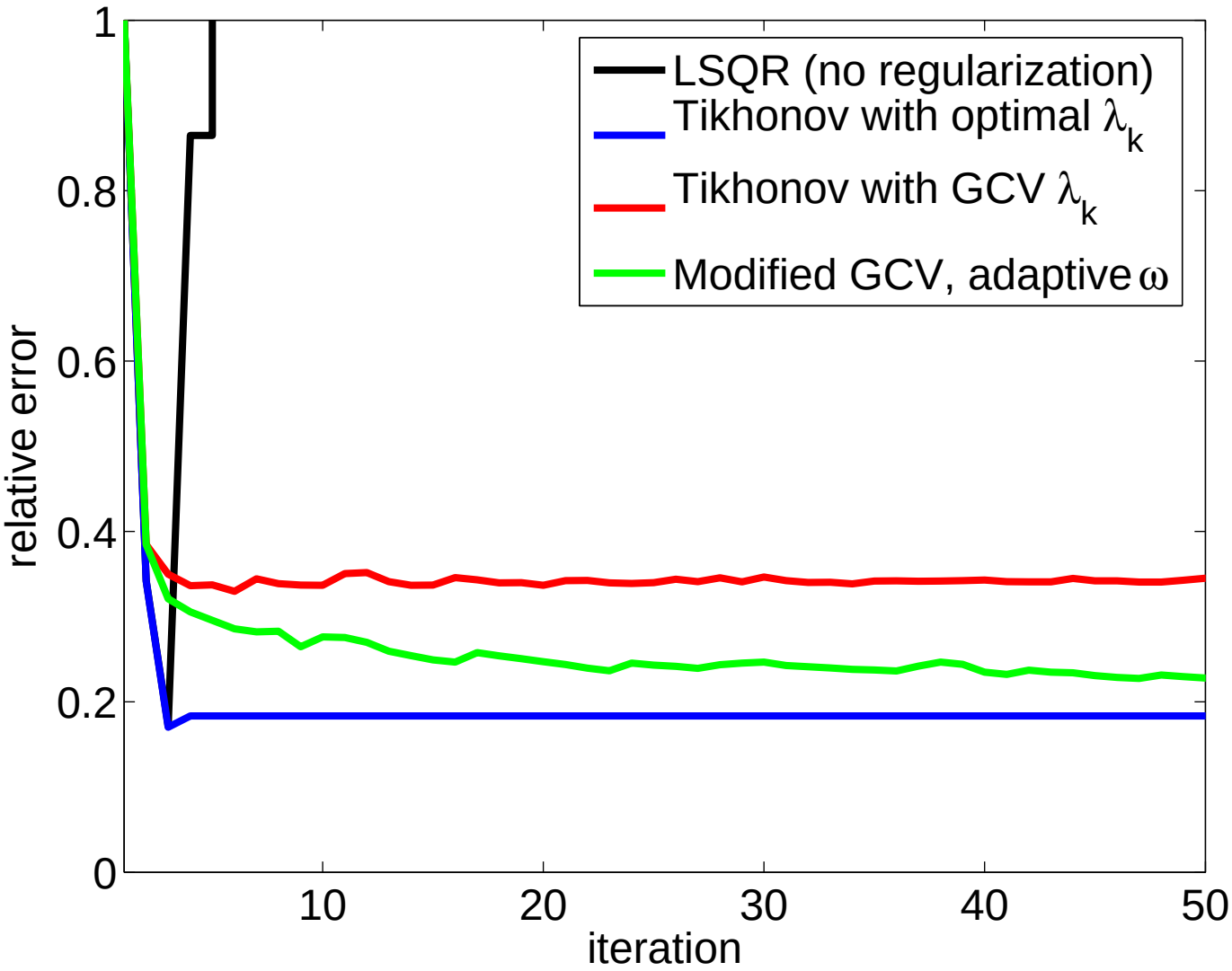
Examples: Regularization Tools, `shaw`



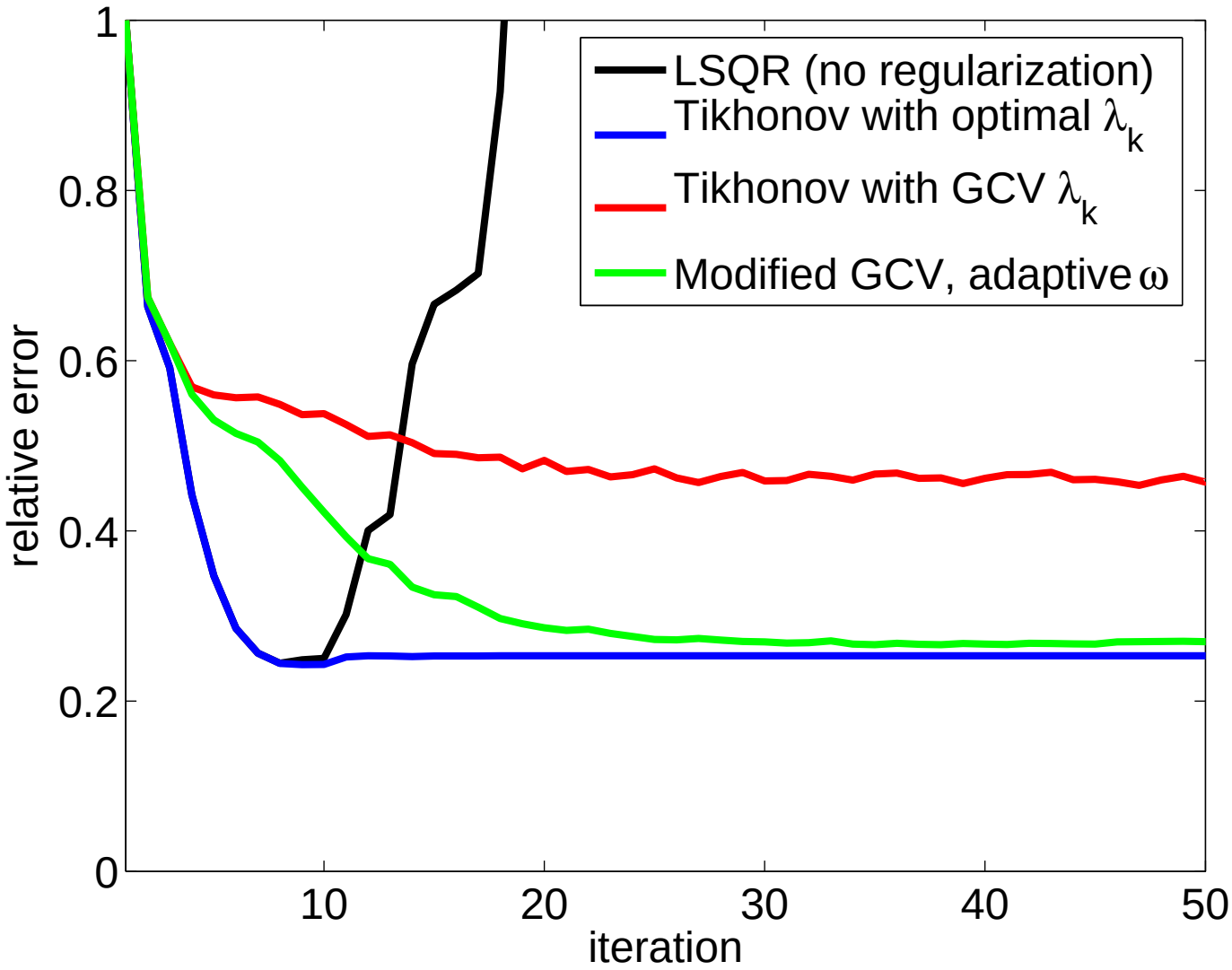
Examples: Regularization Tools, `deriv2`



Examples: Regularization Tools, `baart`



Examples: Regularization Tools, `heat`



Example: Patient Motion in Medical Imaging

PET motion correction for brain imaging:

- Head moves during data acquisition $\Rightarrow$ reconstructed brain image, $\mathbf{b}$ is distorted by motion blur.
- Attach "cap" with fixed markers to patient head.
- Motion detection camera records position of patient head.
- Construct large, sparse matrix $\mathbf{A}$ from position information.
- Solve linear inverse problem, $\mathbf{b} = \mathbf{A}\mathbf{x} + \boldsymbol{\eta}$.

Example: Patient Motion in Medical Imaging

To construct matrix $\mathbf{A}$:

- Assume position at time t_ℓ is known.
- Position information used to construct $\mathbf{A}_\ell = \mathbf{A}(\mathbf{y}_\ell)$, where

$$\mathbf{b}_\ell = \mathbf{A}_\ell \mathbf{x}$$

  is (unknown) image at time t_ℓ.
- Motion blurred image is modeled as:

$$\mathbf{b} = \sum_{\ell=1}^{m} w_\ell \mathbf{b}_\ell + \boldsymbol{\eta}$$

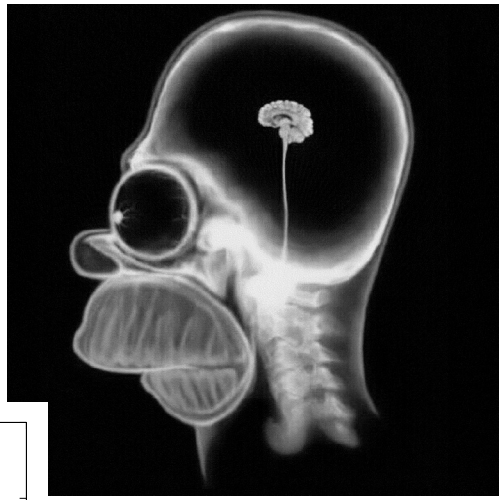
  w_ℓ is normalization weight for the ℓth image (e.g, $w_\ell = \frac{1}{m}$),
- So, matrix modeling motion blur is

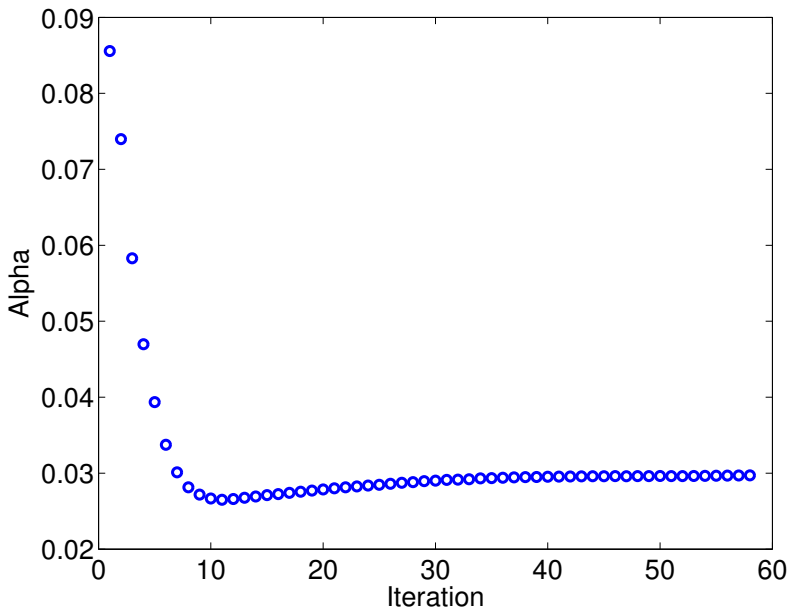
$$\mathbf{A} = \sum_{\ell=1}^{m} w_\ell \mathbf{A}_\ell.$$

2D Simulations: Little Movement

$\mathbf{b} = \mathbf{A}\mathbf{x} + \boldsymbol{\eta}$

$\mathbf{x} = \text{HyBR}(\mathbf{A}, \mathbf{b})$

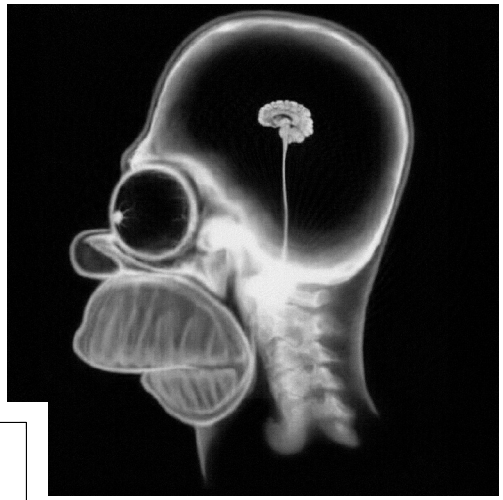


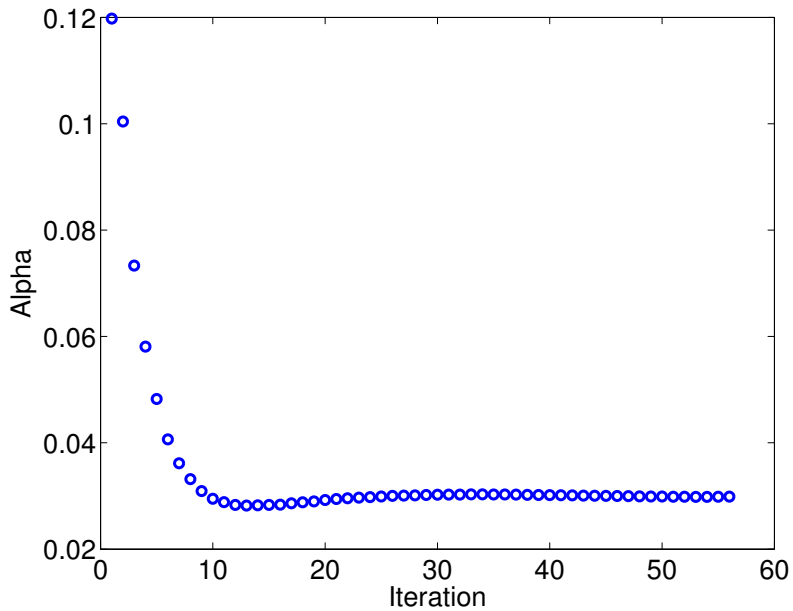


2D Simulations: More Movement

$\mathbf{b} = \mathbf{A}\mathbf{x} + \boldsymbol{\eta}$

$\mathbf{x} = \text{HyBR}(\mathbf{A}, \mathbf{b})$

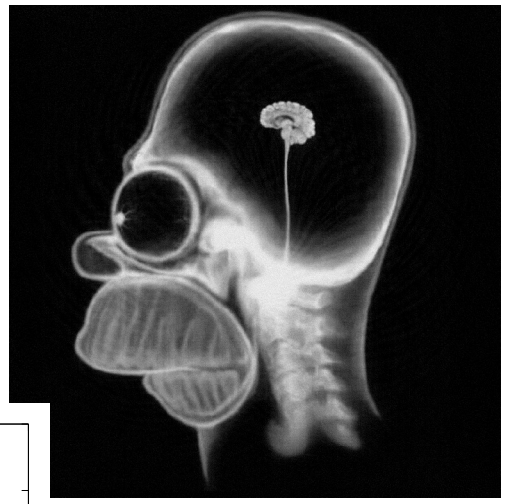


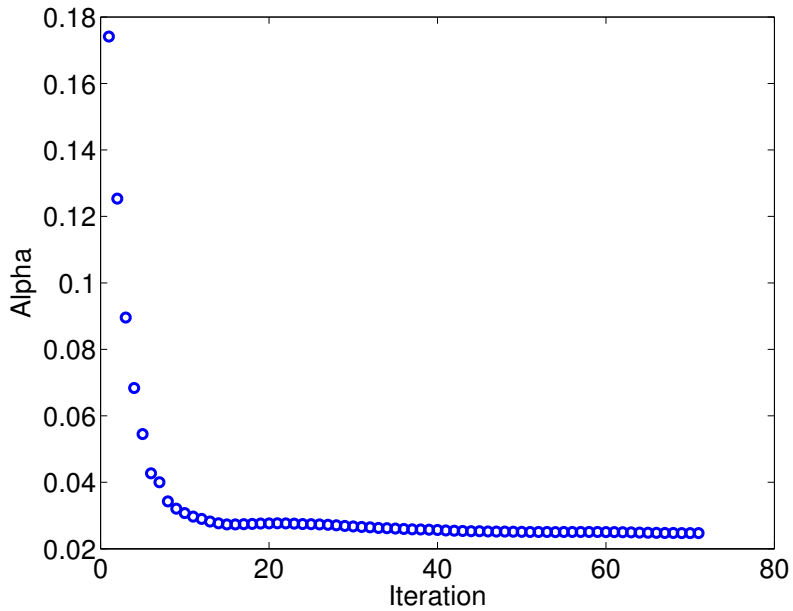


2D Simulations: Large Movement

$\mathbf{b} = \mathbf{A}\mathbf{x} + \boldsymbol{\eta}$

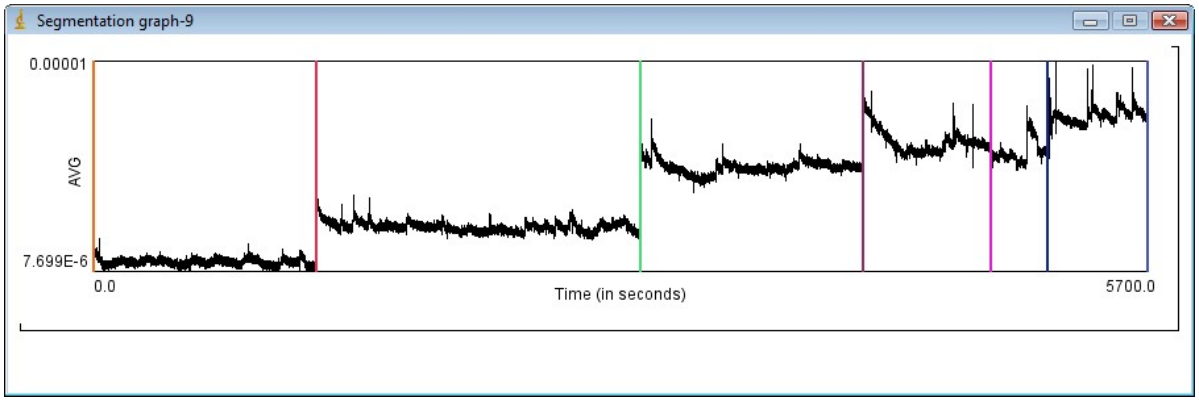
$\mathbf{x} = \text{HyBR}(\mathbf{A}, \mathbf{b})$





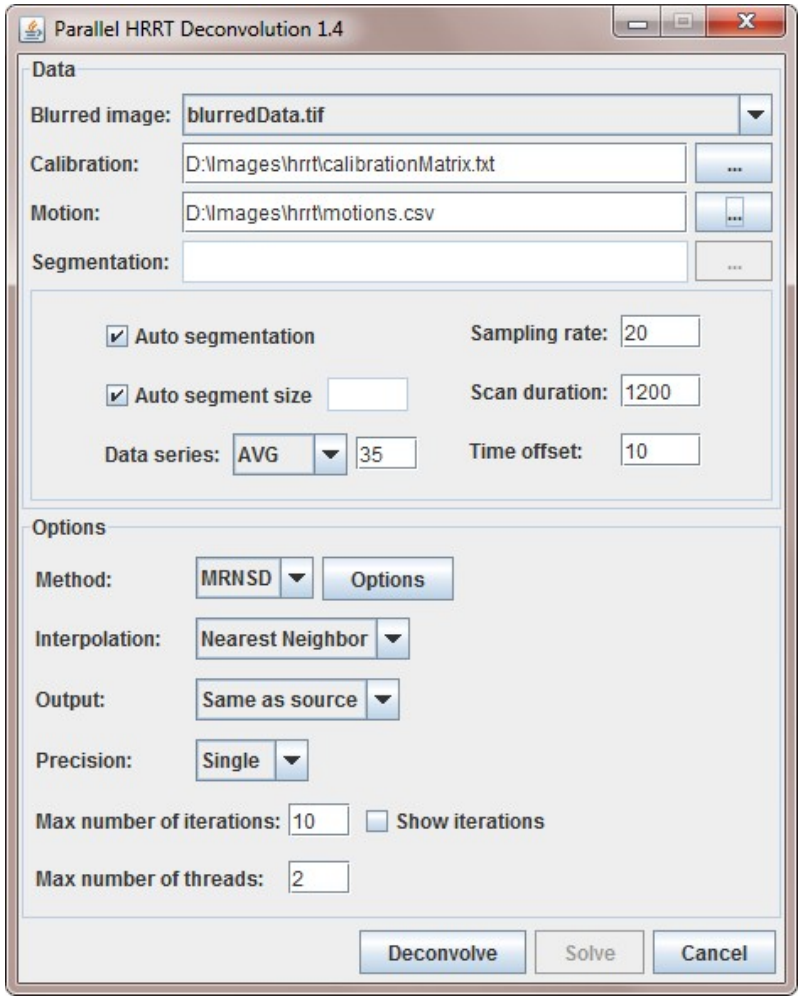
Practical Issue: Cost of constructing $\mathbf{A} = \sum_{\ell=1}^{m} w_\ell \mathbf{A}_\ell$

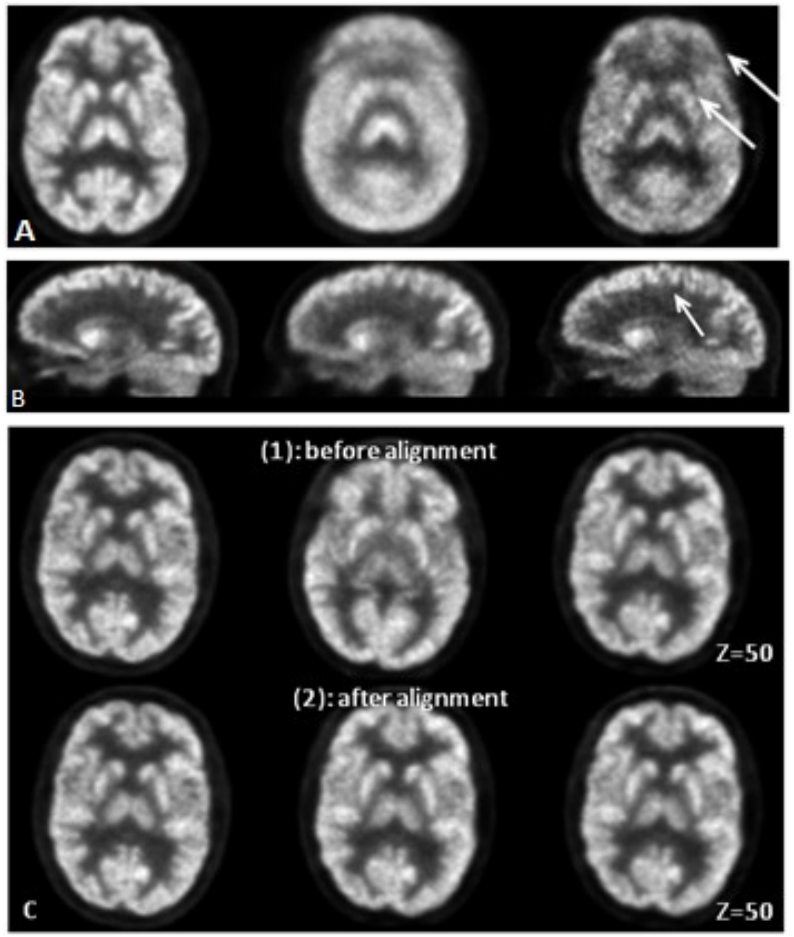
- Images are 3D, so each $\mathbf{A}_\ell = \mathbf{A}(\mathbf{y}_\ell)$ is at least $10^6 \times 10^6$.
- Motion detection collects 20 position quaternions per second.
- Scan time ranges from 20 - 30 minutes.
- Thus, there are $m = 24{,}000$ - $36{,}000$ position quaternions.
- To reduce setup cost, find bins with little movement:



- Use average position quaternion in each bin.

Multithreaded Java Implementation with ImageJ Plugin





http://sites.google.com/site/piotrwendykier/software

The Nonlinear Problem

Until this point, we consider:

$$\mathbf{b} = \mathbf{A}(\mathbf{y})\,\mathbf{x} + \mathbf{e}$$

with $\mathbf{y}$ assumed known.

Now assume $\mathbf{y}$ is not known.

The Nonlinear Problem

- We want to find $\mathbf{x}$ and $\mathbf{y}$ so that

$$\mathbf{b} = \mathbf{A}(\mathbf{y})\mathbf{x} + \mathbf{e}$$

- With Tikhonov regularization, solve

$$\min_{\mathbf{x},\mathbf{y}} \left\| \begin{bmatrix} \mathbf{A}(\mathbf{y}) \\ \lambda \mathbf{I} \end{bmatrix} \mathbf{x} - \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix} \right\|_2^2$$

- As with linear problem, choosing a good regularization parameter λ is important.
- Problem is linear in $\mathbf{x}$, nonlinear in $\mathbf{y}$.
- Often $\mathbf{y} \in \mathcal{R}^p$, $\mathbf{x} \in \mathcal{R}^n$, with $p \ll n$.

Solving Nonlinear Least Squares Problem

Options to solve nonlinear least squares problem:

- Fully coupled approach:

$$\mathbf{x}_k, \mathbf{y}_k = \arg\min_{\mathbf{x},\mathbf{y}} \left\{ \|\mathbf{A}(\mathbf{y})\mathbf{x} - \mathbf{b}\|_2^2 + \lambda \|\mathbf{x}\|_2^2 \right\}$$

- Decoupled approach: Block coordinate descent

$$\begin{aligned} \mathbf{x}_k &= \arg\min_{\mathbf{x}} \left\| \|\mathbf{A}(\mathbf{y}_k)\mathbf{x} - \mathbf{b}\|_2^2 + \lambda_k^2 \|\mathbf{x}\|_2^2 \right\} \\ \mathbf{y}_{k+1} &= \arg\min_{\mathbf{y}} \left\{ \|\mathbf{A}(\mathbf{y})\mathbf{x}_k - \mathbf{b}\|_2^2 + \lambda_k^2 \|\mathbf{x}_k\|_2^2 \right\} \end{aligned}$$

- Partially coupled approach: Variable Projection

$$\mathbf{y}_k = \arg\min_{\mathbf{y}} \left\{ \|\mathbf{A}(\mathbf{y})\mathbf{A}^\dagger(\mathbf{y})\mathbf{b} - \mathbf{b}\|_2^2 + \lambda_k^2 \|\mathbf{A}^\dagger(\mathbf{y})\mathbf{b}\|_2^2 \right\}$$

Variable Projection

Variable Projection Method:

- Implicitly eliminate linear term.
- Optimize over nonlinear term.

Some general references:

Golub and Pereyra, SINUM 1973 (also IP 2003)
Kaufman, BIT 1975
Osborne, SINUM 1975 (also ETNA 2007)
Ruhe and Wedin, SIREV, 1980

How to apply to inverse problems?

Gauss-Newton Algorithm

choose initial $\mathbf{y}_0$

for $k = 0, 1, 2, \ldots$

$$\mathbf{x}_k = \arg\min_{\mathbf{x}} \left\| \begin{bmatrix} \mathbf{A}(\mathbf{y}_k) \\ \lambda_k \mathbf{I} \end{bmatrix} \mathbf{x} - \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix} \right\|_2$$

$$\mathbf{r}_k = \mathbf{b} - \mathbf{A}(\mathbf{y}_k)\,\mathbf{x}_k$$

$$\mathbf{d}_k = \arg\min_{\mathbf{d}} \|\mathbf{J}_\psi \mathbf{d} - \mathbf{r}_k\|_2$$

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \mathbf{d}_k$$

end

Gauss-Newton Algorithm with HyBR

And we use HyBR to solve the linear subproblem:

choose initial $\mathbf{y}_0$

for $k = 0, 1, 2, \ldots$

$$\mathbf{x}_k = \text{HyBR}(\mathbf{A}(\mathbf{y}_k), \mathbf{b})$$

$$\mathbf{r}_k = \mathbf{b} - \mathbf{A}(\mathbf{y}_k)\,\mathbf{x}_k$$

$$\mathbf{d}_k = \arg\min_{\mathbf{d}} \|\mathbf{J}_\psi \mathbf{d} - \mathbf{r}_k\|_2$$

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \mathbf{d}_k$$

end

Example: Digital Super-Resolution

Basic idea:

- Capture several images of a moving object: $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m$
- Combine different $\mathbf{b}_i$ to get one (better) image: $\mathbf{x}$

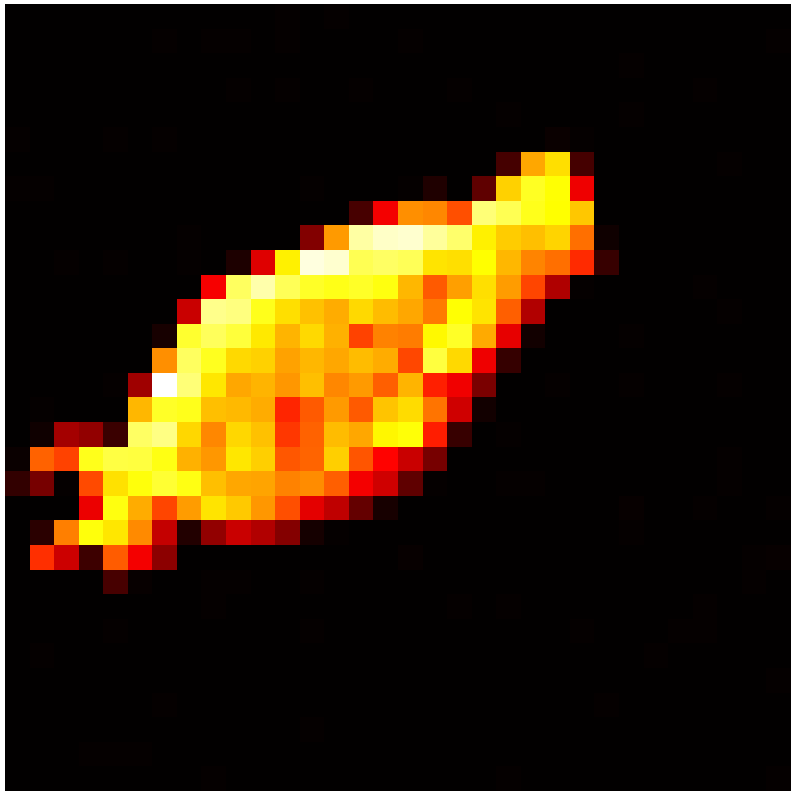
The problem is modeled as: $\mathbf{b} = \mathbf{A}(\mathbf{y})\mathbf{x} + \mathbf{e}$

- $\mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_m \end{bmatrix}$
- $\mathbf{A}(\mathbf{y})$ describes geometric distortion on $\mathbf{x}$ to get $\mathbf{b}_i$
- $\mathbf{y}$ defines geometric distortion (e.g., affine transformation)

Example: Digital Super-Resolution

Measured Images

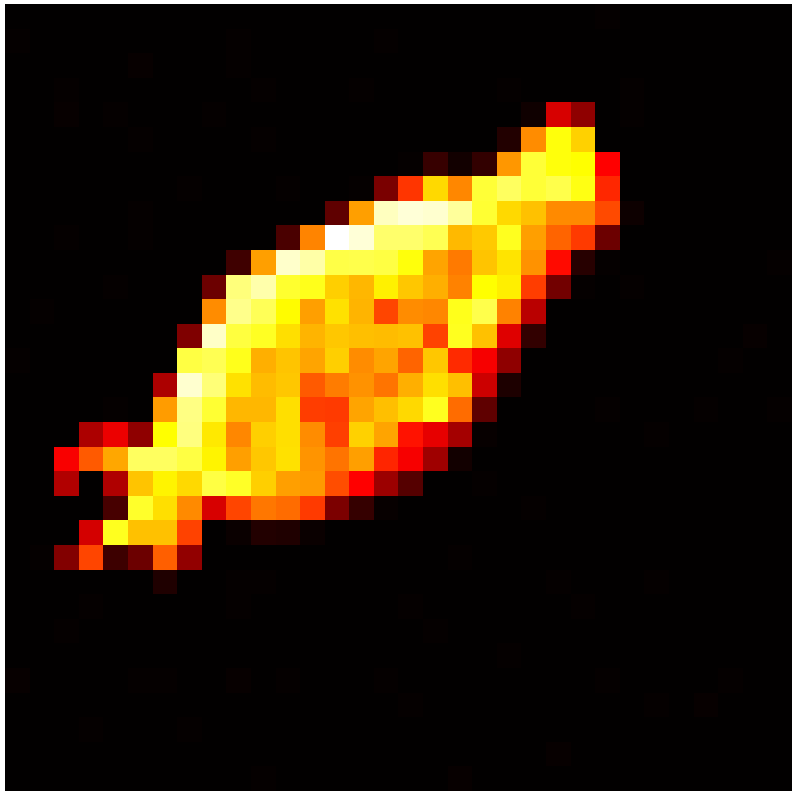
1−th low resolution image



Example: Digital Super-Resolution

Measured Images

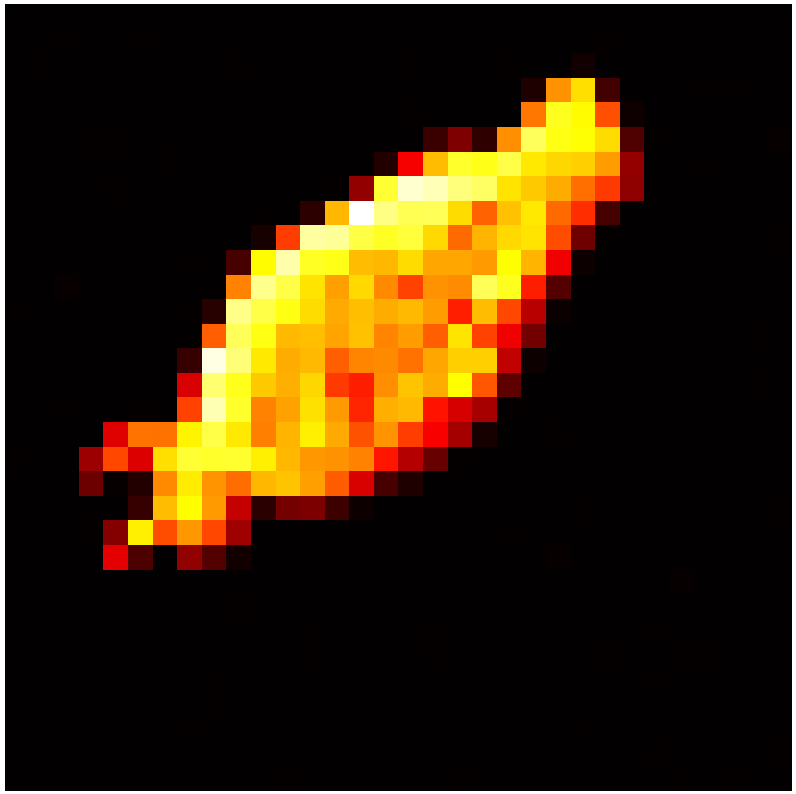
8−th low resolution image



Example: Digital Super-Resolution

Measured Images

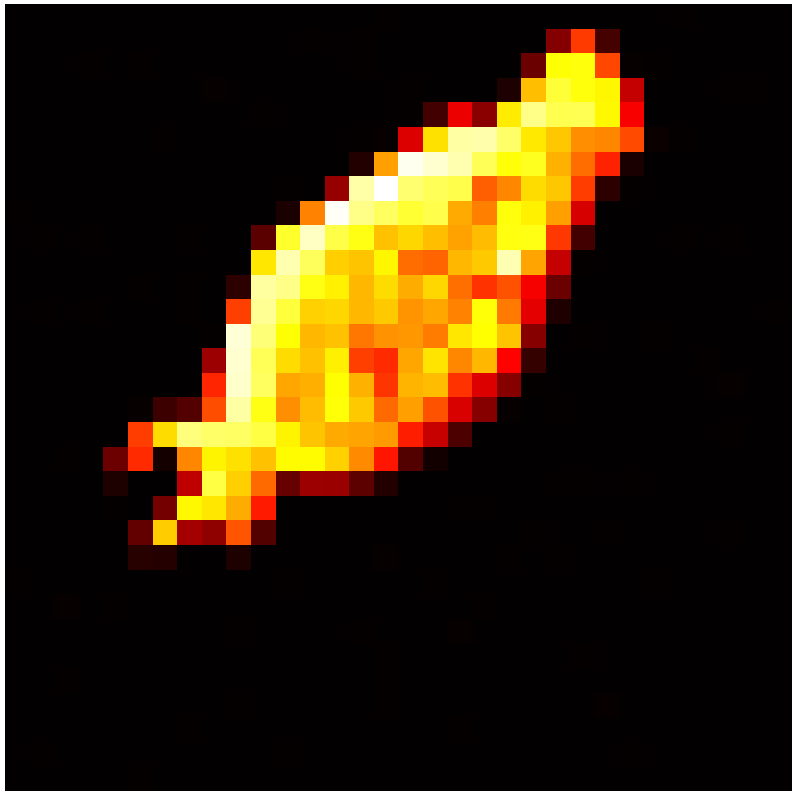
15−th low resolution image



Example: Digital Super-Resolution

Measured Images

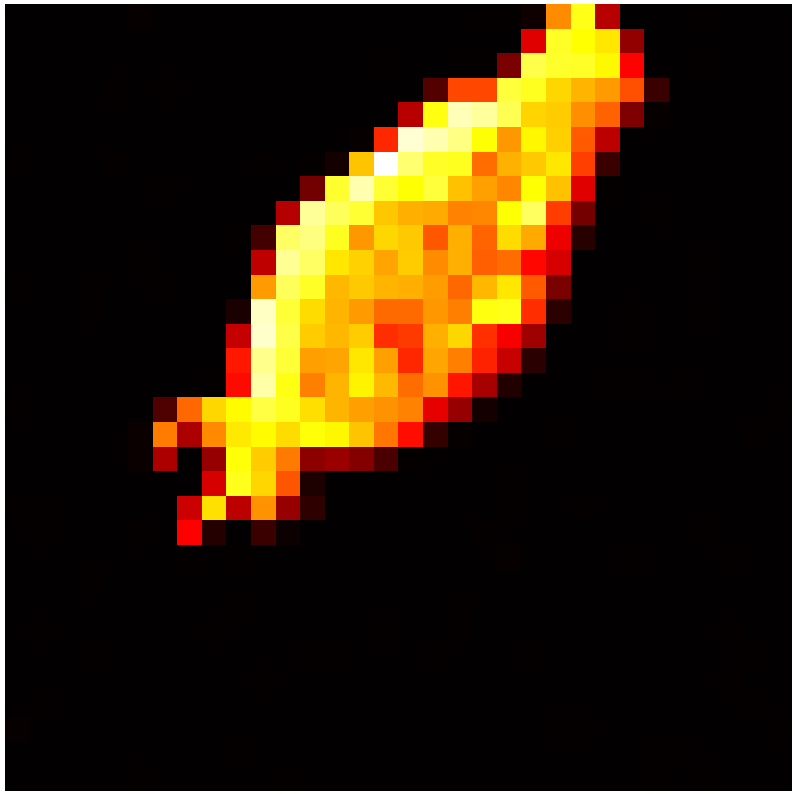
22−th low resolution image



Example: Digital Super-Resolution

Measured Images

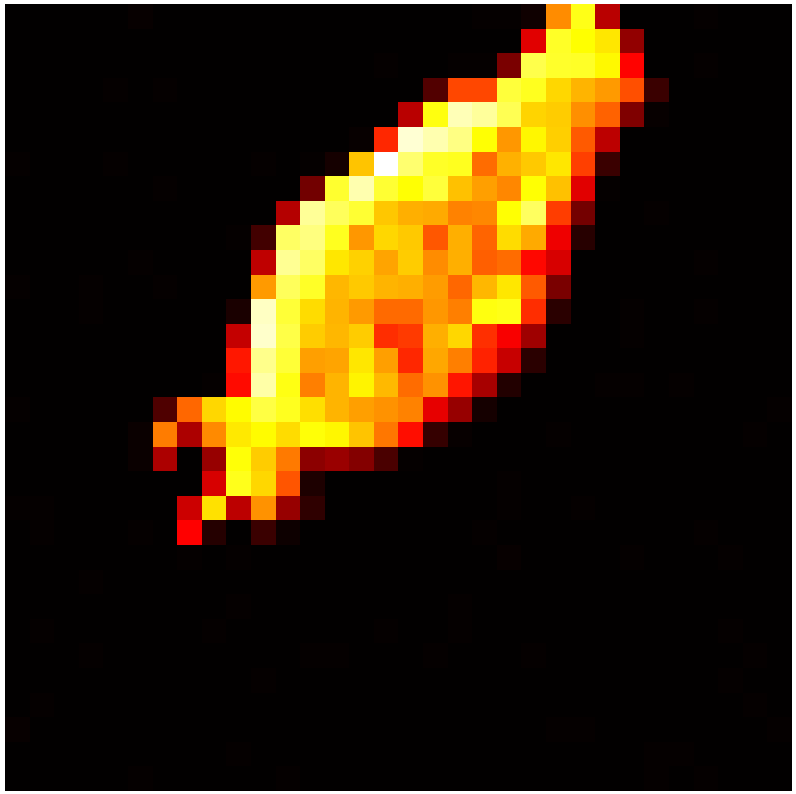
29−th low resolution image



Example: Digital Super-Resolution

Measured Images

29−th low resolution image



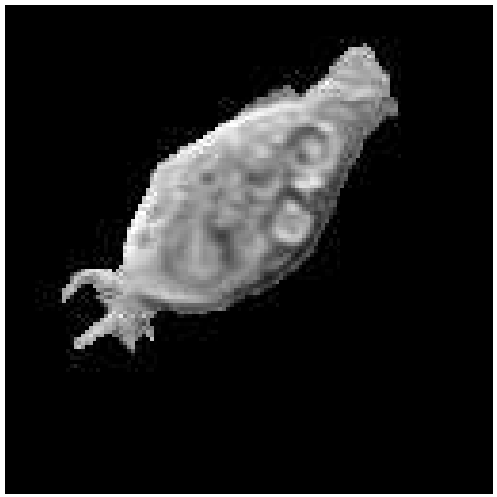
Example: Digital Super-Resolution

Gauss-Newton Iteration History

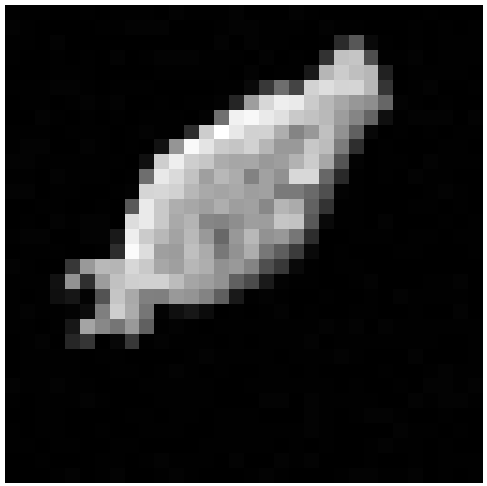
	Fixed $\lambda = 0.01$		Fixed $\lambda = 0.45$		HyBR, variable λ	
G-N Iteration	$\Delta\mathbf{y}$	λ	$\Delta\mathbf{y}$	λ	$\Delta\mathbf{y}$	λ
0	0.1182	0.01	0.1182	0.45	0.1182	0.1324
1	0.1024	0.01	0.0841	0.45	0.0876	0.1127
2	0.0921	0.01	0.1337	0.45	0.0745	0.1062
3	0.0854	0.01	0.2028	0.45	0.0680	0.1052
4	0.0811	0.01	0.2742	0.45	0.0641	0.1040
5	0.0783	0.01	0.3451	0.45	0.0614	0.1036
6	0.0764	0.01	0.4146	0.45	0.0593	0.1034
7	0.0750	0.01	0.4826	0.45	0.0576	0.1034
8	0.0740	0.01	0.5490	0.45	0.0575	0.1034
9	0.0733	0.01	0.6140	0.45	0.0575	0.1034
10	0.0726	0.01	0.6774	0.45	0.0574	0.1034

Example: Digital Super-Resolution

True Image



One Measured Image



Reconstructed Image

Discussion Points

Advantages of variable projection:

- Eliminate degrees of freedom associated with the image
  $\Rightarrow$ low-dimensional reduced optimization problem.
- Effective when the linear least squares problem can be solved efficiently and to high accuracy.

Disadvantages of variable projection:

- May not be able to solve linear least squares problem efficiently.
- Adding constraints on the image intensities is more difficult.

Discussion Points

Current work:

- Applying ideas to MRI motion correction.
- Fully coupled approach seems to behave better for this application.
- Challenge: Hessian is much more complicated.
- Remedy: Use PCG to solve Hessian system.
- Challenge: Find a good preconditioner for Hessian.
- Remedy: Some ideas, but not ready for prime time.

MRI Motion Correction

MRI Motion Correction Problem: Given a set of MRI data sampled in Fourier space $\boldsymbol{b} = (\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_m)$, retrieve the motion parameters $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots \mathbf{y}_m)$ and a complex image $\boldsymbol{x}$ by solving a non-linear optimization problem

$$\min_{\boldsymbol{x},\boldsymbol{\omega}} \frac{1}{2} \sum_{k=1}^{m} \| \boldsymbol{D}_k \boldsymbol{F} \boldsymbol{S} \boldsymbol{T}(\mathbf{y}_k)\mathbf{x} - \mathbf{b}_k \|_2^2 + \frac{\alpha}{2} \boldsymbol{R}(\boldsymbol{x}) \tag{1}$$

where

- $\boldsymbol{D}_k$ is block diagonal matrix representing the data sampling
- $\boldsymbol{F}$ is block-diagonal matrix of 2D FFTs
- $\boldsymbol{S}$ represents the coil sensitivities of the MRI machine
- $\boldsymbol{T}(\mathbf{y}_k)$ is an interpolation matrix
- $\boldsymbol{R}$ is a regularizer

MRI Motion Correction

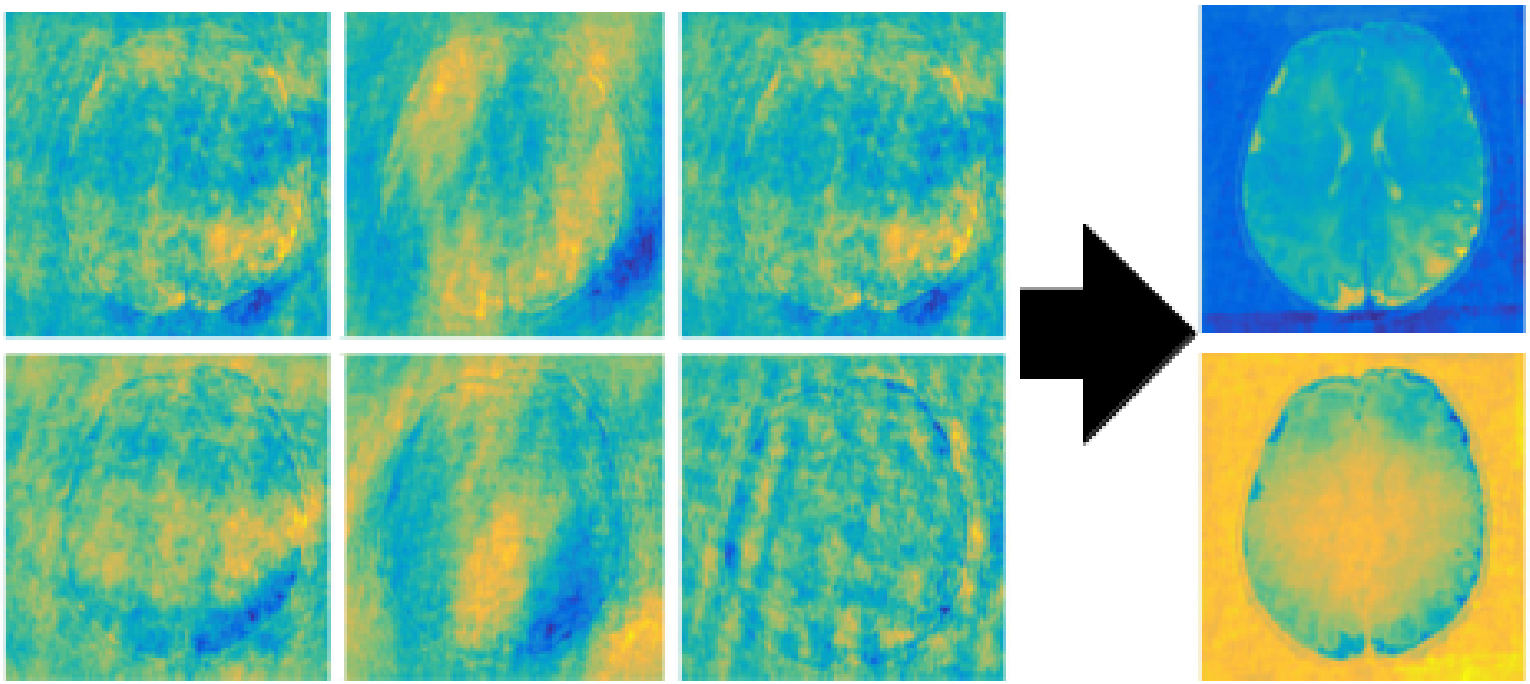


Figure: The top row displays the real part of the MRI data with the real part of the reconstructed image on the far right, while the bottom row show the imaginary part of both the data and reconstructed image. This reconstruction used 16 samples and a random sampling pattern of the Fourier data.

Take Home Points

- Motion can be good or bad.
- In bad case, with some additional information and proper mathematical models, we can effectively correct motion distortions.
- In good case, can use additional information to improve resolution.

Some software:

MATLAB Software:

`http://www.mathcs.emory.edu/~nagy/RestoreTools`

`http://www.math.vt.edu/people/jmchung/hybr.html`

Java Software, with ImageJ plugins:

`http://sites.google.com/site/piotrwendykier/software`