School of Engineering

LSMR and AMRES

Minimum-residual methods for sparse least-squares using Golub-Kahan bidiagonalization

David Fong
Institute for Computational and Mathematical Engineering
Stanford Graduate Fellow
Advisor: Prof. Michael Saunders

October 21, 2011

A brief history of LSQR

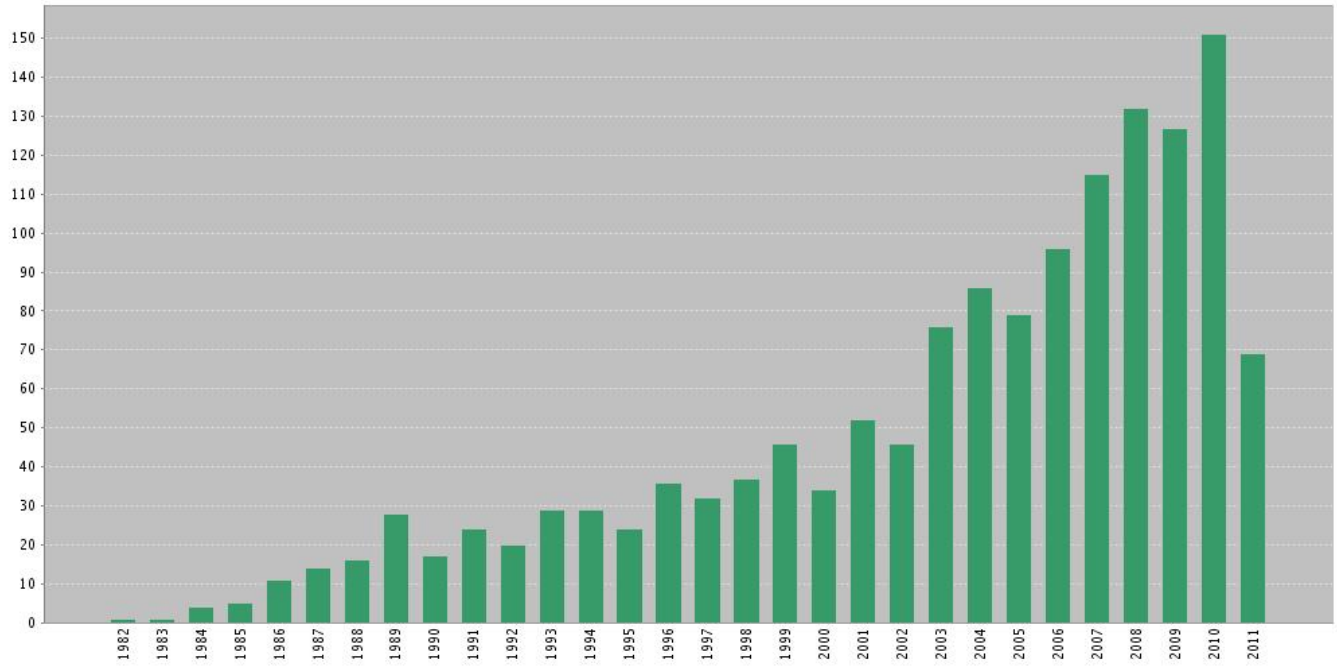
LSQR: An algorithm for sparse linear equations and sparse least squares

CC Paige, MA Saunders - ACM Transactions on Mathematical …, 1982 - portal.acm.org

An iterative method is given for solving Ax=b and min||Ax - b||, where the matrix A is large and sparse. The method is based on the bidiagonalization procedure of Golub and Kahan. It is analytically equivalent to the standard method of conjugate gradients, but possesses **...**

Cited by 1391 - Related articles - All 13 versions - Import into BibTeX

Number of citations each year



Data source: Thomson Reuters

Background

CG and MINRES

CG and MINRES: quick introduction

Iterative algorithms for

$$Ax = b, \qquad A = A^T$$

Lanczos process Tridiag(A, b)

1: $\beta_1 v_1 = b$ (i.e. $\beta_1 = \|b\|_2$, $v_1 = b/\beta_1$)
2: **for** $k = 1, 2, \ldots$ **do**
3: $\quad w = Av_k$
4: $\quad \alpha_k = v_k^T w_k$
5: $\quad \beta_{k+1} v_{k+1} = w - \alpha_k v_k - \beta_k v_{k-1}$
6: **end for**

CG and MINRES subproblems

Lanczos process (Summary)

$$\beta_1 v_1 = b \qquad AV_k = V_{k+1} H_k$$

$$H_k = \begin{pmatrix} T_k \\ \beta_{k+1} e_k^T \end{pmatrix}$$

$$V_k = \begin{pmatrix} v_1 & \cdots & v_k \end{pmatrix}$$

$$T_k = \begin{pmatrix} \alpha_1 & \beta_2 & & \\ \beta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_k \\ & & \beta_k & \alpha_k \end{pmatrix}$$

$$\begin{aligned} r_k &= b - Ax_k \\ &= \beta_1 v_1 - AV_k y_k \\ &= V_{k+1}(\beta_1 e_1 - H_k y_k), \end{aligned}$$

Aim: $\beta_1 e_1 \approx H_k y_k$

Two subproblems

CG	$T_k y_k = \beta_1 e_1$	$x_k = V_k y_k$
MINRES	$\min \|H_k y_k - \beta_1 e_1\|$	$x_k = V_k y_k$

Common practice

$$Ax = b, \;\; A = A^T$$

Positive definite A $\Rightarrow$ Use CG

Indefinite A $\Rightarrow$ Use MINRES

Experiment: CG vs MINRES on $A \succ 0$

Data: Real, symmetric positive-definite matrices from Tim Davis' sparse matrix collection that includes b

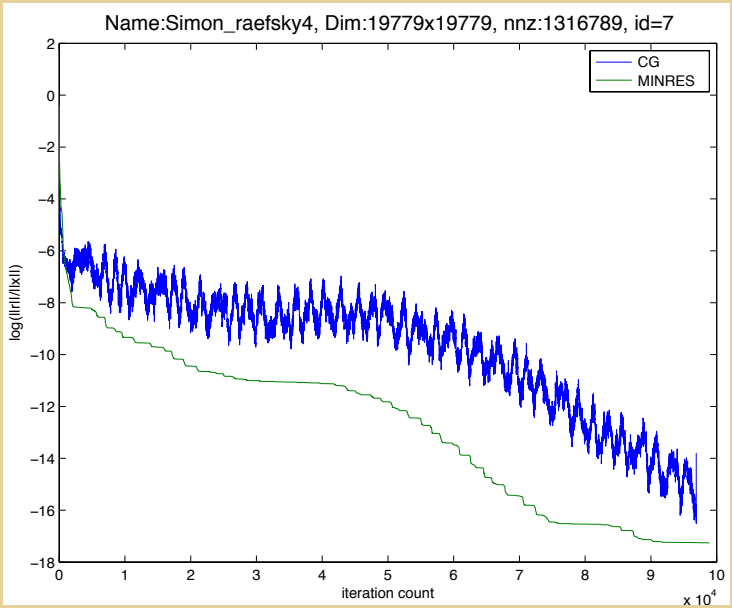
Backward error for square systems

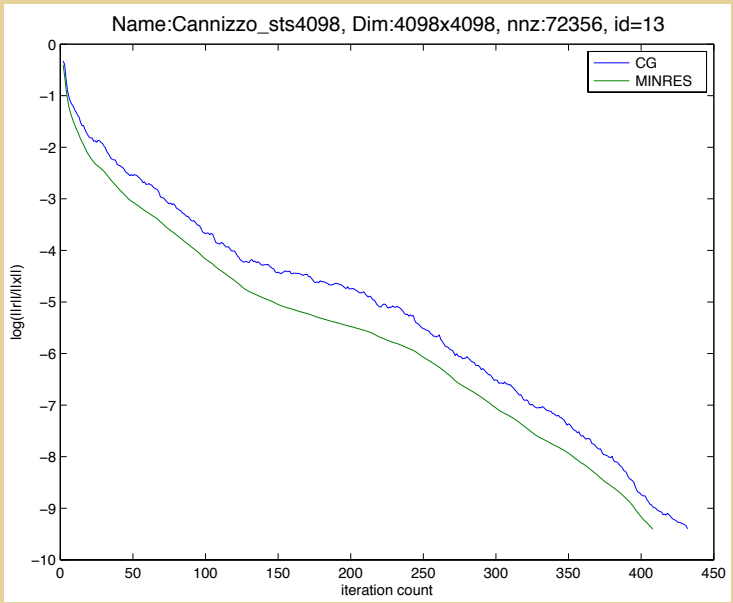
$$(A + E^{(k)})x_k = b$$

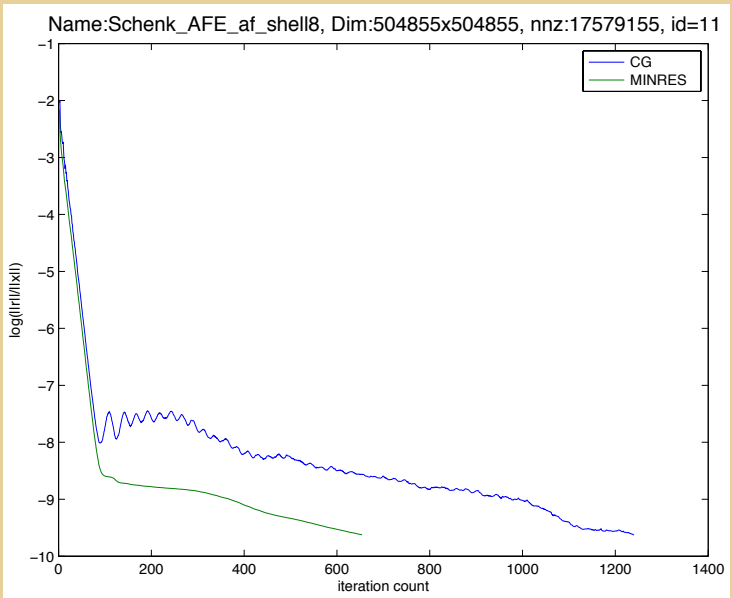
$$E^{(k)} = \frac{r_k x_k^T}{\|x_k\|^2} \qquad \|E^{(k)}\| = \frac{\|r_k\|}{\|x_k\|}$$

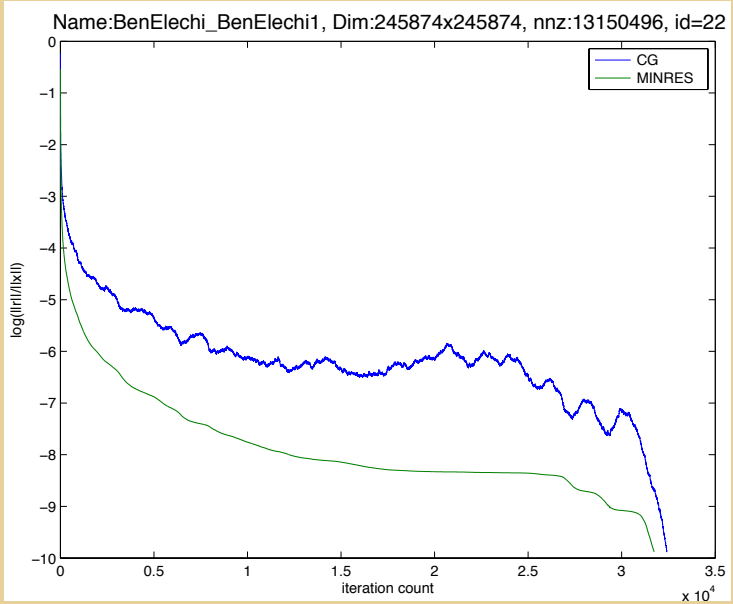
Plot $\log_{10} \|E^{(k)}\|$ for CG and MINRES

Backward Error of CG vs MINRES on $A \succ 0$









Theoretical properties

Theorem

For MINRES on a positive definite system $Ax = b$, $\|x_k\|$ increases monotonically, and the error $\|x^* - x_k\|$ decreases monotonically.

Corollary

For MINRES on a positive definite system $Ax = b$, backward error $\|r_k\|/\|x_k\|$ decreases monotonically to 0.

Relationship between four algorithms

$$\begin{aligned} \text{LSQR} &\equiv \text{CG on the normal equation} \\ \text{LSMR} &\equiv \text{MINRES on the normal equation} \end{aligned}$$

Part I: LSMR

What problems do LSQR and LSMR solve?

solve $Ax = b$ $\qquad$ $\min \|Ax - b\|_2$

$\min_{x\,:\,Ax=b} \|x\|_2$ $\qquad$ $\min \left\| \begin{pmatrix} A \\ \lambda I \end{pmatrix} x - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2$

Properties

- A is rectangular ($m \times n$) and often sparse
- A can be an operator ($\Rightarrow$ allows preconditioning)
- Av, $A^T u$ plus $O(m+n)$ operations per iteration

Why invent another algorithm?

Reason one

CG vs MINRES

Reason two

Monotone convergence of residuals

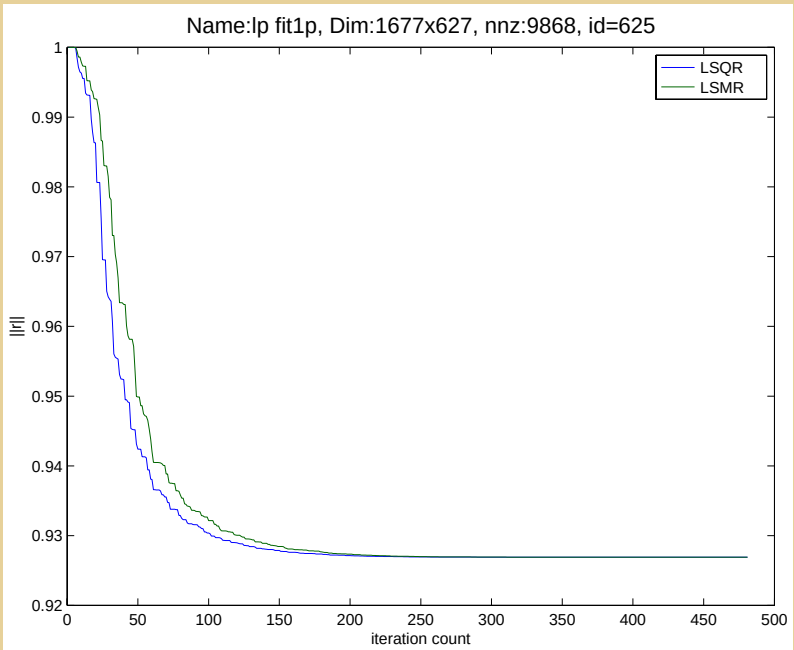
Monotone convergence of residuals

Measure of Convergence

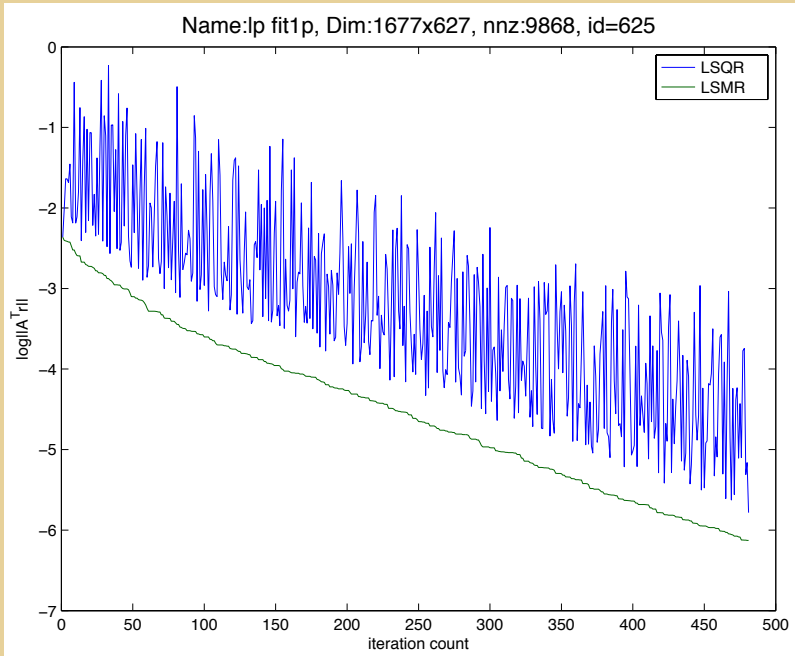
- $r_k = b - Ax_k$
- $\|r_k\| \to \|\hat{r}\|, \|A^T r_k\| \to 0$

LSQR
LSMR

$\|r_k\|$



$\log \|A^T r_k\|$



LSMR Derivation

Golub-Kahan bidiagonalization

Given A $(m \times n)$ and b $(m \times 1)$

Direct bidiagonalization

$$U^T \begin{pmatrix} b & A \end{pmatrix} \begin{pmatrix} 1 & \\ & V \end{pmatrix} = \begin{pmatrix} \times & \times & & \\ & \times & \times & \\ & & \times & \times \\ & & & \times \end{pmatrix} \quad \Rightarrow \quad \begin{pmatrix} b & AV \end{pmatrix} = U \begin{pmatrix} \beta_1 e_1 & B \end{pmatrix}$$

Iterative bidiagonalization $\mathrm{Bidiag}(A, b)$

1: $\beta_1 u_1 = b$, $\alpha_1 v_1 = A^T u_1$.
2: **for** $k = 1, 2, \ldots$ **do**
3: $\quad \beta_{k+1} u_{k+1} = A v_k - \alpha_k u_k$
4: $\quad \alpha_{k+1} v_{k+1} = A^T u_{k+1} - \beta_{k+1} v_k$.
5: **end for**

Golub-Kahan bidiagonalization (2)

The process can be summarized by

$$
\begin{aligned}
b &= U_{k+1}(\beta_1 e_1) \\
AV_k &= U_{k+1}B_k \\
A^T U_k &= V_k B_k^T \begin{pmatrix} I_k \\ 0 \end{pmatrix}
\end{aligned}
$$

where

$$
B_k = \begin{pmatrix}
\alpha_1 & & & \\
\beta_2 & \alpha_2 & & \\
& \ddots & \ddots & \\
& & \beta_k & \alpha_k \\
& & & \beta_{k+1}
\end{pmatrix}
\qquad
\begin{aligned}
U_k &= \begin{pmatrix} u_1 & \cdots & u_k \end{pmatrix} \\
V_k &= \begin{pmatrix} v_1 & \cdots & v_k \end{pmatrix}
\end{aligned}
$$

Golub-Kahan bidiagonalization (3)

V_k spans the Krylov subspace:

$$\text{span}\{v_1, \ldots, v_k\} = \text{span}\{A^T b,\ (A^T A)A^T b,\ \ldots,\ (A^T A)^{k-1} A^T b\}$$

Define $x_k = V_k y_k$

Subproblem to solve

$$\min_{y_k} \|r_k\| = \min_{y_k} \|\beta_1 e_1 - B_k y_k\| \qquad \text{(LSQR)}$$

$$\min_{y_k} \|A^T r_k\| = \min_{y_k} \left\| \bar{\beta}_1 e_1 - \begin{pmatrix} B_k^T B_k \\ \bar{\beta}_{k+1} e_k^T \end{pmatrix} y_k \right\| \qquad \text{(LSMR)}$$

where $r_k = b - Ax_k,\ \bar{\beta}_k = \alpha_k \beta_k$

Least-squares subproblem

$$\min_{y_k} \left\| \bar{\beta}_1 e_1 - \begin{pmatrix} B_k^T B_k \\ \bar{\beta}_{k+1} e_k^T \end{pmatrix} y_k \right\|$$

cf. MINRES on normal eqn

$$= \min_{y_k} \left\| \bar{\beta}_1 e_1 - \begin{pmatrix} R_k^T R_k \\ q_k^T R_k \end{pmatrix} y_k \right\| \qquad Q_{k+1} B_k = \begin{pmatrix} R_k \\ 0 \end{pmatrix}, \quad R_k^T q_k = \bar{\beta}_{k+1} e_k$$

$$= \min_{t_k} \left\| \bar{\beta}_1 e_1 - \begin{pmatrix} R_k^T \\ \varphi_k e_k^T \end{pmatrix} t_k \right\| \qquad t_k \equiv R_k y_k, \quad q_k = \varphi_k e_k$$

$$= \min_{t_k} \left\| \begin{pmatrix} z_k \\ \bar{\zeta}_{k+1} \end{pmatrix} - \begin{pmatrix} \bar{R}_k \\ 0 \end{pmatrix} t_k \right\| \qquad \bar{Q}_{k+1} \begin{pmatrix} R_k^T & \bar{\beta}_1 e_1 \\ \varphi_k e_k^T & 0 \end{pmatrix} = \begin{pmatrix} \bar{R}_k & z_k \\ 0 & \tilde{\zeta}_{k+1} \end{pmatrix}$$

Two cheap QRs, then

$$\bar{R}_k t_k = z_k, \quad R_k y_k = t_k, \quad x_k = V_k y_k$$

Least-squares subproblem (2)

Remember

$$\bar{R}_k t_k = z_k = \begin{pmatrix} \zeta_1 \\ \vdots \\ \zeta_k \end{pmatrix}, \quad R_k y_k = t_k, \quad x_k = V_k y_k$$

$\bar{R}_k$ and R_k both upper-bidiagonal

Two forward substitutions, n rhs's

$$\begin{aligned} x_k &= V_k y_k \\ &= W_k t_k \\ &= \bar{W}_k z_k \\ &= x_{k-1} + \zeta_k \bar{w}_k \end{aligned}$$

$$\begin{aligned} R_k^T W_k^T &= V_k^T \\ \bar{R}_k^T \bar{W}_k^T &= W_k^T \end{aligned}$$

Flow chart of LSMR

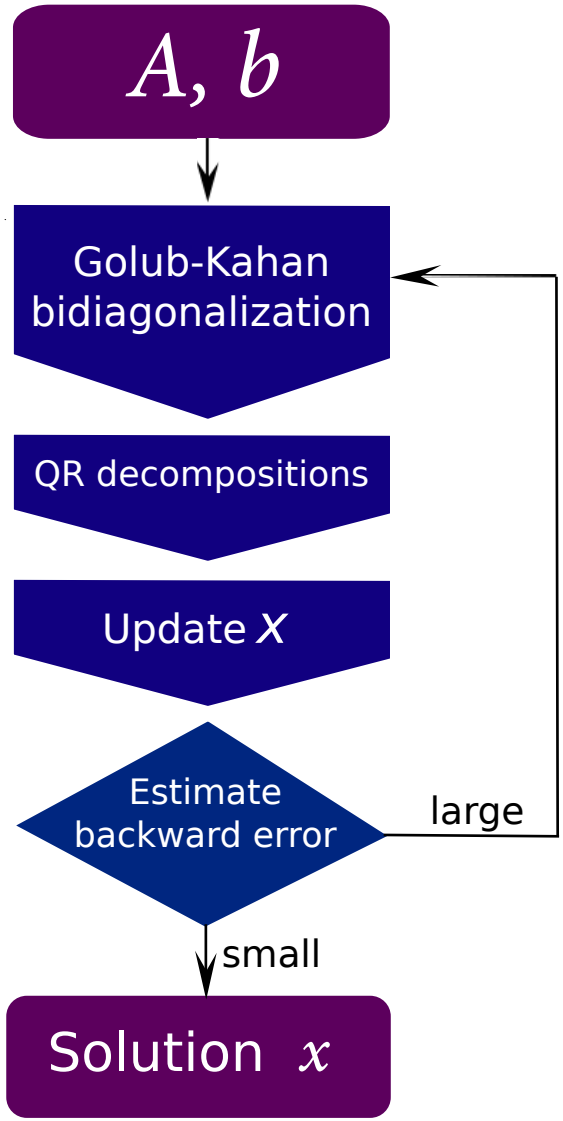
Computational cost

$Av, A^Tv, 3m + 3n$

$O(1)$

$3n$

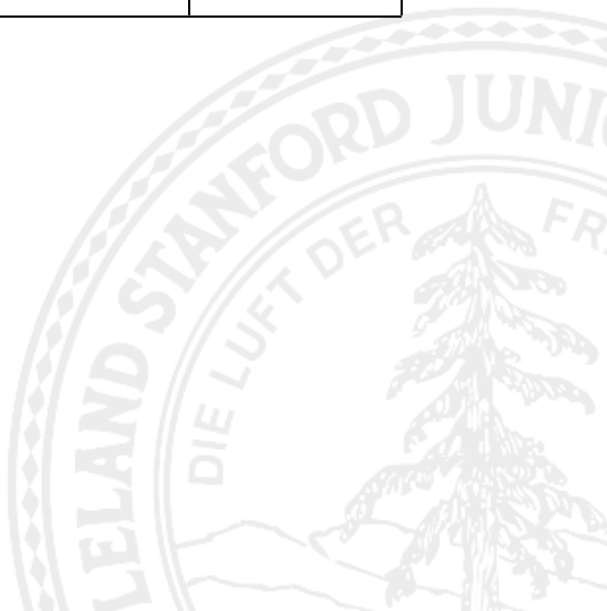
$O(1)$



Computational and storage requirement

	Storage		Work	
	m	n	m	n
MINRES on $A^TAx = A^Tb$	Av_1	x, v_1, v_2, w_1, w_2		8
LSQR	Av, u	x, v, w	3	5
LSMR	Av, u	$x, v, h, \bar{h}$	3	6

where h_k, $\bar{h}_k$ are scalar multiples of w_k, $\bar{w}_k$

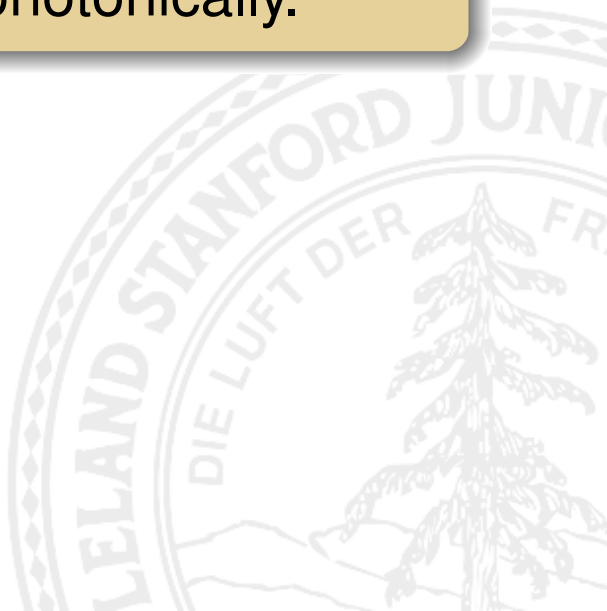


Monotonicity of norms

Theorem

For LSMR on a least-squares problem $\min \|Ax - b\|$,

- $\|x_k\|$ increases monotonically,
- $\|A^T r_k\|$, $\|r_k\|$, and $\|x^* - x_k\|$ decreases monotonically.



LSMR Experiments

Overdetermined systems

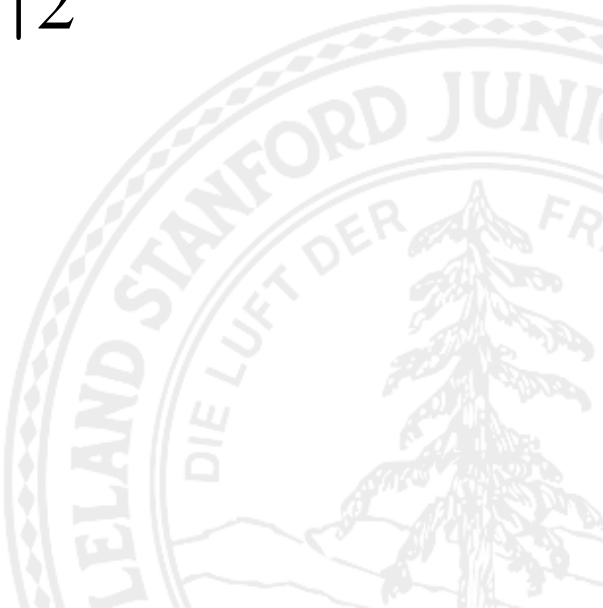
Test Data

- Tim Davis, University of Florida Sparse Matrix Collection
- `LPnetlib`: Linear Programming Problems
- `A = (Problem.A)'  b = Problem.c` (127 problems)

$$\text{Solve } \min \|Ax - b\|_2$$

with LSQR and LSMR

- Backward error tests: $\text{nnz}(A) \leq 63220$
- Reorthogonalization: $\text{nnz}(A) \leq 15977$



Backward error – estimates

$$A^T A\hat{x} = A^T b \qquad \hat{r} = b - A\hat{x} \qquad \text{exact}$$
$$(A + E_i)^T (A + E_i)x = (A + E_i)^T b \qquad r = b - Ax \qquad \text{any } x$$

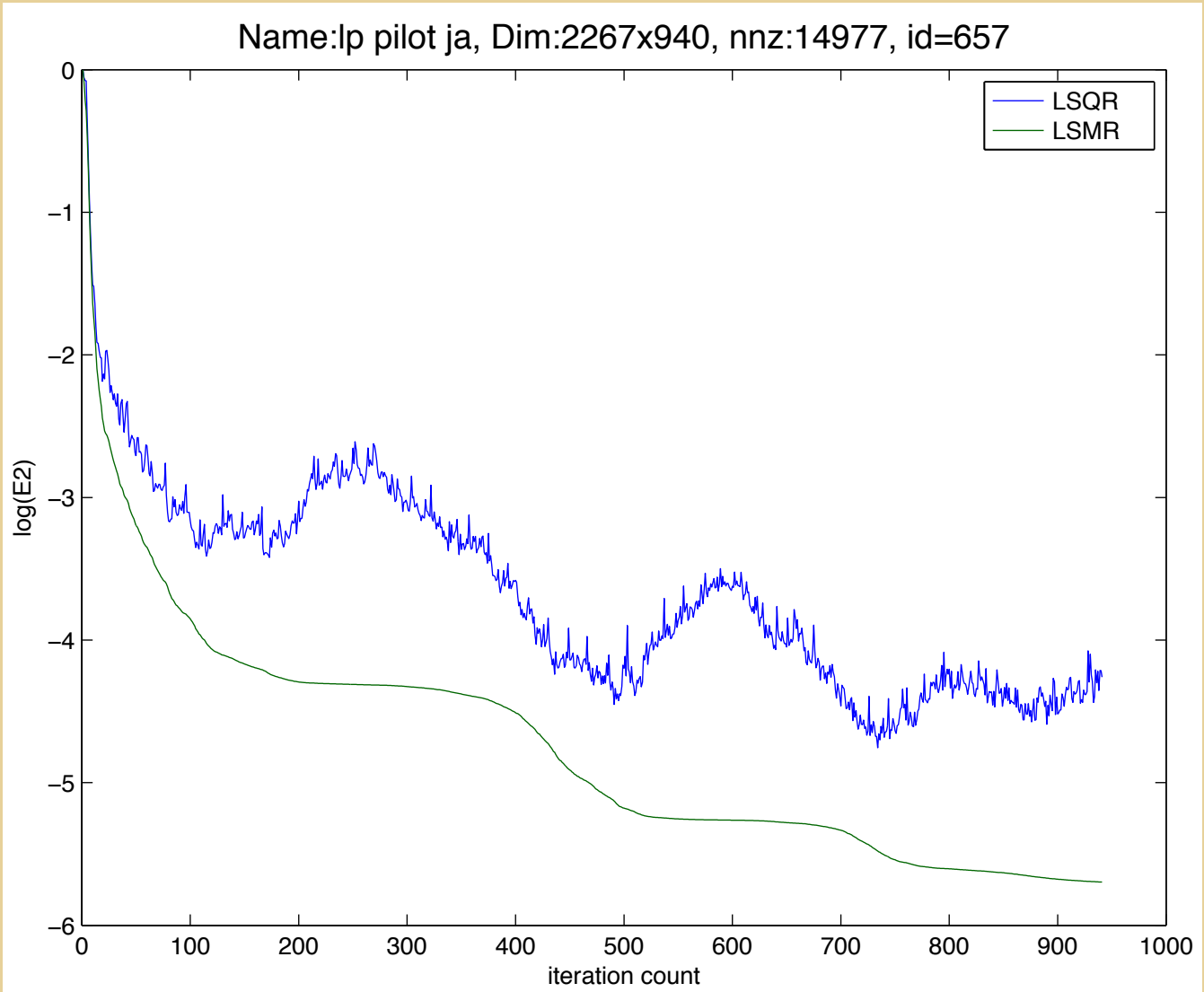
Two estimates given by Stewart (1975 and 1977)

$$E_1 = \frac{ex^T}{\|x\|^2} \qquad \|E_1\| = \frac{\|e\|}{\|x\|} \qquad e = \hat{r} - r$$
$$E_2 = -\frac{rr^T A}{\|r\|^2} \qquad \|E_2\| = \frac{\|A^T r\|}{\|r\|} \qquad \text{computable}$$

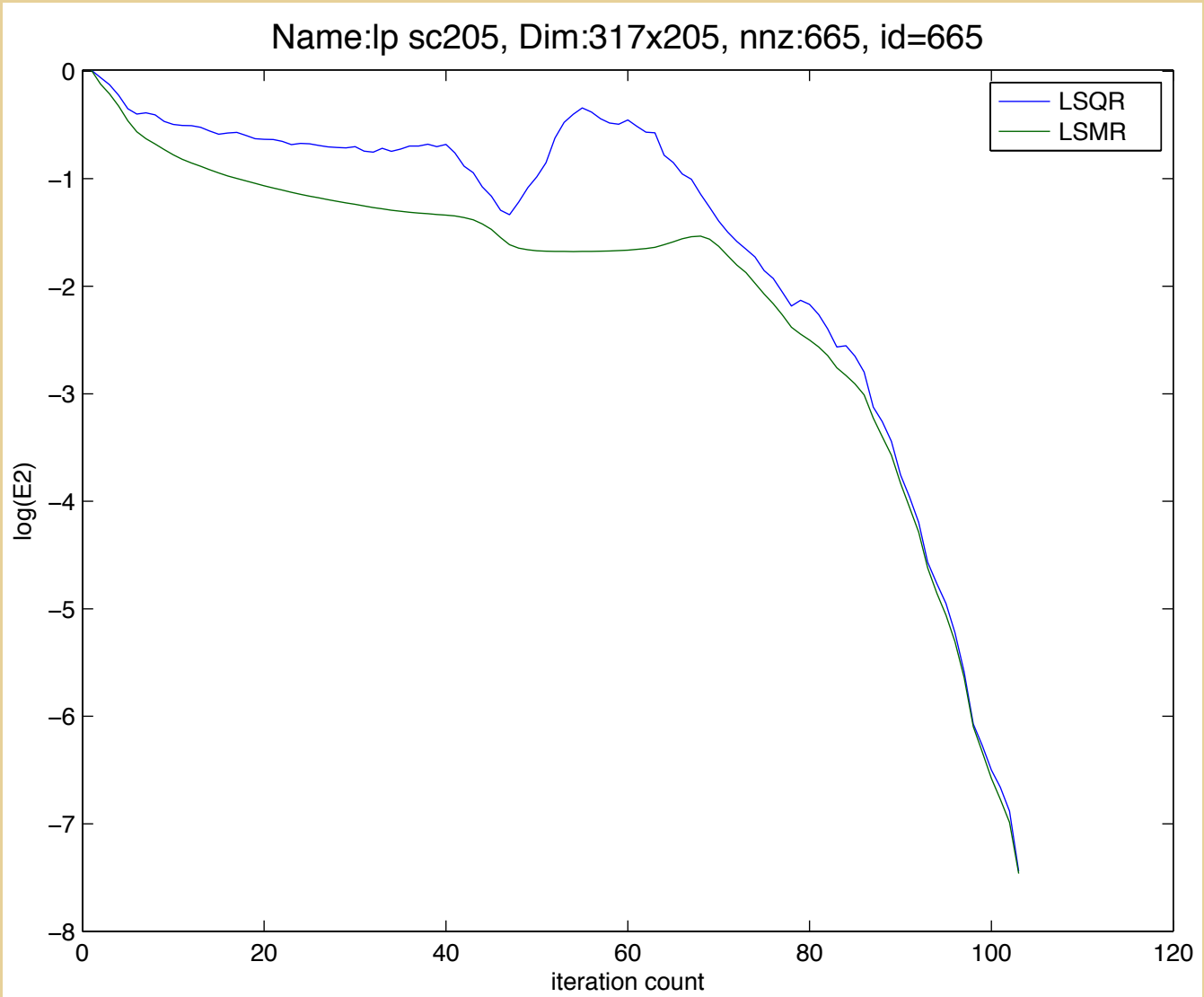
Theorem

$$\|E_2^{\text{LSMR}}\| \leq \|E_2^{\text{LSQR}}\|$$

$\log_{10} \|E_2\|$ for LSQR and LSMR – typical



$\log_{10} \|E_2\|$ for LSQR and LSMR – rare



Backward error - optimal

$$\mu(x) \equiv \min_E \|E\| \quad \text{st} \quad (A+E)^T(A+E)x = (A+E)^T b$$

Exact $\mu(x)$ (Waldén, Karlson, & Sun 1995, Higham 2002)

$$C \equiv \left[A \quad \frac{\|r\|}{\|x\|}\left(I - \frac{rr^T}{\|r\|^2}\right)\right] \qquad \mu(x) = \sigma_{\min}(C)$$

Backward error - optimal

$$\mu(x) \equiv \min_E \|E\| \quad \text{st} \quad (A+E)^T(A+E)x = (A+E)^T b$$

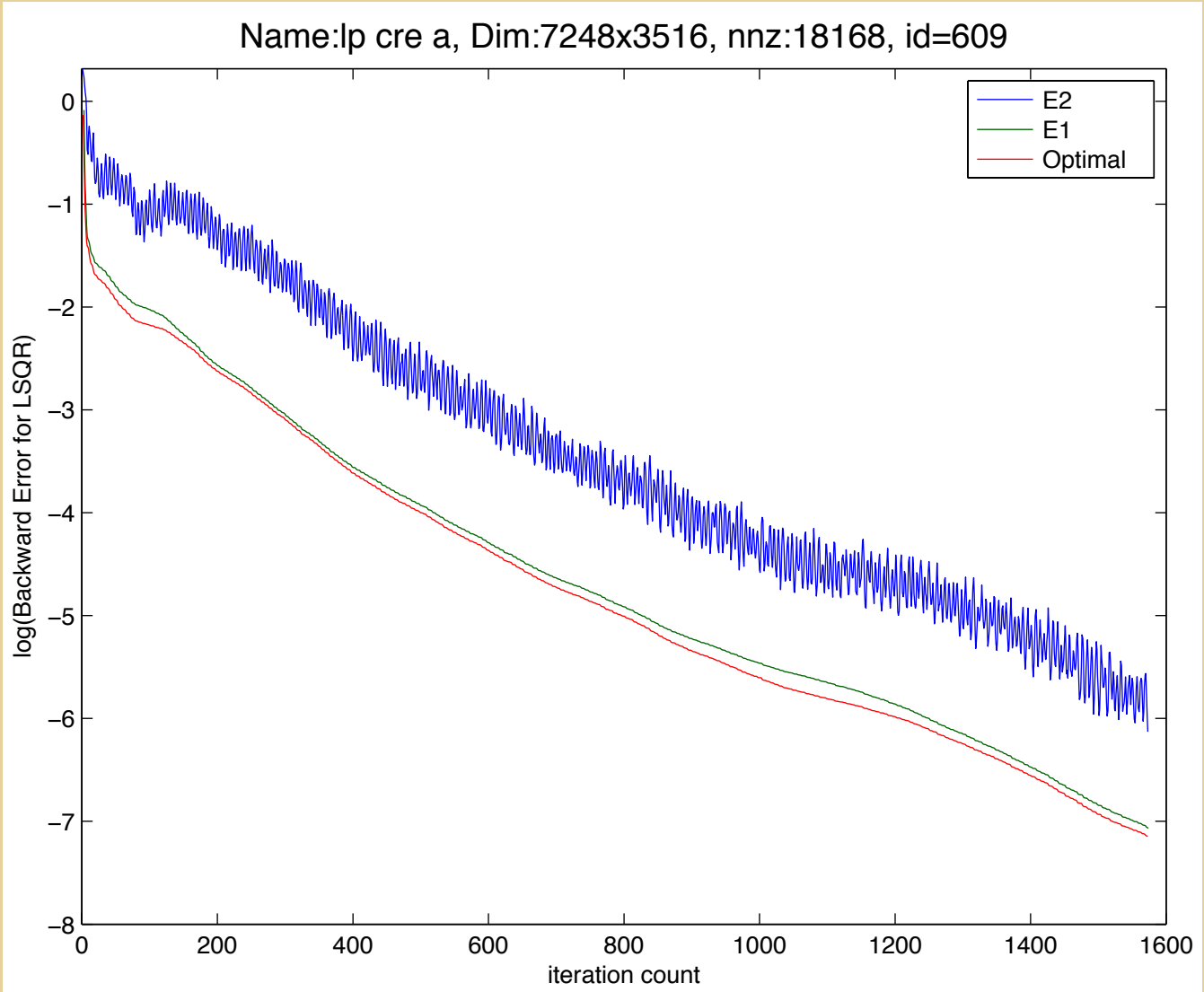
Cheaper estimate $\tilde{\mu}(x)$ (Grcar, Saunders, & Su 2007)

$$K = \begin{pmatrix} A \\ \frac{\|r\|}{\|x\|} I \end{pmatrix} \qquad v = \begin{pmatrix} r \\ 0 \end{pmatrix}$$

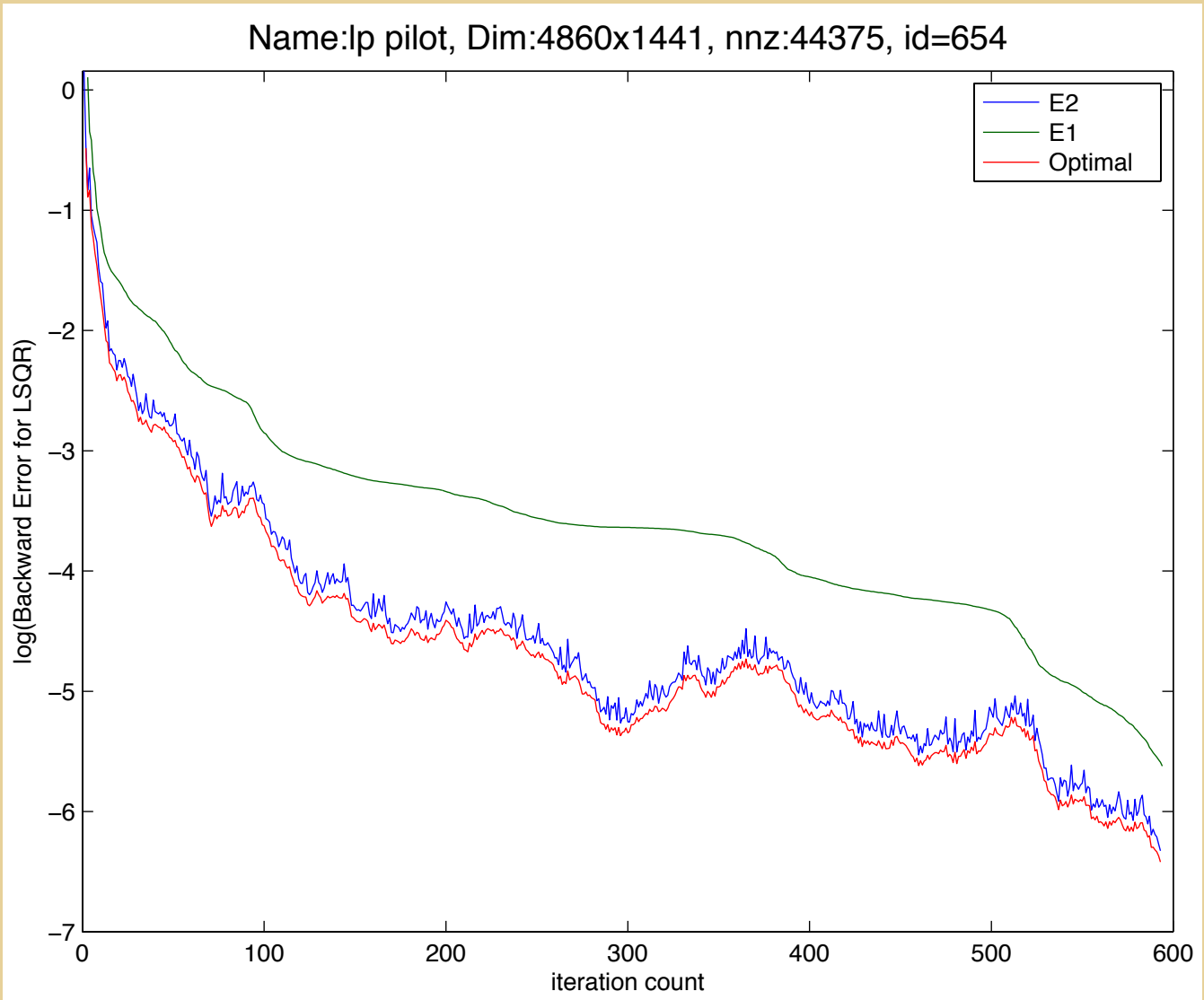
$$\min_y \|Ky - v\| \qquad \tilde{\mu}(x) = \frac{\|Ky\|}{\|x\|}$$

```
r    = b - A*x;
p    = colamd(A);
eta  = norm(r)/norm(x);
K    = [A(:,p); eta*speye(n)];
v    = [r; zeros(n,1)];
[c,R]   = qr(K,v,0);
mutilde = norm(c)/norm(x);
```

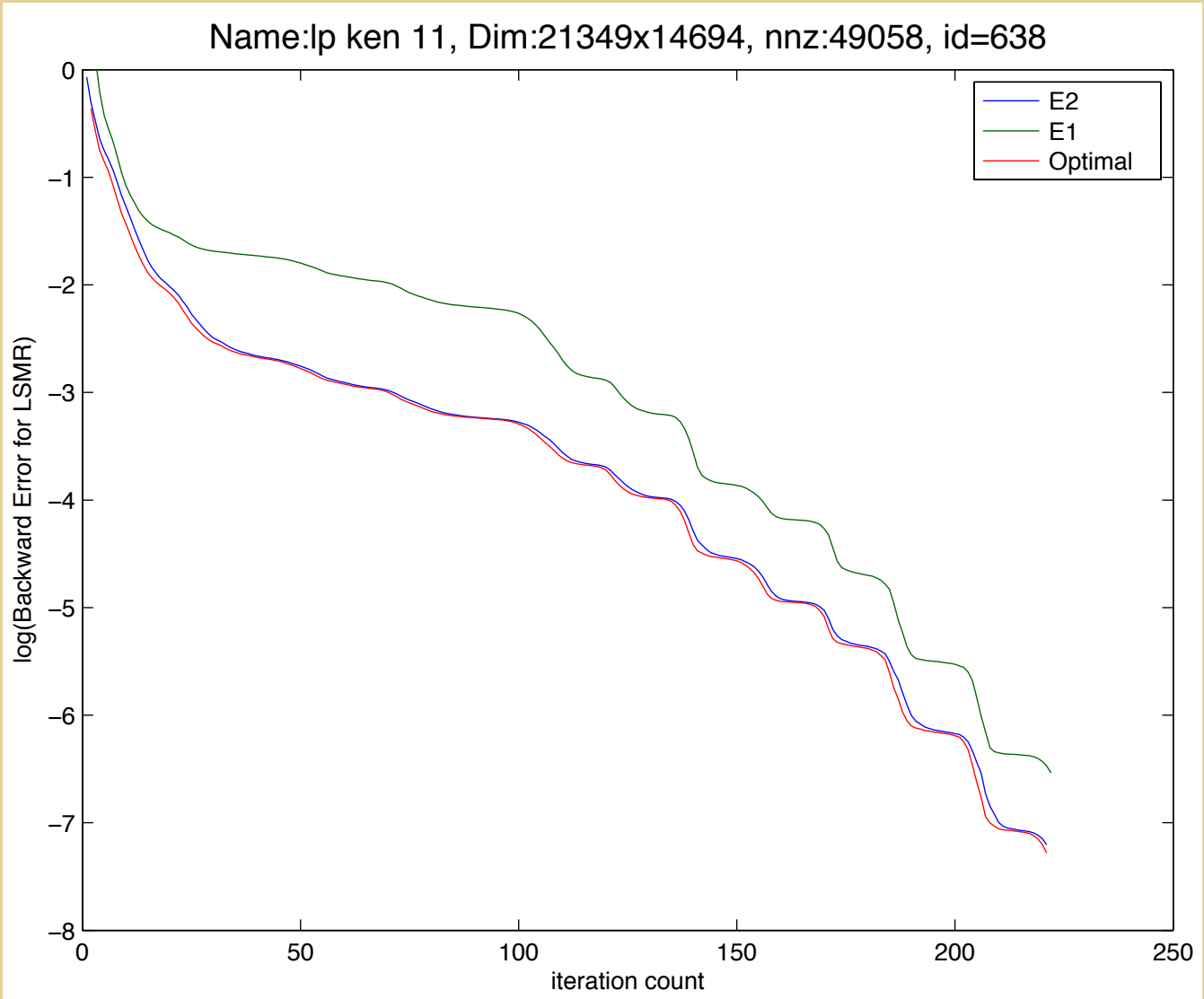
Backward errors for LSQR – typical



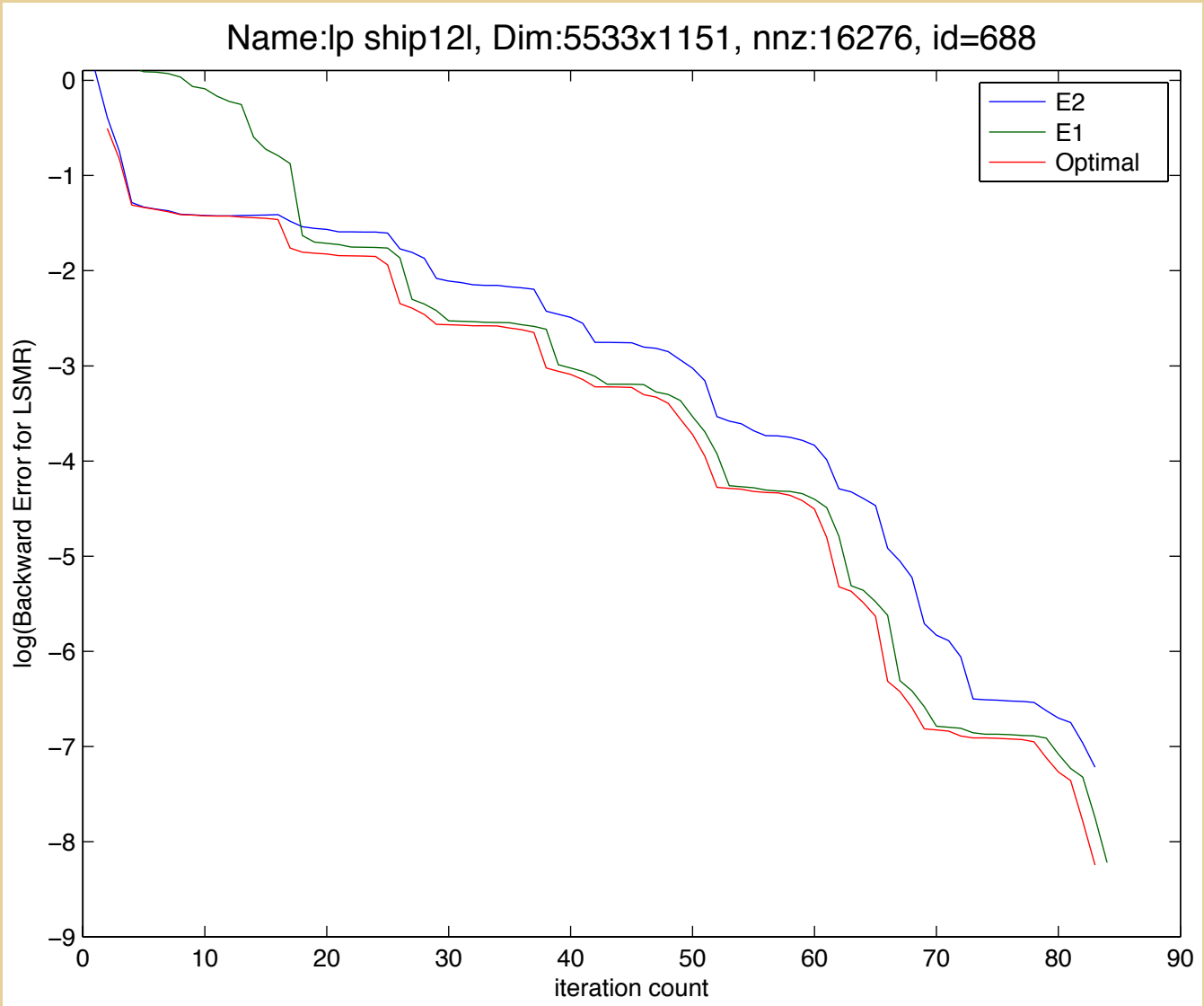
Backward errors for LSQR – rare



Backward errors for LSMR – typical



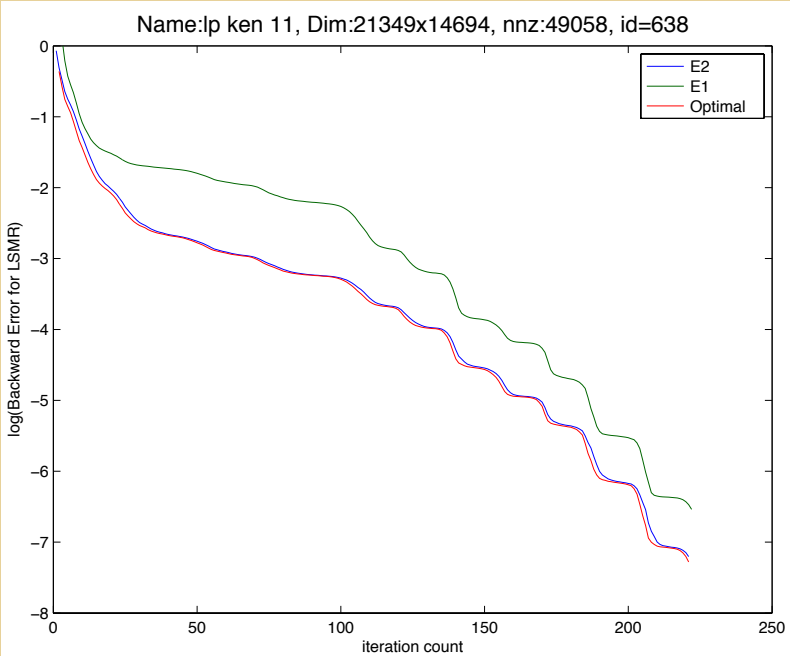
Backward errors for LSMR – rare



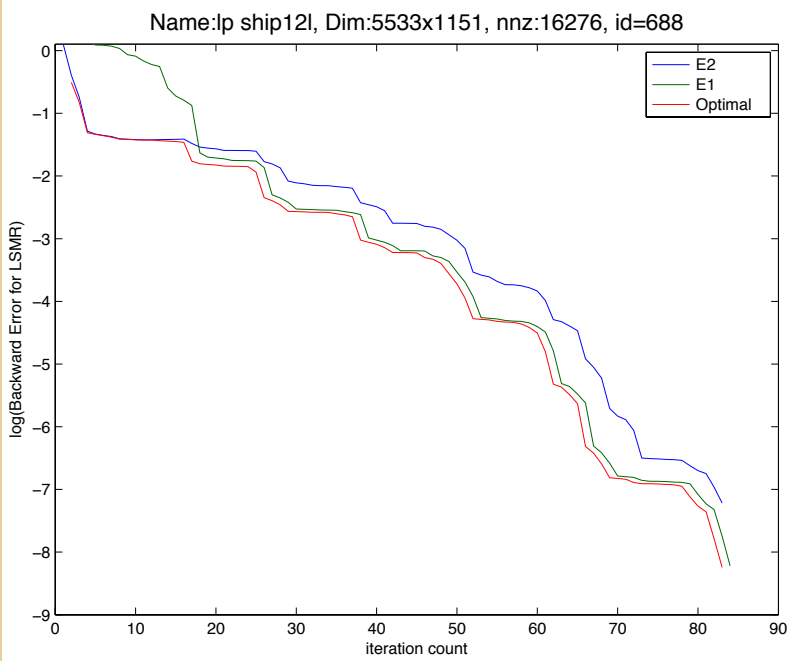
For LSMR

$\|E_2\| \approx$ optimal BE almost always

Typical: $\|E_2\| \approx \tilde{\mu}(x)$



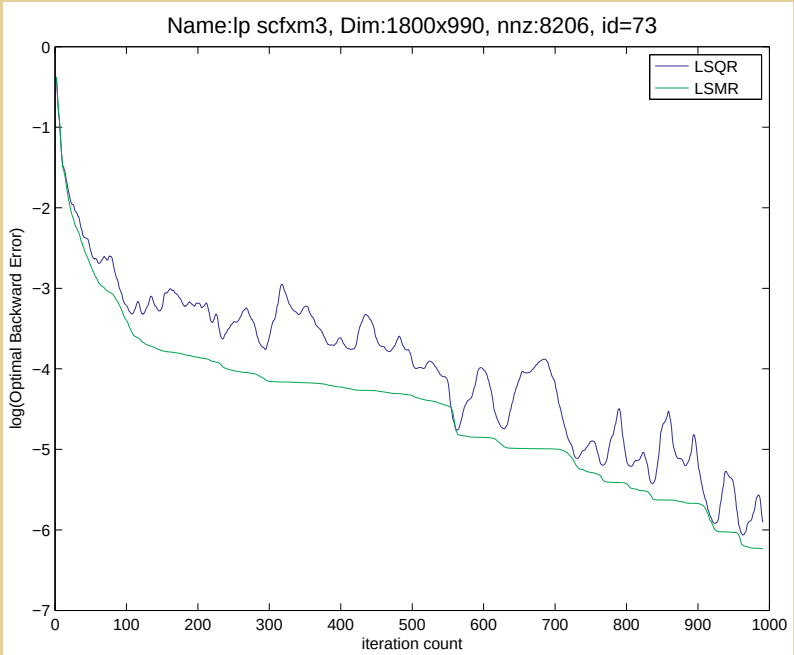
Rare: $\|E_1\| \approx \tilde{\mu}(x)$



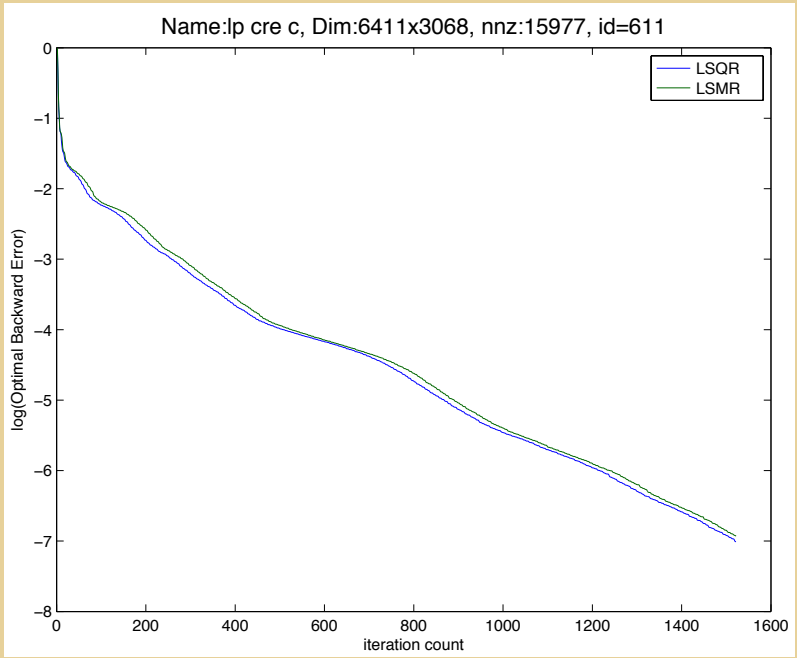
Optimal backward errors $\tilde{\mu}(x)$

Seem monotonic for LSMR Usually not for LSQR

Typical for LSQR and LSMR



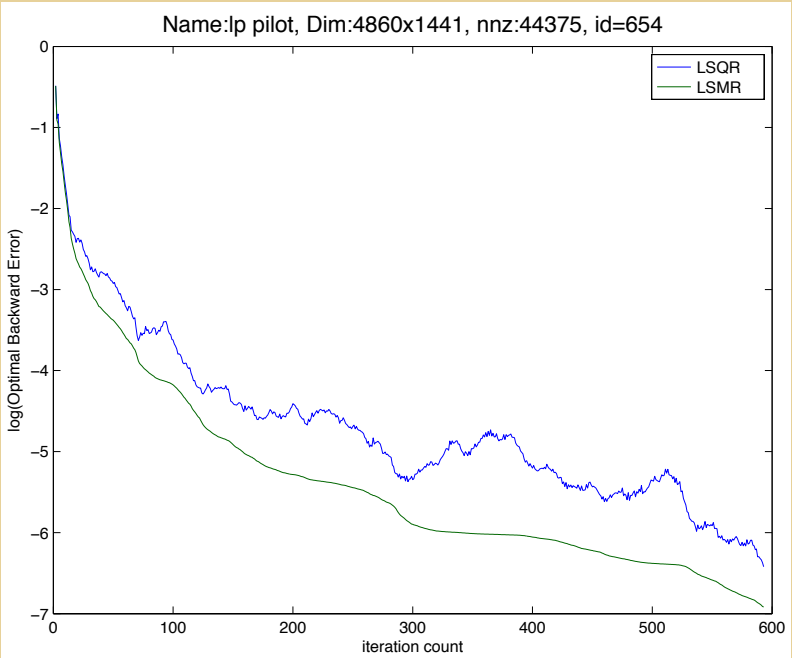
Rare LSQR, typical LSMR



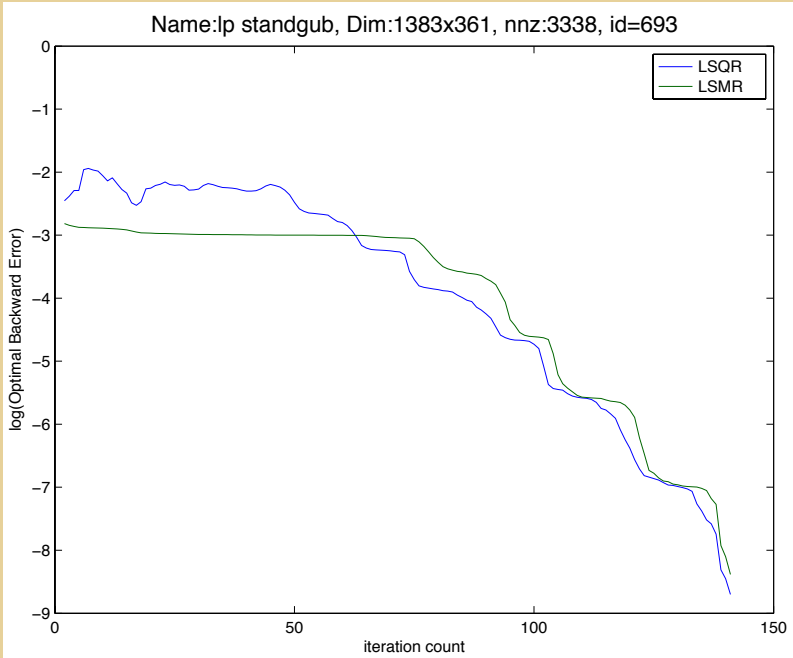
Optimal backward errors

$\tilde{\mu}(x^{\mathrm{LSMR}}) \leq \tilde{\mu}(x^{\mathrm{LSQR}})$ almost always

Typical



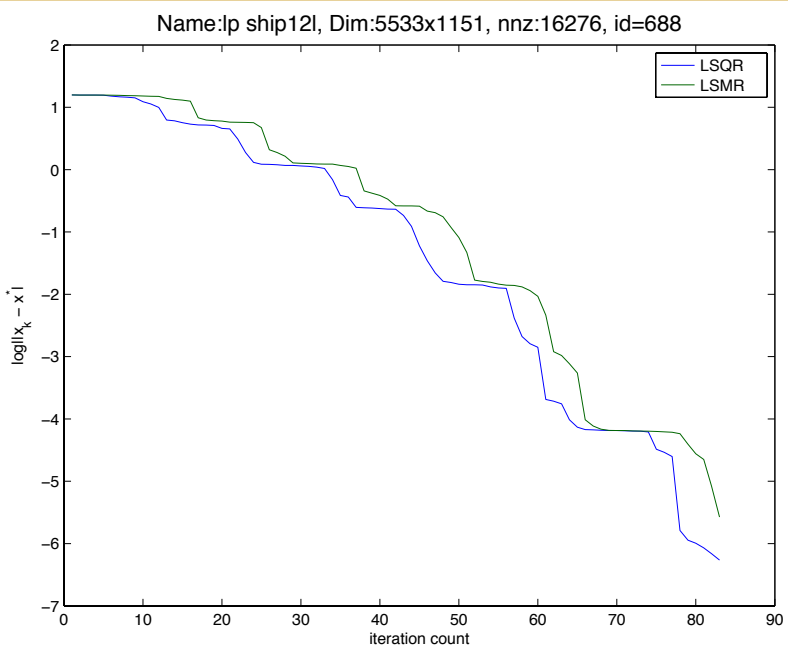
Rare

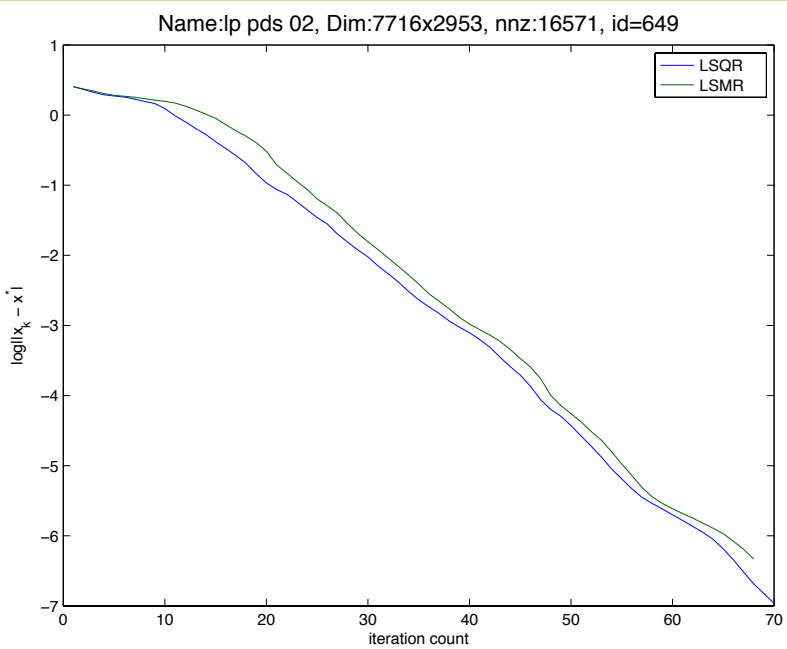


Errors in x_k

- $\|x^{\text{LSQR}} - x^*\| \leq \|x^{\text{LSMR}} - x^*\|$ seems true

$\|x_k - x^*\|$ for LSMR and LSQR





Space-time trade-offs

LSMR is well-suited for limited memory computations.

What if we have

- more memory
- Av expensive

Can we speed things up?

Some ideas:

- Reorthogonalization
- Local reorthogonalization

Reorthogonalization

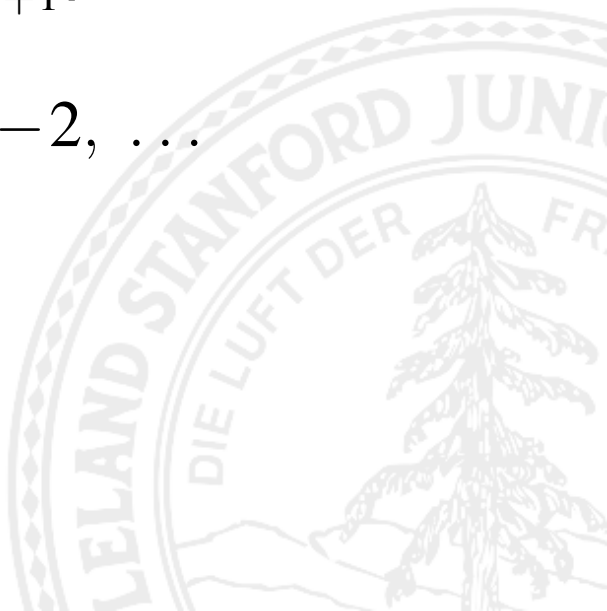
Golub-Kahan process

Infinite precision	Finite precision
U_k, V_k orthonormal	Lose orthogonality
At most $\min(m,n)$ iterations	Could take $10n$ or more

Apply modified Gram-Schmidt to u_{k+1} and/or v_{k+1}:

$$u \leftarrow u - (u_j^T u)u_j \qquad j = k,\ k-1,\ k-2,\ \ldots$$

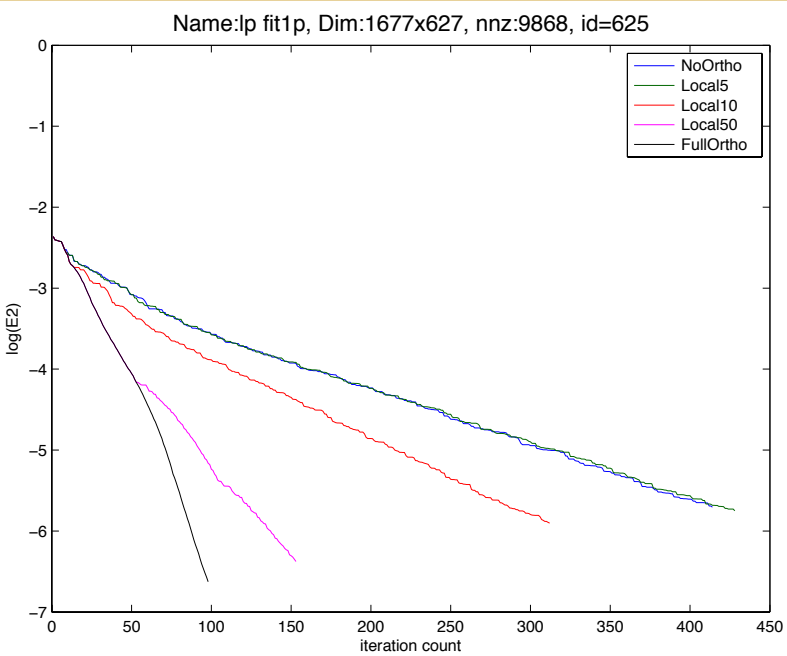
(similarly for v)

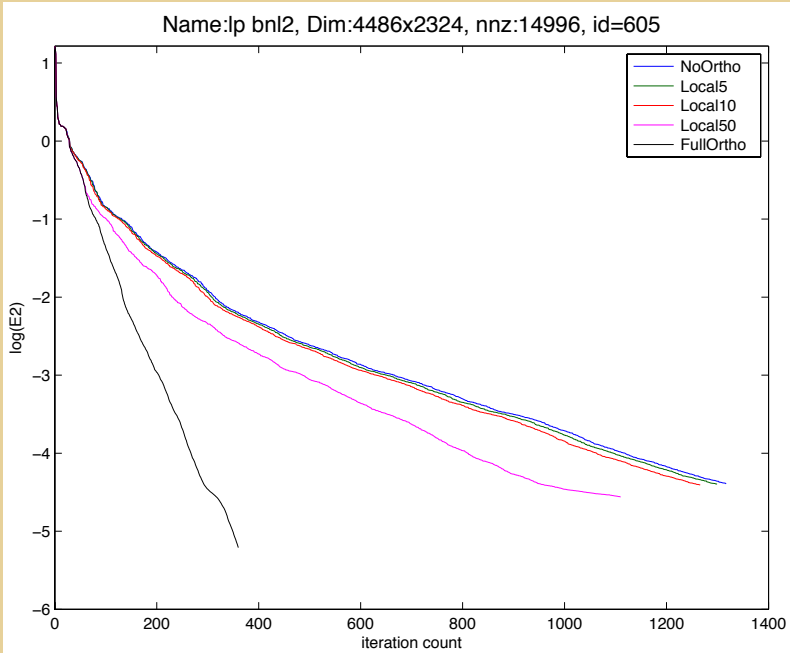


Local reorthogonalization

- Reorthogonalize wrto only the last l vectors
- Partial speed-up
- Less memory
- Depends on efficiency of Av and $A^T u$

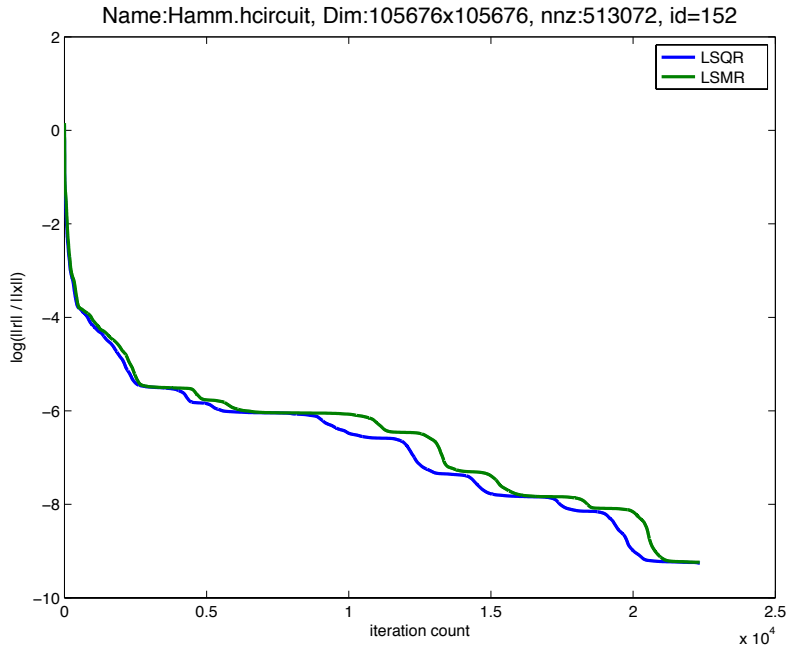
Speed up with local reorthogonalization

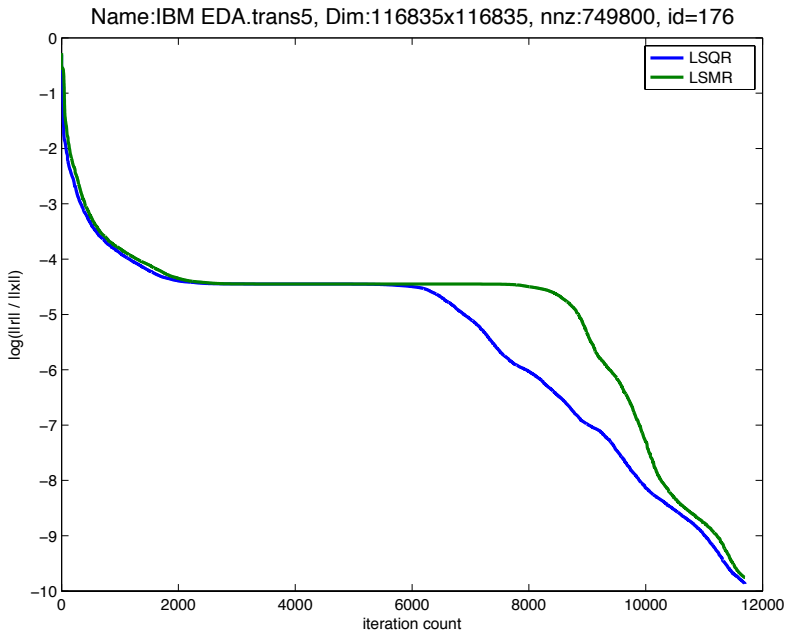




Square consistent systems

- $Ax = b$
- Backward error: $\frac{\|r_k\|}{\|x_k\|}$
- LSQR slightly faster than LSMR in most cases





Underdetermined systems

Infinitely many solutions

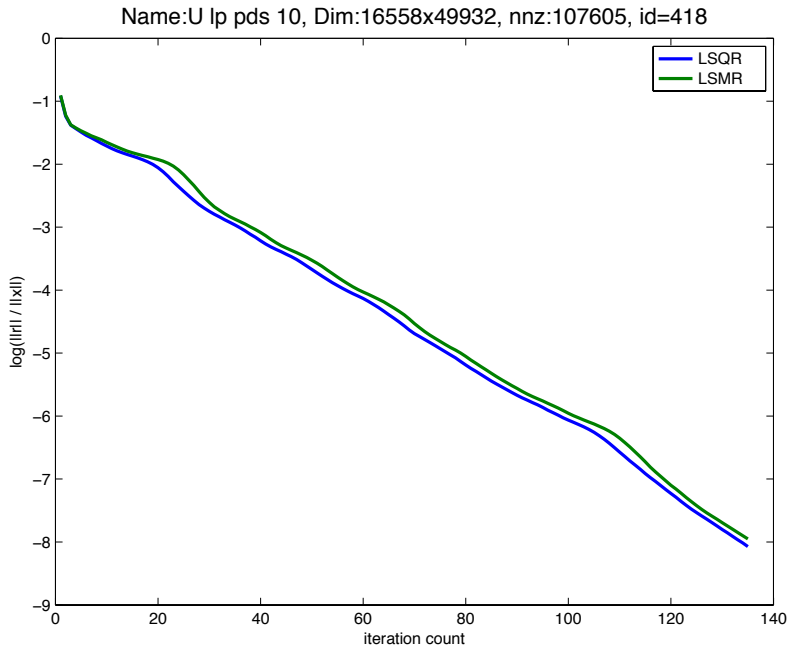
$$Ax = b$$

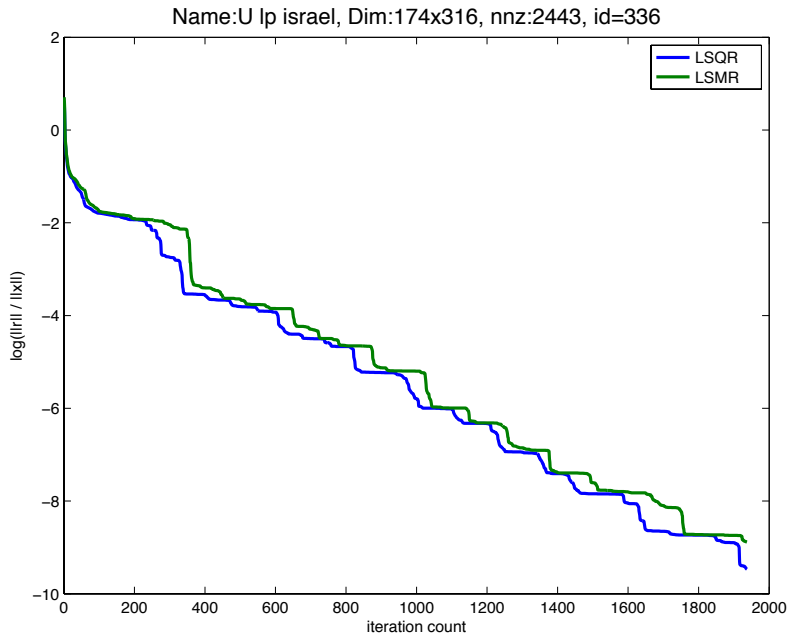
Unique solution

$$\min_{Ax=b} \|x\|_2$$

Theorem

LSQR and LSMR both return the minimum-norm solution





Part II: AMRES

What problem does AMRES solve?

Negatively damped least-squares

$$(A^T A - \delta^2 I)x = A^T b$$

Examples: Curtis-Reid scaling, singular vectors, total least-squares

Equivalent augmented system

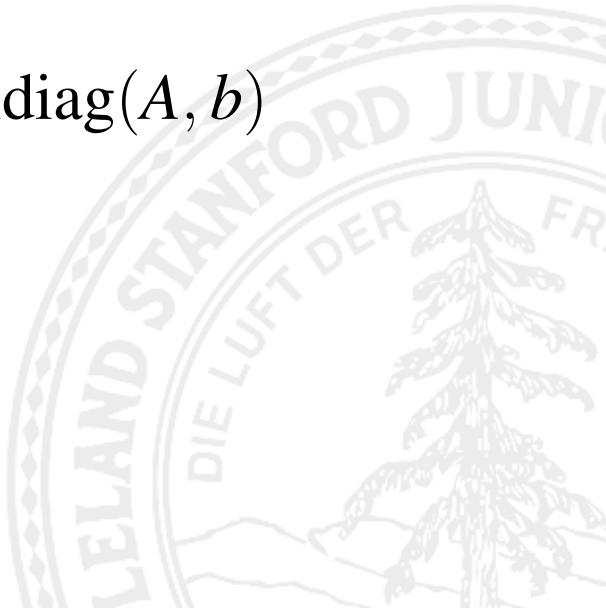
$$\begin{pmatrix} \delta I & A \\ A^T & \delta I \end{pmatrix} \begin{pmatrix} s \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}$$

MINRES for Augmented Systems

$$\begin{pmatrix} \gamma I & A \\ A^T & \delta I \end{pmatrix} \begin{pmatrix} s \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}$$

AMRES = specialized MINRES

- $\gamma,\ \delta$ can be any nonzero scalars of the same sign ($\gamma\delta > 0$)
- If $\gamma\delta \leq 0$, use LSQR or LSMR
- Use the Golub-Kahan bidiagonalization $\mathrm{Bidiag}(A, b)$
- Compute only x (or only s)
- Half the computing cost of MINRES



AMRES Derivation

Lanczos on augmented system

Lanczos process $\mathrm{Tridiag}(\underbrace{\begin{pmatrix} \gamma I & A \\ A^T & \delta I \end{pmatrix}}_{\hat{A}}, \underbrace{\begin{pmatrix} b \\ 0 \end{pmatrix}}_{\hat{b}})$ generates

$$\hat{T}_{2k} = \begin{pmatrix} \gamma & \alpha_1 & & & & \\ \alpha_1 & \delta & \beta_2 & & & \\ & \beta_2 & \gamma & \alpha_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \beta_k & \gamma & \alpha_k \\ & & & & \alpha_k & \delta \end{pmatrix}, \qquad \hat{V}_{2k} = \begin{pmatrix} u_1 & & u_2 & & \dots & u_k \\ & v_1 & & v_2 & \dots & & v_k \end{pmatrix}$$

- Half of the entries in $\hat{V}_{2k}$ are 0
- Golub-Kahan $\mathrm{Bidiag}(A, b)$ generates the same $\{(\alpha_i, \beta_1, v_i, u_i)\}_{i=1}^{k}$ with half the cost of Lanczos

Least squares subproblem

$$\begin{aligned}\widehat{r}_k &= \hat{b} - \hat{A}\widehat{x}_{2k} \\ &= \hat{b} - \hat{A}\hat{V}_{2k}\hat{y}_{2k} \\ &= \hat{V}_{2k+1}\left(\beta_1 e_1 - \hat{H}_{2k}\hat{y}_{2k}\right)\end{aligned} \qquad \begin{aligned}\hat{x}_{2k} &= \begin{pmatrix} s_{2k} \\ x_{2k} \end{pmatrix} = \hat{V}_{2k}\hat{y}_{2k} \\ \hat{H}_{2k} &= \begin{pmatrix} \hat{T}_{2k} \\ e_k^T \beta_{k+1} \end{pmatrix}\end{aligned}$$

$$\begin{aligned}\min \|\widehat{r}_{2k}\| &= \min \|\beta_1 e_1 - \hat{H}_{2k}\hat{y}_{2k}\| \\ &= \min \left\| Q_{2k}^T \left(\begin{pmatrix} z_{2k} \\ \bar{\zeta}_{2k+1} \end{pmatrix} - \begin{pmatrix} R_{2k} \\ 0 \end{pmatrix} \hat{y}_{2k} \right) \right\| \\ &= \min \left\| \begin{pmatrix} z_{2k} \\ \bar{\zeta}_{2k+1} \end{pmatrix} - \begin{pmatrix} R_{2k} \\ 0 \end{pmatrix} \hat{y}_{2k} \right\| \quad \Rightarrow z_{2k} = R_{2k}\hat{y}_{2k}\end{aligned}$$

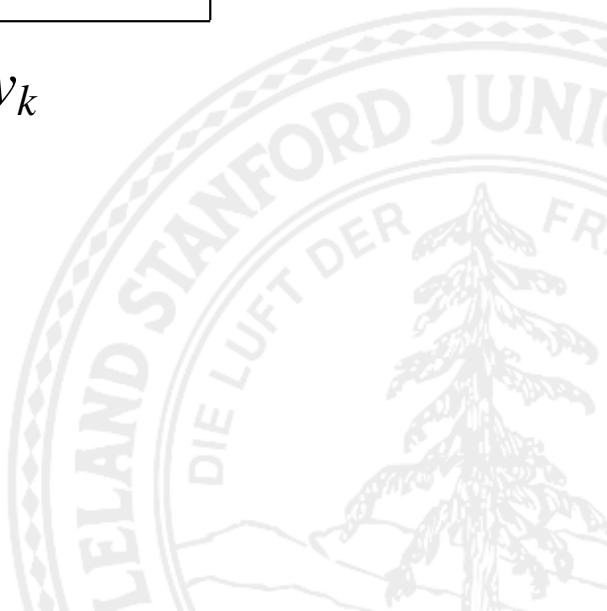
$$\begin{aligned}x_{2k} &= \begin{pmatrix} 0 & v_1 & 0 & v_2 & \dots & 0 & v_k \end{pmatrix} \hat{y}_{2k} \\ &= W_{2k} z_{2k} \\ &= x_{2k-2} + \zeta_{2k-1} w_{2k-1} + \zeta_{2k} w_{2k}\end{aligned}$$

$$R_{2k}^T W_{2k}^T = \begin{pmatrix} 0 \\ v_1^T \\ 0 \\ v_2^T \\ \vdots \end{pmatrix}$$

Computational and storage cost

	Storage		Work	
	m	n	m	n
LSQR	Av, u	x, v, w	3	5
LSMR	Av, u	$x, v, h, \bar{h}$	3	6
AMRES	Av, u	x, v, h_{k-1}, h_k	3	9

where h_k is scalar multiples of w_k



Curtis-Reid Scaling

Matrix scaling

Goal

Scale the row and columns of a sparse matrix A to make all nonzero entries approximately 1.

$$\bar{A} = RAC \qquad R = \operatorname{diag}(\beta^{-r_i}) \qquad C = \operatorname{diag}(\beta^{-c_j})$$

$$|\bar{a}_{ij}| \approx 1 \quad \Rightarrow \quad \beta^{-r_i}|a_{ij}|\beta^{-c_j} \approx 1$$
$$\Rightarrow \quad -r_i + \log_\beta |a_{ij}| - c_j \approx 0$$

- Fulkerson & Wolfe 1962: $\min\limits_{r_i,c_j} \max\limits_{a_{ij}\neq 0} |\log_\beta |a_{ij}| - r_i - c_j|$
- Hamming 1971: $\min\limits_{r_i,c_j} \sum\limits_{a_{ij}\neq 0} (\log_\beta |a_{ij}| - r_i - c_j)^2$

Curtis & Reid 1972

Normal equation for Hamming's optimization problem:

$$\begin{pmatrix} D_1 & E \\ E^T & D_2 \end{pmatrix} \begin{pmatrix} r \\ c \end{pmatrix} = \begin{pmatrix} s \\ t \end{pmatrix}$$

- Positive semidefinite system
- $E = A$ with nonzero a_{ij} changed to 1
- D_1 = `diag`(nnz(rows of A)), D_2 = `diag`(nnz(cols of A))
- Curtis & Reid derived specialized CG for this system
- Half the work of normal CG

AMRES is also applicable after some transformations.

Curtis-Reid scaling via specialized CG and AMRES

Apply solvers to

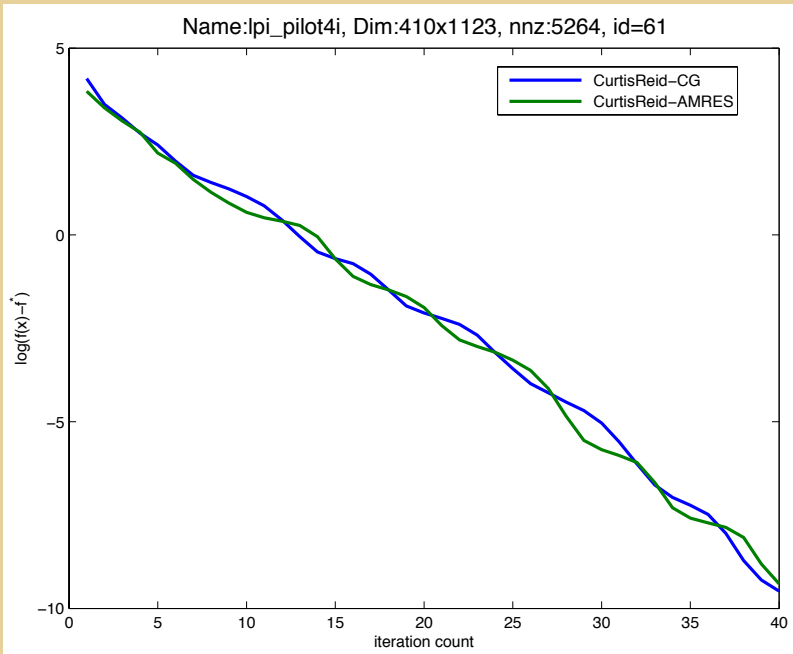
$$\begin{pmatrix} D_1 & E \\ E^T & D_2 \end{pmatrix} \begin{pmatrix} r \\ c \end{pmatrix} = \begin{pmatrix} s \\ t \end{pmatrix}$$

and obtain logs of row and column scales: r and c

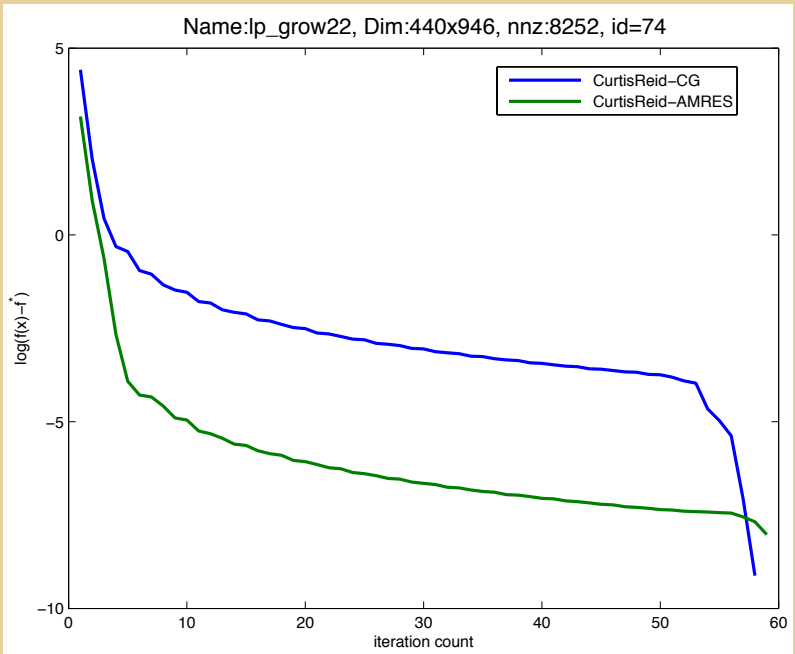
Compare values of LS objective at each iteration:

- At itn k, $f_k(r,c) = \sum_{a_{ij}\neq 0}(\log_{10}|a_{ij}| - r_i - c_j)^2$
- Run each solver K iterations
- Let $f_{\min} =$ smallest final f_K for specialized CG or AMRES
- Plot $\log_{10}(f_k - f_{\min})$ for each solver

Similar



AMRES favorable



- Similar performance on many problems
- In some cases, AMRES converges to a reasonably good solution in very few iterations

Singular Vectors

Finding singular vectors

For any matrix A, a singular triple (σ, u, v) satisfies

$$Av = \sigma u, \qquad A^T u = \sigma v$$

All singular vectors (dense A)

Paul Willems (2009)

Approximate singular vector (sparse A)

When $\tilde{v} \approx v$ is known, find v (and hence σ and u)

Known singular value (sparse A)

When σ is known, find v (and hence u)

Case I

Approximate Singular Vector

Rayleigh quotient iteration

RQI for eigenvector

1: Given eigenvalue and eigenvector estimates ρ_0, x_0
2: **for** $q = 1, 2, \ldots$ **do**
3: Solve $(A - \rho_{q-1} I)t = x_{q-1}$
4: $x_q = t/\|t\|$
5: $\rho_q = x_q^T A x_q$
6: **end for**

RQI for singular vector (RQI-MINRES)

1: Given singular value and right singular vector estimates ρ_0, x_0
2: **for** $q = 1, 2, \ldots$ **do**
3: Solve $(A^T A - \rho_{q-1}^2 I)t = x_{q-1}$
4: $x_q = t/\|t\|$
5: $\rho_q = \|Ax_q\|$
6: **end for**

Rayleigh quotient iteration (2)

$$\begin{pmatrix} -\rho_{q-1} I & A \\ A^T & -\rho_{q-1} I \end{pmatrix} \begin{pmatrix} s \\ t \end{pmatrix} = \begin{pmatrix} 0 \\ x_{q-1} \end{pmatrix}$$

Scale the right-hand side:

$$\begin{pmatrix} -\rho_{q-1} I & A \\ A^T & -\rho_{q-1} I \end{pmatrix} \begin{pmatrix} s \\ t \end{pmatrix} = \begin{pmatrix} 0 \\ \rho_{q-1} x_{q-1} \end{pmatrix}$$

Shift variable $\bar{t} = t + x_{q-1}$:

$$\begin{pmatrix} -\rho_{q-1} I & A \\ A^T & -\rho_{q-1} I \end{pmatrix} \begin{pmatrix} s \\ \bar{t} \end{pmatrix} = \begin{pmatrix} A x_{q-1} \\ 0 \end{pmatrix}$$

Rayleigh quotient iteration (3)

Stable RQI for singular vector (RQI-AMRES)

1: Given singular value and right singular vector estimates ρ_0, x_0
2: **for** $q = 1, 2, \ldots$ **do**
3: Solve for $\bar{t}$ in

$$\begin{pmatrix} -\rho_{q-1} I & A \\ A^T & -\rho_{q-1} I \end{pmatrix} \begin{pmatrix} s \\ \bar{t} \end{pmatrix} = \begin{pmatrix} A x_{q-1} \\ 0 \end{pmatrix}$$

4: $t = \bar{t} - x_{q-1}$
5: $x_q = t / \|t\|$
6: $\rho_q = \|A x_q\|$
7: **end for**

Test Data

Linear Operator: $A = YDZ$

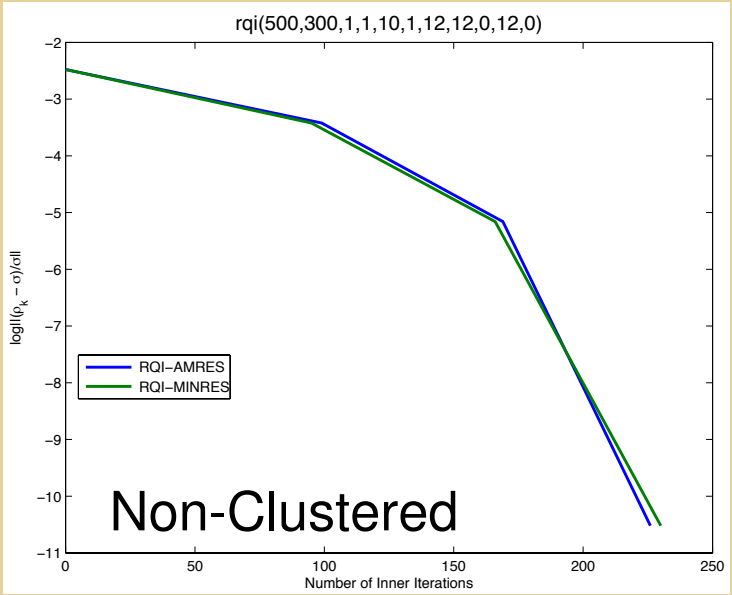
$$Y = I - 2yy^T, \quad Z = I - 2zz^T \qquad \text{(Householder reflectors)}$$

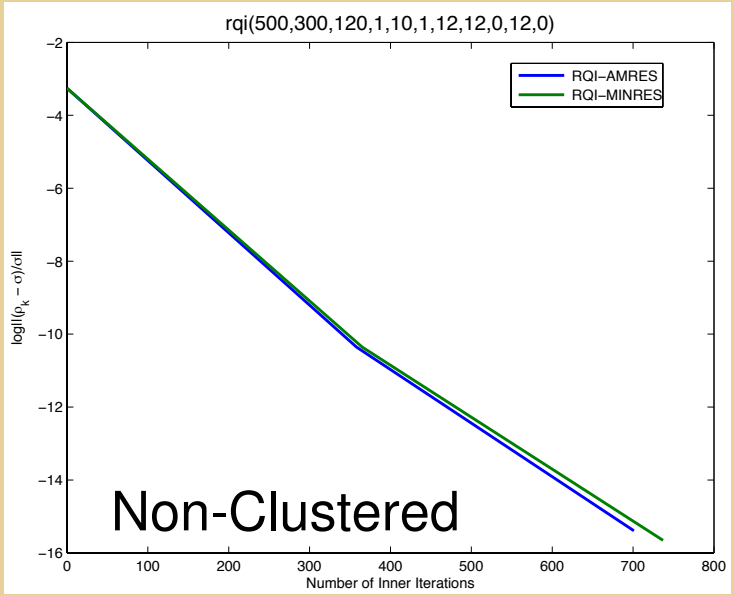
```
y = rand(m,1);    y = y/norm(y);
z = rand(n,1);    z = z/norm(z);
```

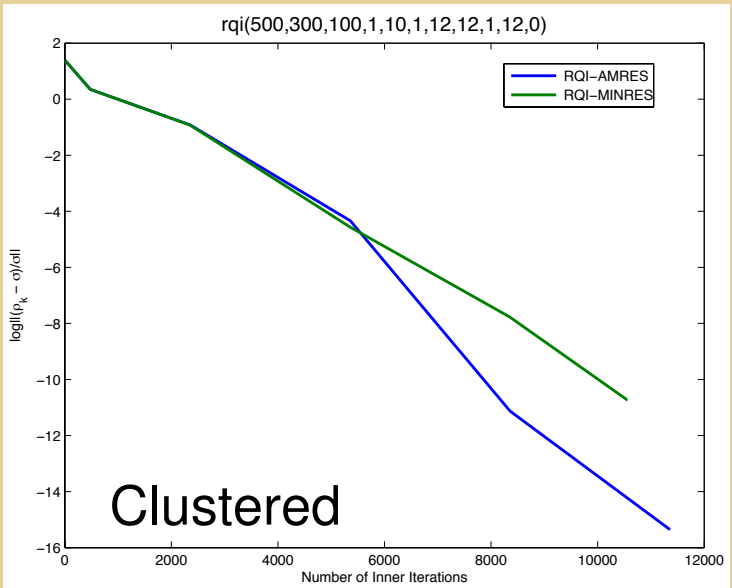
$$D = \begin{pmatrix} d_1 & & \\ & \ddots & \\ & & d_n \end{pmatrix} \qquad \text{(Singular values)}$$

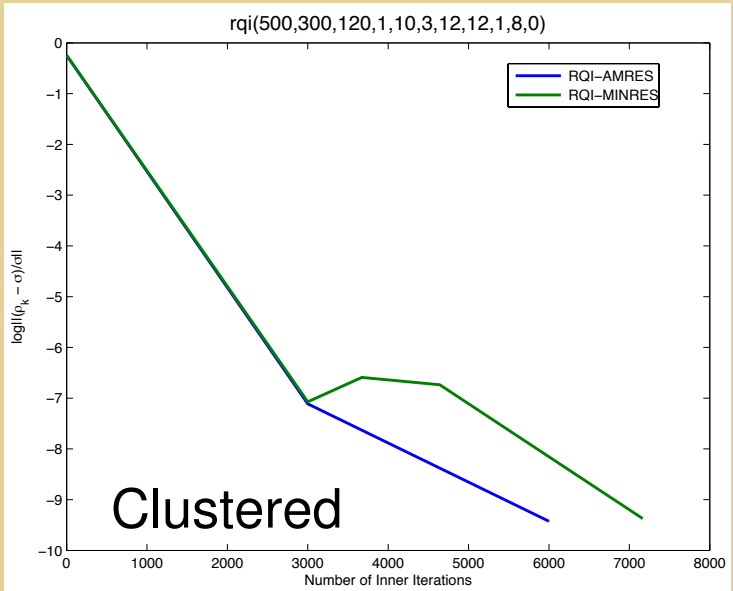
$$\begin{cases} d_j = \sigma_n + (\sigma_1 - \sigma_n)\frac{n-j}{n-1} & \text{(Non-clustered)} \\ d_j = \sigma_n \left(\frac{\sigma_1}{\sigma_n}\right)^{\frac{n-j}{n-1}} & \text{(Clustered)} \end{cases}$$

RQI-MINRES vs RQI-AMRES for singular vectors





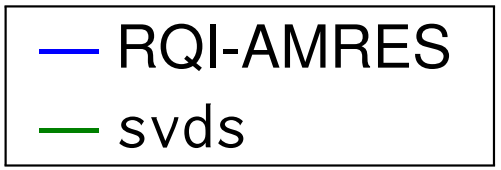


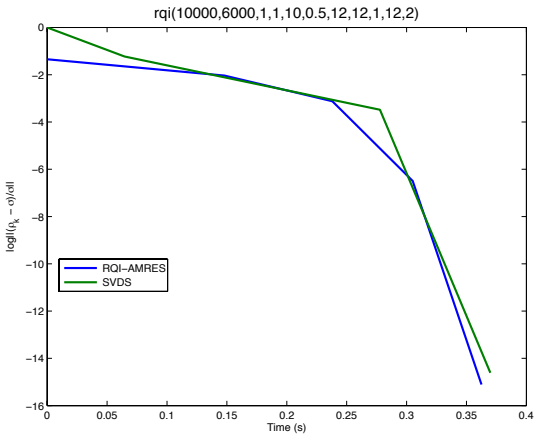


Comparison with svds

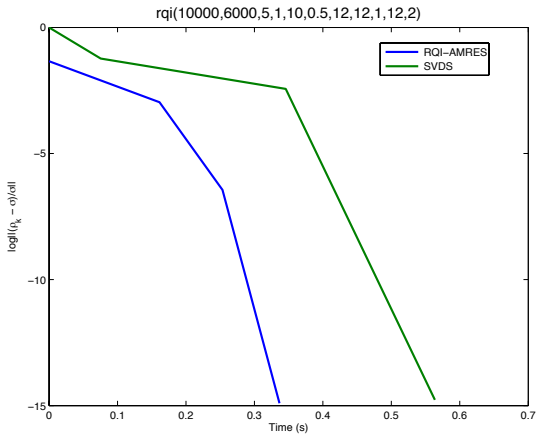
MATLAB svds:

- calls Fortran library ARPACK on $\begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix}$
- For linear operator A, computing σ_k requires computation of $\sigma_1, \ldots, \sigma_k$

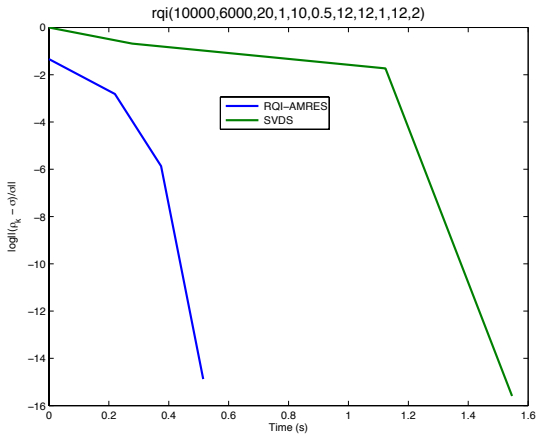




σ_1



σ_5



σ_{20}

Modified RQI

At q-th iteration, RQI-AMRES solves

$$\begin{pmatrix} -\rho_{q-1} I & A \\ A^T & -\rho_{q-1} I \end{pmatrix} \begin{pmatrix} s \\ t \end{pmatrix} = \begin{pmatrix} Ax_{q-1} \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} Ax_0 \\ 0 \end{pmatrix}$$

Question: How to speed up these related solves?

1. Keep right-hand side constant
2. Cached Golub-Kahan bidiagonalization

Cached Golub-Kahan bidiagonalization

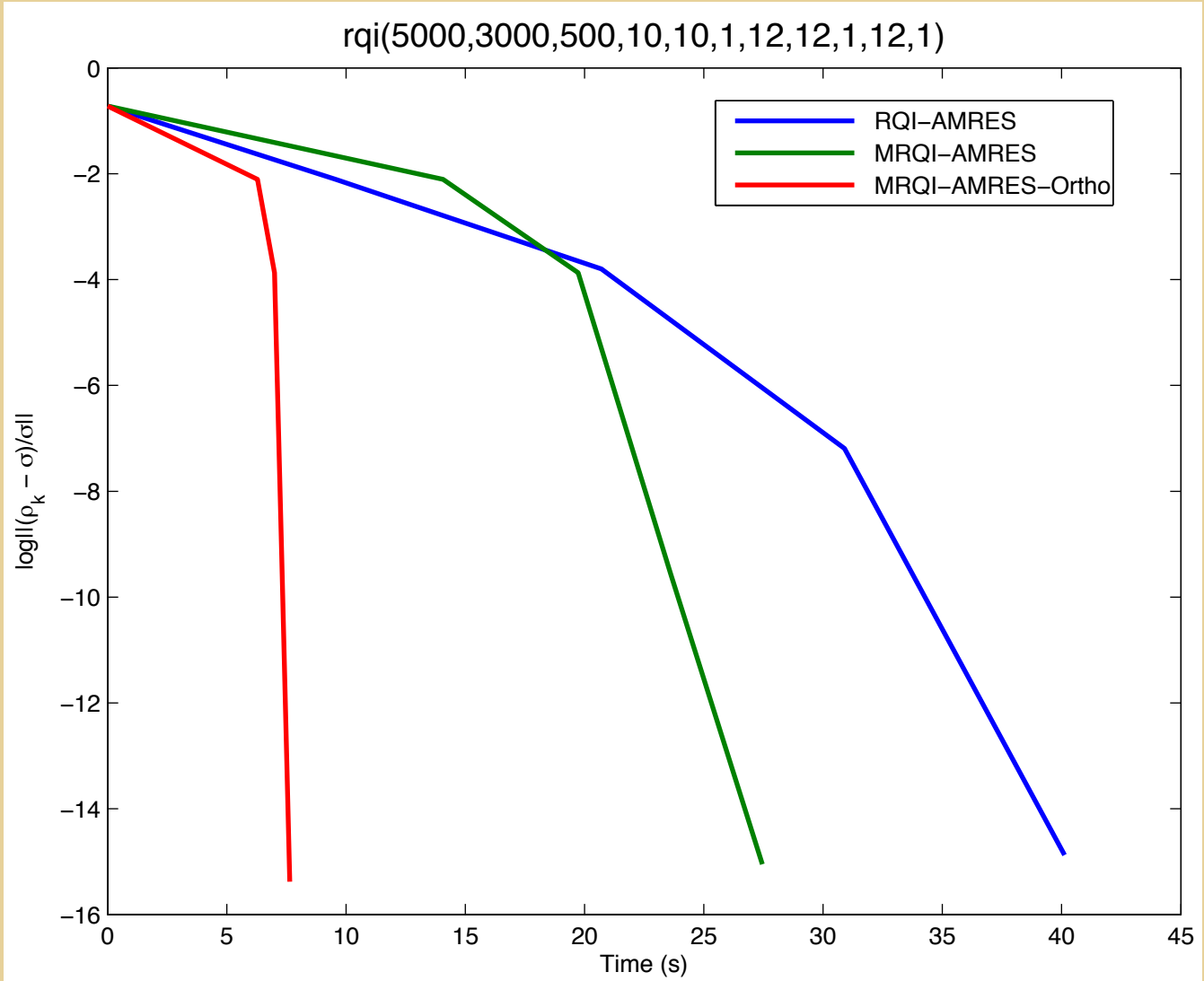
$$\begin{pmatrix} -\rho_{q-1}I & A \\ A^T & -\rho_{q-1}I \end{pmatrix}, \begin{pmatrix} Ax_0 \\ 0 \end{pmatrix} \xrightarrow{\text{Lanczos}} \begin{cases} T_{2k} = \begin{pmatrix} \gamma & \alpha_1 & & & & \\ \alpha_1 & \delta & \beta_2 & & & \\ & \beta_2 & \gamma & \alpha_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \beta_k & \gamma & \alpha_k \\ & & & & \alpha_k & \delta \end{pmatrix} \\ V_{2k} = \begin{pmatrix} u_1 & & u_2 & & \dots & u_k \\ & v_1 & & v_2 & \dots & & v_k \end{pmatrix} \end{cases}$$

In AMRES:

$$\gamma = \delta = -\rho_{q-1} \qquad A, Ax_0 \xrightarrow{\text{Golub-Kahan}} \begin{cases} \alpha_1, \alpha_2, \dots \\ \beta_1, \beta_2, \dots \\ v_1, v_2, \dots \\ u_1, u_2, \dots \end{cases} \quad \text{Cached}$$

Cached $\Rightarrow$ No extra storage cost for reorthogonalization

RQI-AMRES, MRQI-AMRES and MRQI-AMRES-Ortho



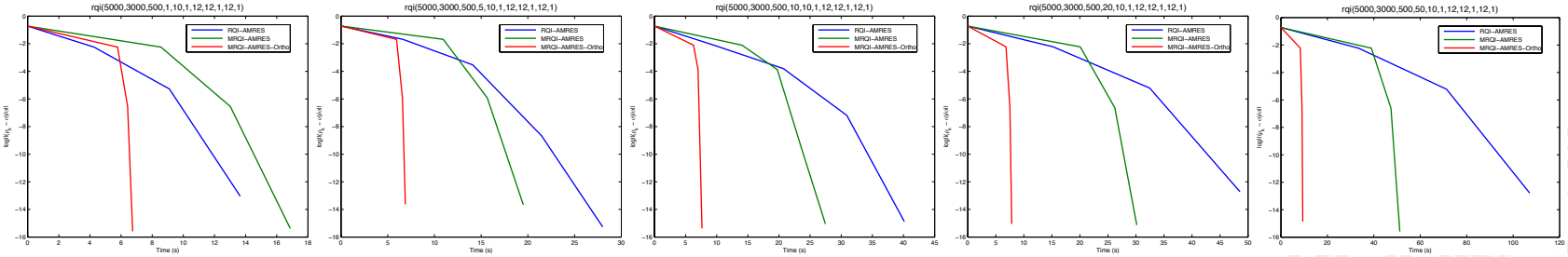
When is MRQI preferable?

Effects of operator cost

Vary the cost $\mathcal{C}$ of linear operator A by

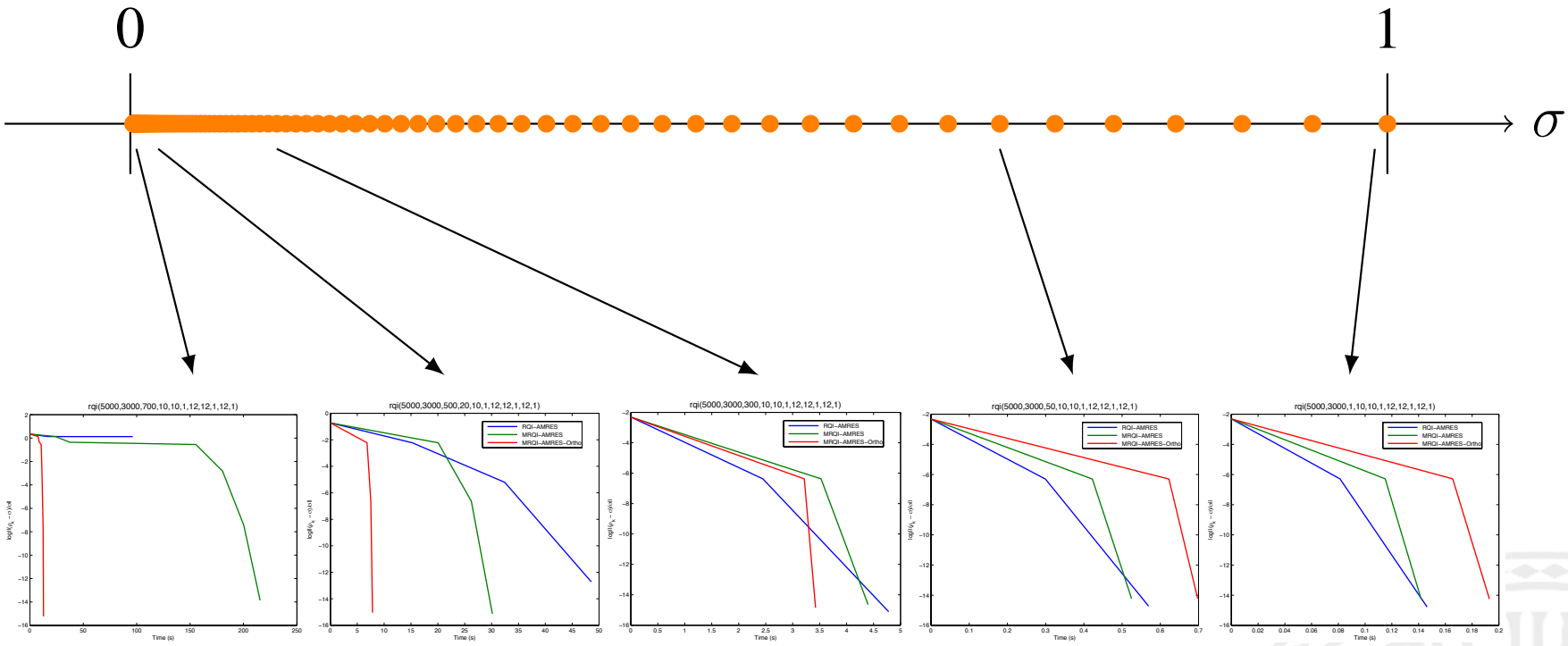
$$A = \left(Y^{(1)} Y^{(2)} \dots Y^{(\mathcal{C})} \right) DZ$$

Increasing cost = Faster MRQI compared to RQI



RQI-AMRES
MRQI-AMRES
MRQI-AMRES-Ortho

Clustering of singular values



Case II

Known Singular Value

Null vector from residual vector

Inverse iteration

To find null vector for singular symmetric A, solve with random b

$$Ax = b$$

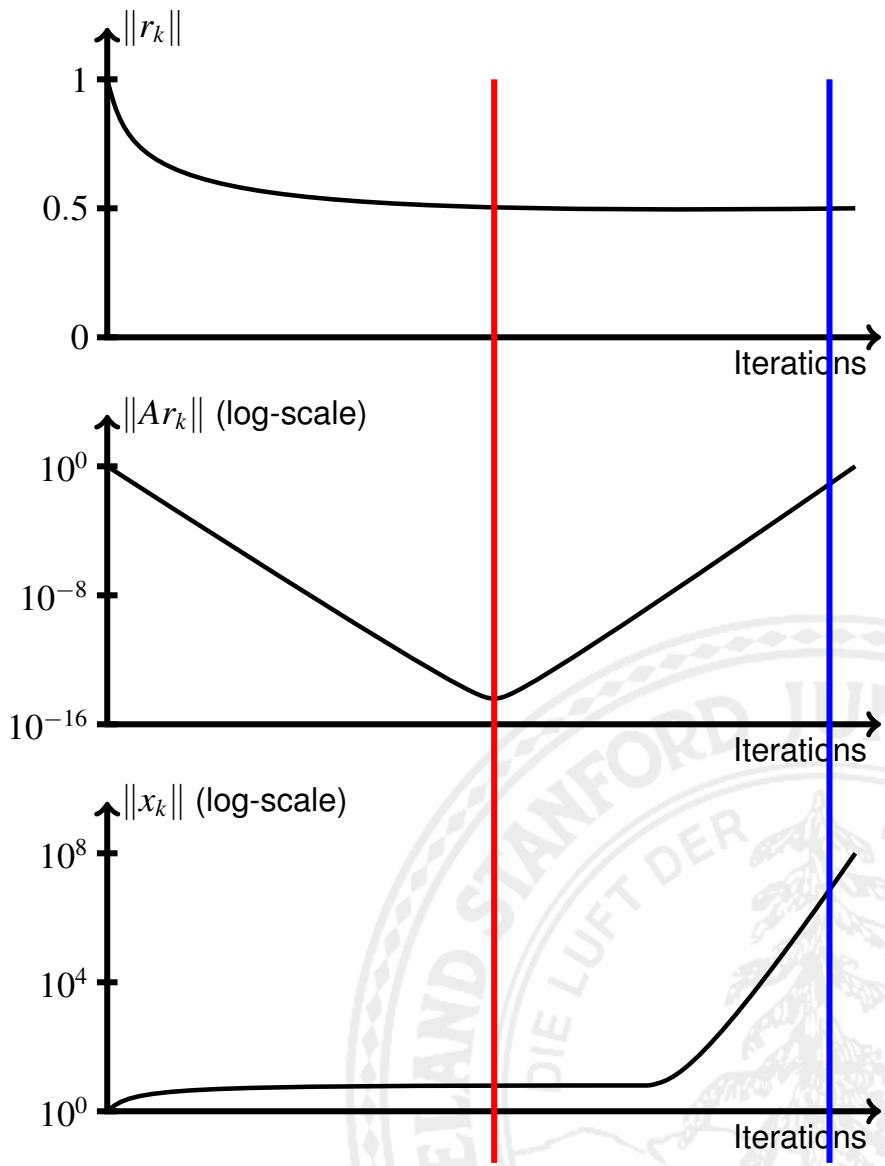
using MINRES until $\|x_k\|$ is large.

Sou-Cheng Choi (2006)

If x solves $\min \|Ax - b\|$, then

$$A^T Ax = A^T b.$$

So $Ar_k = A^T(b - Ax) = 0$.



Singular vector from residual vector

Singular vector from inverse iteration

σ = Singular value, b = Random unit vector

$$\begin{pmatrix} -\sigma I & A \\ A^T & -\sigma I \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}$$

Singular vector from residual vector

$$\min \left\| \begin{pmatrix} b \\ 0 \end{pmatrix} - \begin{pmatrix} -\sigma I & A \\ A^T & -\sigma I \end{pmatrix} \begin{pmatrix} s \\ x \end{pmatrix} \right\|$$

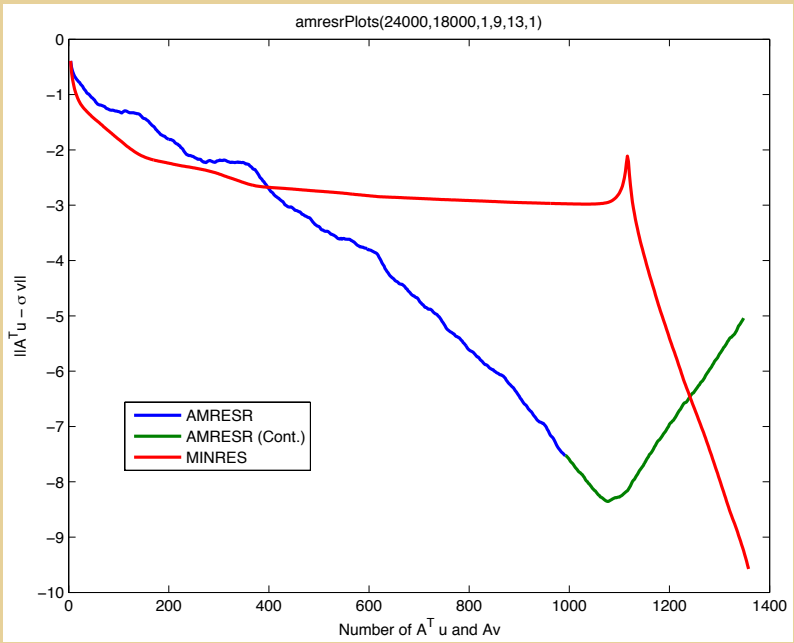
$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix} - \begin{pmatrix} -\sigma I & A \\ A^T & -\sigma I \end{pmatrix} \begin{pmatrix} s \\ x \end{pmatrix}$$

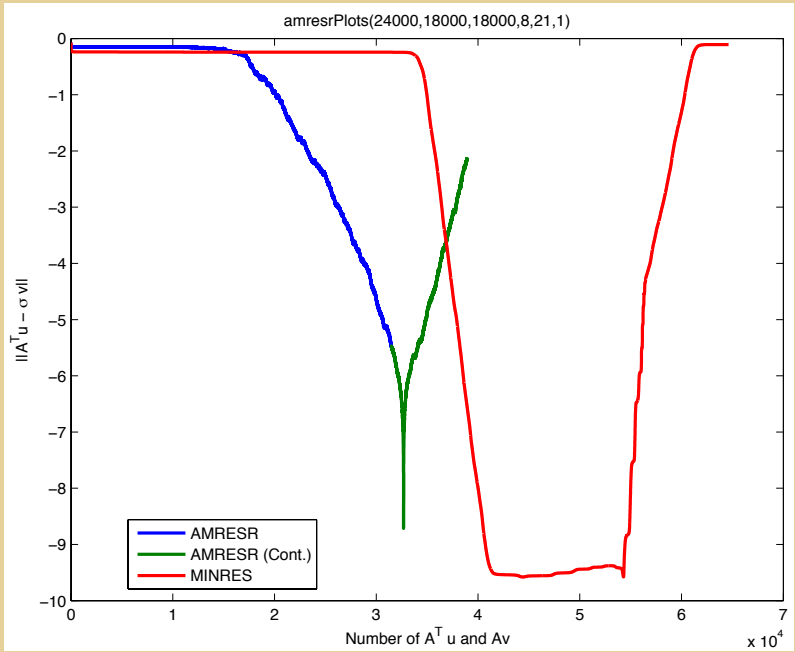
AMRESR

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix} - \begin{pmatrix} -\sigma I & A \\ A^T & -\sigma I \end{pmatrix} \begin{pmatrix} s \\ x \end{pmatrix}$$

- Specialized version of AMRES
- Doesn't compute solution s or x
- Computes v from half of the residual

AMRESR for singular vectors





- AMRESR converges faster
- AMRESR attains a maximum accuracy of $O(\sqrt{\epsilon})$
- Higher accuracy is achievable by inverse iteration

Summary

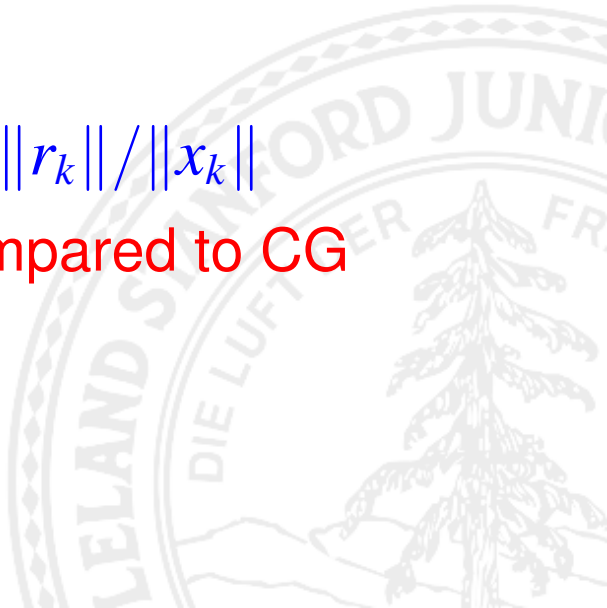
Contributions (1)

Theoretical
Experimental

$$Ax = b \qquad A = A^T$$

MINRES properties when $A \succ 0$

- MINRES gives increasing $\|x_k\|$
- MINRES gives decreasing $\|x^* - x_k\|$
- MINRES gives monotonic backward error $\|r_k\|/\|x_k\|$
- MINRES gives smaller backward error compared to CG



Contributions (2)

solve $Ax = b$ $\qquad$ $\min \|Ax - b\|_2$

$\min \|x\|_2$ s.t. $Ax = b$ $\qquad$ $\min \left\| \begin{pmatrix} A \\ \lambda I \end{pmatrix} x - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2$

Developed algorithm LSMR

Theoretical
Experimental

- $\|x_k\|$ is monotonic
- $\|x_k - x^*\|$ monotonic
- $\|A^T r_k\|$ is monotonic
- $\|r_k\|$ monotonic (nearly as small as for LSQR)
- x_k converges to minimum norm x^* for singular systems
- time-memory tradeoff is available
- $\|E_2\|$ usually monotonic
- $\|E_2\| \approx$ optimal backward error $\Rightarrow$ reliable rule for stopping early

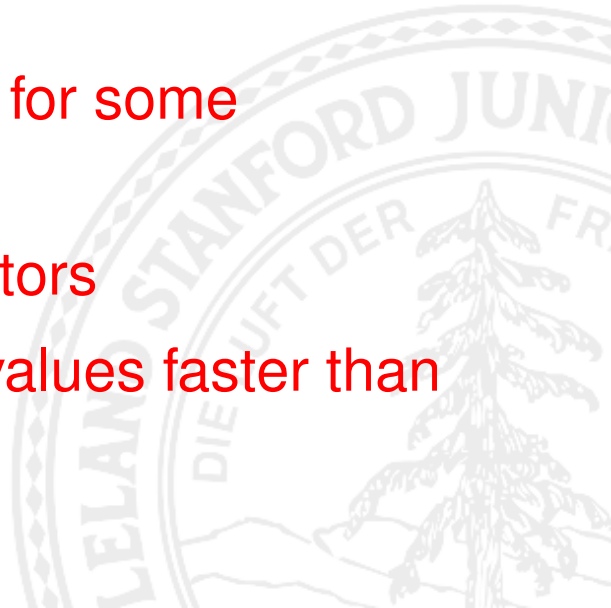
Contributions (3)

Theoretical
Experimental

$$(A^TA - \delta^2 I)x = A^Tb \qquad \text{via} \qquad \begin{pmatrix} \delta I & A \\ A^T & \delta I \end{pmatrix} \begin{pmatrix} s \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}$$

Developed algorithm AMRES

- Solves the negatively-damped least-squares problem
- Half computational cost compared to MINRES on augmented system
- Significantly speed up Curtis-Reid scaling for some matrices
- Stable methods for improving singular vectors
- Computes singular vectors from singular values faster than inverse iteration



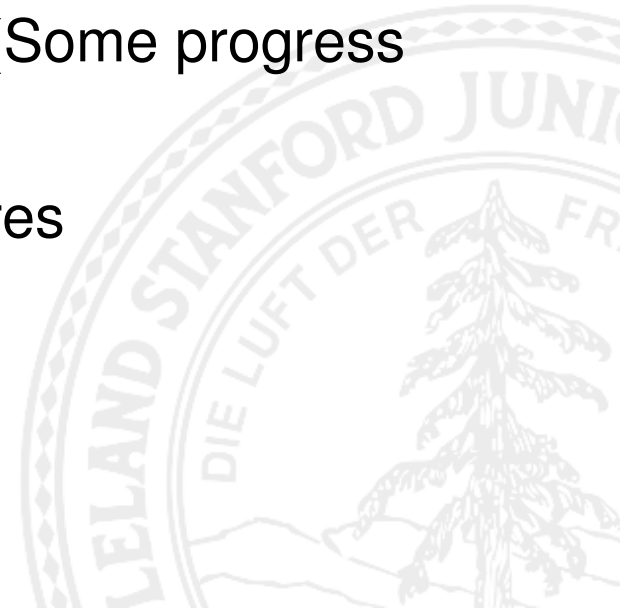
Future directions

Conjecture

- LSMR gives decreasing $\mu(x_k)$

Algorithm design

- Adapt partial reorthogonalization in PROPACK for LSMR and AMRES
- More accurate backward error estimates (Some progress by David Titley-Peloquin)
- SYMMLQ-based algorithm for least-squares



Impact

LSMR Applications

- *A Robust Algorithm for Semidefinite Programming.* Xuan Vinh Doan, Serge Kruk, Henry Wolkowicz
- *Sparse Reconstruction of Images using the Nullspace Method and GRAPPA (SpRING).* Daniel S. Weller, Jonathan R. Polimeni, Leo Grady, Lawrence L. Wald, Elfar Adalsteinsson, and Vivek K Goya
- *Iterative Image Formation using Fast (Re/Back)-projection for Spotlight-mode SAR.* Shaun I. Kelly, Gabriel Rilling, Mike Davies and Bernard Mulgrew
- *A computational approach for large scale nonlinear least squares problems.* Qing Chu, Ying-Wai Fan, James Nagy
- *An l_1 Minimization Algorithm for Sparse Representation Problems in Image Processing.* Carlos Ramirez, Miguel Argaez

Other LSMR citations

- *Algorithm xxx: MINRES-QLP for Singular Symmetric and Hermitian Linear Equations and Least-Squares Problems.* Sou-Cheng T. Choi, Christopher C. Paige, Michael A. Saunders
- *Algorithm xxx: Reliable Calculation of Numerical Rank, Null Space Bases, Pseudoinverse Solutions and Basic Solutions using SuiteSparseQR.* Leslie V. Foster, Timothy A. Davis
- *On the accuracy of the Karlson-Walden's estimate of the backward error for linear least squares problems.* Serge Gratton, Pavel Jiranek, and David Titley-Peloquin

The End