A FAST ALGORITHM FOR LEAST SQUARES PROBLEMS

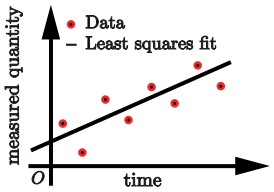
Inner-Outer Iteration Method for Least Squares Problems

Keiichi Morikuni (The Graduate University for Advanced Studies)
Ken Hayami (National Institute of Informatics, The Graduate University for Advanced Studies)
Xiaoke Cui (The University of Tokyo)

WHAT FOR ?

- Survey
- Control
- Signal processing
- Statistical processing
- Image processing
- Optimization
- etc.

e.g.



HOW ?

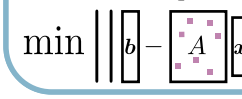
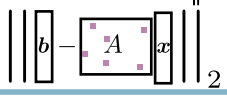
- Development of solvers for solving least squares problems
- Theoretical analysis of the property of the solvers
- Verification through numerical experiments

WHAT DO WE UNDERSTAND ?

Property of the solutions of least squares problems

$$\min_{x \in \mathbb{R}^n} \|b - Ax\|_2$$

where $A \in \mathbb{R}^{m \times n}$ is "large", i.e., find x which minimizes $\|b - Ax\|_2$

$\min$  or $\min$ 

▪ nonzero element

HOW IS IT RELATED TO US ?

For example...

- To determine the 3-D structure of protein molecules
- To determine the inner constitution of the earth from the analysis of seismic waves

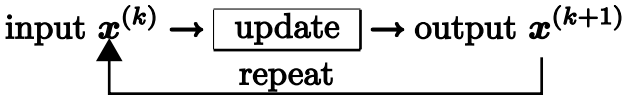
WHY DO WE STUDY ?

- To develop faster, more accurate, and more efficient methods for solving least squares problems
- To solve larger and larger, more and more ill-conditioned problems
- Theoretical understanding of the properties of the solvers

ITERATIVE METHOD

Iterative methods have advantages for large, sparse problems
The process is described as

1. Choose the initial solution x_0
2. For $j = 1, 2, \ldots,$ Do
3. Update x_j with an algorithm
4. EndDo



We should choose or have to develop better algorithms to achieve faster, more robust, and more accurate convergence

Krylov subspace iterative methods for linear systems use

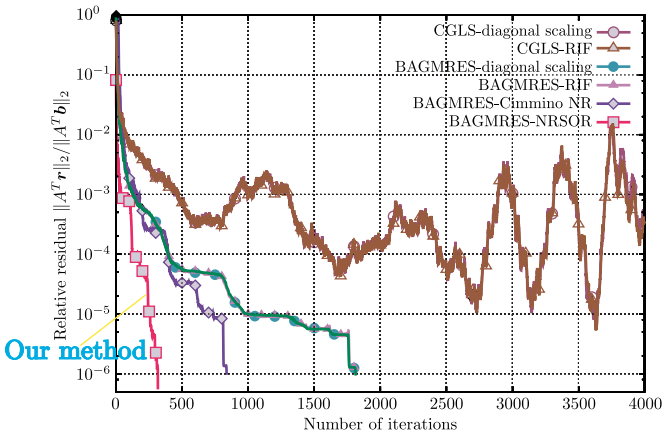
$$\mathcal{K}_m(A, r_0) = \text{span}\{r_0, Ar_0, \ldots, A^{m-1}r_0\} \quad \text{for } A \in \mathbb{R}^{n \times n} \text{ and initial residual } r_0 = b - Ax_0$$

- The generalized minimal residual (GMRES) method minimizes $\|b - Ax\|_2$ over the subspace
  $\text{span}\{v_1, v_2, \ldots, v_k\} = \mathcal{K}_m(A, r_0)$, where $(v_{j+1}, v_i) = 0$ for $i = 1, 2, \ldots, j$ and $(v_{j+1}, v_{j+1}) = 1$
- The conjugate gradient method for least squares problems (CGLS) applies the CG method to $A^T Ax = A^T b$

NUMERICAL EXPERIMENTS

Table 1: Information of the matrices

Problem	m	n	nnz	dens.[%]	rank	$\kappa(A)$
Maragal_6	21,251	10,144	537,694	0.25	8,331	2.91×10^6
Maragal_8	60,845	33,093	1,308,415	0.06	-	-



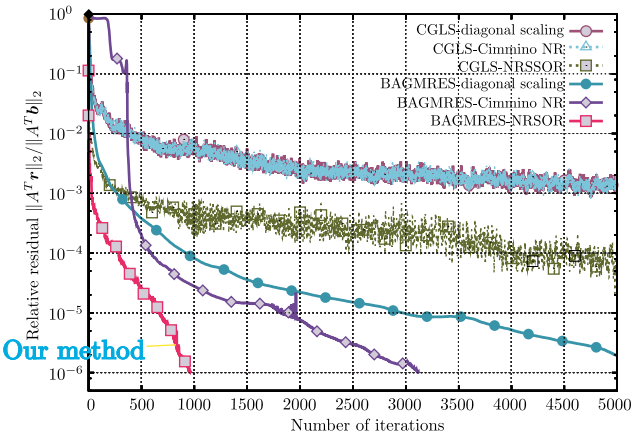


Figure 1: Convergence history of the relative residuals for Maragal_6 and _8

PRECONDITIONING

Iterative methods need good preconditioners to achieve a better convergence and less storage requirement

Preconditioners such as
- Robust Incomplete Factorization by Benzi and Tuma (2003)
- Generalized Approximate Inverse by Cui and Hayami (2009)
- Incomplete Givens Orthogonalization by Yin and Hayami (2009)

for least squares problems make improvements in convergence

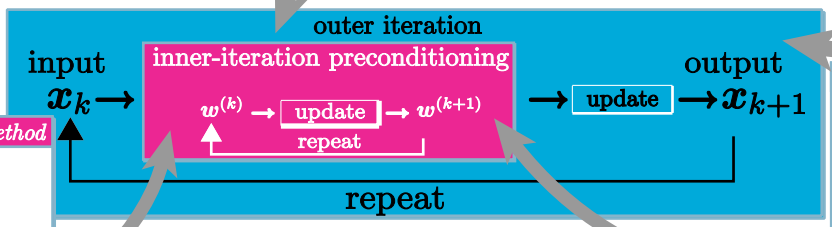
Algorithm 1 *PCGLS method using inner iterations*

1. *Choose* $\boldsymbol{x}_0$ *and compute* $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$
2. *Roughly solve* $A^TA\boldsymbol{z} = A^T\boldsymbol{r}_0$ *to obtain* $\boldsymbol{z} \simeq \boldsymbol{z}_0 = B\boldsymbol{r}_0$ *by using an inner iteration*
3. $\boldsymbol{p}_0 = \boldsymbol{z}_0, \boldsymbol{s}_0 = A^T\boldsymbol{r}_0, \gamma_0 = (\boldsymbol{s}_0, \boldsymbol{z}_0)$
4. *For* $j = 0, 1, 2, \ldots,$ *Do*
5. $\quad \boldsymbol{q}_j = A\boldsymbol{p}_j$
6. $\quad \alpha_j = \gamma_j/(\boldsymbol{q}_j, \boldsymbol{q}_j)$
7. $\quad \boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \alpha_j\boldsymbol{p}_j$
8. $\quad$ *If* $\|A^T(\boldsymbol{b} - A\boldsymbol{x}_{j+1})\|_2/\|A^T\boldsymbol{r}_0\|_2 < \epsilon$ *, then stop*
9. $\quad \boldsymbol{r}_{j+1} = \boldsymbol{r}_j - \alpha_j\boldsymbol{q}_j$
10. $\quad \boldsymbol{s}_{j+1} = A^T\boldsymbol{r}_{j+1}$
11. $\quad$ *Roughly solve* $A^TA\boldsymbol{z} = A^T\boldsymbol{r}_{j+1}$ *to obtain* $\boldsymbol{z} \simeq \tilde{\boldsymbol{z}}_{j+1} = B\boldsymbol{r}_{j+1}$ *by using an inner-iteration*
12. $\quad \gamma_{j+1} = (\boldsymbol{s}_{j+1}, \tilde{\boldsymbol{z}}_{j+1})$
13. $\quad \beta_j = \gamma_{j+1}/\gamma_j$
14. $\quad \boldsymbol{p}_{j+1} = \tilde{\boldsymbol{z}}_{j+1} + \beta_j\boldsymbol{p}_j$
15. *EndDo*

Algorithm 2 *BA-GMRES(k) method using inner iterations*

1. *Choose* $\boldsymbol{x}_0$ *and compute* $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$
2. *Roughly solve* $A^TA\boldsymbol{z} = A^T\boldsymbol{r}_0$ *to obtain* $\boldsymbol{z} \simeq \tilde{\boldsymbol{r}}_0 = B\boldsymbol{r}_0$ *by using an inner iteration*
3. $\beta = \|\tilde{\boldsymbol{r}}_0\|_2, \boldsymbol{v}_1 = \tilde{\boldsymbol{r}}_0/\beta$
4. *For* $j = 1, 2, \ldots, k,$ *Do*
5. $\quad$ *Roughly solve* $A^TA\boldsymbol{z} = A^TA\boldsymbol{v}_j$ *to obtain* $\boldsymbol{z} \simeq \boldsymbol{w}_j = BA\boldsymbol{v}_j$ *by using an inner iteration*
6. $\quad$ *For* $i = 1, 2, \ldots, j,$ *Do*
7. $\quad\quad h_{i,j} = (\boldsymbol{w}_j, \boldsymbol{v}_i)$
8. $\quad\quad \boldsymbol{w}_j = \boldsymbol{w}_j - h_{i,j}\boldsymbol{v}_i$
9. $\quad$ *EndDo*
10. $\quad h_{j+1,j} = \|\boldsymbol{w}_j\|_2$
11. $\quad \boldsymbol{v}_{j+1} = \boldsymbol{w}_j/h_{j+1,j}$
12. $\quad$ *Find* $\boldsymbol{y} \in \mathbb{R}^j$ *that minimizes* $\min_{\boldsymbol{y}\in\mathbb{R}^j} \|\beta\boldsymbol{e}_1 - \bar{H}_j\boldsymbol{y}\|_2 = \min_{\boldsymbol{y}\in\mathbb{R}^j} \|B\boldsymbol{r}_j\|_2$
13. $\quad \boldsymbol{x}_j = \boldsymbol{x}_0 + [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_j]\,\boldsymbol{y}_j$
14. $\quad$ *If* $\|A^T(\boldsymbol{b} - A\boldsymbol{x}_j)\|_2/\|A^T\boldsymbol{r}_0\|_2 < \epsilon$ *, then stop*
15. *EndDo*
16. $\boldsymbol{x}_0 = \boldsymbol{x}_k$ *and go to 2*

Framework of proposed preconditioning techniques: Inner-outer iteration method



Algorithm 3 *Cimmino-NR method*

1. *Choose* $\boldsymbol{x}^{(0)} \in \mathbb{R}^n$
2. $\boldsymbol{r}^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}$
3. *For* $k = 0, 1, \ldots,$ *Do*
4. $\quad$ *For* $i = 1, 2, \ldots, n,$ *Do*
5. $\quad\quad d_i^{(k)} = \lambda\frac{(\boldsymbol{r}^{(k)}, \boldsymbol{a}_i)}{\|\boldsymbol{a}_i\|_2^2}$
6. $\quad$ *EndDo*
7. $\quad \boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \boldsymbol{d}^{(k)}$
8. $\quad \boldsymbol{r}^{(k+1)} = \boldsymbol{r}^{(k)} - A\boldsymbol{d}^{(k)}$
9. *EndDo*

Algorithm 4 *NR-SOR method*

1. *Choose* $\boldsymbol{x}^{(0)} \in \mathbb{R}^n$
2. $\boldsymbol{r} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}$
3. *For* $k = 0,\ 1,\ 2,\ \ldots,$ *Do*
4. $\quad$ *For* $i = 1, 2, \ldots, n,$ *Do*
5. $\quad\quad \delta_i^{(k)} = \omega\frac{\|\boldsymbol{a}_i\|_2^2}{(\boldsymbol{r},\ \boldsymbol{a}_i)}$
6. $\quad\quad x_i^{(k+1)} = x_i^{(k)} + \delta_i^{(k)}$
7. $\quad\quad \boldsymbol{r} = \boldsymbol{r} - \delta_i^{(k)}\boldsymbol{a}_i$
8. $\quad$ *EndDo*
9. *EndDo*

Here, ω is an acceleration parameter

Table 1: Rank deficient cases for $\epsilon = 10^{-6}$

Method	CGLS				*BA*-GMRES			
Prec.	diag.	RIF	Cimm.-NR	NR-SSOR	diag.	RIF	Cimm.-NR	NR-SOR
Maragal_6	11,645	13,287	11,843(1)	2,607(1)	1,817	1,814	838(5)	318(6)
	62.44	58.03(10^2)	76.90(0.3)	31.38(1.1)	79.28	101.38(10^1)	27.62(0.4)	$*$ 10.12(1.5)
Maragal_8	732,583	630,204	726,187(1)	41,516(1)	$> 5,000$	$> 5,000$	3,122(6)	969(6)
	12117.87	12251.78(10^3)	14627.75(1.0)	1416.05(1.0)	1812.42	2333.23(10^3)	815.69(0.2)	$*$ 125.55(1.2)

first row: number of outer iterations (number of inner iterations)
second row: computation time [seconds] (best parameter)

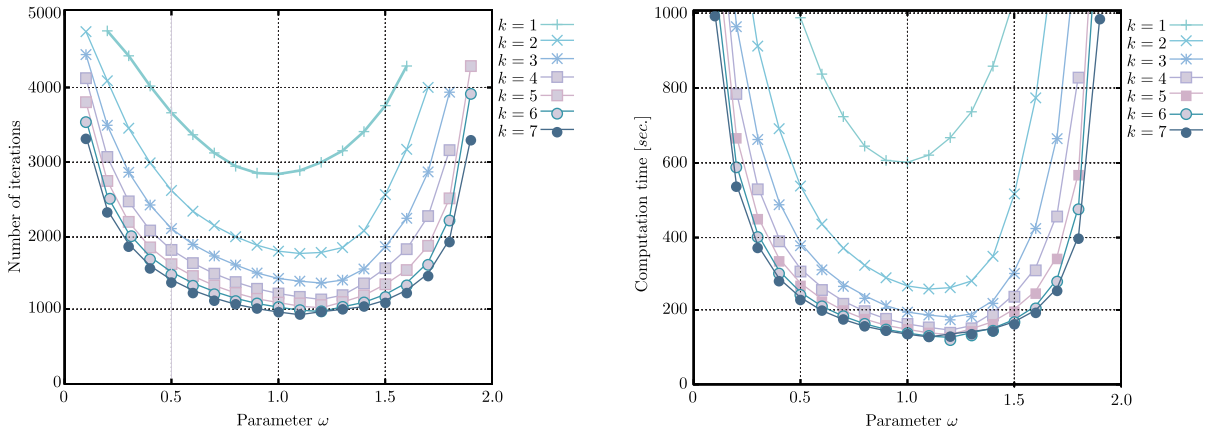


Figure 2: Number of iterations and computation time for Maragal_8 with various parameters

WHAT WILL THIS RESEARCH ACCOMPLISH IN THE NEXT 10 YEARS ?

- The property of the solvers for least squares problems will be sufficiently revealed.
- They will be widely used with nothing to worry about breakdown and slow convergence.
- Massively parallel computers will employ them to solve dramatically large problems.
- With the growing need for solving difficult and complicated problems, our solvers will help science, engineering, and industry.