Two-subspace Randomized Extended Kaczmarz Method for Linear Least-squares Problems

Wen-Ting Wu

School of Mathematics and Statistics
Beijing Institute of Technology

Email: wuwenting@bit.edu.cn

GGMW2022
April 15-19, 2022

Outline

1. Randomized Kaczmarz
2. Two-subspace Randomized Extended Kaczmarz
3. Numerical Results
4. Conclusions and Remarks

Outline

1. Randomized Kaczmarz
2. Two-subspace Randomized Extended Kaczmarz
3. Numerical Results
4. Conclusions and Remarks

For the large-scale system of linear equations

$$Ax = b, \quad \textbf{with} \quad A \in \mathbb{C}^{m\times n} \quad \textbf{and} \quad b \in \mathbb{C}^{m},$$

consider its least-norm least-squares solution $x_\star = A^\dagger b$.

$$Ax = b \quad \Longrightarrow \quad \begin{bmatrix} A^{(1)} \\ A^{(2)} \\ \vdots \\ A^{(m)} \end{bmatrix} x = \begin{bmatrix} b^{(1)} \\ b^{(2)} \\ \vdots \\ b^{(m)} \end{bmatrix}$$

where

- $A^{(i)}(i = 1, 2, \ldots, m)$**: the** i**-th row of the matrix** A**,**
- $b^{(i)}(i = 1, 2, \ldots, m)$**: the** i**-th entry of the vector** b**.**

At the k-th iterate,

$$x_k \longrightarrow x_{k+1} \in \{x \mid A^{(i_k)}x = b^{(i_k)}\}$$

that is

$$x_{k+1} = x_k + \frac{\left(b^{(i_k)} - A^{(i_k)}x_k\right)}{\|A^{(i_k)}\|_2^2}(A^{(i_k)})^*, \quad i_k \in \{1, 2, \ldots, m\}$$

- **Kaczmarz Method** [1]**:**
  Select i_k according to $i_k = (k \mod m) + 1$
- **Randomized Kaczmarz Method** [2]**:**
  Select i_k with the probability $\Pr(\text{row} = i_k) = \frac{\|A^{(i_k)}\|_2^2}{\|A\|_F^2}$

[1] Kaczmarz, Bull. Int. Acad. Polon. Sci. Lett. A(1937).

[2] Strohmer and Vershynin, J. Fourier Anal. Appl.(2009).

Convergence Theorem [3]

Let the linear system $Ax = b$ be consistent. Then, starting from an initial guess $x_0 \in \mathbb{C}^n$ in the column space of A^*, the iteration sequence $\{x_k\}_{k=0}^{\infty}$ generated by the RK method obeys

$$\mathbb{E}\|x_k - x_\star\|_2^2 \leq \left(1 - \frac{\lambda_{\min}(A^*A)}{\|A\|_F^2}\right)^k \|x_0 - x_\star\|_2^2,$$

where $\lambda_{\min}(A^*A)$ represents the smallest nonzero eigenvalue of A^*A.

[3] Gower and Richtárik, arXiv:1512.06890(2015).

Further works

Applied to different methods

- D. Leventhal and A.S. Lewis, Randomized methods for linear constraints: convergence rates and conditioning, Math. Oper. Res., 35(2010), 641-654.

Extended to different problems

- A. Zouzias and N.M. Freris, Randomized extended Kaczmarz for solving least squares, SIAM J. Matrix Anal. Appl., 34(2013), 773-793.
- A. Hefny, D. Needell and A. Ramdas, Rows versus columns: Randomized Kaczmarz or Gauss-Seidel for ridge regression, SIAM J. Sci. Comput., 39(2017), S528-S542.

Accelerated with greedy strategy

- Z.-Z. Bai and W.-T. Wu, On greedy randomized Kaczmarz method for solving large sparse linear systems, SIAM J. Sci. Comput., 40(2018), A592-A606.
- Z.-Z. Bai and W.-T. Wu, On greedy randomized augmented Kaczmarz method for solving large sparse inconsistent linear systems, SIAM J. Sci. Comput., 43(2021), A3892-A3911.

• • •

Block Kaczmarz methods

$$Ax = b \quad \Longrightarrow \quad \begin{bmatrix} A_{\tau_1} \\ A_{\tau_2} \\ \vdots \\ A_{\tau_p} \end{bmatrix} x = \begin{bmatrix} b_{\tau_1} \\ b_{\tau_2} \\ \vdots \\ b_{\tau_p} \end{bmatrix}$$

where

- $\tau_i (i = 1, 2, \ldots, p)$: **a subset of** $\{1, 2, \ldots, m\}$,
- $A_{\tau_i} (i = 1, 2, \ldots, p)$: **the row submatrix of** A **indexed by** τ_i,
- $b_{\tau_i} (i = 1, 2, \ldots, p)$: **the subvector of** b **with components listed in** τ_i.

Block Kaczmarz methods (cont'd)

At the k-th iterate, randomly choose a block $A_{\tau_{i_k}}$ to update x_k.

- **Randomized Block Kaczmarz Method** [4]:
  $x_k \longrightarrow x_{k+1} \in \{x \,|\, A_{\tau_{i_k}} x = b_{\tau_{i_k}}\}$,
  that is $x_{k+1} = x_k + (A_{\tau_{i_k}})^\dagger (b_{\tau_{i_k}} - A_{\tau_{i_k}} x_k)$

- **Randomized Average Block Kaczmarz Method** [5]:
  $$x_{k+1} = x_k + \alpha_k \left(\sum_{i \in \tau_{i_k}} \omega_i^k \frac{b^{(i_k)} - A^{(i_k)} x_k}{\|A^{(i_k)}\|_2^2} (A^{(i_k)})^* \right)$$

[4] Needell and Tropp, Linear Algebra Appl.(2014).

[5] I. Necoara, SIAM J. Matrix Anal. Appl.(2019).

Two-subspace Randomized Kaczmarz [6]

At the k-th iterate, randomly choose two different rows and update

$$x_k \longrightarrow x_{k+1} \in \{x \,|\, A_{\tau_{i_k}} x = b_{\tau_{i_k}}\},$$

where

$$A_{\tau_{i_k}} = \begin{pmatrix} A^{(i_{k_1})} \\ A^{(i_{k_2})} \end{pmatrix} \quad \textbf{and} \quad b_{\tau_{i_k}} = \begin{pmatrix} b^{(i_{k_1})} \\ b^{(i_{k_2})} \end{pmatrix}.$$

[6] Needell and Ward, J. Fourier Anal. Appl.(2013).

Generalized Two-subspace Randomized Kaczmarz Method

1. **Select** $i_{k_1} \in \{1, 2, \ldots, m\}$ **with probability**
   $$\Pr(\text{row} = i_{k_1}) = \frac{\|A^{(i_{k_1})}\|_2^2}{\|A\|_F^2}$$
2. **Select** $i_{k_2} \in \{1, 2, \ldots, m\} \setminus \{i_{k_1}\}$ **with probability**
   $$\Pr(\text{row} = i_{k_2}) = \frac{\|A^{(i_{k_2})}\|_2^2}{\|A\|_F^2 - \|A^{(i_{k_1})}\|_2^2}$$
3. **Set**
   $$\mu_k = \frac{A^{(i_{k_2})}(A^{(i_{k_1})})^*}{\|A^{(i_{k_2})}\|_2 \|A^{(i_{k_1})}\|_2}, \quad \tilde{r}_{k_1} = \frac{b^{(i_{k_1})} - A^{(i_{k_1})} x_k}{\|A^{(i_{k_1})}\|_2}, \quad \tilde{r}_{k_2} = \frac{b^{(i_{k_2})} - A^{(i_{k_2})} x_k}{\|A^{(i_{k_2})}\|_2}$$
4. **Set**
   $$x_{k+1} = x_k + \frac{\tilde{r}_{k_1} - \bar{\mu}_k \tilde{r}_{k_2}}{(1 - |\mu_k|^2)\|A^{(i_{k_1})}\|_2} (A^{(i_{k_1})})^* + \frac{\tilde{r}_{k_2} - \mu_k \tilde{r}_{k_1}}{(1 - |\mu_k|^2)\|A^{(i_{k_2})}\|_2} (A^{(i_{k_2})})^*$$

Generalized TRK Method (cont'd)

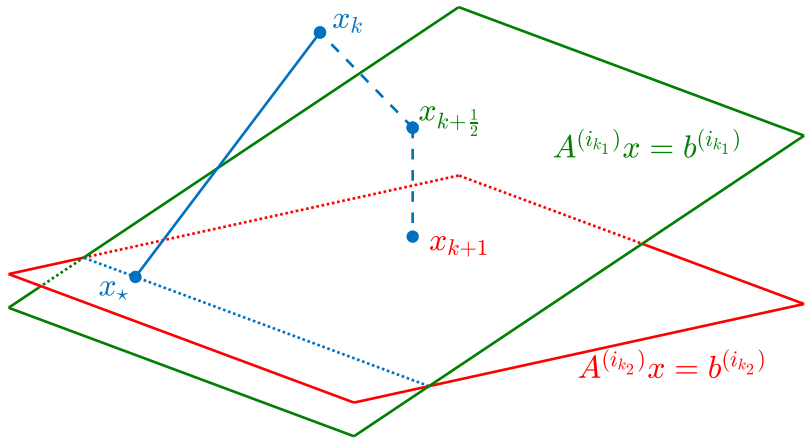
$$\begin{cases} x_{k+\frac{1}{2}} = x_k + \dfrac{\left(b^{(i_{k_2})} - A^{(i_{k_2})} x_k\right)}{\|A^{(i_{k_2})}\|_2^2} \left(A^{(i_{k_2})}\right)^* \\ x_{k+1} = x_{k+\frac{1}{2}} + \dfrac{\left(\beta_k - u_k^* x_{k+\frac{1}{2}}\right)}{\|u_k\|_2^2} u_k, \end{cases}$$

where

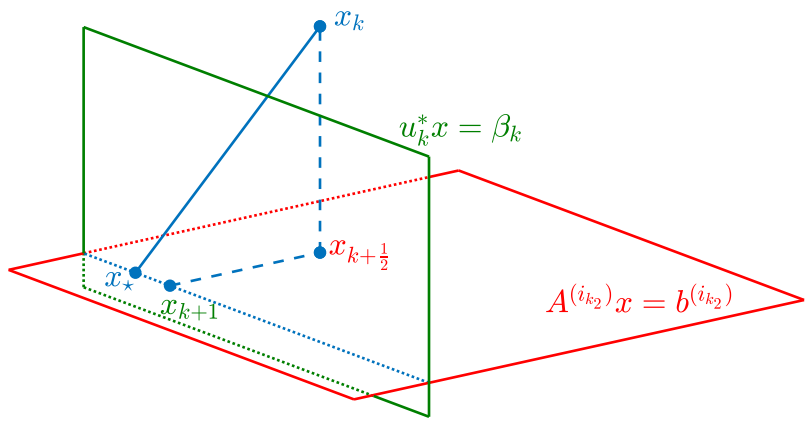
$$\mu_k = \frac{A^{(i_{k_2})} (A^{(i_{k_1})})^*}{\|A^{(i_{k_2})}\|_2 \|A^{(i_{k_1})}\|_2},$$

$$u_k = \frac{(A^{(i_{k_1})})^*}{\|A^{(i_{k_1})}\|_2} - \mu_k \frac{(A^{(i_{k_2})})^*}{\|A^{(i_{k_2})}\|_2} \text{ and } \beta_k = \frac{b^{(i_{k_1})}}{\|A^{(i_{k_1})}\|_2} - \bar{\mu}_k \frac{b^{(i_{k_2})}}{\|A^{(i_{k_2})}\|_2}.$$

RK:


x_k
$x_{k+\frac{1}{2}}$
$A^{(i_{k_1})}x = b^{(i_{k_1})}$
x_{k+1}
$x_\star$
$A^{(i_{k_2})}x = b^{(i_{k_2})}$

GTRK:


x_k
$u_k^* x = \beta_k$
$x_{k+\frac{1}{2}}$
$x_\star$
x_{k+1}
$A^{(i_{k_2})}x = b^{(i_{k_2})}$

Convergence Theorem [7]

If the linear system $Ax = b$ is consistent, then the iteration sequence $\{x_k\}_{k=0}^{\infty}$ generated by the GTRK method starting from an initial guess $x_0 \in \mathbb{C}^n$ in the column space of A^* obeys

$$\mathbb{E}\|x_k - x_\star\|_2^2$$

$$\leq \left[\left(1 - \frac{\lambda_{\min}(A^*A)}{\tau_{\max}}\right)\left(1 - \frac{\lambda_{\min}(A^*A)}{\|A\|_F^2}\right) - \frac{\lambda_{\min}(A^*A)}{\|A\|_F^2}\frac{\tau_{\min}}{\tau_{\max}}\gamma\right]^k \|x_0 - x_\star\|_2^2.$$

where $\gamma = \min\left\{\frac{\delta^2(1-\delta)}{1+\delta}, \frac{\Delta^2(1-\Delta)}{1+\Delta}\right\}$ **with**

$$\delta = \min_{p\neq q} \frac{|A^{(p)}(A^{(q)})^*|}{\|A^{(p)}\|_2\|A^{(q)}\|_2} \quad \textbf{and} \quad \Delta = \max_{p\neq q} \frac{|A^{(p)}(A^{(q)})^*|}{\|A^{(p)}\|_2\|A^{(q)}\|_2},$$

and

$$\tau_{\min} = \|A\|_F^2 - \max_{1\leq i\leq m} \|A^{(i)}\|_2^2, \quad \tau_{\max} = \|A\|_F^2 - \min_{1\leq i\leq m} \|A^{(i)}\|_2^2.$$

[7] Wu, Numer. Algorithms(2022).

Outline

1 Randomized Kaczmarz

2 **Two-subspace Randomized Extended Kaczmarz**

3 Numerical Results

4 Conclusions and Remarks

Randomized Extended Kaczmarz Method [8]

1. **Select** $i_k \in \{1, 2, \ldots, m\}$ **with the probability**
   $\Pr(\text{row} = i_k) = \frac{\|A^{(i_k)}\|_2^2}{\|A\|_F^2}$
2. **Set** $x_{k+1} = x_k + \frac{\left(b^{(i_k)} - z_k^{(i_k)} - A^{(i_k)} x_k\right)}{\|A^{(i_k)}\|_2^2} (A^{(i_k)})^*$
3. **Select** $j_k \in \{1, 2, \ldots, n\}$ **with the probability**
   $\Pr(\text{column} = j_k) = \frac{\|A_{(j_k)}\|_2^2}{\|A\|_F^2}$
4. **Set** $z_{k+1} = z_k - \frac{(A_{(j_k)})^* z_k}{\|A_{(j_k)}\|_2^2} A_{(j_k)}$

where

- $A_{(j_k)}$**: the** j_k**-th column of the matrix** A**,**
- $z_k^{(i_k)}$**: the** i_k**-th entry of the vector** z_k**,**
- $k = 0, 1, 2, \ldots$.

[8] Zouzias and Freris, SIMAX(2013).

Convergence Theorem [8]

Starting from the initial vectors $x_0 = 0$ and $z_0 = b$, the iteration sequence $\{x_k\}_{k=0}^{\infty}$ generated by the REK method obeys

$$\mathbb{E}\|x_k - x_\star\|_2^2 \leq \left(1 - \frac{\lambda_{\min}(A^*A)}{\|A\|_F^2}\right)^{\lfloor \frac{k}{2} \rfloor} \left(1 + 2\kappa^2(A)\right) \|x_\star\|_2^2,$$

where

- $\kappa(A) = \sqrt{\frac{\lambda_{\max}(A^*A)}{\lambda_{\min}(A^*A)}}$,
- $\lambda_{\min}(A^*A)$ **and** $\lambda_{\max}(A^*A)$ **represent the smallest nonzero and the largest eigenvalue of the matrix** A^*A**, respectively,**
- $\lfloor \cdot \rfloor$ **represents the largest integer which is smaller than or equal to the corresponding constant.**

[8] Zouzias and Freris, SIMAX(2013).

Select two distinct rows $A^{(i_{k_1})}$ and $A^{(i_{k_2})}$ randomly,

$$\begin{cases} x_{k+\frac{1}{2}} = x_k + \dfrac{\left(b^{(i_{k_2})} - z_k^{(i_{k_2})} - A^{(i_{k_2})} x_k\right)}{\|A^{(i_{k_2})}\|_2^2} \left(A^{(i_{k_2})}\right)^* \\ x_{k+1} = x_{k+\frac{1}{2}} + \dfrac{\left(\beta_k - u_k^* x_{k+\frac{1}{2}}\right)}{\|u_k\|_2^2} u_k, \end{cases}$$

where $\mu_k = \dfrac{A^{(i_{k_2})}(A^{(i_{k_1})})^*}{\|A^{(i_{k_2})}\|_2 \|A^{(i_{k_1})}\|_2}$**,**

$u_k = \dfrac{(A^{(i_{k_1})})^*}{\|A^{(i_{k_1})}\|_2} - \mu_k \dfrac{(A^{(i_{k_2})})^*}{\|A^{(i_{k_2})}\|_2}$ **and** $\beta_k = \dfrac{b^{(i_{k_1})} - z_k^{(i_{k_1})}}{\|A^{(i_{k_1})}\|_2} - \bar{\mu}_k \dfrac{\left(b^{(i_{k_2})} - z_k^{(i_{k_2})}\right)}{\|A^{(i_{k_2})}\|_2}.$

Then $x_{k+1} \in \{x \,|\, A_{\tau_{i_k}} x = b_{\tau_{i_k}} - (z_k)_{\tau_{i_k}}\}$**, where**

$$A_{\tau_{i_k}} = \begin{pmatrix} A^{(i_{k_1})} \\ A^{(i_{k_2})} \end{pmatrix}, \quad b_{\tau_{i_k}} = \begin{pmatrix} b^{(i_{k_1})} \\ b^{(i_{k_2})} \end{pmatrix}, \quad \textbf{and} \quad (z_k)_{\tau_{i_k}} = \begin{pmatrix} z_k^{(i_{k_1})} \\ z_k^{(i_{k_2})} \end{pmatrix}.$$

Select two distinct columns $A_{(j_{k_1})}$ and $A_{(j_{k_2})}$ randomly,

$$\begin{cases} z_{k+\frac{1}{2}} = z_k - \dfrac{A_{(j_{k_2})}^* z_k}{\|A_{(j_{k_2})}\|_2^2} A_{(j_{k_2})} \\ z_{k+1} = z_{k+\frac{1}{2}} - \dfrac{v_k^* z_{k+\frac{1}{2}}}{\|v_k\|_2^2} v_k, \end{cases}$$

where

$$\nu_k = \frac{A_{(j_{k_2})}^* A_{(j_{k_1})}}{\|A_{(j_{k_2})}\|_2 \|A_{(j_{k_1})}\|_2} \quad \textbf{and} \quad v_k = \frac{A_{(j_{k_1})}}{\|A_{(j_{k_1})}\|_2} - \nu_k \frac{A_{(j_{k_2})}}{\|A_{(j_{k_2})}\|_2}.$$

Then $z_{k+1} \in \{z \mid A_{\omega_{i_k}}^* z = 0\}$, where $A_{\omega_{i_k}} = (A_{(j_{k_1})}, A_{(j_{k_2})})$.

Two-subspace Randomized Extended Kaczmarz Method [7]

1. **Select** $i_{k_1} \in \{1, 2, \ldots, m\}$ **with probability**
   $$\Pr(\text{row} = i_{k_1}) = \frac{\|A^{(i_{k_1})}\|_2^2}{\|A\|_F^2}$$
2. **Select** $i_{k_2} \in \{1, 2, \ldots, m\} \setminus \{i_{k_1}\}$ **with probability**
   $$\Pr(\text{row} = i_{k_2}) = \frac{\|A^{(i_{k_2})}\|_2^2}{\|A\|_F^2 - \|A^{(i_{k_1})}\|_2^2}$$
3. **Set**
   $$\mu_k = \frac{A^{(i_{k_2})}(A^{(i_{k_1})})^*}{\|A^{(i_{k_2})}\|_2 \|A^{(i_{k_1})}\|_2},$$
   $$r_{k_1} = \frac{b^{(i_{k_1})} - z_k^{(i_{k_1})} - A^{(i_{k_1})} x_k}{\|A^{(i_{k_1})}\|_2}, \; r_{k_2} = \frac{b^{(i_{k_2})} - z_k^{(i_{k_2})} - A^{(i_{k_2})} x_k}{\|A^{(i_{k_2})}\|_2}$$
4. **Set**
   $$x_{k+1} = x_k + \frac{r_{k_1} - \bar{\mu}_k r_{k_2}}{(1 - |\mu_k|^2)\|A^{(i_{k_1})}\|_2} (A^{(i_{k_1})})^* + \frac{r_{k_2} - \mu_k r_{k_1}}{(1 - |\mu_k|^2)\|A^{(i_{k_2})}\|_2} (A^{(i_{k_2})})^*$$

Two-subspace Randomized Extended Kaczmarz Method [7] (Cont'd.)

5. **Select** $j_{k_1} \in \{1, 2, \ldots, n\}$ **with probability**
$$\Pr(\text{column} = j_{k_1}) = \frac{\|A_{(j_{k_1})}\|_2^2}{\|A\|_F^2}$$
6. **Select** $j_{k_2} \in \{1, 2, \ldots, n\} \setminus \{j_{k_1}\}$ **with probability**
$$\Pr(\text{column} = j_{k_2}) = \frac{\|A_{(j_{k_2})}\|_2^2}{\|A\|_F^2 - \|A_{(j_{k_1})}\|_2^2}$$
7. **Set**
$$\nu_k = \frac{A_{(j_{k_2})}^* A_{(j_{k_1})}}{\|A_{(j_{k_2})}\|_2 \|A_{(j_{k_1})}\|_2}, \; s_{k_1} = \frac{A_{(j_{k_1})}^* z_k}{\|A_{(j_{k_1})}\|_2}, \; s_{k_2} = \frac{A_{(j_{k_2})}^* z_k}{\|A_{(j_{k_2})}\|_2}$$
8. **Set**
$$z_{k+1} = z_k - \frac{s_{k_1} - \bar{\nu}_k s_{k_2}}{(1 - |\nu_k|^2)\|A_{(j_{k_1})}\|_2} A_{(j_{k_1})} - \frac{s_{k_2} - \nu_k s_{k_1}}{(1 - |\nu_k|^2)\|A_{(j_{k_2})}\|_2} A_{(j_{k_2})}$$

[7] Wu, Numer. Algorithms(2022).

Convergence Theorem [7]

Starting from the initial vectors $x_0 = 0$ and $z_0 = b$, the iteration sequence $\{x_k\}_{k=0}^{\infty}$ generated by the TREK method obeys

$$\begin{aligned}&\mathbb{E}\|x_k - x_\star\|_2^2\\ &\leq (\max\{\alpha,\beta\})^{\lfloor \frac{k}{2} \rfloor}\left(1 + \frac{4}{(1-\Delta)(1-\alpha)}\frac{\tau_{\max}}{\tau_{\min}}\frac{\lambda_{\max}(A^*A)}{\|A\|_F^2}\right)\|x_\star\|_2^2,\end{aligned}$$

where

$$\alpha = \left(1 - \frac{\lambda_{\min}(A^*A)}{\tau_{\max}}\right)\left(1 - \frac{\lambda_{\min}(A^*A)}{\|A\|_F^2}\right) - \frac{\lambda_{\min}(A^*A)}{\|A\|_F^2}\frac{\tau_{\min}}{\tau_{\max}}\gamma$$

and

$$\beta = \left(1 - \frac{\lambda_{\min}(A^*A)}{\tilde{\tau}_{\max}}\right)\left(1 - \frac{\lambda_{\min}(A^*A)}{\|A\|_F^2}\right) - \frac{\lambda_{\min}(A^*A)}{\|A\|_F^2}\frac{\tilde{\tau}_{\min}}{\tilde{\tau}_{\max}}\tilde{\gamma},$$

Convergence Theorem [7] (Cont'd.)

with $\gamma = \min\left\{\frac{\delta^2(1-\delta)}{1+\delta}, \frac{\Delta^2(1-\Delta)}{1+\Delta}\right\}$, $\tilde{\gamma} = \min\left\{\frac{\tilde{\delta}^2(1-\tilde{\delta})}{1+\tilde{\delta}}, \frac{\tilde{\Delta}^2(1-\tilde{\Delta})}{1+\tilde{\Delta}}\right\}$,

$$\delta = \min_{p\neq q} \frac{|A^{(p)}(A^{(q)})^*|}{\|A^{(p)}\|_2\|A^{(q)}\|_2} \quad \textbf{and} \quad \Delta = \max_{p\neq q} \frac{|A^{(p)}(A^{(q)})^*|}{\|A^{(p)}\|_2\|A^{(q)}\|_2},$$

$$\tilde{\delta} = \min_{p\neq q} \frac{|(A_{(p)})^* A_{(q)}|}{\|A_{(p)}\|_2\|A_{(q)}\|_2} \quad \textbf{and} \quad \tilde{\Delta} = \max_{p\neq q} \frac{|(A_{(p)})^* A_{(q)}|}{\|A_{(p)}\|_2\|A_{(q)}\|_2},$$

and

$$\tau_{\min} = \|A\|_F^2 - \max_{1\le i\le m} \|A^{(i)}\|_2^2, \quad \tau_{\max} = \|A\|_F^2 - \min_{1\le i\le m} \|A^{(i)}\|_2^2,$$
$$\tilde{\tau}_{\min} = \|A\|_F^2 - \max_{1\le j\le n} \|A_{(j)}\|_2^2, \quad \tilde{\tau}_{\max} = \|A\|_F^2 - \min_{1\le j\le n} \|A_{(j)}\|_2^2.$$

[7] Wu, Numer. Algorithms(2022).

If

$$\|A^{(i)}\|_2^2, \quad i = 1, 2, \ldots, m,$$

and

$$\|A_{(j)}\|_2^2, \quad j = 1, 2, \ldots, n,$$

are precomputed,

- **each iteration step of the TREK method will cost** $10m + 10n + 20$ **flops,**
- **every two iteration steps of the REK method will cost** $8m + 8n + 4$ **flops.**

Outline

1 Randomized Kaczmarz

2 Two-subspace Randomized Extended Kaczmarz

3 **Numerical Results**

4 Conclusions and Remarks

- **The IT and CPU mean the medians of the required numbers of iteration steps and the elapsed computing times with respect to 30 times of repeated runs of the corresponding method.**

- **All computations of the TREK and REK2 (each iteration of which consists of two REK iterations) methods are started from $x_0 = 0$ and $z_0 = b$. We check for convergence every $4\min(m, n)$ iterations, and terminate the computation once**

$$\frac{\|b - z_k - Ax_k\|_2}{\|A\|_F\|x_k\|_2} \leq 10^{-5} \quad \textbf{and} \quad \frac{\|A^* z_k\|_2}{\|A\|_F^2\|x_k\|_2} \leq 10^{-5}.$$

- **The IT and CPU mean the medians of the required numbers of iteration steps and the elapsed computing times with respect to 30 times of repeated runs of the corresponding method.**

- **All computations of the TREK and REK2 (each iteration of which consists of two REK iterations) methods are started from $x_0 = 0$ and $z_0 = b$. We check for convergence every $4\min(m, n)$ iterations, and terminate the computation once**

$$\frac{\|b - z_k - Ax_k\|_2}{\|A\|_F\|x_k\|_2} \leq 10^{-5} \quad \textbf{and} \quad \frac{\|A^* z_k\|_2}{\|A\|_F^2\|x_k\|_2} \leq 10^{-5}.$$

- **For the RCD method, the computations are started from $x_0 = 0$, and we check for convergence every $4\min(m, n)$ iterations, and terminate the computation once**

$$\frac{\|A^*(b - Ax_k)\|_2}{\|A^*b\|_2} \leq \varepsilon,$$

where ε is chosen to ensure that the relative solution error of the RCD method, which is defined as

$$\textbf{RSE} = \frac{\|x_{\mathrm{IT}} - x_\star\|_2^2}{\|x_\star\|_2^2}$$

with x_{IT} representing the average of the approximate solution vectors obtained from 30 times of repeated runs of the corresponding method, is of the same order of magnitude as those of the TREK and REK2 methods.

Example A

- **The entries of $A \in \mathbb{R}^{m\times n}$ are the independent identically distributed uniform random variables in the interval $(t, 1)$,**

  and the right-hand side vector is $b = Ax_* + r$, where one of the solutions $x_* \in \mathbb{R}^n$ is generated randomly with the MATLAB function `randn`, and $r \in \mathbb{R}^m$ is a nonzero vector in the null space of A^T generated by the MATLAB function `null`.

- **To guarantee the existence of the nonzero vector r,**

  when $m > n$, the matrix $A \in \mathbb{R}^{m\times n}$ is generated by the MATLAB function `rand`;

  while when $m \leq n$, the first $m-1$ rows of A are generated by the MATLAB function `rand` and the m-th row of A is the average of the first and second rows of A.

Example A

- **The entries of $A \in \mathbb{R}^{m\times n}$ are the independent identically distributed uniform random variables in the interval $(t, 1)$,**

  and the right-hand side vector is $b = Ax_* + r$, where one of the solutions $x_* \in \mathbb{R}^n$ is generated randomly with the MATLAB function `randn`, and $r \in \mathbb{R}^m$ is a nonzero vector in the null space of A^T generated by the MATLAB function `null`.

- **To guarantee the existence of the nonzero vector r,**

  when $m > n$, the matrix $A \in \mathbb{R}^{m\times n}$ is generated by the MATLAB function `rand`;

  while when $m \leq n$, the first $m - 1$ rows of A are generated by the MATLAB function `rand` and the m-th row of A is the average of the first and second rows of A.

Table 1: IT, CPU and RSE of REK2, RCD and TREK for m-by-n matrices A with $t = 0.1$, $n = 500$ and different m

m		1000	2000	3000	4000	5000
cond(A)		**158.90**	**93.50**	**79.94**	**72.33**	**69.07**
REK2	**IT**	$188n$	$76n$	$60n$	$56n$	$48n$
	CPU	**4.3242**	**2.2912**	**2.5201**	**4.2182**	**4.8668**
	RSE	**1.31E-6**	**3.59E-7**	**1.12E-7**	**7.33E-8**	**1.19E-7**
RCD	**IT**	$298n$	$128n$	$108n$	$100n$	$88n$
	CPU	**1.4321**	**0.7174**	**0.7204**	**1.0438**	**1.5972**
	RSE	**2.86E-7**	**4.62E-8**	**9.17E-9**	**4.20E-9**	**3.99E-9**
TREK	**IT**	$68n$	$28n$	$24n$	$24n$	$20n$
	CPU	**1.6713**	**0.9547**	**0.9987**	**1.9746**	**2.3735**
	RSE	**4.06E-7**	**6.54E-8**	**1.06E-8**	**4.47E-9**	**4.68E-9**

Table 2: IT, CPU and RSE of REK2 and TREK for m-by-n matrices A with $t = 0.1$, $m = 500$ and different n

n		1000	2000	3000	4000	5000
cond(A)		**Inf**	**Inf**	**Inf**	**Inf**	**Inf**
REK2	**IT**	$176m$	$76m$	$60m$	$56m$	$48m$
	CPU	**4.1023**	**2.4464**	**2.5774**	**5.2528**	**7.5367**
	RSE	**1.31E-6**	**3.91E-7**	**1.10E-7**	**5.56E-8**	**1.48E-7**
TREK	**IT**	$64m$	$28m$	$24m$	$24m$	$20m$
	CPU	**1.5701**	**0.9649**	**1.0963**	**2.4099**	**3.0217**
	RSE	**3.33E-7**	**7.04E-8**	**7.90E-9**	**1.97E-9**	**5.07E-9**

Table 3: IT, CPU and RSE of REK2, RCD and TREK for m-by-n matrices A with $t = 0.5$, $n = 500$ and different m

m		1000	2000	3000	4000	5000
cond(A)		**399.92**	**232.19**	**196.51**	**177.47**	**169.00**
REK2	**IT**	$788n$	$326n$	$280n$	$224n$	$204n$
	CPU	**18.2133**	**10.3647**	**11.6310**	**20.5536**	**24.7022**
	RSE	**1.54E-5**	**4.48E-6**	**7.81E-7**	**1.48E-6**	**1.18E-6**
RCD	**IT**	$1126n$	$536n$	$486n$	$360n$	$332n$
	CPU	**5.4303**	**3.0037**	**3.2598**	**3.8013**	**5.6518**
	RSE	**6.59E-6**	**8.37E-7**	**6.35E-8**	**3.80E-7**	**3.01E-7**
TREK	**IT**	$64n$	$28n$	$24n$	$20n$	$20n$
	CPU	**1.5650**	**0.9497**	**1.0814**	**1.9496**	**2.8463**
	RSE	**3.08E-6**	**2.12E-7**	**4.28E-8**	**5.82E-8**	**1.74E-8**

Table 4: IT, CPU and RSE of REK2 and TREK for m-by-n matrices A with $t = 0.5$, $m = 500$ and different n

n		1000	2000	3000	4000	5000
cond(A)		**Inf**	**Inf**	**Inf**	**Inf**	**Inf**
REK2	**IT**	$774m$	$316m$	$248m$	$224m$	$204m$
	CPU	**17.7455**	**9.6906**	**10.1737**	**19.3899**	**26.7698**
	RSE	**4.31E-6**	**3.72E-6**	**1.89E-6**	**2.16E-6**	**1.53E-6**
TREK	**IT**	$64m$	$28m$	$24m$	$20m$	$20m$
	CPU	**1.5365**	**0.9685**	**1.0407**	**1.9327**	**2.7177**
	RSE	**1.07E-6**	**1.90E-7**	**2.09E-8**	**7.60E-8**	**2.31E-8**

Table 5: IT, CPU and RSE of REK2, RCD and TREK for m-by-n matrices A with $t = 0.9$, $n = 500$ and different m

m		1000	2000	3000	4000	5000
cond(A)		**2430.90**	**1464.02**	**1234.32**	**1137.74**	**1073.23**
REK2	**IT**	$17998n$	$8402n$	$6692n$	$6118n$	$5218n$
	CPU	**416.9974**	**267.0242**	**273.7145**	**580.2049**	**716.2883**
	RSE	**4.92E-4**	**2.49E-4**	**1.68E-4**	**8.71E-5**	**1.27E-4**
RCD	**IT**	$24814n$	$13954n$	$13760n$	$11670n$	$10166n$
	CPU	**126.4919**	**76.0501**	**93.4548**	**165.0929**	**183.0102**
	RSE	**1.98E-4**	**3.84E-5**	**3.46E-6**	**3.73E-6**	**4.22E-6**
TREK	**IT**	$44n$	$20n$	$20n$	$16n$	$16n$
	CPU	**1.0858**	**0.6852**	**0.9021**	**1.6027**	**2.3717**
	RSE	**8.76E-5**	**1.49E-5**	**2.78E-6**	**2.13E-6**	**1.01E-6**

Table 6: IT, CPU and RSE of REK2 and TREK for m-by-n matrices A with $t = 0.9$, $m = 500$ and different n

n		1000	2000	3000	4000	5000
cond(A)		**Inf**	**Inf**	**Inf**	**Inf**	**Inf**
REK2	**IT**	$15134m$	$8762m$	$6780m$	$6006m$	$5264m$
	CPU	**345.6047**	**273.5227**	**284.3943**	**442.4303**	**598.3631**
	RSE	**6.26E-4**	**4.77E-5**	**1.44E-4**	**1.14E-4**	**1.27E-4**
TREK	**IT**	$40m$	$24m$	$20m$	$16m$	$16m$
	CPU	**0.9911**	**0.8163**	**0.9159**	**1.2868**	**1.8594**
	RSE	**5.75E-5**	**1.80E-6**	**2.09E-6**	**2.43E-6**	**8.07E-7**

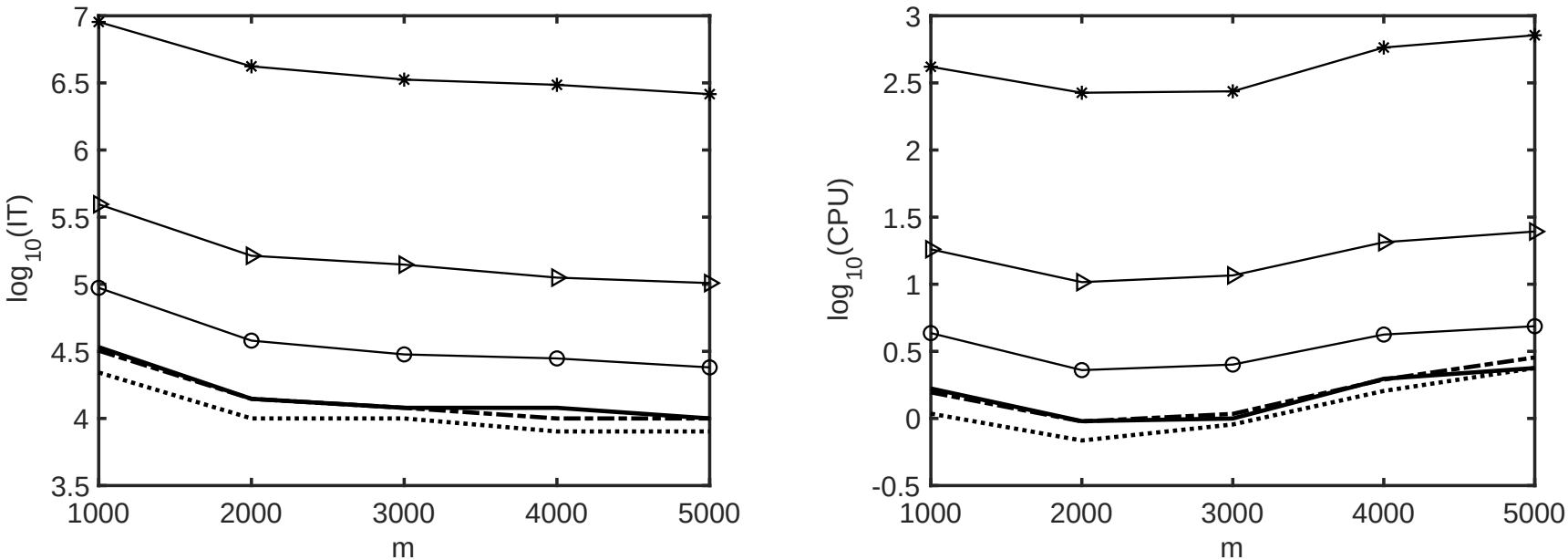


Figure 1: Pictures of $\log_{10}$(IT) (left) and $\log_{10}$(CPU) (right) versus m for REK2 and TREK when $n = 500$. REK2 for $t = 0.1$: "$- \circ -$", REK2 for $t = 0.5$: "$- \triangleright -$", REK2 for $t = 0.9$: "$- * -$", TREK for $t = 0.1$: "—", TREK for $t = 0.5$: "$- \cdot -$" and TREK for $t = 0.9$: "$\cdots$".

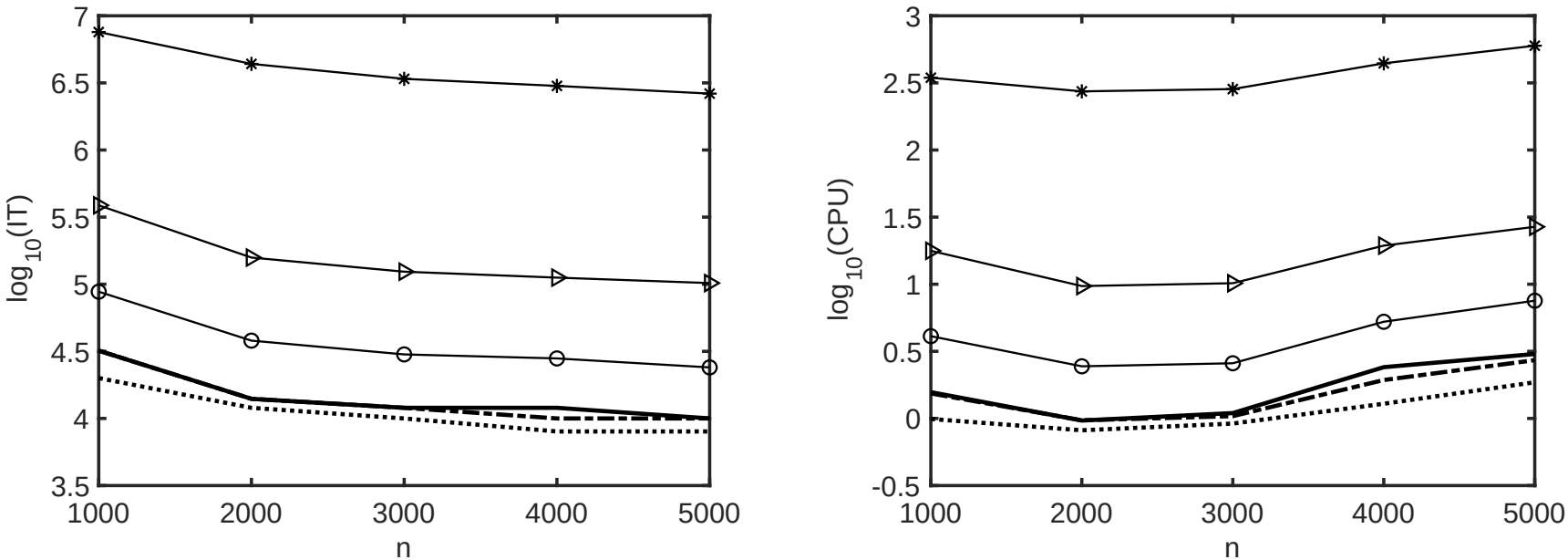


Figure 2: Pictures of $\log_{10}$(IT) (left) and $\log_{10}$(CPU) (right) versus n for REK2 and TREK when $m = 500$. REK2 for $t = 0.1$: "$-\circ-$", REK2 for $t = 0.5$: "$-\triangleright-$", REK2 for $t = 0.9$: "$-*-$", TREK for $t = 0.1$: "—", TREK for $t = 0.5$: "$-\cdot-$" and TREK for $t = 0.9$: "$\cdots$".

Example B

- **We solve the X-ray CT problem generated by a MATLAB package AIR Tools II [9].**
- **The sparse coefficient matrix A is obtained from the discretization schemes of the line integrals along the straight X-rays of the attenuation coefficient of the object which the X-rays penetrate.**
- **The object domain is the square $[-30, 30] \times [-30, 30]$. The source is placed infinitely far from a flat detector with equal spacing between the 90 pixels and each detector pixel being hit by a single X-ray. These rays are parallel and the distance between the first and the last ray is 89. Moreover, the source-detector pair is rotated around the object, and measurements are recorded for angles $0 : 0.5 : 179.5$.**

[9] P.C. Hansen and J.S. Jørgensen, Numer. Algorithms(2018).

Example B

- **We solve the X-ray CT problem generated by a MATLAB package AIR Tools II [9].**
- **The sparse coefficient matrix A is obtained from the discretization schemes of the line integrals along the straight X-rays of the attenuation coefficient of the object which the X-rays penetrate.**
- **The object domain is the square $[-30, 30] \times [-30, 30]$. The source is placed infinitely far from a flat detector with equal spacing between the 90 pixels and each detector pixel being hit by a single X-ray. These rays are parallel and the distance between the first and the last ray is 89. Moreover, the source-detector pair is rotated around the object, and measurements are recorded for angles $0 : 0.5 : 179.5$.**

[9] P.C. Hansen and J.S. Jørgensen, Numer. Algorithms(2018).

Example B

- **We solve the X-ray CT problem generated by a MATLAB package AIR Tools II [9].**
- **The sparse coefficient matrix A is obtained from the discretization schemes of the line integrals along the straight X-rays of the attenuation coefficient of the object which the X-rays penetrate.**
- **The object domain is the square $[-30, 30] \times [-30, 30]$. The source is placed infinitely far from a flat detector with equal spacing between the 90 pixels and each detector pixel being hit by a single X-ray. These rays are parallel and the distance between the first and the last ray is 89. Moreover, the source-detector pair is rotated around the object, and measurements are recorded for angles $0 : 0.5 : 179.5$.**

[9] P.C. Hansen and J.S. Jørgensen, Numer. Algorithms(2018).

Example B (Cont'd.)

- **The right-hand side vector is $b = Ax_\star + r$,**

  where the unique solution $x_\star$ of the least-squares problem is obtained by reshaping the 60×60 image of the well-known Shepp-Logan medical phantom,

  and r is a nonzero vector in the null space of A^T generated by the MATLAB function `null`.

Table 7: IT, CPU and RSE of REK2, RCD and TREK for m-by-n matrix A with $m = 32400$ and $n = 3600$

method	IT	CPU	RSE
REK2	$1930n$	**5465.8577**	**1.82E-4**
RCD	$4124n$	**1326.5625**	**3.78E-5**
TREK	$1694n$	**5279.6786**	**3.30E-5**

(a) exact phantom

(b) REK2

(c) RCD

(d) TREK

Figure 3: Pictures of the exact Shepp-Logan medical phantom, and the approximate solutions obtained by REK2, RCD and TREK.

Outline

1 Randomized Kaczmarz

2 Two-subspace Randomized Extended Kaczmarz

3 Numerical Results

4 Conclusions and Remarks

- **For solving large linear least-squares problems, we construct a block REK method, called the TREK method, each iteration of which utilizes two rows and two columns of the coefficient matrix.**
- **Theoretical analysis and numerical results show that the TREK method outperforms the REK method.**
- **For solving the least-squares problem with full column-rank coefficient matrix, the TREK method can also outperform the RCD method in some cases.**

- **For solving large linear least-squares problems, we construct a block REK method, called the TREK method, each iteration of which utilizes two rows and two columns of the coefficient matrix.**
- **Theoretical analysis and numerical results show that the TREK method outperforms the REK method.**
- **For solving the least-squares problem with full column-rank coefficient matrix, the TREK method can also outperform the RCD method in some cases.**

- **For solving large linear least-squares problems, we construct a block REK method, called the TREK method, each iteration of which utilizes two rows and two columns of the coefficient matrix.**

- **Theoretical analysis and numerical results show that the TREK method outperforms the REK method.**

- **For solving the least-squares problem with full column-rank coefficient matrix, the TREK method can also outperform the RCD method in some cases.**

- **When the angle between the two rows or the two columns used in each iteration is smaller, the TREK iteration will converge more faster than two REK iterations.**

- **However, each TREK iteration will cost $2m + 2n + 16$ extra flops compared to two REK iterations. When the two rows used in each iteration are nearly orthogonal and the two columns used in each iteration are also nearly orthogonal, two REK iterations can already perform well.**

- **Therefore, we can consider to combine the TREK iteration and the REK iteration together and construct a more efficient hybrid method. This is an interesting and valuable topic to be studied in the future.**

- **When the angle between the two rows or the two columns used in each iteration is smaller, the TREK iteration will converge more faster than two REK iterations.**

- **However, each TREK iteration will cost $2m + 2n + 16$ extra flops compared to two REK iterations. When the two rows used in each iteration are nearly orthogonal and the two columns used in each iteration are also nearly orthogonal, two REK iterations can already perform well.**

- **Therefore, we can consider to combine the TREK iteration and the REK iteration together and construct a more efficient hybrid method. This is an interesting and valuable topic to be studied in the future.**

- **When the angle between the two rows or the two columns used in each iteration is smaller, the TREK iteration will converge more faster than two REK iterations.**

- **However, each TREK iteration will cost $2m + 2n + 16$ extra flops compared to two REK iterations. When the two rows used in each iteration are nearly orthogonal and the two columns used in each iteration are also nearly orthogonal, two REK iterations can already perform well.**

- **Therefore, we can consider to combine the TREK iteration and the REK iteration together and construct a more efficient hybrid method. This is an interesting and valuable topic to be studied in the future.**

Thank you for your attention!