Advances of Numerical Methods for Hubbard Quantum Monte Carlo Simulations (Part II)

Zhaojun Bai[†],
Wenbin Chen[‡],
Richard Scalettar[§],
Ichitaro Yamazaki[†]

[†]Computer Science,

[§]Physics, UC Davis

[‡] Mathemtics, Fudan University

Outline

1. Hubbard model and quantum monte carlo simulation: an outline (Part I)
2. Hubbard matrix analysis (Part I)
3. Self-adapting direct linear solvers
4. Preconditioned iterative linear solvers

Self-Adapting Direct Linear Solvers

Block LU factorization

- Gaussian elimination

$$\begin{bmatrix} I & & & B_1 \\ -B_2 & I & & \\ & \ddots & \ddots & \\ & & -B_L & I \end{bmatrix} = \begin{bmatrix} I & & & \\ -B_2 & I & & \\ & \ddots & \ddots & \\ & & -B_L & I \end{bmatrix} \begin{bmatrix} I & & & B_1 \\ & I & & B_2 B_1 \\ & & \ddots & \vdots \\ & & I & B_{L-1} \cdots B_1 \\ & & & \underline{I + B_L \cdots B_2 B_1} \end{bmatrix},$$

- Applicable only for small energy scales, say, $U = 0, \beta = 1$.
- The block $I + B_L \cdots B_2 B_1$ grows exponentially
- Pivoting doesn't help.
- Related work in two-point BVP

Block cyclic reduction

- Block cyclic reduction

$$\begin{bmatrix} I & & & & B_1 \\ -B_2 & I & & & \\ & -B_3 & I & & \\ & & -B_4 & I & \\ & & & -B_5 & I \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix},$$

$$\Rightarrow \quad \begin{bmatrix} I & & B_1 \\ -B_3B_2 & I & \\ & -B_5B_4 & I \end{bmatrix} \begin{bmatrix} x_1 \\ x_3 \\ x_5 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_3^{(2)} \\ b_5^{(2)} \end{bmatrix},$$

$$\Rightarrow \quad \begin{bmatrix} I & B_1 \\ -B_5B_4B_3B_2 & I \end{bmatrix} \begin{bmatrix} x_1 \\ x_5 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_5^{(3)} \end{bmatrix}.$$

- Buzbee, Golub and Nielson, ...
- Full BCR is usable only for small energy scales, $U = 0, 1, 2$.
- However, the reduction idea is powerful.

Block QR decomposition

- Block orthogonal factorization:

$$Q_{L-1}^T \cdots Q_2^T Q_1^T M = R,$$

i.e.

$$\begin{pmatrix} I & & & B_1 \\ -B_2 & I & & \\ & \ddots & \ddots & \\ & & -B_L & I \end{pmatrix} \xrightarrow{Q_k} \begin{pmatrix} R_1 & X & & & X \\ & R_2 & X & & X \\ & & \ddots & \ddots & X \\ & & & R_{L-1} & X \\ & & & & R_L \end{pmatrix}.$$

- The method is stable
- But it requires $O(N^2 L)$ memory and $O(N^3 L)$ flops,

Hybrid method

Block reduction orthogonal factorization method

1. k-step block reduction:

$$Mx = b \quad \Longrightarrow \quad M^{(k)}x^{(k)} = b^{(k)}$$

   i.e.,

$$\underline{\text{block } L\text{–cyclic system}} \quad \Longrightarrow \quad \underline{\text{block } \tfrac{L}{k}\text{–cyclic system}}$$

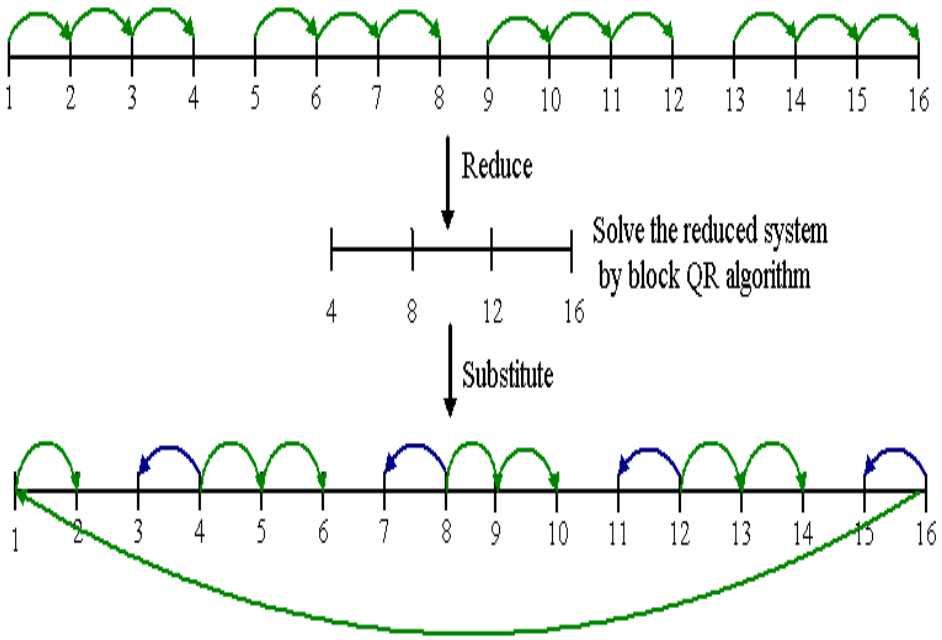
2. Block orthogonal factorization:

$$Q^T_{\frac{L}{k}-1} \cdots Q^T_1 M^{(k)} = R,$$

3. Forward and back substitutions to find the rest of x:

$$x_i \quad \longleftarrow \quad x^{(k)} \quad \longrightarrow \quad x$$

Block reduction orthogonal factorization method



- The order of $M^{(k)}$ is reduced by a factor of k.
- However, the condition number of $M^{(k)}$ increases when k increases.
- Question: how to find the reduction factor k in a **self-adapting** fashion, such that the computed solution has the required accuracy for QMC?

Error analysis

- The relative error in the computed block component $\widehat{x}_\ell$ of x is essentially governed by $\kappa(M^{(k)})\epsilon$ and its propagation in the substitution. Specifically,

$$\frac{\|x_\ell - \widehat{x}_\ell\|}{\|x_\ell\|} \leq \|B_{\frac{k}{2}}\| \cdots \|B_2\| \, \|B_1\| \, \kappa(M^{(k)})\epsilon$$

  where ϵ is the machine precision.

Self-adapting reduction factor

By error analysis of $\widehat{x}_\ell$ and estimation of $\kappa(M^{(k)})$, for a desired accuracy

$$\frac{\|x_\ell - \widehat{x}_\ell\|}{\|x_\ell\|} \leq \text{tol}.$$

the reduction factor k is then **adaptively** determined by

$$k = \left\lfloor \frac{\frac{2}{3}\ln(\text{tol}/\epsilon)}{4t\tau + \nu} \right\rfloor$$

Note: $\nu = \sqrt{U\tau} + ...$

Example: $t = 1$, $\tau = 1/8$, tol $= 10^{-8}$, $\epsilon = 10^{-16}$,

U	0	1	2	3	4	5	6
k	24	14	12	10	9	9	8

Performance data

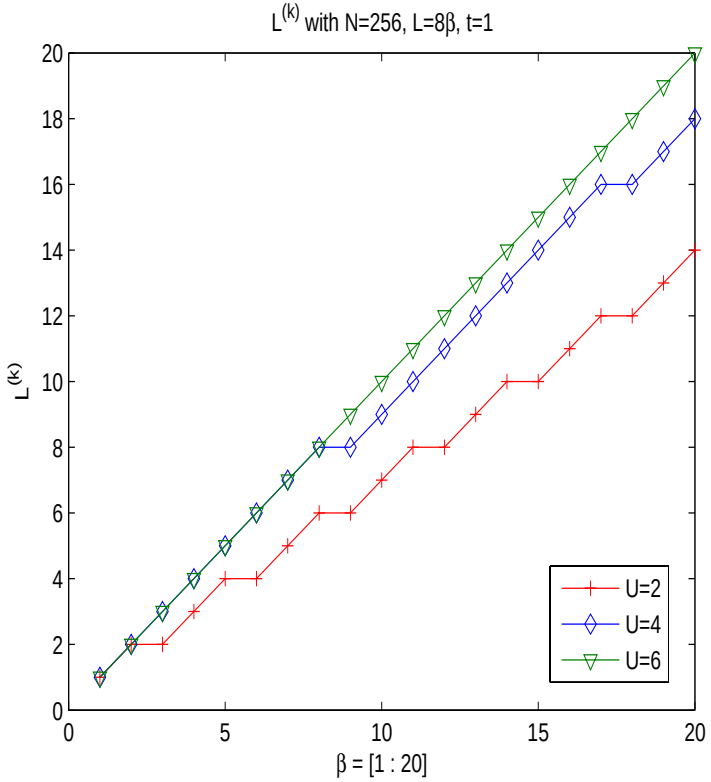
CPU time and and speedup ($N = 256, U = 0, t = 1, \tau = \frac{1}{8}$)

β	$L = 8\beta$	$L^{(k)}$	k	block QR(s)	reduced QR(s)	speedup
1	8	1	8	3.19	0.0293	108
3	24	1	24	10.8	0.0547	197
5	40	2	20	18.6	0.326	57
7	56	3	19	27.2	0.666	40
9	72	3	24	35.1	0.675	52
12	96	4	24	46.0	1.20	38
14	112	5	23	54.0	1.28	42
16	128	6	22	62.9	1.67	37
18	144	6	24	73.2	1.73	42
20	160	7	23	80.2	2.02	39

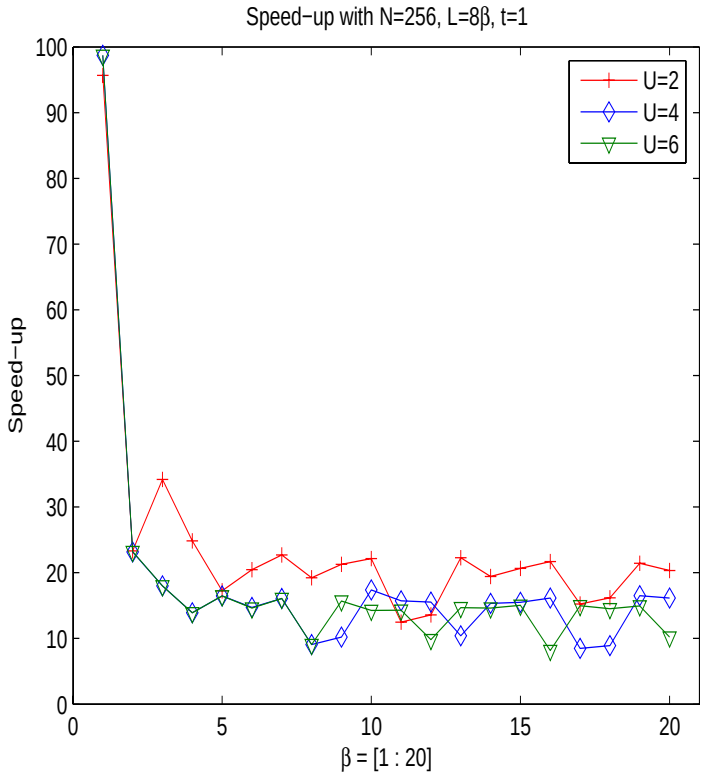
All relative errors of the solution vector are less than 10^{-8}

Performance data

Reduced number of block $\frac{L}{k}$



speedup



$N = 256, \tau = 1/8, L = 8\beta, t = 1$

All relative errors of the solution vectors $\widehat{x}$ are less than 10^{-8}

Preconditioned Iterative Linear Solvers

Preconditioned iterative linear solvers

- Consider the kernel of HQMC

$$M^T M x = b$$

- Symmetrical preconditioned linear system

$$R^{-T}(M^T M)R^{-1} \cdot Rx = R^{-T}b,$$

  where the preconditioner R is constructed such that

  1. $R^T R$ is a good approximation of $M^T M$ in some sense,
  2. the cost of constructing R is affordable,
  3. the application of R is not expensive, namely the system $Rz = c$ is much easier to solve than the original system.

- Preconditioned conjugate gradient (PCG) method.

Early work

Earlier work on preconditioning techniques:

- R is the matrix of M with zero potential energy ($U = 0$)
- R is the matrix of M with zero kinetic energy ($t = 0$)
- R is the square roots of diagonal elments of $M^T M$.

These preconditioners turned out to be of poor quality, such as high costs (memory and flops), or slow convergence when $N, U, \beta(L)$ increase.

Therefore, the questions are

(1) can we have a linear scaling iterative solver in term of the lattice size N?

(2) when U and β increase, whether there is a solver with the number of iterations grow slowly (not "exponentially" as seen from the previous work)?

Incomplete Cholesky preconditioners

- Incomplete Cholesky (IC) factorization is one of the most important preconditioning techniques:

$$M^T M = R^T R + E,$$

  where R is an upper triangular matrix and E is the discard matrix.

- If by only imposing a certain sparsity of the preconditioner R (based on the block structure of M), or by dropping small elements, then typically, it leads to
  - high cost to apply R due to large number of fill-ins,
  - low quality (large number of iterations),
  - not robust, *pivot break-down* due to loss of $M^T M - E > 0$.
- Mathematically, provable existence for such an incomplete decomposition is only for special classes of matrices.

Incomplete Cholesky factorization

- Incomplete Cholesky factorization

$$A = RR^T + \underbrace{S + S^T}_{E},$$

  where R is a lower-triangular matrix, and the error E is a symmetric and S is a *strictly* lower-triangular matrix.

- The ith column of the factorization

$$r_i(i)r_i - e_i = a_i - \sum_{j=1}^{i-1} r_j(i)r_j.$$

  – updating the ith column a_i

$$a_i(i:n) := a_i(i:n) - \sum_{j=i}^{i-1} r_j(i)r_j(i:n).$$

  – compute the pivot $r_i(i) = \sqrt{a_i(i)}$.

  – The remaining elements of r_i, for $j = i+1, \ldots, n$,

$$\begin{cases} r_i(j) = a_i(j)/r_i(i), & s_i(j) = 0 \\ r_i(j) = 0, & s_i(j) = a_i(j) \end{cases}$$

  if *a sparsity constraint* is satisfied, otherwise,

- *Breakdown* when the pivot $a_i(i) \leq 0$.

Robust IC – version 1

- Goal: avoid the pivot breakdown by imposing

$$A - E = A^T - E^T > 0.$$

- RIC1: diagonal updates of E, such that $-E \geq 0$. Thus $A - E > 0$.
- RIC factorization

$$A = RR^T + \underbrace{S + D + S^T}_{E},$$

  where R is lower-triangular, D is diagonal, and S is strictly lower-triangular.

  Two approaches: D is constructed *statically* or *dynamically*.

- Dynamic approach: by the ith column of the factorization

$$\begin{aligned} a_i(i) &:= \sum_{j=1}^{i} r_j(i)^2 + d_i(i), \\ a_i(i+1:n) &= \sum_{j=1}^{i} r_j(i) r_j(i+1:n) + s_i(i+1:n). \end{aligned}$$

– by first updating the ith column a_i

$$a_i(i:n) = a_i(i:n) - \sum_{j=1}^{i-1} r_j(i) r_j(i:n).$$

– Then, for $j = i+1, \ldots, n$,

(1) "decide to keep/drop $a_i(j)$"

$$\begin{cases} a_i(j) = a_i(j), & s_i(j) = 0, & \text{if } \tau_{ij} > \sigma_1, \\ a_i(j) = 0, & s_i(j) = a_i(j), & \text{otherwise,} \end{cases}$$

where $\tau_{ij} = a_i(j)/\sqrt{a_i(i) + d_i(i)}$. and σ_1 is a dropping threshold.

(2) When $s_i(j) \neq 0$, the corresponding diagonal entries $d_i(i)$ and $d_j(j)$ are updated for imposing $-E \geq 0$,

$$d_i(i) = d_i(i) - \delta_i, \quad d_j(j) = d_j(j) - \delta_j,$$

where δ_i and δ_j are chosen such that $\delta_i, \delta_j > 0$ and $\delta_i \delta_j = s_i(j)^2$.

– Compute the ith column of R:

$$\begin{aligned} r_i(i) &= \sqrt{a_i(i) + d_i(i)}, \\ r_i(i+1:n) &= a_i(i+1:n)/r_i(i). \end{aligned}$$

- RIC1 is based on simple diagonal updates of E, the construction of R is computationally efficient.
- If many non-zero entries of R need to be dropped, then RIC1 needs large number of diagonal updates, and the resulting preconditioner may not be a good approximation of Cholesky factor. In this situation, static approaches such as the global shifting of diagonal elements may perform better.
- Robustness: for any $v \neq 0$,

$$v^T(-E)v = \sum_{i,j \text{ s.t. } s_i(j)\neq 0} (\sqrt{\delta_i}v(i) - \sqrt{\delta_j}v(j))^2 \geq 0,$$

- To measure the quality of RIC1 preconditioner R, we note that the norm of the residue

$$R^{-1}AR^{-T} - I = -R^{-1}(D + S + S^T)R^{-T} = -R^{-1}ER^{-T}$$

  can be amplified by the factor of $\|R^{-1}\|^2$ of the norm of the error matrix E.
- When approximating an ill-conditioned matrix A, especially for strong interacting energy scale ($U = 5, 6$), the norm $\|R^{-1}\|$ is large, and the resulting R is often a poor preconditioner.

RIC - version 3

RIC3: Robust Incomplete Cholesky (version 3)

- imposes (1) the structure of E, (2) the sparsification of F, and (3) positive definitness of $M^TM - E$, i.e.,

$$\begin{cases} M^TM - E = R^TR \\ \text{subject to } \; E = R^TF + F^TR + S, \\ \qquad\qquad\quad M^TM - E > 0, \end{cases}$$

- RIC3 is robust, provable no-pivot-break-down
- RIC3 is of high quality because the residual norm

$$R^{-T}(M^TM)R^{-1} = I - \underbrace{FR^{-1} - R^{-T}F - R^{-T}SR^{-1}}$$

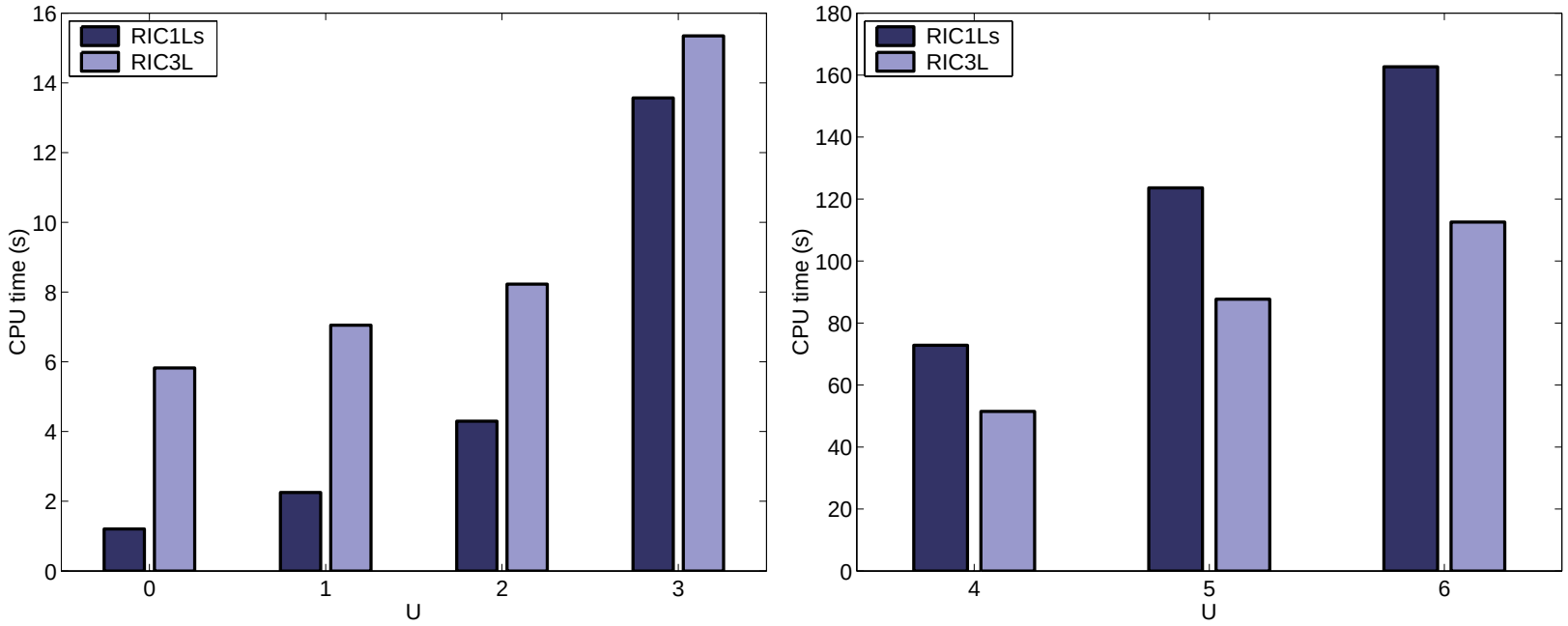
  is amplified only by $\|R^{-1}\|$ if $\|S\| \leq \|F\|/\|R^{-1}\|$. Typically, $\|S\| \leq \|F\|^2$
- RIC3 balances the quality and and construction cost.
- Related work [Ajiz & Jennings'94, Tismenetsky'91, Kaporin '98].

Numerical experiments

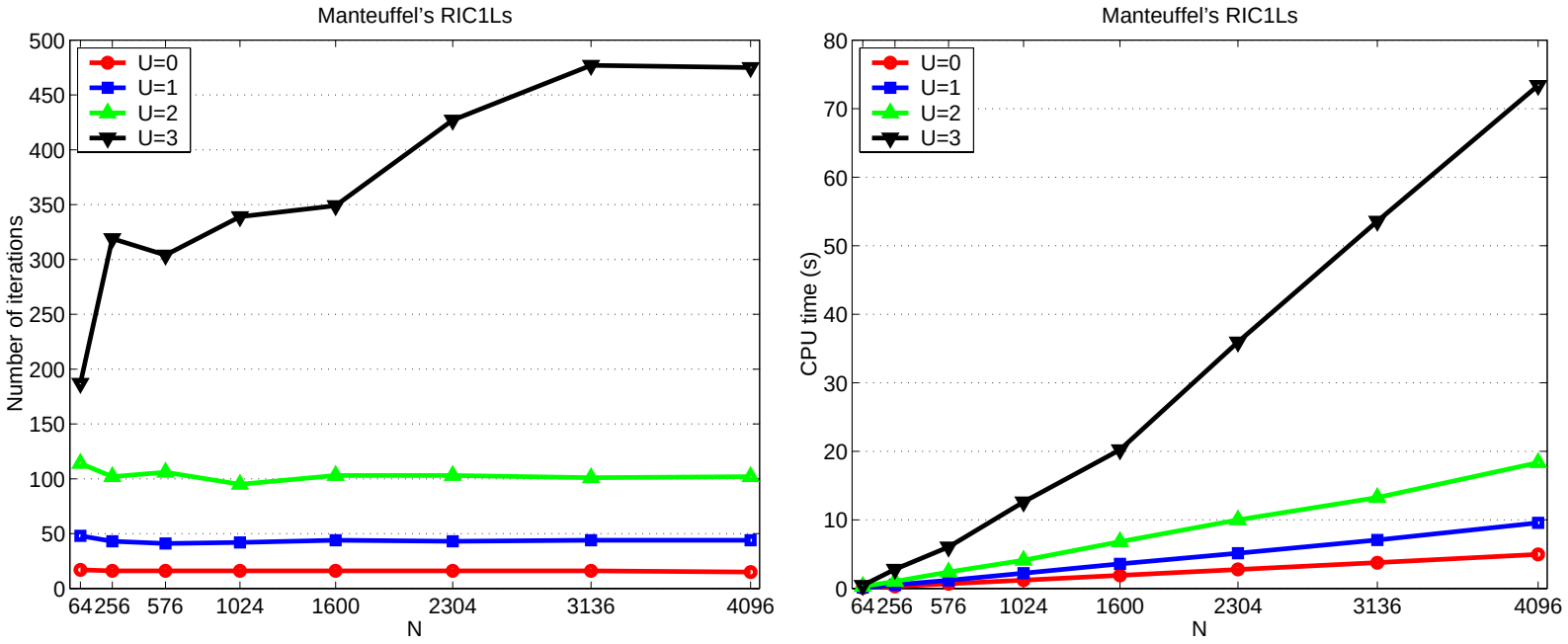
1. *A modified Compressed Sparse Row (CSR) format for sparse matrix data structure is proposed to accommodate the data access pattern in RIC3 factorization.*
2. Dropping tolerance value is set 10^{-2} and 10^{-4} in RIC3.
3. Stopping criterion for the PCG loop is set to 10^{-8}
4. $t = 1, \tau = \frac{1}{8}$ and $\mu = 0$.
5. Itanium2 workstation, 1Ghz CPU and 2GB RAM ...

Performance Data

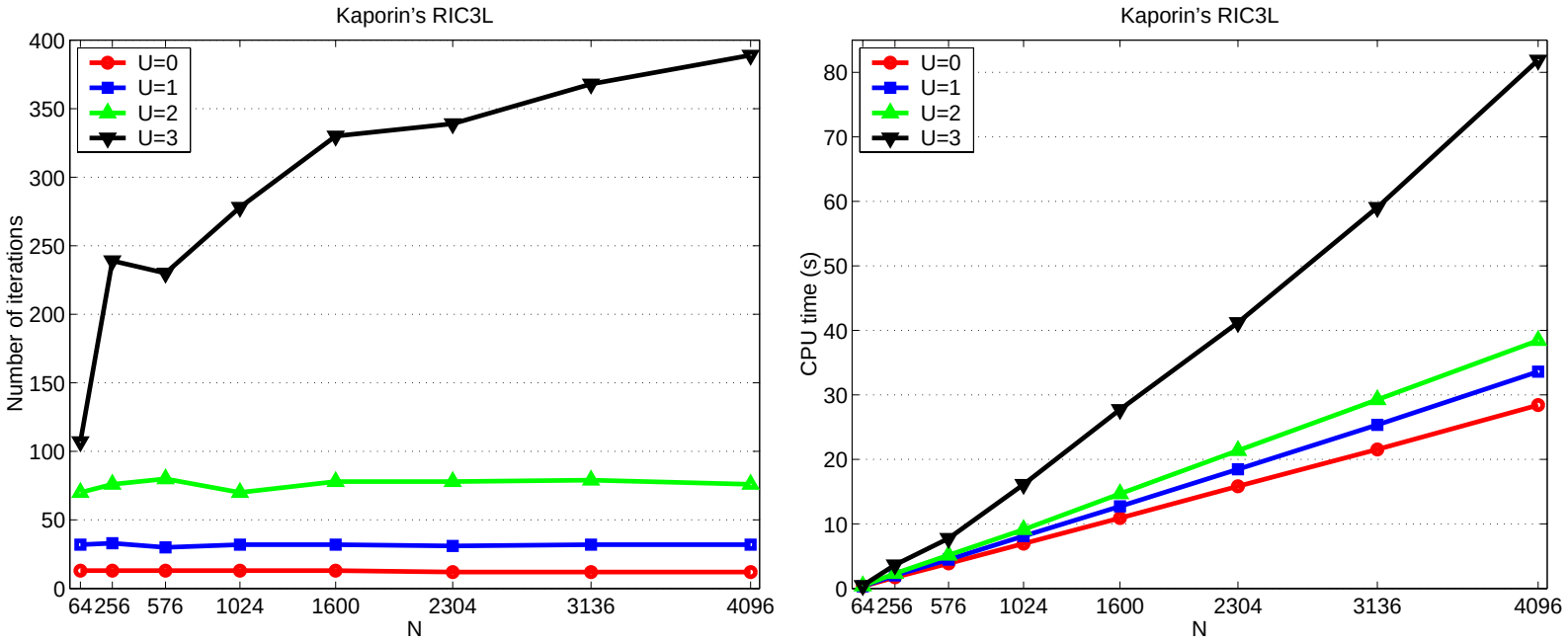
CPU time $(N, L, t, \beta, \mu) = (32 \times 32, 80, 1, 10, 0)$



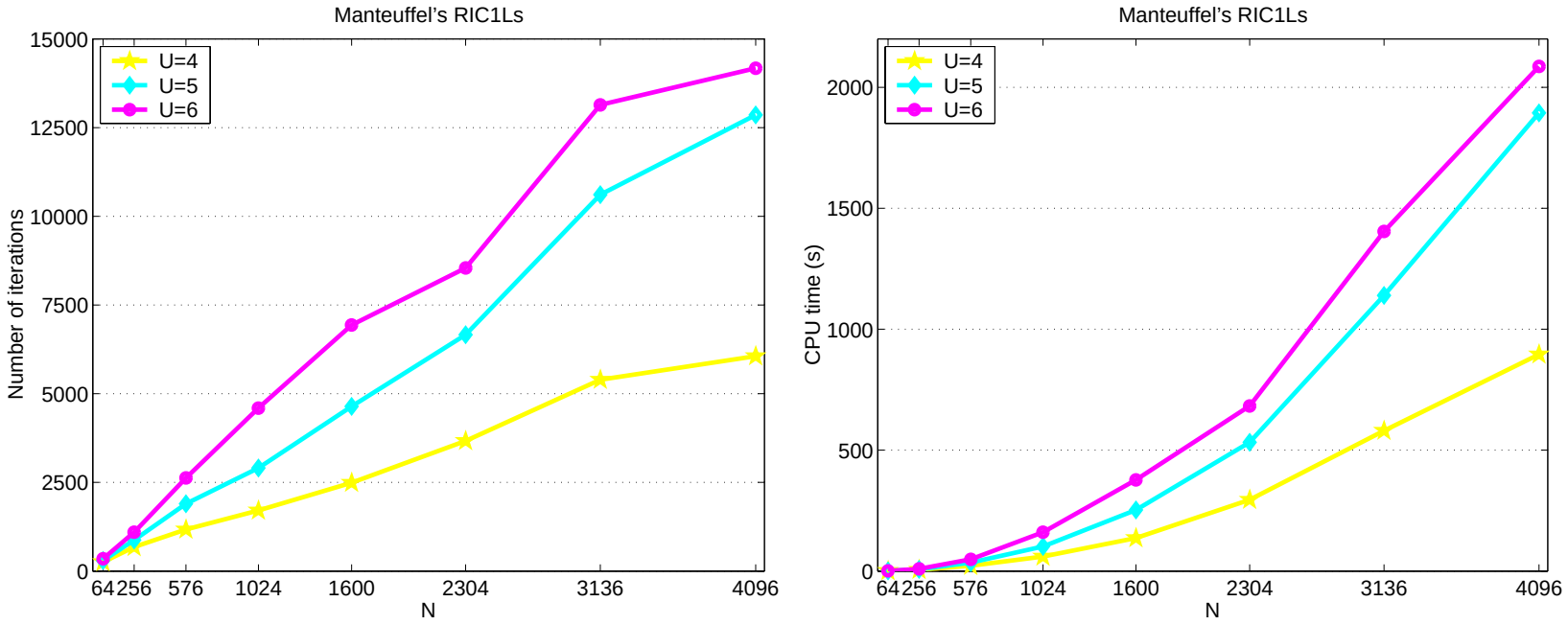
RIC1L$_s$: number of PCG iterations (left) and total CPU time (right) for small U and $(L, t, \beta, \mu) = (80, 1, 10, 0)$.



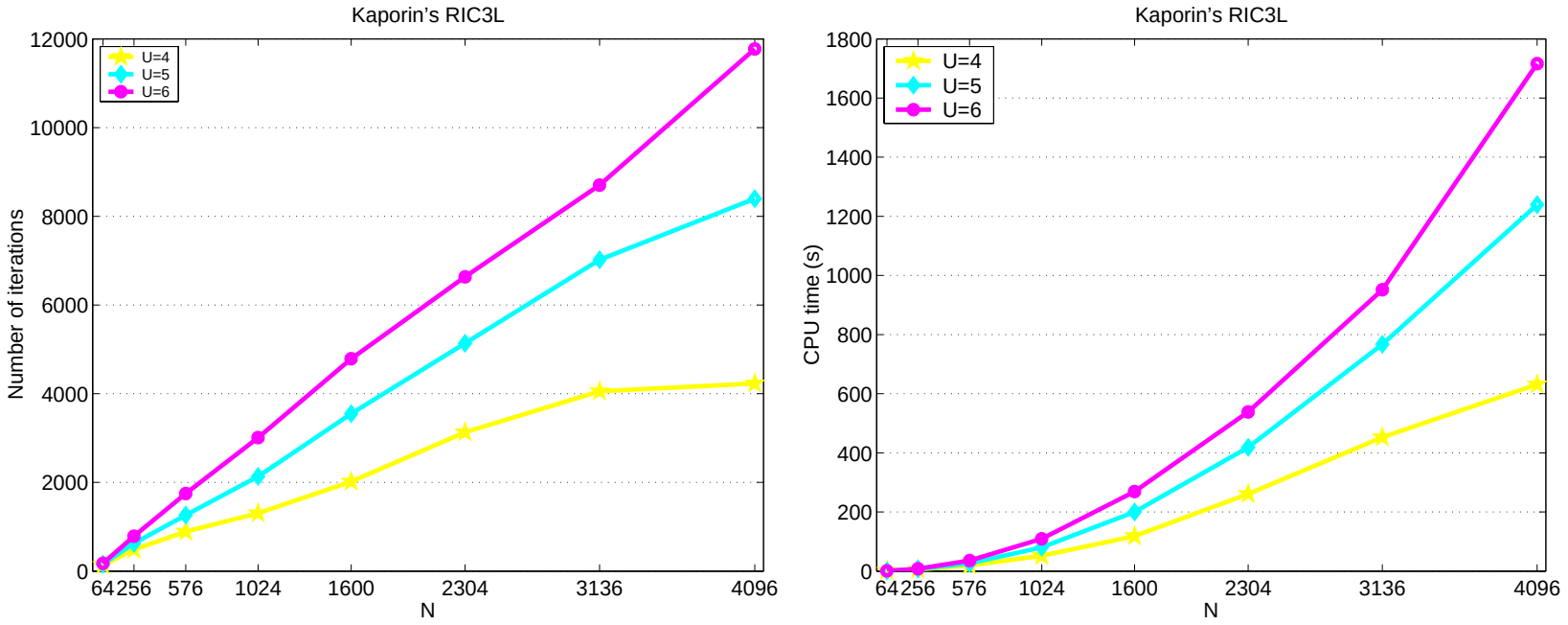
RIC3L: number of PCG iterations (left) and total CPU time (right) for small U and $(L, t, \beta, \mu) = (80, 1, 10, 0)$.



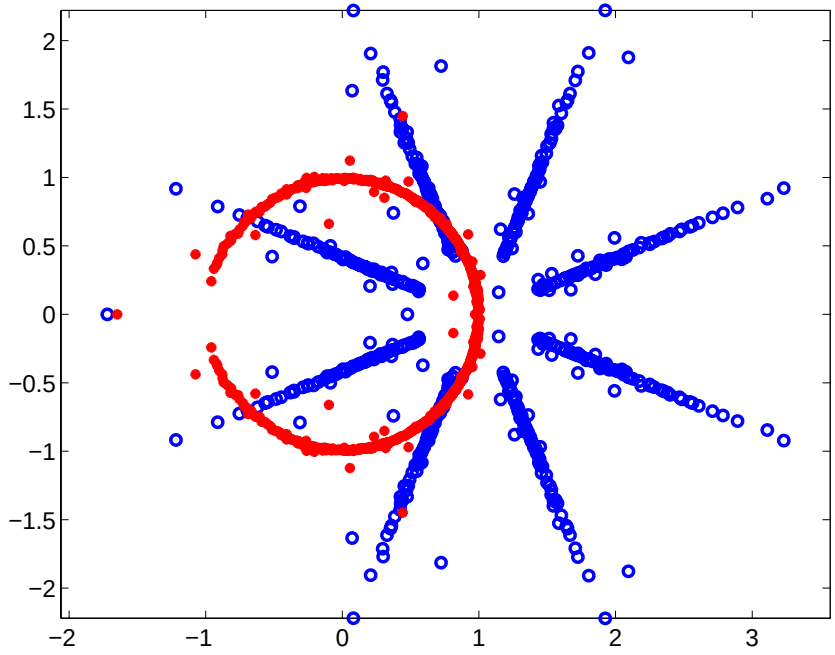
RIC1L$_s$: number of PCG iterations (left) and total CPU time (right) for large U and $(L, t, \beta, \mu) = (80, 1, 10, 0)$.



RIC3L: number of PCG iterations (left) and total CPU time (right) for large U and $(L, t, \beta, \mu) = (80, 1, 10, 0)$.



Eigenvalue distributions of M and MR^{-1}:



Relative residual norm from PCG iterations (Left)