VirginiaTech
Invent the Future

Improve Scaling Quantum Monte Carlo Simulations for Insulators

Eric de Sturler
sturler@vt.edu,
http://www.math.vt.edu/people/sturler

QMC Summer School 2012
University of Illinois at Urbana-Champaign, July 23 – 27, 2012

Collaborators and Support

- Kapil Ahuja, Bryan Clark
- David Ceperley, Jeongnim Kim
- Li Ming, Arielle Grim McNally
- Burkhard Militzer, Ron Cohen

- Funded by NSF 1025327

Overview

- Scaling of QMC Importance Sampling
- Intro to Krylov Subspace Methods
- Approximate Determinant Ratio – Sparse Case
- A Sequence of Preconditioners
- Results
- Conclusions

- (Some New Ideas if Time)

Background: QMC for Deep Earth Materials

- Important for geophysicists to understand properties of materials very deep in the ground
- Under high pressure and temperature DFT and other methods cannot predict certain properties accurately (and DFT has no error bars)
- QMC needs few assumptions, accurate, but currently too expensive – better algorithms for the linear algebra
- Collaboration VT, UIUC, CIW, UC Berkeley
- Ahuja, Ceperley, Clark, de Sturler, and Kim: Make QMC much faster for many particles
- NSF Collaborations Math & Geosciences – 1025327
- Materials Computation Center/NSF (ITR) DMR

Quantum Monte Carlo Method

Quantum Monte Carlo for electronic structure calculations: variational optimization of functions of the density of electrons.

The multiparticle wavefunction: $\Psi_\alpha\left(R\right) = \Psi\left(r_1,\ldots,r_N;\alpha_1,\ldots,\alpha_s\right)$

where r_i has position and spin (we mostly ignore spin in this talk)

Wavefunction gives the probability (density) of finding a particle (and spin) at a region in space (not always normalized).

We want to optimize some function over parameter space (vector α), for example

$$E\left[\Psi_\alpha\right] = \frac{\int \Psi_\alpha^*\left(R\right)\mathcal{H}\Psi_\alpha\left(R\right)dR}{\int \Psi_\alpha^*\left(R\right)\Psi_\alpha\left(R\right)dR}$$

(this talk: real and normalized)

High dimensional (> few particles) need Monte Carlo method to evaluate

Quantum Monte Carlo Method

Sample *local energy* E_L to evaluate expected energy

$$E[\Psi] = \int \Psi(R)\mathcal{H}\Psi(R)\,dR = \int \Psi^2 \left(\frac{\mathcal{H}\Psi}{\Psi}\right) dR = \int \Psi^2 E_L\,dR$$

More generally, for observable $\mathcal{O}(R;\Psi)$: $\int \Psi^2(R)\mathcal{O}(R;\Psi)\,dR$

Approximate integral by Monte Carlo sampling, compute average $\langle \mathcal{O}(R;\Psi) \rangle$

However, Ψ very small except at very localized regions

So need *importance sampling* – sample preferentially from regions with high probability

Need to sample with right probability

Quantum Monte Carlo Method

Approximate $E\left[\Psi\right] = \int \Psi^2\left(R\right) E_L\left(R\right) dR \quad = \int \Psi^2\left(R\right) \left(\frac{\mathcal{H}\Psi\left(R\right)}{\Psi\left(R\right)} \right) dR$

Sample $E_L\left(R\right)$ from density $\Psi^2\left(R\right)$.

Generate multiple configurations R and for each do (repeatedly)

1. generate random step of size d
2. accept move with probability $\min\left(\frac{\Psi^2\left(R+d\right)}{\Psi^2\left(R\right)}, 1 \right)$
3. if accepted evaluate observable at new configuration

Needs initialization period to obtain equilibrium (sequence of configurations has desired density).

In practice we move a single particle in each step and evaluate observable after sufficiently many steps for decorrelation (sweep is N steps)

What is $\Psi_\alpha\left(R\right)$?

Variational form.: the space determines quality of approximation.

Having basis functions that approximate solutions reduces number of basis functions needed: reduce (linear algebra) cost.

Simple and effective: choose wave functions that are products (independence) of single particle wave functions (simple problem)

$$\Psi\left(R\right)=\psi_1\left(r_1\right)\psi_2\left(r_2\right)\cdots\psi_N\left(r_N\right)$$

For electrons we need wave functions that are anti-symmetric

$$\Psi\left(R\right)=\sum_{P(R)}\pm\left(P\right)\psi_1\left(r_1\right)\psi_2\left(r_2\right)\cdots\psi_N\left(r_N\right)=\det\left(\left(\psi_j\left(r_i\right)\right)\right)$$

Slater determinants

We use $\tilde{\Psi}\left(R\right)=e^{-U\left(R\right)}\Psi\left(R\right)$ but extra factor is cheap

Slater Determinants / Matrices

$$A = \begin{pmatrix} \psi_1(r_1) & \psi_2(r_1) & \cdots & \psi_N(r_1) \\ \psi_1(r_2) & \psi_2(r_2) & & \psi_N(r_2) \\ \vdots & & \ddots & \vdots \\ \psi_1(r_N) & \psi_2(r_N) & \cdots & \psi_N(r_N) \end{pmatrix}$$

Slater matrix

$$\Psi(R) = \det(A)$$

$$\tilde{R} = (r_1, \ldots, r_{k-1}, \tilde{r}_k, r_{k+1}, \ldots, r_N)$$

$$u_k^T = \begin{pmatrix} \psi_1(\tilde{r}_k) - \psi_1(r_k) & \psi_2(\tilde{r}_k) - \psi_2(r_k) & \cdots & \psi_N(\tilde{r}_k) - \psi_N(r_k) \end{pmatrix}$$

$$\frac{\Psi(\tilde{R})}{\Psi(R)} = 1 + u_k^T A^{-1} e_k$$

Slater Determinants / Matrices $O\left(N^3\right)$

- The standard algorithm maintains an explicit copy of the inverse matrix
- For an accepted move the inverse is updated using the Sherman-Morrison or Woodbury formula (if multiple particles are moved at once) – cost $O(N^3)$

$$\left(A + e_k u^T\right)^{-1} = A^{-1} - \left(1 + u^T A^{-1} e_k\right)^{-1} A^{-1} e_k u^T A^{-1}$$

- The explicit inverse is useful as the local energy (Hamiltonian) requires

$$\nabla_i \Psi_\alpha\left(R\right) \Big/ \Psi_\alpha\left(R\right) \quad \text{and} \quad \nabla_i^2 \Psi_\alpha\left(R\right) \Big/ \Psi_\alpha\left(R\right)$$

- once per sweep (for all i) and so each column of the inverse will be used

Approximate Determinant Ratio

Major cost computing $\frac{\det\left(\tilde{A}\right)}{\det\left(A\right)}$ where $a_{ij} = \phi_j\left(r_i\right)$ and moving particle k,

$\tilde{a}_{kj} = \phi_j\left(\tilde{r}_k\right)$ - so one row of matrix is changed.

Many ways to compute ratio, but needs to be done many, many times.
Order-N method requires moving all particles once at $O\left(N\right)$ cost.

Current methods $O\left(N^3\right)$

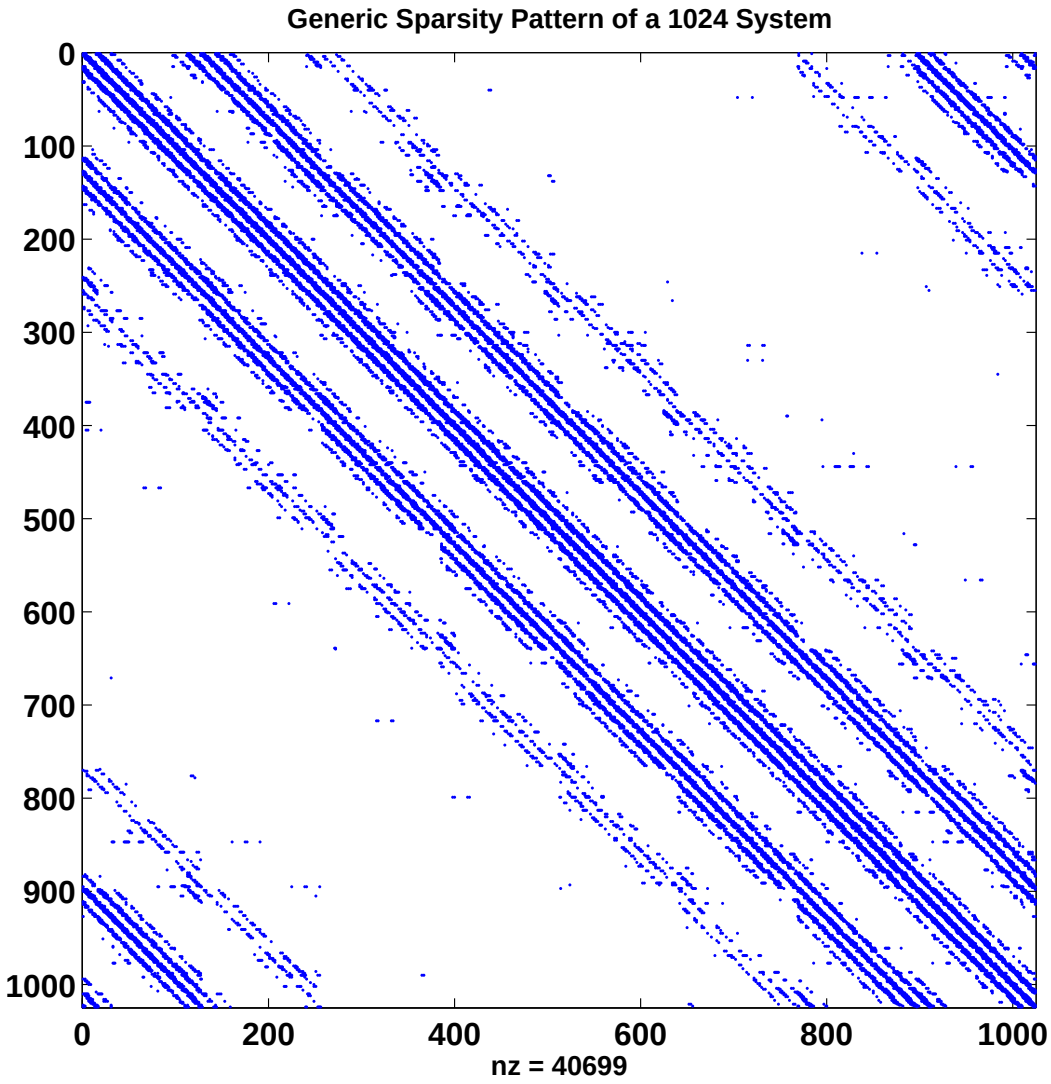
Ratio: $1 + u^T A^{-1} e_k$
Approximate by solving linear system, generalized eigenvalue problem, estimate bilinear form, … (many more possible ways)
Difficulty is solving a simple problem *very fast, many times*

Cost depends on sparsity of matrix – decay/locality of orbitals
Depends on type of problem: insulator, semiconductor, metallic, …

Optimally Sparse Slater Matrices

- For sufficiently localized basis functions the matrix is (quite) sparse
- This can be achieved by optimizing the basis of (standard) single particle wave functions to obtain maximally localized (Wannier) basis functions
- A change of basis does not change the solution to the variational problem
- Reduces computation of Slater matrices to $O(N)$, but determinant ratios still $O(N^3)$ per sweep (N steps)
- Our methods work for any system where the matrix can be made sparse or fast matvec possible
- For metallic systems optimizing locality may not work, but other approaches might be possible

Generic Sparsity Pattern (reordered)



Krylov Methods Crash Course

Consider $Ax = b$ and no (or explicit) preconditioning.
Given x_0 and $r_0 = b - Ax_0$, compute optimal update z from
$K^m(A, r_0) = \text{span}\{r_0, Ar_0, \ldots, A^{m-1}r_0\}$:

$$\min_{z \in K^m(A,r_0)} \|b - A(x_0 + z)\|_2 \quad \Leftrightarrow \quad \min_{z \in K^m(A,r_0)} \|r_0 - Az\|_2$$

Let $K_m = \begin{bmatrix} r_0 & Ar_0 & A^2 r_0 & \cdots & A^{m-1} r_0 \end{bmatrix}$, then $z = K_m \zeta$,
and we must solve the following least squares problem

$$AK_m\zeta \approx r_0 \quad \Leftrightarrow \quad \begin{bmatrix} Ar_0 & A^2 r_0 & \cdots & A^m r_0 \end{bmatrix} \zeta \approx r_0$$

Do this accurately and efficiently every iteration for increasing m.
Arnoldi recurrence: $AV_m = V_{m+1}\underline{H}_m$, where $v_1 = r_0 / \|r_0\|_2$,
$V_{m+1}^H V_{m+1} = I_{m+1}$, and $\text{range}(V_{m+1}) = \text{range}(K_{m+1})$
$\|r_0 - AV_m y_m\|_2 = \|V_{m+1} e_1 \|r_0\|_2 - V_{m+1}\underline{H}_m y_m\|_2 = \|e_1 \|r_0\|_2 - \underline{H}_m y_m\|_2$

GMRES, Saad&Schultz '86

Krylov Methods Crash Course

Consider $Ax = b$ (or preconditioned system $PAx = Pb$)

Given x_0 and $r_0 = b - Ax_0$, compute optimal update z_m from

$$K^m(A, r_0) = \text{span}\{r_0, Ar_0, \ldots, A^{m-1}r_0\}:$$

$$\min_{z \in K^m(A,r_0)} \| b - A(x_0 + z) \|_2 \Leftrightarrow \min_{z \in K^m(A,r_0)} \| r_0 - Az \|_2$$

Let $K_m = \left[r_0 \; Ar_0 \; A^2 r_0 \cdots A^{m-1} r_0 \right]$, then $z = K_m \zeta$,

and we must solve the following least squares problem

$$AK_m\zeta \approx r_0 \quad \Leftrightarrow \quad \left[Ar_0 \; A^2 r_0 \cdots A^m r_0 \right] \zeta \approx r_0$$

GMRES – Saad and Schulz '86, GCR – Eisenstat, Elman, and Schulz '83

$$K_m\zeta = \zeta_0 r_0 + \zeta_1 Ar_0 + \cdots + \zeta_{m-1} A^{m-1} r_0 = p_{m-1}(A) r_0 \;\&\; p_{m-1} \text{ arbitrary}$$

$$r_m = r_0 - Ap_{m-1}(A) r_0 = \left(I - Ap_{m-1}(A) \right) r_0 = q_m(A) r_0 \;\rightarrow\; q_m(0) = 1$$

Minimum Residual Solutions: GMRES

Solve $Ax = b$: Choose x_0; set $r_0 = b - Ax_0$; $v_1 = r_0 / \|r_0\|_2$, $k = 0$.

while $\| r_k \|_2 \geq \varepsilon$ do

$k = k + 1$

$\tilde{v}_{k+1} = A v_k$;

for $j = 1 \ldots k$,

$h_{j,k} = v_j^* \tilde{v}_{k+1}$; $\tilde{v}_{k+1} = \tilde{v}_{k+1} - h_{j,k} v_j$;

end

$h_{k+1,k} = \|\tilde{v}_{k+1}\|_2$; $v_{k+1} = \tilde{v}_{k+1} / h_{k+1,k}$;

Solve LS $\min_\zeta \left\| \eta_1 \|r_0\|_2 - \underline{H}_k \zeta \right\|_2$ $\left(= \|r_k\|_2 \right)$ by construction

(in practice we update the solution each step)

end

$x_k = x_0 + V_k \zeta$;

$r_k = r_0 - V_{k+1} \underline{H}_k \zeta = V_{k+1} \left(\eta_1 \|r_0\| - \underline{H}_k \zeta \right)$ or simply $r_k = b - Ax_k$

GMRES, Saad & Schultz '86

Krylov Spaces

Krylov space is a *space of polynomials in a matrix times a vector*.

Krylov space inherits the approximation properties of polynomials on the real line or in the complex plane.

Let A be diagonalizable, $A = V\Lambda V^{-1}$ (simplify explanation)
Then $A^2 = V\Lambda V^{-1}V\Lambda V^{-1} = V\Lambda^2 V^{-1}$ and generally $A^k = V\Lambda^k V^{-1}$.
So, the polynomial $p_m(t) = \alpha_0 + \alpha_1 t + \cdots + \alpha_m t^m$ applied to A gives

$$p_m(A) = V\left(\alpha_0 I + \alpha_1\Lambda + \alpha_2\Lambda^2 + \cdots + \alpha_m\Lambda^m\right)V^{-1} \quad \text{and hence}$$
$$p_m(A) = Vp_m(\Lambda)V^{-1} = V\,\mathrm{diag}\left(p_m(\lambda_1),\ldots,p_m(\lambda_n)\right)V^{-1}$$

The polynomial is applied to the eigenvalues individually.

Approximate solutions to linear systems, eigenvalue problems, and more general problems using polynomial approximation can be analyzed/understood this way.

Approximation by Matrix Polynomials

Let $A = V \Lambda V^{-1}$, let $\Lambda(A) \subset \Omega \subset \mathbb{C}$.

If $p_{m-1}(t) \approx \frac{1}{t}$ for all $t \in \Omega$, then $p_{m-1}(A) \approx A^{-1}$

Let $r_0 = V\rho$. Then $p_{m-1}(A) r_0 = \sum_i v_i p_{m-1}(\lambda_i) \rho_i \approx \sum_i v_i \frac{\rho_i}{\lambda_i} = A^{-1} r_0$

$$r_m = q_m(A) r_0 = \left(I - A p_{m-1}(A)\right) r_0 = \sum_i v_i \left(1 - \lambda_i p_{m-1}(\lambda_i)\right) \rho_0 \approx 0$$

If we can construct such polynomials for modest m, we have an efficient linear solver.

This is possible if the region Ω is nice – small region away from origin: clustered eigenvalues

If this is not the case, we improve by *preconditioning*: $PAx = Pb$ s.t. PA has clustered eigenvalues and product with P is cheap.

Approximation by Matrix Polynomials

Let $B = V \Lambda V^{-1}$, let $\Lambda\left(B\right) \subset \Omega \subset \mathbb{C}$.

If $p_m\left(t\right) \approx \frac{1}{t}$ for all $t \in \Omega$, then $p_m\left(B\right) \approx B^{-1}$.

Let $y = V\zeta$. Then $p_m\left(B\right)y = \sum_i v_i p_m\left(\lambda_i\right)\zeta_i \approx \sum_i v_i \frac{\zeta_i}{\lambda_i} = B^{-1}y$

Furthermore, let $\varepsilon \approx 0$ and $\left|\lambda_i - \lambda_j\right| > \delta$ (for some eigenvalue λ_i)

If $p_m\left(t\right) = \begin{cases} \varepsilon, & t \in \Omega \text{ and } \left|t - \lambda_i\right| > \delta, \\ 1, & t = \lambda_i, \end{cases}$

then $p_m\left(B\right)y \approx v_i\zeta_i$.

If we can construct such polynomials for modest m we have an efficient linear solver or eigensolver.

Convergence Bounds

Residual at iteration m: $r_m = p_m(A) r_0$ optimal (2-norm)

Eigenvalue bound $\|r_m\| \le \|V\| \|V^{-1}\| \|r_0\| \min_{\substack{p \in \Pi_m, \\ p(0)=1}} \max_{\lambda \in \Lambda(A)} |p(\lambda)|$

FOV bound $\|r_m\| \le 2 \|r_0\| \min_{\substack{p \in \Pi_m \\ p(0)=1}} \max_{\gamma \in W(A)} |p(\gamma)|$

Alternative FOV bound

$$\|r_m\| \le 2 \|r_0\| \min_{\substack{p \in \Pi_m \\ p(0)=1}} \left[\|P_Q\| \max_{\gamma_1 \in W(Q^* A Q)} p(\gamma_1) + \|P_Y\| \max_{\gamma_2 \in W(Y^* A Y)} p(\gamma_2) \right]$$

Pseudospectrum bound $\|r_m\| \le \|r_0\| \dfrac{\mathcal{L}(\mathcal{C}_\varepsilon)}{2\pi\varepsilon} \min_{\substack{p \in \Pi_m, \\ p(0)=1}} \max_{\gamma \in \mathcal{C}_\varepsilon} |p(\gamma)|$

Preconditioned Iterative Solver

- Compute determinant ratios by preconditioned iterative solve
  - efficient for sparse matrices if fast convergence
  - more generally need preconditioned matvec to be cheap
- Iterative solver with ILU (ILUTP) converges well, but
  - matrix degrades due to continual updating – particle moves
  - loss of diag. dom. leads to unstable ILU decomposition
  - need periodic reordering of matrix and recomputing prec.
  - reordering aims to pair orbitals with nearby particles
  - recomputing ILU expensive, so compute cheap updates to preconditioner (until more expensive than new ILU)
- Three improvements:
  - cheap updates to the preconditioner
  - cheap way to monitor instability
  - effective reordering of matrix for diagonal dominance

Cheap Updates to Preconditioner

Assume we have a good preconditioner: AP is nice (e.g., spectrum)

Update $\tilde{A} = A + e_k u^T = A\left(I + A^{-1} e_k u_k^T\right)$

Update preconditioner: $\tilde{P} = \left(I + A^{-1} e_k u_k^T\right)^{-1} P = \left(I - w u_k^T\right) P$ with

$w = \left(1 + u_k^T A^{-1} e_k\right)^{-1} A^{-1} e_k$ (already available for determinant ratio)

Note that breakdown occurs with zero probability

By construction $\tilde{A}\tilde{P} = AP$, so has same favorable properties
Increasing cost in applying sequence of products of type $\left(I - w u_k^T\right)$
At some point cheaper to recompute ILU

Links with update exact inverse in Broyden type methods (book Kelley) and updating preconditioners in Bergamaschi et al.

Effective Stability of Preconditioner

Compute incomplete decomposition $A = LU + R$ (ignore all or some fill-in)

Accuracy of preconditioner: $\left\| A - LU \right\|_F$

Stability of preconditioner: $\left\| I - A\left(LU\right)^{-1} \right\|_F$

(papers by Benzi, Saad, Chow)

In general stability the most important, but metric expensive to compute.

Replace by *effective* or *local stability*: $\max_i \left\| v_i - A\left(LU\right)^{-1} v_i \right\|_2$

The v_i are (ortho)normal vectors in the Arnoldi recurrence. If local stability small, then the preconditioner is stable over the Krylov space.
Even if the preconditioner is not stable (over whole space).

Reordering Algorithm

1. Label the particles (P_i) and orbitals (O_j) from 1 to n, giving the following Slater matrix A:

$$
\begin{array}{c} \\ P_1 \\ P_2 \\ \\ P_n \end{array}
\begin{array}{c}
\begin{array}{cccc} O_1 & O_2 & & O_n \end{array} \\
\begin{pmatrix}
\phi_1(r_1) & \phi_2(r_1) & \cdots & \phi_n(r_1) \\
\phi_1(r_2) & \phi_2(r_2) & \cdots & \phi_n(r_2) \\
\vdots & \vdots & & \vdots \\
\phi_1(r_n) & \phi_2(r_n) & \cdots & \phi_n(r_n)
\end{pmatrix}
\end{array},
$$

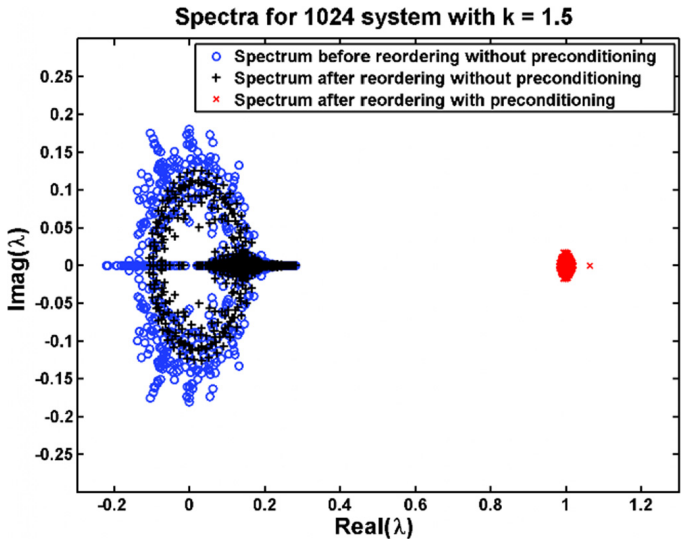
2. for $i = 1, \ldots, n-1$ do
   Find the closest orbital O_j to P_i for $j \in \{i, i+1, \ldots, n\}$
   if $j \neq i$ then
      renumber O_j as O_i and O_i as O_j (swap columns j and i)
   else
      find the particle P_k closest to orbital O_i for $k \in \{i, i+1, \ldots, n\}$
      if $k \neq i$ then
         renumber P_k as P_i and P_i as P_k (swap rows i and k)
      end if
   end if
   end for

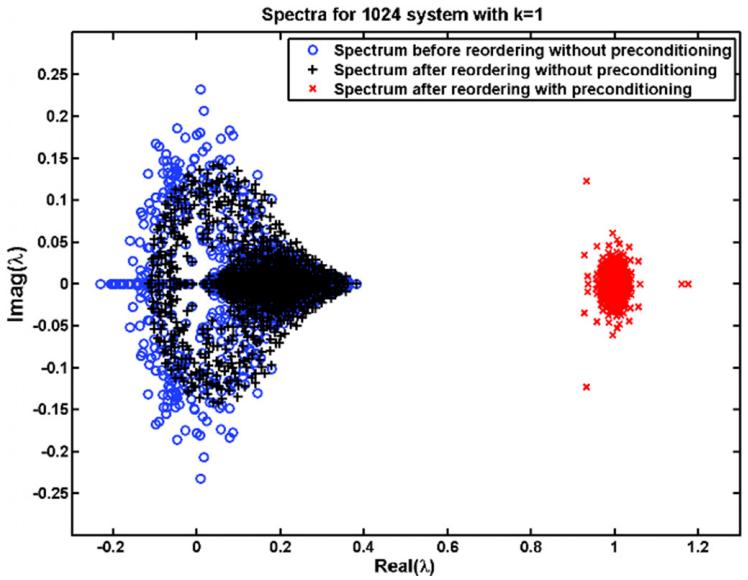
Matrix Reordering

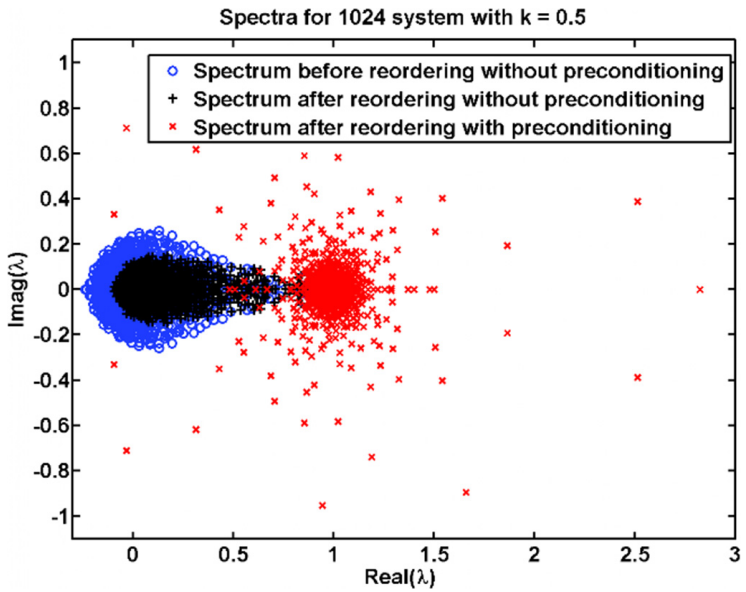
- Greedy algorithm based on geometry
  - Variant of Edmunds '98 (control)
- At each step:
  - Pick new particle (from remaining)
    - Find nearest orbital (swap columns to give same index)
    - If same as current find nearest particle to that orbital (swap rows to give same index)

- Greedy algorithms can be far from optimal, but usually quite effective
- Considering other algorithms for improving near diagonal dominance (Duff and Koster, HSL lib/RAL)
- Really want local incremental update of ordering

Effect of Reordering and ILUTP







Results for Model Problem

- BCC lattice with Gaussian orbitals $\psi_j(x) = e^{-k\|r-Z_j\|^2}$
- Parameter $k = 1$,
- size of cube ≈ 2 ($^3/_{4\pi}$ particles per unit volume)
- $n = 2K^3$ particles

- Check accuracy, statistics, and scaling (in time)

Results for Model Problem

Probability wrong decision in Metropolis algorithm:

$$f = \left| \min\left(q,1\right) - \min\left(q_a,1\right) \right|$$

Average f over random walk (MCMC sequence) gives expected number of errors in acceptance/rejection test

extremely good: $f < 0.0001$
very good: $f < 0.001$
good: $f < 0.01$

Size	686	1024	1458	2000	2662	3456	4394	5488
Exp. Errors	4.45e-6	4.22e-6	5.03e-6	4.41e-6	4.63e-6	4.50e-6	3.96e-6	4.07e-6
Extr. Good	99.49	99.53	99.57	99.49	99.50	99.47	99.56	99.56
Very Good	99.99	99.98	99.99	99.99	99.99	99.99	99.99	99.99
Good	100	100	100	100	100	100	100.00	100.00
Acc. Ratio	0.5879	0.5880	0.5887	0.5880	0.5881	0.5898	0.5878	0.5883

Cost Analysis of Algorithm

Size	686	1024	1458	2000	2662	3456	4394	5488
nr. GMRES iter.	8.91	9.34	9.53	9.59	9.83	10.20	10.22	10.10
nnz(A)/n	42.38	42.38	42.39	42.38	42.37	42.37	42.37	42.37
nnz(L+U)/n	55.04	54.48	54.33	53.61	53.27	53.28	53.28	53.01
nr. reorder/sweep	0.65	1.03	1.19	1.73	2.23	2.63	2.73	3.12

Timing Comparison

Size	686	1024	1458	2000	2662	3456	4394	5488
std alg. (s)	2.52	7.18	16.09	36.83	81.27	173.24	340.94	649.94
sparse alg. (s)	5.71	12.18	24.67	47.55	86.11	167.43	312.05	549.17

Scaling (fit to power law):

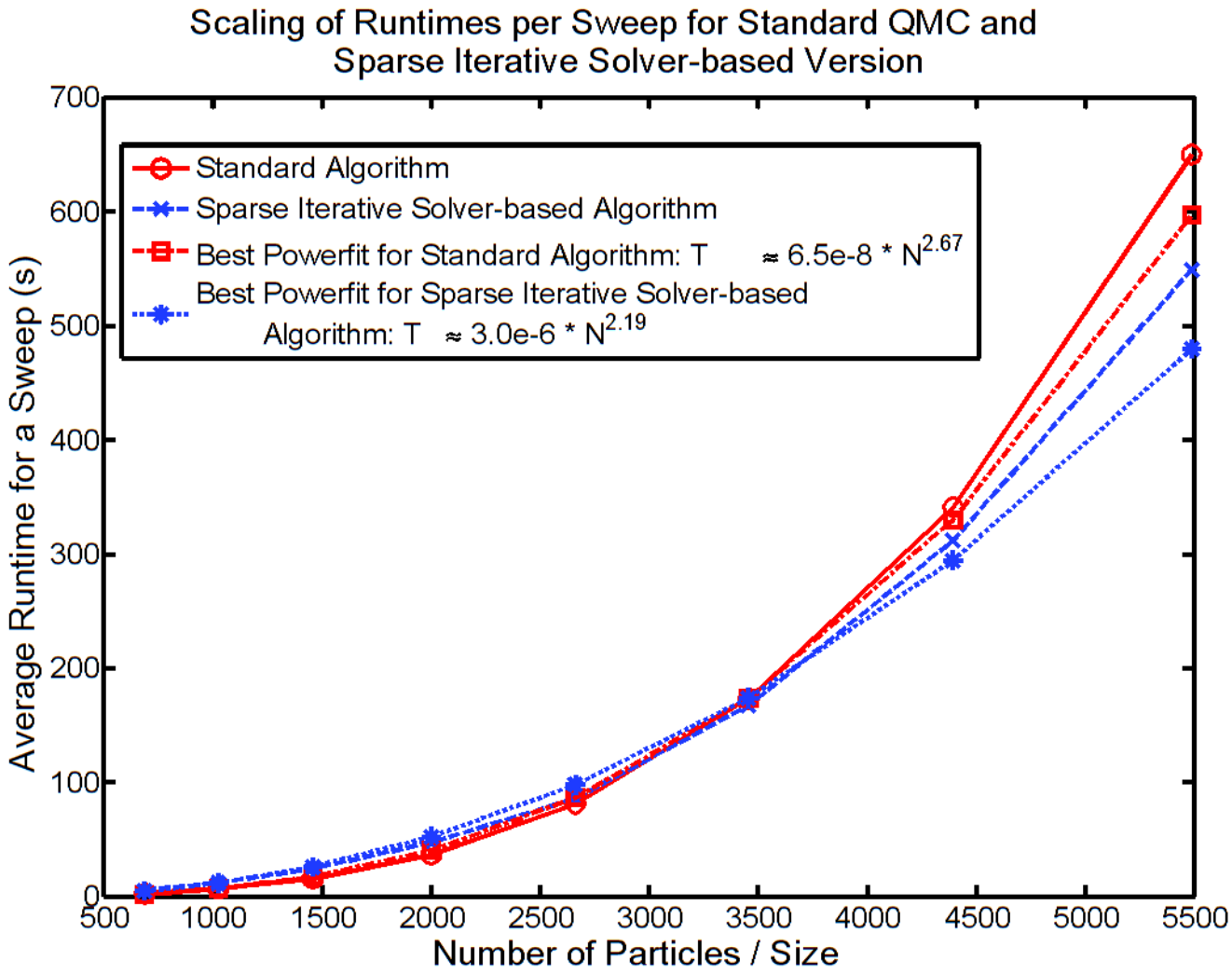
std algorithm: $O(n^{2.67})$

new algorithm: $O(n^{2.19})$

For large n standard algorithm will be cubic (based on algorithm)

Several ways to improve scaling new algorithm

Timing/Scaling Comparison



Further Improvements for Current Approach

- Better data structures – reduce overhead ~20%
- Change # steps of cheap preconditioner update – better scaling (better time)
- Improve reordering global algorithm
  - Faster and/or better diagonal dominance
  - Better preconditioners (reduce iterations)
- Change reordering to incremental local reordering
  - Faster and better scaling
- Compute bilinear form solving two systems by BiCG
  - Products of residual norms governs convergence (exact)
  - In practice, property is lost relatively quickly
  - Clever algorithm by Strakos and Tichy preserves property in floating point arithmetic

Conclusions and Future Work

- Significant improvement in scaling for QMC
- Several further improvements obvious (but implementation not necessarily easy)
- Improve reordering – local and incremental
- Better, multi-level preconditioners
  - easier to update
  - use underlying structure – physics, interpolation
- Test on several realistic materials
- For metallic systems need something for (nearly) dense matrices
  - Fast matrix vector products possible?
  - Representation?

Good Reading

- Ahuja, Clark, de Sturler, Ceperley, and Kim, *Improved Scaling for Quantum Monte Carlo on Insulators*, SIAM Journal on Scientific Computing 33(4), 1837-1859, 2011
- www.math.vt.edu/people/sturler/class_homepages/5485_00000.html
- Iterative Krylov Methods for Large Linear Systems, van der Vorst, Cambridge Univ. Press
- Iterative Methods for Sparse Linear Systems, Yousef Saad, SIAM