Highly Parallel Solver for Multi-scale Parquet Quantum Modeling of Strongly Correlated Materials*

S. Yang[1], E. D'Azevedo[2], H. Fotso[1], M. Jarrell[1], J. Liu[1], T. Maier[2], C. Sen[2], and K. Tomko[3]

[1] University of Cincinnati, Cincinnati, Ohio, 45221, USA
[2] Oak Ridge National Laboratory, Oak Ridge, Tennessee, 37831, USA
[3] Ohio Supercomputer Center, Columbus, Ohio, 43212, USA

Abstract. The parquet formalism to calculate the two-particle Green's functions of large systems requires the solution of a large, sparse, complex system of quadratic equations. If N_f Matsubara frequencies are used for a system of size N_c, and Newton's method is used to solve the nonlinear system, the Jacobian system has $O(8{N_t}^3)$ variables and $O(40{N_t}^4)$ complex entries where $N_t = N_c N_f$. For $N_t = 256$, the nonlinear system has over 134 million degrees of freedom and the sparse Jacobian will require over 2.7 TBytes of memory. The Jacobian is too large to store but the matrix-vector products can be computed directly. We are developing a highly scalable parallel solver that uses both OpenMP and MPI to exploit the multicore nodes. We present initial scalability results on the IBM BlueGene/P and IBM Opteron cluster that suggests the code can be scaled to solve larger problems with $N_t \geq 512$.

1 Introduction

The two-dimensional Hubbard model has been accepted in the community as a minimum model to study the high-temperature superconducting cuprates. An adequate solution of this model is extremely challenging, especially in the interesting parameter regime of intermediate to strong coupling, where the Coulomb repulsion between electrons is of the same order or stronger than the electronic kinetic energy. Despite some recent success, our understanding of the properties of this model in this regime is therefore still limited.

* This research is sponsored by the Office of Advanced Scientific Computing Research; U.S. Department of Energy. The work was performed at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725. This project was supported by SciDAC grant DE-FC02-06ER25792. S. Yang and H. Fotso were supported by NSF PIRE grant OISE-0730290. This research used resources of the Center for Computational Sciences at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. This work was supported in part by an allocation of computing time from the Ohio Supercomputer Center.

The Hubbard Hamiltonian on a square lattice is written as

$$H = \sum_{\langle i,j \rangle} (t_{i,j} - \mu) c^{\dagger}_{i,\sigma} c_{j,\sigma} + U \sum_{i} n_{i,\uparrow} n_{i,\downarrow} \tag{1}$$

where $c^{\dagger}_{i,\sigma}$ ($c_{i,\sigma}$) creates (destroys) an electron with spin σ on site i and $n_{i,\sigma} = c^{\dagger}_{i,\sigma} c_{i,\sigma}$ is the corresponding occupation operator. The first term describes the hopping of electrons between sites i and j and the second term stands for the Coulomb repulsion U two electrons feel when residing on the same site.

Perturbation theory provides useful results only in the limits of either weak or strong coupling, but fails in the interesting regime of intermediate coupling. Methods employing resummations of Feynman diagrams to infinite order are usually biased due to the particular choice of diagrams. Exact methods such as Quantum Monte Carlo (QMC) and exact diagonalization are restricted to relatively small system size due to the computational complexity. In principle, one can imagine carrying out an analytical calculation which includes all the Feynmann diagrams. This would theoretically yield exact results on the single-particle and two-particle levels if one manages to include all relevant diagrammatic equations up to that level.

The parquet formalism is based on a two-particle self-consistent theory, where all the relevant physical conservation rules are preserved as well as the crossing symmetries at two-particle level are obeyed. It is based on the following four diagrammatic equations. To simplify the formalism, we consider systems that preserve the spin $SU(2)$ symmetry. We denote $k \equiv (\mathbf{k}, i\omega_n)$ and $v \equiv (\mathbf{k}, iv_n)$ with $\omega_n = (2n+1)\pi T$ and $v_n = 2n\pi T$. In the following, r is used as a general label for $r = d, m, s, t$ which label the particle-hole density and magnetic channels (d and m), and the particle-particle singlet and triplet channels (s and t). All indexes are in modulo arithmetic so that a negative index such as $-k$ is equal to $\mod(N_t - k, N_t)$.

– The Dyson equation

$$G^{-1}(k) = G_0^{-1}(k) + \Sigma(k) \tag{2}$$

– The Schwinger-Dyson equation

$$\Sigma(k) = -\left(\frac{T}{N}\right)^2 \frac{U}{2} \sum_{k',v} G(-k'+v) G(k') G(-k+v) F_s(v)_{-k',-k}, \tag{3}$$

– The Bethe Salpeter equation

$$F_r(v)_{k,k'} = \Gamma_r(v)_{k,k'} + \sum_{k''} F_r(v)_{k,k''} \chi^0_r(v)_{k''} \Gamma_r(v)_{k'',k'} \tag{4}$$

for $r = d, m, s, t$ and with

$$\begin{aligned} \chi^0_r(v)_{k''} &= G\left(k''\right) G\left(k'' + v\right), \quad r = d, m \\ \chi^0_r(v)_{k''} &= -\frac{1}{2} G\left(k''\right) G\left(k'' + v\right), \quad r = s, t \end{aligned} \tag{5}$$

– The Parquet equation

$$
\begin{aligned}
\Gamma_d^{ph}(v)_{k,k'} &= \Lambda_d^{ph}(v)_{k,k'} - \frac{1}{2}(\Phi_d + 3\Phi_m)(k-k')_{k'+v,k'} + \\
&\quad \frac{1}{2}(\Psi_s + 3\Psi_t)(k+k'+v)_{-k',-(k'+v)} \\
\Gamma_m^{ph}(v)_{k,k'} &= \Lambda_m^{ph}(v)_{k,k'} - \frac{1}{2}(\Phi_d - \Phi_m)(k-k')_{k'+v,k'} - \\
&\quad \frac{1}{2}(\Psi_s - \Psi_t)(k+k'+v)_{-k',-(k'+v)} \\
\Gamma_s^{pp}(v)_{k,k'} &= \Lambda_s^{pp}(v)_{k,k'} + \frac{1}{2}(\Phi_d - 3\Phi_m)(k-k')_{k'+v,-k} + \\
&\quad \frac{1}{2}(\Phi_d - 3\Phi_m)(k+k'+v)_{-k',-k} \\
\Gamma_t^{pp}(v)_{k,k'} &= \Lambda_t^{pp}(v)_{k,k'} + \frac{1}{2}(\Phi_d + \Phi_m)(k-k')_{k'+v,-k} - \\
&\quad \frac{1}{2}(\Phi_d + \Phi_m)(k+k'+v)_{-k',-k}
\end{aligned}
\tag{6}
$$

with

$$
\begin{aligned}
\Phi_r(v)_{k,k'} &= \sum_{k''} F_r(v)_{k,k''}\, \chi_r^0(v)_{k''}\, \Gamma_r(v)_{k'',k'}, \quad r = d, m \\
\Psi_r(v)_{k,k'} &= \sum_{k''} F_r(v)_{k,k''}\, \chi_r^0(v)_{k''}\, \Gamma_r(v)_{k'',k'}, \quad r = s, t
\end{aligned}
\tag{7}
$$

where for $r = d, m$, it is labeled as Φ, while for $r = s, t$, labeled as Ψ.

The parquet equations are derived from enforcing two-particle crossing symmetries, while the other equations are necessary for a conserving approximation. The above set of equations, however, does not form a closed loop because the fully irreducible vertices, Λ_r, are not determined internally. When Λ_r is replaced by the bare interaction, one obtains the parquet approximation. One can also imagine taking the results of an exact calculation for a small system size, using e.g., exact diagonalization or QMC methods, as an input for Λ_r in the parquet equations. In this case, one obtains the full multi-scale parquet formalism, that has been discussed in [1].

2 Algorithm

The parquet equations (2) to (7) form a system of nonlinear equations. The strength and spatial range of correlations depends on the ratio between the Coulomb repulsion U and temperature T and the nonlinear system becomes increasingly difficult to solve for large U. Each entity such as $\Gamma_t^{pp}(v)_{k,k'}$ or $F_r(v)_{k_1,k_2}$ can be discretized and represented as three-index $N_t \times N_t \times N_t$ arrays $\Gamma_t(k,k',v)$ and $F_r(k_1,k_2,v)$. A simple algorithm is to perform fixed-point

iteration by freezing some of the unknowns to solve a subset of the nonlinear equations. Starting from an initial guess, we can solve the Bethe Salpeter equation (4) to generate $F_r(v)_{k_1,k_2}$. The Φ_r and Ψ_r values can be updated using the new F_r values using (7). The new $\Gamma_r(v)_{k,k'}$ values are computed by the Parquet equations (6). One can iteratively solve (4) and (6) until convergence, then update the Dyson equation and self energy in (2) and (3). Although this method is efficient and simple to implement, the iterations may become unstable for large values of $U \geq 18$.

A careful examination of the system shows that the Bethe Salpeter equations can be written as

$$\begin{aligned} F_r(v)_{k,k'} &= \Gamma_r(v)_{k,k'} + \Phi_r(v)_{k,k'}, \quad r = d, m \\ F_r(v)_{k,k'} &= \Gamma_r(v)_{k,k'} + \Psi_r(v)_{k,k'}, \quad r = s, t \end{aligned} \tag{8}$$

so that the Bethe Salpeter equations and the Parquet equations can be written as linear expressions in F_r, Γ_r, Φ_r and Ψ_r. The nonlinearity comes mainly from the Φ_r and Ψ_r variables that are products of just $F_r(v)$ and $\Gamma_r(v)$; therefore (4) to (7) form a complex system of affine quadratic equations.

The literature is sparse on methods for solving a general system of quadratic equations. The solution of a system of quadratic equations using Newton's method has been analyzed in [2]. It shows that multiple solutions are possible and the Jacobian matrix may be exactly singular if it is evaluated at a midpoint of two solutions. Cohen and Tomasi [3] have considered the solution of a special case of a system of homogeneous bilinear equations. Their results show the problem is related to solving the generalized eigenvalue problem. Bouaricha and Schnabel [4, 5, 6] have considered an extension of the Newton's method to solve $F(x) = 0$ by including a low rank tensor approximation of higher derivatives

$$M(x_c + d) = F(x_c) + F'(x_c)d + \frac{1}{2}T_c dd, \quad F : \mathbb{R}^n \rightarrow \mathbb{R}^m \ , \tag{9}$$

where T_c is a three index tensor object formed by interpolation of past Newton steps. The tensor T_c is not the second derivative of $F(x)$ but is chosen to be a sum of p rank-one tensor objects. Equation (9) may be viewed as a particular system of quadratic equations. Ultimately, Newton's method or Levenberg-Marquardt method is used to solve (9) in a least squares sense as a smaller system of p quadratic equations. The analysis of quadratic matrix equation $AX^2 + BX + C = 0$ in $n \times n$ matrices and the connection to quadratic eigenvalue problem $(\lambda^2 A + \lambda B + C)x = 0$ have been considered by Higham and Kim [7]. The authors used Newton's method with exact line searches [8] to solve the quadratic matrix equations.

We have developed a parallel solver using Newton's method with line search to solve the complex system of biaffine quadratic equations. The unknowns are eight $N_t \times N_t \times N_t$ arrays for F_d, F_m, F_s, F_t, Γ_d, Γ_m, Γ_s, and Γ_t. The complicated vertex rotations to enforce crossing symmetries in the Parquet equations (6) can be viewed as permutation operations on a long vector of length ${N_t}^3$. The permutation is implemented using the Message Passing Interface (MPI)

all-to-all communication primitive. This operation places high demands on the communication network and is one of the most time consuming kernels. The sparse Jacobian matrix of the bilinear quadratic equations can be analytically computed but it has $O(40{N_t}^4)$ nonzeros. For problems of interest $N_t \geq 256$, the sparse Jacobian matrix is too costly to store in memory. For example, if $N_t = 256$ then there are over 134 million degrees of freedom and explicit storage of the sparse Jacobian still requires over 2.7 TBytes of memory and for a larger case of $N_t = 512$, there are over 1073 million degrees of freedom and storage of the Jacobian requires over 44 TBytes of memory! For this application, we have found that computing the matrix-vector operation by finite differences introduces an unacceptably high error due to numerical cancellation. Instead the action of matrix-vector multiply is computed analytically without explicit formation of the Jacobian. The Φ and Ψ expressions in (7) are simple products of F_r and Γ_r. The interaction of derivatives of Φ_r or Ψ_r with respect to F_r or Γ_r can be computed as dense matrix matrix products of N_t by N_t matrices. The entries in the Jacobian matrix consist of terms such as

$$\frac{\partial \Phi_r(v)_{k,k'}}{\partial F_r(v)_{k,k''}} = \chi_r^0(v)_{k''} \Gamma_r(v)_{k'',k'} \text{ or } \quad \frac{\partial \Phi_r(v)_{k,k'}}{\partial \Gamma_r(v)_{k'',k'}} = F_r(v)_{k,k''} \chi_r^0(v)_{k''} \ . \quad (10)$$

The large sparse Jacobian system is solved using the BICGSTAB(L=2) [9] Krylov iterative method[4] without preconditioning. Preconditioning using simple Jacobian diagonal scaling is not effective for this problem.

3 Numerical Experiments

The code has been ported to several parallel machines, including the International Business Machine (IBM) Opteron cluster (Glenn) at the Ohio Supercomputer Center (OSC), the Sun Constellation Linux cluster (Ranger) at the Texas Advanced Computing Center (TACC) and the IBM BlueGene/P (Eugene) at the National Center for Computational Sciences (NCCS) at the Oak Ridge National Laboratory (ORNL).

We present performance results obtained on Eugene and Glenn. Eugene is a 27 TFlops/s IBM Blue Gene/P System operated by the NCCS. Eugene consists of 2048 850 Mhz IBM quad core 450d PowerPC processors and 2 GBytes of memory per each node. Eugene has 64 I/O nodes or one I/O node for every 32 compute nodes. Glenn is a 22 TFlops/s IBM Cluster 1350 operated by OSC. It consists of 877 nodes each with two 2.6 Ghz Advanced Micro Devices (AMD) Opteron Dual core processors and 8 GBytes of memory and an additional 88 nodes with four 2.6 Ghz Dual core Opterons and 64 GBytes of memory. In total there are 4212 cores connected by a 10 Gbps Infiniband interconnect.

Parallel jobs can be launched on the Blue Gene/P using three modes:

[4] The code for bicgstab2 is available at `http://www.math.uu.nl/people/vorst/zbcg2.f90`

- in Symmetrical Multiprocessor (SMP) mode, one MPI task is spawned on each node to use the full 2 GBytes of memory. The application may use OpenMP[5] or parallel Engineering Scientific Subroutine Library (ESSL) to utilize all four cores.
- in Virtual Node (VN) mode, four MPI tasks are spawned on each node to utilize all four cores. However, each MPI task has access to only a single core and 512 MBytes of memory.
- in Dual mode, two MPI tasks are spawned and each task has access to two cores and 1 GBytes of memory.

Similarly, on the IBM Opteron cluster, the number of MPI tasks and threads per task can be specified. The MP version of AMD Core Math Library (ACML) was used for the Basic Linear Algebra Subroutine (BLAS) calls. Note that on Eugene, the default stack size for threads created by OpenMP was configured to be 4 MBytes. Large automatic arrays that are allocated on the stack may exceed this limit and lead to a program crash. The code avoids this problem by allocating large temporary arrays on the heap. The application code was compiled using `mpixlf95_r` with `-qstrict -O3 -qsmp=omp -lbgesslsmp -lbgessl` as the optimization options.

The application was tested using a low value of $U = 4$. It took three outer iterations in computing the self energy, each inner iteration required three Newton iterations. The Jacobian system required less than 60 iterations in BICGSTAB(L=2). We expect overall memory requirements in the iterative solver to scale as the number of degrees of freedom $O({N_t}^3)$ and total work scales as $O({N_t}^4)$ in computing matrix-vector multiplies.

Tables 1 and 2 show the run times for $N_t = 256$ on Eugene and Glenn respectively and Tables 3 and 4 show the run times for $N_t = 512$. The first two columns of Table 1 show that there is some extra overhead in creating and managing threads since 256 cores are used in both cases. The OpenMP results on Eugene show good (strong) scalability since the overall time decreased from 668s on 64 nodes to 367s on 128 nodes to 191s on 256 nodes. The data also suggest the time taken in MPI all-to-all communication decreases as more cores are added but is greater on Glenn than Eugene. Performance of MPI all-to-all is dependent on the interconnect hardware and on the vendor's implementation of MPI library. Note that the majority of run time was spent in computing matrix-vector multiplies called by the BICGSTAB linear solver on Eugene but was less dominant on Glenn. Time spent in dense matrix multiplies (zgemm) and MPI all-to-all communication accounts for over 70% of total time. Table 3 shows hybrid parallelization using OpenMP and MPI is effective in using many cores. Figure 1 shows the decrease in run time as threads and cores are increased for a fixed number of MPI tasks. Multi-threading performs well on both systems. The overall run time decreased almost linearly from 4370s to 1126s (94% parallel efficiency) as we increase the number of cores from 512 cores (128 MPI tasks x 4 threads per task) to 2048 cores (512 MPI tasks x 4 threads per task). On the IBM Opteron Cluster we gathered hardware performance counter profile data

[5] Include "export `OMP_NUM_THREADS=4`" in batch script.

using the *oprofile* performance monitoring utility. The resulting performance data is included in the last few rows of Tables 2 and 4. From these results we see that many configurations performed quite well overall, achieving more than one TFlops/s. In particular, in the case of $N_t = 512$ on 512 cores (256 MPI tasks $\times$ 2 threads), the measured performance was 1.69 TFlops/s, or 63% of peak utilizing 12% of a 22 TFlops/s system.

4 Summary

We have described the development of a parallel solver for multi-scale parquet quantum modeling of highly correlated materials. The code uses Newton's method with line search to solve the large system of affine quadratic equations. The large sparse Jacobian is too large to store and the action of matrix-vector multiply is computed analytically. Performance results on the BlueGene/P and Opeteron cluster suggest hybrid OpenMP and MPI programming technique can effectively use large numbers of cores to solve problems with a billion degrees of freedom.

Future development will focus on more sophisticated continuation method for generating good initial guesses and the exploration of effective preconditioners and iterative methods for solving the Jacobian system.

Table 1. Run time for $N_t = 256$ on various configurations on the BlueGene/P.

	Cores (MPI Tasks $\times$ Threads per Task)				
	256 (256 $\times$ 1)	256 (64 $\times$ 4)	512 (256 $\times$ 2)	512 (128 $\times$ 4)	1024 (256 $\times$ 4)
total	584s	668s	358s	367s	191s
bicgstab	557s	630s	335s	344s	175s
matvec	499s	549s	301s	304s	154s
zgemm	352s	391s	194s	194s	98s
mpi_alltoall	78s	74s	70s	69s	37s

Table 2. Run time for $N_t = 256$ on various configurations on the Opteron Cluster (Glenn).

	Cores (MPI Tasks × Threads per Task)				
	256 (256 × 1)	256 (64 × 4)	512 (256 × 2)	512 (128 × 4)	1024 (256 × 4)
total	542s	724s	340s	277s	208s
bicgstab	510s	679s	316s	257s	173s
matvec	449s	570s	272s	204s	166s
zgemm	203s	213s	106s	108s	55s
scatter	218s	310s	151s	68s	102s
total memory usage	42 GB	35 GB	44 GB	37 GB	46 GB
aggregate Mem BW	331 GB/s	237 GB/s	502 GB/s	574 GB/s	549 GB/s
aggregate FP Perf.	657 Gflop/s	498 Gflop/s	995 Gflops/s	1132 Gflops/s	1097 Gflops/s

Table 3. Performance for $N_t = 512$ on various configurations on the BlueGene/P.

	Cores (MPI Tasks × Threads per Task)		
	512 (256 × 2)	512 (128 × 4)	2048 (512 × 4)
total	4213s	4370s	1126s
bicgstab	4012s	4120s	1038s
matvec	3771s	3869s	975s
zgemm	2889s	2964s	733s
mpi_alltoall	586s	583s	165s

Table 4. Run time for $N_t = 512$ on various configurations on the OSC Opteron Cluster (Glenn).

	Cores (MPI Tasks × Threads per Task)		
	512 (512 × 1)	512 (256 × 2)	512 (128 × 4)
total	7282s	3351s	4116s
bicgstab	6505s	3175s	3863s
matvec	4664s	2815s	3395s
zgemm	1612s	1625s	1633s
scatter	2766s	987s	1577s
total memory usage	305 GB	267 GB	259 GB
aggregate Mem BW	298 GB/s	626 GB/s	512 GB/s
aggregate FP Perf.	779 Gflop/s	1689 Gflop/s	1380 Gflops/s

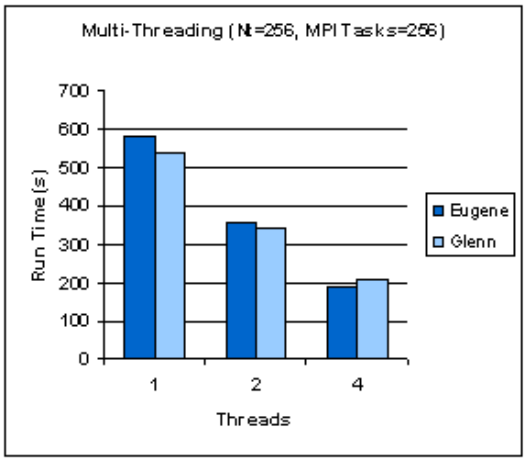

Fig. 1. Run time for $N_t = 256$ with increasing threads per MPI task on the BlueGene/P and the IBM Opteron Cluster.

References

[1] Slezak, C., Jarrell, M., Maier, T., Deisz, J.: Multi-scale extensions to quantum cluster methods for strongly correlated electron systems. arXiv:cond-mat/0603421v2 [cond-mat.str-el] (2006)

[2] Makarov, Y.V., Hill, D.J., Hiskens, I.A.: Properties of quadratic equations and their application power system analysis. Electrical Power and Energy Systems **22** 313–323 (2000)

[3] Cohen, S., Tomasi, C.: Systems of bilinear equations. Technical Report CS-TR-97-1588, Computer Science Department, Stanford University, Stanford, CA 94305 (1997)

[4] Schnabel, R.B., Frank, P.D.: Tensor methods for nonlinear equations. SIAM J. Numer. Anal. 815–843 (1984)

[5] Bouaricha, A., Schnabel, R.B.: Algorithm 768: TENSOLVE: A software package for solving systems of nonlinear equations and nonlinear least-squares problems using tensor methods. ACM Trans. Math. Softw. **23**(2) 174–195 (1977)

[6] Bouaricha, A., Schnabel, R.B.: Tensor methods for large sparse systems of nonlinear equations. Math. Program. **82** 377–400 (1998)

[7] Higham, N.J., Kim, H.M.: Numerical analysis of a quadratic matrix equation. IMA Journal of Numerical Analysis **20** 499–519 (2000)

[8] Higham, N.J., Kim, H.M.: Solving a quadratic matrix equation by Newton's method with exact line searches. SIAM J. Matrix Anal. Appl. **23**(2) 303–316 (2001)

[9] Sleijpen, G.L.G., Fokkema, D.: BiCGstab(L) for linear equations involving non-symmetric matrices with complex spectrum. Electronic Transactions on Numerical Analysis **1** 11–32 (1993)