# Many-Body Electronic Correlation Energy using Krylov Subspace Linear Solvers

Shikhar Shah
*Computational Science and Engineering*
*Georgia Institute of Technology*
Atlanta, Georgia, USA
sshikhar@gatech.edu

Boqin Zhang
*Civil and Environmental Engineering*
*Georgia Institute of Technology*
Atlanta, Georgia, USA
bzhang376@gatech.edu

Hua Huang
*Computational Science and Engineering*
*Georgia Institute of Technology*
Atlanta, Georgia, USA
huangh223@gatech.edu

John E. Pask
*Physics Division*
*Lawrence Livermore National Laboratory*
Livermore, California, USA
pask1@llnl.gov

Phanish Suryanarayana
*Civil and Environmental Engineering*
*Georgia Institute of Technology*
Atlanta, Georgia, USA
phanish.suryanarayana@ce.gatech.edu

Edmond Chow
*Computational Science and Engineering*
*Georgia Institute of Technology*
Atlanta, Georgia, USA
echow@cc.gatech.edu

***Abstract*—This paper presents the formulation and implementation of a high performance algorithm to compute the many-body electronic correlation energy via the random-phase approximation within density functional theory. Our approach circumvents computational inefficiencies inherent in direct approaches which exhibit quartic scaling with respect to system size. Our formulation requires solving block linear systems whose coefficient matrices are complex symmetric; these systems are of widely-varying numerical difficulty. We develop a short-term recurrence block Krylov subspace solver for these systems and leverage a dynamic block size selection to mitigate load imbalances. This selection balances the increased cost per linear solver iteration with a reduction in the number of iterations for slowly-converging systems. Numerical experiments show that our implementation exhibits good parallel scalability, achieves faster solution times than direct approaches on even the smallest chemical system tested, and scales to larger systems and processor counts due to its cubic scaling and greater computational locality.**

***Index Terms*—computational chemistry, density functional theory, iterative methods, linear systems**

## I. Introduction

Density functional theory (DFT) is a popular method in computational chemistry to investigate the electronic structure of chemical systems [1]–[3]. DFT calculations require the formulation of an exchange-correlation functional to approximate the energy of electron-electron interactions. There is a hierarchy of approximations to this functional [4], the simplest and least accurate of which have allowed DFT calculations to scale to thousands of atoms and more [5]; however, these approximations have well-known deficiencies. One of the most accurate and sophisticated approximations to the exchange-correlation functional uses a technique known as the random-phase approximation (RPA) to compute the electronic correlation energy. RPA is specifically useful for small-gap and metallic systems where other exchange-correlation functionals readily break down [6], and for systems dominated by long-range electron correlation [7]. Furthermore, in many cases RPA finds chemical quantities of interest to higher accuracy [8], [9] and it is considered the "gold standard" for determining adsorption energies [10].

Direct approaches for computing the RPA correlation energy exhibit quartic scaling, either in the number of finite difference grid points or the number of plane waves, due to the explicit construction of the irreducible polarizability matrix $\chi^0$. This results in RPA being prohibitively expensive for large chemical systems. Many software packages have been proposed to perform operations with $\chi^0$ in a computationally tractable way, either using a plane wave basis set [11]–[13] or an atom-centered basis set [14]–[16]. VASP and BerkeleyGW both exhibit sub-quartic scaling, but require both the occupied *and* unoccupied orbitals from a prior DFT calculation. For large chemical systems, the number of unoccupied orbitals becomes large and hampers the scaling of such methods. WEST [17] and SternheimerGW [18] employ perturbation theory and thus only require the occupied orbitals; furthermore, their key computational kernel involves solving a set of block linear systems. However, both solve these systems in a non-block fashion. Additionally, many of these software packages operate in the reciprocal (i.e., Fourier) space. Real space approaches have the advantage of avoiding Fourier and inverse Fourier transforms which can oftentimes be difficult to parallelize [19]. Real space approaches are also more amenable than reciprocal space approaches to Dirichlet boundary conditions (for simulating molecules, wires, and surfaces) [20]. A high performance computation of the RPA correlation energy should utilize the inherent parallelism available in real space approaches and should require the prior computation of *only* the occupied orbitals.

In this work, we present a real space formulation of RPA which is amenable to large chemical systems and parallelizes efficiently to a large number of processors relative to existing direct approaches. Our formulation requires only the occupied orbitals and scales cubically with the number of finite differ-

SC24, November 17–22, 2024, Atlanta, Georgia, USA
979-8-3503-5291-7/24/$31.00 ©2024 IEEE

ence grid points. In addition, the bulk of the computational cost is solving a set of block linear systems; in each block linear system, the coefficient matrix is the sum of a Hermitian matrix and a complex constant multiple of the identity matrix. These block linear systems are solved in parallel using block Krylov subspace methods with little loss of parallel efficiency. Additionally, these block linear systems are of widely-varying numerical difficulty; thus, selecting the block size *a priori* is difficult. Krylov subspace methods with no short-term recurrence become the computational bottleneck when a large number of iterations is needed to converge particularly difficult linear systems.

To solve the required linear systems, we develop a new short-term recurrence block Krylov subspace method. This novel linear solver is augmented with a dynamic block size selection algorithm allowing each processor to independently select the appropriate block size for the difficulty of linear systems it is required to solve. To the best of the authors' knowledge, this work is the first instance of block linear system solvers used in the computation of the many-body electronic correlation energy, and is the first real space formulation of the RPA correlation energy. We implement our framework in the electronic structure software package SPARC [21] and achieve faster calculation times than direct approaches on even the smallest chemical system tested (8 atoms), with the ability to scale beyond existing direct approaches by virtue of high parallel efficiency.

## II. Background

Computing the RPA correlation energy $E_{\text{RPA}}$ requires evaluating the semi-infinite integral

$$E_{\text{RPA}} = \frac{1}{2\pi}\int_0^\infty \text{Tr}[\ln(I - \nu\chi^0(i\omega)) + \nu\chi^0(i\omega)]\ \mathrm{d}\omega \quad (1)$$

where $\nu = -4\pi(\nabla^2)^{-1}$ is proportional to the inverse of the discrete Laplacian matrix (representing the Coulomb operator) and $\chi^0$ is a matrix known as the irreducible polarizability matrix, which is a function of the integration variable (i.e., frequency) $\omega$ [22]. Note that while $\nu$ and $\chi^0$ are separately Hermitian matrices, their product is non-Hermitian. In real space $\Gamma$-point calculations, $\nu, \chi^0 \in \mathbb{R}^{n_d \times n_d}$ where $n_d$ is the number of finite difference grid points in the computational domain; however, $\nu$ does not need to be explicitly constructed, and instead any method which numerically solves the Poisson equation can directly replace the matrix $(\nabla^2)^{-1}$.

Let $H \in \mathbb{R}^{n_d \times n_d}$ be the Hamiltonian matrix from a Kohn-Sham DFT (KS-DFT) calculation; then, denote the eigenpairs of $H$ by $(\lambda_j, \Psi_j)$, where each eigenvector $\Psi_j$ (also known as an orbital) has an associated *occupation* $g_j$ indicating how many pairs of electrons are contained within that orbital. Then, we write an explicit form for the irreducible polarizability matrix

$$\chi^0(r, r', i\omega) = 2\sum_{m,n=1}^{n_d}(g_m - g_n)\frac{\Psi_m^*(r)\Psi_n(r)\Psi_n^*(r')\Psi_m(r')}{\lambda_m - \lambda_n - i\omega} \quad (2)$$

where $r$ and $r'$ are positions in discretized space and $z^*$ indicates complex conjugation of $z$. A direct computation of $\chi^0$ has complexity $\mathcal{O}(n_d^4)$. This computation requires all orbitals of $H$, both occupied ($g_j = 1$) and unoccupied ($g_j = 0$). In practice, the number of occupied orbitals is roughly half the number of valence electrons in the system, which is usually a very small fraction of $n_d$.

There are two overarching methods for performing linear algebraic operations with the irreducible polarizability matrix. The first method is by directly computing $\chi^0$ via the formula of Adler and Wiser [23], [24], which has a similar form to Equation (2). This method is applied in plane wave codes such as ABINIT [11], VASP [12], and BerkeleyGW [13], and in atom-centered codes such as CP2K [14], TURBOMOLE [15], and FHI-aims [16]. However, as mentioned previously, this requires both the occupied *and* unoccupied orbitals from a DFT calculation, which may be computationally intractable or incur a prohibitively high memory consumption. VASP and BerkeleyGW have simplifications which reduce their complexity to $\mathcal{O}(n_{pw}^3)$ and $\mathcal{O}(n_{pw}^3 \log n_{pw})$, respectively (where $n_{pw}$ is the number of plane waves); the technical details of these simplifications are omitted for brevity.

The second method is by obtaining the product of $\chi^0$ and a vector through density functional perturbation theory (DFPT), which sidesteps the explicit construction of $\chi^0$. This is employed in WEST [17] and SternheimerGW [18], which only require the occupied orbitals and compute the eigenvectors of $\chi^0$ via perturbation theory. This involves solving a set of block linear systems (known as the Sternheimer equations; see Equation (4)), which are solved in a non-block fashion. We take a similar approach to WEST and SternheimerGW but in real space.

In our RPA framework, the integral in Equation (1) is computed via numerical quadrature

$$\int_0^\infty \text{Tr}[f(\nu\chi^0(i\omega))]\ \mathrm{d}\omega \approx \sum_{k=1}^{\ell} w_k \,\text{Tr}[f(\nu\chi^0(i\omega_k))] \quad (3)$$

for $\ell$ integration quadrature points $\omega_k$ and weights $w_k$ [25], [26]; however, there are several ways to approximate the integrand (a functional trace) at each quadrature point. Experimentally, the spectrum of $\nu\chi^0$ decays rapidly to zero [27], as shown in Figure 1; thus, one approach is to compute a subset of the largest magnitude eigenvalues of $\nu\chi^0$ via an iterative method. Another approach is to use Lanczos or Arnoldi quadrature to directly estimate the diagonal entries of the matrix function $f(\nu\chi^0(i\omega))$ [28]. A third approach is to use the Hutchinson stochastic trace estimator [29]. All of these approaches require the ability to apply $\nu\chi^0$ to a vector $v$. The product $\chi^0 v$ is computed via a two-step approach [17], [30]. First, we solve the Sternheimer equations

$$(H - \lambda_j I + i\omega_k I)y_j = -v \odot \Psi_j,\ j = 1, 2, \ldots, n_s \quad (4)$$

where $n_s$ is the number of occupied orbitals and $\odot$ is the

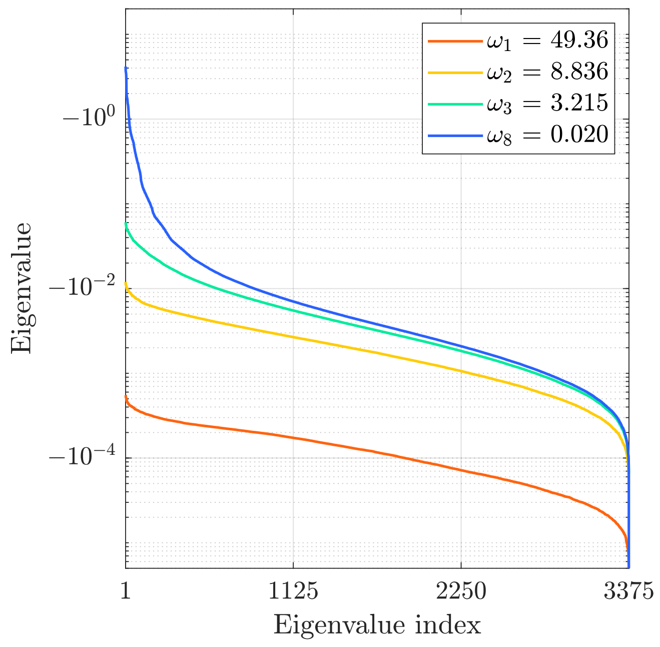

Fig. 1. Spectrum of $\nu\chi^0$ for $\text{Si}_8$ at a variety of integration quadrature points. In all cases the spectrum rapidly decays to 0; additionally, the lowest magnitude portion of the spectrum converges to a fixed spectrum as $\omega \to 0$.

Hadamard product. Then, we compute the product $\chi^0 v$ via

$$\chi^0 v = 4\,\text{Re}\left(\sum_{j=1}^{n_s} \Psi_j^* \odot y_j\right). \tag{5}$$

Applying $\nu$ to the product $\chi^0 v$ only requires a single linear solve with a finite-difference discretization of the Laplacian matrix $\nabla^2$. Note that the eigenpairs $(\lambda_j, \Psi_j), j = 1, 2, \ldots, n_s$ are computed in a prior KS-DFT calculation and they correspond to the minimum eigenvalues (lowest-energy orbitals) of $H$. The computed $E_{\text{RPA}}$ then replaces the correlation energy computed in this prior KS-DFT calculation where some other exchange-correlation functional was chosen.

For all of the methods for approximating the trace in Equation (3) mentioned earlier, we need to apply the same matrix $\chi^0$ to multiple vectors. The two-step procedure outlined in Equations (4) and (5) provides a way to compute the product $\chi^0 V$ for a *block* of $n_v$ vectors $V$, transforming Equation (4) into a set of linear systems with multiple right-hand side vectors by replacing $v$ with $V$ (and replacing $y_j$ with a block of vectors $Y_j$). The full set of Sternheimer equations for a *single* matrix multiplication of the form $\chi^0(i\omega_k)V$ is

$$\begin{aligned}(H - \lambda_1 I + i\omega_k I)Y_1 &= -V \odot \Psi_1\\ (H - \lambda_2 I + i\omega_k I)Y_2 &= -V \odot \Psi_2\\ \vdots \quad &= \quad \vdots\\ (H - \lambda_{n_s} I + i\omega_k I)Y_{n_s} &= -V \odot \Psi_{n_s}\end{aligned} \tag{6}$$

where each line of Equation (6) has a total of $n_v$ linear systems to solve. It is simple to extend Equation (5) to block form, as well.

Solving linear systems with multiple right-hand side vectors is a very rich topic in numerical linear algebra. The two main approaches are block methods, where $AX = B$ is solved by treating all right-hand side vectors simultaneously [31], and seed methods, where a "seed system" $Ax^{(i)} = b^{(i)}$ is solved by itself [32]. In seed methods, we extract an orthogonal basis to the Krylov subspace while solving the seed system and reuse this Krylov subspace to project the remaining linear systems. This process is then iterated for another seed system until all linear systems are sufficiently converged.

Block methods require having access to all the right-hand side vectors *a priori*, and may require deflation if the residual vectors become linearly dependent. There is some theory on block Krylov subspace methods indicating that the theoretical convergence rate improves when using a larger block size [33]. On the other hand, seed methods only require the right-hand side vector for the seed system to be available at first. However, reusing the seed Krylov subspace to project the remaining linear systems may result in slow convergence if the coefficient matrix is highly non-normal or if the right-hand side vectors are unrelated. We expect the right-hand side vectors to be effectively random in the Sternheimer equations, so seed methods are not considered in this work. Instead, we solve each linear system in Equation (6) using a block Krylov subspace method.

Algorithm 1 shows the overall procedure we employ to numerically compute the RPA correlation energy in Equation (1). By avoiding the explicit construction of $\chi^0$ in Equation (2), we circumvent quartic scaling; instead, we achieve cubic scaling by computing $E_k$ with cubic complexity in $n_d$. The specific choice of how to perform line 3 is the bulk of our novel work and is described in detail in Section III.

**Algorithm 1** Computing the RPA correlation energy via many-body perturbation theory

```
1: $E_{\text{RPA}} \leftarrow 0$
2: for k = 1, 2, ..., $\ell$ do
3:     $E_k \leftarrow$ approximation of $\text{Tr}[\ln(I - \nu\chi^0(i\omega_k)) + \nu\chi^0(i\omega_k)]$, requiring instances of solving Equation (6) and performing Poisson solves
4:     $E_{\text{RPA}} \leftarrow E_{\text{RPA}} + w_k E_k / 2\pi$
5: end for
```

## III. High Performance Formulation of RPA

In this section, we present a new high performance method for computing the RPA correlation energy following the general structure of Algorithm 1. Our method is extremely parallelizable and is intended for large chemical systems on a large number of processors. Additionally, we present numerical optimizations for our method which result in convergence acceleration and circumvent issues of load imbalance.

### A. Subspace Iteration with Polynomial Filtering

From Figure 1, the spectrum of $\nu\chi^0$ decays rapidly to zero independent of $\omega$; consequently, we approximate the trace at each quadrature point in Equation (3) by first computing the smallest (largest in magnitude) $n_{\text{eig}}$ eigenvalues $\mu_j$ of $\nu\chi^0$ using an iterative method. Then, we approximate the trace via

$$\sum_{j=1}^{n_{\text{eig}}}(\ln(1-\mu_j)+\mu_j).$$

The value of $n_{\text{eig}}$ should be chosen as small as possible, but large enough such that the approximate trace of $f(\nu\chi^0)$ we calculate is close to the exact trace. This can be done in many ways (e.g., a fixed percentage of $n_d$), but we show in Section IV-A that we are able to select a relatively small value for $n_{\text{eig}}$ without impacting the ability to achieve chemical accuracy in the RPA correlation energy.

To compute this partial spectrum we use subspace iteration with polynomial filtering, shown in Algorithm 2. In an analogous fashion to the Chebyshev-filtered subspace iteration (CheFSI) [34], we use a Chebyshev polynomial for $p$ to dampen the unwanted part of the spectrum. This accelerates convergence compared to standard subspace iteration (i.e., $p(x) = x$). Note that CheFSI is commonly used for the *nonlinear* eigenvalue problem appearing in KS-DFT; here, we apply a similar technique but for the *linear* eigenvalue problem occurring at each quadrature point.

**Algorithm 2** Subspace iteration with polynomial filtering to compute the eigenvalues of $A$

```
1: Initialize $V_0$
2: for j = 0, 1, 2, ... do
3:     $V_{j+1} \leftarrow p(A)V_j$
4:     $H_s \leftarrow V_{j+1}^H A V_{j+1}$ and $M_s \leftarrow V_{j+1}^H V_{j+1}$
5:     Solve the eigenvalue problem $H_s Q = M_s Q D$
6:     $V_{j+1} \leftarrow V_{j+1} Q$
7: end for
```

As it is currently stated, the eigenvalue problem in line 5 of Algorithm 2 is a *generalized, non-Hermitian eigenvalue problem*, stemming from the fact that $\nu\chi^0$ is non-Hermitian. Since we are uninterested in the *eigenvectors* of $\nu\chi^0$, we apply a similarity transform $\nu^{-1/2}\nu\chi^0\nu^{1/2}$. The resulting matrix $\nu^{1/2}\chi^0\nu^{1/2}$ *is* a Hermitian matrix with the same spectrum as $\nu\chi^0$, and thus converts the eigenvalue problem in line 5 to a *generalized, Hermitian eigenvalue problem*. Here, $\nu^{1/2}$ denotes the matrix square root of $\nu$, which is well-posed since $\nu$ is symmetric positive-definite. Replacing the matrix $A$ in Algorithm 2 with $\nu^{1/2}\chi^0\nu^{1/2}$ gives the overarching procedure we employ. Each application of $\nu^{1/2}\chi^0\nu^{1/2}$ requires two multiplications with the matrix $\nu^{1/2}$, or equivalently, two solves with the matrix $(-\nabla^2)^{-1/2}$, which is done efficiently by exploiting the Kronecker product formulation of the discrete Laplacian [35], [36]. The details of this method are not discussed here; however, the multiplications by $\nu^{1/2}$ contribute only a small fraction of the overall time of multiplying by $\nu^{1/2}\chi^0\nu^{1/2}$. Additionally, as later discussed in Section III-D, the multiplications by $\nu^{1/2}$ do not incur any parallel communication for the type of matrix partitioning we consider in this work.

Each product $AV$ in Algorithm 2 requires solving Equations (4) and (5) for a block of $n_{\text{eig}}$ vectors. As subspace iteration proceeds, columns of $V$ will converge to eigenvectors of $\nu^{1/2}\chi^0\nu^{1/2}$ and $D$ will converge to its spectrum. We terminate the procedure when

$$\frac{\sum_{j=1}^{n_{\text{eig}}}\|AV_j - D_{jj}V_j\|_2}{n_{\text{eig}}\sqrt{\sum_{j=1}^{n_{\text{eig}}} D_{jj}^2}} \le \tau_{\text{SI}} \tag{7}$$

for a user-prescribed subspace iteration tolerance $\tau_{\text{SI}}$.

### B. Short-Term Recurrence Block Krylov Solver

For each multiplication by $\chi^0$, we need to solve a total of $n_{\text{eig}}n_s$ linear systems, each of the form listed in Equation (4). These equations can be grouped together to form $n_s$ *block* linear systems

$$A_{j,k}Y_j = B_j,\ j = 1, 2, \dots, n_s \tag{8}$$

in an analogous fashion to Equation (6), where $B_j = -V \odot \Psi_j \in \mathbb{R}^{n_d \times n_{\text{eig}}}$. We use the convention $\lambda_1 \le \lambda_2 \le ... \le \lambda_{n_s}$ for the lowest $n_s$ eigenvalues of $H$ and $\omega_1 > \omega_2 > \dots > \omega_\ell > 0$ for the $\ell$ integration quadrature points. From this, the spectrum of $A_{j,k}$ is written succinctly as

$$\lambda(A_{j,k}) = \{\lambda(H) - \lambda_j + i\omega_k\} \tag{9}$$

where $\lambda(A)$ is the set of all eigenvalues of $A$. In the case $\omega_k = 0$, we have $0 \in \lambda(A_{j,k})$ and the coefficient matrix is singular; while this does not occur in our case, quadrature points may be very close to 0 leading to ill-conditioning.

The difficulty of the individual linear systems in Equation (8) is dependent on the $(j, k)$ index pair. In the extreme case $(1, 1)$, the coefficient matrix is positive semi-definite (i.e., its spectrum is *not* in the negative real half-plane) and its spectrum is shifted far from the origin on the imaginary axis. Linear systems of this form require fewer iterations to achieve specified tolerances [37]. The other extreme case of $(n_s, \ell)$ results in a coefficient matrix whose spectrum is highly indefinite and has an eigenvalue only a distance $\omega_\ell \approx 0$ from the origin. Linear systems of this form are difficult to converge and may encounter residual stagnation.

Regardless of the $(j, k)$ index pair, the coefficient matrix in Equation (8) is *not Hermitian*, but instead complex symmetric; that is, it satisfies the property $A = A^T$, $A \in \mathbb{C}^{n_d \times n_d}$. GMRES [38] is able to iteratively solve any linear system, including one of this form; however, it becomes computationally expensive as the iteration count grows due to lacking a short-term recurrence in generating the Krylov subspace basis vectors. Numerical methods for complex symmetric matrices have been developed [39], and complex symmetric matrices appear in a wide range of scientific computing contexts, including numerically solving the Helmholtz equation [40] and in machine learning applications [41]. Additionally, the Sternheimer equations appear in vibrational spectra calculations. The specific method we consider here is the conjugate orthogonal conjugate gradient (COCG) method [42]. COCG, derived originally as a non-block method, utilizes the relation

$A = A^T$ to derive a new three-term recurrence for the Krylov subspace basis vectors. However, unlike other methods, COCG does not satisfy an optimality result in the residual or error norms [43]. For numerically difficult linear systems, using a block size $s > 1$ (i.e., solving with multiple right-hand side vectors simultaneously) may improve the convergence behavior [33]; thus, we extend COCG to its block version in Algorithm 3, solving Equation (8).

**Algorithm 3** Block COCG for solving $AY = B$

```
1: Initialize $Y_0$ and $W_0 \leftarrow B - AY_0$
2: $\rho_0 \leftarrow W_0^T W_0$
3: $P_{-1} \leftarrow 0$, $\beta_{-1} \leftarrow 0$
4: for j = 0, 1, 2, ... do
5:     $P_j \leftarrow W_j + P_{j-1}\beta_{j-1}$
6:     $U_j \leftarrow AP_j$
7:     $\mu_j \leftarrow U_j^T P_j$
8:     $\alpha_j \leftarrow \mu_j^{-1}\rho_j$
9:     $Y_{j+1} \leftarrow Y_j + P\alpha_j$
10:    $W_{j+1} \leftarrow W_j - U_j\alpha_j$
11:    $\rho_{j+1} \leftarrow W_{j+1}^T W_{j+1}$
12:    $\beta_{j+1} \leftarrow \rho_j^{-1}\rho_{j+1}$
13: end for
```

Letting $B \in \mathbb{R}^{n_d \times s}$ for a block size $s$, there are three main costs per iteration in block COCG. First, each iteration requires one application of the matrix $A$ to a block of $s$ vectors (line 6). Second, each iteration requires five matrix-matrix multiplications of cost $\mathcal{O}(n_d s^2)$ (lines 5, 7, 9, 10, 11). Finally, each iteration requires two linear solves of cost $\mathcal{O}(s^3)$ (lines 8, 12).

The dominant computational term in Algorithm 3 solely depends on how expensive multiplication by $A$ is relative to $\mathcal{O}(n_d s^2)$ and $\mathcal{O}(s^3)$. In our case, $A$ is the sum of a complex constant multiple of the identity matrix and the KS-DFT Hamiltonian which consists of two main terms: a high-order finite difference discretization of the Laplacian $\nabla^2$ and a sparse outer product of the form $\mathcal{X}\mathcal{X}^H$. Increasing $s$ has the effect of *decreasing* the number of iterations required for convergence. Using a larger $s$ also makes the computation more efficient (to be discussed in Section III-C). Thus, the optimal $s$ balances the reduction in the number of applications of $A$ with the increased matrix-matrix multiplication cost. When the block linear system is difficult to converge (i.e., when its spectrum is highly indefinite and close to the origin), we expect larger block sizes to yield an overall reduction in computational time.

We terminate the procedure in Algorithm 3 when

$$\frac{\|W\|_F}{\|B\|_F} \leq \tau_{\text{Sternheimer}} \tag{10}$$

for a user-prescribed linear solver tolerance $\tau_{\text{Sternheimer}}$, where $\|\cdot\|_F$ denotes the Frobenius norm.

## C. Performance Considerations for the Block Solver

The choice of block size $s$ not only decides the rate of convergence of COCG, but also has a significant impact on the parallel performance of the linear algebra operations used in COCG. With $s = 1$ (the block solver reduced to its non-block version), lines 5, 7, 9, 10, and 11 in Algorithm 3 are reduced from matrix-matrix multiplications (matmult) to matrix-vector multiplications (matvec). For the same convergence criterion, using $s > 1$ usually requires more floating point operations (FLOPs) than using $s = 1$, since increasing $s$ by a factor of $k$ will not reduce the number of iterations required by this same factor $k$. Nevertheless, using a larger block size can still be more computationally efficient. Since lines 5, 6, 7, 9, 10, and 11 in Algorithm 3 contribute most of the FLOPs when $s \ll n$, using $s > 1$ shifts from BLAS-2 matvec to BLAS-3 matmult, and the BLAS-3 matmult can perform more FLOPs per second since matmults have a higher arithmetic intensity (AI).

We further note that the partially-matrix-free multiplication $U_j \leftarrow AP_j$ contains both matmult and matvec operations if $s > 1$. For the sparse projection matrix $\mathcal{X}\mathcal{X}^H$, we use sparse-dense matrix-matrix multiplication for a higher AI. The high-order finite difference discretization of the Laplacian operator is applied to each of the input vectors using stencil operations (the matrix-free part in $U_j \leftarrow AP_j$). To achieve a higher AI, we apply the stencil operation to one input vector at a time instead of applying it to multiple input vectors simultaneously. Consider the classical fast-slow memory performance model where the fast memory has a capacity of $C$ words. The finite difference stencil used in our software is a six-axis $(6r+1)$-point stencil with radius $r$. The stencil computation for an $m \times n \times k$ block of grid points (the output domain) requires

$$mnk + 2r(mn + mk + nk)$$

grid points from the input domain. We need to fit both the input and output domains in fast memory, which requires

$$2mnk + 2r(mn + mk + nk) \leq C$$

and gives an AI of

$$I_1(m,n,k) = \frac{2(6r+1)mnk}{2mnk + 2r(mn + mk + nk)}. \tag{11}$$

When $m = n = k$, $I_1(m,n,k)$ is maximized and we have

$$\max(I_1(m)) = \frac{(6r+1)m}{m+3r}.$$

If we apply the stencil to $s$ input vectors simultaneously, we need to satisfy

$$s(2mnk + 2r(mn + mk + nk)) \leq C$$

and we obtain an AI of

$$I_s(m,n,k) = \frac{2(6r+1)mnks}{(2mnk + 2r(mn + mk + nk))s}. \tag{12}$$

$I_s(m,n,k)$ is also maximized when $m = n = k$, and we have

$$\max(I_s(m)) = \frac{(6r+1)m}{m+3r} = \max(I_1(m)).$$

$\max(I_1(m))$ increases monotonically with respect to $m$, while the largest available $m$ for $I_s(m)$ is only $1/s$ of the largest available $m$ for $I_1(m)$ due to the constraint of fast memory size. Therefore, applying the stencil operation to one input vector at a time can achieve better performance than applying the stencil operation to multiple vectors simultaneously.

## *D. Parallelization Structure*

Each time we perform a multiplication by $\chi^0$, we need to solve exactly $n_{\text{eig}} n_s$ linear systems that can be grouped into $n_s$ blocks of $n_{\text{eig}}$ systems each, with a constant coefficient matrix within each block (Equation (8)). Among $p$ processors, each processor needs to be given $n_{\text{eig}} n_s / p$ linear systems to solve. As mentioned in Section III-B, the difficulty of these systems (and thus the number of iterations and time they require to converge) is highly dependent on the $(j, k)$ index pair. For every multiplication by $\chi^0(i\omega_k)$ the $k$ index is fixed across all linear systems; thus, the most drastic impact on the linear system difficulty is due to the $j$ index. Parallelizing across the $n_s$ orbitals results in a large load balancing issue where processors given lower-valued indices $j$ converge their linear systems quickly (as the coefficient matrix is only slightly indefinite), while processors given higher-valued indices $j$ are slow to converge their assigned linear systems. This can be mitigated by assigning a nonconstant number of orbitals to each processor to appropriately balance the workload; however, this partitioning is itself a function of the index $k$ and is thus not considered in this work.

Instead, the simpler idea is to only parallelize across $n_{\text{eig}}$; each processor is assigned $n_{\text{eig}}/p$ eigenvalues and eigenvectors of $\nu^{1/2}\chi^0\nu^{1/2}$ and, for each solve of the Sternheimer equations, is responsible for those right-hand side vectors *for all $n_s$ orbitals*. We only consider $p \leq n_{\text{eig}}$ in this work so no processor will ever be assigned zero eigenvalues. However, this has the unfortunate effect of limiting the maximum block size to $s \leq n_{\text{eig}}/p$. As we later discuss in Section IV-B, this is not practically an issue for the systems tested in this work. Solving the systems one at a time (i.e., $s = 1$) is the standard approach; however, Algorithm 3 lets us leverage a block algorithm to use $s > 1$ in instances where it reduces the overall computation time.

Note that since each processor owns every row of its assigned approximate eigenvectors $V$, each processor can independently perform the multiplication $AV$ in Algorithm 2; in other words, this operation is embarrassingly parallel. However, the matrix-matrix multiplications (lines 4, 6) and the generalized eigensolve (line 5) are handled via ScaLAPACK [44]. This incurs a data redistribution cost from a block column-parallel distribution to a block cyclic distribution and requires some parallel communication.

## *E. Dynamic Block Size Selection*

For each solve of the Sternheimer equations, each processor needs to solve $n_s$ block linear systems where each system contains $n_{\text{eig}}/p$ right-hand side vectors. As mentioned previously, the optimal choice of the block size $s$ balances the reduction in linear solver iterations with the increased computational work in using larger blocks sizes. This optimal block size is a function of the $(j, k)$ index pair and cannot be determined *a priori* for every chemical system, orbital, and quadrature point. Instead, we allow each processor to independently decide on a block size for each of its block linear systems using Algorithm 4.

**Algorithm 4** Dynamic block size selection for COCG

```
 1: $s \leftarrow 1$, $t_{\text{old}}$ = time using block size of $s$
 2: $s \leftarrow 2$, $t_{\text{new}}$ = time using block size of $s$
 3: while $\exists$ linear systems remaining do
 4:     if $t_{\text{new}} \leq 2t_{\text{old}}$ then
 5:         $s \leftarrow 2s$
 6:         $t_{\text{old}} \leftarrow t_{\text{new}}$
 7:         $t_{\text{new}}$ = time using block size of $s$
 8:     else
 9:         $s \leftarrow \frac{1}{2}s$
10:         break
11:     end if
12: end while
13: Solve remaining systems with block size of $s$
```

The idea behind Algorithm 4 is to test increasingly larger block sizes (in powers of 2) until finding one where the increased cost from the matrix-matrix multiplications outweighs the reduction in iteration count. Note that it is *usually* the case that the iteration count is non-increasing with block size; thus, there is little to no risk of the algorithm terminating early when a larger block size would have resulted in a faster computation.

## *F. Choosing the Initial Guess*

The only thing unspecified about the computation of the RPA correlation energy is the initial guess $V_0$ in Algorithm 2 (line 1) and the initial guess $Y_0$ in Algorithm 3 (line 1). To address the latter first, note that from the KS-DFT calculation we have previously obtained the bottom $n_s$ eigenpairs $(\lambda_j, \Psi_j)$ of $H$. The coefficient matrix in Equation (8) shares these *eigenvectors* but shifts the eigenvalues by $-\lambda_j + i\omega_k$. We use these shifted eigenpairs to construct an initial guess $Y_0$ via Galerkin projection

$$Y_0 = \Psi(E - \lambda_j I + i\omega_k I)^{-1}\Psi^H B \tag{13}$$

where $\Psi$ contains the known eigenvectors of $H$ and $E$ is a diagonal matrix containing the known eigenvalues of $H$. This initial guess is relatively cheap to construct when $n_s$ is not too large and effectively deflates the spectrum of $A_{j,k} = H - \lambda_j I + i\omega_k I$ by removing its most negative real part eigenvectors from the initial residual. This greatly dampens numerical difficulties that arise for index pairs near the problematic $(n_s, \ell)$ pair.

To address the initial guess in subspace iteration, note that for the first quadrature point $\omega_1$, there is no information available to select an initial guess better than a pointwise random one. However, at the end of subspace iteration for $\omega_1$, we converge a set of $n_{\text{eig}}$ eigenvectors of the matrix $\nu^{1/2}\chi^0(i\omega_1)\nu^{1/2}$. We use these eigenvectors as the initial guess for the matrix $\nu^{1/2}\chi^0(i\omega_2)\nu^{1/2}$, and so on. Experimentally, we find that when $|\omega_{k+1} - \omega_k| \rightarrow 0$, the eigenvectors for $\omega_k$ serve as

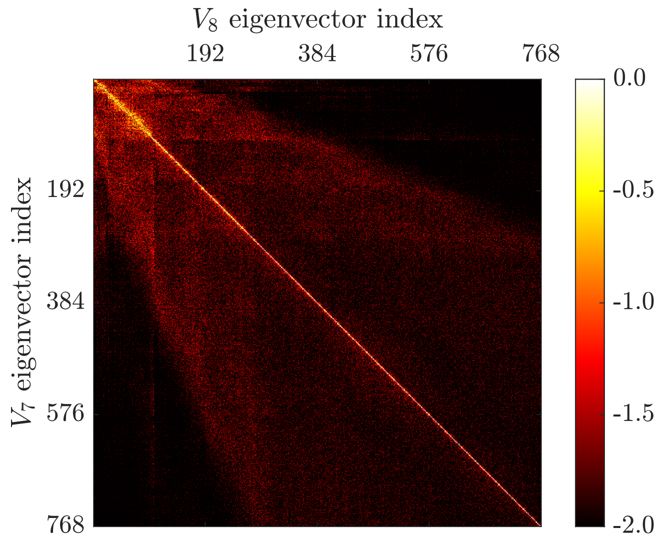

Fig. 2. Base-10 logarithm of the absolute values of $V_7^H V_8$, where $V_7$ and $V_8$ are the exact eigenvectors for the lowest 768 eigenvalues of $\nu\chi^0(i\omega_7)$ and $\nu\chi^0(i\omega_8)$ for Si$_8$, respectively. The line of high-magnitude elements along the main diagonal indicates $V_7$ is a reasonable approximation to $V_8$ to initialize subspace iteration.

excellent approximations to the eigenvectors for $\omega_{k+1}$. For our numerical integration scheme, the change in successive $\omega$ values goes to zero rapidly (see Table II). Figure 2 shows this explicitly, displaying the base-10 logarithm of the absolute values of $V_7^H V_8$, where $V_7$ and $V_8$ are the lowest 768 *exact eigenvectors* of $\nu\chi^0(i\omega_7)$ and $\nu\chi^0(i\omega_8)$ for Si$_8$, respectively. The striking feature is a diagonal line of high-magnitude elements with relatively low-magnitude elements away from the main diagonal. This indicates that each eigenvector in $V_7$ serves as a good approximation to the same index eigenvector in $V_8$, since we have $V_{7,j}^H V_{8,j} \approx 1$.

This approximation is so accurate that for later indices $k$, we skip the polynomial filtering step in Algorithm 2 (line 3) for the first subspace iteration and check whether or not Equation (7) is satisfied. We find that for the experimental systems tested in this work, the last few indices $k$ skip polynomial filtering altogether. This choice of initial guess and also the ordering of $\omega_k$ (from largest to smallest) allow us to minimize the number of Sternheimer solves required for $\omega_\ell$, which produces the most difficult linear systems to converge. This modification to the subspace iteration algorithm is presented in Algorithm 5.

The entire framework of our software is provided in Algorithm 6. Each process owns every row of its assigned $n_{\text{eig}}/p$ columns of $V$, i.e., a block column-parallel distribution. Matrix-matrix multiplications and eigensolves (lines 8, 9, 10, 15, 16, 17) are performed using a block cyclic distribution with ScaLAPACK. Each time we need to perform a multiplication of the form $(\nu^{1/2}\chi^0\nu^{1/2})V$ (lines 8, 11, 14, 15, 18), each processor individually performs Algorithm 7 completely in parallel. Since $H$ is sparse, each linear solver iteration (for one right-hand side vector) incurs a cost of $\mathcal{O}(n_d)$. Assuming the total number of iterations required is independent of $n_d$, each step in Algorithm 6 scales with cubic complexity in $n_d$, and thus our formulation scales with cubic complexity in the system size. This claim will be evaluated experimentally in Section IV-C.

**Algorithm 5** Subspace iteration with polynomial filtering to compute the eigenvalues of $\nu^{1/2}\chi^0\nu^{1/2}$, utilizing an accurate initial guess

```
1: Initialize $V_0$
2: $H_s \leftarrow V_0^H(\nu^{1/2}\chi^0\nu^{1/2})V_0$ and $M_s \leftarrow V_0^H V_0$
3: Solve the eigenvalue problem $H_sQ = M_sQD$
4: $V_0 \leftarrow V_0 Q$
5: Check convergence via Equation (7)
6: $j \leftarrow 0$
7: while not converged do
8:     $V_{j+1} \leftarrow p(\nu^{1/2}\chi^0\nu^{1/2})V_j$
9:     $H_s \leftarrow V_{j+1}^H(\nu^{1/2}\chi^0\nu^{1/2})V_{j+1}$ and $M_s \leftarrow V_{j+1}^H V_{j+1}$
10:    Solve the eigenvalue problem $H_sQ = M_sQD$
11:    $V_{j+1} \leftarrow V_{j+1}Q$
12:    Check convergence via Equation (7)
13:    $j \leftarrow j+1$
14: end while
```

**Algorithm 6** High performance, real space, many-body perturbation theory formulation for computing the RPA correlation energy

```
1: $E_{\text{RPA}} \leftarrow 0$
2: for k = 1, 2, ..., $\ell$ do
3:     if k = 1 then
4:         Initialize $V_0$ to a random $\mathbb{R}^{n_d \times n_{\text{eig}}}$ matrix
5:     else
6:         Initialize $V_0$ to converged $V$ for loop index $k-1$
7:     end if
8:     $H_s \leftarrow V_0^H(\nu^{1/2}\chi^0(i\omega_k)\nu^{1/2})V_0$ and $M_s \leftarrow V_0^H V_0$
9:     Solve the eigenvalue problem $H_sQ = M_sQD$
10:    $V_0 \leftarrow V_0 Q$
11:    Check convergence via Equation (7)
12:    $j \leftarrow 0$
13:    while not converged do
14:        $V_{j+1} \leftarrow p(\nu^{1/2}\chi^0(i\omega_k)\nu^{1/2})V_j$
15:        $H_s \leftarrow V_{j+1}^H(\nu^{1/2}\chi^0(i\omega_k)\nu^{1/2})V_{j+1}$ and
           $M_s \leftarrow V_{j+1}^H V_{j+1}$
16:        Solve the eigenvalue problem $H_sQ = M_sQD$
17:        $V_{j+1} \leftarrow V_{j+1}Q$
18:        Check convergence via Equation (7)
19:        $j \leftarrow j+1$
20:    end while
21:    $E_k \leftarrow \sum_{a=1}^{n_{\text{eig}}}(\ln(1-D_{aa})+D_{aa})$
22:    $E_{\text{RPA}} \leftarrow E_{\text{RPA}} + w_k E_k/2\pi$
23: end for
```

## IV. Numerical Experiments

All experiments in this section are performed on the Georgia Institute of Technology PACE–Phoenix cluster. Each compute

**Algorithm 7** The matrix multiplication $(\nu^{1/2}\chi^0(i\omega_k)\nu^{1/2})V$ from the view of a single process $r$ (in-place)

```
1: Input vector $V^{(r)} \in \mathbb{R}^{n_d \times (n_{\text{eig}}/p)}$
2: $V^{(r)} \leftarrow \nu^{1/2} V^{(r)}$
3: for j = 1, 2, …, $n_s$ do
4:     Solve $(H - \lambda_j I + i\omega_k I)Y_j^{(r)} = -V^{(r)} \odot \Psi_j$ for $Y_j^{(r)}$
       using Algorithms 3 and 4
5: end for
6: $V^{(r)} \leftarrow 4\,\mathrm{Re}\left(\sum_{j=1}^{n_s} \Psi_j^* \odot Y_j^{(r)}\right)$
7: $V^{(r)} \leftarrow \nu^{1/2} V^{(r)}$
```

TABLE I
EXPERIMENTAL PARAMETERS

| Parameter | Value |
|---|---|
| Mesh spacing | 0.69 Bohr |
| $n_{\text{eig}}$ per atom | 96 |
| $\ell$ (number of quadrature points) | 8 |
| $\deg p$ (filter polynomial degree) | 2 |
| $\tau_{\text{SI},1}$ | $4 \cdot 10^{-3}$ |
| $\tau_{\text{SI},2}$ | $2 \cdot 10^{-3}$ |
| $\tau_{\text{SI},3-8}$ | $5 \cdot 10^{-4}$ |

node has two sockets, each with an Intel Xeon Gold 6226 CPU, for a total of 24 CPU cores per compute node. Compute nodes are connected with 100 Gbps InfiniBand networking. KS-DFT calculations are performed with SPARC [21]. The Kohn-Sham occupied orbitals, the occupied orbital energies, and the electron density are saved from these calculations and used in the computation of the RPA correlation energy. Information about the compiler and library versions can be found in the appendix.

## A. Experimental Systems and Parameters

Table I shows the experimental parameters used in all presented results. Note that these parameters were chosen to be the loosest ones necessary to achieve chemical accuracy in energy differences. Between a perturbed $\text{Si}_8$ crystal and the same crystal with a vacancy ($\text{Si}_7$), ABINIT reports an energy difference of $\Delta E_{\text{RPA}} = 1.73 \cdot 10^{-3}$ Ha/atom (using a cutoff energy of $E_{\text{cut}} = 35$ Ha). Our software, with the parameters in Table I, reports an energy difference of $\Delta E_{\text{RPA}} = 1.28 \cdot 10^{-3}$ Ha/atom. The difference between these two values is $4.5 \cdot 10^{-4}$ Ha/atom which is within the accepted range for chemical accuracy. Since it is exceedingly expensive to compute absolute correlation energies [12], achieving chemical accuracy in the relative energy between two related systems suffices.

We select $n_{\text{eig}}$ to be a constant multiple of the number of atoms as in [27]. Experimentally, we find that the *entire* spectrum of $\nu\chi^0(i\omega)$ tends toward zero for large $\omega$, as in Figure 1. For this reason, we use a looser subspace iteration tolerance $\tau_{\text{SI}}$ for the first two quadrature points, as their contributions to the total RPA correlation energy are relatively small, and converging these values accurately does not impact the final reported energy. However, we use a tolerance close to chemical accuracy for the remainder of the quadrature points.

TABLE II
GAUSSIAN QUADRATURE POINTS AND WEIGHTS

| $k$ | $\omega_k$ (point) | $w_k$ (weight) |
|---|---|---|
| 1 | 49.36 | 128.4 |
| 2 | 8.836 | 10.76 |
| 3 | 3.215 | 2.787 |
| 4 | 1.449 | 1.088 |
| 5 | 0.690 | 0.518 |
| 6 | 0.311 | 0.270 |
| 7 | 0.113 | 0.138 |
| 8 | 0.020 | 0.053 |

TABLE III
EXPERIMENTAL SYSTEMS

| System | $n_d$ | $n_s$ | $n_{\text{eig}}$ |
|---|---|---|---|
| $\text{Si}_8$ | 3375 | 16 | 768 |
| $\text{Si}_{16}$ | 6750 | 32 | 1536 |
| $\text{Si}_{24}$ | 10125 | 48 | 2304 |
| $\text{Si}_{32}$ | 13500 | 64 | 3072 |
| $\text{Si}_{40}$ | 16875 | 80 | 3840 |

Table II shows the 8 quadrature points and weights used to approximate the semi-infinite integral via Equation (3). These values are computed using a Gauss-Legendre integration scheme analogous to one found in ABINIT.

Table III shows the experimental systems tested in this work. Each system is an 8-atom silicon crystal replicated between 1 and 5 times in one dimension. All atom positions are randomly perturbed (uniformly, as a fraction of the lattice constant). Based on the parameters in Table I, a single 8-atom cell has a finite difference domain of $15^3$ grid points and requires 768 eigenvalues of $\nu\chi^0$ to be computed to approximate the functional trace at each quadrature point.

## B. Optimizing the Linear Solver

While $\tau_{\text{SI}}$ is determined relatively easily, how strict $\tau_{\text{Sternheimer}}$ needs to be to achieve convergence of *subspace iteration* is a difficult question. Figure 3 shows the RPA correlation energy for $\text{Si}_8$ and the elapsed time (24 CPU cores) at a variety of linear solver tolerances. For this experiment, we do not use the dynamic block size approach in Algorithm 4, and instead simply fix $s = 1$.

As expected, the elapsed time decreases rapidly as the Sternheimer tolerance is loosened. However, the RPA correlation energy remains steady up to a tolerance of $2 \cdot 10^{-2}$. Convergence fails when using a tolerance greater than $4 \cdot 10^{-2}$ (i.e., the number of subspace iterations required for convergence is greater than 10, the maximum we allow). For the rest of this paper, we select a fixed Sternheimer tolerance of $10^{-2}$. This tolerance is sufficiently loose such that the linear system corresponding to index pair $(1, 1)$ is solved in roughly 3 iterations of COCG *regardless of block size*.

While this tolerance seems surprisingly loose, there is some theory regarding the convergence of subspace iteration for evolving matrices, i.e., matrices which are perturbed between iterations [45]. In our case the matrix whose eigenvalues we seek is a constant; however, solving the Sternheimer equations

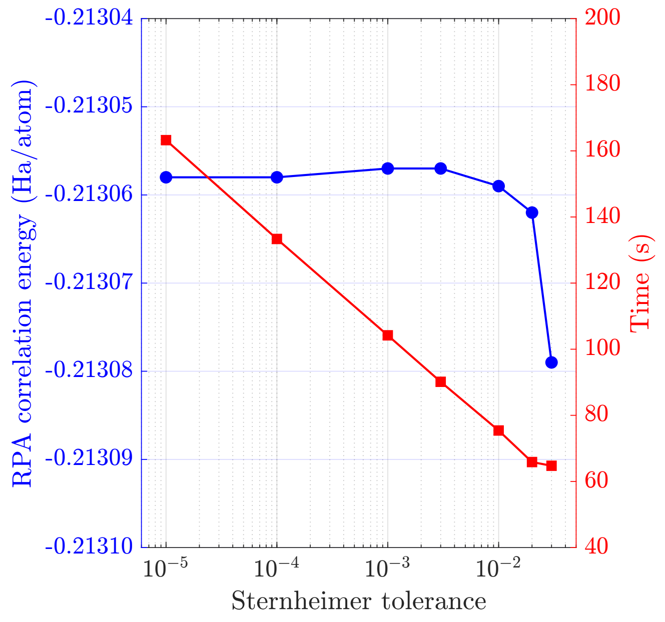

Fig. 3. RPA correlation energy (blue; circle) and total time (red; square) for $\mathrm{Si}_8$ at a number of Sternheimer tolerances, using 24 CPU cores. We are able to use relatively loose tolerances without disturbing the final reported RPA correlation energy.

with a loose tolerance is, in some sense, performing the multiplication $(\nu^{1/2}\chi^0\nu^{1/2})V$ in Algorithm 5 with a built-in perturbation. The exact process by which subspace iteration converges under this *specific* perturbation is not yet understood and is not considered in this work; however, the tangible effect this has is allowing us to use relatively loose Sternheimer tolerances, further mitigating numerical complications that might arise near the problematic $(n_s, \ell)$ index pair.

We now analyze the effectiveness of our dynamic block size selection in Algorithm 4. Table IV shows the block size frequencies selected, summed over all 24 CPU cores and over all Sternheimer solves, for the smallest three experimental systems. For $\mathrm{Si}_8$ and $\mathrm{Si}_{16}$, a block size of 2 was chosen most commonly at nearly 90% of the time. However, this fraction drops to around 40% for $\mathrm{Si}_{24}$. There are three factors involved in both relatively low block sizes being selected and the fractional increase in $s = 1$ with increasing system size. First, since the Sternheimer tolerance is loose, most $(j, k)$ index pairs result in numerically easy systems to converge to the desired tolerance. In the most extreme case, the number of COCG iterations required is nearly independent of the block size. In these instances, selecting a block size $s > 1$ is strictly worse due to the added computational cost of matrix-matrix multiplications. Second, the initial guess in Equation (13) deflates the entire portion of the spectrum in the negative real half-plane, and the resulting *effective spectrum* is positive semi-definite. While COCG may encounter numerical issues with eigenvalues close to zero, it will not additionally encounter issues that arise when performing linear solves with an indefinite coefficient matrix. Finally, as the system size increases, $n_s$ also increases. This results in a larger fraction

TABLE IV
DYNAMIC BLOCK SIZE FREQUENCIES

| Block size | $\mathrm{Si}_8$ | $\mathrm{Si}_{16}$ | $\mathrm{Si}_{24}$ |
|---|---|---|---|
| 1 | 2269 | 3688 | 40099 |
| 2 | 22373 | 42972 | 24395 |
| 4 | 272 | 188 | 18 |
| 8 | 13 | 0 | 0 |
| 16 | 33 | 0 | 0 |

of the Sternheimer equations having numerically easy $(j, k)$ index pairs, as only $j \approx n_s$ results in numerically difficult linear systems.

While the effect of block size is dampened by other numerical choices made in this work, note that the ability to dynamically select the block size is necessary for a computationally efficient and load-balanced implementation. For chemical systems where the Sternheimer tolerance must be stricter, or for metallic systems where the spectrum of the Hamiltonian does not have large gaps, an increased number of linear solver iterations required will implicitly drive Algorithm 4 to select larger block sizes. Additionally, it may not always be feasible to construct the initial guess in Equation (13); while this is not an issue for the system sizes tested in this work, this dynamic block size selection is imperative when scaling to larger chemical systems where this issue arises.

## C. Scalability Results

Figure 4 shows strong scaling results for the five experimental systems in Table III using between 1 and 32 compute nodes (between 24 and 768 CPU cores). In all cases, the maximum number of processors was chosen so that $n_{\text{eig}}/p \geq 4$; this way, the dynamic block size selection algorithm will not be skipped and the timings are comparable. Meanwhile, different computational kernels exhibit different performance characteristics in parallel execution. We further investigate the scalability of these kernels with a breakdown of the elapsed times.

Figure 5 shows the timing breakdown for $\mathrm{Si}_{40}$ between the major computational kernels in Algorithm 5. The $\nu^{1/2}\chi^0\nu^{1/2}$ data corresponds to the multiplications $(\nu^{1/2}\chi^0\nu^{1/2})V$ occurring in lines 2, 8, and 9; `matmult` corresponds to the matrix-matrix multiplications in lines 2, 4, 9, and 11; `eigensolve` corresponds to the solution of the generalized eigenvalue problem in lines 3 and 10; and `eval error` corresponds to the convergence checks done in lines 5 and 12. The evaluation of error using Equation (7) consists of two steps: performing the multiplication $(\nu^{1/2}\chi^0\nu^{1/2})V$ and an `MPI_Allreduce` so each processor is able to compute the error and determine whether or not convergence has been reached.

Applying $\nu^{1/2}\chi^0\nu^{1/2}$ is performed with more than 85% parallel efficiency when using no more than 192 CPU cores. When using 384 or more CPU cores, the parallel efficiency starts to decrease due to load imbalance. Such imbalance is due to a difference in linear system difficulty for Sternheimer equations assigned to different processors. The parallel scalability of `eval error` is similar to that of $\nu^{1/2}\chi^0\nu^{1/2}$. However, the `MPI_Allreduce` operation for checking the convergence

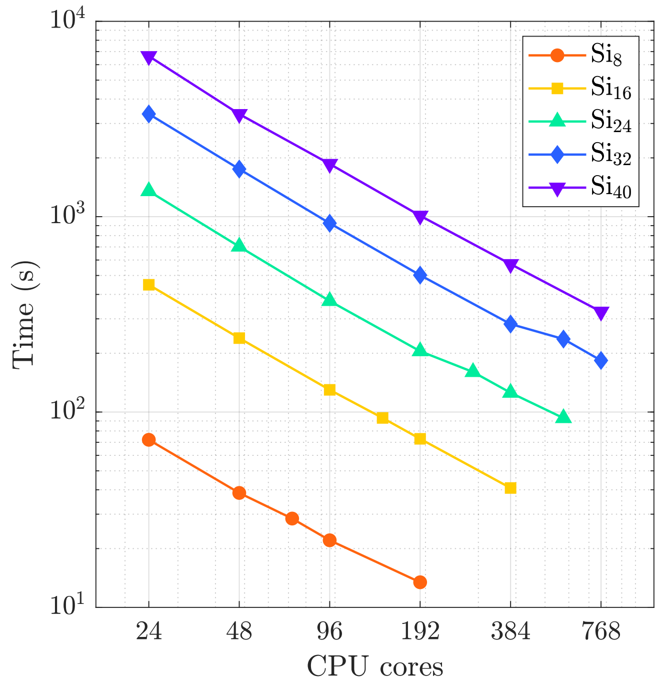

Fig. 4. Strong scaling results for every experimental system, using between 1 and 32 compute nodes (24 and 768 CPU cores). Our implementation exhibits good parallel efficiency across a wide range of chemical system sizes and processor counts.

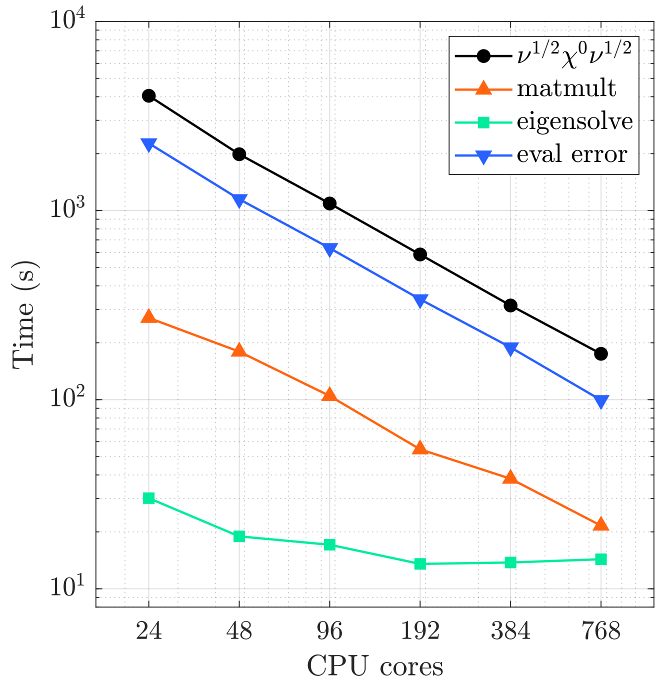

Fig. 5. Timing breakdown of key computational kernels for the $Si_{40}$ experimental system, using between 1 and 32 compute nodes (24 and 768 CPU cores). The matrix multiplication and eigensolve steps exhibit poor parallel scaling which ultimately hampers the parallel scaling of the overall approach at high processor counts.

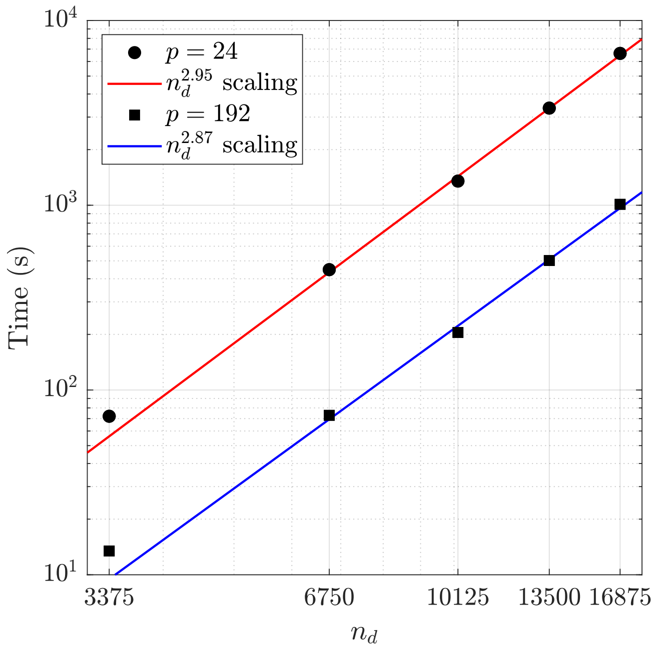

Fig. 6. Computational complexity with respect to $n_d$ for 24 (red) and 192 (blue) CPU cores. Solution times scale sub-cubically across a wide range of chemical system sizes and processor counts.

on all processors becomes relatively more expensive and limits the parallel efficiency. Like other iterative solvers, this unavoidable global reduction is one of the major challenges for scaling up to a large number of processors. The parallel efficiency of `matmult` is less satisfying mainly due to the fact that the dense matrices involved in these `matmult` operations are extremely tall-and-skinny (i.e., having many more rows than columns). Switching from ScaLAPACK to new parallel matrix multiplication algorithms, such as COSMA [46] or CA3DMM [47], can improve the performance of these `matmult` operations. The `eigensolve` scales poorly since the size of the dense matrix (3840 at most for the systems we test) is too small for a dense eigensolver to achieve good parallel efficiency on more than around 100 CPU cores.

We also evaluate the computational complexity with respect to the number of finite difference grid points $n_d$. Figure 6 shows that the elapsed time scales approximately as $\mathcal{O}(n_d^{2.95})$ and $\mathcal{O}(n_d^{2.87})$ for 24 and 192 CPU cores, respectively, showing sub-cubic scaling across a wide range of chemical system sizes.

We have shown that our formulation achieves good parallel efficiency and the largest computational kernel in our formulation is parallelized effectively. For comparison, there are unfortunately few high performance timing results for computing the RPA correlation energy in the literature. Additionally, many of the timing results that *do* exist (e.g., in WEST [48]) do not use methods that are easily analogized to Algorithm 1 and have significant hardware differences (e.g., the use of hardware accelerators). ABINIT computes the RPA correlation energy in a similar fashion to our formulation, albeit in reciprocal space. For $Si_8$ on 24 CPU cores, using 8 integration quadrature points and a cutoff energy of $E_{\text{cut}} = 35$ Ha, ABINIT has a time-to-solution of around 2940 seconds. For the same system, our formulation has a time-to-solution of around 72 seconds, a relative speedup of $40\times$. Note that ABINIT does not use the same parallelization scheme as presented here; it parallelizes over the quadrature points whereas we step through them sequentially to leverage good initial guesses in subspace iteration. The salient point is not the exact speedup but the order of magnitude difference between our formulation and ABINIT. Our formulation is faster than direct methods and scales more efficiently, both in terms of system size and processor count.

## V. Conclusions and Future Work

In this work, we present a high performance real space formulation of the RPA correlation energy which does not require the unoccupied Kohn-Sham orbitals, scales cubically in the number of finite difference grid points, and leverages the computational efficiency of solving large block linear systems. To solve the required linear systems, we develop a new short-term recurrence block Krylov subspace solver, augmented by an on-the-fly block size selection. Our approach allows each processor to dynamically select a suitable block size for its assigned linear systems. The key computational kernel in our formulation scales with high parallel efficiency and provides the ability to scale RPA calculations to large chemical systems while efficiently utilizing a large number of processors.

There are a few key avenues for future work. The first is addressing the poor parallel efficiency of the generalized eigensolve in subspace iteration. This computational kernel is difficult to parallelize and cannot efficiently leverage the full number of processors at very large processor counts. We are uninterested in computing the eigenvectors and eigenvalues of the subspace eigenvalue problem and only need to compute the trace of a nonlinear function of the subspace matrix spectrum. We can replace the eigensolve with Lanczos quadrature which can be much more easily parallelized. As long as $p \leq n_{\text{eig}}$, Lanczos quadrature can be done in an embarrassingly parallel way utilizing the full processor count. Lanczos quadrature can additionally take advantage of a block-type algorithm (in a similar fashion to block COCG) to potentially accelerate convergence.

Second, while our dynamic block size selection algorithm mitigates load balancing issues arising from the $(j, k)$ index pair, it is possible for load imbalance to arise between the right-hand side vectors even for a fixed index pair. This is currently not well-characterized (neither experimentally nor theoretically) but causes increased wall times due to some processors having more difficult linear systems to solve; thus, the time to perform $\nu^{1/2}\chi^0\nu^{1/2}V$ is governed by the slowest processor, and this slowest time scales with poor parallel efficiency as $n_{\text{eig}}/p$ decreases. A full understanding of this effect and a transition to a manager-worker model of work distribution would remove any load balancing issue that results from solving the Sternheimer equations.

Finally, we only consider unpreconditioned block COCG in this work; however, since a key term in the Hamiltonian is the discrete Laplacian matrix, we can leverage fast Poisson solves to use the *inverse* Laplacian as a preconditioner. For Sternheimer systems which are relatively easy to solve, it is unlikely any decrease in iteration count will offset the increased time per iteration. However, such a preconditioner may improve the spectral properties of more difficult Sternheimer systems and should be dynamically applied only in those cases.

## Acknowledgments

The authors gratefully acknowledge support from the U.S. Department of Energy, Office of Science under grant DE-SC0023445. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Department of Energy, or the U.S. Government. This research was supported by computing resources provided by the Partnership for an Advanced Computing Environment (PACE) at the Georgia Institute of Technology, including the PACE–Phoenix cluster.

## References

[1] P. Hohenberg and W. Kohn, "Inhomogeneous Electron Gas," *Phys. Rev.*, vol. 136, pp. B864–B871, 1964. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRev.136.B864

[2] W. Kohn and L. J. Sham, "Self-Consistent Equations Including Exchange and Correlation Effects," *Phys. Rev.*, vol. 140, pp. A1133–A1138, 1965. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRev.140.A1133

[3] A. D. Becke, "Perspective: Fifty years of density-functional theory in chemical physics," *The Journal of Chemical Physics*, vol. 140, no. 18, p. 18A301, 2014. [Online]. Available: https://doi.org/10.1063/1.4869598

[4] J. P. Perdew and K. Schmidt, "Jacob's ladder of density functional approximations for the exchange-correlation energy," *AIP Conference Proceedings*, vol. 577, no. 1, pp. 1–20, 07 2001. [Online]. Available: https://doi.org/10.1063/1.1390175

[5] V. Gavini, S. Baroni, V. Blum, D. R. Bowler, A. Buccheri, J. R. Chelikowsky, S. Das, W. Dawson, P. Delugas, M. Dogan, C. Draxl, G. Galli, L. Genovese, P. Giannozzi, M. Giantomassi, X. Gonze, M. Govoni, F. Gygi, A. Gulans, J. M. Herbert, S. Kokott, T. D. Kühne, K.-H. Liou, T. Miyazaki, P. Motamarri, A. Nakata, J. E. Pask, C. Plessl, L. E. Ratcliff, R. M. Richard, M. Rossi, R. Schade, M. Scheffler, O. Schütt, P. Suryanarayana, M. Torrent, L. Truflandier, T. L. Windus, Q. Xu, V. W.-Z. Yu, and D. Perez, "Roadmap on electronic structure codes in the exascale era," *Modelling and Simulation in Materials Science and Engineering*, vol. 31, no. 6, p. 063301, aug 2023. [Online]. Available: https://dx.doi.org/10.1088/1361-651X/acdf06

[6] X. Ren, P. Rinke, C. Joas, and M. Scheffler, "Random-phase approximation and its applications in computational chemistry and materials science," *Journal of Materials Science*, vol. 47, no. 21, p. 7447–7471, Jun. 2012. [Online]. Available: http://dx.doi.org/10.1007/s10853-012-6570-4

[7] J. Hermann, R. A. DiStasio Jr, and A. Tkatchenko, "First-principles models for van der Waals interactions in molecules and materials: Concepts, theory, and applications," *Chemical Reviews*, vol. 117, no. 6, pp. 4714–4758, 2017.

[8] H. Eshuis, J. E. Bates, and F. Furche, "Electron correlation methods based on the random phase approximation," *Theoretical Chemistry Accounts*, vol. 131, pp. 1–18, 2012.

[9] Z.-H. Cui, F. Wu, and H. Jiang, "First-principles study of relative stability of rutile and anatase TiO2 using the random phase approximation," *Physical Chemistry Chemical Physics*, vol. 18, no. 43, pp. 29 914–29 922, 2016.

[10] P. S. Schmidt and K. S. Thygesen, "Benchmark database of transition metal surface and adsorption energies from many-body perturbation theory," *The Journal of Physical Chemistry C*, vol. 122, no. 8, pp. 4381–4390, 2018.

[11] X. Gonze, F. Jollet, F. Abreu Araujo, D. Adams, B. Amadon, T. Applencourt, C. Audouze, J.-M. Beuken, J. Bieder, A. Bokhanchuk, E. Bousquet, F. Bruneval, D. Caliste, M. Côté, F. Dahm, F. Da Pieve, M. Delaveau, M. Di Gennaro, B. Dorado, C. Espejo, G. Geneste, L. Genovese, A. Gerossier, M. Giantomassi, Y. Gillet, D. Hamann, L. He, G. Jomard, J. Laflamme Janssen, S. Le Roux, A. Levitt, A. Lherbier, F. Liu, I. Lukačević, A. Martin, C. Martins, M. Oliveira, S. Poncé, Y. Pouillon, T. Rangel, G.-M. Rignanese, A. Romero, B. Rousseau, O. Rubel, A. Shukri, M. Stankovski, M. Torrent, M. Van Setten, B. Van Troeye, M. Verstraete, D. Waroquiers, J. Wiktor, B. Xu, A. Zhou, and J. Zwanziger, "Recent developments in the ABINIT software package," *Computer Physics Communications*, vol. 205, pp. 106–131, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010465516300923

[12] M. Kaltak, J. Klimeš, and G. Kresse, "Low Scaling Algorithms for the Random Phase Approximation: Imaginary Time and Laplace Transformations," *Journal of Chemical Theory and Computation*, vol. 10, no. 6, pp. 2498–2507, 2014, pMID: 26580770. [Online]. Available: https://doi.org/10.1021/ct5001268

[13] J. Deslippe, G. Samsonidze, D. A. Strubbe, M. Jain, M. L. Cohen, and S. G. Louie, "BerkeleyGW: A massively parallel computer package for the calculation of the quasiparticle and optical properties of materials and nanostructures," *Computer Physics Communications*, vol. 183, no. 6, pp. 1269–1289, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010465511003912

[14] T. D. Kühne, M. Iannuzzi, M. Del Ben, V. V. Rybkin, P. Seewald, F. Stein, T. Laino, R. Z. Khaliullin, O. Schütt, F. Schiffmann *et al.*, "CP2K: An electronic structure and molecular dynamics software package-Quickstep: Efficient and accurate electronic structure calculations," *The Journal of Chemical Physics*, vol. 152, no. 19, 2020.

[15] S. G. Balasubramani, G. P. Chen, S. Coriani, M. Diedenhofen, M. S. Frank, Y. J. Franzke, F. Furche, R. Grotjahn, M. E. Harding, C. Hättig *et al.*, "TURBOMOLE: Modular program suite for ab initio quantum-chemical and condensed-matter simulations," *The Journal of chemical physics*, vol. 152, no. 18, 2020.

[16] X. Ren, P. Rinke, V. Blum, J. Wieferink, A. Tkatchenko, A. Sanfilippo, K. Reuter, and M. Scheffler, "Resolution-of-identity approach to Hartree–Fock, hybrid density functionals, RPA, MP2 and GW with numeric atom-centered orbital basis functions," *New Journal of Physics*, vol. 14, no. 5, p. 053020, 2012.

[17] M. Govoni and G. Galli, "Large Scale GW Calculations," *Journal of Chemical Theory and Computation*, vol. 11, no. 6, pp. 2680–2696, 2015, pMID: 26575564. [Online]. Available: https://doi.org/10.1021/ct500958p

[18] M. Schlipf, H. Lambert, N. Zibouche, and F. Giustino, "SternheimerGW: A program for calculating GW quasiparticle band structures and spectral functions without unoccupied states," *Computer Physics Communications*, vol. 247, p. 106856, 2020.

[19] J. J. Mortensen, L. B. Hansen, and K. W. Jacobsen, "Real-space grid implementation of the projector augmented wave method," *Phys. Rev. B*, vol. 71, p. 035109, Jan 2005. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevB.71.035109

[20] S. Ghosh and P. Suryanarayana, "SPARC: Accurate and efficient finite-difference formulation and parallel implementation of Density Functional Theory: Isolated clusters," *Computer Physics Communications*, vol. 212, pp. 189–204, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010465516303046

[21] Q. Xu, A. Sharma, B. Comer, H. Huang, E. Chow, A. J. Medford, J. E. Pask, and P. Suryanarayana, "SPARC: Simulation Package for Ab-initio Real-space Calculations," *SoftwareX*, vol. 15, p. 100709, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352711021000546

[22] J. Paier, X. Ren, P. Rinke, G. E. Scuseria, A. Grüneis, G. Kresse, and M. Scheffler, "Assessment of correlation energies based on the random-phase approximation," *New Journal of Physics*, vol. 14, no. 4, p. 043002, apr 2012. [Online]. Available: https://dx.doi.org/10.1088/1367-2630/14/4/043002

[23] S. L. Adler, "Quantum Theory of the Dielectric Constant in Real Solids," *Phys. Rev.*, vol. 126, pp. 413–420, Apr 1962. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRev.126.413

[24] N. Wiser, "Dielectric Constant with Local Field Effects Included," *Phys. Rev.*, vol. 129, pp. 62–69, Jan 1963. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRev.129.62

[25] G. H. Golub and J. H. Welsch, "Calculation of Gauss Quadrature Rules," *Mathematics of Computation*, vol. 23, no. 106, pp. 221–s10, 1969. [Online]. Available: http://www.jstor.org/stable/2004418

[26] X. Gonze, B. Amadon, G. Antonius, F. Arnardi, L. Baguet, J.-M. Beuken, J. Bieder, F. Bottin, J. Bouchet, E. Bousquet, N. Brouwer, F. Bruneval, G. Brunin, T. Cavignac, J.-B. Charraud, W. Chen, M. Côté, S. Cottenier, J. Denier, G. Geneste, P. Ghosez, M. Giantomassi, Y. Gillet, O. Gingras, D. R. Hamann, G. Hautier, X. He, N. Helbig, N. Holzwarth, Y. Jia, F. Jollet, W. Lafargue-Dit-Hauret, K. Lejaeghere, M. A. Marques, A. Martin, C. Martins, H. P. Miranda, F. Naccarato, K. Persson, G. Petretto, V. Planes, Y. Pouillon, S. Prokhorenko, F. Ricci, G.-M. Rignanese, A. H. Romero, M. M. Schmitt, M. Torrent, M. J. van Setten, B. Van Troeye, M. J. Verstraete, G. Zérah, and J. W. Zwanziger, "The ABINIT project: Impact, environment and recent developments," *Computer Physics Communications*, vol. 248, p. 107042, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010465519303741

[27] H. F. Wilson, D. Lu, F. m. c. Gygi, and G. Galli, "Iterative calculations of dielectric eigenvalue spectra," *Phys. Rev. B*, vol. 79, p. 245106, Jun 2009. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevB.79.245106

[28] G. H. Golub and G. Meurant, *Matrices, Moments and Quadrature with Applications*. Princeton University Press, 2010. [Online]. Available: http://www.jstor.org/stable/j.ctt7tbvs

[29] M. Hutchinson, "A Stochastic Estimator of the Trace of the Influence Matrix for Laplacian Smoothing Splines," *Communications in Statistics - Simulation and Computation*, vol. 18, no. 3, pp. 1059–1076, 1989. [Online]. Available: https://doi.org/10.1080/03610918908812806

[30] V. Goncharov, *Nonlinear Optical Response in Atoms, Molecules and Clusters*. Springer, 08 2014.

[31] M. H. Gutknecht, "A Brief Introduction to Krylov Space Methods for Solving Linear Systems," in *Frontiers of Computational Science*, Y. Kaneda, H. Kawamura, and M. Sasai, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 53–62.

[32] T. F. Chan and W. L. Wan, "Analysis of Projection Methods for Solving Linear Systems with Multiple Right-Hand Sides," *SIAM Journal on Scientific Computing*, vol. 18, no. 6, pp. 1698–1721, 1997. [Online]. Available: https://doi.org/10.1137/S1064827594273067

[33] D. P. O'Leary, "The block conjugate gradient algorithm and related methods," *Linear Algebra and its Applications*, vol. 29, pp. 293–322, 1980, special Volume Dedicated to Alson S. Householder. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0024379580902475

[34] Y. Zhou, Y. Saad, M. L. Tiago, and J. R. Chelikowsky, "Self-consistent-field calculations using Chebyshev-filtered subspace iteration," *Journal of Computational Physics*, vol. 219, no. 1, pp. 172 – 184, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S002199910600146X

[35] A. Sharma and P. Suryanarayana, "On real-space Density Functional Theory for non-orthogonal crystal systems: Kronecker product formulation of the kinetic energy operator," *Chemical Physics Letters*, vol. 700, pp. 156–162, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0009261418302914

[36] X. Jing and P. Suryanarayana, "Efficient real space formalism for hybrid density functionals," 2024. [Online]. Available: https://arxiv.org/abs/2406.16998

[37] M. Embree, "How Descriptive are GMRES Convergence Bounds?" 2022.

[38] Y. Saad and M. H. Schultz, "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM Journal on Scientific and Statistical Computing*, vol. 7, no. 3, pp. 856–869, 1986. [Online]. Available: https://doi.org/10.1137/0907058

[39] R. W. Freund, "Conjugate Gradient-Type Methods for Linear Systems with Complex Symmetric Coefficient Matrices," *SIAM Journal on Scientific and Statistical Computing*, vol. 13, no. 1, pp. 425–448, 1992. [Online]. Available: https://doi.org/10.1137/0913023

[40] G. Borzi, "Solution of the finite element vector Helmholtz equation by means of eigenvalue displacement," *IEEE Transactions on Antennas and Propagation*, vol. 54, no. 6, pp. 1758–1765, 2006.

[41] E. Aune, D. Simpson, and J. Eidsvik, "Parameter estimation in high

dimensional Gaussian distributions," *Statistics and Computing*, vol. 24, no. 2, pp. 247–263, 2014.

[42] H. van der Vorst and J. Melissen, "A Petrov-Galerkin type method for solving Axk=b, where A is symmetric complex," *IEEE Transactions on Magnetics*, vol. 26, no. 2, pp. 706–708, 1990.

[43] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *Journal of Research of the National Bureau of Standards*, vol. 49, pp. 409–435, 1952. [Online]. Available: https://api.semanticscholar.org/CorpusID:2207234

[44] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley, *ScaLAPACK Users' Guide*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1997.

[45] Y. Saad, "Analysis of Subspace Iteration for Eigenvalue Problems with Evolving Matrices," *SIAM Journal on Matrix Analysis and Applications*, vol. 37, no. 1, pp. 103–122, 2016. [Online]. Available: https://doi.org/10.1137/141002037

[46] G. Kwasniewski, M. Kabić, M. Besta, J. VandeVondele, R. Solcà, and T. Hoefler, "Red-blue pebbling revisited: near optimal parallel matrix-matrix multiplication," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3295500.3356181

[47] H. Huang and E. Chow, "CA3DMM: A New Algorithm Based on a Unified View of Parallel Matrix Multiplication," in *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2022, pp. 1–15.

[48] V. W.-z. Yu and M. Govoni, "GPU Acceleration of Large-Scale Full-Frequency GW Calculations," *Journal of Chemical Theory and Computation*, vol. 18, no. 8, pp. 4690–4707, 2022, pMID: 35913080. [Online]. Available: https://doi.org/10.1021/acs.jctc.2c00241

# Appendix: Artifact Description / Artifact Evaluation

## I. Overview of Contributions and Artifacts

### A. Paper's Main Contributions

$C_1$ A high-performance real-space code which computes the electronic correlation energy via the many-body random phase approximation (RPA) within density functional theory (DFT); this code interfaces with the computational chemistry code SPARC (**S**imulation **P**ackage for **A**b-initio **R**eal-space **C**alculations)

### B. Computational Artifacts

$A_1$ https://zenodo.org/doi/10.5281/zenodo.12194237

| Artifact ID | Contributions Supported | Related Paper Elements |
|---|---|---|
| $A_1$ | $C_1$ | Figures 3–6 |

## II. Artifact Identification

### A. Computational Artifact $A_1$

#### Relation To Contributions

The artifact contains the source code for the computational chemistry code SPARC as well as the source code for our novel contribution in computing the RPA correlation energy. Additionally, the artifact contains the experimental systems (including all necessary input files) shown in the main text.

#### Expected Results

Results should indicate:

- Cubic scaling with respect to system size (i.e., the number of atoms), substantiating Figure 6
- Good parallel scaling (for a fixed system size) for up to 1000 CPU cores, substantiating Figure 4

#### Expected Reproduction Time (in Minutes)

The artifact setup time is a few minutes. The artifact execution time depends on the system size and number of CPU cores used. For each experiment performed in the main text, execution time ranged from approximately 0.3 minutes to approximately 120 minutes.

#### Artifact Setup (incl. Inputs)

Both SPARC and our RPA code need to be compiled. SPARC does *not* need to be run; all output files required from SPARC are already provided in the artifact. Please follow these steps:

1) Clone the SPARC and RPA codes from GitHub via `git clone` https://github.com/SPARC-RPA-SC/SPARC-RPA.git (use the default `rpa` branch)
2) Navigate to the `src` directory
3) Compile SPARC using `make`
4) Navigate to the `RPA/src` directory
5) Compile RPA using `make`
6) Navigate to the `RPA/tests` directory

Our code requires:

- Intel C compiler (main text uses v19.1.3.304)
- Intel MPI library (main text uses v2019 – Update 9)
- Intel MKL (main text uses v2020 – Update 4)

Newer versions of the Intel compiler may require `mpiicx` (instead of `mpiicc`) in both the SPARC and RPA makefiles.

Our RPA code has no hardware dependencies. Intel MKL requires using Intel or AMD x86-64 processors. Experiments in the main text were performed on Intel Xeon Gold 6226 12-core CPUs.

All input files are provided (`Si8.rpa` up to `Si40.rpa`) in the `RPA/tests` directory and do not need to be altered to reproduce the results in the main text, except for the `TOL_STERN_RES` parameter *only if* reproducing Figure 3. A sample input file `Si8.rpa` is shown below.

```
N_NUCHI_EIGS: 768
N_OMEGA: 8
TOL_EIG: 4e-3 2e-3 5e-4 5e-4 5e-4 5e-4 5e-4 5e-4
TOL_STERN_RES: 1e-2
MAXIT_FILTERING: 10
CHEB_DEGREE_RPA: 2
FLAG_PQ_OPERATOR: 0
FLAG_COCGINITIAL: 1
```

#### Artifact Execution

The artifact can be run directly. For example, for $Si_8$, navigate to `RPA/tests/Si08`. Then, this case can be run using the command `mpirun -np <nprocs> ../../lib/rpacalc -name Si8 > out.log`. The output is placed into a file `Si8.out` and any logging/debugging data (such as timings for individual computational kernels) is placed into a file `out.log`.

If using a job scheduling system like SLURM, a sample job submission script is provided for each system, located in the file `RPA/tests/Si<n>/submit.sbatch`. This needs to be properly filled out and then the job can be submitted via `sbatch submit.sbatch`.

#### Artifact Analysis (incl. Outputs)

Output files are denoted by `Si8.out` up to `Si40.out` and are produced automatically upon running the code. Output files begin with a long preamble repeating the input parameters. A sample output file `Si8.out` is shown below, excluding this preamble. The final RPA correlation energy (total and per atom) and the elapsed walltime are found near the bottom. These can be used to reproduce Figures 3–6 in the main text. The output additionally includes timings for each subspace iteration performed at each quadrature point.

Individual kernel timings (for Figure 5) can easily be extracted from the log file via `grep -i "subspace iteration" out.log > subtimings.log`. The file `subtimings.log` includes the time for each computational step in each subspace iteration performed at each quadrature point, which sums to the total walltime (sans any overhead).

```
***************************************************************************
                         RPA Parallelization
***************************************************************************
NP_NUCHI_EIGS_PARAL_RPA: 24
NP_SPIN_PARAL_RPA: 1
NP_KPOINT_PARAL_RPA: 1
NP_BAND_PARAL_RPA: 1
***************************************************************************
Estimated memory usage in RPA calculation is 36.97 MB
***************************************************************************
q-point 1 (reduced coords 0.000 0.000 0.000), weight 1.000
 omega 1 (value 49.365, 0~1 value 0.020, weight 0.051)
ncheb|ErpaTerm(Ha/atom)| First 2 eigs & Last 2 eigs of nu chi0|eig Error|Timing (s)
  1      -6.266E-05     -0.00055 -0.00053 ; -0.00014 -0.00014  7.384E-03 3.16
  2      -7.422E-05     -0.00055 -0.00055 ; -0.00017 -0.00017  3.256E-03 1.75
***************************************************************************
q-point 1 (reduced coords 0.000 0.000 0.000), weight 1.000
 omega 2 (value 8.836, 0~1 value 0.102, weight 0.111)
ncheb|ErpaTerm(Ha/atom)| First 2 eigs & Last 2 eigs of nu chi0|eig Error|Timing (s)
  1      -2.316E-03     -0.01197 -0.01120 ; -0.00255 -0.00249  3.753E-03 5.14
  2      -2.463E-03     -0.01197 -0.01120 ; -0.00304 -0.00303  1.559E-03 2.87
***************************************************************************
q-point 1 (reduced coords 0.000 0.000 0.000), weight 1.000
 omega 3 (value 3.215, 0~1 value 0.237, weight 0.157)
ncheb|ErpaTerm(Ha/atom)| First 2 eigs & Last 2 eigs of nu chi0|eig Error|Timing (s)
  1      -8.918E-03     -0.06053 -0.05872 ; -0.00585 -0.00575  2.686E-03 7.18
  2      -9.133E-03     -0.06053 -0.05872 ; -0.00670 -0.00659  1.201E-03 4.01
  3      -9.194E-03     -0.06053 -0.05872 ; -0.00717 -0.00710  6.414E-04 3.98
  4      -9.217E-03     -0.06053 -0.05872 ; -0.00741 -0.00739  3.959E-04 4.01
***************************************************************************
q-point 1 (reduced coords 0.000 0.000 0.000), weight 1.000
 omega 4 (value 1.449, 0~1 value 0.408, weight 0.181)
ncheb|ErpaTerm(Ha/atom)| First 2 eigs & Last 2 eigs of nu chi0|eig Error|Timing (s)
  1      -2.093E-02     -0.21159 -0.20104 ; -0.00896 -0.00884  7.409E-04 9.18
  2      -2.096E-02     -0.21159 -0.20104 ; -0.00915 -0.00900  4.160E-04 5.16
***************************************************************************
q-point 1 (reduced coords 0.000 0.000 0.000), weight 1.000
 omega 5 (value 0.690, 0~1 value 0.592, weight 0.181)
ncheb|ErpaTerm(Ha/atom)| First 2 eigs & Last 2 eigs of nu chi0|eig Error|Timing (s)
  1      -3.882E-02     -0.56367 -0.54784 ; -0.00964 -0.00945  3.697E-04 12.02
***************************************************************************
q-point 1 (reduced coords 0.000 0.000 0.000), weight 1.000
 omega 6 (value 0.311, 0~1 value 0.763, weight 0.157)
ncheb|ErpaTerm(Ha/atom)| First 2 eigs & Last 2 eigs of nu chi0|eig Error|Timing (s)
  0      -5.468E-02     -1.38269 -1.31821 ; -0.00975 -0.00956  3.816E-04 4.29
***************************************************************************
q-point 1 (reduced coords 0.000 0.000 0.000), weight 1.000
 omega 7 (value 0.113, 0~1 value 0.898, weight 0.111)
ncheb|ErpaTerm(Ha/atom)| First 2 eigs & Last 2 eigs of nu chi0|eig Error|Timing (s)
  0      -5.423E-02     -2.70747 -2.59334 ; -0.00978 -0.00958  3.506E-04 5.05
***************************************************************************
q-point 1 (reduced coords 0.000 0.000 0.000), weight 1.000
 omega 8 (value 0.020, 0~1 value 0.980, weight 0.051)
ncheb|ErpaTerm(Ha/atom)| First 2 eigs & Last 2 eigs of nu chi0|eig Error|Timing (s)
  0      -3.262E-02     -4.17389 -3.66537 ; -0.00979 -0.00959  3.546E-04 5.66
***************************************************************************
Energy terms in every (qpt, omega) pair (Ha)
q-point 1
omega 1: -5.93784E-04, omega 2: -1.97012E-02, omega 3: -7.37333E-02,
omega 4: -1.67702E-01, omega 5: -3.10573E-01, omega 6: -4.37417E-01,
omega 7: -4.33824E-01, omega 8: -2.60925E-01,
Total RPA correlation energy: -1.70447E+00 (Ha), -2.13059E-01 (Ha/atom)
***************************************************************************
                               Timing info
***************************************************************************
Total walltime                      :  73.856 sec
```