Any order imaginary time propagation method for solving the Schrodinger equation

Article *in* Chemical Physics Letters · September 2008

DOI: 10.1016/j.cplett.2009.01.068

CITATIONS
53

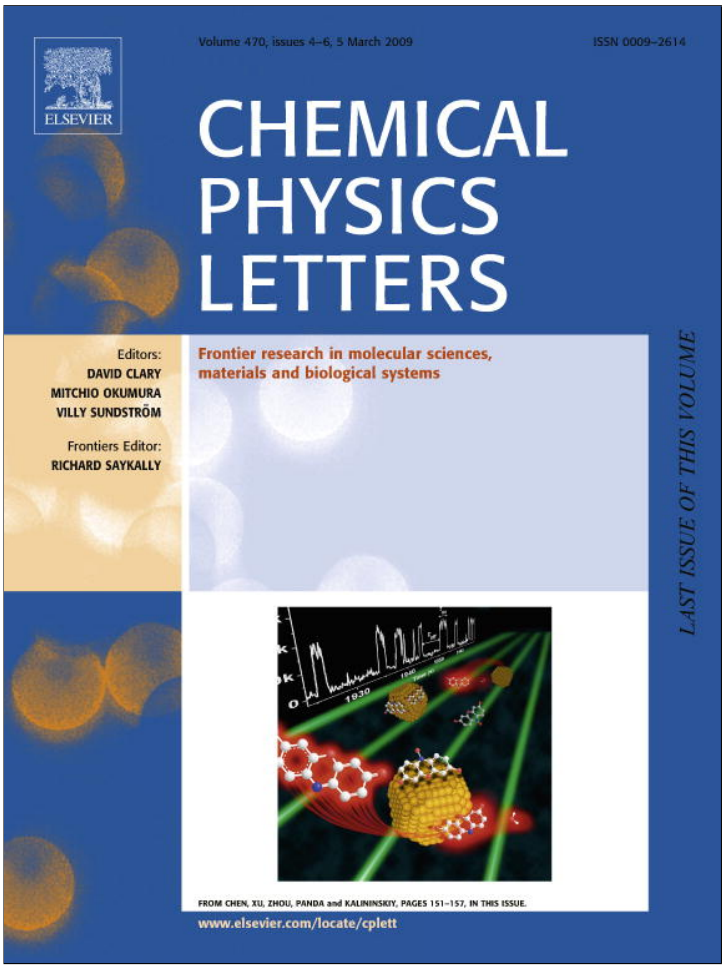
READS
1,393

3 authors:

Siu A. Chin
Texas A&M University
161 PUBLICATIONS **5,380** CITATIONS
SEE PROFILE

Stefan Janecek
Austrian Academy of Sciences (OeAW)
22 PUBLICATIONS **235** CITATIONS
SEE PROFILE

E. Krotscheck
University at Buffalo, State University of New York
323 PUBLICATIONS **5,846** CITATIONS
SEE PROFILE



Chemical Physics Letters 470 (2009) 342–346

Contents lists available at ScienceDirect

Chemical Physics Letters

journal homepage: www.elsevier.com/locate/cplett

Any order imaginary time propagation method for solving the Schrödinger equation

Siu A. Chin [a,*], S. Janecek [b], E. Krotscheck [b]

[a] Department of Physics, Texas A&M University, MS 4242, College Station, TX 77843, USA
[b] Institut für Theoretische Physik, Johannes Kepler Universität Linz, A-4040 Linz, Austria

ARTICLE INFO

Article history:
Received 24 September 2008
In final form 25 January 2009
Available online 29 January 2009

ABSTRACT

The eigenvalue-function pair of the 3D Schrödinger equation can be efficiently computed by use of high order, imaginary time propagators. Due to the diffusion character of the kinetic energy operator in imaginary time, algorithms developed so far are at most 4th order. In this work, we show that for a grid based algorithm, imaginary time propagation of any even order can be devised on the basis of multi-product splitting. The effectiveness of these algorithms, up to the 12th order, is demonstrated by computing all 120 eigenstates of a model C_{60} molecule to very high precisions. The algorithms are particularly useful when implemented on parallel computer architectures.

1. Introduction

With the advance of the density-functional method of solving diverse solid state physics and quantum chemistry problems [1–3], it is of growing importance to solve the Schrödinger equation on a large 3D mesh with greater than $N = 10^6$ grid points. (See Ref. [1] for an excellent summary of early real-space grid methods.) For such a large mesh, conventional matrix methods are impractical, since even the minimal matrix–vector multiplication would be prohibitively slow. Among $\mathcal{O}(N)$ methods, we have previously shown that 4th-order imaginary time propagation [4] provides an effective means of solving the Kohn–Sham and related equations [5]. The use of all *forward* time step 4th-order algorithms in solving the imaginary time Schrödinger equation has since been adapted by many research groups [6–8].

The lowest n states of the one-body Schrödinger equation

$$H\psi_j(\mathbf{r}) = E_j\psi_j(\mathbf{r}) \tag{1.1}$$

with Hamiltonian

$$H = -\frac{\hbar^2}{2m}\nabla^2 + V(\mathbf{r}) \equiv T + V \tag{1.2}$$

can be obtained in principle by applying the evolution operator ($\epsilon = -\Delta t$)

$$\mathscr{T}(\epsilon) \equiv e^{\epsilon(T+V)} \tag{1.3}$$

repeatedly on the ℓth time step approximation $\left\{\psi_j^{(\ell)}(\mathbf{r})\right\}$ to give the set of states $\{\phi_j(\mathbf{r}), 1 \leqslant j \leqslant n\}$,

$$\phi_j^{(\ell+1)} \equiv \mathscr{T}(\epsilon)\psi_j^{(\ell)} \tag{1.4}$$

and orthogonalizing the states after every step,

$$\psi_j^{(\ell+1)} \equiv \sum_i c_{ji}\phi_i^{(\ell+1)}, \quad \left(\psi_j^{(\ell+1)}|\psi_i^{(\ell+1)}\right) = \delta_{ij}. \tag{1.5}$$

The method is made practical by approximating the exact evolution operator (1.3) by a general product form,

$$\mathscr{T}(\epsilon) = \prod_{i=1}^{M} e^{a_i\epsilon T}e^{b_i\epsilon V}. \tag{1.6}$$

The simplest second-order decomposition, or the split operator method [9], is

$$\mathscr{T}_2(\epsilon) \equiv e^{\frac{1}{2}\epsilon V}e^{\epsilon T}e^{\frac{1}{2}\epsilon V} = \mathscr{T}(\epsilon) + O(\epsilon^3). \tag{1.7}$$

When this operator acts on a state $\psi_j(\mathbf{r})$, the two operators $e^{\frac{1}{2}\epsilon V}$ correspond to point-by-point multiplications and $e^{\epsilon T}$ can be evaluated by one complete (forward and backward) Fast Fourier Transform (FFT). Both are $\mathcal{O}(N)$ processes. For $\mathscr{T}_2(\epsilon), |\epsilon|$ has to be small to maintain good accuracy and many iterations are therefore needed to project out the lowest n states. To achieve faster convergence, one could in principle iterate higher-order algorithms at larger time steps. Unfortunately, Sheng [10] and Suzuki [11] have proved that, beyond second order, no factorization of the form (1.6) can have all positive coefficients $\{a_i, b_i\}$. This forward time step requirement is essential for imaginary time propagation because if any a_i were negative, then the operator $e^{-a_i\Delta t T}$ would be unbounded, resulting in unstable algorithms corresponding to unphysical backward diffusion in time. To derive forward, all positive time step 4th-order algorithms, Suzuki [12] and Chin [13] have shown that a correction to the potential of the form $[V,[T,V]] = (\hbar^2/m)|\nabla V|^2$, as first used by Takahashi–Imada [14] and later suggested by Suzuki [15], must be included in the decomposition process. We have shown [4] previously that these forward 4th-order algorithms can achieve similar accuracy at an order-of-magnitude larger step sizes than the

* Corresponding author.
E-mail address: chin@physics.tamu.edu (S.A. Chin).

0009-2614/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.cplett.2009.01.068

second-order splitting (1.7). More recently Bandrauk et al. [16] have suggested that, instead of including such a gradient term, one can use complex coefficients $\{a_i, b_i\}$ having positive real parts. For real time propagation, their complex time step algorithms are not left–right symmetric and therefore are not time-reversible. For imaginary time propagation, their 4th-order algorithm S'_4 requires *five* complete complex-to-complex FFTs, whereas our forward algorithm 4A only needs two real-to-complex/complex-to-real FFTs [4].

2. Multi-product expansion

If the decomposition of $\mathscr{T}(\epsilon)$ is restricted to a single product as in (1.6), then there is no practical means of implementing a sixth- or higher-order forward algorithm [17]. However, if this restriction is relaxed to a sum of products,

$$e^{\epsilon(T+V)} = \sum_k c_k \prod_i e^{a_{k,i}\epsilon T} e^{b_{k,i}\epsilon V}, \tag{2.1}$$

then the requirement that $\{a_{k,i}, b_{k,i}\}$ be positive means that each product can only be second order. Since $\mathscr{T}_2(\epsilon)$ is second order with positive coefficients, its powers $\mathscr{T}_2^k(\epsilon/k)$ can form a basis for such a multi-product expansion. Recent work [18] shows that such an expansion is indeed possible and takes the form

$$e^{\epsilon(T+V)} = \sum_{k=1}^{n} c_k \mathscr{T}_2^k\left(\frac{\epsilon}{k}\right) + O(\epsilon^{2n+1}) \equiv \mathscr{T}_n(\epsilon) + O(\epsilon^{2n+1}), \tag{2.2}$$

where the coefficients c_k are given in closed form for any n:

$$c_i = \prod_{j=1(\neq i)}^{n} \frac{k_i^2}{k_i^2 - k_j^2} \tag{2.3}$$

with $\{k_1, k_2, \ldots, k_n\} = \{1, 2, \ldots, n\}$. Since the symmetric $\mathscr{T}_2(\epsilon)$ has only odd powers in ϵ,

$$\mathscr{T}_2(\epsilon) = \exp[\epsilon(T+V) + \epsilon^3 E_3 + \epsilon^5 E_5 + \cdots], \tag{2.4}$$

where E_i are higher-order commutators of T and V, the expansion (2.2) is just a systematic extrapolation which successively removes each odd order error. Explicitly, for $n = 2$–6, we have the following order 4 to order 12 multi-product expansions:

$$\mathscr{T}_4(\epsilon) = -\frac{1}{3}\mathscr{T}_2(\epsilon) + \frac{4}{3}\mathscr{T}_2^2\left(\frac{\epsilon}{2}\right), \tag{2.5}$$

$$\mathscr{T}_6(\epsilon) = \frac{1}{24}\mathscr{T}_2(\epsilon) - \frac{16}{15}\mathscr{T}_2^2\left(\frac{\epsilon}{2}\right) + \frac{81}{40}\mathscr{T}_2^3\left(\frac{\epsilon}{3}\right), \tag{2.6}$$

$$\mathscr{T}_8(\epsilon) = -\frac{1}{360}\mathscr{T}_2(\epsilon) + \frac{16}{45}\mathscr{T}_2^2\left(\frac{\epsilon}{2}\right) - \frac{729}{280}\mathscr{T}_2^3\left(\frac{\epsilon}{3}\right) + \frac{1024}{315}\mathscr{T}_2^4\left(\frac{\epsilon}{4}\right), \tag{2.7}$$

$$\mathscr{T}_{10}(\epsilon) = \frac{1}{8640}\mathscr{T}_2(\epsilon) - \frac{64}{945}\mathscr{T}_2^2\left(\frac{\epsilon}{2}\right) + \frac{6561}{4480}\mathscr{T}_2^3\left(\frac{\epsilon}{3}\right) - \frac{16384}{2835}\mathscr{T}_2^4\left(\frac{\epsilon}{4}\right) + \frac{390625}{72576}\mathscr{T}_2^5\left(\frac{\epsilon}{5}\right), \tag{2.8}$$

$$\mathscr{T}_{12}(\epsilon) = -\frac{1}{302400}\mathscr{T}_2(\epsilon) + \frac{8}{945}\mathscr{T}_2^2\left(\frac{\epsilon}{2}\right) - \frac{2187}{4480}\mathscr{T}_2^3\left(\frac{\epsilon}{3}\right) + \frac{65536}{14175}\mathscr{T}_2^4\left(\frac{\epsilon}{4}\right) - \frac{9765625}{798336}\mathscr{T}_2^5\left(\frac{\epsilon}{5}\right) + \frac{17496}{1925}\mathscr{T}_2^6\left(\frac{\epsilon}{6}\right). \tag{2.9}$$

Since each $\mathscr{T}_2$ requires one complete FFT, the above series of $2n$-order algorithms only requires $n(n+1)/2$ complete FFTs. Thus algorithms of order 4, 6, 8, 10 and 12 only require 3, 6, 10, 15 and 21 complete FFTs. The low-order extrapolation (2.5) has been used previously [19]. Here, we have a systematic expansion to any even order. Note that Romberg-type extrapolation [19], such as

$$\mathscr{T}_6(\epsilon) = -\frac{1}{15}\mathscr{T}_4(\epsilon) + \frac{16}{15}\mathscr{T}_4^2\left(\frac{\epsilon}{2}\right), \tag{2.10}$$

which triples the number of FFTs in going from order $2n$ to $2n+2$, is not competitive with Eq. (2.2)'s linear increase of only $n+1$ additional FFTs.

Since some coefficients c_k are negative, this requires that the corresponding product, when acting on state ψ_j, be subtracted. This is doable for a grid based discretization of the wave function, which is just a point-by-point subtraction.

3. Model calculations

To demonstrate the convergence of this new family of algorithms, and to compare with alternative methods, we apply them to two representative models: The first is a simple model of a C_{60} molecule in 3D, the second is the (an-)isotropic three-dimensional harmonic oscillator.

The calculations are started at a certain time step ϵ, using plane waves in an appropriate box as initial states. One iteration of the algorithm consists of propagating the states with the nth-order evolution operator $\mathscr{T}_n(\epsilon)$, and subsequently orthogonalizing the propagated states. After each iteration, we calculate the variance of all states with respect to the evolution operator,

$$R_j^E = \frac{1}{|\epsilon|}\sum_k \left|\mathscr{T}_n(\epsilon)\psi_j(\mathbf{r}_k) - e^{\epsilon E_j}\psi_j(\mathbf{r}_k)\right|^2. \tag{3.1}$$

Only states with $R_j^E > \gamma$, where γ is a prescribed error bound, need to be propagated and orthogonalized in the subsequent iterations. As soon as all states have converged at the time step ϵ, their variances with respect to the Hamiltonian,

$$R_j^H = \sum_k \left|H\psi_j(\mathbf{r}_k) - E_j\psi_j(\mathbf{r}_k)\right|^2 \tag{3.2}$$

are calculated. If $R_j^H < \gamma$ for all states j, the iterations are terminated, otherwise the time step is reduced by a constant factor, i.e. ϵ is replaced by $\alpha\epsilon$, and the whole process is repeated, taking the result of the previous iteration as initial values. We have empirically determined the optimal reduction factor to be $\alpha \approx 0.5$. Recently, Lehtovaara et al. [7] have suggested that the time step size can be optimally adjusted with added efforts.

3.1. Quantum well model of C_{60}

In our model of a 3D C_{60} molecule, the effective attraction of the carbon ions is modeled by a potential of the form:

$$V(\mathbf{r}) = -\sum_i \frac{V_0}{\cosh^2(|\mathbf{r} - \mathbf{R}_i|/d)}, \tag{3.3}$$

where $\mathbf{R}_i$ are the locations of the carbon atoms in the C_{60} cage. The strength V_0 was chosen 1 in units of $\hbar^2/2m$ and the width of the troughs $d = 0.05$ a.u. This potential accommodates 120 bound states as needed for a C_{60} calculation. We have also applied the method in 2D to a square grid of 9×9 quantum dots described by the same potential. The convergence behavior of the algorithms in both cases are practically identical and need not be discussed separately.

While the above potential is not a realistic description of a C_{60} molecule, it serves to highlight an important and realistic aspect of such computations. In the 3D case, the lowest 240 eigenvalues consist of two bands of almost degenerate eigenvalues: One band of 60 eigenvalues centered around an energy of $-5.245\hbar^2/2m$ with an average level spacing of $0.0034\hbar^2/2m$, and another band of 180 eigenvalues around $-2.992\hbar^2/2m$ with an average level spacing of $0.0077\hbar^2/2m$. Taking spin degeneracy into account, 120 occupied states are needed for a C_{60} calculation. Thus, the highest occupied

state lies within the second, highly degenerate band of eigenvalues. Degenerate eigenvalues in general pose a notorious problem for eigenvalue solvers [2,3] that contain an orthogonalization step. In our implementation of the algorithm, we use the subspace orthogonalization method described in Ref. [5], which is an application of the Petrov–Galerkin method [2]. In that method, problems only occur whenever the *highest calculated state* lies within a band of almost degenerate states. This degrades the convergence rate of the algorithm: In fact, computing 120 states for the above C_{60} model is only insignificantly faster than calculating 240 states, because in this case, the highest state is at an energy gap. Independent of that, we find that the convergence rates for the highest and the lowest states are the same.

Figs. 1 and 2 show the convergence of various algorithms for both the lowest and the highest state of the model C_{60} molecule. To generate the figures, we started with a time step of $\Delta t = 0.5$ and a set of plane wave initial states in an appropriate box. We then reduced the time step by a factor of 0.9 each time after convergence has been reached, so that the power law behavior can be seen cleanly. As described above, it suffices to reduce the time step by a factor 0.5 in realistic calculations. Thus, one can conclude from Figs. 1 and 2 that the 12th-order algorithm can reach the 10^{-10} error level in just two iterations. Also note that these orders of convergence apply to eigenstates and the eigenvalues deduced from the changing normalization of eigenstates. As noted in Ref. [4], the eigenvalues of a $(2n)$th-order algorithm converge with order $2(2n)$ when computed as expectation values.

3.2. Harmonic oscillator models

As a second example we have studied the (an-)isotropic harmonic oscillator in three dimensions,

$$V(x,y,z) = \frac{1}{2}(\alpha_x x^2 + \alpha_y y^2 + \alpha_z z^2). \tag{3.4}$$

To compare the operator factorization methods with an alternative eigenvalue solver, namely the 'Implicitly Restarted Lanczos Method' (IRLM) provided by ARPACK [20], we have calculated the lowest $N = 120$ eigenvectors of an anisotropic harmonic oscillator with

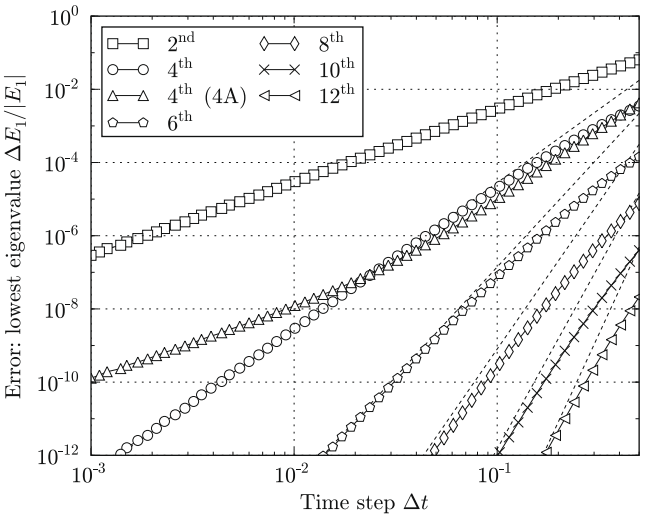


Fig. 1. The convergence of the 2nd to 12th-order algorithms for the lowest eigenstate of a model 'C_{60}' molecule are as shown by markers defined in the inset. The dashed lines, as a guide to the eye, are the power laws Δt^n for $n = 2, 4, \ldots, 12$. Also shown is the convergence curve for algorithm 4A of Ref. [4]. Its characteristic deviation from the Δt^4 power law behavior at small Δt is due to discretization errors in evaluating the double commutator $[V,[T,V]]$.

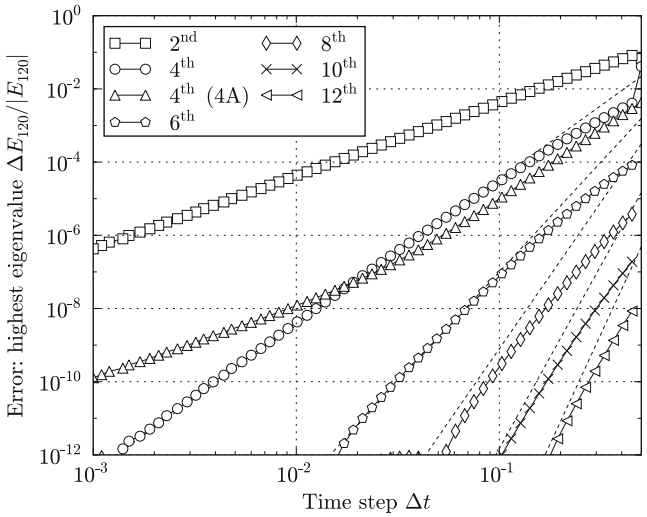


Fig. 2. Same as Fig. 1 for the 120th eigenstate of the model C_{60} molecule.

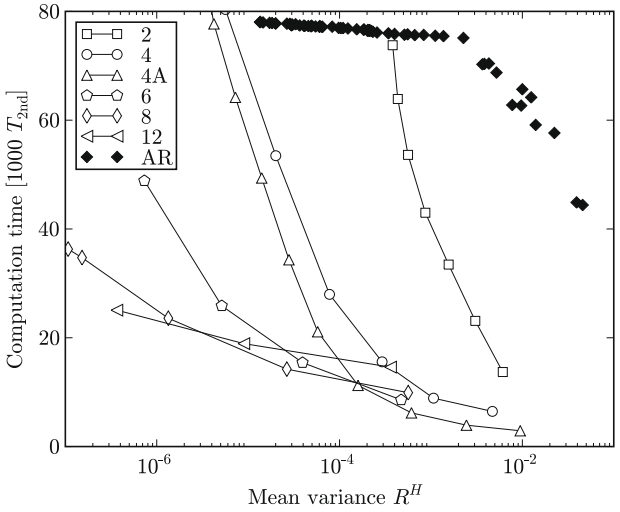


Fig. 3. The figure shows a comparison of the performances of the 2nd-, 4th-, 6th-, 8th-, and 12th-order algorithm and the 4th-order version of Ref. [4] of our imaginary time propagation method and the Lanczos method as implemented in the ARPACK [20] (labeled AR) for the anisotropic harmonic oscillator as discussed in the text.

the parameters $\alpha_x = 1.0, \alpha_y = 1.1$ and $\alpha_z = 1.2$. Fig. 3 shows the achieved average variance

$$R^H = \sqrt{\frac{\sum_j R_j^H}{\sum_j E_j^2}} \tag{3.5}$$

of the eigenvectors as a function of the total computation time. Results are shown for the 2nd, 4th, 6th, 8th and 12th-order algorithm, the 4th-order algorithm 4A of Ref. [4], and for the Lanczos method. The computation time needed to perform the second-order propagation of *one* state, $T_{2nd} = T[\mathcal{T}_2(\epsilon)\psi(\mathbf{r})]$, provides a natural time unit for the $2n$th-order operator factorization algorithms, cf. Eqs. (2.5)–(2.9). Since it is also comparable to the time needed for the $H\psi$ operations in the Lanczos method, we have used T_{2nd} as a time unit in Fig. 3. For the operator factorization methods, we have propagated only the lowest ('occupied') 120 states, while for the Lanczos calculation we have used 240 states, following the suggestion of the ARPACK manual that the number of 'active' states should be roughly twice the number of 'occupied' states.

Whereas the performance of the second-order operator factorization is comparable to the Lanczos method, and does not produce variances better then $R^H \approx 10^{-3}$, higher-order methods provide a significant improvement. Especially when only relatively low accuracies up to $R^H < 10^{-4}$ are required, the higher-order operator factorization methods performed almost an order-of-magnitude faster than Lanczos. When higher accuracy is required (which is the case, for example, for finding equilibrium configurations of metal clusters, or for the calculation of NMR shifts) the higher-order methods are somewhat more time consuming, but still outperform the Lanczos method (which did not, in our tests, reach that level of accuracy) by a factor of four to five. The extra time needed for improved accuracy is largely compensated by the fact that parallelization is easy and quite efficient especially for the high-order methods.

We conclude this section by remarking that the convergence of the operator factorization method is practically the same for the isotropic harmonic oscillator as long as the highest computed state is not in a band of degenerate states. For the isotropic case, the Lanczos method normally did not produce all states. We also note that our conclusions on the relative performance of the above algorithms are similar for the C_{60} model discussed above.

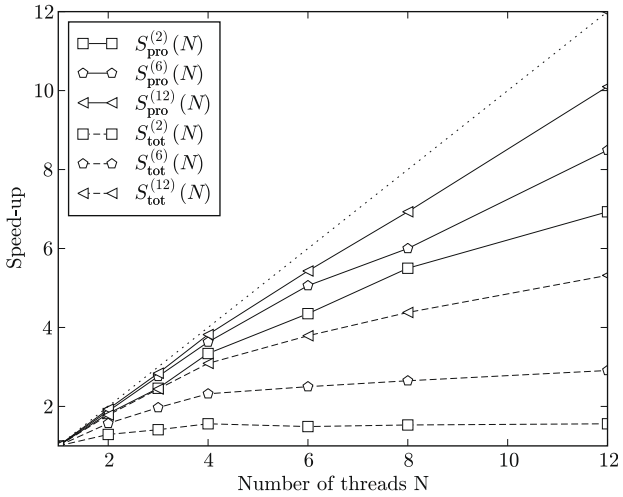


Fig. 4. The total speed-up time factor $S_{\rm tot}$ (solid lines) and the propagation only (without orthogonalization) time speed-up factor $S_{\rm pro}$, as a function of parallel threads, for the 2nd-, 6th-, and 12th-order algorithm (filled squares, circles, and triangles, respectively). Also shown is the 'ideal' speed-up factor (dotted line).

4. Parallelization

The advantage of high-order propagation methods is particularly compelling on parallel computer architectures: The propagation step (1.4) can be parallelized efficiently without having to abandon the advantage of using FFTs by simply distributing the states ψ_j across different processors. In such an arrangement, however, the parallelization of the orthogonalization step is notoriously difficult, and we have made no specific efforts to parallelize this step.

As stated above, the propagation time $T_{\rm pro}$, i.e. the time it takes to carry out step (1.4) for the $2n$-order algorithm, increases as $n(n+1)/2$ whereas the orthogonalization time $T_{\rm ort}$ remains the same. Hence, parallelization becomes more effective since more time is spent for propagation than for orthogonalization.

The time $T_{\rm tot}$ for one iteration step on an ideal machine with N processors is

$$T_{\rm tot}(N) = T_{\rm pro}/N + T_{\rm ort} \tag{4.1}$$

and the speed-up ratio for the 'propagation only' and the total time step including orthogonalization for the jth-order algorithm is

$$S^{(j)}_{\rm pro/tot}(N) = \frac{T_{\rm pro/tot}(1)}{T_{\rm pro/tot}(N)}, \tag{4.2}$$

assuming the number of states is larger than the number of processors allocated for the task. The actual speed-up ratio will be less than this ideal since we have neglected communication overhead and other hardware/system specific issues.

Fig. 4 shows the speed-up ratio for the C_{60} model calculation in the case of the 2nd, 6th, and 12th-order algorithms on a 256 Itanium [21] processor Altix [22] machine for up to 12 threads. We show the two speed-up ratios $S^{(j)}_{\rm pro}(N)$ and $S^{(j)}_{\rm tot}(N)$. No particular effort was made to parallelize the orthogonalization step. Evidently, the speed-up of the propagation step alone is a reasonably linear function of the number of threads. The performance improves significantly with the order of the algorithm because the increase computational effort for propagation can be distributed while the cost of communication remained the same.

The 12th-order algorithm can reach about 80% of the optimal performance. The actual speed-up is limited by the orthogonalization step; while the 12th-order algorithm can still attain a more than fivefold speed-up, it is hardly worth parallelizing the second-order algorithm. The specific speed-up factor for higher-order algorithms also depends on the number N of needed eigenstates. In general, the time for propagation is essentially proportional to N, whereas the time for orthogonalization goes as N^2.

Thus, high-order algorithms provide two advantages: they have faster convergence at larger time steps and are more adaptive to parallel computing environments. In the single processor mode, we have determined that the 6th-order algorithm performed the best.

5. Conclusions

The impressive convergence of our high-order algorithms has a computing cost: One propagation step of the $2n$th-order algorithm is equivalent to $n(n+1)/2$ propagation steps of the second-order algorithm. This cost is compensated by two effects: The first is shown in the figures: The much faster convergence as a function of time step implies that fewer iterations are needed to complete the calculation. The second advantage is less obvious; since orthogonalization is carried out *after* the propagation step (2.2), the relatively costly number of orthogonalization steps is dramatically reduced.

The most likely use of the high-order algorithms will be in real-space implementations of density-functional theory. For realistic systems, one must include non-local pseudo-potentials. We have recently [23] implemented algorithm 4A using pseudo-potentials of the Kleinman–Bylander form [24]. Calculating the double commutator $[V,[T,V]]$ for such non-local potentials is possible, but the computational cost is twice that of propagating just the potential. Moreover, if the electron density in the vicinity of the ion cores deviates from spherical symmetry, then some approximate treatments may degrade the order of the algorithm. Thus, our new algorithms, without needing the double commutator, should be even more effective for realistic density-functional calculations.

Acknowledgement

This work was supported, in part, by the Austrian Science Fund FWF (to EK) under project P18134.

References

[1] T.L. Beck, Rev. Mod. Phys. 72 (2000) 1041.

[2] T. Torsti et al., Phys. Status Solidi B 243 (2006) 1016.
[3] N.R. Wijesekera, G. Feng, T.L. Beck, Phys. Rev. B 75 (2007) 115101.
[4] J. Auer, E. Krotscheck, S.A. Chin, J. Chem. Phys. 115 (2001) 6841.
[5] M. Aichinger, E. Krotscheck, Comput. Mater. Sci. 34 (2005) 188.
[6] L. Brualla, K. Sakkos, J. Boronat, J. Casulleras, J. Chem. Phys. 121 (2004) 636.
[7] L. Lehtovaara, J. Toivanen, J. Eloranta, J. Comput. Phys. 221 (2006) 148.
[8] G.B. Ren, J.M. Rorison, Phys. Rev. B 77 (2008) 245318.
[9] D. Feit, J.A. Fleck Jr., J. Chem. Phys. 78 (1982) 301.
[10] Q. Sheng, IMA J. Numer. Anal. 9 (1989) 199.
[11] M. Suzuki, J. Math. Phys. 32 (1991) 400.
[12] M. Suzuki, in: D.P. Landau, K.K. Mon, H.-B. Schüttler (Eds.), Computer Simulation Studies in Condensed Matter Physics, vol. VIII, Springer, Berlin, 1996, p. 1.
[13] S.A. Chin, Phys. Lett. A 226 (1997) 344.
[14] M. Takahashi, M. Imada, J. Phys. Soc. Jpn. 53 (1984) 3765.
[15] M. Suzuki, Phys. Lett. A 201 (1995) 425.
[16] A.D. Bandrauk, E. Dehghanian, H. Lua, Chem. Phys. Lett. 419 (2006) 346.
[17] S.A. Chin, Phys. Rev. E 71 (2005) 016703.
[18] S.A. Chin, Multi-product splitting and Runge–Kutta–Nyström integrators, 2008. Available from: <arXiv:0809.0914>.
[19] K.E. Schmidt, M.A. Lee, Phys. Rev. E 51 (1995) 5495.
[20] <http://www.caam.rice.edu/software/ARPACK/>.
[21] Itanium is a registerd trademark of Intel Corp.
[22] Altix is a registered trademark of Silicon Graphics Inc.
[23] E.R. Hernández, S. Janecek, M.S. Kaczmarski, E. Krotscheck, Phys. Rev. B 75 (2007) 075108.
[24] L. Kleinman, D.M. Bylander, Phys. Rev. Lett. 48 (1982) 1425.