A comparison of numerical implementations of the Eigenstate expansion method for quantum molecular dynamics simulations

Author(s): Arbenz, Peter; Bröker, Oliver; Chinellato, Oscar; Gander, Walter; Geus, Roman; Strebel, Rolf

Publication Date: 2000

Permanent Link: https://doi.org/10.3929/ethz-a-006653779

Rights / License: In Copyright - Non-Commercial Use Permitted
A COMPARISON OF NUMERICAL IMPLEMENTATIONS OF THE EIGENSTATE EXPANSION METHOD FOR QUANTUM MOLECULAR DYNAMICS SIMULATIONS

PETER ARBENZ†, OLIVER BRÖKER†, OSCAR CHINELLATO†, WALTER GANDER†, ROMAN GEUS†, AND ROLF STREBEL†

Abstract. We investigate the efficient computation of a few of the lowest eigenvalues of a symmetric eigenvalue problem occurring in quantum dynamical molecular simulations. The large sparse system of order \( n^3 \) is highly structured such that its multiplication with a vector costs \( O(n^3 \log n) \) floating point operations only.

We compare a number of eigensolvers: subspace iteration, two variants of the restarted Lanczos algorithm, among them ARPACK, the Jacobi-Davidson algorithm and the classical algorithms from LAPACK for full matrices. Some of them are applied with a shift-and-invert spectral transformation. The arising systems of equations are solved by the preconditioned conjugate gradient method for which new efficient ADI-type preconditioners are proposed and analyzed.

Key words. quantum molecular dynamics, eigenfunction expansion method, sparse eigenvalue problem, preconditioned conjugate gradient method.

AMS subject classifications. 65F10, 65F15

1. Introduction. Numerical methods derived from classical molecular (CM) dynamics allow one to perform dynamics on a nanosecond time scale for systems involving tens of thousands of atoms. However, classical (as opposed to quantum) molecular dynamics prevents the study of processes like proton transfers and the making and breaking of chemical bonds that are of much interest in enzyme reactions or other biochemical processes. Unfortunately, the complexity of a fully quantum mechanical (QM) treatment of chemical systems with more than a very small number of degrees of freedom is far beyond the capacity of nowadays computers. Therefore, in molecular dynamics computer simulation packages like GROMOS [6, 13], mixed QM/CM simulation schemes are implemented in which a small part of the degrees of freedom are treated quantum mechanically, generally regarding light particles such as electrons and protons that are directly involved in the reactive molecular rearrangement, while the many more degrees of freedom of the environment of the reactive site are treated classically.

The QMD method is based on so-called real-time-path-integral techniques [16], which are aimed at obtaining the time evolution of a state function \( \psi \) under the influence of the Hamilton operator \( \mathcal{H} \). If the Hamilton operator

\[
\mathcal{H} \psi = (\mathcal{K} + \mathcal{V}) \psi = -\frac{\hbar^2}{2m} \Delta \psi + V(\psi, x).
\]

in the time-dependent Schrödinger equation

\[
i\hbar \frac{\partial \psi}{\partial t} = \mathcal{H} \psi(t)
\]

does not change with time, the state \( \psi \) at time \( t \) is given formally by

\[
\psi(t) = \exp\left(-\frac{i}{\hbar} \mathcal{H} t\right) \psi(0) = \exp\left(-\frac{i}{\hbar} (\mathcal{K} + \mathcal{V}) t\right) \psi(0).
\]

†Swiss Federal Institute of Technology (ETH), Institute of Scientific Computing, 8092 Zurich, Switzerland ([arbzenz,broeker,chinellato,gander,geus,strebel]@inf.ethz.ch)

There are two problematic aspects regarding the Hamilton operator: the kinetic energy operator $K$ is non-local in the coordinate representation, and the kinematic energy operator $K$ and the potential energy operator $V$ do not commute. In the literature a number of schemes have been proposed and tested to integrate (1.2). These are based on so-called split operator formulae and the use of the fast Fourier transforms (FFT) to obtain a local kinetic energy operator in momentum space. In [5], Billeter and van Gunsteren found that an integration technique involving an eigenfunction expansion of $\psi(t)$ is suited for mixed QD/MD simulations. If $\{\phi_k\}_{k=1}^{\infty}$ is the complete set of eigenfunctions of $\mathcal{H}$,

$$\mathcal{H}\phi_k = \lambda_k\phi_k,$$

we can expand the state $\psi$ as

$$\psi = \sum_{k=1}^{\infty} \alpha_k \phi_k, \quad \alpha_k = \langle \phi_k, \psi \rangle. \quad (1.3)$$

As the Schrödinger equation (1.1) can be solved if the initial state equals an eigenfunction of $\mathcal{H}$, $\phi_k(t) = \phi_k(0)e^{-(i/\hbar)\lambda_k t}$, we get for an initial state of the form (1.3)

$$\psi(t) = \sum_k \langle \phi_k, \psi(t) \rangle \phi_k = \sum_k \langle \phi_k, \exp(-it\mathcal{H})\psi(0) \rangle \phi_k = \sum_k e^{-i\lambda_k t/\hbar} \langle \phi_k, \psi(0) \rangle \phi_k$$

Since the state $\psi$ takes on only low energy levels with non-negligible probability, just a few of the lowest eigenvalues $\lambda_k$ of $\mathcal{H}(t)$ are needed to get accurate approximations. If the Hamilton operator is time-dependent, an approximate solution is obtained by linearization.

We investigate three dimensional as well as simplified one-dimensional model problems. In the one-dimensional case $\psi$ in (1.1) is represented as a Fourier polynomial of order $n$. In three dimensions $\psi$ is represented as the tensor product of Fourier polynomial of order $n$. The resulting symmetric large sparse matrix eigenvalue problems thus have orders $n^d$ where $d$ is the number of dimensions.

Solving the large eigenvalue problem with the QR algorithm as available in LAPACK [1] requires $O(n^{3d})$ floating point operations and $O(n^{2d})$ memory locations which makes this approach infeasible for approximations in 3D of even moderate orders $n$. We therefore investigate algorithms that are adapted to sparse large scale eigenvalue problems in that they access the system matrix only by means of matrix-vector products. The special matrix structure admits using FFT techniques such that a matrix-vector product costs only $O(n \log(n))$ floating point operations.

We solve the resulting symmetric large sparse matrix eigenvalue problems by two variants of the restarted Lanczos algorithm [17], by subspace iteration [20] and by the Jacobi-Davidson algorithm [24, 9]. The latter two algorithms have been chosen as they can naturally exploit good initial guesses for the eigenvectors. This is important in QD simulations where in each of thousands of time steps an eigenvalue problem has to be solved that differs only little from the one of the previous point in time. The system of equations that arise with the shift-and-invert spectral transformation (subspace iteration and Jacobi-Davidson algorithm) are solved with the preconditioned conjugate gradient method. We propose a new preconditioner to enhance convergence.

The outline of the paper is as follows. In section 2 we investigate the simplified 1D problem followed by numerical experiments in section 3. In section 4 we turn to the theory for the 3D eigenvalue problem. Numerical experiments are presented in 5. We close with some remarks in section 6.

2. The 1D problem. The Schrödinger equation (1.1) for a one-dimensional particle of mass $m$ is given by

$$-\frac{\hbar^2}{2m}u''(x) + V(x)u(x) = \lambda u(x), \quad 0 < x < 2\pi,$$

with periodic boundary conditions. We approximate $u(x)$ as a Fourier polynomial of order $n$, $n$ even (mostly $n = 2^q$ for some $q \in \mathbb{N}$),

$$u(x) = \sum_{k=-\frac{n}{2}+1}^{\frac{n}{2}} c_k e^{ikx}, \quad \text{(2.2)}$$

whence

$$u''(x) = \sum_{k=-\frac{n}{2}+1}^{\frac{n}{2}} -k^2 c_k e^{ikx}.$$  

Solving (2.1) by collocation at the points $x_j = 2\pi j/n$, $j = 0, \ldots, n-1$ yields the $n$ equations

$$\sum_{k=-\frac{n}{2}+1}^{\frac{n}{2}} \left( \frac{\hbar^2}{2m}k^2 + V(x_j) - \lambda \right) c_k e^{ikx_j} = 0, \quad j = 0, \ldots, n-1. \quad \text{(2.3)}$$

Noting that $e^{-ikx_j} = e^{i(n-k)x_j}$ and identifying $c_{-k}$ with $c_{n-k}$ eq. (2.3) becomes

$$\sum_{k=0}^{n-1} \left( \frac{\hbar^2}{2m} \min(k, n-k)^2 + V(x_j) - \lambda \right) c_k e^{ikx_j} = 0, \quad j = 0, \ldots, n-1. \quad \text{(2.4)}$$

The matrix form of these equations is

$$(F_n K_n + V_n F_n - \lambda F_n) c = 0, \quad \text{(2.5)}$$

where,

$$K_n := \frac{\hbar^2}{2m} \text{diag} \left( 0, 1, \ldots, (\frac{n}{2})^2, (\frac{n}{2} - 1)^2, \ldots, 1 \right) \in \mathbb{R}^{n \times n},$$

$$V_n = \text{diag} \left( V(x_0), \ldots, V(x_{n-1}) \right) \in \mathbb{R}^{n \times n},$$

$$F_n = (\omega_n^{jk})_{j,k=0}^{n-1} \in \mathbb{C}^{n \times n}, \quad \omega_n = e^{2\pi i/n}.$$

$F_n$ is the $n$-by-$n$ Fourier matrix [18]. Notice, that $F_n F_n^* = n I_n$, $I_n$ the $n$-by-$n$ identity matrix, such that $F_n^{-1} = \frac{1}{n} F_n^*$. Here, $F_n^*$ denotes the conjugate transpose of $F_n$. Set $x := F_n c$ to get the eigenvalue problem

$$(H_n + V_n) x := (F_n K_n F_n^{-1} + V_n) x = \lambda x. \quad \text{(2.6)}$$

We have

**Lemma 2.1.** The matrix $H_n := F_n K_n F_n^{-1}$ is real symmetric and circulant.

Thus, the matrix $A_n := F_n K_n F_n^{-1} + V_n$ is real symmetric and up to the diagonal circulant as well. This implies that its eigenvalues are real and the eigenvectors form an orthonormal basis of $\mathbb{R}^n$. As the matrix-vector multiplication $A_n x$ is cheap if the

fast Fourier transform is employed, $A_n$ is called \textit{structurally sparse}. Forming $A_n x$ costs $10n \log_2 n + O(n)$ real floating point operations.

\textbf{Proof of Lemma 2.1.} Instead of the explicit $K_n$ in \eqref{eq:2.5} we start with an arbitrary real diagonal $n$-by-$n$ matrix $\Delta_n = \text{diag}(\delta_1, \ldots, \delta_n)$. Evidently, $F_n \Delta_n F_n^{-1} = (1/n) F_n \Delta_n F_n^*$ is Hermitian. The $(k,j)$-element of $T_n := F_n \Delta_n F_n^{-1}$ is given by

$$
t_{k,j}^{(n)} = \frac{1}{n} |F_n \Delta_n F_n^*|_{k,j} = \frac{1}{n} \sum_{\ell=0}^{n-1} f_{k\ell} \delta_{\ell,j} f_{j\ell} = \frac{1}{n} \sum_{\ell=0}^{n-1} \delta_{\ell} \omega_n^{k \ell} \omega_n^{-j \ell} = \frac{1}{n} \sum_{\ell=0}^{n-1} \delta_{\ell} \omega_n^{(k-j) \ell},
$$

where $\omega_n = e^{2\pi i/n}$. As the value of $t_{k,j}^{(n)}$ only depends on $k-j$, $T_n$ is a Toeplitz matrix. $T_n$ is even \textit{circulant}. This is easily verified by showing that $t_{1,0}^{(n)} = t_{0,n-1}^{(n)}$ and noting that $\omega_n^n = 1$. Finally, if $\delta_k = \delta_{n-k}$ for $k=1, \ldots, n-1$, then

$$
t_{k,j}^{(n)} = \sum_{\ell=0}^{n-1} \delta_{\ell} \omega_n^{(k-j) \ell} = \delta_0 + \delta_\omega \left((-1)^{k-j} + \sum_{\ell=1}^{n-1} \left( \delta_{\ell} \omega_n^{(k-j) \ell} + \delta_{n-\ell} \omega_n^{-(k-j) \ell} \right) \right)
$$

$$
= \delta_0 + \delta_\omega \left((-1)^{k-j} + \sum_{\ell=1}^{\frac{n}{2}-1} \delta_{\ell} \left( \omega_n^{(k-j) \ell} + \omega_n^{-(k-j) \ell} \right) \right)
$$

$$
= \delta_0 + \delta_\omega \left((-1)^{k-j} + \sum_{\ell=1}^{\frac{n}{2}-1} 2\delta_{\ell} \cos 2\pi (k-j) \ell/n. \right)
$$

So, $T_n$ is real symmetric and circulant. Notice that the diagonal elements of $K_n$ satisfy all the above assumptions on the $\delta_i$. Thus, $H_n = F_n K_n F_n^{-1}$ is real symmetric and circulant. \hfill \Box

\section{The numerical solution of the 1D problem.}

As the multiplication with the matrix $A_n = F_n K_n F_n^{-1} + V_n$ is cheap, eigensolvers that are commonly applied to compute a few of the eigenvalues of sparse matrices can be used to compute a few eigenvalues of the structurally sparse matrix $A_n$. In this section we will present results that we have obtained with subspace iteration (SIVIT), with the implicitly restarted Lanczos Algorithm (IRL) and with the Jacobi-Davidson algorithm (JDQR). With the small sized problems, full matrix methods can also be applied: Householder transformation to tridiagonal form followed by the symmetric tridiagonal QR algorithm (as in the original GROMOS \cite{GROMOS} implementation) or by bisection, respectively. Notice that in the QR algorithm the complete spectrum is computed!

When computing a few of the lowest eigenvalues of a sparse matrix eigenvalue problem

\begin{equation}
A_n x = \lambda x, \quad A_n = A_n^T,
\end{equation}

it is advantageous to apply a spectral transformation to get a reasonable speed of convergence \cite{21,18,19,12}. In the shift-and-invert approach \eqref{eq:3.1} is transformed into

\begin{equation}
(A_n - \sigma I)^{-1} x = \mu x, \quad \mu = \frac{1}{\lambda - \sigma}.
\end{equation}

The spectral transformation leaves the eigenvectors unchanged. The eigenvalues of \eqref{eq:3.1} close to $\sigma$ become the largest absolute of \eqref{eq:3.2}. In addition they are relatively well-separated which improves the speed of convergence of Krylov subspace methods. The cost of the improved convergence rate is solving a linear system of equations of the form $(A_n - \sigma I)x = b$ in each iteration step.

As is commonly done, we choose the shift \( \sigma \) below but close to the smallest eigenvalue \( \lambda_1 \). This makes the shifted matrix positive definite. We set \( \sigma = v_k := \min_i v_i \). Thus \( V_n - \sigma I_n \) is positive semidefinite with null space span\( \{e_k\} \). As \( F_nK_nF_n^{-1} \) is positive semidefinite with a different nullspace span\( \{1\} \), \( 1 = (1, \ldots, 1)^T \), we see that \( A_n \) is positive definite. Numerical experiments have shown that this choice of shift leads to a good convergence behavior.

We solve the system

\[
(A_n - \sigma I_n)x = (F_nK_nF_n^{-1} + V_n - \sigma I_n)x = b.
\]

by the preconditioned conjugate gradient method. When looking for a preconditioner it seems to be the most appropriate to choose a matrix, say \( M_n \), that has a similar structure as \( A_n \) but is easy to invert. We therefore determine \( M_n \) to be the circulant matrix that is closest to \( A_n \) with respect to the Frobenius norm,

\[
\|A_n - M_n\|_F = \left( \sum_{i,j=0}^{n-1} |a_{ij}^{(n)} - m_{ij}^{(n)}|^2 \right)^{1/2} = \min.
\]

It is easy to see, that all diagonals of \( M_n \) can be determined independently of each other. \( M_n \) has thus the same off-diagonals as \( A_n \). The value of the diagonal elements of \( M_n \) is the average of those of \( A_n \). Thus, the solution of (3.4) is

\[
M_n = F_nK_nF_n^{-1} + (\tilde{v} - \sigma)I_n = F_n(K_n + (\tilde{v} - \sigma)I_n)F_n^{-1}, \quad \tilde{v} = \frac{1}{n} \sum_{i=0}^{n-1} V(x_i).
\]

The most time consuming operations in the \( k \)-th step of the preconditioned conjugate gradient method are the computation of the residual and of the ‘preconditioned’ residual,

\[
r_k = b - (A_n - \sigma I_n)x_k, \quad z_k = M_n^{-1}r_k,
\]

respectively. With the above choice of shift the inverse of \( M_n \) exists and is

\[
M_n^{-1} = F_n(K_n + (\tilde{v} - \sigma)I_n)^{-1}F_n^{-1}.
\]

Notice that we can interpret \( \tilde{v} \) as a constant function approximating \( V(x) \). Higher order (Fourier) approximations are possible. The application of \( M_n^{-1} \) would then become more involved however.

The asymptotic convergence of the preconditioned conjugate gradient method for solving (3.3) is given by the factor [11, p.51]

\[
\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}
\]

where the generalized condition number \( \kappa = \kappa_{M_n}(M_n^{-1}A_n) = \|M_n^{-1}A_n\|_{M_n}, \|A_n^{-1}M_n\|_{M_n} = \lambda_n/\lambda_1 \) is the quotient of the largest and the smallest eigenvalue of \( M_n^{-1}A_n \). Here, \( \|\cdot\|_{M_n} \) is the matrix norm with respect to \( (x^*M_nx)^{1/2} \). We have the following

**Theorem 3.1.** If \( \sigma = \min V(x_j) \) then the generalized condition number \( \kappa(A_n, M_n) \) is bounded independently of the problem size,

\[
\kappa_{M_n}(M_n^{-1}A_n) \leq \frac{\left(\frac{\hbar^2}{2m} + \tilde{v} - \sigma\right)(\tau - \sigma)}{\frac{\hbar^2}{2m}(\tilde{v} - \sigma)}, \quad \text{for all } n = 2^q, \ q \in \mathbb{N}.
\]
Proof. The eigenvalues $\lambda_1$ and $\lambda_n$ are the extremal points of the range of the Rayleigh quotient $\rho(x)$,

$$
\lambda_1 \leq \rho(x) \leq \lambda_n, \quad \rho(x) := \frac{x^* A_n x}{x^* M_n x}, \quad x \neq 0.
$$

We give (positive) upper and lower bounds for $\rho(x)$. Let $\tau := \max V(x_j)$. We assume that $\tau > \sigma$ which implies $\bar{v} > \sigma$. (Otherwise, $M_n = A_n$ and the Theorem holds trivially.) We first derive an upper bound for $\rho(x)$.

$$
x^* A_n x \over x^* M_n x = x^* (F_n K_n F_n^{-1} + (V - \sigma) I) x
$$

$$
= 1 + \frac{x^* (V - \bar{v} I) x}{x^* (F_n K_n F_n^{-1} + (\bar{v} - \sigma) I) x} \leq 1 + \frac{\tau - \bar{v}}{\bar{v} - \sigma} = \frac{\tau - \sigma}{\bar{v} - \sigma}.
$$

As this inequality holds for all $x \neq 0$ we have

$$
\lambda_n = \|M_n^{-1} A_n\|_{M_n} = \max \rho(x) \leq \frac{\tau - \sigma}{\bar{v} - \sigma}.
$$

To get a lower bound for $\rho(x)$ we proceed as follows. Let $1 \in \mathbb{R}^n$ be the vector of all ones. We give lower bounds for the Rayleigh quotient separately for $x \in \text{span}\{1\}$ and for $x \in \text{span}\{1\}^\perp$. From the convexity of the field of values [15], i.e. the range of the Rayleigh quotient, it follows that the lower of the two bounds is a lower bound for $\lambda_1$ in (3.8).

First, let $x$ be a multiple of $1$. As $1^* (V - \bar{v} I) 1 = 0$ we immediately have

$$
x^* A_n x \over x^* M_n x = 1 + \frac{x^* (V - \bar{v} I) x}{x^* (F_n K_n F_n^{-1} + (\bar{v} - \sigma) I) x} = 1.
$$

Second, let $1^* x = 0$. Then

$$
x^* A_n x \over x^* M_n x = 1 + \frac{x^* (V - \bar{v} I) x}{x^* (F_n K_n F_n^{-1} + (\bar{v} - \sigma) I) x}
$$

$$
\geq 1 + \frac{(\sigma - \bar{v}) x^* x}{x^* (F_n K_n F_n^{-1} + (\bar{v} - \sigma) I) x} = 1 - \frac{(\bar{v} - \sigma) x^* x}{x^* F_n K_n F_n^{-1} x + (\bar{v} - \sigma) x^* x}
$$

$$
\geq 1 - \frac{\bar{v} - \sigma}{2 \pi m} = \frac{\beta^2}{2 \pi m} + (\bar{v} - \sigma).
$$

This last number is positive. As this inequality holds for all $x \neq 0$, we have

$$
\frac{1}{\lambda_1} = \|A_n^{-1} M_n\|_{M_n} = \min \frac{1}{\rho(x)} \leq \frac{\beta^2}{2 \pi m} + \frac{(\bar{v} - \sigma)}{2 \pi m}
$$

which gives the desired lower bound. \(\square\)

Remark 1. If we note that $\sigma \approx \min_x V(x)$, $\tau \approx \max_x V(x)$, and that $\bar{v}$ is determined by the $n$-point trapezoidal rule to approximate $(1/2\pi) \int_0^{2\pi} V(x) dx$ [26, p.106],

$$
\bar{v} = \frac{1}{n} \sum_{i=0}^{n-1} V(x_i) = \frac{1}{2 \pi} \int_0^{2\pi} V(x) dx + O(\frac{1}{n^2}),
$$

then we see that the bound in (3.7) is essentially independent of $n$. Thus, Theorem 3.1 implies that the number of iteration steps needed to solve (3.4) to a fixed accuracy does
not grow as the problem size increases. Further, the cost for solving (3.4) only grows like the cost for a matrix-vector multiplication with the matrix $M_n^{-1}A_n$ which is $\mathcal{O}(n \log_2 n)$ in our case.

We executed the experiments discussed in the sequel on one processor of a Sun Enterprise 3500 with six 336 Mhz UltraSparc processors and 3072 MB of main memory. The operating system was Solaris 2.6. The program was written in C. We however made extensive use of Fortran libraries as BLAS and LAPACK.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$n_{ev}$</th>
<th>SIVIT</th>
<th>IRL</th>
<th>IRLPR</th>
<th>JDQR</th>
<th>QR</th>
<th>bisection</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$t[\text{sec}]$</td>
<td>$n_{it}$</td>
<td>$t[\text{sec}]$</td>
<td>$n_{it}$</td>
<td>$t[\text{sec}]$</td>
<td>$n_{it}$</td>
</tr>
<tr>
<td>32</td>
<td>15</td>
<td>0.15</td>
<td>14.0</td>
<td>0.02</td>
<td>0.01</td>
<td>0.05</td>
<td>1.1</td>
</tr>
<tr>
<td>128</td>
<td>15</td>
<td>0.69</td>
<td>14.1</td>
<td>0.18</td>
<td>0.32</td>
<td>0.24</td>
<td>0.5</td>
</tr>
<tr>
<td>512</td>
<td>15</td>
<td>2.59</td>
<td>14.1</td>
<td>3.34</td>
<td>1.52</td>
<td>0.70</td>
<td>0.6</td>
</tr>
<tr>
<td>30</td>
<td>2.30</td>
<td>14.0</td>
<td>4.28</td>
<td>2.32</td>
<td>2.55</td>
<td>0.2</td>
<td>12.19</td>
</tr>
</tbody>
</table>

Table 3.1

Performance of the various eigensolvers for computing $n_{ev}$ eigenvalues 1D-problems of size $n$ with initial vectors chosen randomly.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$n_{ev}$</th>
<th>SIVIT</th>
<th>IRL</th>
<th>IRLPR</th>
<th>JDQR</th>
<th>QR</th>
<th>bisection</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$t[\text{sec}]$</td>
<td>$n_{it}$</td>
<td>$t[\text{sec}]$</td>
<td>$n_{it}$</td>
<td>$t[\text{sec}]$</td>
<td>$n_{it}$</td>
</tr>
<tr>
<td>32</td>
<td>15</td>
<td>0.09</td>
<td>10.0</td>
<td>0.01</td>
<td>0.00</td>
<td>0.03</td>
<td>0.0</td>
</tr>
<tr>
<td>128</td>
<td>15</td>
<td>0.19</td>
<td>10.3</td>
<td>0.16</td>
<td>0.22</td>
<td>0.15</td>
<td>0.0</td>
</tr>
<tr>
<td>512</td>
<td>15</td>
<td>0.97</td>
<td>10.0</td>
<td>3.04</td>
<td>1.12</td>
<td>0.37</td>
<td>0.9</td>
</tr>
<tr>
<td>30</td>
<td>2.60</td>
<td>9.4</td>
<td>3.93</td>
<td>2.00</td>
<td>1.55</td>
<td>0.0</td>
<td>11.51</td>
</tr>
</tbody>
</table>

Table 3.2

Performance of the various eigensolvers for computing $n_{ev}$ eigenvalues 1D-problems of size $n$ with ‘good’ initial vectors.

In Tables 3.1 and 3.2 we present timings of six algorithms for computing $n_{ev}$ eigenvalues and corresponding eigenvectors of (2.6) for three problem sizes $n$. The problem was a QD proton in a 1D harmonic oscillator[6]. The algorithms employed were

1. Subspace iteration with shift-and-invert also called simultaneous inverse vector iteration (SIVIT). For all subspace methods the dimensions of the subspaces are larger than the number of computed eigensolutions. With SIVIT we iterated with $2n_{ev}$ vectors. The linear systems of equations induced by the shift-and-invert spectral transformation are solved iteratively. The shift was chosen such that the system matrix becomes symmetric positive definite (spd). This made it possible to solve the system by the conjugate gradient method (CG) with the preconditioner $M_n$ introduced in (3.5).

2. The implicitly restarted Lanczos algorithm (IRL) as implemented in ARPACK [17]. With IRL the Krylov subspaces were generated directly by $A_n$. We varied the subspace dimension between $k_{\text{min}} = n_{ev}$ and $k_{\text{max}} = 2n_{ev}$, i.e., as soon as the dimension of the trial space reached $2n_{ev}$ we deflated the $n_{ev}$ least wanted eigenvalue approximations and continued the computation with the remaining $n_{ev}$-dimensional Krylov subspace.

3. A variant of IRL with partial re-orthogonalization (IRLPR). As the matrix-vector multiplication in our case is very cheap, the re-orthogonalization process becomes...
the dominant computational part in an ARPACK-like implementation. To reduce this cost, we introduced partial re-orthogonalization into IRL. Partial re-orthogonalization has been invented by Simon [22, 23, 12] to prevent at a reasonable cost the loss of orthogonality among the Krylov vectors in the Lanczos algorithm. Furthermore, we worked with much larger trial spaces than with the original IRL. Its dimension ranged from \( k_{\text{min}} = n_{\text{ev}} + 8 \) to \( k_{\text{max}} \). Initially, \( k_{\text{max}} = 2n_{\text{ev}} + 32 \). This value was increased by 4 after every deflation step up to \( k_{\text{max}} = 3n_{\text{ev}} + 40 \). These values have been found reasonable in experiments. Of course, their optimal values are problem dependent.

4. The Jacobi-Davidson algorithm with shift-and-invert (JDQR). In contrast to the situation in SIVIT, in Jacobi-Davidson algorithm the shift varies during the algorithm. As the system of equations that has to be solved is symmetric indefinite, we used the Conjugate Gradient Squared (CGS) method to solve it. We used the same preconditioner \( M_n \) as with SIVIT. As with IRL we varied the subspace dimensions between \( k_{\text{min}} = n_{\text{ev}} \) and \( k_{\text{max}} = 2n_{\text{ev}} \).

5. The QR algorithm This is the classical algorithm to compute the complete spectral decomposition of a full (dense) matrix [1]. It proceeds in two steps. First the matrix \( A_n \) is reduced to a similar tridiagonal form; then the symmetric tridiagonal QR algorithm is applied. This is the algorithm used in the original GROMOS [6] implementation.

6. Bisection. If only a part (less than 30%) of the spectrum of a full matrix is needed it is recommended [1] to replace the symmetric tridiagonal QR algorithm by tridiagonal bisection. The reduction to tridiagonal form remains the same as with the QR algorithm.

For all eigensolvers we used a relative error tolerance of \( 10^{-6} \), which is accurate enough, if compared to approximation errors resulting from other parts of the calculation. In the iterative equation solvers, the relative residual was reduced to \( 10^{-8} \). The timings of Table 3.1 refer to the situation where no approximations of the searched eigenvectors are known. Here we started the iterations with random vectors. However, the vectors were chosen to be equal in the various algorithms. For the timings of Table 3.2 we started with the eigenvectors of the matrix of the previous time step. These numbers are much more relevant because in real simulations one actually has to solve a sequence of eigenvalue problems that differ only in the diagonal matrix \( V_n \). The columns indicated by \( \bar{n}_u \) give the average number of iterations needed by the linear system solver to reduce the relative residual to \( 10^{-8} \). The optimality of our preconditioner is justified by the small variance of \( \bar{n}_u \). Notice that this number is very small with JDQR as only little accuracy is required in the initial phase of the algorithm. Only at the end the maximal accuracy requirement is enforced.

The results show that for the small and medium size problems bisection is fastest. In these cases a considerable fraction of of the eigenpairs is requested. For the small problem size, the difference with the ordinary QR algorithm and with IRLPR is not big, however. With the largest problem size the full matrix methods cannot compete any more due to their \( O(n^3) \) complexity. Besides their high flop count the full matrix methods also consume much memory. Subspace methods have lower computational cost. If only 15 eigenvalues are desired then Jacobi-Davidson is clearly fastest. If 30 eigenpairs are desired then IRL catches up and is even faster if good eigenvector approximations are lacking.

Notice that SIVIT exploits good starting vectors very well. It is second fastest if only 15 eigenpairs are computed. Also JDQR makes good use of good initial vectors. In particular the number \( \bar{n}_u \) of ‘inner iterations’ is very small or even negligible.

*ETH Zürich, CS Technical Report #333, Institute of Scientific Computing, January 13, 2000*
performs best if good initial vectors are available. However, its computation time increases by about a factor of four if \( n_{\text{ev}} \) is doubled. This is because of the many projections needed in Jacobi-Davidson. For larger \( n_{\text{ev}} \) IRLPR will be competitive.

Notice that IRL and IRLPR perform in a similar way in the medium size problem. For the large problem size IRL performs poorly due to the excessive number of re-orthogonalizations. At the same time the short execution times of IRLPR show that partial re-orthogonalization is effective. As the trial spaces with IRLPR are very big, re-orthogonalization is extremely expensive. The timings indicate that only a small number of re-orthogonalizations take place.

4. The 3D problem. The Schrödinger equation (1.1) for a 3-dimensional particle of mass \( m \) that is trapped in a cubic box is given by

\[
\mathcal{H}u(x) = -\frac{\hbar^2}{2m} \Delta u(x) + V(x)u(x) = \lambda u(x), \quad x \in G := (0, 2\pi)^3,
\]

where \( u(x) \) satisfies periodic boundary conditions. We approximate \( u(x) \) as a tensor product of Fourier polynomials of order \( n, n = 2^q, \)

\[
u(x) = \sum_{k_d = -\frac{n}{2} + 1}^{\frac{n}{2}} c_k e^{ikx}.
\]

Thus

\[-\Delta u(x) = \sum_{k_d = -\frac{n}{2} + 1}^{\frac{n}{2}} |k|^2 c_k e^{ikx}, \quad |k|^2 = k_1^2 + k_2^2 + k_3^2.\]

Solving (4.2) by collocation at the points \( x_j = 2\pi/n(j_1, j_2, j_3)^T, j_d = 0, \ldots, n - 1, d = 1, \ldots, 3, \)
yields the \( n^3 \) equations

\[
\sum_{k_d = -\frac{n}{2} + 1}^{\frac{n}{2}} \left( \frac{\hbar^2}{2m} |k|^2 + V(x_j) - \lambda \right) c_k e^{ikx} = \frac{2\pi}{n} |V_n|, \quad j_d = 0, \ldots, n - 1, \quad d = 1, \ldots, 3.
\]

Transformations analogous to those from (2.3) to (2.4) and from (2.5) to (2.6), respectively, lead to the matrix eigenvalue problem

\[
\tilde{A}_n x = (\tilde{F}_n \tilde{K}_n \tilde{F}_n^{-1} + \tilde{V}_n)x = \lambda x
\]

where

\[
\tilde{F}_n = F_n \otimes F_n \otimes F_n, \quad \tilde{K}_n = I_n \otimes I_n \otimes K_n + I_n \otimes K_n \otimes I_n + K_n \otimes I_n \otimes I_n, \quad \tilde{V}_n = \text{diag}(V(x_{0,0,0}), \ldots, V(x_{(n-1,n-1,n-1)})).
\]

\( \tilde{V}_n \) contains the values of \( V(x) \) at the grid points \( x_{i,j,k} \) that are arranged in lexicographic order. Setting \( H_n = F_n K_n F_n^{-1} \) as in (2.6) we can write

\[
\tilde{H}_n = \tilde{F}_n \tilde{K}_n \tilde{F}_n^{-1} = I_n \otimes I_n \otimes H_n + I_n \otimes H_n \otimes I_n + H_n \otimes I_n \otimes I_n.
\]

The zero pattern of the three summands in (4.5) is depicted in Fig. 4.1. \( \tilde{A}_n \) is a sparse matrix; it has only \( 3n^4 - 2n^3 \) nonzero elements, giving a sparsity of about \( 3/n^2 \). Forming the matrix-vector product \( \tilde{A}_n x \) requires \( 6n^4 - 4n^3 \) floating point operations (flops) if \( A \)'s zero pattern is taken into account. This can be improved to \( 30n^3 \log_2 n \) flops if the FFT is employed.

5. The numerical solution of the 3D problem.

5.1. The matrix-vector product with $\tilde{A}_n$. The multiplication with $\tilde{A}_n$ in (4.4) with a vector $x \in \mathbb{R}^{n^3}$ can be executed in different ways. The multiplication with $\tilde{V}_n$ is straightforward. We therefore can limit our attention to the multiplication with $\tilde{H}_n$. On one hand we employed the fast Fourier transform (FFT),

$$\tilde{H}_n = (F_n \otimes F_n \otimes F_n)(I_n \otimes I_n \otimes K_n + I_n \otimes K_n \otimes I_n + K_n \otimes I_n \otimes I_n)(F_n^{-1} \otimes F_n^{-1} \otimes F_n^{-1})$$

where the 3-dimensional Fourier transform was implemented as

$$F_n \otimes F_n \otimes F_n = (F_n \otimes I_n \otimes I_n)(I_n \otimes F_n \otimes I_n)(I_n \otimes I_n \otimes F_n).$$

The Fourier transforms were executed by the FFTW implementation of the fast Fourier transform (FFT), an implementation that adapts itself to the underlying hardware [10]. FFTW outperformed the vendor provided FFTs in most of the problem sizes. The complexity of this implementation is $n^3(30 \log_2 n + 12)$ flops.

On the other hand we multiplied by

$$I_n \otimes I_n \otimes H_n + I_n \otimes H_n \otimes I_n + H_n \otimes I_n \otimes I_n.$$  

(5.1)

Here, again there are a number of possible implementations. As $H_n$ is a small $n$-by-$n$ matrix, $n \leq 20$, it could be stored as full matrix. Operating with $H_n$ then could be performed with a call to the basic linear algebra subroutine $\text{dgemm}$. Notice that each of the three terms in (5.1) amounts to multiplying $H_n$ with $n^2$ vectors. The first and the third term can actually be implemented by a single invocation of $\text{dgemm}$ by varying the access pattern of $x$ which is treated as a matrix inside the $\text{dgemm}$. Operating with $I_n \otimes H_n \otimes I_n$ was implemented through $n$ calls to $\text{dgemm}$, each multiplying $H_n$ with $n^2$ elements in $x$ that were arranged as a $n$-by-$n$ matrix (without actually moving elements of $x$).

It turned out to be advantageous to exploit the 2-by-2 block structure of $H_n$,

$$H_n = \begin{bmatrix} H_n^{(1)} & H_n^{(2)} \\ H_n^{(2)} & H_n^{(1)} \end{bmatrix}, \quad H_n^{(1)}, H_n^{(2)} \text{ symmetric Toeplitz},$$

and quadrupling the number of matrix-vector multiplies while halving their order. The complexity of this implementation is $3n^4 + 6n^3$ flops in contrast to the straightforward implementation which is $6n^4$ flops.

We performed timings of all the described versions of matrix-vector multiplications $\tilde{A}_n x$. We finally chose to use FFTW for this purpose. FFTW performed best in the average on all compute platforms and was always among the best.

5.2. An ADI-Type preconditioner for the 3D problem. As in the 1-dimensional case we want to solve the eigenvalue problem (4.4) by the shift and invert approach. The system of equations

\[ \tilde{A}_\sigma x := (\tilde{A}_n - \sigma I_n) x = b \]

is again solved by the preconditioned conjugate gradient method. A direct adaptation of the 1D construction would lead to a preconditioner of the form \( \tilde{F}_n^{-1} + \bar{v} I_n \), where \( \bar{v} \) is the average of the diagonal elements of \( \tilde{V} \). This preconditioner turned out not to be satisfactory. By replacing \( \tilde{V}_n \) by a multiple of the identity the 3-dimensional character of the equations gets lost.

We therefore took an approach similar to the alternating direction iteration (ADI) method known from the Poisson equation in which alternately one-dimensional problems are solved [4, 27, 7]. In [2] we suggested to split equation (5.2) in three ways such that one of the three summands in (4.5) appears on the left the other two terms on the right of the equality sign. One step of the ADI-like (stationary) iteration method then consists of solving the three systems of equations

\[ M_1 x^{(k+1/3)} = (M_1 - \tilde{A}_\sigma) x^{(k)} + b, \]

\[ M_2 x^{(k+2/3)} = (M_2 - \tilde{A}_\sigma) x^{(k+1/3)} + b, \]

\[ M_3 x^{(k+1)} = (M_3 - \tilde{A}_\sigma) x^{(k+2/3)} + b, \]

where

\[ M_1 = \tilde{H}_1 + \tilde{V}_1 + r I_n - \sigma I_n, \quad \tilde{H}_1 = I_n \otimes I_n \otimes H_n, \]

\[ M_2 = \tilde{H}_2 + \tilde{V}_2 + r I_n - \sigma I_n, \quad \tilde{H}_2 = I_n \otimes H_n \otimes I_n, \]

\[ M_3 = \tilde{H}_3 + \tilde{V}_3 + r I_n - \sigma I_n, \quad \tilde{H}_3 = H_n \otimes I_n \otimes I_n. \]

\( r \) is a positive parameter whose value will be determined later. If ADI is used as a stationary iteration method \( r \) is varied in each iteration step [27]. The diagonal elements of the diagonal matrices \( \tilde{V}_1, \tilde{V}_2, \tilde{V}_3 \) contain averages of values of \( \tilde{V} \). \( \tilde{V}_1, \) e.g., can be written in the form

\[ \tilde{V}_1 = \tilde{V}_1 \otimes I_n, \quad \tilde{V}_1 \in \mathbb{R}^{n^2 \times n^2}. \]

The \( i \)-th diagonal element of \( \tilde{V}_1 \) contains the averages of \( V(x) \) along the first coordinate direction,

\[ [\tilde{V}_1]_{ii} = \frac{1}{n} \sum_{\ell=0}^{n-1} V(x_\ell, x_j, x_k), \quad i = j + nk, \quad 0 \leq j, k < n. \]

In a similar way, the diagonal elements of \( \tilde{V}_2 \) and \( \tilde{V}_3 \) are averages of \( V(x) \) along the second and third coordinate direction, respectively.

Notice that we may write (5.3) in the form

\[ x^{(k+1/3)} = x^{(k)} + M_1^{-1} x^{(k)} \]

\[ x^{(k+2/3)} = x^{(k+1/3)} + M_2^{-1} x^{(k+1/3)} \]

\[ x^{(k+1)} = x^{(k+2/3)} + M_3^{-1} x^{(k+2/3)}. \]

with \( r^{(m)} := b - \tilde{A}_x x^{(m)} \) being the residual in \( x^{(m)} \). Each of the three systems of equations consists of \( n^2 \) independent systems of equations of order \( n \) that differ only in the diagonal. They can be solved in parallel.

The ADI-like iteration (5.5) can be collected in one equation,

\[
x^{(k+1)} = x^{(k)} + (I - (I - M_3^{-1} \tilde{A}_\sigma)(I - M_2^{-1} \tilde{A}_\sigma)(I - M_1^{-1} \tilde{A}_\sigma)) \tilde{A}_\sigma^{-1} r^{(k)}.
\]

As the \( M_i \) do not commute, the matrix that is multiplied with \( r^{(k)} \) is not symmetric positive definite. Therefore, it is not suited right-away as a preconditioner for the conjugate gradient method. We can however premultiply it with its transpose (adjoint) to get an admissible (symmetric positive definite) preconditioner [4, p.289],

\[
\begin{align*}
x^{(k+1/6)} & = x^{(k)} + M_1^{-1} r^{(k)} \\
x^{(k+3/6)} & = x^{(k+2/6)} + M_3^{-1} r^{(k+2/6)} \\
x^{(k+5/6)} & = x^{(k+4/6)} + M_2^{-1} r^{(k+4/6)} \\
\end{align*}
\]

or

\[
x^{(k+1)} = x^{(k)} + (I - (I - M_3^{-1} \tilde{A}_\sigma)(I - M_2^{-1} \tilde{A}_\sigma)(I - M_1^{-1} \tilde{A}_\sigma)) \tilde{A}_\sigma^{-1} r^{(k)}.
\]

\[
= : x^{(k)} + M^{-1} r^{(k)}.
\]

\( M^{-1} \) can alternatively be written in the form

\[
M^{-1} = \tilde{A}_\sigma^{-1} - \tilde{A}_\sigma^{-1/2}(I - \tilde{A}_\sigma^{-1/2} M_1^{-1} \tilde{A}_\sigma^{-1/2})(I - \tilde{A}_\sigma^{-1/2} M_2^{-1} \tilde{A}_\sigma^{-1/2})(I - \tilde{A}_\sigma^{-1/2} M_3^{-1} \tilde{A}_\sigma^{-1/2})
\]

\[
\cdot (I - \tilde{A}_\sigma^{-1/2} M_3^{-1} \tilde{A}_\sigma^{-1/2})(I - \tilde{A}_\sigma^{-1/2} M_2^{-1} \tilde{A}_\sigma^{-1/2})(I - \tilde{A}_\sigma^{-1/2} M_1^{-1} \tilde{A}_\sigma^{-1/2}) \tilde{A}_\sigma^{-1/2}
\]

The thus defined \( M^{-1} \) (or \( M \)) is evidently symmetric. We use this matrix as a preconditioner for the conjugate gradient method in the following way when solving the second equation in (3.6). We solve \( \tilde{A}_\sigma \zeta = r_k \) approximatively by one step of the ADI-like iteration (5.6). The initial vector is \( \zeta^{(0)} = 0 \) and \( \zeta^{(k)} = \zeta^{(1)} \).

Remark 2. A different symmetric preconditioner is obtained if \( M_3^{-1} \) in (5.6) is applied only once. This symmetrizing technique is known from overlapping Schwarz preconditioning [25, p.14]. This preconditioner has of only five substeps,

\[
M^{-1} = \tilde{A}_\sigma^{-1} - \tilde{A}_\sigma^{-1/2}(I - \tilde{A}_\sigma^{-1/2} M_1^{-1} \tilde{A}_\sigma^{-1/2})(I - \tilde{A}_\sigma^{-1/2} M_2^{-1} \tilde{A}_\sigma^{-1/2})(I - \tilde{A}_\sigma^{-1/2} M_3^{-1} \tilde{A}_\sigma^{-1/2})
\]

\[
\cdot (I - \tilde{A}_\sigma^{-1/2} M_3^{-1} \tilde{A}_\sigma^{-1/2})(I - \tilde{A}_\sigma^{-1/2} M_2^{-1} \tilde{A}_\sigma^{-1/2})(I - \tilde{A}_\sigma^{-1/2} M_1^{-1} \tilde{A}_\sigma^{-1/2}) \tilde{A}_\sigma^{-1/2}
\]

One application of (5.9) is slightly cheaper than one application of the original preconditioner in (5.8). ⊛

Remark 3. Notice that we are free in how we select the sequence in which we go ‘through the dimensions’. Here, we chose the natural sequence 1-2-3 as there is formally no distinguished direction in the potential \( V(x) \). In practice however the selection of the sequence can be crucial for the performance of the preconditioner. ⊛

To investigate the convergence behaviour of the preconditioned conjugate gradient algorithm we estimate the generalized condition number of \( M^{-1} \tilde{A}_\sigma \). As the eigenvalue problems \( \tilde{A}_\sigma x = \lambda M x \) and \( M^{-1} \zeta = \lambda \tilde{A}_\sigma^{-1} \zeta \) are similar, the inclusion (3.8) also holds for the Rayleigh quotient of the latter eigenvalue problem,

\[
0 < \lambda_1 \leq \rho'(z) := z^* M^{-1} z z^* \tilde{A}_\sigma^{-1} z < \lambda_n, \quad z \neq 0,
\]

which implies \( \kappa \leq \lambda_i/A_1 \). According to (5.8) we rewrite \( \rho'(z) \) as

\[
\rho'(z) = 1 - \frac{y^*(I - \tilde{A}_\sigma^{1/2} M_1^{-1} \tilde{A}_\sigma^{1/2})^2 y}{y^* y}
\]

(5.10)

where

\[
y = \tilde{A}_\sigma^{-1/2} z, \quad y_1 = (I - \tilde{A}_\sigma^{1/2} M_1^{-1} \tilde{A}_\sigma^{1/2}) y,
\]

\[
y_2 = (I - \tilde{A}_\sigma^{1/2} M_2^{-1} \tilde{A}_\sigma^{1/2}) y_1 = (I - \tilde{A}_\sigma^{1/2} M_2^{-1} \tilde{A}_\sigma^{1/2})(I - \tilde{A}_\sigma^{-1/2} M_1^{-1} \tilde{A}_\sigma^{-1/2}) y.
\]

Clearly, \( \rho'(z) \leq 1 \). We want to show that \( \rho'(z) \) is bounded away from zero. We do this by treating the three quotients in (5.10) individually showing that for sufficiently large \( r \) there are numbers \( \tilde{\lambda}_1(r), \tilde{\lambda}_2(r), \tilde{\lambda}_3(r) \) such that

\[
\rho_i(y) := \frac{y^*(I - \tilde{A}_\sigma^{1/2} M_i^{-1} \tilde{A}_\sigma^{1/2})^2 y}{y^* y} \leq \tilde{\lambda}_i(r) < 1, \quad \forall y \neq 0, \quad i = 1, 2, 3.
\]

Then, we get

(5.11) \[ 0 < 1 - \tilde{\lambda}_1(r) \tilde{\lambda}_2(r) \tilde{\lambda}_3(r) \leq \rho'(z) \leq 1. \]

Of course, we would like the above lower bound to be as large as possible. To investigate the three individual Rayleigh quotients we can w.l.o.g. let \( i = 1 \). \( \rho_1(y) \) is the Rayleigh quotient of the eigenvalue problem

(5.12) \[ (I - \tilde{A}_\sigma^{1/2} M_1^{-1} \tilde{A}_\sigma^{1/2})^2 y = \tilde{A}_\sigma^{1/2}(I - M_1^{-1} \tilde{A}_\sigma) \tilde{A}_\sigma^{-1/2} y = \lambda y. \]

By substituting \( z = \tilde{A}_\sigma^{-1/2} y \) we get the eigenvalue problem

(5.13) \[ (I - M_1^{-1} \tilde{A}_\sigma) z = \lambda z \]

which has the same eigenvalues as (5.12). The eigenvalues of (5.13) are the squares of the eigenvalues of

(5.14) \[ (I - M_1^{-1} \tilde{A}_\sigma) z = \mu z \]

or, equivalently, of

(5.15) \[ (M_1 - \tilde{A}_\sigma) z = \mu M_1 z. \]

The Rayleigh quotient corresponding to (5.15) is

(5.16) \[ \frac{z^* (M_1 - \tilde{A}_\sigma) z}{z^* M_1 z}. \]

To improve the quality of the estimates we consider as in the 1D case the subspaces span\{1\}, \( 1 = (1, \ldots, 1)^* \), and its orthogonal complement separately.

Let first \( x = 1 \). \( 1 \) is in the null space of all the \( H_i \) and further, as in one dimension, \( 1^*(V - \nabla_1)1 = 0 \). As \( \nabla_1 - \sigma I \) is positive definite, we get

\[
1^* M_1 1 = (r + \bar{v} - \sigma) \|1\|^2, \quad \bar{v} = \sum_{i=1}^{n^3} V_{ii} = \sum_{i=1}^{n^3} (\nabla_1)_{ii},
\]

\[ ETH Zürich, CS Technical Report #333, Institute of Scientific Computing, January 13, 2000 \]
and
\begin{equation}
\frac{1^* (M_1 - \tilde{A}_d)}{1^* M_1} = \frac{r}{r + v - \sigma}.
\end{equation}

Now let $x^* 1 = 0$. With the notation of (5.4) we have
\begin{equation}
\frac{x^* (M_1 - \tilde{A}_d)x}{x^* M_1 x} = \frac{x^* (H_1 + V_1 + (r - \sigma)I - H_1 - H_2 - H_3 - V + \sigma I)x}{x^* (H_1 + V_1 + (r - \sigma)I)x} \\
= \frac{x^* (rI - H_2 - H_3 + (V_1 - V))x}{x^* (rI + H_1 + (V_1 - \sigma I))x} \\
\leq \frac{x^* (rI - 2k_1 + (V_1 - V))x}{x^* (rI + k_1 + (V_1 - \sigma I))x} \\
= \max_i \frac{r - 2k_1 + (\sigma I_{ii} - V_{ii})}{r + k_1 + (V_1)_{ii} - \sigma} =: \tilde{\mu}_1(r).
\end{equation}

Both upper bound $\tilde{\mu}_1(r)$ and lower bound $\tilde{\mu}_1(r)$ increase with $r$ and converge to 1 from below as $r$ tends to infinity. $\tilde{\mu}_1$ is positive for $r > 2k_1$ and less than 1. $\tilde{\mu}_1$ is negative for $r = 0$. If $r$ is chosen so big that the lower bound is above $-1$ then the eigenvalues of (5.13) are bounded by $\tilde{\lambda}_1(r) = \bar{p}_i(r)^2 := \max\{|\tilde{\mu}_i(r)|, \tilde{\mu}_i(r)\}^2 < 1$. In a similar way we can bound $\rho_2(z)$ and $\rho_3(z)$.

By construction, $\tilde{\mu}_i(r) < \tilde{\mu}_i(r)$. In particular, if $r > r_i$ where $-\tilde{\mu}_i(r_i) = \tilde{\mu}_i(r_i)$, then $|\tilde{\mu}_i(r)| \leq \tilde{\mu}_i(r)$. If such an $r_i$ does not exist we set $r_i = 0$. From (5.18) we see that $\tilde{\mu}_i(r_i)$ does not depend on $n$.

If we set $r_0 := \max\{r_1, r_2, r_3\}$, then we have
\begin{equation}
\text{THEOREM 5.1. Let } \sigma = \min V(x_j) \text{ and let } M = M(r_0). \text{ The generalized condition number } \kappa(\tilde{A}_d, M) \text{ is bounded independently of the problem size,}
\end{equation}
\begin{equation}
\kappa = \kappa_M(r_0)(M(r_0)^{-1}\tilde{A}_d) \leq \frac{1}{1 - \tilde{\mu}_1(r_0)^2\tilde{\mu}_2(r_0)^2\tilde{\mu}_3(r_0)^2}
\end{equation}
where $r_0 := \max\{r_1, r_2, r_3\}$ and the $r_i$ are defined by
\[ r_i := \max\{0, \{r > 0\} - \tilde{\mu}_i(r) = \tilde{\mu}_i(r)\} \].

\textbf{Remark 4.} A better bound than in (5.20) may be obtained if the function
\[ \prod_{i=1}^{3} \bar{p}_i(r)^2, \quad \bar{p}_i(r) := \max\{|\tilde{\mu}_i(r)|, \tilde{\mu}_i(r)\}. \]
is directly minimized for $r \geq 0$. \qed
We now turn to the alternative preconditioner (5.9). Instead of the Rayleigh quotient in (5.10) we have to deal with

\[ \rho''(z) = \frac{y^*(I - \tilde{A}^3 M_1^{-1} \tilde{A}^2) y}{\rho_1(y)} \frac{y_1^*(I - \tilde{A}^3 M_2^{-1} \tilde{A}^2) y_1}{\rho_2(y_1)} \frac{y_2^*(I - \tilde{A}^3 M_3^{-1} \tilde{A}^2) y_2}{\rho_3(y_2)} \]

The estimates for \( \rho_1(y) \) and \( \rho_2(y_1) \) carry over from (5.18)-(5.19). \( \rho_3''(y_2) \) is up to a change of variables the Rayleigh quotient given in (5.16) with \( M_1 \) replaced by \( M_3 \). Hence,

\[ 1 - \tilde{\mu}_1(r)^2 \tilde{\mu}_2(r)^2 \tilde{\mu}_3(r) \leq \rho_3''(z) \leq 1 - \tilde{\mu}_1(r)^2 \tilde{\mu}_2(r)^2 \tilde{\mu}_3(r). \]

Notice that \( \tilde{\mu}_3(r) \) may be negative. In fact this is the interesting situation. If \( \tilde{\mu}_3(r) \) for some reason approaches -1 only very slowly then the \( r_0 \) in Theorem 5.1 will be very large. This means that the \( \tilde{\mu}_3(r) \) are all very close to 1 implying a high condition number \( \kappa \). If we can choose \( r_0 \) much smaller, \( \tilde{\mu}_3(r) \) may still be below -1, but \( 1 - \tilde{\mu}_1(r)^2 \tilde{\mu}_2(r)^2 \tilde{\mu}_3(r) \) is much further away from 0 leading to a much better condition number. We formulate

**Theorem 5.2.** The condition number

\[ \kappa \leq \frac{1 - \tilde{\mu}_1(r)^2 \tilde{\mu}_2(r)^2 \tilde{\mu}_3(r)}{1 - \tilde{\mu}_1(r)^2 \tilde{\mu}_2(r)^2 \tilde{\mu}_3(r)} \]

depends "slightly" on the problem size \( n \) as the numerator in (5.22) contains the term \( \frac{\kappa^2}{2m} \left( \frac{n}{2} \right)^2 \).

5.3. Numerical experiments. In this section we report on experiments with a realistic example from mixed QM/CM simulation [6]. There is a single quantum particle, an excess proton, immersed in fluid water (at 300 K) modeled by 216 water molecules. The proton state is modeled by a tensor product of three Fourier polynomials of degree \( n \), cf. (4.2). In our examples \( n \) takes on the values 12, 16, and 18, giving rise to matrix orders \( N = n^3 = 1728, 4096, \) and 5832, respectively. The computations were performed on the same computer platform as the 1D problems, Sun Enterprise 3500. For details see section 3.

We computed the 15 and 30, respectively, smallest eigenvalues and corresponding eigenvectors with the methods that we found the most promising from the 1D experiments, subspace iteration (SIVIT), the Jacobi-Davidson (JDQR) algorithm, and the implicitly restarted Lanczos algorithm with partial re-orthogonalization.

Table 5.1 shows the timings for eigensolvers when started with randomly chosen starting vectors. Table 5.2 shows the timings for solving (almost) the same problems with initial vectors that are the eigenvectors of the eigenvalue problem of the previous time step. The execution times \( t \) are given in seconds. \( n_{\text{cg}} \) is the number of times the (preconditioned) conjugate gradient algorithm was invoked. With SIVIT, this happens \( 2n_{\text{it}} \) times in each update of the trial space. The columns indicated by \( n_{\text{it}} \) give the average number of iterations of pcg until convergence. Notice, that in JDQR the error tolerance in the conjugate gradient method varies. Being loose initially, the error tolerance is tightened in each inner iteration [3, 9]. Therefore, the number of iterations until convergence of cg is not constant. We only list timings with an ADI-like preconditioners of type (5.9). The subscripts in ADI\(_{13231} \) indicate the sequence in which the \( G_i \) have been applied. We set the parameter \( r = 500 \), a value that was found through experiments. The results reported in [2] show that the preconditioners of the form (5.8) do not perform as well as those of the form (5.9), whence they are omitted here. In IRLPR there are no systems of equations to
be solved. So, the columns $n_{cg}$ and $\bar{n}_{it}$ are empty. With IRLPR the dimension of the trial space oscillates between $k_{\text{min}} = 12 + 8$ and $3n_{ev} + 40$, i.e. between 23 and 95 for $n_{ev} = 15$ and between 38 and 130 for $n_{ev} = 30$. To have a comparison with the other methods that use a much lower number of vectors we reran IRLPR with $k_{\text{max}} = 2k_{\text{min}} = 2n_{ev}$. These times are denoted by IRLPR$^-$. They indicate that IRLPR gains its speed from the large dimension of the trial space. JDQR converges faster than IRLPR$^-$ if the work space is restricted. This is in agreement with similar observations made in [3], where JDQR and ARPACK were compared with the block Lanczos algorithm.

The implicitly restarted Lanczos algorithm, IRLPR, turned out to be fastest in all cases. If only 15 eigenvalues are desired, JDQR comes close, but as in the 1D case performs poorer if 30 eigenvalues are desired.

Of all preconditioners the cheap diagonal preconditioner leads to the fastest solver. Compared with no preconditioning the average number of iterations $\bar{n}_{it}$ is reduced substantially as $A - \sigma I$ is quite diagonally dominant. $\bar{n}_{it}$ increases with the problem size but only slowly. This is probably true because in these experiments the very large values in the potential $V$ have been replaced by a maximal value, thus preventing the condition number of $A$ from growing with $n$.

The iteration counts show that the ADI-type preconditioner $ADI_{12321}$ is effective in that $\bar{n}_{it}$ is independent of the problem size. With respect to the diagonal preconditioner the $\bar{n}_{it}$ is however reduced by only a factor 2 or even less. The ADI-type preconditioner is therefore unsuccessful with respect to execution time. Because of the high (computational) complexity of this preconditioner the execution time is increased considerably.

In contrast to the numbers in Tab. 5.1 the numbers in Tab. 5.2 have been obtained with good initial vectors, i.e. with the eigenvectors of the nearby eigenvalue problem of the previous time step. One notices that SIVIT exploits the good initial vectors much better.

---

**Table 5.1**

Performance of the various eigensolvers for computing $n_{ev}$ eigenvalues of the 3D-problem of size $n^3$ with randomly chosen initial vectors.

<table>
<thead>
<tr>
<th>eigensolver/preconditioner</th>
<th>$n_{ev}$</th>
<th>$n = 12$</th>
<th>$n = 16$</th>
<th>$n = 18$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$n_{cg}$</td>
<td>$\bar{n}_{it}$</td>
<td>$n_{cg}$</td>
<td>$\bar{n}_{it}$</td>
</tr>
<tr>
<td>SIVIT/none</td>
<td>15</td>
<td>97</td>
<td>80</td>
<td>257</td>
</tr>
<tr>
<td>SIVIT/diagonal</td>
<td>15</td>
<td>34</td>
<td>1050</td>
<td>26</td>
</tr>
<tr>
<td>SIVIT/ADI$_{12321}$</td>
<td>15</td>
<td>516</td>
<td>1050</td>
<td>18</td>
</tr>
<tr>
<td>JDQR/none</td>
<td>15</td>
<td>12</td>
<td>152</td>
<td>40</td>
</tr>
<tr>
<td>JDQR/diagonal</td>
<td>15</td>
<td>4</td>
<td>148</td>
<td>3</td>
</tr>
<tr>
<td>JDQR/ADI$_{12321}$</td>
<td>15</td>
<td>51</td>
<td>152</td>
<td>3</td>
</tr>
<tr>
<td>IRLPR/—</td>
<td>15</td>
<td>3</td>
<td>9</td>
<td>148</td>
</tr>
<tr>
<td>IRLPR$^-$/—</td>
<td>15</td>
<td>28</td>
<td>88</td>
<td>148</td>
</tr>
</tbody>
</table>

---

than the other methods. The execution times are reduced by at least a factor 3. With JDQR this factor is only about 1.2. A closer look at the convergence history of JDQR reveals that only the computation of the first eigenpair is sped up. After this, the quality of the trial space is not affected anymore by the starting vectors. A comparison with SIVIT indicates that a blocked version of JDQR should be most advantageous. Further experiments showed that this is not true.

IRLPR is insensitive to good starting vectors. As the IRLPR iteration starts with a single vector the ‘good’ initial vector was set to be the average of the eigenvectors computed in the previous time step.

6. Conclusions. The implicitly restarted Lanczos (IRL) algorithm turned out to be the most effective solver for the eigenvalue problems we had to solve. These problems are characterized by a very cheap matrix vector product based on a large number of FFTs. For IRL to be successful it was important to eliminate as much as possible the time consuming re-orthogonalizations. This could be achieved by introducing partial re-orthogonalization into IRL. The reduced number of re-orthogonalizations makes it possible to work with large dimensional trial spaces which leads to fast convergence, cf. [14].

The other methods investigated, Jacobi-Davidson algorithm and subspace iteration, were applied with a shift-and-invert spectral transformation. The resulting systems of equations were solved by preconditioned conjugate gradient methods. An ADI-type preconditioners introduced in this paper was only partially successful. It turned out to be optimal as claimed but its application is too time consuming to be effective for these relatively small problem sizes. A further drawback of the ADI-type preconditioners is its crucial dependence on a parameter that is difficult to determine. The best results with subspace iteration as well as with the Jacobi-Davidson algorithm have been obtained with simple diagonal preconditioning.

Table 5.2

<table>
<thead>
<tr>
<th>eigensolver/preconditioner</th>
<th>$n_{ev}$</th>
<th>$n = 12$</th>
<th>$n = 16$</th>
<th>$n = 18$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$t$ [sec]</td>
<td>$n_{cg}$</td>
<td>$\bar{n}_{it}$</td>
<td>$t$ [sec]</td>
</tr>
<tr>
<td>SIVIT/none</td>
<td>15</td>
<td>26</td>
<td>75</td>
<td>15</td>
</tr>
<tr>
<td>SIVIT/diagonal</td>
<td>15</td>
<td>10</td>
<td>300</td>
<td>26</td>
</tr>
<tr>
<td>SIVIT/ADI$_{13231}$</td>
<td>15</td>
<td>146</td>
<td>300</td>
<td>18</td>
</tr>
<tr>
<td>JDQR/none</td>
<td>15</td>
<td>10</td>
<td>134</td>
<td>17</td>
</tr>
<tr>
<td>JDQR/diagonal</td>
<td>15</td>
<td>3</td>
<td>120</td>
<td>3</td>
</tr>
<tr>
<td>JDQR/ADI$_{13231}$</td>
<td>15</td>
<td>38</td>
<td>129</td>
<td>2</td>
</tr>
<tr>
<td>IRLPR/-</td>
<td>15</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IRLPR/-/—</td>
<td>15</td>
<td>22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIVIT/none</td>
<td>30</td>
<td>51</td>
<td>600</td>
<td>73</td>
</tr>
<tr>
<td>SIVIT/diagonal</td>
<td>30</td>
<td>19</td>
<td>600</td>
<td>26</td>
</tr>
<tr>
<td>SIVIT/ADI$_{13231}$</td>
<td>30</td>
<td>291</td>
<td>600</td>
<td>18</td>
</tr>
<tr>
<td>JDQR/none</td>
<td>30</td>
<td>28</td>
<td>261</td>
<td>14</td>
</tr>
<tr>
<td>JDQR/diagonal</td>
<td>30</td>
<td>11</td>
<td>229</td>
<td>2</td>
</tr>
<tr>
<td>JDQR/ADI$_{13231}$</td>
<td>30</td>
<td>71</td>
<td>241</td>
<td>2</td>
</tr>
<tr>
<td>IRLPR/-</td>
<td>30</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IRLPR/-/—</td>
<td>30</td>
<td>16</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.2

Performance of the various eigensolvers for computing $n_{ev}$ eigenvalues of the 3D-problem of size $n^3$ with ‘good’ initial vectors.

The problem at hand is best solved with our implementation of the implicitly restarted Lanczos algorithm provided that there is enough memory space available to increase the dimension of the trial space to three or even four times the number of eigenvalues desired. Jacobi-Davidson is to be preferred to the implicitly restarted Lanczos algorithm if the workspace is limited and only few vectors can be stored besides the eigenvectors that are to be computed.

REFERENCES


