SLAC-PUB-5961
October 1992
T/E

A Novel Algorithm for Calculation of the Extreme Eigenvalues of Large Hermitian Matrices*

YUKO OKAMOTO[1]

Stanford Linear Accelerator Center
Stanford University, Stanford, CA 94309, USA

and

HUMPHREY J. MARIS

Department of Physics, Brown University
Providence, RI 02912, USA

Submitted to *Computer Physics Communications*

* Work supported, in part, by the Department of Energy through Contract DE-AC03-76SF00515 and the National Science Foundation through Grant DMR-9120982.
1 On leave of absence from Department of Physics, Nara Women's University, Nara 630, Japan.

ABSTRACT

A new fast algorithm for calculating a few maximum (or minimum) eigenvalues and the corresponding eigenvectors of large $N \times N$ Hermitian matrices is presented. The method is based on a molecular dynamics algorithm for N coupled harmonic oscillators. The time step for iteration is chosen so that only the normal mode with the maximum eigenvalue grows exponentially. Other eigenvalues and eigenvectors are obtained one by one from the largest eigenvalue by repeating the process in subspaces orthogonal to the previous modes. The characteristics of the algorithm lie in the simplicity, speed (CPU time $\propto N^2$), and memory efficiency ($\mathcal{O}(N)$ besides the matrix). The effectiveness of the algorithm is illustrated by calculation of the groundstate and first-excited state energies of the Heisenberg model for an antiferromagnetic chain with N up to 16384.

1. Introduction

It is quite common in many fields of science and engineering that the calculation of a few of the lowest or highest eigenvalues and the corresponding eigenvectors of large Hermitian matrices is required. As the size of the matrix becomes large, however, calculations by conventional methods become prohibitively difficult, since the required CPU time as well as the memory space grows very rapidly. One of the most popular and powerful methods for the diagonalization of large matrices is the Lanczos method.[1] In this article we present a new algorithm which is very simple, and yet has similar computation time and memory requirements as the Lanczos method. The algorithm is related to the method of ref.[2,3] in which the vibrational normal modes (frequencies and eigenvectors) for a system of masses coupled together by linear springs are obtained by means of an unusual molecular dynamics simulation.[4] In its simplest form the algorithm that we derive gives the largest eigenvalue and the associated eigenvector. Lower eigenvalues can be obtained one by one by repeating the process while restricting the initial conditions to subspaces orthogonal to the previously obtained modes. By a trivial modification, one can also use the algorithm to study the smallest eigenvalue and other eigenvalues adjacent to it.

The paper is organized as follows. In section 2 we describe the algorithm. In section 3 we demonstrate the effectiveness of the method by calculating the eigenvalues of the lowest and first-excited states of Heisenberg antiferromagnetic chain. In section 4 conclusions are given. The FORTRAN program for the algorithm is presented in the appendix.

2. Method

2.1 Equation of motion for a system of harmonic oscillators

Let us suppose that we want to study the eigenvalues of the $N \times N$ Hermitian matrix ϕ. The problem is equivalent to finding the normal modes of vibration of a lattice of N particles. To make this association we can consider that the (l, l')

element of this matrix, $\phi_{ll'}$, is the spring constant connecting the particles l and l'. The equation of motion of this system is then given by

$$M\ddot{u}_l(t) = -\sum_{l'=1}^{N} \phi_{ll'}u_{l'}(t) \ , \quad (l = 1, \cdots, N) \ , \tag{2.1}$$

where M and $u_l(t)$ are the mass and the displacement of the l-th particle, respectively. Hereafter, we consider a real symmetric matrix ϕ only. The generalization to the case of a general Hermitian matrix is straightforward. We also set $M = 1$, for simplicity. By discretizing time with a step τ, the equation of motion (2.1) becomes a molecular dynamics equation:[2]

$$v_l(n+1) = v_l(n) - \tau \sum_{l'=1}^{N} \phi_{ll'}u_{l'}(n) \ , \tag{2.2.a}$$

$$u_l(n+1) = u_l(n) + \tau v_l(n+1) \ , \tag{2.2.b}$$

where $v_l(n)$ is the velocity of the l-th particle at time $t = n\tau$. Each displacement $u_l(n)$ and velocity $v_l(n)$ can be decomposed into a sum of normal modes as

$$u_l(n) = \sum_{\lambda} Q_\lambda(n) e_l(\lambda) \ , \tag{2.3.a}$$

$$v_l(n) = \sum_{\lambda} P_\lambda(n) e_l(\lambda) \ , \tag{2.3.b}$$

where $Q_\lambda(n)$ and $P_\lambda(n)$ are the amplitudes with which mode λ contributes to $u_l(n)$ and $v_l(n)$, respectively, and $e_l(\lambda)$ is the normalized eigenvector of the mode λ satisfying

$$\sum_{l'=1}^{N} \phi_{ll'} e_{l'}(\lambda) = \mu_\lambda e_l(\lambda) \equiv \omega_\lambda^2 e_l(\lambda) \ , \tag{2.4.a}$$

$$\sum_{l=1}^{N} e_l(\lambda) e_l(\lambda') = \delta_{\lambda\lambda'} \ . \tag{2.4.b}$$

Here, μ_λ is the eigenvalue of the matrix ϕ and $\omega_\lambda \equiv \sqrt{\mu_\lambda}$ is the corresponding eigenfrequency of oscillation for mode λ (ω_λ is imaginary for $\mu_\lambda < 0$).

By substituting (2.3) into (2.2) and using (2.4), we have

$$P_\lambda(n+1) = P_\lambda(n) - \mu_\lambda \tau Q_\lambda(n) \ , \tag{2.5.a}$$

$$Q_\lambda(n+1) = Q_\lambda(n) + \tau P_\lambda(n+1) \ . \tag{2.5.b}$$

These coupled equations can be rewritten as

$$Q_\lambda(n+1) - (2 - \mu_\lambda \tau^2) Q_\lambda(n) + Q_\lambda(n-1) = 0 \ . \tag{2.6}$$

Assuming a solution of the form

$$Q_\lambda(n) = (\beta_\lambda)^n \ , \tag{2.7}$$

where β_λ is a constant, we have two solutions

$$\beta_\lambda^\pm = \frac{2 - \mu_\lambda \tau^2 \pm \sqrt{\mu_\lambda \tau^2 (\mu_\lambda \tau^2 - 4)}}{2} \ . \tag{2.8}$$

The general solution for the amplitude is then given by

$$Q_\lambda(n) = c_\lambda^+(\beta_\lambda^+)^n + c_\lambda^-(\beta_\lambda^-)^n \ , \tag{2.9.a}$$

$$P_\lambda(n) = \frac{1}{\tau}(Q_\lambda(n) - Q_\lambda(n-1)) \ , \tag{2.9.b}$$

where c_λ^+ and c_λ^- are to be determined by the initial conditions. This solution describes the time development of the system under the molecular dynamics equation (2.2). In an ordinary molecular dynamics algorithm, it is necessary to make the time step τ as small as possible in order to minimize the error caused by discretizing time. Here, we take the very unusual approach of using large values of τ, i.e., values that are comparable to a period of oscillation of the system of coupled particles. Of course, such values would not give an accurate time-development of the motion of the mechanical system that we are considering. However, as we will now show, such a simulation can provide a very effective method for the determination of the extremal eigenvalues. To see how this comes about we note that depending on the value of τ (and μ_λ), we can classify the behavior of the solution into two cases:

$$\text{(A)} \qquad |\beta_\lambda| = 1 \quad \text{for} \quad 0 \le \mu_\lambda \tau^2 \le 4 \ , \tag{2.10.a}$$

$$\text{(B)} \qquad |\beta_\lambda| \neq 1 \quad \text{for} \quad \mu_\lambda \tau^2 < 0 \quad \text{or} \quad \mu_\lambda \tau^2 > 4 \ . \tag{2.10.b}$$

Furthermore, examining (2.8) for case (B) above, we find that one solution, which we refer to as $\beta_\lambda^>$, satisfies

$$|\beta_\lambda^>| > 1 \ , \tag{2.11.a}$$

while the other solution, $\beta_\lambda^<$, satisfies

$$|\beta_\lambda^<| < 1 \ . \tag{2.11.b}$$

This is the key observation which leads to the present algorithm: for $\mu_\lambda \tau^2 < 0$ or $\mu_\lambda \tau^2 > 4$, the amplitude of the mode λ grows exponentially as a function of time n, while for $0 \leq \mu_\lambda \tau^2 \leq 4$, the amplitude oscillates and stays of the same order as time progresses. Moreover, $\beta_\lambda^>$ in (2.11.a) can be written as

$$|\beta_\lambda^>| = \frac{|x| + \sqrt{|x+2|}\sqrt{|x-2|}}{2} \ , \tag{2.12}$$

for $x < -2$ or $x > 2$, where

$$x = x(\mu_\lambda) \equiv \mu_\lambda \tau^2 - 2 \ . \tag{2.13}$$

Hence, $|\ \beta_\lambda^>\ |$ is an even function of x. This equation determines which mode grows fastest. Namely, when there is more than one mode which corresponds to case (B), we have

(*i*) the mode for the maximum eigenvalue $\mu_{\lambda_{max}}$ grows fastest, if all the modes of case (B) satisfy $\mu_\lambda \tau^2 > 4$ ($x > 2$),

(*ii*) the mode for the minimum eigenvalue $\mu_{\lambda_{min}}$ grows fastest, if all the modes of case (B) satisfy $\mu_\lambda \tau^2 < 0$ ($x < -2$),

(*iii*) the mode for the maximum (or minimum) eigenvalue μ_λ grows fastest when $|\ x(\mu_{\lambda_{max}})\ |>|\ x(\mu_{\lambda_{min}})\ |$ (or $|\ x(\mu_{\lambda_{max}})\ |<|\ x(\mu_{\lambda_{min}})\ |$), if some modes of case (B) satisfy $\mu_\lambda \tau^2 > 4$ and some other modes satisfy $\mu_\lambda \tau^2 < 0$.

We can likewise determine which mode grows next fastest.

In order to avoid the above complication, we assume hereafter that all the eigenvalues μ_λ are non-negative (only situation (*i*) above is possible). This can always be achieved by the following transformation:

$$\phi \to \phi + bI \; , \tag{2.14}$$

where b is a constant satisfying $b \geq | \mu_{\lambda_{min}} |$ and I is the identity matrix.

2.2 Algorithm

Our algorithm is now summarized as follows. We *choose* the time step τ so that only the maximum eigenvalue $\mu_{\lambda_{max}}$ satisfies

$$\mu_\lambda \tau^2 > 4 \; , \tag{2.15.a}$$

and the rest of the eigenvalues satisfy

$$0 \leq \mu_\lambda \tau^2 \leq 4 \; . \tag{2.15.b}$$

We then obtain $\mu_{\lambda_{max}}$ and the corresponding eigenvector by just iterating (2.2) with time n from random initial conditions $u_l(0)$ and $v_l(0)$. As n increases $u_l(n)$ converges to a multiple of the normalized eigenvector $e_l(\lambda_{max})$ (see (2.4)) associated with the maximum eigenvalue. The maximum eigenvalue itself can be obtained from the final $u_l(n)$ by the Rayleigh quotient:

$$\mu_{\lambda_{max}} = \frac{\sum_{l,l'=1}^{N} u_l(n)\phi_{ll'}u_{l'}(n)}{\sum_{l=1}^{N} u_l(n)^2} \; . \tag{2.16}$$

Since the mode for $\mu_{\lambda_{max}}$ grows exponentially, it may be necessary to rescale $u_l(n)$ and $v_l(n)$ from time to time during time evolution.

When we need the next highest eigenvalue, we can repeat the above process with the initial conditions chosen to be orthogonal to the normalized eigenvector

$e_l(\lambda_{max})$ for the maximum eigenvalue $\mu_{\lambda_{max}}$. Thus, we first pick a set of random displacements $\tilde{u}_l(0)$ and $\tilde{v}_l(0)$, and then choose as initial displacements $u_l(0)$ and $v_l(0)$ according to the relations:

$$u_l(0) = \tilde{u}_l(0) - \Big(\sum_{l'=1}^{N} \tilde{u}_{l'}(0) e_{l'}(\lambda_{max})\Big) e_l(\lambda_{max}) \ , \tag{2.17.a}$$

$$v_l(0) = \tilde{v}_l(0) - \Big(\sum_{l'=1}^{N} \tilde{v}_{l'}(0) e_{l'}(\lambda_{max})\Big) e_l(\lambda_{max}) \ . \tag{2.17.b}$$

Note that due to round-off errors, we may have to re-orthogonalize $u_l(n)$ and $v_l(n)$ with respect to $e_l(\lambda_{max})$ again after some number of iterations.

In principle, this procedure can be repeated to yield more eigenvalues. However, when we want the k-th highest eigenvalue, we have to re-orthogonalize $u_l(n)$ with respect to the $k-1$ higher eigenvectors, which causes an increase in required computation time and memory space.

We could eliminate the need to re-orthogonalize by transforming ϕ as follows:

$$\phi_{ll'} \rightarrow \phi_{ll'} - \mu_{\lambda_{max}} e_l(\lambda_{max}) e_{l'}(\lambda_{max}) \ . \tag{2.18}$$

This is very effective for a medium size matrix ϕ. However, it is not a useful approach for a large matrix, since this transformation destroys the sparseness of the matrix and again slows down the calculation. Hence, depending on the size of the matrix, we have to choose one of the above two possibilities, (2.17) and (2.18).

In order to calculate a few minimum eigenvalues, we can apply the above algorithm with the matrix ϕ replaced by, for example,

$$\phi \rightarrow -\phi + bI \; , \tag{2.19.a}$$

where the constant b satisfies

$$b > \mu_{max} \; , \tag{2.19.b}$$

and I is the identity matrix. We obtain the lowest eigenvalue first, then the next lowest eigenvalue, and so forth. Note that we do not need the exact value of $\mu_{\lambda_{max}}$ but only a rough estimate for the transformation (2.19).

2.3 Choice for τ

It is desirable that we find the appropriate time step τ which satisfies the condition (2.15) with minimum effort. For this we consider the following quantity that we may think of in some regards as the total potential energy E_P of the system:

$$E_P(n) = \frac{1}{2}\sum_{l,l'=1}^{N} u_l(n)\phi_{ll'}\left[\frac{u_{l'}(n+1) + 2u_{l'}(n) + u_{l'}(n-1)}{4}\right] \; , \quad (n \geq 1) \; . \tag{2.20}$$

Using eqns. (2.2), (2.3), and (2.4), we can rewrite this as

$$E_P(n) = \frac{1}{8}\sum_{\lambda} \mu_\lambda(4 - \mu_\lambda \tau^2) Q_\lambda(n)^2 \; . \tag{2.21}$$

Hence, if (2.15) is satisfied, then $E_P(n)$ will become negative for some value n no matter what the initial value $E_P(1)$ may be. This is because the amplitude $Q_\lambda(n)$ corresponding to the eigenvalue that satisfies (2.15.a) grows exponentially as a function of n, while the rest of the amplitudes do not. This provides us with a strategy for finding τ. The method is simply to calculate $E_P(n_0)$ after a certain number of iterations n_0 and see if it has become negative. If it has not, the value of τ is then increased and the procedure repeated, until a negative value of $E_P(n_0)$ is obtained. For the Heisenberg model discussed in the next section, we found that $n_0 = 10$ is a good choice.

The discussion of the algorithm in the previous section was given under the assumption that $\mu_\lambda \tau^2$ was greater than 4 for only the largest eigenvalue. It is important to note that even if τ has been chosen so that there are *several* eigenvalues for which $\mu_\lambda \tau^2 > 4$, the algorithm still works because the mode with the highest eigenvalue grows fastest. This is an important consideration in systems that have a quasi-continuous distribution of eigenvalues, because for these systems it is time-consuming to find a value of τ such that only one eigenvalue has $\mu_\lambda \tau^2 > 4$. It turns out that the algorithm works surprisingly well even if τ has been chosen so that several eigenvalues satisfy the condition. The reason for this can be seen by examination of eq. (2.12). As a function of x, the quantity $|\beta_\lambda^>|$ has a square root singularity at $x = 2$. Consequently, for those modes with eigenvalues such that x is in the range just above 2, $|\beta_\lambda^>|$ increases very rapidly with increasing x. Thus, the rate of growth of the mode with largest eigenvalue is significantly larger than that of the next lower eigenvalue mode, even though the difference in the eigenvalues may be quite small.

We can also define a "kinetic energy" $E_K(n)$ by

$$E_K(n) = \frac{1}{2}\sum_{l=1}^{N}\left[\frac{v_l(n+1)+v_l(n)}{2}\right]^2 . \tag{2.22}$$

Then using eqns.(39) and (40) of ref. 2, one can show that the total energy $E_{tot} \equiv E_K + E_P$ is given by

$$E_{tot} = \frac{1}{2}\sum_{l=1}^{N} v_l(0)^2 + \frac{1}{2}\sum_{l,l'=1}^{N} (u_l(0) - v_l(0)\tau)\phi_{ll'}u_{l'}(0) \ . \qquad (2.23)$$

This quantity has the remarkable property that it is conserved regardless of the value of τ. In fig. 1 we show a typical time evolution of $E_P(n)$ and $E_K(n)$ for τ such that all the eigenvalues satisfy (2.15.b) (fig. 1a), and τ such that one eigenvalue satisfies (2.15.a) and the rest satisfy (2.15.b) (fig. 1b). The matrix is taken from the Heisenberg antiferromagnetic chain (with $N = 1024$) which is analyzed in the next section. Note that the total energy E_{tot} is conserved in both cases.

To choose τ for the second largest eigenvalue and so on, we apply the orthogonalization procedure (2.17), and again sweep τ upwards until $E_P(n_0)$ becomes negative.

3. Results

We have tested the algorithm for the one-dimensional Heisenberg antiferromagnetic chain with spin $s = \frac{1}{2}$. The Hamiltonian for a system of N_S spins is given by

$$H = -J\sum_{i=1}^{N_S} \vec{S}(i)\cdot\vec{S}(i+1) \ , \qquad (3.1)$$

where $\vec{S}(i)$ is the usual spin operator at the lattice site i and we set $J = -1$. Here, we assume periodic boundary conditions. The dimension of the Hamiltonian matrix is then

$$N = 2^{N_S} \ . \qquad (3.2)$$

By using the transformation (2.19), we have calculated the groundstate and first-excited state energies of the Heisenberg model for N_S up to 14 (i.e., $N = 16384$).

Double-precision arithmetic has been employed. In table 1 we present the results obtained after 100 iterations. The CPU time required for calculating the groundstate energy (with 100 iterations) was roughly two minutes on IBM9021 for $N = 16384$. These results all agree with those calculated by the Lanczos method [5] up to seven digits. We do not know the sources of discrepancies compared to ref. [5] after the sixth decimal place, but we remark that we have checked our results against the results obtained by the conventional method for N up to 256 and obtained complete agreement. At least in this example, the number of iterations required for our algorithm does not vary rapidly with the matrix size. The results in table 1 are all for 100 iterations. In table 2 we show a few examples of how the eigenvalues converge as a function of iterations for $N = 64$, 1024, and 16384. Note that we need up to about 60 iterations for convergence to the tenth decimal place.

4. Conclusions

In this paper we have presented a fast algorithm for calculation of a few maximum (or minimum) eigenvalues and the corresponding eigenvectors of an $N \times N$ Hermitian matrix. The simplicity of the algorithm is the outstanding virtue, as illustrated in sec. 2 and the appendix. The algorithm is also very efficient in that it requires $\mathcal{O}(N^2)$ computation time and $\mathcal{O}(N)$ memory space. These are the same order of magnitudes as required for the Lanczos method. One should note, however, that when more than one eigenvalue is needed the necessity for re-orthogonalization causes an increase in required computation time as well as memory space, which in turn limits the largest matrix size we can deal with. This is the same problem encountered by the original form of the Lanczos method. The subsequent modifications of the Lanczos method alleviate this re-orthogonalization problem, but create other complications such as occurrence of spurious eigenvalues.[1] Our algorithm is simple and does not suffer from these complications. Hence, if one needs only a few extreme eigenvalues, our method is a very attractive alternative.

Acknowledgements:

One of us (YO) is grateful to the members of Stanford Linear Accelerator Center for hospitality. This work was supported in part by the Department of Energy through Contract DE-AC03-76SF00515 and the National Science Foundation through Grant DMR-9120982.

Appendix

We present the FORTRAN program of the algorithm in the following. It is assumed that the matrix ϕ is already transformed so that all the eigenvalues μ_λ are non-negative. We remark that if one already has good estimates for the appropriate τ as described in detail in section 2, then one needs only *DO loop 210* in the main program.

```
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      PARAMETER (N=N) ! N is the matrix size.
      PARAMETER (NEIG=NEIG) ! NEIG is the no. of eigenvalues wanted.
      COMMON/CORD/U(N),V(N),TAU
      COMMON/MAT/PHI(N,N)
      COMMON/INTL/UINIT(N),VINIT(N)
      COMMON/REN/UEIG(N,NEIG),IEIG
      DIMENSION UNM(N),UN(N),UNP(N),U0(N),V0(N)
C*** PARAMETERS
      NITER=100
      NSTEP=10
      NSTORE=NITER/10
      NRESC=5
      NTAU=100
      TI=0.0D0
      BOUND=BOUND ! BOUND is the lower bound for eigenvalue search.
      TF=2.0D0/DSQRT(BOUND)
      DTAU=(TF - TI)/DFLOAT(NTAU)
      DO 10 IEIG=1,NEIG
      IF(IEIG.EQ.1) THEN
      CALL INIT
      DO 20 IL=1,N
      U0(IL)=UINIT(IL)
      V0(IL)=VINIT(IL)
   20 CONTINUE
      ELSE
      DO 30 IL=1,N
      SUM1=0.0D0
      SUM2=0.0D0
      DO 40 JL=1,N
      SUM1=SUM1 + UINIT(JL)*U(JL)
      SUM2=SUM2 + VINIT(JL)*U(JL)
   40 CONTINUE
      U0(IL)=U0(IL) - SUM1*U(IL)
      V0(IL)=V0(IL) - SUM2*U(IL)
   30 CONTINUE
      ENDIF
C*** SWEEP TAU TO LOOK FOR APPROPRIATE TAU
      DO 50 IT=1,NTAU
      IF(IEIG.EQ.1) THEN
      TAU=TF - DTAU*DFLOAT(IT-1)
      ELSE
      TAU=TI + DTAU*DFLOAT(IT-1)
      ENDIF
      DO 60 IL=1,N
      U(IL)=U0(IL)
      V(IL)=V0(IL)
   60 CONTINUE
      DO 70 ISTEP=1,NSTEP
      IF(MOD(ISTEP,NRESC).EQ.0) CALL RENORM
      DO 80 IL=1,N
      UNM(IL)=U(IL)
   80 CONTINUE
      CALL TSTEP
      DO 90 IL=1,N
      UN(IL)=U(IL)
   90 CONTINUE
      CALL TSTEP
      DO 100 IL=1,N
      UNP(IL)=U(IL)
  100 CONTINUE
C*** CALCULATE POTENTIAL ENERGY
      EP=0.0D0
      DO 110 IL=1,N
      DO 110 JL=1,N
      EP=EP + 0.125D0*UN(IL)*PHI(IL,JL)*(UNP(JL)+ 2.0D0*UN(JL)+UNM(JL))
  110 CONTINUE
      IF(IEIG.EQ.1) THEN
      IF(EP.LT.0.0D0) GOTO 50
      IF(EP.GE.0.0D0.AND.ISTEP.EQ.NSTEP) THEN
      TAU=TAU + DTAU
      GOTO 200
      ENDIF
      ELSE
      IF(EP.LT.0.0D0) GOTO 200
      ENDIF
   70 CONTINUE
   50 CONTINUE
C*** MAKE A LONG RUN TO OBTAIN EXACT EIGENVALUE
  200 DO 210 ITER=1,NITER
      CALL TSTEP
      IF(MOD(ITER,NRESC).EQ.0) CALL RENORM
      IF(MOD(ITER,NSTORE).EQ.0) THEN
      CALL EIGEN(EIGVAL)
      WRITE(6,1000) ITER,EIGVAL
 1000 FORMAT(' IT = ',I3,' EIGVAL = ',F16.10/)
      ENDIF
  210 CONTINUE
C*** STORE THE EIGENVECTOR
      DO 220 IL=1,N
      UEIG(IL,IEIG)=U(IL)
  220 CONTINUE
      TI=TAU
      DTAU=(TF - TI)/DFLOAT(NTAU)
   10 CONTINUE
      STOP
      END
C***
      SUBROUTINE  INIT
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      PARAMETER (N=N)
      COMMON/MAT/PHI(N,N)
      COMMON/INTL/UINIT(N),VINIT(N)
```

Prepare matrix elements, PHI(l,l'), *and random initial conditions*, UINIT(l) *and* VINIT(l).

```
      RETURN
      END
C***
```

```
      SUBROUTINE  TSTEP
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      PARAMETER (N=N)
      COMMON/CORD/U(N),V(N),TAU
      COMMON/MAT/PHI(N,N)
      DO 10 IL=1,N
      FORCE=0.0D0
      DO 20 JL=1,N
      FORCE=FORCE - PHI(IL,JL)*U(JL)
   20 CONTINUE
      V(IL)=V(IL) + TAU*FORCE
   10 CONTINUE
      DO 30 IL=1,N
      U(IL)=U(IL) + TAU*V(IL)
   30 CONTINUE
      RETURN
      END
C***
      SUBROUTINE  RENORM
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      PARAMETER (N=N)
      PARAMETER (NEIG=NEIG)
      COMMON/CORD/U(N),V(N),TAU
      COMMON/REN/UEIG(N,NEIG),IEIG
      SUM=0.0D0
      DO 10 IL=1,N
      SUM=SUM + U(IL)*U(IL)
   10 CONTINUE
      USCALE=DSQRT(SUM)
      DO 20 IL=1,N
      U(IL)=U(IL)/USCALE
      V(IL)=V(IL)/USCALE
   20 CONTINUE
C*** GRAM-SCHMIDT
      IF(IEIG.NE.1) THEN
      DO 30 NEG=1,IEIG-1
      SUMU=0.0D0
      SUMV=0.0D0
      DO 40 IL=1,N
      SUMU=SUMU + UEIG(IL,NEG)*U(IL)
      SUMV=SUMV + UEIG(IL,NEG)*V(IL)
   40 CONTINUE
      DO 50 IL=1,N
      U(IL)=U(IL) - SUMU*UEIG(IL,NEG)
      V(IL)=V(IL) - SUMV*UEIG(IL,NEG)
   50 CONTINUE
   30 CONTINUE
      ENDIF
      RETURN
      END
C***
      SUBROUTINE  EIGEN(EIGVAL)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      PARAMETER (N=N)
      COMMON/CORD/U(N),V(N),TAU
      COMMON/MAT/PHI(N,N)
      G0=0.0D0
      G2=0.0D0
      DO 10 IL=1,N
      G0=G0 + U(IL)*U(IL)
      AL=0.0D0
      DO 20 JL=1,N
      AL=AL + PHI(IL,JL)*U(JL)
   20 CONTINUE
      G2=G2 + U(IL)*AL
   10 CONTINUE
      EIGVAL=G2/G0
      RETURN
      END
```

REFERENCES

1. For a review, see J.K. Cullum and R.A. Willoughby, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations* (Birkhäuser, Boston, MA, 1985).

2. M.L. Williams and H.J. Maris, *Phys. Rev. B* **35** (1985) 4508.

3. K. Yakubo, T. Nakayama and H.J. Maris, *J. Phys. Soc. Jpn.* **60** (1991) 3249.

4. A similar algorithm which is based on the same mechanical system but uses different molecular dynamics equations has been proposed to calculate a few minimum eigenvalues. See M. Arjunwadkar and D.G. Kanhere, *Comput. Phys. Commun.* **62** (1991) 8.

5. D. Medeiros and G.G. Cabrera, *Phys. Rev. B* **43** (1991) 3703.

TABLE CAPTIONS

1. Groundstate and first-excited state energies of Heisenberg antiferromagnetic chain with N_S spins. The results are rounded off at the eleventh decimal place.

2. Groundstate and first-excited state energies (μ_{λ_0} and μ_{λ_1}) of Heisenberg antiferromagnetic chain as a function of iterations for $N = 64, 1024$, and 16384.

Table 1.

N_S	N	Groundstate	First-excited state
4	16	−2.0	−1.0
6	64	−2.8027756377	−2.1180339887
8	256	−3.6510934089	−3.1284190638
10	1024	−4.5154463545	−4.0922073467
12	4096	−5.3873909174	−5.0315434037
14	16384	−6.2635495335	−5.9564438240

Table 2.

IT	$\mu_{\lambda_0}(N=64)$	$\mu_{\lambda_0}(N=1024)$	$\mu_{\lambda_0}(N=16384)$
20	−2.8027750983	−4.5153136281	−6.2632192228
40	−2.8027756377	−4.5154463545	−6.2635493911
60	−2.8027756377	−4.5154463545	−6.2635495335
80	−2.8027756377	−4.5154463545	−6.2635495335
100	−2.8027756377	−4.5154463545	−6.2635495335
IT	$\mu_{\lambda_1}(N=64)$	$\mu_{\lambda_1}(N=1024)$	$\mu_{\lambda_1}(N=16384)$
20	−2.1180325187	−4.0922058679	−5.9564352761
40	−2.1180339887	−4.0922073467	−5.9564438234
60	−2.1180339887	−4.0922073467	−5.9564438240
80	−2.1180339887	−4.0922073467	−5.9564438240
100	−2.1180339887	−4.0922073467	−5.9564438240

FIGURE CAPTIONS

1) Typical time evolution of energies. (a) All the modes satisfy oscillatory condition (A) of eq. (2.10.a). (b) One mode satisfies condition (B) and the rest satisfy condition (A).

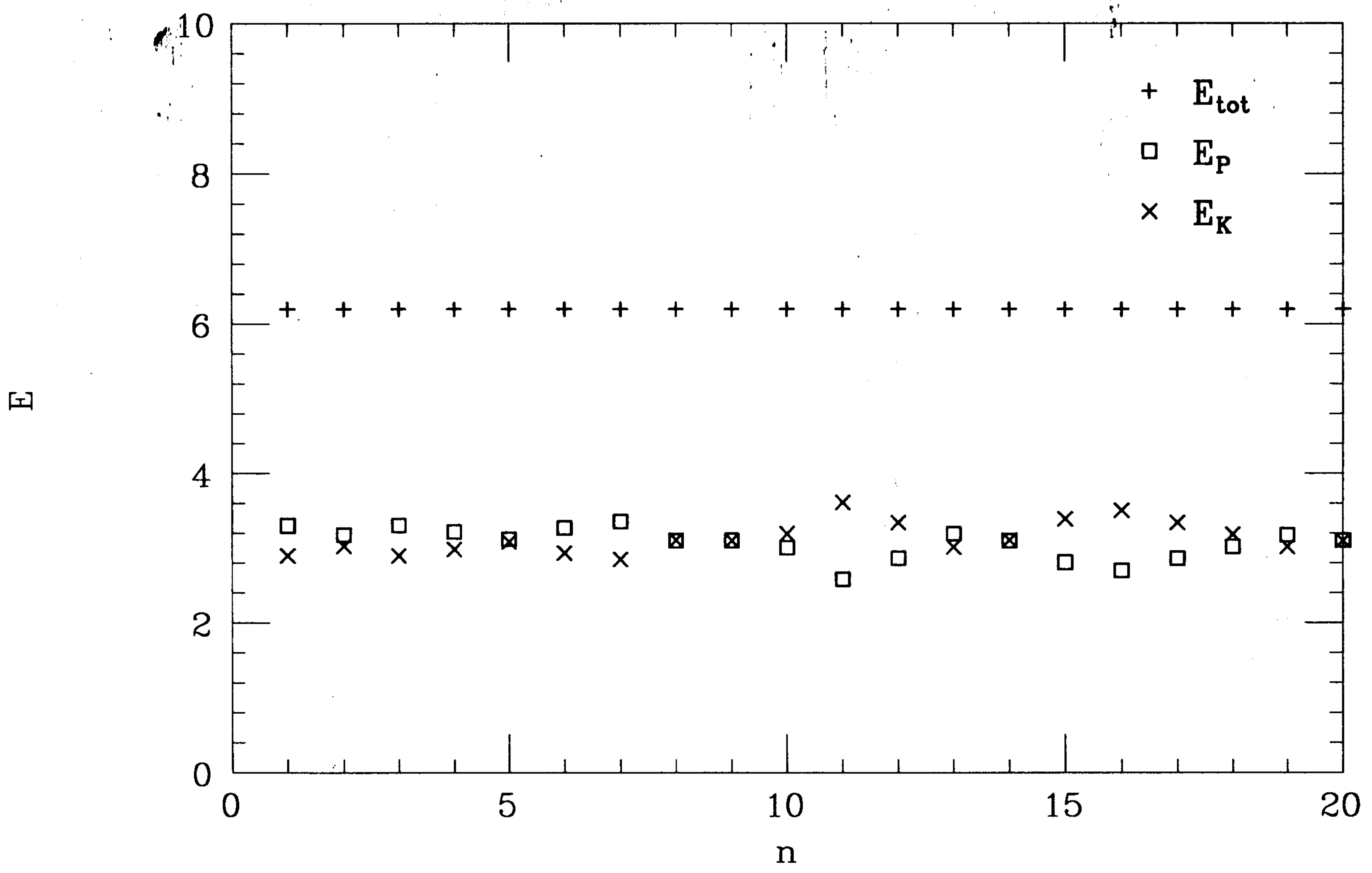

Fig. 1. (a)

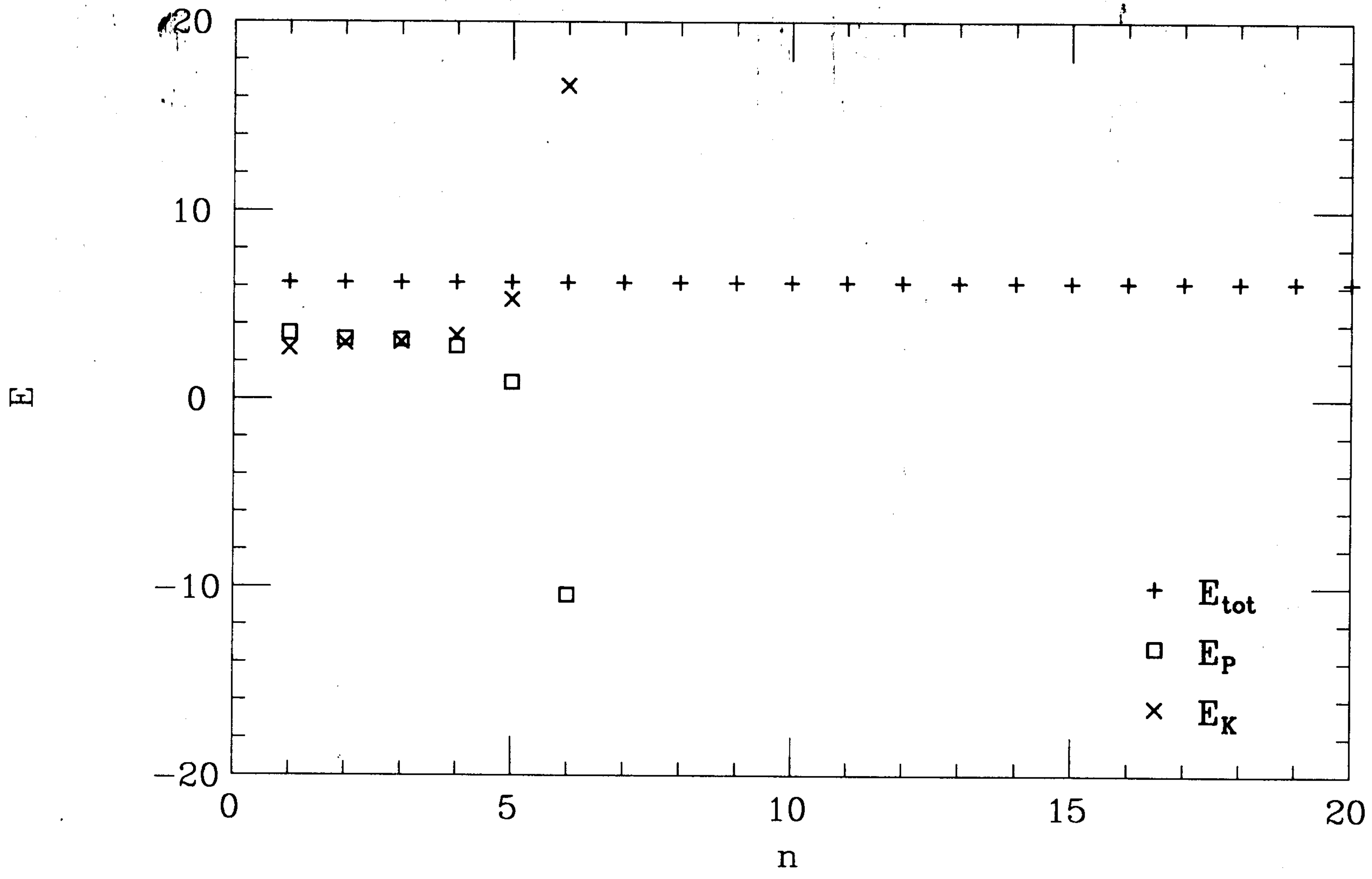

Fig. 1. (b)