PRODUCT FORMULA ALGORITHMS FOR SOLVING THE TIME DEPENDENT SCHRÖDINGER EQUATION

Hans De RAEDT

Physics Department, University of Antwerp, Universiteitsplein 1, B-2610 Wilrijk, Belgium
and
Natuurkundig Laboratorium, University of Amsterdam, Valckenierstraat 65, 1018 XE Amsterdam, The Netherlands

1987

NORTH-HOLLAND – AMSTERDAM

Contents

Computer Physics Reports 7 (1987) 1–72
North-Holland, Amsterdam

PRODUCT FORMULA ALGORITHMS FOR SOLVING THE TIME DEPENDENT SCHRÖDINGER EQUATION

Hans De RAEDT

Physics Department, University of Antwerp, Universiteitsplein 1, B-2610 Wilrijk, Belgium *

and

Natuurkundig Laboratorium, University of Amsterdam, Valckenierstraat 65, 1018 XE Amsterdam, The Netherlands

Received 29 June 1987

This paper introduces a new family of explicit and unconditionally stable algorithms for solving linear parabolic difference equations. The mathematical foundation is presented and it is shown how the algorithms can be implemented on scalar and vector processors. The performance is evaluated and compared to standard methods. It is demonstrated that some of the proposed algorithms are orders of magnitude more efficient than conventional schemes. The most efficient algorithm is employed to solve Schrödinger equations for problems including, localization in an almost-periodic potential and two-dimensional Anderson localization. By combining product formula algorithms and the variational principle a method is devised to compute the low-lying states of a quantum system, capable of separating nearly-degenerate eigenstates. The usefulness of this method is illustrated by applying it to a spin-boson system.

* Permanent address.

0167-7977/87/$25.20

1. Introduction

The time evolution of a state of a non-relativistic quantum-mechanical system is governed by the time-dependent Schrödinger equation (TDSE)

$$\frac{\partial \Psi(\boldsymbol{r}, t)}{\partial t} = -\mathrm{i} H \Psi(\boldsymbol{r}, t), \tag{1.1}$$

where H is the Hamiltonian of the model system, $\Psi(\boldsymbol{r}, t)$ is the normalized, complex-valued wave function, $\Psi(\boldsymbol{r}, 0)$ is the initial state at time $t = 0$, and units are such that $\hbar = 1$. The solution of the TDSE contains all dynamical information on the system. Methods to solve the TDSE emerge as important tools to simulate for instance molecular [1,2] and nuclear collisions [3–5], atom–surface interactions [6,7], high-resolution electron-microscopy image simulation [8,9], light propagation in optical fibers [10–14], electron motion in disordered materials [15–18], etc. Common to all these simulations is that many of the model properties can be unraveled by examining the propagation of wave packets.

For systems evolving in continuum space, the TDSE belongs to the class of linear parabolic partial differential equations. Although it is easy to see that the formal solution of (1.1) is

$$\Psi(\boldsymbol{r}, t) = \mathrm{e}^{-\mathrm{i}tH} \Psi(\boldsymbol{r}, 0), \tag{1.2}$$

in general the explicit expression for the solution of such equations cannot be written down in closed form and one has to resort to numerical techniques to solve the initial value problem. Usually this is accomplished by means of finite-difference methods [19].

Consider, for example, a particle moving on a line of length X. In appropriate units its Hamiltonian is given by

$$\mathscr{H} = -\frac{\mathrm{d}^2}{\mathrm{d}x^2} + V(x), \tag{1.3}$$

where $V(x)$ represents the potential at position x. The wave function of the particle $\psi(x, t) = 0$ unless $0 \leqslant x \leqslant X$. Obviously, numerical solution of (1.3) requires some discretization of continuum space. Let δ by the mesh length in the x-direction and approximate $\psi(x, t)$ by $\psi_l(t)$ for $l\delta < x \leqslant (l+1)\delta$. Since free-boundary conditions have been adopted, $\psi_l(t) = 0$ if $l \leqslant 0$ or $l > L + 1$. Replacing the second derivative with respect to x by its simplest finite-difference approximation ($\mathrm{d}^2\psi(x, t)/\mathrm{d}x^2 \approx \delta^{-2}[\psi_{l+1}(t) - 2\psi_l(t) + \psi_{l-1}(t)]$) yields the TDSE

$$\frac{\partial}{\partial t}\psi_l(t) = -\mathrm{i}\{-\delta^{-2}[\psi_{l+1}(t) + \psi_{l-1}(t)] + v_l \psi_l(t)\}, \tag{1.4}$$

with $v_l = V(l\delta) + 2\delta^{-2}$. In all practical situations the number of mesh points $L + 1$ will be finite. Hence (1.4) can be written in matrix form

$$\frac{\partial \psi(t)}{\partial t} = -\mathrm{i} H \psi(t), \tag{1.5}$$

where $\psi(t) = (\psi_1(t), \ldots, \psi_{L+1}(t))^{\mathsf{T}}$ is a column vector of length $L+1$ and H is the tri-diagonal matrix

$$H = \begin{pmatrix} v_1 & -\delta^{-2} & & & \\ -\delta^{-2} & v_2 & & & \\ & \ddots & \ddots & & \\ & & v_L & -\delta^{-2} & \\ & & -\delta^{-2} & v_{L+1} \end{pmatrix}. \tag{1.6}$$

The numerical solution of the difference equation (1.4) will be an approximation to the solution of the TDSE with Hamiltonian (1.3). The accuracy of the approximation is, in this case, determined by the mesh length δ. By construction the approximate solution will converge to the continuum result if $\delta \to 0$, i.e. the numerical method is "consistent" [19].

Difference equation (1.4) can also be interpreted as the TDSE for a particle moving on a chain of $L+1$ lattice sites. Most conveniently such a system is described to be a tight-binding Hamiltonian

$$H = V \sum_{l=1}^{L} (c_l^+ c_{l+1} + c_{l+1}^+ c_l) + W \sum_{l=1}^{L+1} \epsilon_l n_l, \tag{1.7}$$

where $c_l^+(c_l)$ creates (annihilates) a particle at the site l, $n_l = c_l^+ c_l$ counts the number of particles at site l (i.e. $n_l = 0, 1$ since here there is at most one particle), V sets the kinetic energy scale and $W\epsilon_l$ is the potential at site l felt by the particle. Although not essential, the operators c_l^+ and c_l are taken to obey fermion operator algebra, i.e. $\{c_l^+, c_{l'}^+\} = \{c_l, c_{l'}\} = 0$ and $\{c_l^+, c_{l'}\} = \delta_{l,l'}$. A state $|\Phi(t)\rangle$ of this single particle system can be written as $|\Phi(t)\rangle = \sum_{l=1}^{L+1} \Phi_l(t) c_l^+ |0\rangle$ in which $|0\rangle$ denotes the vacuum state. Substituting this representation in the TDSE

$$\frac{\partial \Phi(t)}{\partial t} = -\mathrm{i} H \Phi(t), \tag{1.8}$$

with H given by (1.7) yields

$$\frac{\partial}{\partial t} \Phi_l(t) = -\mathrm{i}\{V[\Phi_{l+1}(t) + \Phi_{l-1}(t)] + W\epsilon_l \Phi_l(t)\}; \quad l = 1, \ldots, L+1, \tag{1.9}$$

and because of the free boundary conditions $\Phi_l(t) = 0$ if $l \leqslant 0$ or $l > L+1$. Comparison of (1.4) and (1.9) learns that both formulations are equivalent provided we set $V = -\delta^{-2}$ and $W\epsilon_l = v_l$. The intimate relationship between discretized Schrödinger equation (1.4), matrix equation (1.5) and lattice model (1.8) shows that the latter, properly generalized to the d-dimensional case, may be regarded as a generic model, encompassing both genuine lattice models themselves as well as the (simplest) difference approximation to continuum problems. In chapter 2 it will become clear that the introduction of the fermion-operator notation is instrumental for the construction of product-formula methods. Therefore lattice model (1.7) and its d-dimensional generalizations will be the starting point in the development of these algorithms.

The fundamental problem hidden in for instance matrix equation (1.8) is that if H cannot be brought into diagonal form, either by analytical manipulations or by brute force numerical diagonalization, it is impossible to compute the action of the propagator $e^{-i\tau H}$ on an arbitrary wave function. Therefore the first step in the development of any numerical method for solving the TDSE is to propose an approximation for the time-evolution operator $e^{-i\tau H}$ for a well-chosen time step τ. Integration of the TDSE is then accomplished by repeated application of this approximate time-step operator.

The quantum mechanical nature of the problem requires additional attention. For example it is important to make sure that the method being used conserves the total probability, i.e. $\sum_{l=1}^{l+1} |\psi_l(t)|^2 = 1$, since otherwise the numerical method itself would be responsible for creating or annihilating "particles" in the course of the integration process. This problem can be solved formally by demanding that the approximate time-step operator is unitary. Then the norm of the wave function will be conserved. In addition, unitarity of the time-step operator implies unconditional stability of the numerical method. Indeed, if U denotes the time-step operator that satisfies $UU^\dagger = 1$ then $\|U\| = 1$ whereas a necessary and sufficient condition for stability is $\|U\| \leqslant 1$ [19]. In practice, unitarity of the time-step operator guarantees that errors due to the approximation scheme itself and also rounding errors, always present in floating-point computations, cannot grow indefinitely.

One of the most popular methods to solve the TDSE is the Cranck–Nicholson (CN) method [19–21]. In this scheme, the time-evolution operator $e^{-i\tau H}$ is approximated by a time-step operator of the form

$$U_{\mathrm{CN}}(\tau) \equiv (1 - i\tau H/2)(1 + i\tau H/2)^{-1}. \tag{1.10}$$

Since U_{CN} is unitary, this scheme is unconditionally stable [14,16].

The presence in (1.10) of the operator $(1 + i\tau H/2)^{-1}$ makes the CN method of the implicit type. A major drawback of the CN method is that each time step requires the inversion of the matrix $1 + i\tau H/2$, a highly undesirable feature since in general the number of arithmetic operations will scale with $(L+1)^{3d}$, where here and in the rest of this paper d stands for the dimensionality of the system. In spite of the fact that in many cases the matrix to be inverted can be reduced to tri-diagonal form, for systems of dimensionality larger than one, implicit schemes such as the CN scheme are often very costly to use. This seems to hamper the application of the CN method to problems in two or three space dimensions and depending on the problem at hand, also puts considerable demand on memory and CPU time for systems of lower dimensionality. Explicit, stable methods of the same order of accuracy should be preferred over the CN method. However, most of these explicit schemes turn out to be only conditionally stable [22–24]. Therefore it is worthwhile to search for explicit, unconditionally stable algorithms, not necessarily based on rational approximations [19] to the matrix $e^{-i\tau H}$.

In this paper a new family of algorithms for solving linear parabolic differential (difference) equations is introduced. The main idea is to use generalizations of the Lie–Trotter product-formula [25–30] (PF) in place of rational approximations, to construct systematic approximations to the exponential operators. To considerable extent this approach has been inspired by methods originally developed to perform Monte Carlo simulations for quantum lattice models [31–33]. In particular, symmetrized versions [34,35] of PF's (SPF's) have found to be of central importance

in the development of algorithms based on PF's. In addition they provide a convenient setting for proving the stability and convergence of such algorithms.

The underlying idea of this approach is the following. In all cases of practical interest, the Hamiltonian can be written as a sum of operators $H = \sum_{q=1}^{p} H_q$ in such a manner that each H_q is sufficiently simple so that it can be diagonalized easily, i.e. analytically. The time-step operator $e^{-i\tau H}$ is then approximated by some ordered product of exponents of $-i\tau H_q$. Such approximate time-step operators are unitary and therefore algorithms based on them are unconditionally stable. Since the eigenvalues and eigenvectors of H_q are known, calculation of $e^{-i\tau H_q}$ is straightforward, at least in principle. In practical applications the choice of the H_q's and the SPF is of crucial importance because it will determine the efficiency of the algorithm.

Completely different viewpoints and motivation have led to a similar evolution of high-resolution electron-microscopy (HREM) image simulation techniques. There the objective is to compute the propagation of electrons througha crystal foil. Although a first-principle description of the interaction of the relativistic electron with the material necessarily starts from the Dirac equation, it is generally accepted that the problem can be simplified such that it reduces to the solution of a two-dimensional Schrödinger equation [36]. Already in the early development of the numerical methods for solving this equation it has been recognized that numerical techniques based on the so-called slice methods [37,38] are the most efficient. For a recent review and comparison of various methods see ref. [9]. In this approach the specimen is thought to consist of a number of two-dimensional slices, and the electron propagator is replaced by a suitable "short-time" (or thin slice) approximation for the particular slice. Given the wave function at the entrance plane, integration of the TDSE is accomplished by repeatedly letting the approximate electron propagator act on the wave function. The multi-slice method is a special case of a second-order SPF algorithm.

Likewise similar progress has been made in numerical methods to compute the propagation of light in optical fibers [10–14]. In this instance the corresponding wave equation is such that the index of refraction does not depend on the distance along wave guide. This property has been used to establish a close relationship between the solution of the Helmholtz equation and that of the corresponding Fresnel form of the wave equation [10–14]. As the latter is of the parabolic type, it is easier to solve than the elliptic Helmholtz equation. It has been demonstrated that the so-called split-operator FFT method is very effective in solving such problems [10–14]. As a matter of fact this technique is identical to the multi-slice FFT method used in HREM image simulation [9], and consequently it is also an example of a particular SPF algorithm.

This paper is largely intended as an original contribution rather than a review. It is presented in this journal in the hope that it will reach a much more diverse audience than otherwise would have been possible and that it will stimulate further research in and application of SPF algorithms. As not much research on this subject has been carried out, a large part of this paper is devoted to the mathematical foundation (chapter 2) and implementation (chapter 3) of SPF-based algorithms. A short note on part of the material presented in this paper has been published elsewhere [39].

In the literature a vast number of different integration schemes has been proposed, each one having its own merits and drawbacks. Therefore it would be a very laborous task to compare them all with the SPF algorithms consequently it was decided to compare SPF algorithms with only one standard technique, the CN method. A quantitative investigation of the performance

and efficiency of SPF schemes as well as a detailed comparison with the CN approach is given in chapter 4. Chapter 5 discusses applications to three different systems, relevant to condensed matter physics. They serve to illustrate that SPF algorithms allow us to explore the time-dependent behavior of systems on length and time scales that are substantially larger than previously accessible.

Although the emphasis of this paper is on SPF methods for solving the TDSE, the same approach works equally well for "imaginary-time" problems. Such diffusion equations can also be solved by SPF algorithms. The details of this are given in chapter 6, where it is also shown how a SPF scheme and the variational principle can be combined to yield a novel method to compute simultaneously a number of low-lying states also in cases where the ground state is nearly degenerate. Finally, chapter 7 briefly considers some problems for further study and contains the conclusions.

2. Theory

This chapter deals with the theoretical concepts underlying the SPF algorithms. The theory will be presented in several steps. In section 2.1, a second-order real-space (RS) symmetrized product-formula (SPF) method for a one-dimensional (1D) system is introduced using conventional matrix notation, illustrating the principal ideas of the approach. Extending these concepts to d-dimensional systems and to other second-order or fourth-order schemes requires a more appropriate notation than matrix representations. In order to be able to express the theoretical ideas in a well-defined and compact form, it has been found advantageous to use fermion-operator language. The equivalence between this formalism and the matrix representation rests on the well-known fact that a matrix can always be written as a tight-binding-like Hamiltonian of a single particle system, as shown explicitly for the 1D case in the Introduction.

Readers who are not familiar with the use of fermion operators should not worry about having to digest complicated many-body theory. In this paper fermion operators are only used to represent matrices which otherwise could not have been written down in an orderly manner, and to allow straightforward calculation of commutators of these matrices.

In section 2.2, several second-order algorithms are presented. The scheme presented in section 1 is reconstructed using fermion-operator notation. It is hoped that this will help the reader in getting more acquainted with the notation. It is then shown how to extend the SPF approach to d-dimensional systems. Finally in section 3 several fourth-order methods are introduced. It is here that the use of fermion-operator algebra proves to be essential.

In these three sections emphasis is on the construction of algorithms that do not require Fourier transforms but directly operate on the wave function in its RS representation. The goal is to devise explicit algorithms of which the number of operations per time step is proportional to the number of lattice (or mesh) points. An additional requirement is that they should easily vectorize and be highly parallel.

As already indicated, in each section the theory will first be developed for the 1D case because this simplifies the notation considerably and brings out the essential points more clearly. The modularity of the SPF approach will allow straightforward extension to the d-dimensional case.

General expressions for operators entering the formalism can be found in appendix B. In order not to get completely lost in a zoo of indices the theory will be developed for the case of a hyper-cubic lattice and nearest-neighbor coupling only.

Apart from unitarity (see chapter 1), another important property of the approximate time-step operator is its "local" error, i.e. the maximum error made in taking a single time step. As usual this error is measured by computing $\mathrm{RMS}(U) \equiv \| \mathrm{e}^{-\mathrm{i}\tau H} - U \|$. Note that according to this definition (see appendix A), $\mathrm{RMS}(U)$ is precisely the maximum of the Root-Mean-Square (RMS) error of the approximate wave function $\psi_U(\tau) \equiv U\psi(0)$ to the corresponding true value $\psi_{\mathrm{exact}}(\tau) \equiv \mathrm{e}^{-\mathrm{i}\tau H}\psi(0)$, taken over all (normalized) values of the initial state $\psi(0)$. The maximum "global"error made by taking m of such steps is then simply $m\mathrm{RMS}(U)$, as shown in appendix A.

We illustrate the use of these concepts by taking the Cranck–Nicholson (CN) method as an example. In the CN approach the propagator $\mathrm{e}^{-\mathrm{i}\tau H}$ is replaced by the rational approximation $(1 - \mathrm{i}\tau H/2)(1 + \mathrm{i}\tau H/2)^{-1}$ [14]. From (A.8) it follows that the local error is bounded by $\mathrm{RMS(CN)} \leqslant \tau^3 \| H \|^3/12$, reflecting the fact that the local error of the CN method is $\mathcal{O}(\tau^3)$, for τ sufficiently small. The global error for m steps is then bounded by $t\tau^2 \| H \|^3/12$ where $t = m\tau$.

In the following sections several possibilities to decompose H will be examined. In general, the simplest product-formula (PF) approximation consists of replacing $\mathrm{e}^{-\mathrm{i}\tau H}$ by [27,28]

$$E_1\big(\tau, \{H_q\}\big) = \prod_{q=1}^{p} \mathrm{e}^{-\mathrm{i}\tau H_q}, \tag{2.1}$$

where $H = \sum_{q=1}^{p} H_q$. It can be shown (see appendix A) that [30]

$$\Big\| \mathrm{e}^{-\mathrm{i}\tau H} - E_1\big(\tau, \{H_q\}\big) \Big\| \leqslant \frac{\tau^2}{2} \sum_{1 \leqslant q < q'}^{p} \Big\| \big[H_q, H_{q'}\big] \Big\|, \tag{2.2}$$

showing that this approximation is correct to order τ. As it is easy to improve the order of correctness in τ without affecting the simplicity of the approximation, algorithms based on (2.1) are not of great interest, especially when one recalls that the CN method is already correct to order τ^2. The reason for introducing (2.1) here is that it will be used later in the construction of the fourth-order methods.

2.1. Second-order method: matrix formulation

In chapter 1 it was shown that the numerical solution of typical 1D TDSE's can be cast into the form

$$\frac{\partial \psi(t)}{\partial t} = -\mathrm{i}H\psi(t), \tag{2.3}$$

where $\psi(t) = (\psi_1(t), \ldots, \psi_{L+1}(t))^{\mathsf{T}}$ denotes a (column) vector in a $(L+1)$-dimensional space and

$$H = \begin{pmatrix} W\epsilon_1 & V & & & & & \\ V & W\epsilon_2 & V & & & & \\ & V & W\epsilon_3 & V & & & \\ & & & \ddots & & & \\ & & & V & W\epsilon_L & V & \\ & & & & V & W\epsilon_{L+1} \end{pmatrix}, \tag{2.4}$$

is a tri-diagonal $(L+1) \times (L+1)$ matrix representing the (discretized) Hamiltonian. The formal solution of (2.3) is given by

$$\psi(t) = \mathrm{e}^{-\mathrm{i}Ht}\psi(0), \tag{2.5}$$

where $\psi(0)$ stands for the wave function at time $t = 0$. The numerical problem to be solved is to compute the exponent of the matrix H without diagonalizing the matrix H explicitly.

Second-order schemes are obtained by symmetrizing first-order formula (2.1) as follows [34,35]

$$\mathrm{e}^{-\mathrm{i}\tau H} = \mathrm{e}^{-\mathrm{i}\tau H/2}\,\mathrm{e}^{-\mathrm{i}\tau H/2} \approx E_2\big(\tau, \{H_q\}\big) \equiv E_1\big(\tau/2, \{H_q\}\big)E_1^{\dagger}\big(\tau/2, \{H_q\}\big). \tag{2.6}$$

In most practical situations it suffices to consider the case where $H = H_1 + H_2$. Then

$$E_2(\tau, H_1, H_2) \equiv \mathrm{e}^{-\mathrm{i}\tau H_1/2}\,\mathrm{e}^{-\mathrm{i}\tau H_2}\,\mathrm{e}^{-\mathrm{i}\tau H_1/2}, \tag{2.7}$$

is a unitary approximation to $\mathrm{e}^{-\mathrm{i}\tau H}$ and just as in the first-order case, one can obtain an upperbound for the error of this approximation namely

$$\big\|\mathrm{e}^{-\mathrm{i}\tau H} - E_2(\tau, H_1, H_2)\big\| \leqslant \tau^3\big(\|[H_1, [H_1, H_2]]\| + 2\|[H_2, [H_1, H_2]]\|\big)/24, \tag{2.8}$$

as demonstrated in appendix A. As (2.8) indicates, the error per time step is at most of order τ^3, the same as for the CN scheme. There is however an essential difference. As shown by (2.8), the error due to the second-order SPF approximation vanishes with $\|[H_1, H_2]\|$. It is also worthwhile to note that in the CN method the energy is conserved within numerical accuracy of the arithmetic operations, whereas in the second-order scheme (2.7) this is not the case since in general $[H, E_2(\tau, H_1, H_2)] \neq 0$.

Although upperbounds such as (2.8) are general and powerful theoretical tools, in actual applications it is difficult to use them to estimate the minimum value of τ required to guarantee a specified accuracy. The reason is that in most cases they are much too crude and tend to over-estimate the actual errors considerably. In chapter 4 it will be demonstrated that in typical situations, the error scales with the time step and the number of time steps in accordance with the dependence on time step and number of time steps of the corresponding upperbound. Hence the knowledge of the upperbounds on the error is very useful for estimating the accuracy of a particular calculation on the basis of previous data.

One way to construct a second-order algorithm is to choose [40]

$$e^{-i\tau H} = e^{-i\tau(H_O+H_E)} \approx E_2(\tau, H_O, H_E) = e^{-i\tau H_O/2}\, e^{-i\tau H_E}\, e^{-i\tau H_O/2}, \tag{2.9}$$

where

$$H_O = \begin{pmatrix} W\epsilon_1 & V & & & & & & & \\ V & W\epsilon_2/2 & 0 & & & & & & \\ & 0 & W\epsilon_3/2 & V & & & & & \\ & & V & W\epsilon_4/2 & 0 & & & & \\ & & & 0 & \ddots & 0 & & & \\ & & & & 0 & W\epsilon_{L-1}/2 & V & & \\ & & & & & V & W\epsilon_L/2 & 0 & \\ & & & & & & 0 & 0 & \end{pmatrix}, \tag{2.10a}$$

and

$$H_E = \begin{pmatrix} 0 & 0 & & & & & & & \\ 0 & W\epsilon_2/2 & V & & & & & & \\ & V & W\epsilon_3/2 & 0 & & & & & \\ & & 0 & \ddots & 0 & & & & \\ & & & 0 & W\epsilon_{L-2}/2 & V & & & \\ & & & & V & W\epsilon_{L-1}/2 & 0 & & \\ & & & & & 0 & W\epsilon_L/2 & V & \\ & & & & & & V & W\epsilon_{L+1} & \end{pmatrix}, \tag{2.10b}$$

are tri-diagonal matrices that are block-diagonal, i.e. build up from 2×2 matrices. For simplicity of notation it has been assumed that L is even. In this section the subscripts O (odd) and E (even) have no particular meaning but are taken as such to keep the notation used here consistent with that of the following two sections. The block-diagonal structure of H_O or H_E simplifies the calculation of $e^{-i\tau H_O/2}$ and $e^{-i\tau H_E}$ tremendously, since the problem has essentially been reduced to the exponentiation of 2×2 matrices of the form

$$A_l = \begin{pmatrix} W_l & V \\ V & W_{l+1} \end{pmatrix}; \quad l = 1, \ldots, l+1, \tag{2.11}$$

where $W_1 \equiv W\epsilon_1$, $W_l \equiv W\epsilon_l/2$ for $2 \leqslant l \leqslant L$ and $W_{L+1} \equiv W\epsilon_{L+1}$. This is easily done by hand. Denoting the 2×2 matrix $B_l(\tau) \equiv e^{-i\tau A_l}$ the second-order approximant $E_2(\tau, H_O, H_E)$ can be

written as

$$E_2(\tau, H_O, H_E) = \begin{pmatrix} B_1(\tau/2) & 0 & \dots & & 0 \\ 0 & B_3(\tau/2) & & & \vdots \\ \vdots & & \ddots & & \\ & & & B_{L-1}(\tau/2) & \\ 0 & \dots & & & 0 \end{pmatrix} \times \begin{pmatrix} 0 & \dots & & & 0 \\ 0 & B_2(\tau) & 0 & & \vdots \\ \vdots & 0 & B_4(\tau) & & \\ & & & \ddots & \\ 0 & \dots & & & B_L(\tau) \end{pmatrix} \times \begin{pmatrix} B_1(\tau/2) & 0 & \dots & & 0 \\ 0 & B_3(\tau/2) & & & \vdots \\ \vdots & & \ddots & & \\ & & & B_{L-1}(\tau/2) & \\ 0 & \dots & & & 0 \end{pmatrix}. \tag{2.12}$$

Apparently the simple break-up $H = H_O + H_E$ leads to a so-called RS algorithm because at no stage it requires a Fourier transform of the wave function.

In d dimensions one can proceed in exactly the same manner. Write the Hamiltonian as a sum of $2d$ matrices $H = \sum_{q=1}^{2d} H_q$ and use (2.7) recursively to approximate [35] $e^{-i\tau H}$ by $e^{-i\tau H_1/2} \cdots e^{-i\tau H_{2d-1}/2} e^{-i\tau H_{2d-1}/2} \cdots e^{-i\tau H_1/2}$. In order to be useful in practice it should be possible to bring each of the matrices H_q into the same form as (2.10) by interchanging rows and columns.

What has been accomplished in this way is to replace the calculation of $e^{-i\tau H}$ by a much simpler, albeit approximate, expression which involves multiplication of the vector $\psi(t) = (\psi_1(t), \dots, \psi_{L+1}(t))^T$ by very sparse matrices only. To be more precise, for complex 2×2 matrices B_l, the number of operations required to compute $E_2(\tau, H_O, H_E)\psi(t)$ is $12L$ multiplications and $8L$ additions. For a d-dimensional system each of these two numbers has to be multiplied by d.

It should be clear by now that the matrix formulation is quite cumbersome to use, even for 1D problems. Extending the ideas put forward in this section to multidimensional systems would be nothing but a tedious exercise in writing down big matrices. Moreover it is conceivable that it would be hard to keep apart the ideas underlying the SPF approach and complications solely due to the desire to stick to the more conventional matrix notation. The reader interested in knowing how to extend the formalism to d-dimensional problems and higher-order SPF's should first get acquainted with the much more compact fermion-operator language and then continue with the next section. We stress that the use of fermion-operator algebra is forced upon us mainly because of its notational compactness and simplicity.

2.2. Second-order methods

Let us start by considering the simplest example, namely a free particle moving on an open-ended chain of $L+1$ sites. Its Hamiltonian is given by

$$K = V \sum_{l=1}^{L} (c_l^+ c_{l+1} + c_{l+1}^+ c_l). \tag{2.13}$$

A single-particle state can be represented as

$$|\Phi(t)\rangle = \sum_{l=1}^{L+1} \Phi_l(t) |l\rangle, \tag{2.14}$$

where $|l\rangle \equiv c_l^+ |0\rangle$. To compute the wave function at time $t+\tau$ from the wave function at time t, we must be able to calculate the action of the propagator $e^{-i\tau K}$ on an arbitrary state $|\Phi(t)\rangle$. Obviously, direct calculation of the propagator is feasible of the eigenvalue problem can be solved. This is of course the case for an operator of the form (2.13) which can be diagonalized by Fourier transformation. Putting

$$|k\rangle = \sqrt{\frac{2}{L+2}} \sum_{l=1}^{L+1} \sin\frac{\pi k l}{L+2} |l\rangle; \quad k = 1, \ldots, L+1, \tag{2.15}$$

it is readily shown that

$$e^{-i\tau K} = \sum_{k=1}^{L+1} |k\rangle \exp\left(-2i\tau V \cos\frac{\pi k}{L+2}\right) \langle k|, \tag{2.16}$$

and that propagation with $e^{-i\tau K}$ can be written as

$$\begin{aligned} e^{-i\tau K} |\Phi(t)\rangle &\equiv \sum_{l'=1}^{L+1} \Phi_{l'}(t+\tau) |l'\rangle \\ &= \sum_{l,l',k=1}^{L+1} |l'\rangle\langle l'|k\rangle \exp\left(-2i\tau V \cos\frac{\pi k}{L+2}\right)\langle k|l\rangle \Phi_l(t), \end{aligned} \tag{2.17}$$

or, equivalently

$$\Phi_{l'}(t+\tau) = \frac{2}{L+2} \sum_{k,l=1}^{L+1} \sin\frac{\pi k l'}{L+2} \exp\left(-2i\tau V \cos\frac{\pi k}{L+2}\right) \sin\frac{\pi k l}{L+2} \Phi_l(t). \tag{2.18}$$

This can also be written as

$$\Phi_{l'}(t+\tau) = F_S^{-1}\left\{\exp\left(-2i\tau V \cos\frac{\pi k}{L+2}\right) F_S\{\Phi_l(t)\}\right\}, \tag{2.19}$$

where $F_S\{\}$ denotes the (discrete) sine-transform and $F_S^{-1}\{F_S\{\}\} \equiv 1$. As is well-known, such transforms are most effectively performed by means of the Fast-Fourier Transform (FFT) algorithm.

In practice (2.19) means that we have to perform the FFT on the set of data $\{\Phi_l(t)\}$, multiply each element of the transformed set by $e^{-2i\tau V\cos[\pi k/(L+2)]}$ and perform another FFT to transform back to the coordinate representation. The number of operations required by this algorithm scales with the number of lattice points as $L(a + b \log L)$, where a and b are constants.

If the original problem is formulated in continuum space rather than on a lattice, one only has to replace in (2.19) $e^{-2i\tau V\cos[\pi k/(L+2)]}$ by $\exp[(i\tau V/2)(\pi k/(L+2))^2]$ as usual. This then leads to the split-step FFT [10–14,41] or multi-slice FFT [9] method.

As FFT routines usually require the number of points to be a power of 2, this may limit the range of applications. Furthermore, it is always worthwhile to search for methods that scale with L instead of $L \log L$ for large L. Such an algorithm can be devised by breaking up the Hamiltonian in local two-site contributions. Assuming that L is even for purely notational convenience, K can be decomposed as $K = K_O + K_E$ where [40]

$$K_O = V \sum_{l=0}^{L/2-1} (c_{2l+1}^+ c_{2l+2} + c_{2l+2}^+ c_{2l+1}), \tag{2.20a}$$

and

$$K_E = V \sum_{l=0}^{L/2-1} (c_{2l+2}^+ c_{2l+3} + c_{2l+3}^+ c_{2l+2}). \tag{2.20b}$$

The subscripts O and E refer to the odd and even sublattice, respectively. Invoking second-order formula (2.6) the free-particle propagator $e^{-i\tau K}$ can be approximated as

$$e^{-i\tau K} \approx E_2(\tau, K_O, K_E) = e^{-i\tau K_O/2}\, e^{-i\tau K_E}\, e^{-i\tau K_O/2}. \tag{2.21}$$

Noting that K_O (K_E) is the sum of commuting two-site operators immediately leads to

$$e^{-i\tau K_O/2}\, e^{-i\tau K_E}\, e^{-i\tau K_O/2} = \left(\prod_{l=0}^{L/2-1} e^{-i\tau K_{2l+1}/2}\right)\left(\prod_{l=0}^{L/2-1} e^{-i\tau K_{2l+2}}\right) \times \left(\prod_{l=0}^{L/2-1} e^{-i\tau K_{2l+1}/2}\right), \tag{2.22a}$$

where

$$K_l \equiv V(c_l^+ c_{l+1} + c_{l+1}^+ c_l). \tag{2.22b}$$

The key-point of this approach is to remark that according to (2.22), free-particle propagation is approximated by an ordered sequence of free-particle propagators of two-site systems. Clearly, each of these two-site propagators is easy to calculate analytically. The result is

$$e^{-i\tau K_l} = \mathbb{1} \cos \tau V - i(c_l^+ c_{l+1} + c_{l+1}^+ c_l) \sin \tau V, \tag{2.23}$$

where $\mathbb{1}$ stands for the identity operator. Remark that (2.23) is just an abstract representation of a unitary transformation in a two-dimensional vector space spanned by $(\Phi_l(t), \Phi_{l+1}(t))^{\mathrm{T}}$. Thus, we may equally well write

$$e^{-i\tau K_l}\begin{pmatrix}\Phi_l(t)\\ \Phi_{l+1}(t)\end{pmatrix} = \begin{pmatrix}\Phi_l(t+\tau)\\ \Phi_{l+1}(t+\tau)\end{pmatrix} = \begin{pmatrix}\cos \tau V & -i \sin \tau V\\ -i \sin \tau V & \cos \tau V\end{pmatrix}\begin{pmatrix}\Phi_l(t)\\ \Phi_{l+1}(t)\end{pmatrix}. \tag{2.24}$$

It is now easy to see how (2.22) operates on a given state $|\Phi(t)\rangle$ (or equivalently $\{\Phi_1(t), \ldots, \Phi_{L+1}(t)\}$). Letting the right-most factor of (2.22) act on $|\Phi(t)\rangle$ means that we have to take all pairs $\{\Phi_{2l+1}(t), \Phi_{2l+2}(t)\}$, $l = 0, \ldots, L/2 - 1$ and apply the two-dimensional transformation (2.24) (with τ replaced by $\tau/2$) to each of these pairs. Next we have to apply (2.24) to all pairs $\{\Phi_{2l+2}(t), \Phi_{2l+3}(t)\}$, $l = 0, \ldots, L/2 - 1$ and finally repeat the procedure (with τ replaced by $\tau/2$) for all pairs $\{\Phi_{2l+1}(t), \Phi_{2l+2}(t)\}$, $l = 0, \ldots, L/2 - 1$.

As the algorithm based on (2.22) does not require transforms to Fourier space but rather operates on the wave function in its real-space representation, it is a genuine RS method. Obviously, the number of operations required by this algorithm is proportional to L, but this not necessarily implies that it is more efficient than the FFT-based method. In the RS method free-particle propagation is approximated by a sequence of unitary transformations, each one involving two sites. The FFT method allows the exact (numerical) calculation of the effect of the free-particle propagator. Therefore for large L some compromise between the speed of calculation (which is better for the RS than for the FFT method) and the accuracy (which is worse for the RS than for the FFT method) has to be found. For the free-particle case this discussion is of academic interest only, since there clearly is no point to use an approximation scheme if an exact evaluation can be performed.

The ideas put forward for the free-particle system, extend easily to the case where there is interaction. For the 1D Hamiltonian $H = K + U$ with

$$K = V \sum_{l=1}^{L} (c_l^+ c_{l+1} + c_{l+1}^+ c_l), \tag{2.25a}$$

and

$$U = W \sum_{l=1}^{L+1} \epsilon_l n_l, \tag{2.25b}$$

a first second-order SPF algorithm is based on the approximation

$$e^{-i\tau H} \approx E_2(\tau, K, U) = e^{-i\tau K/2}\, e^{-i\tau U}\, e^{-i\tau K/2}. \tag{2.26}$$

Alternatively one could replace $e^{-\tau H}$ by $e^{-i\tau U/2}\, e^{-i\tau K}\, e^{-i\tau U/2}$ but, as explained in chapter 3, in practice this does not make much difference. From the preceding discussion it should be clear how to compute the effect of $E_2(\tau, K, U)$ on the state $|\Phi(t)\rangle$. Either the FFT or the RS algorithm can be used to perform the free-particle propagation ($e^{-i\tau K/2}$). Propagation with $e^{-i\tau U}$

is almost trivial: each component $\Phi_l(t)$ is to be multiplied with the site-dependent phase factor $e^{-i\tau W\epsilon_l}$. A full RS algorithm would correspond to the replacement

$$e^{-i\tau H} \approx e^{-i\tau K_O/4}\, e^{-i\tau K_E/2}\, e^{-i\tau K_O/4}\, e^{-i\tau U} e^{-i\tau K_O/4}\, e^{-i\tau K_E/2}\, e^{-i\tau K_O/4}. \tag{2.27}$$

In the rest of this paper this scheme will be denoted by RS_2. There is however an other possibility to set up a genuine RS scheme. Indeed, the Hamiltonian can be written as $H = H_O + H_E$ where [35]

$$H_O = V \sum_{l=1}^{L/2-1} \left[(c_{2l+1}^+ c_{2l+2} + c_{2l+2}^+ c_{2l+1}) + \frac{W}{2}(\epsilon_{2l+1} n_{2l+1} + \epsilon_{2l+2} n_{2l+2}) \right]$$

$$+ V(c_1^+ c_2 + c_2^+ c_1) + W\epsilon_1 n_1, \tag{2.28a}$$

and

$$H_E = V \sum_{l=0}^{L/2-2} \left[(c_{2l+2}^+ c_{2l+3} + c_{2l+3}^+ c_{2l+2}) + \frac{W}{2}(\epsilon_{2l+2} n_{2l+2} + \epsilon_{2l+3} n_{2l+3}) \right]$$

$$+ V(c_L^+ c_{L+1} + c_{L+1}^+ c_L) + W\epsilon_{L+1} n_{L+1}. \tag{2.28b}$$

Using this break-up the second-order RS algorithm would read

$$e^{-i\tau H} \approx e^{-i\tau H_O/2}\, e^{-i\tau H_E}\, e^{-i\tau H_O/2}. \tag{2.29}$$

Since H_O (H_E) is a sum of commuting two-site operators, the explicit expressions for $e^{-i\tau H_O/2}$ and $e^{-i\tau H_E}$ are easily calculated analytically. This scheme is identical to the one presented in the previous section and will be called RS_2' in the sequel.

From computational point of view, (2.29) is more efficient than (2.27) since it involves less multiplications by 2×2 matrices. However, as we will see, extending these concepts to the fourth-order methods will force us to decompose the Hamiltonian in kinetic and potential energy first, just as we did to derive (2.27). Note however that decomposing the Hamiltonian or kinetic energy into two blocks of commuting two-site operators is not the only way to set up a RS scheme. In principle any symmetrized product of exponential operators will do as indicated by (2.6). The main reason for adopting the "odd–even" approach is that the resulting algorithms are easy to vectorize and exhibit a high degree of parallelism.

Let us now address the question of how these ideas carry over to two- and three-dimensional systems. Consider the case of the free-particle system first. The kinetic energy operator T is itself a sum of d commuting kinetic energy operator T_e, i.e. $T = \Sigma_e T_e$, where the sum over e goes over the d unit vectors,

$$T_e = V \sum_{n \in \Lambda} (c_n^+ c_{n+e} + c_{n+e}^+ c_n), \tag{2.30}$$

and the sum over n runs over all lattice points of the d-dimensional hypercube. Since

$[T_e, T_{e'}] = 0$ for all e and e', the second-order RS scheme can be applied to each T_e separately. More concrete we can write

$$e^{-i\tau T} = \prod_e e^{-i\tau T_e} \approx \prod_e (e^{-i\tau T_{e,O}/2}\, e^{-i\tau T_{e,E}}\, e^{-i\tau T_{e,O}/2}). \tag{2.31}$$

Just as in the 1D case, the subscripts O and E refer to the $(d-1)$-dimensional odd- and even-labeled sublattice. Of course in the d-dimensional case the meaning of odd and even should not be taken too literally. What is meant is that each T_e is written as a sum of two contributions $T_e = T_{e,O} + T_{e,E}$ such that $T_{e,O}$ ($T_{e,E}$) is the sum of commuting two-site operators.

If the interaction $U = W\sum_{n\in\Lambda}\epsilon_n n_n$ is present there are at least two ways to proceed. One can first break up the hamiltonian into kinetic and potential energy and then follow the prescription of the preceeding paragraph to approximate the kinetic energy propagator. Alternatively one can write the Hamiltonian as a sum of $2d$ contributions, each contribution itself being a sum (over half of the lattice) of commuting two-site operators. If one is satisfied with the second-order accuracy of these schemes the latter is to be preferred because it is superior from the point of view of efficiency.

2.3. Fourth-order methods

Having discussed to considerable extent several second-order SPF schemes, we will now examine the possibility to construct more powerful, efficient algorithms by exploiting the systematics of the SPF approach. Our goal will be to increase the order in τ to which the SPF is correct and investigate whether such approximants are useful in practice. The first step in developing fourth-order SPF methods is to approximate $e^{-i\tau H} = e^{-i\tau(H_1+H_2)}$ by the unitary operator [34]

$$E_4(\tau, H_1, H_2) = e^{-i\tau H_1/2}\, e^{-i\tau H_2/2}\, e^{i\tau^3 C(H_1,H_2)}\, e^{-i\tau H_2/2}\, e^{-i\tau H_1/2}, \tag{2.32a}$$

where

$$C(H_1, H_2) \equiv [H_1 + 2H_2, [H_1, H_2]]/24. \tag{2.32b}$$

Just as for the second-order method one can prove that

$$\| e^{-i\tau H} - E_4(\tau, H_1, H_2) \| \leqslant c_4\tau^5 + \mathcal{O}(\tau^7), \tag{2.33}$$

where c_4 is a constant, its explicit form being given in appendix A. In general the complexity of the expression resulting from working out the double commutator appearing in (2.32) will be such that an additional approximation for $e^{i\tau^3 C(H_1,H_2)}$ is required. Writing $C(H_1, H_2) = \sum_{s=1}^{r} C_s$ and invoking the first-order approximation (2.1) yields

$$e^{i\tau^3 C(H_1, H_2)} \approx \prod_{s=1}^{r} e^{i\tau^3 C_s}. \tag{2.34}$$

Note that this does not affect the fourth-order correctness of (2.33) since according to (2.2) errors introduced by (2.34) are of order τ^6, as shown explicitly in appendix A. The resulting fourth-order SPF reads

$$\tilde{E}_4(\tau, H_1, H_2) = \mathrm{e}^{-\mathrm{i}\tau H_1/2}\,\mathrm{e}^{-\mathrm{i}\tau H_2/2}\left(\prod_{s=1}^{r}\mathrm{e}^{\mathrm{i}\tau^3 C_s}\right)\mathrm{e}^{-\mathrm{i}\tau H_2/2}\,\mathrm{e}^{-\mathrm{i}\tau H_1/2}, \tag{2.35}$$

and satisfies

$$\left\|\mathrm{e}^{-\mathrm{i}\tau H} - \tilde{E}_4(\tau, H_1, H_2)\right\| \leqslant c_4\tau^5 + \mathcal{O}(\tau^6), \tag{2.36}$$

so that the bound on the error of the fourth-order method will be of order τ^5 which is clearly much better than the τ^3 proportionality of the second-order algorithms. This of course does not immediately imply that the former methods are also more efficient than the latter. Indeed, when compared to for instance (2.29), the presence of the extra exponentials already indicates that it will require more computation to perform a time step using $\tilde{E}_4(\tau, H_1, H_2)$ than when $E_2(\tau, H_1, H_2)$ is invoked. On the other hand, since $\tilde{E}_4(\tau, H_1, H_2)$ can be expected to be more accurate than $E_2(\tau, H_1, H_2)$ integration of the TDSE can be done with larger time steps when $\tilde{E}_4(\tau, H_1, H_2)$ is employed. To summarize, whether or not fourth-order methods should be preferred to second-order schemes will depend on accuracy requirements, the length of the time interval over which the solution of the TDSE is to be obtained, and as will be discussed in more detail in chapter 3, on the amount of high-speed memory that can be accessed without making explicit or implicit (e.g. page-fault driven) I/O operations. Experience has shown that the fourth-order algorithms are much more efficient than their second-order counterparts provided sufficient memory is available.

So far the presentation has been quite general because we have not specified how the Hamiltonian should be decomposed nor did we discuss how to actually calculate $\tilde{E}_4(\tau, H_1, H_2)\,|\,\Phi(t)\rangle$. As before we consider the 1D case first. Chosing $H_1 = K$ and $H_2 = U$ with K and U given by (2.25), it follows directly from (B.4) that

$$\begin{aligned} C(K, U) &= \sum_{s=1}^{r} C_s \\ &= \frac{V^2 W}{12}\left(\sum_{l=1}^{L}(\epsilon_l - \epsilon_{l+1})n_l + \sum_{l=2}^{L+1}(\epsilon_l - \epsilon_{l-1})n_l\right) \\ &\quad + \frac{V^2 W}{24}\sum_{l=1}^{L-1}(\epsilon_{l+2} - 2\epsilon_{l+1} + \epsilon_1)(c_l^+ c_{l+2} + c_{l+2}^+ c_l) \\ &\quad - \frac{VW^2}{12}\sum_{l=1}^{L}(\epsilon_l - \epsilon_{l+1})^2(c_l^+ c_{l+1} + c_{l+1}^+ c_l), \end{aligned} \tag{2.37}$$

but this does not yet fixes C_s in a unique manner since the only requirement is that $C = \sum_{s=1}^{r} C_s$.

In view of the fact that there is no theoretical argument that can be of help to decide how to break up (2.37), it makes a lot of sense to optimize the choice of C_s with respect to computational demands. As we have seen repeatedly by now, it is easy to compute the effect on the wave function of products of propagators as long as each propagator involves most two sites. This then leads to the choice

$$C_1 = \frac{V^2W}{12}\left(\sum_{l=1}^{L}(\epsilon_l-\epsilon_{l+1})n_l+\sum_{l=2}^{L+2}(\epsilon_l-\epsilon_{l-1})n_l\right), \tag{2.38a}$$

$$C_2 = \frac{V^2W}{24}\sum{}'(\epsilon_{4l+3}-2\epsilon_{4l+2}+\epsilon_{4l+1})(c^+_{4l+1}c_{4l+3}+c^+_{4l+3}c_{4l+1}) + \frac{V^2W}{24}\sum{}'(\epsilon_{4l+4}-2\epsilon_{4l+3}+\epsilon_{4l+2})(c^+_{4l+2}c_{4l+4}+c^+_{4l+4}c_{4l+2}), \tag{2.38b}$$

$$C_3 = \frac{V^2W}{24}\sum{}'(\epsilon_{4l+5}-2\epsilon_{4l+4}+\epsilon_{4l+3})(c^+_{4l+3}c_{4l+5}+c^+_{4l+5}c_{4l+3}) + \frac{V^2W}{24}\sum{}'(\epsilon_{4l+6}-2\epsilon_{4l+5}+\epsilon_{4l+4})(c^+_{4l+4}c_{4l+6}+c^+_{4l+6}c_{4l+4}), \tag{2.38c}$$

$$C_4 = -\frac{VW^2}{12}\sum{}'(\epsilon_{2l+1}-\epsilon_{2l+2})^2(c^+_{2l+2}c_{2l+2}+c^+_{2l+2}c_{2l+1}), \tag{2.38d}$$

$$C_5 = -\frac{VW^2}{12}\sum{}'(\epsilon_{2l+2}-\epsilon_{2l+3})^2(c^+_{2l+2}c_{2l+3}+c^+_{2l+3}c_{2l+2}), \tag{2.38e}$$

where Σ' is a shorthand notation for the sum taken such that all subscripts appearing in the expression to be summed stay within the interval $[1,\ldots,L+1]$. Remark that each of the C_s, $s=2,\ldots,5$ is itself a sum of commuting, Hermitian operators. Hence, $\prod_{s=2}^{r} e^{i\tau^3 C_s}$ is nothing but an ordered product of two-site operators. The precise order in which these two-site operators appear is irrelevant. Clearly the calculation of $e^{i\tau^3 C(K,U)}$ is entirely performed in real space, without invoking Fourier transforms.

Having dealt with $e^{i\tau^3 C(K,U)}$, let us now focus on the remaining problem namely constructing a fourth-order RS method for computing $e^{-i\tau K/2}$. We only have to repeat the same steps as above but this time for $K = K_O + K_E$ instead of $H = K + U$ [39]. More explicitly we have

$$E_4(\tau/2,\ K_O,\ K_E) = e^{-i\tau K_O/4}\ e^{-i\tau K_E/4}\ e^{i\tau^3 C(K_O,K_E)/8}\ e^{-i\tau K_E/4}\ e^{-i\tau K_O/4}, \tag{2.39}$$

where the expression for $C(K_O, K_E)$ follows from (B.8). The next step is to approximate $e^{i\tau^3 C(K_O,K_E)}$ by a product of two-site propagators. The result is

$$\tilde{E}_4(\tau/2,\ K_O,\ K_E) = e^{-i\tau K_O/4}\ e^{-i\tau K_E/4}\left(\prod_{s=1}^{5} e^{i\tau^3 K_s/8}\right) e^{-i\tau K_E/4}\ e^{-i\tau K_O/4}, \tag{2.40}$$

where in this particular case it is most evident to chose

$$K_1 = -\frac{V^3}{12}\sum_{l=0}^{L/2-2}(c_{2l+1}^+ c_{2l+4} + c_{2l+4}^+ c_{2l+1}), \tag{2.41a}$$

$$K_2 = \frac{V^3}{6}\sum_{l=1}^{L/2-1}(c_{2l}^+ c_{2l+3} + 3c_{2l+3}^+ c_{2l}), \tag{2.41b}$$

$$K_3 = \frac{V^3}{12}\sum_{l=0}^{L/2-1}(c_{2l+2}^+ c_{2l+3} + c_{2l+3}^+ c_{2l+2}), \tag{2.41c}$$

$$K_4 = -\frac{V^3}{6}\sum_{l=0}^{L/2-1}(c_{2l+1}^+ c_{2l+2} + c_{2l+2}^+ c_{2l+1}), \tag{2.41d}$$

and

$$K_5 = \frac{V^3}{12}(c_1^+ c_2 + c_2^+ c_1) - \frac{V^3}{24}(c_L^+ c_{L+1} + c_{L+1}^+ c_L), \tag{2.41e}$$

since this break-up automatically divides the lattice in disjunct sublattices. This then concludes the construction of a fourth-order RS method for a 1D system. Algorithms based on this approach will be denoted by RS_4.

In two or more dimensions, the procedure is very similar. First split up H in kinetic (T) and potential (U) energy. Use expressions (B.4) with the appropriate values of the $t_{n,n'}$'s to approximate $e^{i\tau^3 C(T,U)}$ by a product of single-site and two-site propagators. From computational standpoint, i.e. vectorizability and parallelism, it is important to group these propagators such that all two-site propagators in each group commute with each other and the product of them acts on as many lattice sites as possible. To approximate the kinetic energy propagator $e^{-i\tau T}$ we first use the fact that $T = \Sigma_e T_e$, where $T_E = V\Sigma_{n\in\Lambda} t_{n,n+e}(c_n^+ c_{n+e} + c_{n+e}^+ c_n)$ and that $[T_e, T_{e'}] = 0$ for all e and e'. Consequently this problem reduces to approximating the kinetic energy propagator of a particle moving on a chain. In other words we simply have to repeat the procedure for K, presented above, for each of the d dimensions of the lattice.

Although the fourth-order scheme outlined above is the one that has been found to be most suitable in actual applications, it is nevertheless of interest to investigate other ways of constructing fourth-order SPF algorithms. As already indicated in the previous section there is no reason to start by breaking up the Hamiltonian into kinetic and potential energy and then decompose the kinetic energy further into odd and even sublattice operators. Instead one can write the Hamiltonian itself as the sum of odd and even sublattice operators. For a 1D system the fourth-order approximant reads

$$E_4(\tau, H_O, H_E) = e^{-i\tau H_O/2}\, e^{-i\tau H_E/2}\, e^{i\tau^3 C(H_O,H_E)}\, e^{-i\tau H_E/2}\, e^{-i\tau H_O/2}, \tag{2.42}$$

Again, further approximation of $e^{i\tau^3 C(H_O, H_E)}$ in terms of two-site propagators is required. The most obvious choice is

$$\tilde{E}_4(\tau, H_O, H_E) = e^{-i\tau H_O/2}\, e^{-i\tau H_E/2}\left(\prod_{s=1}^{5} e^{i\tau^3 A_s}\right) e^{-i\tau H_E/2}\, e^{-i\tau H_O/2}, \tag{2.43}$$

where, for the more simple case of *periodic-boundary conditions*,

$$A_1 = \frac{1}{24}\sum_{l=0}^{L/2-1}\left[(2V^3 + VW^2\epsilon_{2l+3}^2 - 2VW^2\epsilon_{2l+2}\epsilon_{2l+3})(c_{2l+2}^+ c_{2l+3} + c_{2l+3}^+ c_{2l+2})\right.$$
$$\left. + 2V^2W(\epsilon_{2l+2} + 2\epsilon_{2l+3})n_{2l+2}\right], \tag{2.44a}$$

$$A_2 = \frac{1}{24}\sum_{l=0}^{L/2-1}\left[(-4V^3 - 2VW^2\epsilon_{2l+2}^2 + VW^2\epsilon_{2l+1}\epsilon_{2l+2})(c_{2l+1}^+ c_{2l+2} + c_{2l+2}^+ c_{2l+1})\right.$$
$$\left. - 2V^2W(\epsilon_{2l+2} + 2\epsilon_{2l+1})n_{2l+1}\right], \tag{2.44b}$$

$$A_3 = \frac{V^2W}{24}\sum_{l=0}^{L/2-1}(\epsilon_{2l+3} + 4\epsilon_{2l+4} - 2\epsilon_{2l+2})(c_{2l+2}^+ c_{2l+3} + c_{2l+3}^+ c_{2l+2})$$
$$+ \frac{V^2W}{24}\sum_{l=0}^{L/2-1}(\epsilon_{2l+1} - 2\epsilon_{2l+2} - 2\epsilon_{2l+3})(c_{2l+1}^+ c_{2l+3} + c_{2l+3}^+ c_{2l+1}), \tag{2.44c}$$

$$A_4 = \frac{V^3}{6}\sum_{l=0}^{L/2-1}(c_{2l+2}^+ c_{2l+5} + c_{2l+5}^+ c_{2l+2}), \tag{2.44d}$$

and

$$A_5 = -\frac{V^3}{12}\sum_{l=0}^{L/2-1}(c_{2l+1}^+ c_{2l+4} + c_{2l+4}^+ c_{2l+1}). \tag{2.44e}$$

As will be demonstrated in chapter 4, for 1D problems this approximant, to be called RS_4', is more efficient than all others. Unfortunately it has a drawback. Extending the calculation of $C(H_O, H_E)$ to the case of two- or three-dimensional systems is a horrible task. In other words this approach lacks the modularity of the fourth-order scheme based on a T–U decomposition and therefore it is also much harder to implement.

3. Implementation

The main advantage of explicit algorithms is that they are relatively easy to implement. The awkward expressions appearing in chapter 2 suggest that explicit schemes based on symmetrized

product-formula's (SPF's) do not share this appealing feature. The purpose of this chapter is to demonstrate that once taken the formalism for granted, translating the abstract representation of a SPF of a propagator into computer code is straightforward, even for three-dimensional problems.

It is obvious that it is important to search for methods that need as few arithmetic operations per time step as possible, but one should not expect that one can do better than keep the number of operations proportional to the number of lattice (or mesh) points L^d. In this respect RS methods are optimal since each time step requires $\mathcal{O}(L^d)$ multiplications or additions. Algorithms based on FFT's need $\mathcal{O}(L^d \log L^d)$ operations but as already explained in chapter 2, are more accurate.

Another point of concern is the size of memory necessary to store the wave function and propagators. Also from this viewpoint it is desirable to devise algorithms the memory demands of which scale linearly with the number of lattice points. Obviously, storage of the wave function requires $\mathcal{O}(L^d)$ bytes. To store the full propagator $e^{-i\tau H}$ one would need $\mathcal{O}(L^d) \times \mathcal{O}(L^d)$ bytes and this clearly limits the range of application considerably. The sparseness of the propagators constructed by means of SPF reduces this amount to $\mathcal{O}(L^d)$, the same as for the wave function. As the fourth-order SPF approximants involve more exponential operators than a second-order scheme, it can be expected that the latter will need less memory than the former.

As our main design objective has been to minimize the computation time per time step, constants representing non-zero matrix elements of the propagators are calculated once and then kept available in high-speed memory. In case the size of high-speed memory proves to be insufficient, one might contemplate using external storage either by exploiting some of the features offered by virtual memory systems or by performing explicit I/O. In most of our own applications, it was essential to run the code at the highest speed attainable and consequently the idea of using external storage had to be abandonned.

It is also worthwhile to pay some attention to the order in which numbers are fetched from memory. The presence in scalar or vector processors of cache memory or of a virtual memory system suggests that it may be more efficient to access memory sequentially, with relatively small strides, instead of "almost" randomly. RS algorithms offer the opportunity to do this at no cost. Although on some processors the gain obtained by taking into account this property of "locality" may be marginal, on others such as the CYBER 205 or the IBM 3090/VF it may prove to be substantial.

In general the wave function of the d-dimensional system is represented by L^d complex numbers $\{\psi_1(t), \ldots, \psi_{L^d}(t)\}$. These complex numbers can be stored as an array of complex variables or as two arrays of real variables. For reasons to be explained below, the latter storage method is to be preferred. Then there are still two different ways to store the real (imaginary) parts in an array. For a d-dimensional system, one could decide to make use of d-dimensional system, one could decide to make use of d-dimensional arrays. It is well-known that this way of storing elements is not optimal, certaintly not for vector operations, because the elements do not occupy consecutive memory locations (except when the size of the lattice is as large as the dimension of the array). Hence it is more appropriate to consider the L^d numbers representing the real (or imaginary) part of the wave function as one long one-dimensional array and to take care of the explicit address calculation oneself. In our implementation we have adopted the most obvious convention, namely the index of the array to a particular site is calculated as $i_x + (i_y -$

$1)L_x + (i_z - 1)L_xL_y$ where $i_\alpha(L_\alpha)$, $\alpha = x, y, z$ is the index (linear size) in (of) the α-direction.

Section 3.1 is devoted to the implementation on a scalar machine of second- and fourth-order RS algorithms. It will be shown that all necessary calculations can be performed by using three different procedures, the FORTRAN code of which is given. In section 3.2 the vectorized version of the computational kernel, written for the CYBER 205, is discussed. From its simplicity it will be clear that it is not difficult to adapt this code for use on other vector machines such as the CRAY 1S or IBM 3090/VF. In chapter 4 an evaluation is given of the performance of scalar and vectorized codes, by combining the results of a detailed analysis of the accuracy of the different schemes with measurements of CPU times necessary to perform the calculations.

Before presenting details of the algorithms, a general remark is in order. Consider for instance the RS_2 scheme. The time-step operator is then $e^{-\tau K/2}\, e^{-\tau U}\, e^{-\tau K/2}$. In many applications one is not interested in knowing the wave function at each time step but rather after a number, say n, time steps. Then by trivial rearrangement $(e^{-\tau K/2}\, e^{-\tau U} e^{-\tau K/2})^n = e^{-\tau K/2}\, e^{-\tau U}(e^{-\tau K}\, e^{-\tau U})^{n-1}\, e^{-\tau K/2}$ showing that it is possible to reduce the amount of computation in this manner. Note, however, that since RS_2 is employed to compute the propagators $e^{-\tau K/2}$ and $e^{-\tau K}$, it is to be expected that the error resulting from the application of $(e^{-\tau K_O/4}\, e^{-\tau K_E/2}\, e^{-\tau K_O/4}\, e^{-\tau U}\, e^{-\tau K_O/4} \times e^{-\tau K_E/2}\, e^{-\tau K_O/4})^n$ is smaller that the error coming from the replacement of $e^{-in\tau H}$ by $e^{-\tau K_O/4}\, e^{-\tau K_E/2}\, e^{-\tau K_O/4}\, e^{-\tau U}(e^{-\tau K_O/2}\, e^{-\tau K_E}\, e^{-\tau K_O/2}\, e^{-\tau U})^{n-1}\, e^{-\tau K_O/4}\, e^{-\tau K_E/2} e^{-\tau K_O/4}$ because in the latter, the approximant to $e^{-i\tau K}$ will not be "as good" as the approximant to $e^{-i\tau K/2}$. Using this rearrangement of exponentials in the implementation of RS_2 or RS_2' also implies that it will not make much difference if the role of K and U is interchanged, as long as the number of intermediate steps n is large compared to the number of time values for which the wave function is to be known. For the fourth-order RS algorithm there is never a substantial difference if one interchanges K and U.

3.1. Scalar algorithm

The simplest calculation that has to be performed is definitely the propagation by $e^{-i\tau U}$, with $U = W\Sigma_{n \in \Lambda}\epsilon_n n_n$. Since U consists of operators that commute, letting $e^{-i\tau U}$ act on the wave function is tantamount to multiplying each element of the wave function $\psi_n(t)$ by a phase factor $e^{-i\tau W\epsilon_n}$. The scalar code that realizes these fairly simple manipulations is shown below.

```
      subroutine expU(psiR,psiI,XR,XI,l)
      real * 8 psiR( * ),psiI( * ),XR( * ),XI( * )
c
      do i = 1,l
      r0 = psiR(i)
      r1 = psiI(i)
      c = XR(i)
      s = XI(i)
      psiR(i) = c * r0 + s * r1
      psiI(i) = c * r1 − s * r0
      end do
      return
      end
```

It is assumed that the propagator $e^{-i\tau U}$ is already stored in the arrays XR (real part) and XI (minus the imaginary part) using the same storage scheme as the one used for the wave function. One reason for keeping real and imaginary part apart is that on the machine on which the original code was developed, a VAX 8200, this was forced upon us because the FORTRAN compiler (VAX FORTRAN V4.5) did not generate optimal code (i.e. it coded calls to library routines) for executing double precision (COMPLEX * 16) multiplications. Furthermore, anticipating migration of the code to vector machines, the use of complex variables is to be avoided as much as possible.

A piece of code that performs all calculations for the approximate free-particle propagator $e^{-i\tau T_{y,O}/2}\, e^{-i\tau T_{y,E}}\, e^{-i\tau T_{y,O}/2}\, e^{-i\tau T_{x,O}/2}\, e^{-i\tau T_{x,E}}\, e^{-i\tau T_{x,O}/2}$ (see (2.31)) for a 2D system is given below.

```
c Free-electron propagation in x-direction
      call expK(psiR,psiI,c2,s2,1,lx-1,2,1,1,lx,1,lx)
      call expK(psiR,psiI,c1,s1,2,lx-1,2,1,1,lx,1,lx)
      call expK(psiR,psiI,c2,s2,1,lx-1,2,1,1,lx,1,lx)
c Free-electron propagation in y-direction
      call expK(psiR,psiI,c2,s2,1,lx,1,lx,1,lx-1,2,lx)
      call expK(psiR,psiI,c1,s1,1,lx,1,lx,2,lx-1,2,lx)
      call expK(psiR,psiI,c2,s2,1,lx,1,lx,1,lx-1,2,lx)
      ...
      subroutine expK(psiR,psiI,cosin,sinus
     1                ,ix0,ix1,ix2,ix3,iy0,iy1,iy2,iy3)
      real * 8 psiR( * ),psiI( * )
c
      do j0 = iy0,iy1,iy2
      j1 = (j0 - 1) * iy3
      do i0 = ix0,ix1,ix2
      i = i0 + j1
      j = i + ix3
      r0 = psiR(i)
      r1 = psiI(i)
      r2 = psiR(j)
      r3 = psiI(j)
      psiR(i) = r0 * cosin + r3 * sinus
      psiI(i) = r1 * cosin - r2 * sinus
      psiR(j) = r2 * cosin + r1 * sinus
      psiI(j) = r3 * cosin - r0 * sinus
      end do
      end do
      return
      end
```

In the code that calls expK, c1 = cos τV, s1 = sin τV, c2 = cos($\tau V/2$), s2 = sin($\tau V/2$) and lx is the linear size x- and y-direction. For simplicity it has been assumed that the lattice is isotropic (i.e. interchanging the x- and y-direction is a symmetry operation of the system). The action of

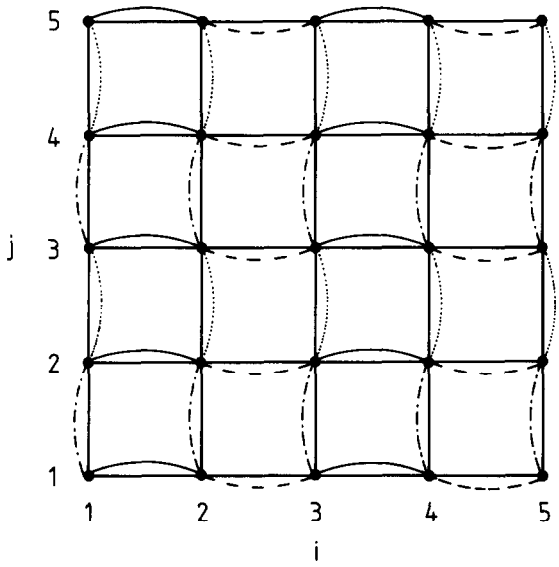

Fig. 3.1. Selection of pairs of lattice points resulting from the application of the second-order symmetrized product-formula algorithm RS_2.

the four different propagators $e^{-i\tau T_{x,O}/2}$, $e^{-i\tau T_{x,E}}$, $e^{-i\tau T_{y,O}/2}$ and $e^{-i\tau T_{y,E}}$ is to replace pairs of coefficients of the wave function $\psi_n(t)$ by new pairs. Which pairs are taken is determined by the variables ix0,...,ix3 and iy0,...,iy3 as illustrated by the code that actually implements the 2D RS scheme for the free-particle propagator. Instead of determining which elements are selected by studying the code, it is easier to understand what happens by looking at a graphical representation of the 2D lattice.

In fig. 3.1 I have drawn a 5×5 lattice and indicated which lattice points are grouped in pairs for each of the four different propagators. To compute $e^{-i\tau T_{x,O}/2}$, the elements of the set $\{\psi_n(t)\}$ are grouped into pairs, as indicated by the solid curved lines. The unitary transformation (2.24) is then applied to each of these applied to each of these pairs. This completes the calculation of $e^{-i\tau T_{x,O}/2}|\psi(t)\rangle$. The same steps are to be repeated to perform propagation by $e^{-i\tau T_{x,E}}$ but instead of grouping pairs according to the solid curves, the two elements are to be chosen such as to be chosen such as to be interconnected by the dashed curves. The procedure for $e^{-i\tau T_{y,O}/2}$ or $e^{-i\tau T_{y,E}}$ is similar. Pairs of lattice points are selected as indicated by the dashed-dotted and dotted curves respectively. Extending expK to handle also three-dimensional problems is almost trivial. One just needs to add another DO-loop and four variables that play a role similar to ix0, ix1, ix2 and ix3.

The computational kernels expU and expK (properly generalized) suffice to implement a d-dimensional second-order RS algorithm based on the break up in kinetic and potential energy. From the discussion of section 2.2 it follows that it is more efficient to decompose H itself rather than first break up H into kinetic and potential energy and then invoke the second-order SPF again to compute $e^{-i\tau T}$ by a RS algorithm. Generalizing expK to perform these calculations is straightforward. Selection of pairs of lattice points remains unaltered but instead of the 2×2 matrix appearing in (2.24), a slightly more complicated 2×2 matrix will enter in the transforma-

tion formula of each pair. The complex-valued elements of such a matrix will depend on the site indices via the potential energy $W\epsilon_n$. Of course this requires an appropriate modification of the code of expK. As a matter of fact the modified code, called expC is exactly the same as the one needed to implement the full fourth-order scheme and is presented below.

```
      subroutine expC(psiR,psiI,XR,XI
     1                ,ix0,ix1,ix2,ix3,iy0,iy1,iy2,iy3)
      real * 8 psiR(*),psiI(*),XR(*),XI(*)
c
      do j0 = iy0,iy1,iy2
      j1 = (j0 - 1)*iy3
      do i0 = ix0,ix1,ix2
      i = i0 + j1
      j = i + ix3
      cosin = XR(i)
      sinus = XI(i)
      r0 = psiR(i)
      r1 = psiI(i)
      r2 = psiR(j)
      r3 = psiI(j)
      psiR(i) = r0 * cosin + r3 * sinus
      psiI(i) = r1 * cosin - r2 * sinus
      psiR(j) = r2 * cosin + r1 * sinus
      psiI(j) = r3 * cosin - r0 * sinus
      end do
      end do
      return
      end
```

It is seen to be almost identical to expK except that instead of having cosin and sinus as transformation coefficients, the values of these coefficients depend on the indices of the two lattice sites involved.

Implementation of the fourth-order RS scheme for the free-particle propagator $e^{-i\tau T}$ proceeds along the same line. Remark however that because of the free boundary conditions, care has to be taken of some correction terms, i.e. the third term in (B.8a)and (B.8b) respectively. The code that implements the fourth-order RS scheme (2.40) for a 2D free-particle propagator is shown below.

```
c Fourth-order algorithm for the 2D free-particle propagator
      if (lx.gt.1) then
c Free-electron propagation in x-direction
c nearest-neighbor terms: e^{-i\tau K_O/4}
      call expK(psiR,psiI,cX1,sX1,1,lx - 1,2,1,1,ly,1,lx)
c nearest-neighbor terms: e^{-i\tau(K_E/4-\tau^2 K_3/8)}
      call expK(psiR,psiI,cX2,sX2,2,lx - 1,2,1,1,ly,1,lx)
```

```
c nearest-neighbor terms: e^{iτ^3 K_4/8}
      call expK(psiR,psiI,cX3,sX3,1,lx − 1,2,1,1,ly,1,lx)
c correction terms: e^{iτ^3 K_5/8}
      call expK(psiR,psiI,cX6,sX6,1,2,2,1,1,ly,1,lx)
      call expK(psiR,psiI,cX7,sX7,lx − 1,lx,2,1,1,ly,1,lx)
c next-next-nearest-neighbor terms: e^{iτ^3 K_1/8}
      call expK(psiR,psiI,cX4,sX4,1,lx − 3,2,3,1,ly,1,lx)
c next-next-nearest-neighbor terms: e^{iτ^3 K_2/8}
      call expK(psiR,psiI,cX5,sX5,2,lx − 3,2,3,1,ly,1,lx)
c nearest-neighbor terms: e^{−iτK_E/4}
      call expK(psiR,psiI,cX1,sX1,2,lx − 1,2,1,1,ly,1,lx)
c nearest-neighbor terms: e^{−iτK_O/4}
      call expK(psiR,psiI,cX1,sX1,1,lx − 1,2,1,1,ly,1,lx)
      endif
c
      if(ly.gt.1) then
c Free-electron propagation in y-direction
c nearest-neighbor terms: e^{−iτK_O/4}
      call expK(psiR,psiI,cY1,sY1,1,lx,1,lx,1,ly − 1,2,lx)
c nearest-neighbor terms: e^{−iτ(K_E/4−τ^2 K_3/8)}
      call expK(psiR,psiI,cY2,sY2,1,lx,1,lx,2,ly − 1,2,lx)
c nearest-neighbor terms: e^{iτ^3 K_4/8}
      call expK(psiR,psiI,cY3,sY3,1,lx,1,lx,1,ly − 1,2,lx)
c correction terms: e^{iτ^3 K_5/8}
      call expK(psiR,psiI,cY6,sY6,1,lx,1,lx,1,1,2,lx)
      call expK(psiR,psiI,cY7,sY7,1,lx,1,lx,ly − 1,ly,2,lx)
c next-next-nearest-neighbor terms: e^{iτ^3 K_1/8}
      call expK(psiR,psiI,cY4,sY4,1,lx,1,3 * lx,1,ly − 3,2,lx)
c next-next-nearest-neighbor terms: e^{iτ^3 K_2/8}
      call expK(psiR,psiI,cY5,sY5,1,lx,1,3 * lx,2,ly − 3,2,lx)
c nearest-neighbor terms: e^{−iτK_E/4}
      call expK(psiR,psiI,cY1,sY1,1,lx,1,lx,2,ly − 1,2,lx)
c nearest-neighbor terms: e^{−iτK_O/4}
      call expK(psiR,psiI,cY1,sY1,1,lx,1,lx,1,ly − 1,2,lx)
      endif
```

For a change here it has not been assumed that the lattice is isotropic, as is reflected by the appearance of the linear size in the y-direction (ly), and the possibly different values of the transformation coefficients cY1 etc. The example above demonstrates very clearly the modularity of the SPF approach. All operations are done with the same routine. Pairs of lattice sites are picked out in different ways and the "cosine" and "sine" of the plane rotation is chosen accordingly. The precise values of the transformation cosines cX1, ..., cY7 and sines sX1, ..., sY7 follow directly from (2.41). Remark that this fourth-order algorithm for the free-particle propagator does not require a lot of extra high-speed memory. Except for the wave function

itself, storage of which requires $\mathcal{O}(L^d)$ floating-point words, one only needs to store the cosines and sines and this takes only $\mathcal{O}(1)$ floating-point words. Also note that the order in which the K_s appear in (2.40) is different from the order chosen to implement the algorithm, the reason being that in this way one pass over the lattice can be eliminated without losing accuracy.

Let us now focus on the implementation of the first of the full fourth-order algorithms proposed in section 2.3 and consider the 1D case first. From the standpoint of programming, the new problem is to perform propagation by $\prod_{s=1}^{r} e^{i\tau^3 C_s}$ with C_s given by (2.38). By inspection of (2.38) it directly follows that we already know how to compute $e^{i\tau^3 C_1}$, since we can simply use the routine expU, assuming that the arrays XR and XI contain the appropriate values of the phase factors.

To compute $e^{i\tau^3 C_s}$, $s = 2,\ldots,5$ the routine expC can be used. For instance, for 1D system $XR(l) = \cos[VW^2\tau^3(\epsilon_l - \epsilon_{l+1})^2/12]$ and $XI(l) = \sin[VW^2\tau^3(\epsilon_l - \epsilon_{l+1})^2/12]$ in the case of $e^{i\tau^3 C_4}$ or $e^{i\tau^3 C_5}$. It is not difficult to imagine that propagation with $e^{i\tau C_2}$ and $e^{i\tau C_3}$ can be done in a similar manner, the main difference being that the distance between two lattice sites, belonging to a pair, is not one but two lattice spacings.

In the general, d-dimensional case, the situation is not much more complicated. The new feature is the presence in $C(T, V)$ of terms involving sums with $\boldsymbol{e} > \boldsymbol{e}'$ (see appendix B). Taking care of such terms is a matter of picking out the correct pairs of lattice sites. Note that storage of the matrix elements of all the various propagators $e^{i\tau^3 C_s}$ requires $2L^d$ floating-point variables per propagator.

To summarize, the three scalar routines expU, expK and expC suffice to construct any of the RS algorithms proposed in chapter 2. The number of operation required to perform a single time step increases linearly with the number of lattice points L^d. The particular choice of the RS schemes is such that recursive manipulation of data is avoided.

3.2. *Vectorized algorithm*

The explicit non-recursive character of RS_2 and RS_4 suggests that the computational kernels are vectorizable to a high degree and this is indeed the case. A functionally equivalent vectorized code of expU is given below.

```
      SUBROUTINE TDSE5A(XR,XI)
C     INCLUDE 'COMMON'
      PARAMETER (MAXVEC = 65535)
      REAL XR(MSI),XI(MSI)
C
      I = 1
      N = MAXVEC
      J = (NSI - 1)/N
      DO 1 L = 0,J
      IF(L.EQ.J) THEN
      N = NSI - I + 1
      ENDIF
      TMP1(I;N) = PSIR(I;N) * XI(I;N)
```

```
      TMP2(I;N) = PSII(I;N) * XI(I;N)
      PSIR(I;N) = PSIR(I;N) * XR(I;N)
      PSII(I;N) = PSII(I;N) * XR(I;N)
      PSIR(I;N) = PSIR(I;N) + TMP2(I;N)
      PSII(I;N) = PSII(I;N) - TMP1(I;N)
1     I = I + N
      RETURN
      END
```

This routine was written for use on CYBER 205 but is so simple that it is easily converted to run on any type of vector machine. All it does is multiply element by element two complex-valued arrays of length NSI. The number of lattice sites NSI the arrays representing the wave function (PSIR,PSII) as well as the temporary arrays (TMP1,TMP2) are passed as common variables. The arrays XR and XI are supposed to contain the real and imaginary part of the phase factors (e.g. $e^{-i\tau\epsilon_n}$) respectively. The DO-loop can be left out if the number of lattice sites NSI is always less than the maximum vector length (i.e. $2^{16} - 1$ on the CYBER 205).

Vectorization of the routine expK needs some extra attention if optimal use of the vector facilities is to be made. On the CYBER 205, even the most inner DO-loop of expK is not directly vectorizable because of the variable stride ix2. Ideally, we want the machine to execute vector instructions on vectors of length equal to the number of lattice points, just as in the previous example. From the discussion in section 3.1 it directly follows that this will be possible if all pairs of lattice sites are suitably arranged in vectors. This can be accomplished in two different ways, either by gather operations or be selecting pairs of lattice sites by means of control vectors. The first approach requires a lot of extra memory because all index vectors have to be stored somewhere when repeated calculation of these indices is to be avoided. In the second approach the control variables take only two values and this can be exploited to reduce memory requirements considerably, especially on the CYBER 205 where BIT-type arrays can be used to store the control vectors. For this reason the control-vector approach was preferred over the gather technique. For the model systems studied in this paper, it is also more efficient.

The vectorized code that performs the same calculations as expK is shown below.

```
      SUBROUTINE TDSE5B(COSIN,SINUS,BIT,LBIT,OFFSET)
C     INCLUDE 'COMMON'
      PARAMETER (MAXVEC = 65535)
      BIT BIT(MSI)
      INTEGER OFFSET
      I = 1
      N = MAXVEC - OFFSET
      J = (LBIT - 1)/N
      DO 2 L = 0, J
      IF(L.EQ.J) THEN
      N = LBIT - I + 1
      ENDIF
      N0 = N + OFFSET
      I0 = I + OFFSET
```

```
      TMP1(I;N0) = PSIR(I;N0) * COSIN
      TMP2(I;N0) = PSII(I;N0) * SINUS
      TMP3(I;N0) = PSIR(I;N0) * SINUS
      TMP4(I;N0) = PSII(I;N0) * COSIN
      WHERE(BIT(I;N))
      PSIR(I;N) = TMP1(I;N) + TMP2(I0;N)
      PSII(I;N) = TMP4(I;N) - TMP3(I0;N)
      PSIR(I0;N) = TMP1(I0;N) + TMP2(I;N)
      PSII(I0;N) = TMP4(I0;N) - TMP3(I;N)
      ENDWHERE
2     I = I + N
      RETURN
      END
```

The smallest of the two lattice-site indices runs over all lattice sites and the other element of the pair is determined by adding an offset "OFFSET" to this index. Whether the pair of lattice sites is actually replaced by a new one is controlled by the value of the corresponding element of the control vector BIT. As seen from the code above, only one half of all operations are done under control of the BIT-vector. Strictly counting all operations learns that in the first four vector instructions, there are redundant multiplications, due to fact that the presence of boundaries was not taken into account. The amount of time lost by doing these useless multiplications is more than just compensated for by the gain in speed resulting from the possibility to perform vector operations on (very) long vectors. For completeness it should be mentioned that the variable LBIT determines the length of the vector operation.

As in the scalar version, the two procedures TDSE5A and TDSE5B suffice to construct RS_2 and RS_4 algorithm for the free-particle propagator. To be able to implement the full fourth-order scheme an additional piece of code is necessary. As pointed out already, what is needed is a routine that does quite similar things as TDSE5B except that the value of the variables COSIN and SINUS changes with the lattice-site index. The code that performs the necessary calculations is given below.

```
      SUBROUTINE TDSE5C(XR,XI,BIT,LBIT,OFFSET)
C     INCLUDE 'COMMON'
      PARAMETER (MAXVEC = 65535)
      BIT BIT(MSI)
      INTEGER OFFSET
      REAL XR(MSI),XI(MSI)
C
      I = 1
      N = MAXVEC-OFFSET
      J = (LBIT - 1)/N
      DO 3 L = 0,J
      IF(L.EQ.J) THEN
      N = LBIT - I + 1
      ENDIF
```

```
      N0 = N + OFFSET
      I0 = I + OFFSET
      I1 = I-OFFSET
      TMP1(I;N0) = Q8VMASK(XR(I;1),XR(I1;1),BIT(I;N0);TMP1(I;N0))
      TMP2(I;N0) = Q8VMASK(XI(I;1),XI(I1;1),BIT(I;N0);TMP2(I;N0))
      TMP3(I;N0) = PSIR(I;NO) * TMP2(I;N0)
      TMP4(I;N0) = PSII(I;N0) * TMP1(I;N0)
      TMP1(I;N0) = PSIR(I;N0) * TMP1(I;N0)
      TMP2(I;N0) = PSII(I;N0) * TMP2(I;N0)
      WHERE(BIT(I;N))
      PSIR(I;N) = TMP1(I;N) + TMP2(I0;N)
      PSII(I;N) = TMP4(I;N) - TMP3(I0;N)
      PSIR(I0;N) = TMP1(I0;N) + TMP2(I;N)
      PSII(I0;N) = TMP4(I0;N) - TMP3(I;N)
      ENDWHERE
3     I = I + N
      RETURN
      END
```

The similarity with TDSE5B is clear. The "cosines" for each pair are obtained from XR by merging, under control of the vector BIT, XR with XR, properly shifted by the offset OFFSET. The "sines" are determined from XI in exactly the same manner. These two vector operations could be eliminated at the cost of twice as much memory. The remaining eight vector operations are the same as those in TDSE5B.

The three routines TDSE5A, TDSE5B and TDSE5C constitute the computational kernel of the vectorized fourth-order RS algorithm. Except for the implementation of the correction terms, reflecting boundary effects (the last term of (B.8a) and (B.8b)), the fourth-order scheme is realized by successive calls to these three routines. As the number of correction terms is $\mathcal{O}(L^{d-1})$ (with a small prefactor!) there is reason to claim that the fourth-order RS algorithm is vectorizable to a very high degree. Note that it is not difficult to distribute almost all operations over a large number of parallel processors. As we did not have the opportunity to investigate the practical consequences of this inherent parallelism of SPF based algorithms, we will not discuss this aspect any further.

4. Performance analysis

To evaluate the performance of a particular scalar or vector code two different aspects have to be taken into consideration: 1) The computational resources (CPU time, memory, etc.) it takes to integrate the TDSE (Time-Dependent Schrödinger Equation) over a specified time interval $[0, t = m\tau]$, and 2) the loss of accuracy, due to the approximate nature of the integration scheme used.

For a fixed number of time steps (m) and time (t), it is clear that a more accurate integration method will require more computational resources than a less accurate scheme of the same type.

However the gain in accuracy, resulting from the use of a higher-order scheme can be exploited such as to reduce the amount of computation by taking a larger time step or, in other words, by doing less steps. In this reasoning it has been assumed that the maximum error on the properties of interest (amplitudes, energy, etc.) that can be tolerated is fixed. To decide which scheme is the most efficient demands a careful study of the global errors of and the resource used by the different algorithms. This chapter addresses questions related to the efficiency of SPF (symmetrized product-formula) based algorithms for solving the TDSE.

In section 4.1 the simplest case, that of the free-particle propagator, is studied in detail. As is well-known (see for example the discussion in section 2.2), a numerically exact calculation of the free-particle propagator can be carried out by means of Fourier transforms. For systems with free-boundary conditions, the most common situation, only the sine transform of the wave function has to be calculated (see section 2.2). In order to compute such transforms efficiently, i.e. with much less than $\mathcal{O}(L^d \times L^d)$ arithmetic operations, the problem has to be formulated such that the FFT algorithm can be applied. In practice most but not all FFT algorithms demand the number of data points to be a power of two. For the case at hand, this means that $L = 2(2^k - 1)$ for some $k \geqslant 0$. Clearly this requirement puts some severe constraints on the size and shape of the lattices. In addition computation of the sine transform requires an extra $\mathcal{O}(L^d)$ operations to rearrange the data before and after the application of the FFT.

To make a comparison of RS- and FFT-type algorithms more transparant it has been decided to study also systems with periodic-boundary conditions. Although in chapters 2 and 3 emphasis has been on systems with free-boundary conditions, setting up similar algorithms for systems with periodic boundaries requires only minor changes. As far as the theoretical concepts are concerned no new problem enter. See for example the fourth-order scheme introduced at the end of chapter 2 (eqs. (2.42)–(2.44)). Programming these algorithms is only a matter of making tiny changes to the codes presented in chapter 3. One has to build in that the indices of a lattice point are taken modulo L, a trivial change indeed. Although the use of periodic-boundary conditions has no impact on the efficiency of RS schemes, FFT-based methods perform much better than in the case of free-boundary conditions.

The advantage of working with periodic-boundary conditions is that the wave function can directly be fed into the FFT routine without rearranging the data. The FFT routine used in this work is CFFT of the CERN library GENLIB. It has the appealing feature that it does not need extra memory to store the transformation "sines" and "cosines".

As within numerical accuracy, the error on the free-particle propagator calculated via FFT's is zero, it does not make much sense to compare the efficiency of this approach with other schemes. This comparison is therefore postponed to section 2 where the full problem, i.e. the case where the potential is non-zero, is addressed. There it is demonstrated that for most practical situations of interest, the fourth-order RS method is the most efficient method.

4.1. Free-particle propagator

To begin with, let us focus on the error analysis of second- and fourth-order schemes (RS_2 and RS_4) for the free-particle propagator. This is the most simple case to analyze since there is only one parameter governing the accuracy, namely τV. In table 4.1, some typical results for the RMS error on the wave function, i.e. the distance in L^d-dimensional space between approximate

Table 4.1
RMS-errors on the wave function of a 2D free-particle system of size 33×33 for several values of the time t, calculated by means of the second-order RS algorithm RS_2. CPU time is expressed in seconds

tV	$\tau V = 0.025$	$\tau V = 0.05$	$\tau V = 0.1$	$\tau V = 0.2$
0	0	0	0	0
1	0.146×10^{-3}	0.583×10^{-3}	0.233×10^{-2}	0.937×10^{-2}
2	0.206×10^{-3}	0.823×10^{-3}	0.329×10^{-2}	0.132×10^{-1}
3	0.258×10^{-3}	0.103×10^{-2}	0.414×10^{-2}	0.166×10^{-1}
4	0.345×10^{-3}	0.138×10^{-2}	0.552×10^{-2}	0.221×10^{-1}
5	0.424×10^{-3}	0.170×10^{-2}	0.680×10^{-2}	0.273×10^{-1}
6	0.505×10^{-3}	0.202×10^{-2}	0.808×10^{-2}	0.324×10^{-1}
7	0.581×10^{-3}	0.232×10^{-2}	0.930×10^{-2}	0.373×10^{-1}
8	0.662×10^{-3}	0.265×10^{-2}	0.106×10^{-1}	0.425×10^{-1}
9	0.745×10^{-3}	0.298×10^{-2}	0.119×10^{-1}	0.479×10^{-1}
10	0.823×10^{-3}	0.329×10^{-2}	0.132×10^{-1}	0.528×10^{-1}
CPU time	114	57.7	29.6	15.4

and exact wave function, have been collected. These data have been calculated by means of the second-order RS method, as explained in section 2.2, for a square lattice of linear size $L_x = L_y = 33$ and for different values of τV with time covering the interval $0 \leqslant t \leqslant 10$. For $\tau V = 0.025$ the maximum number of time steps used is $m = 400$. The initial wave function $|\Phi(t=0)\rangle = \Sigma_{\boldsymbol{n}} \Phi_{\boldsymbol{n}} c_{\boldsymbol{n}}^{+} |0\rangle$ was chosen to be

$$\Phi_{\boldsymbol{n}} \equiv \Phi_{l_x,l_y} = \frac{2}{\sqrt{(L_x+2)(L_y+2)}} \sin\frac{\pi k_x l_x}{L_x+2} \sin\frac{\pi k_y l_y}{L_y+2}, \tag{4.1}$$

with $(k_x, k_y) = (12, 3)$. Recall that (4.1) is an eigenstate of the 2D free-particle propagator (see for instance eq. (2.15)). This property facilitates the calculation of the exact time evolution of $|\Phi(t)\rangle$.

From the data presented in table 4.1 it follows directly that the data for the RMS errors fit excellent to the formula

$$\mathrm{RMS}(RS_2) = r_2(\tau V)^2(tV), \tag{4.2}$$

where for this particular set of data $r_2 \approx 0.132$. In general, the precise value of r_2 will depend on the initial state and system size. Calculations have shown that if the initial state is an eigenstate of H, the RMS error tends to be smaller than when the initial state is for instance of the form $\Phi_{\boldsymbol{n}} = \delta_{\boldsymbol{n},\boldsymbol{n}_0}$ where $\boldsymbol{n}_0$ is an arbitrary lattice site. The dependence of the RMS error on the system size turns out to be weak and not systematic.

The formal similarity of the right-hand side of (4.2) with the upperbound for the RMS error derived in appendix A (see (A.20b)) is striking. The upperbound predicts correctly the most prominent dependencies, those on τ and t. To be sure that the scaling law (4.2) does not hold accidentally it has been verified over and over again, with different sets of parameters. It has

been found that (4.2) provides reliable estimates for the global (and therefore also local) error without doing extensive computations. Note that the system size is not entering the expressions for the upperbounds, supporting the observation that the RMS errors do not increase with the system size.

From the upperbound on the RMS error of RS_4 (see(A.25)), we may now speculate that a similar scaling behavior will hold in this case too. That this is indeed the case is demonstrated in table 4.2 where we have repeated exactly the same calculations as those presented in table 4.1 but this time with RS_4. From these data it follows that

$$\mathrm{RMS}(RS_4) = r_4(\tau V)^4(tV), \tag{4.3}$$

with $r_4 \approx 0.082$. Just as in the previous example, the value of r_4 will depend on the initial state and system size. The scaling behavior itself is however universal, as long as the RMS error does not approach its asymptotic value 2 (see appendix A for more details on this). Also in this case (4.3) has been found to possess predictive power.

On the bottom line of tables 4.1, 2 the approximate CPU time, as measured on a VAX 8200, is given. It is seen that for the same number of time steps the CPU time used by RS_4 is less than a factor of three larger than for RS_2. To compare the efficiency of RS_2 and RS_4, assume that we want to solve the TDSE for a given interval $[0, t]$ and with a global RMS error on the wave function less than some fixed value ϵ. Calling the number of time steps needed by the RS_2 and RS_4 method to cover the interval $[0, t]$ m_2 and m_4 respectively, it follows from (4.2) and (4.3) that

$$m_2/m_4 = r_2^{1/2} r_4^{-1/4} (tV/\epsilon)^{1/4}. \tag{4.4}$$

As the total number of arithmetic operations, denoted by n_2 and n_4, increases linearly with the number of time steps (with constants of proportionality a_2 and a_4 respectively) we obtain

$$n_2/n_4 = a_2 a_4^{-1} r_2^{1/2} r_4^{-1/4} (tV/\epsilon)^{1/4}. \tag{4.5}$$

Table 4.2
Same as in table 4.1 but instead of the RS_2 the fourth-order RS algorithm RS_4 was used to solve the TDSE

tV	$\tau V = 0.025$	$\tau V = 0.05$	$\tau V = 0.1$	$\tau V = 0.2$
0	0	0	0	0
1	0.346×10^{-7}	0.553×10^{-6}	0.885×10^{-5}	0.141×10^{-3}
2	0.662×10^{-7}	0.106×10^{-5}	0.169×10^{-4}	0.269×10^{-3}
3	0.984×10^{-7}	0.157×10^{-5}	0.251×10^{-4}	0.399×10^{-3}
4	0.131×10^{-6}	0.209×10^{-5}	0.334×10^{-4}	0.530×10^{-3}
5	0.162×10^{-6}	0.259×10^{-5}	0.414×10^{-4}	0.657×10^{-3}
6	0.194×10^{-6}	0.310×10^{-5}	0.495×10^{-4}	0.787×10^{-3}
7	0.226×10^{-6}	0.361×10^{-5}	0.576×10^{-4}	0.915×10^{-3}
8	0.257×10^{-6}	0.412×10^{-5}	0.657×10^{-4}	0.104×10^{-2}
9	0.289×10^{-6}	0.463×10^{-5}	0.739×10^{-4}	0.117×10^{-2}
10	0.321×10^{-6}	0.513×10^{-5}	0.820×10^{-4}	0.130×10^{-2}
CPU time	271	139	70.7	35.5

Comparison of the scalar code for the second- and fourth-order algorithm for the free particle propagator (see section 3.1) learns that $a_4 \approx 2a_2$ if we neglect small corrections due to boundary effects. To get some feeling for the order of magnitude of the constants entering in (4.5) the values of r_2 and r_4 may be taken from the data of tables 4.1, 2 and this then results in

$$n_2/n_4 \approx 0.34(tV/\epsilon)^{1/4}. \tag{4.6}$$

As the CPU time of a particular RS scheme is proportional to the number of arithmetic operations, (4.6) demonstrates that for reasonable values of tV/ϵ (i.e. $TV/\epsilon \gg 100$) RS_4 is more efficient than RS_2. In general, the prefactor 0.34 appearing in (4.6) will depend on the initial state and on the computer on which the code is executed. Nevertheless the main conclusion will remain the same. For the case presented in tables 4.1, 2, RS_4 is roughly five times more efficient than RS_2, which is not very much. However, if we would have wanted the solution of the TDSE for much longer times, say $t = 10000$, with the same RMS error ϵ employing RS_4 instead of RS_2 would save us a factor of 50 in CPU time.

A complete evaluation of the efficiency should also include the effect of the cost of memory usage. Unfortunately this cost depends very strongly on the particular computer system used and this makes it difficult to include this aspect in the discussion. To simplify the reasoning presented above it has therefore been assumed that sufficient high-speed memory is accessible at no cost.

To compare the RS approach with the conventional Cranck–Nicholson (CN) algorithm the calculations that led to the results of tables 4.1, 2 have to be repeated using $(1 - i\tau H/2)(1 + i\tau H/2)^{-1}$ as the approximate time-step operator. For the special case treated in this section, this calculation can be done analytically. Indeed, the initial state was chosen to be an eigenstate of H, with energy

$$E_k = 2V\left(\cos\frac{\pi k_x}{L_x+2} + \cos\frac{\pi k_y}{L_y+2}\right), \tag{4.7}$$

with $k = (12, 3)$ in the sample discussed above. It then follows that the RMS error on the wave function after m time steps is given by

$$\mathrm{RMS}(CN) = |e^{i(\tau m E_k - 2m\phi_k)} - 1|, \tag{4.8}$$

where $\phi_k = \arctan(\tau E_k/2)$. For the same model parameters as the ones used in tables 4.1, 2, (4.8) produces the data presented in table 4.3. As before, the upperbound for the RMS error of CN (see (A.9)) predicts the correct τ and t dependence. The data of table 4.3 fits well to the scaling law

$$\mathrm{RMS}(CN) = r_{CN}(\tau V)^2(tV), \tag{4.9}$$

where for this particular set of data $r_{CN} \approx 1.85$. Assuming that in actual applications the number of operations of the CN algorithm is the same as for RS_2 (an assumption that strongly favors the CN method), it is seen that in this 2D case, RS_2 is approximately 14 times more efficient than the CN method.

Table 4.3
Same as in tables 4.1, 2 but instead of using a SPF algorithm, the Cranck–Nicholson method has been used to compute the RMS errors on the wave function

tV	$\tau V = 0.025$	$\tau V = 0.05$	$\tau V = 0.1$	$\tau V = 0.2$
0	0	0	0	0
1	0.116×10^{-2}	0.463×10^{-2}	0.184×10^{-1}	0.710×10^{-1}
2	0.232×10^{-2}	0.927×10^{-2}	0.367×10^{-1}	0.142
3	0.348×10^{-2}	0.139×10^{-1}	0.551×10^{-1}	0.213
4	0.464×10^{-2}	0.185×10^{-1}	0.735×10^{-1}	0.283
5	0.581×10^{-2}	0.232×10^{-1}	0.918×10^{-1}	0.353
6	0.697×10^{-2}	0.278×10^{-1}	0.110	0.423
7	0.813×10^{-2}	0.324×10^{-1}	0.129	0.492
8	0.929×10^{-2}	0.371×10^{-1}	0.147	0.561
9	0.104×10^{-1}	0.417×10^{-1}	0.165	0.628
10	0.116×10^{-1}	0.463×10^{-1}	0.183	0.695

The main conclusion from this comparison is that if one is satisfied with the τ^2 correctness of the CN or RS_2 scheme, there is no reason to prefer the CN algorithm over RS_2, the latter being an explicit, unconditionally stable, more accurate and more efficient algorithm for performing the free-particle propagation. As pointed out above, RS_4 is even more preferable if $tV/\epsilon \gg 100$.

Finally some remarks about the numerical precision are in order. The calculations that led to the results of tables 4.1–3 have been done with 64-bit floating-point arithmetic (D-floating format) on a VAX 8200. In this format the degree of precision in representing a floating-point number is typically 16 digits which is more than sufficient for the present purpose. The computation time can be reduced further by switching to 32-bit floating-point arithmetic but then it becomes impossible to compute the numbers of the second and third column of table 4.2 as can be expected on the basis of the smallness (relative to 1) of these numbers. In many practical cases however it is sufficient to get results with much less accuracy than the one required for the calculation of the RMS errors presented in table 4.2 so that an additional speed-up can be realized by using 32-bit floating-point arithmetic where appropriate. On a VAX 8200 the maximum speed-up that can be achieved by this trick is about a factor of two.

4.2. General case

Performing an error analysis of the various SPF algorithms in the case where also the potential is present ($U \neq 0$) is somewhat more tedious than in the free-particle case. The exact results to compare with have to be computed by diagonalization of the Hamiltonian. This already puts severe restrictions on the size of the systems that can be studied. Fortunately, from the expressions for the upperbounds on the RMS errors, it is to be expected that the size of the system will not have a dramatic effect on the errors and all our calculations indicate very strongly that this is indeed the case. In this section the RMS error on the wave function is computed from the data obtained from SPF approximants and from the results obtained by direct diagonalization of the exact propagator. All calculations related to the exact evaluation of the propagator e^{-iHt} have been performed by EISPACK routines, using 32-bit arithmetic on a

VAX 8200 (approximately 5 digits accuracy) and 64-bit arithmetic on a CYBER 205 (approximately 14 digits). All other calculations have been done using 64-bit arithmetic. An important finding is that the scaling behavior with respect to τ and t discussed in the previous section remains the same. This releaves us from the necessity to present data taken at different values of the time t.

It is obviously impossible to consider all types of potentials. Hence it is necessary to select a particular form for the potential. This section contains the analysis for the case where the potential $W\epsilon_L$ is a random variable, distributed uniformly over the interval $[-W/2,W/2]$. The time scale will be fixed by putting $V=1$. With this choice of potential the model is precisely the model introduced by Anderson to describe localization of electrons (see also chapter 5) [42]. For the present purpose this model is as good or as bad as any other. It only serves to illustrate the typical behavior of the RMS errors of the various integration schemes as a function of the potential strength W. Remember that since this system is a genuine lattice model, questions related to taking the continuum limit are completely irrelevant.

To compare with the CN method the knowledge of the eigenstates and eigenvalues of H is used to compute the action of $(1-\mathrm{i}\tau H/2)(1+\mathrm{i}\tau H/2)^{-1}$ directly. This makes it impossible to compare on a quantitative basis the efficiency of CN with any of the other methods. For 1D problems it is relatively straightforward to estimate the amount of arithmetic operations per time step for each of the algorithms discussed in this paper and this will give us a good indication of the amount of CPU time necessary to perform a time step. For 2- and 3-dimensional systems the situation is much more complicated to analyse since the performance of the CN method will depend strongly on the implementation. Fortunately there is some consensus about the efficiency of implicit schemes such as CN when applied to multi-dimensional problems, namely that they are very costly. Therefore we take the point of view that if the actual RMS error of CN is larger than that of another scheme, the latter will also be (much) more efficient.

Having discussed some general aspects of the performance analysis to be made, let us now consider results for the 1D case. In table 4.4 some typical results for the RMS errors have been collected. As explained above, a fair comparison of RS schemes and FFT-based algorithms demands that periodic-boundary conditions are adopted, and this is what has been done. In principle, for $W=0$ the RMS errors of FFT_2 and FFT_4 should be zero. The non-zero value

Table 4.4
RMS errors on the wave function obtained by solving the TDSE by the Cranck–Nicholson method and four different SPF algorithms for the case of a 1D Anderson localization model of length $L=64$, subject to periodic-boundary conditions. The time step used was $\tau=0.1V$ and the number of time steps was taken to be 100. *CN*: Cranck–Nicholson method. FFT_2: Second-order formula (eq. (2.26)), free-particle propagation by Fast Fourier Transforms. RS_2': Real-space second-order formula (eq. (2.9) or eq. (2.28)). FFT_4: Fourth-order formula (eqs. (2.35–38)), free-particle propagation by Fast Fourier Transforms, RS_4': Real-space fourth-order formula (eqs. (2.42–44))

Method	$W=0$	$W=6V$	$W=12V$	$W=24V$
CN	0.37×10^{-1}	0.82×10^{-1}	0.21	–
FFT_2	0.12×10^{-4}	0.18×10^{-1}	0.27×10^{-1}	0.34×10^{-1}
RS_2'	0.84×10^{-2}	0.14×10^{-1}	0.16×10^{-1}	0.20×10^{-1}
FFT_4	0.12×10^{-4}	0.62×10^{-3}	0.27×10^{-2}	0.11×10^{-1}
RS_4'	0.71×10^{-4}	0.27×10^{-3}	0.72×10^{-3}	0.23×10^{-2}

Table 4.5
CPU times in seconds required by the different SPF algorithms to solve the TDSE for a ring of $L = 1024$ sites and 100 time steps as measured on a VAX 8200. The notation used is the same as in table 4.4

Method	FFT_2	RS_2'	FFT_4	RS_4'
CPU time	87	24	112	58

0.12×10^{-4} merely reflects the five-digit precision used in this calculation. The dash in table entry (CN, $W = 24V$) indicates that for a time step $\tau = 0.1$ and 100 time steps the RMS error of CN has already reached its maximum and has started to oscillate as a function of the number of steps. If this happens the outcome of this CN calculation should be considered as incorrect. The primes on the RS symbols indicate that those RS algorithms have been used that, for 1D problems, are the most efficient (see the equations referred to in the table caption).

From the results of table 4.4 the following conclusions can be drawn. Compared to the SPF algorithms, CN performs rather bad. For $W \neq 0$ the RS algorithms compare favorably to the corresponding FFT algorithms, as far as the RMS errors are concerned. Except for $W = 0$, the fourth-order 1D RS method, based on eqs. (2.42–44), yields the best results.

To determine which of the four SPF based methods is the most efficient let us first estimate the number of arithmetic operations per time step. From the expressions referred to in the caption of table 4.4 one readily finds the approximate number of operations to be proportional to $2L(a + b \log L)$ for FFT_2, $2L(2a + c + 2b \log L)$ for FFT_4, $3a'L$ for RS_2', and $8a'L$ for RS_4' where a, b, c and a' are constants of order unity. Therefore it can be expected that RS_2 (RS_4) will become faster than FFT_2 (FFT_4) if L is made large enough. In table 4.5 some typical results for the CPU time used by the four different methods are given. It is seen that for a ring of length $L = 1024$ the $L \log L$ dependence of the FFT dominates over the linear dependency. Note that RS_4' even takes less time than FFT_2. Clearly the conclusion must be that for any reasonable interval $[0, t]$, RS_4' is the most efficient algorithm.

As explained in chapter 2, for two- or three-dimensional problems the fourth-order RS approach used for 1D systems is not very well suited because it lacks modularity with respect to the dimensionality of the lattice. A more systematic approach is to decompose the Hamiltonian into kinetic and potential energy first and then decompose the kinetic energy further into blocks

Table 4.6
RMS errors on the wave function obtained by solving the 2D TDSE by the Cranck–Nicholson method and two RS algorithms for a square lattice of $L^2 = 64$ sites, subject to periodic-boundary conditions. The time step used was $\tau = 0.1V$ and the number of time steps was taken to be 100. The notation is the same as in table 4.4

Method	$W = 2V$	$W = 4V$	$W = 6V$	$W = 8V$
CN	0.17	0.19	0.22	0.24
FFT_2	0.50×10^{-2}	0.18×10^{-1}	0.29×10^{-1}	0.41×10^{-1}
RS_2'	0.14×10^{-1}	0.25×10^{-1}	0.35×10^{-1}	0.45×10^{-1}
FFT_4	0.12×10^{-3}	0.45×10^{-3}	0.10×10^{-2}	0.19×10^{-2}
RS_4'	0.19×10^{-3}	0.50×10^{-3}	0.11×10^{-2}	0.19×10^{-2}

Table 4.7
CPU time in seconds required by the different SPF algorithms to solve the TDSE for 100 time steps for a 2D system with periodic-boundary conditions as measured on a CYBER 205 running the SCALAR code. The notation is the same as in table 4.4

$L \times L$	RS_2'	RS_4'	FFT_2	FFT_4
32×32	0.9	3.6	5.4	7.2
64×64	3.5	14.6	25.8	33.0
128×128	16.4	61.9	131.8	160.6

of commuting two-site propagators. Real-space methods based on this philosophy are denoted by RS_2 or RS_4, i.e. without prime.

A typical collection of RMS errors for the 2D Anderson model is given in table 4.6. The gross features are the same as for the 1D case with one exception: For large W/V the RMS errors of RS_2 (RS_4) approach those of FFT_2 (FFT_4). Of course for the RS scheme used here this has to be so because if W/V becomes large, the RMS error will be determined by the error made by breaking up H into kinetic (T) and potential (U) energy.

Obviously, decomposition of the kinetic energy implies that RS_4 will take more CPU time than RS_4' simply because more passes over the lattice are necessary. An idea of the CPU time required for solving a 2D problem by each of the four different algorithms can be obtained by looking at the data presented in table 4.7. As might be expected on the basis of the experience with the 1D case, the conclusion remains the same. Any of the real-space algorithms runs faster than an algorithm using the FFT.

Table 4.8 contains a typical selection of RMS errors for 1D, 2D, and 3D systems with free-boundary conditions, computed by means of the same RS scheme and the same program. The notation used here is the same as before. Remark that CN is doing rather poorly compared

Table 4.8
RMS errors on the wave function obtained by solving the TDSE by the Cranck–Nicholson method and two different RS algorithms for the case of the one-, two- and three-dimensional Anderson localization model, subject to free-boundary conditions. The time step used was $\tau = 0.1V$ and the number of time steps was taken to be 100. The linear size of the system $L = 125$, $L = 11$ and $L = 5$ for the 1D, 2D and 3D model respectively. RS_2: Real-space second-order formula (eq. (2.26) and eq. (2.27)). RS_4: Real-space fourth-order formula (eq. (2.35) and eqs. (2.40–44))

Lattice	Method	$W = 0$	$W = 6V$	$W = 12V$	$W = 24V$
1D	CN	0.19×10^{-1}	0.90×10^{-1}	0.23	–
	RS_2	0.85×10^{-2}	0.22×10^{-1}	0.46×10^{-1}	0.10
	RS_4	0.49×10^{-4}	0.67×10^{-3}	0.41×10^{-2}	0.19×10^{-1}
2D	CN	0.69×10^{-1}	0.25	–	–
	RS_2	0.98×10^{-2}	0.38×10^{-1}	0.78×10^{-1}	0.20
	RS_4	0.60×10^{-4}	0.14×10^{-2}	0.85×10^{-2}	0.38×10^{-1}
3D	CN	0.14	–	–	–
	RS_2	0.80×10^{-2}	0.37×10^{-1}	0.68×10^{-1}	0.93×10^{-1}
	RS_4	0.51×10^{-4}	0.14×10^{-2}	0.62×10^{-2}	0.23×10^{-1}

Table 4.9
Typical CPU-time in seconds used by RS_4 to solve the 2D TDSE for 100 time steps, as measured on a single-pipeline CYBER 205

	$N = L^2 = 31 \times 31$	$N = L^2 = 63 \times 63$	$N = L^2 = 127 \times 127$
SCALAR	3.9	16.3	66.5
VECTOR	0.6	2.2	8.5

to RS. The reader might convince himself that these data show the same trends as previously discussed.

All the results presented in this chapter have been obtained by programs running in scalar mode. We now address the question to what extent the efficiency of the algorithms can be increased by using vector processors. First note that the knowledge gained above can be put at good use to limit the amount of work necessary to perform this comparison. Indeed, it will not be worthwhile to test codes that use a (vectorized) FFT provided RS can be vectorized to a high degree. From chapter 3 we already know that this is the case so it is justified to leave out of the discussion FFT-based algorithms.

From the structure of the vectorized RS code, it is clear that in order to study the gain in performance due to vectorization, it is entirely irrelevant whether the TDSE is solved for a 1D, 2D or 3D system since all vector operators have a length approximately equal to the number of lattice points. Table 4.9 gives some idea of the speed-up due to vectorization as measured on a CYBER 205. As far as the program is concerned, the scalar and vectorized version differ in the computational kernel as already discussed in chapter 3, and in the code necessary to set up the BIT vectors. The latter has no effect on the computation time whatsoever. The conclusion is that the speed-up due to the vectorization is about a factor of 8.

Table 4.10 gives the amount of high-speed memory used by the scalar and vectorized version of the same program as a function of the number of lattice points, i.e. the number of words needed to store the different elements of all two-site propagators appearing in the full RS_4 scheme. The reduction in going from scalar to vector code stems from the replacement of INTEGER type control vectors used in the scalar version, by BIT type control vectors used by the vector code. For the scalar code the same reduction could have been achieved by packing the INTEGER control variables into words. In our applications (see chapter 5) it has been observed that storing all numbers, representing two-site propagators originating from $e^{i\tau^3 C(T,U)}$, in half

Table 4.10
Number of CYBER 205 words used by RS_4 as a function of the total number of lattice sites N. Included is all scratch storage required by the vector operations. The prime indicates that part of the look-up tables are of the HALF PRECISION type (see text)

	1D	2D	3D
SCALAR	$17N$	$35N$	$61N$
VECTOR	$(13+\frac{3}{32})N$	$(21+\frac{1}{4})N$	$(33+\frac{15}{32})N$
VECTOR′	$(11+\frac{3}{32})N$	$(15+\frac{1}{4})N$	$(21+\frac{15}{32})N$

precision (32-bit) floating point format does not affect the results. Hence further reduction of memory requirements was possible, as indicated by the bottom line of table 4.10.

5. Applications

This chapter describes several applications in which SPF algorithms are used to solve parabolic difference equations. Instead of presenting results for simple model system such as the harmonic oscillator, the TDSE of which is readily solved by any of the standard algorithms, only some of our applications will be discussed for which the use of SPF algorithms has been instrumental. Common to all of them is that they involve the study of the time evolution of a wave packet. In general the procedure goes as follows. First the initial wave packet is constructed according to certain specifications, depending on the particular case at hand. Then the fourth-order SPF algorithm RS_4 is invoked to solve the TDSE. During the integration process the wave packet is analysed and the physical properties are extracted at regular time intervals.

The first two applications deal with the problem of localization of waves. The concept of localization, originally introduced by Anderson [42] in this study of the motion of an electron moving in a random potential, has recently found to be applicable to a wide range of physical phenomena including particles moving in almost periodic potentials [43–48], phonons [49–53], enhanced back scattering of light by random media [54–56], etc.. The physical process responsible for the occurrence of localization effects is interference of waves, a process which is not inherently quantum mechanical but as illustrated by recent light scattering experiments [54–56], is present in classical wave mechanics as well [57–59]. If the medium through which the wave travels consists of "strongly" scattering objects that are not "too" far apart, interference may prevent the wave packet to spread out indefinitely, even for infinite times. By definition a wave is said to be localized if its amplitude decays exponentially with distance.

From the definition of localization it is apparent that it may be difficult to study this kind of phenomenon by directly solving the wave equation, for what is required is information about the behavior of the wave packet at "large" distances and "long" times. The most direct manner to extract from the wave packet the relevant information is to assume a parametrized form for the wave packet and compute the parameters. Usually these parameters can be related to moments of the amplitude of the wave. Examination of the behavior of these parameters as a function of time will reveal whether the asymptotic form is appropriate or not. A list of the most commonly encountered asymptotic forms in given in table 5.1.

In section 1 the concepts of localized and extended states are illustrated by solving the TDSE for the Aubry model, a single-orbital tight-binding model of an electron in an incommensurate potential [60]. An interesting feature of this model is that even in one dimension (1D) it exhibits a metal-insulator (extended versus localized) transition, quite unlike 1D random systems where almost all states are localized. The vast amount of knowledge about the Aubry model make it an excellent, non-trivial example for investigating localization phenomena by direct computer simulation [61]. As far as I know no such simulations have already been reported. It should be mentioned however that for 1D systems there are more efficient methods of analysis [62–65] than the one based on numerically solving the TDSE by direct integration. The main purpose of section 1 is to illustrate the TDSE approach by applying it to a model system with peculiar

properties and to demonstrate how these properties reveal themselves in the time-dependent wave function.

In section 5.2, exactly the same computer code and method of analysis as those used in section 1 are used to study the two-dimensional (2D) Anderson model of localization. The aim is to find out, by direct numerical simulation, whether or not there is a critical value of the strength of the random potential below which there exist extended states. Most numerical simulations suggest that there is indeed such a critical value, in contradiction with current theories. It therefore remains to be settled why most numerical experiments, indicate that there is an abrupt transition from a localized to an extended regime as the degree of disorder diminishes. It will be demonstrated that by solving the TDSE for sufficiently large systems and long times and answer to this question can be given.

Whereas the systems treated in sections 5.1 and 5.2 are defined by lattice models, section 5.3 is devoted to the application of SPF algorithms to continuum problems. In particular it is shown how RS_4 can be extended to deal with higher-order approximations to the Laplacian. The generalized SPF algorithm is then used to compute High-Resolution Electron-Microscopy (HREM) images and comparison is made with other numerical techniques.

5.1. Almost-periodic potential

The Aubry model is described by the Hamiltonian

$$H = V\sum_l (c_l^+ c_{l+1} + c_{l+1}^+ c_l) + W\sum_l \cos(Ql) n_l, \tag{5.1}$$

where V is the hopping energy, W is the potential strength, Q is the wave vector of the almost periodic potential (almost periodic with respect to the underlying lattice), and the lattice constant is taken to be one. Remark that the Aubry model is a genuine lattice model so that we do not have to worry about taking the continuum limit. Setting $\epsilon_l = \cos(Ql)$ it is clear that (5.1) is precisely of the form used in the construction of SPF algorithms. Hence RS_4 as discussed in chapter 3 can be used without further modification.

To set up the initial state, two different procedures have been adopted. In the first one, a sub-system of typically 125 sites centered around the middle of the chain is solved by numerical diagonalization. Then the initial state $\psi(t=0)$ is chosen from the 125 eigenstates in such a way

Table 5.1
Asymptotic behavior of the second moment of $|\psi(\boldsymbol{r}, t)|^2$, the probability of finding the particle at point $\boldsymbol{r}$ at time t, for the different cases of interest. The dimensionality of the system, the linear size, the diffusion constant, and the localization length are denoted by d, L, D, and ξ respectively

	Wave function	$\langle \boldsymbol{r}^2(t)\rangle - \langle \boldsymbol{r}(t)\rangle^2$
Uniform	$\lvert\psi(\boldsymbol{r}, t)\rvert^2 \propto L^{-d}$	$d(L^2-1)/12$
Extended	$\psi(\boldsymbol{r}, t) \propto \mathrm{e}^{-\mathrm{i}\boldsymbol{k}\cdot\boldsymbol{r}}$	t^2
Diffusion	$\partial\lvert\psi(\boldsymbol{r}, t)\rvert^2/\partial t = D\Delta\lvert\psi(\boldsymbol{r}, t)\rvert^2$	$2dDt$
Localization	$\lvert\psi(\boldsymbol{r}, t)\rvert^2 \propto \mathrm{e}^{-r/\xi}$	$d(d+1)\xi^2$

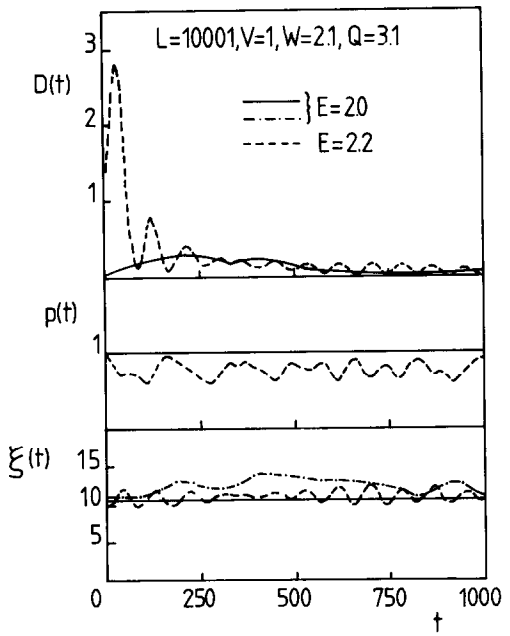

Fig. 5.1. The functions $\xi(t) = \sqrt{(\langle r^2(t)\rangle - \langle r(t)\rangle^2)/2}$, $D(t) = \xi^2(t)/t$ ant the return probability $p(t)$ (see text) for a particle moving in an incommensurate potential.

that the difference between its energy and the given energy E is minimal. The value of $\sigma(t) \equiv \langle\psi(t)\,|\,H^2\,|\,\psi(t)\rangle - \langle\psi(t)\,|\,H\,|\,\psi(t)\rangle^2$ for $t = 0$ will indicate to what extent $\psi(t=0)$ is an exact eigenstate of the full system. In the second procedure the particle is put at the site the potential energy of which is the closest to the given energy E. In that case $\sigma(t) = 2V$ takes its maximum value.

During the solution of the TDSE we monitor the norm of the wave function $\|\psi(t)\| = \langle\psi(t)\,|\,\psi(t)\rangle^{1/2}$ which has to be one to numerical precision, the energy $E(t) = \langle\psi(t)\,|\,H\,|\,\psi(t)\rangle$ and its variance $\sigma(t)$ which both should be constant (a first check on the accuracy of the SPF since an SPF does not conserve energy in a strict sense), a combination of the first and second moment as indicated in table 5.1, and the return probability [42] $p(t) = |\langle l_0\,|\,\mathrm{e}^{-Ht}\,|\,l_0\rangle|^2$ in case the particle starts at site l_0. If the initial state was prepared by diagonalization of a subsystem Λ' the return probability is defined as $p(t) = \sum_{l\in\Lambda'} |\langle l\,|\,\mathrm{e}^{-\mathrm{i}Ht}\,|\,l\rangle|^2$.

Some representative results are shown in figs. 5.1–3. In all cases $V = 1$ and $Q = 3.1$ and the time step τ was chosen such that at least three-digit accuracy was obtained. The initial wave packet was prepared by diagonalization of a subsystem except when explicitly stated otherwise. In practice quantities such as ξ and D as defined in table 5.1 will depend on time and approach their asymptotic constant value for sufficiently long time only if the ansatz is correct.

In fig. 5.1 the three relevant quantities are shown for the case where $W = 2.1$. It is seen that the calculation for $E = 2.2$ strongly indicates that there is localization. The function $\xi(t)$ oscillates around a localization length of $\xi \approx 11$. The return probability is substantially different from zero, and there is no reminiscence of diffusive or extended motion. But what happens at $E = 2.0$? If the calculation for $E = 2.2$ already suggests a localized state, the data for $\xi(t)$

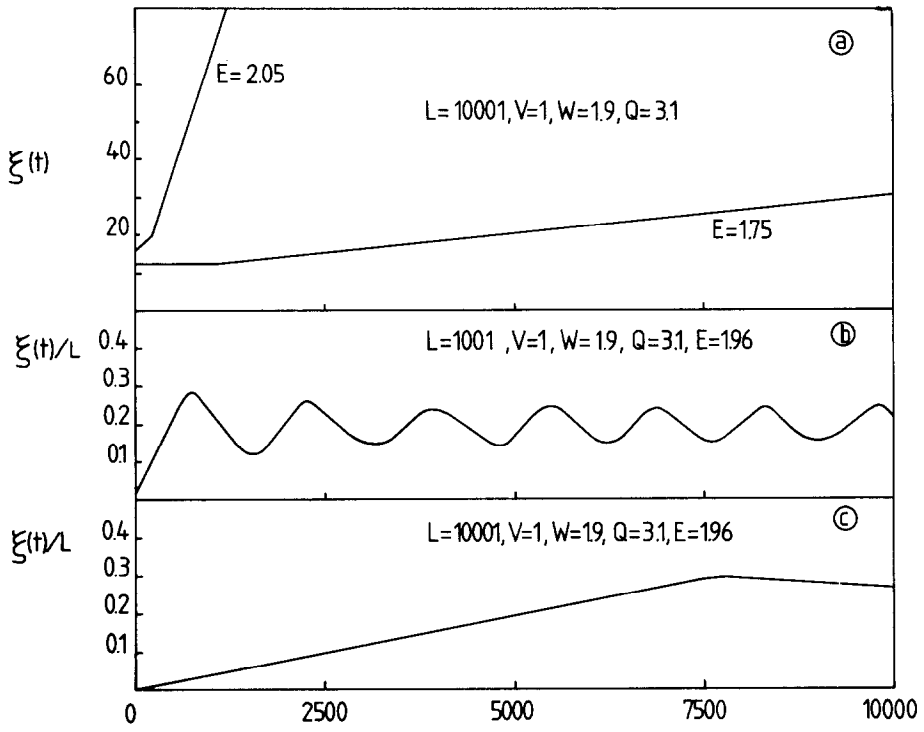

Fig. 5.2. Results for $\xi(t) = \sqrt{(\langle r^2(t)\rangle - \langle r(t)\rangle^2)/2}$ for a particle moving in an almost periodic potential.

represented by the solid line should also be interpreted as such. As André and Aubry [46] showed that (5.1) is self-dual if $W = 2V$ and that all states are exponentially localized if $W > 2V$, the data of fig. 5.1 would clearly be compatible with theory. However some more care is required when analyzing results obtained from the solution of the TDSE. Closer inspection shows that almost to machine precision, the initial state is also an exact eigenstate of the full system and remains unchanged during the integration process. Therefore it is no surprise that $p(t)$ and $\xi(t)$ show little time dependence. In other words not much can be learned from these data. Fortunately this problem is easily circumvented by changing the initial state. Also shown in fig. 5.1 are data of $\xi(t)$ for $E = 2.0$ (dashed-dotted line) whereby the particle starts at a site at which the potential energy matches the specified energy. The presence of time-dependent variations is now apparent. The asymptotic behavior however does not depend on the choice of the initial state. The simulation results are in concert with theory.

Fig. 5.2 shows $\xi(t)$ for a potential strength of $W = 1.9$. From theory [46] one expects all states to be extended. First note the difference in time scale with respect to fig. 5.1. For $E = 2.05$ (see fig. 5.2a), $\xi(t)$ grows linearly with time, the behavior expected for an extended state. For $E = 1.75$ we face the same problem as in the case $E = 2.0$, $W = 2.1$ when the interval of integration is confined to [0, 1000]. However for $1000 < t < 3000$, $\xi(t)$ starts to increase slowly and once $t > 4000$, $\xi^2(t) \propto \langle r^2(t)\rangle - \langle r(t)\rangle^2 \propto t^2$ such that the particle moves like a free one. In other words for $E = 1.75$ the wave function is of the extended type, in agreement with theory.

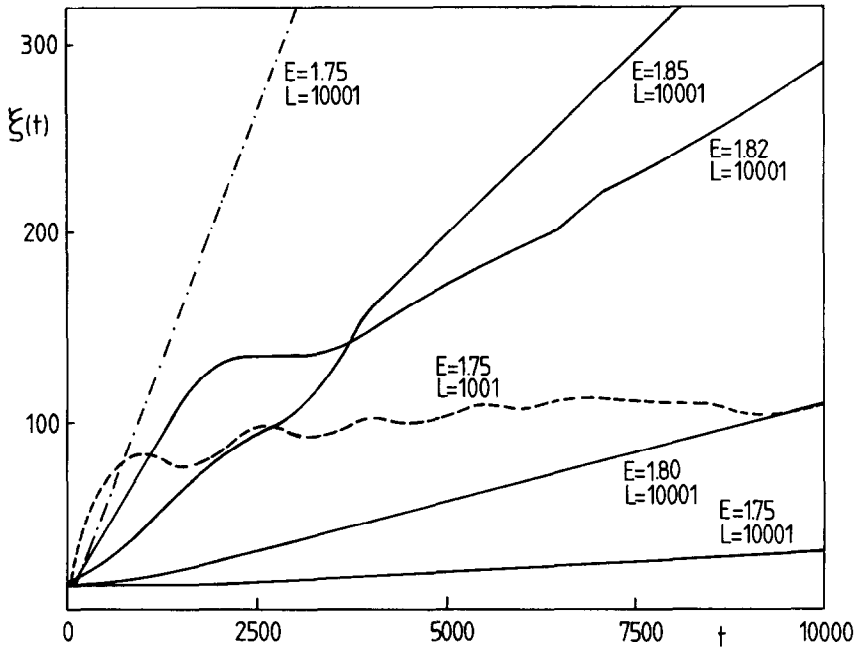

Fig. 5.3. Same as fig. 5.2 except that the data represented by the dashed and dashed-dotted line have been obtained from a calculation whereby the initial state was not prepared by diagonalization of a sub-system.

After waiting for a sufficiently long time, the particle escapes from the 125-site sub-system and behaves as if it were free.

From these calculations it is already clear that one must be cautious when interpreting the data and this is once more demonstrated by figs. 5.2b, c where it shown how the finiteness of the lattice can lead to artifacts. Extended-state behavior is expected and from fig. 5.2b it follows that this is indeed the case if $t < 750$. For $t > 750$ large oscillations in $\xi(t)/L$ are observed. Repeating the same calculation for a lattice that is ten times larger (see fig. 5.2c) reveals that for $t < 7500$, $\xi(t)/L$ is almost exactly the same as in fig. 5.2b provided the time scale of the latter is expanded by a factor 10. There is no doubt that this behavior is only due to the finite size of the lattice. The oscillations result from interference with waves that have been reflected as the boundary. The smaller the system, the larger these interference effects and the more pronounced the oscillations will be. Note that the origin of these oscillations is different than in the case $W = 2.1$, $E = 2.0$ (see fig. 5.1). There the average value of $\xi(t) \approx 11 \ll L = 1001$ whereas in fig. 5.2b, $100 < \xi(t) < 300$ for $t > 500$ and $L = 1001$ so that $\xi(t)$ is of the order of L.

Theoretical studies [47,48] have shown that the spectrum of Hamiltonians with almost periodic potentials can be rather peculiar and complicated and may depend very much on the specific form of the potential. In particular, for the Aubry model it has been established that for $W < 2V$ there may appear very narrow bands of extended states and small band gaps [66]. In fig. 5.3 results for $\xi(t)$ are presented for the ase $W = 1.9$ and several values of $E < W$. Clearly for $E = 1.80$, 1.82, and 1.85 there is no doubt that the particle is in an extended state since $\xi(t) \propto t^2$. For $E = 1.75$ two additional calculations are shown. The dashed curve is obtained by letting the

particle start at the site where the potential $W\cos(Ql) \approx 1.75$ but the motion is restricted to an open ended chain of $L = 1001$ sites. It is seen that $\xi(t)$ has a tendency to saturate at a value of about 100 lattice sites. Repeating this calculation on a $L = 1001$ site lattice (with the same initial condition) yields the dashed-dotted line, suggesting an extended state. The difference between the slope of this straight line and the slope of the solid ($E = 1.75$) line for $t > 3000$ is striking. Close examination learns that both states have a slightly different energy (the fourth digit differs) but more importantly in terms of eigenstates of the Hamiltonian, the initial wave packet for the dashed-dotted case is completely different from the solid-line case. Therefore it should be no surprise that the slope of $\xi(t)$ is not the same.

We now summarize what has been learned from this example. Interpreting the data in terms of extended or localized states is not difficult, provided sufficient checks have been made to rule out boundary effects. Also the wave packet should actually exhibit some non-trivial time dependence because otherwise one may draw a completely wrong conclusion. If the data are analyzed properly the conclusions are in full agreement with established theoretical results.

5.2. *Anderson localization*

The Anderson model of localization [36] is governed by the Hamiltonian

$$H = V \sum_{n\in\Lambda} \sum_{e} (c_n^+ c_{n+e} + c_{n+e}^+ c_n) + W \sum_{n\in\Lambda} \epsilon_n n_n, \tag{5.2}$$

and describes a particle moving on a d-dimensional hyper-cubic lattice Λ and feeling a random potential $W\epsilon_n$. The distribution of the random variable ϵ_n is uniform over the interval $[-1/2, 1/2]$, W sets the strength of the random potential, and V is the hopping energy which in our calculations is taken to be one.

Model (5.2) is one of the simplest models that describes most of the essential features of electrons in disordered systems. It has been the subject of extensive theoretical studies and observable effects due to localization have been predicted and confronted with experimental data [67–72].

From the point of view of computer simulation the key problem is to reconcile results obtained from simulation of two-dimensional (2D) systems with current theories. In order to avoid the repeated use of the term 2D, it will be assumed for the remainder of this section that the discussion is about the 2D system only, unless stated otherwise. Theoretical analysis based on mode-coupling approximations [73,75], diagrammatic expansions [76,77], and field-theoretical methods [78,79] all show that there is always localization. The predictions of these theories are in agreement with a theory based on scaling arguments [80,81]. However as the strength of the random potential diminishes, the localization length becomes so astronomically large that for all practical (experimental or numerical) purposes the system is metallic.

On the other hand numerical simulations not analysed with the help of scaling theory, suggest that there is indeed a critical value, the estimates of the critical disorder being in the range $6 \leqslant W/V \leqslant 7$ at the band center [16–18,82–87]. One finds that there is an abrupt transition from a localized to an extended regime as the degree of disorder diminishes. More recent, refined calculations analyzed by means of scaling hypotheses [88,89] yield results which are compatible

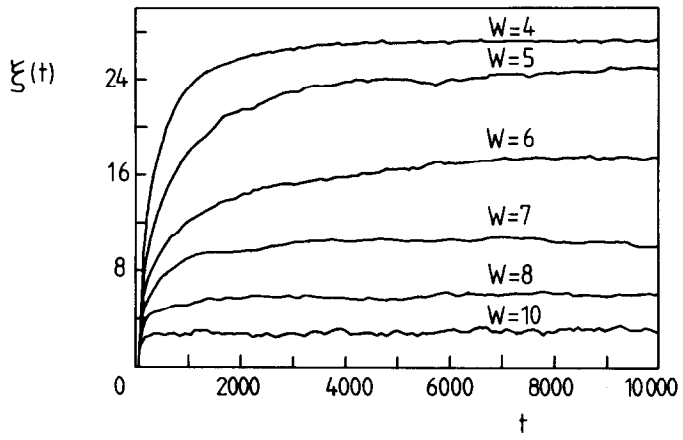

Fig. 5.4. The function $\xi(t) = \sqrt{(\langle r^2(t)\rangle - \langle r(t)\rangle^2)/2}$ for the 2D Anderson model of localization as a function of time t and for several values of the disorder W, obtained from the solution of the TDSE for 179×179 lattices. The energy of the wave packets $E \approx 0$.

with the scaling theory and can be understood in terms of a bound state in a potential well [90].

Following the motion of the particle as it moves through the random medium is the most direct numerical method for probing localization effects, as is most evident from Anderson's original formulation [42] of localization. It is therefore remarkable that simulations based on solving the TDSE [16–18,87], do not agree with other numerical and theoretical work as far as the absence of an Anderson transition is concerned. As SPF algorithms are more efficient that the methods used in previous work, it is possible to examine whether this disagreement can be resolved by simulating larger systems for longer times.

The quantities computed from the time-dependent wave function are the same as in the case of the almost-periodic potential model treated in the previous section. For each value of W/V, the time step τ was chosen such that there is no noticeable (within three digits) effect on the physical properties of interest. For the largest systems studied (209×209) some of the propagator look-up tables have been stored in 32-bit floating point arrays, as discussed in chapter 4. It has been checked that this did not change the results (within three-digit accuracy). For all values of W ($V = 1$) studied taking a time step τ within the range [0.05, 0.2] was found to be adequate. The largest simulations (a square of $L^2 = 43681$ sites, 10^5 time steps) took approximately 6 hours of CPU time on the one-pipeline CYBER 205.

A collection of results for $\xi(t)$ for several values of the disorder W($V = 1$ in all our calculations and time t is measured in units of V) is shown in fig. 5.4. Each curve is obtained by averaging two to five statistically independent TDSE solutions of square lattices of 179×179 sites. In all our calculations the energy of the wave packet $E \approx 0$, i.e. we concentrate on those states for which the localization length for a fixed value of W is the largest. From computational point of view this clearly is the most difficult situation to study because if the calculation shows that the $E \approx 0$ states are localized all the other states are also localized. For $W \geqslant 7$ there is little doubt that there is localization. Over a large time interval $\xi(t)$ is effectively constant. It should be mentioned that the most advanced calculations of this type have covered a time interval which is at least an order of magnitude smaller than in our simulations, thereby employing an

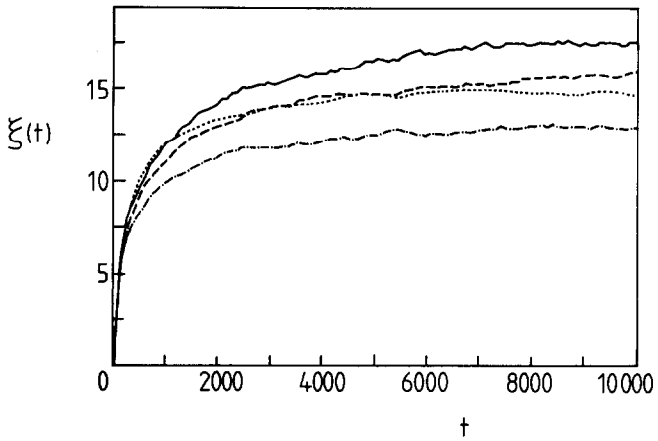

Fig. 5.5. The function $\xi(t)$ for the 2D Anderson model of localization as a function of time t for $W = 6$. Solid line: 179×179 lattice, initial state by diagonalization of an 11×11 subsystem. Dashed line: 179×179 lattice, particle at a site with energy $W\epsilon_n \approx 0$. Dotted line: 127×127 lattice, initial state by diagonalization of an 11×11 sub-system. Dashed-dotted line: 127×127 lattice, particle at a site with energy $W\epsilon_n \approx 0$.

algorithm which is numerically unstable [87]. From fig. 5.4 it is clearly seen that for $W \approx 7$ and $t \leqslant 1000$ the asymptotic regime has not yet been reached and therefore earlier estimates [87] for the localization length are to be considered as unreliable. For $W = 6$ one might draw the preliminary conclusion that the asymptotic regime sets in a $t \geqslant 6000$, but a more detailed analysis is required in order to make a definite statement. We will return to this case later on.

For $W = 4$ and $W = 5$ there is also a clear tendency for saturation in $\xi(t)$. If this were indeed the true behavior it would be quite remarkable since comparison with $W = 6$ shows that the asymptotic regime is approached more rapidly for $W = 4$ than for $W = 6$. However, from the first entry of table 5.1 it follows that for $W = 4$, $\xi(t)$ for $t > 5000$ is close to the asymptotic value for a uniform distribution over all sites, meaning that the wave packet is spread out over the whole lattice without being much influenced by the presence of disorder. Clearly, for $W = 4$ and $W = 5$ boundary effects are important. Starting from $t = 0$, the extent of the wave packet grows at fast pace and saturates after a certain period of time (which increases with the lattice size) because of reflection by the lattice boundaries.

The major question is what happens at $W = 6$. If there is localization, as in expected from theory, earlier simulations [16–18,87] may have failed to recognize this because the size of the system was to small and the time interval to short, and the conclusion that there is an abrupt transition at $W \approx 6$ would have been based on the behavior of the wave packet in a non-asymptotic regime. To examine this problem in detail, calculations with different initial conditions and on lattices with different size have been carried out.

In fig. 5.5 results are shown for lattices of size 127×127 and 179×179, for the two different kinds of initial states. Each of the curves is the average of four independent runs. The average spread in energy $\sigma(t) \approx 0.3$ if the wave packet is prepared by diagonalization of a sub-lattice. The difference between the pairs of 127×127 and 179×179 data can be traced back to the choice of the initial state [91]. If at $t = 0$ the particle is put at a particular site, the wave packet, as a linear combination of eigenstates, will be a superposition of all eigenstates, including those with $E \neq 0$.

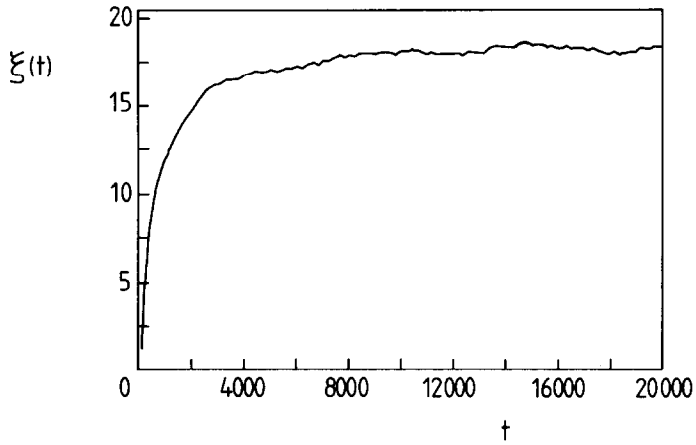

Fig. 5.6. The function $\xi(t)$ for the 2D Anderson model of localization as a function of time t for $W = 6$ and a lattice of size 209×209.

As states with $E \neq 0$ are localized better than the $E = 0$ eigenstate(s), the value of $\xi(t)$ will be lower than in the case where the initial state is an eigenstate of the subsystem. Only if $t \to \infty$ both calculations should lead to the same answer, if there is localization at all.

The difference between the same kind of 127×127 and 179×179 data suggests that boundary effects may still be substantial. Suppose that the value of $\xi(t = 10000) \approx 17$ estimated on the basis of the 179×179 simulation is close to the correct localization length. Then, according to the exponential fall-off of the wave function, the probability for finding the particle at a distance half the lattice size of the smallest system ($L = 127$) would have dropped to a 3% level, which I believe is too large to be neglectable. In an effort to eliminate all boundary effects, simulations have been carried out for systems of size 209×209 for times up to $t = 20000$, the averaged results of three runs being depicted in fig. 5.6. Detailed analysis of the numerical data reveals that there is no evidence that for $W = 6$ there is no localization. For $1000 \leqslant t \leqslant 20000$, $\xi(t)$ fits very well (RMS error less than 1%) to a function of the form $a = bt^{-0.8}$. Extrapolation of the 179×179 yields an estimate for the localization length which is in excellent agreement with the value $\xi = 19.0 \pm 0.3$ derived from the 209×209 simulations.

The values of the localization length as a function of W are given in table 5.2, together with results obtained by other methods. It is clear that there is satisfactory agreement. Pushing the

Table 5.2
Comparison of results for the localization length ($\lambda \equiv 2\xi$) of the 2D Anderson model at $E = 0$ obtained by the TDSE approach with the numerical data of refs. [88] and [90]. The trivial factor 2 between λ and ξ stems from a slight difference in the definition of the localization length

W	λ(present)	λ(ref. [88])	λ(ref. [90])
6	38.0	37.46	41.26
7	21.4	18.53	18.79
8	12.8	11.07	10.91
10	5.6	5.45	5.34

TDSE approach to still lower values of W, one again faces the problem that because of the rapid increase of the localization length, both the lattice and time interval have to be enlarged accordingly.

On the intermediate time scale, i.e. for times larger than the microscopic time scale ($V = 1$) but smaller than the time scale on which interference effects are dominant, $\xi(t)$ fits well (RMS error less than 1%) to the power law $\xi(t) = d + ct^{\gamma}$ with $\gamma = 0.5 \pm 0.1$, in agreement with the mode-coupling theory [73–75]. For instance if $W = 6$ the power law holds for $t \in [10, 50]$. The time interval over which this behavior is observed shrinks as W increases.

To summarize, it has been demonstrated that the SPF allows the solution of the TDSE for sufficiently large lattices and time periods such that the results are in concert with the present knowledge on 2D Anderson localization.

5.3. Continuum problems

To illustrate the application of the SPF approach to models formulated in continuum space I will confine the discussion to linear parabolic partial differential equations of the type

$$\frac{\partial \phi(\boldsymbol{r})}{\partial z} = \mathrm{i}\lambda\left(\frac{\partial^2}{\partial x^2} + \frac{\mathrm{i}k_x}{\lambda k_z}\frac{\partial}{\partial x} + \frac{\partial^2}{\partial y^2} + \frac{\mathrm{i}k_y}{\lambda k_z}\frac{\partial}{\partial y} + V(\boldsymbol{r})\right)\phi(\boldsymbol{r}), \tag{5.3}$$

as it is of sufficient interest, generality and complexity to serve as a typical example.

Theoretical determination of high-resolution electron-microscopy (HREM) images requires the solution of (5.3) provided $\lambda = 1/4\pi k_z$ and $V(\boldsymbol{r}) \equiv V(x, y, z) = 2meU(\boldsymbol{r})/\hbar^2$ in which m and $\boldsymbol{k}$ are the relativistic values of the electron mass and wave vector respectively and $U(\boldsymbol{r})$ is the potential representing the thin specimen foil. In (5.3) the convention is such that the direction of electron incidence is along the z-axis. The electron wave function $\psi(\boldsymbol{r})$ can be obtained from $\phi(\boldsymbol{r})$ by using the relation

$$\psi(\boldsymbol{r}) = \mathrm{e}^{2\pi \mathrm{i}\boldsymbol{k}\cdot\boldsymbol{r}}\phi(\boldsymbol{r}). \tag{5.4}$$

Identifying the space coordinate z with the "time" t, solving (5.3) is tantamount to solving the TDSE for a particle moving a two-dimensional *time-dependent* potential, provided we restrict ourselves to the case of normal incidence ($k_x = k_y = 0$). In most applications the time dependence of the potential is not taken into account [9]. The objective of HREM-image simulation is to solve eq. (5.3) numerically and as already mentioned in chapter 1 this is most conveniently done by means of PF methods.

Alternatively, puting $\lambda = -1/2k$, $k = \omega/\mathrm{c}$, $k_x = k_y = 0$ and $V(\boldsymbol{r}) = V(x, y) = k^2[n(x, y)^2 - 1]$, (5.3) is the Fresnel approximation to the wave equation of light of frequency ω propagating through a medium of which the index of refraction depends on the transverse position (x, y) only. Under certain conditions the properties of light in an optical fiber can be described by such an equation [10–14]. As already mentioned in the introduction, the numerical methods used for this optical problem are identical to those for the HREM image simulation. In the discussion that follows the terminology used in the latter application will be adopted.

In the first applications of the multi-slice approach, matrix multiplications were used to perform the propagation and hence the calculation time was proportional to the square of the

number of sampling points (diffracted electron beams) N used to represent the electron wave function (see for example ref. [38]). Significant improvement in multi-slice computations has been the use of the FFT to calculate the free-electron propagator [10–14,92] thereby making the computation time proportional to $N \log N$ instead of N^2 (see also section 2.2). The main advantage of transforming to Fourier space is that it is easy to take derivatives with respect to the coordinates, just as in conventional spectral methods [24,93]. Although the gain in efficiency resulting from the use of FFT's is substantial, the FFT multi-slice approach still has some drawbacks, from physical as well as computational point of view. In particular the artificial periodicity resulting from the use of the FFT and the practical constraint of being limited to a number of mesh-points that is a power of two, reduce the usefulness ofthis type of algorithms. As an extensive discussion of these aspects is given in ref. [9] there is no need to reproduce it here.

Improvement of the computational technique requires the development of accurate and efficient methods that perform the free-electron propagation on the electron wave function in its real-space representation, without making the detour to Fourier space [9,94]. The so-called real-space multi-slice (RSMS) method, proposed by Van Dyck [94] has been a first step in this direction. It has been pointed out that the original proposal had some serious deficiencies [95,96], originating from the fact that the time-step operator for free-electron propagation was approximated in a way that violates the basic requirement of stability. In fact this approach amounts to solving the parabolic equation by standard explicit method [19].

Recently a more systematic approach has been introduced in which the action of the free-electron propagator on the wave function in real space is approximated by forming a linear combination of the values of the wave function before propagation [97,98]. The coefficients specifying the linear combination are chosen such that the resulting approximant to the free-electron propagator combines the values of the wave function in a limited number of sampling points. This implies that the computation time becomes proportional to N instead of $N \log N$. A possible disadvantage of this approach is that it needs a "tuning" of the parameters determining the accuracy of the approximant to the propagator. In addition, both the FFT-based approach and the RSMS method require a smoothing procedure [97,98] for the crystal potential to reduce the number of mesh points to an acceptable level.

Clearly one of the central problems is to perform the free-electron propagation by a real-space method. It is sufficient to concentrate on propagation in only one direction since in most cases a two-dimensional orthogonal sampling grid can be used [9]. Experience has shown that for many materials, employing the most simple three-point approximation to $\partial^2/\partial x^2$ may render the computational method less efficient [9,97,98], and similar conclusions have been reached in quite different contexts [3,99–101]. Therefore it may be of interest to develop a real-space SPF algorithm that can deal with this more complicated situation.

In general it is clear that the increase in computing time, due to the use of a five or more point approximation to $\partial^2/\partial x^2$, should be compared to the gain in accuracy or, in other words to the extent by which the number of mesh points can be decreased. As the shape and strength of the potential set upperlimits to both the mesh size and time step, the question of efficiency can only be studied by considering explicit examples. In this section this will be done by comparing a properly generalized version of RS SPF algorithms with existing RSMS code [97,98] for computing HREM images.

To solve a continuum problem such as (5.3) some discretization procedure is necessary. The

most straightforward approach is to replace $\partial^2/\partial x^2$ by one of its finite difference approximations. Quite generally one can write

$$-\frac{\partial^2}{\partial x^2}\psi(x)\Big|_{x=x_0} \approx \delta^{-2}\sum_{q=-Q}^{Q} c_q(Q)\psi(x_0+q\delta), \tag{5.5}$$

where Q determines the number of points in the difference formula, $c_q(Q)=c_{-q}(Q)$ are known coefficients (which will depend on the difference scheme used) and δ is the mesh size. Remark that instead of using finite differences it may be expedient to discretize the problem by other methods. For instance generalized meshes [101], more appropriate for the case at hand may be used or, as shown in section 6.3, changing the representation of the states may also be beneficial.

Following exactly the same procedure as outlined in chapter 1, the discretized kinetic energy can be expressed in terms of the lattice-model kinetic energy

$$K=\delta^{-2}\sum_{q=1}^{Q}\sum_{l=1}^{L} c_q(Q)\left(c_l^+ c_{l+q}+c_{l+q}^+ c_l\right)+c_0(Q)\delta^{-2}\sum_{l=1}^{L} n_l. \tag{5.6}$$

Note that the last term in (5.6) is the unit matrix multiplied by a constant factor. Hence it commutes with any other matrix and consequently $\exp(-\mathrm{i}tc_0(Q)\delta^{-2}\Sigma_{l=1}^{L}n_l)$ can be taken into account before or after the integration procedure. It changes the phase of the wave function in a trivial manner. In what follows this term will be dropped.

From (5.6) one can guess that generalizing the lattice-model description to the d-dimensional case merely amounts to replacing the indices l and q by vectors $\boldsymbol{n}$ and $\boldsymbol{e}$, i.e. the kinetic energy takes the form

$$T=\sum_{\boldsymbol{n}\in\Lambda}\sum_{\boldsymbol{e}\in Q} b_{\boldsymbol{e}}\left(c_{\boldsymbol{n}}^+ c_{\boldsymbol{n}+\boldsymbol{e}}+c_{\boldsymbol{n}+\boldsymbol{e}}^+ c_{\boldsymbol{n}}\right), \tag{5.7}$$

where in this case Q denotes the set of lattice vectors connecting two mesh points and $b_{\boldsymbol{e}}$ are known numbers. As already mentioned above, for most cases of interest it is sufficient to concentrate on propagation in x and y direction separately. In other words approximating $\mathrm{e}^{-\mathrm{i}\tau T}$ by a RS SPF is tantamount to finding an approximation to $\mathrm{e}^{-\mathrm{i}\tau K}$, as explained in appendix B. Actual computations however are carried out for 2D systems.

In the RSMS approach [98] the free-particle propagator for a slice of thickness ϵ is approximated by

$$\mathrm{e}^{-\mathrm{i}\tau K}\approx\sum_{l=1}^{L}\sum_{p=0}^{P} a_p(P,Q)\left(c_l^+ c_{l+p}+c_{l+p}^+ c_l\right), \tag{5.8}$$

where $\tau=\epsilon/4\pi k_z\delta^2$, $a_p(P,Q)$ is given by [98]

$$a_p(P,Q)=\lim_{N\to\infty}\frac{1}{N}\sum_{l=1}^{N}\mathrm{e}^{2\pi\mathrm{i}lp/N}\,\mathrm{e}^{-\mathrm{i}\tau N^2[c_0(Q)+2\Sigma_{q=1}^{Q}c_q(Q)\cos(2\pi lq/N)]}, \tag{5.9}$$

$P \leqslant L/2$ is an adjustable parameter that determines the accuracy of the approximation. The underlying idea of this approximation is the following [9,97,98]. As the velocity of the incident electron is large, the spreading-out of the wave function is limited to a relatively small region in real space of the electron propagates freely over a distance of one slice. Hence the wave function at a certain mesh point may be expressed as a restricted sum of wave functions at neighboring points before propagation. The computation time of RSMS is proportional to PL^d.

In (5.9) limit $N \to \infty$ has deliberately not been replaced by an integral sign in order to emphasize an apparent inconsistency with respect to the discretization procedure that led to (5.6). From (5.6) and (5.8) one would expect to find that $a_p(P, Q)$ is related to the *discrete* Fourier transform of $e^{-i\tau K}$, i.e. formula (5.9) without $\lim_{N\to\infty}$. As in most practical applications L is relatively small, the coefficients $a_p(P, Q)$ do not vanish rapidly as p increases, a vital property for the truncation to P terms to make sense. This is then remedied by taking as in (5.9) the limit $N \to \infty$ [98].

Note that the operator at the right-hand side of (5.8) is not unitary. In practice P is determined by comparing RSMS data obtained with different values of P to results of high-precision reference calculations and selecting the minimum value of P for which the RSMS data satisfy a particular error criterion [97,98]. An advantage of this approach is that once for a given material parameters such as Q, P, δ and ϵ have been determined, they can also be used to compute images of different structures of the same constituents or of other materials with similar electron scattering characteristics.

To develop RS_2 and RS_4 for the more complicated model (5.6) (or equivalently (5.7)) we simply repeat the procedure explained in chapter 2. The first step is to write

$$e^{-i\tau K} = \prod_{q=0}^{Q} e^{-i\tau K_q}, \tag{5.10}$$

with $K_q \equiv \delta^{-2} c_q(Q) \sum_{l=1}^{L} (c_l^+ c_{l+q} + c_{l+q}^+ c_l)$ and the second to construct an RS algorithm for each $e^{-i\tau K_q}$, as is explained in more detail in appendix B (see eqs. (B.9–11)). Implementation of these algorithms only requires some minor modifications of the code of expK presented in chapter 3. Clearly the computation of RS_2 or RS_4 will be proportional to QL^d.

To make contact with the real-space HREM-simulation technique, the coefficients $c_q(Q)$ are chosen to be [98]

$$c_q(Q) = \frac{2(-1)^q}{q^2} \frac{(Q!)^2}{(Q+q)!(Q-q)!}; \quad q \neq 0, \tag{5.11}$$

whereby $c_0(Q) \equiv -2\sum_{q=1}^{Q} c_q(Q)$. This choice corresponds to the $(2Q+1)$-point Lagrange formula for the second derivative (see for instance ref. [102], p. 914).

Table 5.3 contains some results for the error between the exact *continuum* free-electron wave function ($H = -\partial^2/\partial x^2 - \partial^2/\partial y^2$) and approximate wave functions computed by RS_4. The electron moves on a square of linear size $X = 1$ and is subject to periodic-boundary conditions. No results for RS_2 are shown because for the particular choice of τ and δ (see table 5.3) the error is too large for RS_2 to make sense. On the basis of the additivity of error bounds (see also

Table 5.3
Representative results for the absolute value of the error between the exact *continuum* wave function and the approximate wave functions for a free particle moving on a square of length $X=1$, computed by RS_4. The initial state is of the form $e^{2\pi i \boldsymbol{n}\cdot\boldsymbol{l}\delta}$ ($\boldsymbol{n}=(n_x, n_y)$), i.e. an eigenstate of the free-particle propagator. The number of time steps $m=100$, the mesh size $\delta=1/16$, and the time step $\tau=0.0005$. A dash indicates that the error has reached its maximum value in less than 100 integration steps

(n_x, n_y)	$Q=1$	$Q=2$	$Q=3$	$Q=4$	$Q=5$
(0, 0)	0	0	0	0	0
(0, 1)	0.26×10^{-1}	0.15×10^{-2}	0.19×10^{-2}	0.26×10^{-2}	0.31×10^{-2}
(1, 1)	0.51×10^{-1}	0.31×10^{-2}	0.37×10^{-2}	0.51×10^{-2}	0.62×10^{-2}
(2, 2)	0.78	0.66×10^{-1}	0.12×10^{-1}	0.86×10^{-2}	0.99×10^{-2}
(3, 3)	-	0.66	0.14	0.33×10^{-1}	0.10×10^{-1}

appendix A) it is to be expected that the error will grow with Q. Moreover from (5.11) it follows that for a fixed q, $c_q(Q)$ increases with Q and consequently this will lead to an additional contribution to the error on the SPF. However as Q increases the finite difference approximation itself will become more accurate. This competition between increase in accuracy resulting from the use of a better difference scheme and decrease in accuracy due to the increasing number of operators in the SPF is nicely illustrated in the second and third line of table 5.3. Once the regime is entered where the SPF effectively determined the error, the only way to improve the approximation is to reduce τ. In this regime the error scales with τ^4 as might have been anticipated from the discussion in chapter 2. From table 5.3 it is seen that for fixed Q the error increases as the wave vector of the initial state increases. This is not surprising. The larger the wave vector, the more rapid the oscillations of the wave function and hence the more difficult it becomes to represent accurately the derivatives of the wave function with respect to the coordinates by means of a $(2Q+1)$-point formula.

Both RS_2 and RS_4 have been incorporated into a RSMS program [97,98] by replacing calls to the RSMS code by calls to RS_2 or RS_4. The efficiency of RS_2 and RS_4 as compared to RSMS (with $L=16$, $P=7$, $Q\leqslant 6$) has been evaluated by using the same error criteria employed in RSMS HREM-image simulations. These differ from the RMS-based error analysis performed in chapter 4 in that only a few components (the most central, $\boldsymbol{n}$ close to zero) of the Fourier-transformed image are used to estimate the error [97,98], a criterion based on the experimental situation in which there is dominant forward scattering of the electrons. As discussed above, the error on these components can be reduced by enlarging Q. It follows that RS_2 is not as efficient as RSMS, the reason being that for a fixed slice thickness $\epsilon(\tau=\epsilon/4\pi k_z\delta^2\approx 0.14$ in this case) the number of intermediate steps m for each slice has to be taken rather large ($m\leqslant 10$) in order to satisfy the error criteria. Repeating the calculations using RS_4 leads to the conclusion that RS_4 is competitive with respective with respect to the RSMS algorithm [103].

6. Diffusion equation

From the theoretical foundation of the SPF (symmetrized product-formula) approach, presented in chapter 2, it should be evident that the principle ideas remain the same if "it" is

replaced by a positive number β. Instead of solving the TDSE (Time-Dependent Schrödinger Equation) we then solve the initial value problem for the "generalized" diffusion equation

$$\frac{\partial \Phi(\beta)}{\partial \beta} = -H\Phi(\beta), \tag{6.1}$$

where $\Phi(\beta) = (\Phi_1(\beta), \ldots, \Phi_N(\beta))^T$ is a vector representing the values of the variables at the "sites" $1, \ldots, N$, H is a real symmetric $N \times N$ matrix and $\Phi \equiv \Phi(0)$ is the initial state.

Apart from applications to genuine diffusion problems where H is a finite-difference approximation to the Laplacian, methods to solve (6.1) are of interest to quantum mechanics because they can be used to compute the ground state properties of a model described by a Hamiltonian H. The aim of this chapter is to demonstrate the usefulness of SPF-based algorithms by applying them to quantum-mechanical problems. I stress however that the SPF algorithm to be discussed below can equally well serve to solve a d-dimensional diffusion equation on a lattice of arbitrary shape. My motivation for chosing quantum problems is merely a matter of having ample examples at hand.

To see how (6.1) or equivalently, its formal solution $\Phi(\beta) = e^{-\beta H}\Phi$, can be employed to compute the eigenvector corresponding to smallest eigenvalue of H, let us denote eigenvalues and (normalized) eigenvectors of H by $E_0 \leqslant E_1 \leqslant \cdots$ and ψ_n, $n = 0, \ldots, N-1$ respectively and assume that the ground state ψ_0 is non-degenerate. Expanding the arbitrary initial state Φ in terms of ψ_n gives

$$\Phi(\beta) = e^{-\beta H}\Phi = e^{-\beta E_0}\left(\psi_0\langle\psi_0 \mid \Phi\rangle + \sum_{n\geqslant 1}^{N-1} e^{-\beta(E_n - E_0)}\psi_n\langle\psi_n \mid \Phi\rangle\right), \tag{6.2}$$

showing that for $\beta(E_n - E_0) \gg 0$, $n \geqslant 1$ the sum in (6.2) can be neglected so that $\lim_{\beta\to\infty}\Phi(\beta)/\| \Phi(\beta) \| = \psi_0$.

The reader may have noticed that this technique of determining the eigenstate corresponding to the smallest eigenvalue of H is basically the same as the well-known inverse iteration method, i.e. the power method applied to the matrix H^{-1} [104]. Indeed putting $\beta = m\tau$ and keeping τ fixed, ψ_0 is found from $(e^{-\tau H})^m\Phi(0)$, i.e. by repeated iteration with $e^{-\tau H}$. Note that the replacement of H^{-1} by the exponent of H will increase the rate of convergence considerably. As the effect of applying $e^{-\tau H}$ to an arbitrary state Φ is to filter out all but the projection onto the ground state ψ_0, this approach is often called projector method (PM) [105–107]. Application of this approach to for instance many-body systems is impossible because of the large number of arithmetic operations required to complete one iteration step. A possible way out is then to perform the inverse iteration by importance sampling methods [105,108–110].

Just as in the case of the TDSE, in practice it will not be possible to compute $(e^{-\tau H})^m\Phi$ when H is not diagonal and Φ is arbitrary but it can be calculated approximately by means of the SPF. In this chapter we will confine ourselves to a discussion of second-order SPF algorithms since for our purpose they have proven to be adequate. To simplify the notation somewhat introduce the operator $H(\tau)$ defined by

$$e^{-\tau H(\tau)} \equiv e^{-\tau H_1/2}\, e^{-\tau H_2}\, e^{-\tau H_1/2}, \tag{6.3}$$

where $H = H_1 + H_2$. This construction makes sense because the right-hand side of (6.3) is positive definite. Also note that $e^{-\tau H(\tau)}$ is symmetric. In concert with the convention above, the eigenvalues and corresponding eigenvectors of $H(\tau)$ will be denoted by $E_0(\tau) \leqslant E_1(\tau) \leqslant \cdots$ and $\psi_n(\tau)$, $n = 0, \ldots, N-1$ respectively.

In order to show that $H(\tau)$ converges to H first note that by definition $\| e^{-\tau H} \| = e^{-\tau E_0}$, $\| e^{-\tau H(\tau)} \| = e^{-\tau E_0(\tau)}$. Furthermore by the Rayleigh–Ritz variational principle $E_0 \geqslant E_0^{(1)} + E^{(2)}$ where $E_0^{(1)}$ and $E_0^{(2)}$ are the smallest eigenvalues of H_1 and H_2 respectively. Putting $A = -H_1$ and $B = -H_2$ in identity (A.19) and making use of the elementary properties just mentioned we obtain

$$\| e^{-\tau H} - e^{-\tau H(\tau)} \| \leqslant c_2 \tau^3 \, e^{-\tau(E_0^{(1)} + E_0^{(2)})}, \tag{6.4}$$

from which it follows immediately that $\lim_{\tau \to 0} H(\tau) = H(0) = H$. Approximating $\Phi(m\tau) = e^{-m\tau H}\Phi$ by $\tilde{\Phi}(m\tau) = e^{-m\tau H(\tau)}\Phi$, and invoking (6.4) reveals that

$$\lim_{m \to \infty} \lim_{\tau \to 0} \tilde{\Phi}(m\tau) / \| \tilde{\Phi}(m\tau) \| = \psi_0, \tag{6.5}$$

hardly a surprising result.

Not surprisingly, in practice it will be impossible to take the limit $\tau \to 0$ before we let $m \to \infty$ because this would imply that in order for $\beta = m\tau$ to be sufficiently large (remember that we must have $\beta(E_n - E_0) \gg 0$, $n \geqslant 1$), m has to be so large that this approach will become prohibitively costly. What is feasible though is to fix τ, perform a number (m) of "imaginary-time" steps and compare the result for $\tilde{\Phi}(m\tau)$ with ψ_0. As in the case of $e^{-\beta H}\Phi$, if β is sufficiently large $e^{-\beta H(\tau)}\Phi$ will be proportional to the ground state of $H(\tau)$, i.e. $\psi_0(\tau)$. The question is now to what extent $E_0(\tau)$ and $\psi_0(\tau)$ are related to E_0 and ψ_0.

6.1. Well-separated eigenvalues

To investigate this problem theoretically assume for the sake of simplicity that the ground state ψ_0 is not nearly-degenerate, i.e. $E_n - E_0 \gg 0$, $n \geqslant 1$. Our aim is to derive upperbounds on the difference between the exact and approximate ground-state energy $| E_0 - E_0(\tau) |$ and the RMS error $\| \psi_0 - \psi_0(\tau) \|$. Application of the inequality $| \| A \| - \| B \| | \leqslant \| A - B \|$ to (6.4) gives

$$| e^{-\tau E_0} - e^{-\tau E_0(\tau)} | \leqslant c_2 \tau^3 \, e^{-\tau(E_0^{(1)} + E_0^{(2)})}. \tag{6.6}$$

To derive an upperbound on $| E_0 - E_0(\tau) |$ we first prove that $E_0(\tau) \leqslant E_0$. As $e^{\tau H(-\tau)} = (e^{-\tau H(\tau)})^{-1}$, repeatedly using $\| AB \| \leqslant \| A \| \, \| B \|$ gives $\langle \psi_0(\tau) | e^{\tau H(-\tau)} \psi_0(\tau) \rangle = e^{\tau E_0(\tau)} = \langle e^{\tau H_2/2} \, e^{\tau H_1/2} \psi_0(\tau) | e^{\tau H_2/2} \, e^{\tau H_1/2} \psi_0(\tau) \rangle \leqslant \| e^{\tau H_2/2} \, e^{\tau H_1/2} \|^2 \leqslant e^{\tau(E_0^{(1)} + E_0^{(2)})} \leqslant e^{\tau E_0}$ and hence $E_0(\tau) \leqslant E_0$. As $e^x - 1 > x$, (6.6) then directly leads to

$$| E_0 - E_0(\tau) | \leqslant c_2 \tau^2 \, e^{\tau(E_0 - E_0^{(1)} - E_0^{(2)})}. \tag{6.7}$$

An upperbound on $\| \psi_0 - \psi_0(\tau) \|$ can be derived as follows. Multiply both sides of the identities

$$\langle \psi_0(\tau) | \psi_n \rangle = \frac{\langle \psi_0(\tau) | e^{-\tau H} - e^{-\tau H(\tau)} | \psi_n \rangle}{e^{-\tau E_n} - e^{-\tau E_0(\tau)}}, \tag{6.8a}$$

and

$$\langle \psi_n | \psi_0(\tau) \rangle = \frac{\langle \psi_n | e^{\tau H(-\tau)} - e^{\tau H} | \psi_0(\tau) \rangle}{e^{\tau E_0(\tau)} - e^{\tau E_n}}, \tag{6.8b}$$

to obtain

$$| \langle \psi_n | \psi_0(\tau) \rangle |^2 = \frac{\langle \psi_0(\tau) | e^{-\tau H} - e^{-\tau H(\tau)} | \psi_n \rangle \langle \psi_n | e^{\tau H(-\tau)} - e^{\tau H} | \psi_0(\tau) \rangle}{4 \sinh^2 [\tau (E_0(\tau) - E_n)/2]}, \tag{6.8c}$$

A similar expression for $| \langle \psi_n(\tau) | \psi_0 \rangle |^2$ is found by interchanging the role of $\psi_n(\tau)$ and ψ_n, $n \geqslant 0$. Consider how the case $n \geqslant 1$ and make the very crucial assumption that

$$| E_0(\tau) - E_n | \geqslant \epsilon_1; \quad n \geqslant 1, \tag{6.9a}$$

and

$$| E_0 - E_n(\tau) | \geqslant \epsilon_2; \quad n \geqslant 1, \tag{6.9b}$$

where $\epsilon_1 > 0$ and $\epsilon_2 > 0$ are constants, the meaning of which will be discussed later. From (6.8) it follows that

$$\sum_{n \geqslant 1}^{N-1} | \langle \psi_n | \psi_0(\tau) \rangle |^2 \leqslant \frac{1}{\epsilon_1^2 \tau^2} \sum_{n \geqslant 1}^{N-1} \langle \psi_0(\tau) | e^{-\tau H} - e^{-\tau H(\tau)} | \psi_n \rangle \langle \psi_n | e^{\tau H(-\tau)} - e^{\tau H} | \psi_0(\tau) \rangle. \tag{6.10}$$

Extending the sum on the right-hand side of (6.10) to include the term with $n = 0$, noting that $\| e^{\tau H} - e^{\tau H_1/2} e^{\tau H_2} e^{\tau H_1/2} \| \leqslant \| e^{-\tau H} - e^{-\tau H_1/2} e^{-\tau H_2} e^{-\tau H_1/2} \| e^{\tau (E_0 + E_0^{(1)} + E_0^{(2)})}$ and involving (6.40) yields $\sum_{n \geqslant 1}^{N-1} | \langle \psi_n | \psi_0(\tau) \rangle |^2 \leqslant c_2^2 \epsilon_1^{-2} \tau^4 e^{\tau (E_0 - E_0^{(1)} - E_0^{(2)})}$. This is equivalent to $1 - | \langle \psi_0 | \psi_0(\tau) \rangle |^2 \leqslant c_2^2 \epsilon_1^{-2} \tau^4 e^{\tau (E_0 - E_0^{(1)} - E_0^{(2)})}$. In the same manner one finds $1 - | \langle \psi_0 | \psi_0(\tau) \rangle |^2 \leqslant c_2^2 \epsilon_2^{-2} \tau^4 e^{\tau (E_0 - E_0^{(1)} - E_0^{(2)})}$ and combining both results then gives

$$1 - | \langle \psi_0 | \psi_0(\tau) \rangle | \leqslant c_2^2 \epsilon^{-2} \tau^4 e^{\tau (E_0 - E_0^{(1)} - E_0^{(2)})}, \tag{6.11}$$

where $\epsilon = \max(\epsilon_1, \epsilon_2)$. To obtain the desired bound on the RMS error it is necessary to remove a rather trivial ambiguity. Clearly if ψ_0 is an eigenstate so is $-\psi_0$ and this change of sign will have a drastic effect on $\| \psi_0 - \psi_0(\tau) \|$. In order to circumvent this problem choose the sign of ψ_0 such that $\mathrm{Re} \langle \psi_0 | \psi_0(\tau) \rangle \geqslant 0$. Further note that since H and $H(\tau)$ are real and symmetric all

eigenvectors are real. Then $\| \psi_0 - \psi_0(\tau) \| = \sqrt{2(1 - \mathrm{Re}\langle \psi_0 \mid \psi_0(\tau)\rangle)} = \sqrt{2(1 - |\langle \psi_0 \mid \psi_0(\tau)\rangle|}$, and making use of (6.11) gives

$$\| \psi_0 - \psi_0(\tau) \| \leqslant c_2 \epsilon^{-1} \tau^2 \, \mathrm{e}^{\tau(E_0 - E_0^{(1)} - E_0^{(2)})/2}. \tag{6.12}$$

From upperbound (6.7) we learn that if $m\tau(E_n - E_0) \gg 0$ ($n \geqslant 1$), m successive applications of $\mathrm{e}^{-\tau H(\tau)}$ to an almost arbitrary state Φ will yield an estimate for the ground-state energy $E_0(\tau)$, the error vanishing with τ^2. The upperbound (6.12) shows that if (6.9) is satisfied with $\epsilon = \mathcal{O}(1)$ (compared to τ), the same procedure will also yield a good estimate for the ground state itself. However (6.12) also suggest that if ϵ is small, there is no guarantee that the SPF approach will provide us with an accurate estimate of ψ_0, and this in spite of the fact that $| E_0 - E_0(\tau) |$ may be very small.

The most important but not the only class of problems where this break-down of SPF-based projector methods may occur are systems with nearly-degenerate ground states. Obviously for such systems it will be difficult to meet requirement (6.9) with ϵ not too small or, in other words, τ must be taken so small and the number of steps m so large (remember we should also satisfy $\beta(E_1 - E_0) \gg 0$) that the technique will become very inefficient. A way out of this dilemma is to try to find approximations to all nearly-degenerate eigenstates simultaneously. How this can be accomplished is explained in the next section.

6.2. Very close eigenvalues

For simplicity of presentation it will be assumed that $E_0 \approx E_1$ and $E_1 < E_2 \leqslant E_3 \leqslant \cdots$, i.e. only the two states with the smallest eigenvalues are (nearly) degenerate. The method to be proposed is sufficiently general to allow extension to situations where more than two eigenvalues are poorly separated. It is inspired by a well-known technique to determine a set of very close dominant eigenvalues but for our applications has proven to be superior. The basic idea is to combine the project technique and the variational principle into what will be called extended projector method (EPM).

Under the conditions stated above, iterating with $\mathrm{e}^{-\tau H}$ on two different states $\Phi_0 \equiv \Phi_0(0)$ and $\Phi_1 \equiv \Phi_1(0)$ will give for $j = 0, 1$

$$\Phi_j(\beta) = \mathrm{e}^{-\beta E_0}\big(\psi_0\langle \psi_0 \mid \Phi_j\rangle + \mathrm{e}^{-\beta(E_1 - E_0)}\psi_1\langle \psi_1 \mid \Phi_j\rangle + \mathcal{O}(\mathrm{e}^{-\tau(E_2 - E_0)})\big). \tag{6.13}$$

From (6.13) it is seen that if $\beta(E_2 - E_0) \gg 0$, the states $\Phi_0(\beta)$ and $\Phi_1(\beta)$ will span the two-dimensional subspace containing the eigenstates ψ_0 and ψ_1 provided $\langle \psi_i \mid \Phi_j\rangle \neq 0$, $i, j = 0, 1$. As $E_1 \approx E_0$, the PM is expected to be inefficient because a very large number of iterations m ($\beta = m\tau$) will be required to render the second term of the right-hand side of (6.13) sufficiently small.

To separate the two eigenvalues the variational principle is applied as follows. Consider an iteration step taking $\Phi_j(m\tau)$ to the states $\Phi_j((m+1)\tau)$. Assuming these vectors to be linearly independent, they span a two-dimensional subspace. Denote the projector onto this sub-space by P_{m+1}. By Poincaré's theorem [105,106] the eigenvalues of $P_{m+1}HP_{m+1}$, to be denoted by $E_j^{(m+1)}$, are upperbounds to the two smallest eigenvalues of H, i.e. $E_j \leqslant E_j^{(m+1)}$. To compute $E_j^{(m+1)}$ we

should calculate the matrix elements $P_{m+1}HP_{m+1}$, but this can be avoided by applying the same variational principle to the operator $e^{\tau H}$ directly. Solving the eigenvalue problem of $P_{m+1}\, e^{\tau H}P_{m+1}$ will yield upperbounds on $e^{\tau E_j}$ instead of on E_j and as shown below it is more efficient to compute the matrix elements of $P_{m+1}\, e^{\tau H}P_{m+1}$ than those of $P_{m+1}HP_{m+1}$. The iteration step then ends by applying to $\Phi_j((m+1)\tau)$ the unitary transformation diagonalizing $P_{m+1}HP_{m+1}$. The resulting states are then used as input for the next iteration step. In this manner optimal use has been made of the information hidden in $\Phi_j((m+1)\tau)$, both from theoretical and computational point of view.

In this discussion it has tacticly been assumed that both the projection and the variational calculation can be carried out using the exact operator $e^{-\beta H}$ but in practice this will likely prove to be impossible and we have to invoke the SPF. In practice this algorithm has been implemented as follows. Use the SPF scheme RS_2 to compute $\tilde{\Phi}_j((m+1)\tau) = e^{-\tau H(\tau)}\tilde{\Phi}_j(m\tau)$. Construct the matrices

$$B_{i,j} = \left\langle \tilde{\Phi}_i((m+1)\tau) \,|\, \tilde{\Phi}_j((m+1)\tau) \right\rangle, \tag{6.14a}$$

and

$$\begin{aligned} A_{i,j} &= \left\langle \tilde{\Phi}_i((m+1)\tau) \,|\, e^{\tau H(\tau)} \,|\, \overline{\Phi}_j((m+1)\tau) \right\rangle \\ &= \left\langle \tilde{\Phi}_i((m+1)\tau) \,|\, \tilde{\Phi}_j(m\tau) \right\rangle, \end{aligned} \tag{6.14b}$$

note that B is positive definite, and solve the 2×2 eigenvalue problem $Ax = \lambda Bx$ using a standard method. Applying the unitary transformation that diagonalizes A and B to the vector $\tilde{\Phi}_j((m+1)\tau)$ and normalize the resulting vectors for numerical convenience. Replace $\tilde{\Phi}_j(m\tau)$ by the states thus obtained to complete the iteration step.

The presentation above has been restricted to the case where simultaneous determination of the two smallest eigenvalues and the corresponding eigenvectors was required but it is obvious how the method should be generalized. One simply has to take as many different initial states as the number of eigenvalues desired. In principle there is the possibility that in the course of the procedure, two or more of the iterated states become linearly dependent. Then the iteration scheme would break down. In practice however we have never seen this happen if the initial states were chosen randomly. Needless to say that each of these states must have a non-zero projection onto at least one of the searched-for eigenstates.

6.3. Spin-boson system

This section illustrates the application of the EPM to the problem of determining the low-lying states of two-level system coupled to an oscillator. The model Hamiltonian reads

$$H = -h\sigma^x + \sqrt{\Omega C}\,(a^+ + a)\sigma^z + \Omega a^+ a, \tag{6.15}$$

where σ^α, $\alpha = x, y, z$ are the Pauli-spin matrices describing the two-state system and $a^+(a)$ is

the creation (annihilation) operator of an oscillator mode of frequency Ω (units are such that $\hbar = 1$). If the coupling constant $C = 0$ the tunnel frequency of the spin is $2h$. For large C the ground state is nearly degenerate, implying that the effective tunnel frequency is small. Consequently keeping h and Ω fixed and changing C from zero to a large value ($C \gg h$, $C \gg \Omega$) will alter the properties of the eigenvalues and eigenstates of (6.15) considerably.

To be more specific let us concentrate on the case $\Omega > h$. If $C \approx 0$ the two lowest energy levels will differ by about $2h$. This difference diminishes monotonically with increasing C. As the exact eigenvalues and eigenstates of (6.15) can be calculated to high accuracy [113–117], model (6.15) provides an excellent example to test both the power of the SPF-based PM and its extension, the EPM. In our discussion we will confine ourselves to those aspects relevant to the material presented above, a much more complete analysis being given elsewhere [118].

We start by decomposing $H = H_1 + H_2$, $H_1 = -h\sigma^x$ and use $e^{-\tau H_1} = \mathbb{1} \cosh \tau h + \sigma^x \sinh \tau h$ to compute its action. To calculate the action of $e^{-\tau H_2}$ we first have to choose a representation for the states of the system. It is appropriate to take as basis states the direct product of the oscillator eigenstates $|n\rangle$, $n = 0, \ldots, N-1$ (i.e. $a^+ a\,|n\rangle = n\,|n\rangle$) and the eigenstates of σ^z, i.e. $\sigma^z\,|S\rangle = S\,|S\rangle$, $S = \pm 1$ [113]. In this basis H_2 is tridiagonal with respect to the oscillator states and diagonal in spin space. This means that in fermion language H_2 can be written as

$$H_2 = \sum_{n=0}^{N-1} \left[\sqrt{n\Omega C}\,\sigma^z (c_n^+ c_{n+1} + c_{n+1}^+ c_n) + n\Omega c_n^+ c_n\right]. \tag{6.16}$$

Clearly (6.16) has the same structure as Hamiltonian (1.7) and consequently the methodology developed in chapter 2 can be applied. The break-up used in this particular case reads

$$e^{-\tau H} \approx e^{-\tau H_1/2}\, e^{-\tau U/2}\, e^{-\tau K_O/2}\, e^{-\tau K_E}\, e^{-\tau K_O/2}\, e^{-\tau U/2}\, e^{-\tau H_1/2}, \tag{6.17a}$$

or

$$e^{-\tau H} \approx e^{-\tau H_1/2}\, e^{-\tau H_O/2}\, e^{-\tau H_E}\, e^{-\tau H_O/2}\, e^{-\tau H_1/2}, \tag{6.17b}$$

where U, K_O etc. have the same meaning as in chapter 2. Extensive calculations have shown that the former decomposition is slightly better than the latter and has therefore been used to compute the results presented below. Remark however that there is no argument pro or contra this finding.

To compute the low-lying states of (6.15) without invoking the SPF, we rely on Poincaré's theorem once more and solve numerically the eigenvalue problem of the truncated matrix, representing H in the basis $|S\rangle \otimes |n\rangle$, $S = \pm 1$, $n = 0, \ldots, N-1$. The value of N is determined by requiring the eigenvalues and eigenstates to be independent of N. In practice $N \leqslant 32$ has found to be adequate for the majority of parameter values.

Fig. 6.1 shows the RMS error between the exact ground state and the "ground state" obtained from the PM. In this calculation the model parameters were chosen to be $h = 1$, $\Omega = 8$, and $C = 10$, 20, 30. Table 6.1 lists the corresponding differences $E_1 - E_0$ and $E_2 - E_0$. From these data we may expect on the basis of the theory of the two previous sections that the RMS error on the ground state will grow significantly if $C \to 30$ and as demonstrated in fig. 6.1 this is indeed

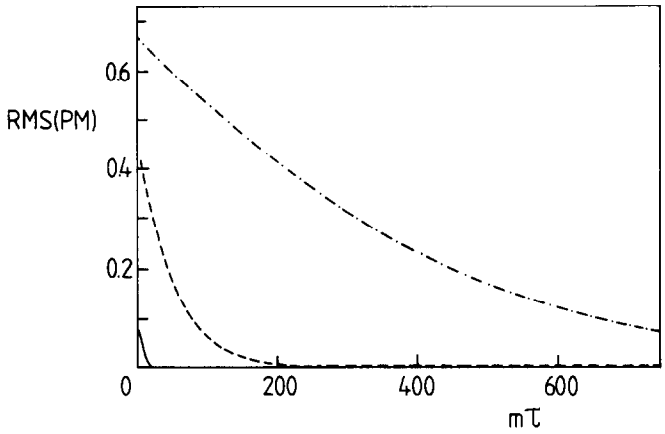

Fig. 6.1. The RMS error $\mathrm{RMS}(PM) = \|\psi_0 - \phi(m\tau)\|$ between the exact ground state ψ_0 and iterated state $\phi(m\tau) = \mathrm{e}^{-m\tau H(\tau)}\Phi / \|\mathrm{e}^{-m\tau H(\tau)}\Phi\|$ as a function of the number of iterations m, calculated by applying the projector method to a two-level system coupled to an oscillator. The initial state Φ was chosen randomly and $h = 1$ and $\Omega = 8$. Solid line: $C = 10$ and $\tau = 0.015$, dashed line: $C = 20$ and $\tau = 0.010$, dashed-dotted line: $C = 30$ and $\tau = 0.005$.

the case. Note that if $C = 30$, doing 150000 iteration steps is not sufficient to obtain the ground state with reasonable accuracy although the ground-state energy is reproduced quite well. This example shows that the PM can be very inefficient.

Employing the EPM yields the results depicted in fig. 6.2. Apparently the problem due to the nearly-degenerate ground state has been removed. For $C = 30$, within less than 300 iterations the exact ground state is recovered within an error determined by τ^2. Iterating on for instance four instead of on two states the EPM reproduces the first four low-lying states. Experience [118] has shown that when applied to this model, the EPM performs much better than techniques based on orthogonalization [104].

In general one cannot know beforehand whether or not the ground state is nearly two-fold degenerate. Therefore it is always good practice to employ the EPM with at least two but preferably more states. Also in the case where the ground state is well separated from the first

Table 6.1
Exact ground-state energy and differences between the lowest (approximate) energies of a two-level system coupled to an oscillator. The tunnel energy $h = 1$ and phonon frequency $\Omega = 8$

	$C = 10$ $\tau = 0.015$	$C = 20$ $\tau = 0.010$	$C = 30$ $\tau = 0.005$
E_0	10.114	20.021	30.029
$E_1 - E_0$	0.164	0.014	0.003
$E_2 - E_1$	7.756	7.940	7.991
$E_0 - E_0(\tau)$	0.033	0.039	0.020
$E_1 - E_0(\tau)$	0.179	0.053	0.021
$E_1(\tau) - E_0$	0.135	0.022	0.016

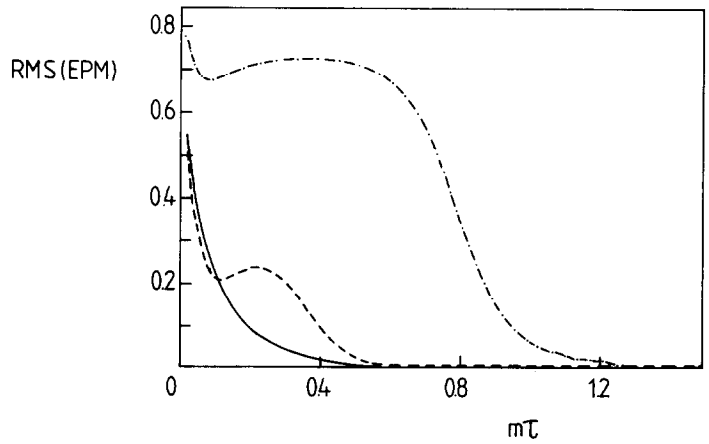

Fig. 6.2. Same as fig. 6.1 but instead of the projector-method the extended projector method was used to compute RMS(*EPM*). Note that compared to fig. 6.1 there is a large difference with respect to the scale of m.

excited state, the EPM is more efficient than the PM and in addition the calculation will tell us that the ground state is indeed well-separated.

7. Concluding remarks

Starting from generalized symmetrized formulae (SPF), a new family of explicit and unconditionally stable algorithms for solving parabolic equations has been introduced. Considerable attention has been paid to the analysis, both mathematically and numerically, of several different schemes that can be derived by applying the basic idea's behind the SPF approach in a straightforward manner.

From computational point of view, appealing features of some of the SPF algorithms are that they are easily vectorizable and intrinsically parallel. The computational efficiency of the algorithms has been evaluated and their performance has been compared to that of the Cranck–Nicholson method. An important conclusion is that for one-, two- or three-dimensional problems, the SPF algorithms are much more efficient than the Cranck–Nicholson scheme. This is not only due to the explicit character of the algorithms but is also due to the fact that SPF's are providing more accurate approximations to the time-step operator than conventional methods.

The most efficient SPF algorithm has been used to study Anderson localization on time and length scales previously inaccessible. Other applications have shown that the SPF method is flexible and can easily be adapted to deal with various kinds of boundary conditions, systems with different types of degrees of freedom (such as spin-boson models) etc. As only a small fraction of the possibilities to apply the product formula philosophy have been explored, there is a much room for further research in this field. I hope that this will encourage the reader to apply the SPF approach to solve other problems or develop new and more efficient algorithms than the ones proposed in this paper.

Acknowledgements

In the course of this project I have benefitted a lot from stimulating discussions with W. Coene, S. Das Sarma, P. de Vries, W. Götze, I. Heynderickx, A. Lagendijk, R. Tahir-Kheli, D. Van Dyck, and M. Vanhimbeeck. I am grateful for their constructive criticism and useful suggestions. I thank B. Kramer for providing me with a copy of G.M. Scher's Ph.D. thesis. This work is partially supported by the "Supercomputer Project" of the Belgian National Science Foundation (N.F.W.O.), the research program of the Dutch "Stichting voor Fundamenteel Onderzoek der Materie" (FOM) which is financially supported by the "Nederlandse Organizatie voor Zuiver Wetenschappelijk Onderzoek" (ZWO), and NATO grant no. RG.86/0026. The author thanks the N.F.W.O. for financial support.

Appendix A. Error bounds

This appendix contains the necessary mathematics for proving the convergence of the various approximation schemes. For numerical applications it is sufficient to consider the case where wave functions are vectors in a finite-dimensional Hilbert space and operators being finite, square matrices. The norm of a vector in this space is taken to be the usual one, namely $\| \phi \| \equiv \sqrt{\langle \phi | }$ where $\langle \phi | \psi \rangle \equiv \Sigma_i \phi_i^* \psi_i$ stands for the scalar product of vector $\phi \equiv (\ldots, \phi_i, \ldots)$ and $\psi \equiv (\ldots, \psi_i, \ldots)$. The norm of an operator X is defined by $\| X \| = \max_{\| \psi \| = 1} \| X\psi \|$. Throughout this paper, wave functions are assumed to be normalized.

To obtain upperbounds for the RMS error between the exact and approximate wave function at time t the following general result, easily proven by mathematical induction, is needed. For two operators X and Y one has

$$\| X^m - Y^m \| \leqslant \sum_{n=0}^{m-1} \| X - Y \| \, \| X \|^n \, \| Y \|^{m-n-1}, \tag{A.1}$$

which in the special case that X and Y are unitary operators simplifies to

$$\| X^m - Y^m \| \leqslant m \| X - Y \|. \tag{A.2}$$

Putting $X = \mathrm{e}^{-\mathrm{i}\tau H}$, $t = m\tau$ and denoting the approximant to the time-step operator by Y yields

$$\| \psi_X(t) - \psi_Y(t) \| \leqslant m \| X - Y \|, \tag{A.3}$$

where $\psi_X(t) = X^m \psi(0)$, $\psi_Y(t) = Y^m \psi(0)$ and $\psi(0)$ is the wavefunction at time $t = 0$, i.e. the initial condition. From (A.3) it is seen that the desired error bound follows directly from the knowledge of $\| X - Y \|$. Note however that because of the triangle inequality $\| A + B \| \leqslant \| A \| + \| B \|$, there is the trivial bound $\| \psi_X(t) - \psi_Y(t) \| \leqslant 2$. Therefore the right-hand side of (A.3) has to be less than 2 in order for the bound obtained from (A.3) to be meaningfull.

We now compute $\| X - Y \|$ for the Cranck–Nicholson (CN), the first-order, second-order

and fourth-order SPF (symmetrized product-formula) method. In the case of the CN method $Y = (1 - i\tau H/2)(1 + i\tau H/2)^{-1}$. Consider the operator

$$F(\tau) = 1 - X^{-1}Y = 1 - e^{i\tau H}\left(1 - \frac{i\tau H}{2}\right)\left(1 + \frac{i\tau H}{2}\right)^{-1}. \tag{A.4}$$

Differentiating (A.4) with respect to τ gives

$$\frac{\partial F(\tau)}{\partial \tau} = -\frac{i\tau^2 H^3}{4} e^{i\tau H}\left(1 + \frac{i\tau H}{2}\right)^{-2}. \tag{A.5}$$

Integration of (A.5) over τ and making use of $F(0) = 0$ yields

$$F(\tau) = -\frac{iH^3}{4}\int_0^\tau d\lambda\ \lambda^2\, e^{i\lambda H}\left(1 + \frac{i\lambda H}{2}\right)^{-2}, \tag{A.6}$$

from which it follows directly that

$$\left\| e^{-i\tau H} - \left(1 - \frac{i\tau H}{2}\right)\left(1 + \frac{i\tau H}{2}\right)^{-1} \right\| \leqslant \tfrac{1}{4}\int_0^\tau d\lambda\ \lambda^2 \left\| H^3\, e^{-i(\tau-\lambda)H}\left(1 + \frac{i\lambda H}{2}\right)^{-2} \right\|. \tag{A.7}$$

Since $\| e^{-i(\tau-\lambda)H} \| = 1$ and $\| (1 + i\lambda H/2)^{-1} \| \leqslant 1$, (A.7) leads to

$$\left\| e^{-i\tau H} - \left(1 - \frac{i\tau H}{2}\right)\left(1 + \frac{i\tau H}{2}\right)^{-1} \right\| \leqslant \frac{\tau^3 \| H \|^3}{12}, \tag{A.8}$$

hence

$$\| \psi_{\text{exact}}(t) - \psi_{\text{CN}}(t) \| \leqslant \frac{t\tau^2 \| H \|^3}{12}, \tag{A.9}$$

where $\psi_{\text{exact}}(t) \equiv e^{-itH}\psi(0)$ and $\psi_{\text{CN}}(t) \equiv [(1 - i\tau H/2)(1 + i\tau H/2)^{-1}]^m \psi(0)$.

Calculation of similar bounds for the product-formulae proceeds along the same line but is more complicated than in the CN case because of the presence of noncommuting operators. The derivation of bounds on the first and second order formula presented below is taken from the work of Suzuki [30]. Introducing $F(\lambda) = 1 - e^{\lambda A} e^{\lambda B} e^{-\lambda(A+B)}$, differentiation with respect to λ gives

$$\frac{\partial F(\lambda)}{\partial \lambda} = e^{\lambda A}[e^{\lambda B}, A]\, e^{-\lambda(A+B)}. \tag{A.10}$$

Application of Kubo's identity

$$[A, e^{\lambda B}] = \int_0^\lambda d\mu\ e^{(\lambda-\mu)B}[A, B]\ e^{\mu B} = \int_0^\lambda d\mu\ e^{\mu B}[A, B]\ e^{(\lambda-\mu)B}, \tag{A.11}$$

yields

$$\frac{\partial F(\lambda)}{\partial \lambda} = -\int_0^{\lambda} \mathrm{d}\mu \, \mathrm{e}^{\lambda A} \, \mathrm{e}^{\mu B} [A, B] \, \mathrm{e}^{(\lambda-\mu)B} \, \mathrm{e}^{-\lambda(A+B)}. \tag{A.12}$$

Integrating (A.12) and using $F(0) = 0$ leads to

$$F(\tau) = -\int_0^{\tau} \mathrm{d}\lambda \int_0^{\lambda} \mathrm{d}\mu \, \mathrm{e}^{\lambda A} \, \mathrm{e}^{\mu B} [A, B] \, \mathrm{e}^{(\lambda-\mu)B} \, \mathrm{e}^{-\lambda(A+B)}, \tag{A.13}$$

and this in turn yields the identity

$$\mathrm{e}^{\tau(A+B)} = \mathrm{e}^{\tau A} \, \mathrm{e}^{\tau B} - \int_0^{\tau} \mathrm{d}\lambda \int_0^{\lambda} \mathrm{d}\mu \, \mathrm{e}^{\lambda A} \, \mathrm{e}^{\mu B} [A, B] \, \mathrm{e}^{(\lambda-\mu)B} \, \mathrm{e}^{(\tau-\lambda)(A+B)}. \tag{A.14}$$

In our case $a = -\mathrm{i}H_1$, $B = -\mathrm{i}H_2$, $H = H_1 + H_2$ and therefore

$$\| \mathrm{e}^{-\mathrm{i}\tau H} - \mathrm{e}^{-\mathrm{i}\tau H_1} \, \mathrm{e}^{-\mathrm{i}\tau H_2} \| \leqslant \frac{\tau^2}{2} \| [H_1, H_2] \|, \tag{A.15a}$$

Generalization of (A.15a) to the case where $H = \sum_{q=1}^{p} H_q$ is straightforward. Repeated application of the triangle inequality yields

$$\left\| \mathrm{e}^{-\mathrm{i}\tau H} - \prod_{q=1}^{p} \mathrm{e}^{-\mathrm{i}\tau H_q} \right\| \leqslant \frac{\tau^2}{2} \sum_{1 \leqslant q < q'}^{p} \| [H_q, H_{q'}] \|. \tag{A.15b}$$

Exactly the same procedure is used to compute an upperbound on the RMS error of the second-order product formula. Putting $F(\lambda) = 1 - \mathrm{e}^{\lambda A/2} \, \mathrm{e}^{\lambda B} \, \mathrm{e}^{\lambda A/2} \, \mathrm{e}^{-\lambda(A+B)}$, and differentiating with respect to λ gives

$$\frac{\partial F(\lambda)}{\partial \lambda} = \mathrm{e}^{\lambda A/2} \, \mathrm{e}^{\lambda B} G(\lambda) \, \mathrm{e}^{\lambda A/2} \, \mathrm{e}^{-\lambda(A+B)}, \tag{A.16a}$$

where

$$G(\lambda) = \tfrac{1}{2} \mathrm{e}^{-\lambda B} [\mathrm{e}^{\lambda B}, A] + [\mathrm{e}^{\lambda A/2}, B] \, \mathrm{e}^{-\lambda A/2}. \tag{A.16b}$$

Observing that $G(0) = \partial G(\lambda)/\partial \lambda \,|_{\lambda=0} = 0$ and integrating

$$\frac{\partial^2 G(\lambda)}{\partial \lambda^2} = \tfrac{1}{2} \mathrm{e}^{-\lambda B} [B, [A, B]] \, \mathrm{e}^{\lambda B} + \tfrac{1}{4} \mathrm{e}^{\lambda A/2} [A, [A, B]] \, \mathrm{e}^{-\lambda A/2}, \tag{A.17}$$

twice directly leads to

$$G(\lambda) = \int_0^{\lambda} \mathrm{d}\mu \int_0^{\mu} \mathrm{d}\nu \left(\tfrac{1}{2} \mathrm{e}^{-\nu B} [B, [A, B]] \, \mathrm{e}^{\nu B} + \tfrac{1}{4} \mathrm{e}^{\nu A/2} [A, [A, B]] \, \mathrm{e}^{-\nu A/2} \right). \tag{A.18}$$

Substituting (A.18) into (A.16) and integrating (A.16) then finally gives [35]

$$F(\tau) = \tfrac{1}{4}\int_0^\tau d\lambda \int_0^\lambda d\mu \int_0^\mu d\nu\ e^{\lambda A/2}\, e^{\lambda B}\{e^{-\nu B}[2B, [A, B]]\, e^{\nu B} + e^{\nu A/2}[A, [A, B]]\, e^{-\nu A/2}\}\, e^{\lambda A/2}\, e^{-\lambda(A+B)}. \tag{A.19}$$

Application of identity (A.19) to the case at hand results in

$$\| e^{-i\tau H} - e^{-i\tau H_1/2}\, e^{-i\tau H_2}\, e^{-i\tau H_1/2} \| \leqslant c_2 \tau^3, \tag{A.20a}$$

and

$$\| \psi_{\text{exact}}(t) - \psi_2(t) \| \leqslant c_2 t \tau^2, \tag{A.20b}$$

where $\psi_2(t) \equiv (e^{-\tau H_1/2}\, e^{-i\tau H_2}\, e^{-i\tau H_1/2})^m \psi(0)$. The constant c_2 appearing in (A.20) is defined by $c_2 \equiv (\,\|[H_1, [H_1, H_2]]\| + 2\,\|[H_2, [H_1, H_2]]\|)/24$. Generalizations of (A.20) can be found in Suzuki's paper [35].

Calculation of the error bounds of the fourth-order SPF is much more tedious but nevertheless straightforward. As the method of calculation has now been examplified twice in the above, we merely give the final results without proof. It can be shown that [35]

$$\| e^{-i\tau H} - e^{-i\tau H_1/2}\, e^{-i\tau H_2/2}\, e^{i\tau^3 C(H_1, H_2)}\, e^{-i\tau H_2/2}\, e^{-i\tau H_2/2} \| \leqslant c_4 \tau^5 + \mathcal{O}(\tau^7), \tag{A.21}$$

where $C(H_1, H_2) \equiv [H_1 + 2H_2, [H_1, H_2]]/24$ and

$$c_4 \equiv \| [C, [H_1, H_2]]/20 + ([H_1, [H_1, [H_1, [H_1, H_2]]]] + 4[H_2, [H_1, [H_1, [H_1, H_2]]]] + 6[H_2, [H_2, [H_1, [H_1, H_2]]]] + 4[H_2, [H_2, [H_2, [H_1, H_2]]]])/80 \|. \tag{A.22}$$

From (A.21) it then follows that

$$\| \psi_{\text{exact}}(t) - \psi_4(t) \| \leqslant c_4 t \tau^4 + \mathcal{O}(\tau^6), \tag{A.23}$$

in which $\psi_4(t) \equiv e^{-i\tau H_1/2}\, e^{-i\tau H_2/2}\, e^{i\tau^3 C(H_1, H_2)}\, e^{-i\tau H_2/2}\, e^{-i\tau H_1/2}\, \psi(0)$.

In practical applications of the fourth-order SPF, it is necessary to approximate $e^{i\tau^3 C(H_1, H_2)}$ by $\prod_{s=1}^r e^{i\tau^3 C_s}$ where $C(H_1, H_2) \equiv \sum_{s=1}^r C_s$. It is easy to prove that this replacement does not affect the order of correctness of the fourth-order SPF. By the triangle inequality

$$\| e^{-i\tau H} - e^{-i\tau H_1/2}\, e^{-i\tau H_2/2}\, e^{i\tau^3 C(H_1, H_2)}\, e^{-i\tau H_2/2}\, e^{-i\tau H_1/2} + e^{-i\tau H_1/2}\, e^{-i\tau H_2/2} \left(e^{i\tau^3 C(H_1, H_2)} - \prod_{s=1}^r e^{i\tau^3 C_s} \right) e^{-i\tau H_2/2}\, e^{-i\tau H_1/2} \| \leqslant c_4 \tau^5 + \mathcal{O}(\tau^6) + \left\| e^{i\tau^3 C(H_1, H_2)} - \prod_{s=1}^r e^{i\tau^3 C_s} \right\|. \tag{A.24}$$

Invoking (A.15b) then gives

$$\left\| e^{-i\tau H} - e^{-i\tau H_1/2}\, e^{-i\tau H_2/2}\left(\prod_{s=1}^{r} e^{i\tau^3 C_s}\right) e^{-i\tau H_2/2}\, e^{-i\tau H_1/2} \right\| \leqslant c_4\tau^5 + \mathcal{O}(\tau^6), \tag{A.25}$$

as claimed.

Appendix B. $C(T, U)$ and $C(K_O, K_E)$

As the expressions of $C(T, U)$ and $C(K_O, K_E)$ (and generalizations of it) have been obtained by straightforward calculation of commutators only the final results are given. The kinetic energy for a particle moving on a d-dimensional hypercubic lattice Λ reads

$$T = V \sum_{n\in\Lambda} \sum_{e} t_{n,n+e}(c_n^+ c_{n+e} + c_{n+e}^+ c_n), \tag{B.1}$$

where the sum over n runs over all lattice points. The sum over e goes over all unit vectors and the site-dependent hopping energy $t_{n,n'}$ has been introduced to facilitate taking into account boundary conditions. For free-boundary conditions, $t_{n,n'} = 0$ if $n \notin \Lambda$ or $n' \notin \Lambda$ and $t_{n,n'} = 1$ if $n, j' \in \Lambda$.

The potential energy is given by

$$U = W \sum_{n\in\Lambda} \epsilon_n n_n. \tag{B.2}$$

The creation (annihilation) operators $c_n^+ (c_n)$ and the occupation-number operators n_n satisfy the commutation relations

$$[c_n^+, n_j] = -\delta_{nj} c_n^+; \quad [c_n, n_j] = \delta_{nj} c_n. \tag{B.3}$$

To calculate $C(T, U) = [T + 2U, [T, U]]/24$ we need

$$[U, [T, U]] = -VW^2 \sum_{n\in\Lambda} \sum_{e} t_{n,n+e}(\epsilon_{n+e} - \epsilon_n)^2 (c_n^+ c_{n+e} + c_{n+e}^+ c_n), \tag{B.4a}$$

and

$$\begin{aligned}[T, [T, U]] = {} & +2V^2 W \sum_{n\in\Lambda} \sum_{e} \left[t_{n,n-e}^2(\epsilon_n - \epsilon_{n-e}) + t_{n,n+e}^2(\epsilon_n - \epsilon_{n+e}) \right] n_n \\ & + V^2 W \sum_{n\in\Lambda} \sum_{e} t_{n,n+e} t_{n+e,n+2e}(\epsilon_{n+2e} + \epsilon_n - 2\epsilon_{n+e})(c_n^+ c_{n+2e} + c_{n+2e}^+ c_n) \\ & + V^2 W \sum_{n\in\Lambda} \sum_{e>e'} t_{n,n-e'} t_{n-e',n+e-e'}(\epsilon_{n+e-e'} + \epsilon_n - 2\epsilon_{n-e'}) \\ & \times (c_n^+ c_{n+e-e'} + c_{n+e-e'}^+ c_n)\end{aligned}$$

$$+V^2W\sum_{n\in\Lambda}\sum_{e>e'}t_{n,n+e}t_{n+e,n+e-e'}(\epsilon_{n+e-e'}+\epsilon_n-2\epsilon_{n+e})$$

$$\times(c_n^+c_{n+e-e'}+c_{n+e-e'}^+c_n)$$

$$+V^2W\sum_{n\in\Lambda}\sum_{e>e'}[t_{n,n+e}t_{n+e,n+e+e'}(\epsilon_{n+e+e'}+\epsilon_n-2\epsilon_{n+e})$$

$$+t_{n,n+e'}t_{n+e',n+e+e'}(\epsilon_{n+e+e'}+\epsilon_n-2\epsilon_{n+e'})]$$

$$\times(c_n^+c_{n+e+e'}+c_{n+e+e'}^+c_n). \tag{B.4b}$$

Although admittedly complicated because of the presence of all the t's, employing (B.4) is actual applications is much easier as might be expected at first sight. The reason is that in the course of computing matrix elements of $e^{i\tau^3C(T,U)}$, crossing the boundaries of the lattice reflects itself directly in values of array indices going out of bounds.

As explained in chapter 2, to approximate the kinetic-energy propagator of a d-dimensional system by a real-space, fourth-order SPF it is sufficient is consider the 1D case only. The kinetic energy of the 1D system with free-boundary conditions reads

$$K=V\sum_{l=1}^{L}(c_l^+c_{l+1}+c_{l+1}^+c_l). \tag{B.5}$$

For purely technical reasons we assume that the number of lattice sites is odd or, in other words that L is even. As discussed earlier, K is decomposed as $K=K_O+K_E$ with

$$K_O=V\sum_{l=0}^{L/2-1}(c_{2l+1}^+c_{2l+2}+c_{2l+2}^+c_{2l+1}), \tag{B.6}$$

and

$$K_E=V\sum_{l=0}^{L/2-1}(c_{2l+2}^+c_{2l+3}+c_{2l+3}^+c_{2l+2}). \tag{B.7}$$

After some algebra, one finds

$$[K_O,[K_O,K_E]]=-2V^3\sum_{l=0}^{L/2-2}(c_{2l+1}^+c_{2l+4}+c_{2l+4}^+c_{2l+1})$$

$$+2V^3\sum_{l=0}^{L/2-1}(c_{2l+2}^+c_{2l+3}+c_{2l+3}^+c_{2l+2})-V^3(c_L^+c_{L+1}+c_{L+1}^+c_L), \tag{B.8a}$$

and

$$[K_E, [K_O, K_E]] = +2V^3 \sum_{l=1}^{L/2-1} (c_{2l}^+ c_{2l+3} + c_{2l+3}^+ c_{2l})$$

$$-2V^3 \sum_{l=0}^{L/2-1} (c_{2l+1}^+ c_{2l+2} + c_{2l+2}^+ c_{2l+1}) + V^3(c_1^+ c_2 + c_2^+ c_1), \tag{B.8b}$$

from which $C(K_O, K_E)$ follows directly.

In some applications, notably continuum problems for which the three-point approximation to the Laplacian may not be adequate, a more complicated expression for the kinetic energy T will appear. Most frequently T will take the form

$$T = \sum_{n\in\Lambda} \sum_{e\in Q} b_e(c_n^+ c_{n+e} + c_{n+e}^+ c_n), \tag{B.9}$$

where b_e are known constants and Q is a set of vectors connecting two mesh points. For instance if $d = 1$ and $Q = 2$, the sum in (B.9) runs over nearest and next-nearest neighbors. With an appropriate choice of the b_e, T would then represent a five-point approximation to $\partial^2/\partial x^2$, the trivial diagonal term not being taken into account.

In the case of periodic-boundary conditions the general expression of $C(T, U)$ is mostly easily obtained by first transforming T and U to Fourier space, working out the commutators and transforming back. One finds

$$[U, [T, U]] = -W^2 \sum_{n\in\Lambda} \sum_{e\in Q} b_e(\epsilon_{n+e} - \epsilon_n)^2(c_n^+ c_{n+e} + c_{n+e}^+ c_n), \tag{B.10a}$$

$$[T, [T, U]] = +2W \sum_{n\in\Lambda} \sum_{e\in Q} b_e^2(2\epsilon_n - \epsilon_{n-e} - \epsilon_{n+e}) n_n$$

$$+ W \sum_{n\in\Lambda} \sum_{e\in Q} b_e^2(\epsilon_{n+2e} + \epsilon_n - 2\epsilon_{n+e})(c_n^+ c_{n+2e} + c_{n+2e}^+ c_n)$$

$$+2W \sum_{n\in\Lambda} \sum_{\substack{e,e'\in Q \\ e>e'}} b_e b_{e'}(\epsilon_{n+e+e'} - \epsilon_{n+e} - \epsilon_{n+e'} + \epsilon_n)(c_n^+ c_{n+e+e'} + c_{n+e+e'}^+ c_n)$$

$$+2W \sum_{n\in\Lambda} \sum_{\substack{e,e'\in Q \\ e>e'}} b_e b_e'(\epsilon_{n+e-e'} - \epsilon_{n+e} - \epsilon_{n-e'} + \epsilon_n)(c_n^+ c_{n+e-e'} + c_{n+e-e'}^+ c_n), \tag{B.10b}$$

where " > " stands for an arbitrary ordering on the elements of Q.

To evaluate the expressions that appear in the fourth-order formula of the free-particle propagator proceed as follows. As before, first use the fact that $T \equiv \Sigma_{e\in Q} T_e$ and $[T_e, T_{e'}] = 0$ to reduce the problem to the calculation of $C(T_{O_e}, T_{E_e})$. We have added subscripts to the sets O and

E to indicate that in general they depend on the choice of e. For instance if $d=1$ and $e=2$, $O_2=\{1, 2, 5, 6, \ldots\}$ and $E_2=\{3, 4, 7, 8, \ldots\}$. Note that since we must have $O_e \cup E_e = \Lambda$ and $O_e \cap E_e = \varnothing$, there will be some minor restrictions on the linear size of the lattice, as can be seen in the example above where the number of lattice sites needs to be a multiple of four. Keeping this in mind, a straightforward calculation yields

$$\left[T_{O_e}, \left[T_{O_e}, T_{E_e}\right]\right] = +2b_e \sum_{n \in O_e} (c_n^+ c_{n+e} + c_{n+e}^+ c_n) - 2b_e \sum_{n \in O_e} (c_n^+ c_{n+3e} + c_{n+3e}^+ c_n), \tag{B.11a}$$

and

$$\left[T_{E_e}, \left[T_{O_e}, T_{E_e}\right]\right] = -2b_e \sum_{n \in E_e} (c_n^+ c_{n+e} + c_{n+e}^+ c_n) + 2b_e \sum_{n \in E_e} (c_n^+ c_{n+3e} + c_{n+3e}^+ c_n), \tag{B.11b}$$

whereby it has been assumed that periodic-boundary conditions are taken into account. Remark that as a direct consequence of this $C(T_{O_e}, T_{E_e})$ will vanish if e and the shape of the lattice are such that for all n, lattice site $n+4e$ coincides with site n. Obviously the corresponding second-order formula then becomes exact. This is a generalization of a result obtained earlier [26]. For example if $d=1$, $L=8$ and $e=2$, then $C(T_{O_e}, T_{E_e})=0$.

References

[1] R. Kosloff and D. Kosloff, J. Chem. Phys. 79 (1983) 1823.
[2] M. Horbatsch, J. Phys. B 17 (1984) 2591.
[3] H. Flocard, S.E. Koonin and M.S. Weiss, Phys. Rev. C 17 (1978) 1682.
[4] N. Grün, A. Müllhans and W. Scheid, J. Phys. B 15 (1982) 4043.
[5] J.W. Negele, Rev. Mod. Phys. 54 (1982) 913.
[6] R. Kosloff and C. Cerjan, J. Chem. Phys. 81 (1984) 3722.
[7] H. Tal-Ezer and R. Kosloff, J. Chem. Phys. 81 (1984) 3967.
[8] J.C.H. Spence, Experimental High-Resolution Electron Microscopy (Clarendon, Oxford, 1981).
[9] D. Van Dyck, Advances in Electronics and Electron Physics 65 (Academic New York, 1985) p. 295.
[10] M.D. Feit and J.A. Fleck Jr., Appl. Opt. 17 (1978) 3990.
[11] M.D. Feit and J.A. Fleck Jr., Appl. Opt. 18 (1979) 2843.
[12] M.D. Feit and J.A. Fleck Jr., Appl. Opt. 19 (1980) 1154.
[13] M.D. Feit and J.A. Fleck Jr., Appl. Opt. 19 (1980) 2240.
[14] M.D. Feit and J.A. Fleck Jr., Appl. Opt. 19 (1980) 3140.
[15] R. Alben, M. Blume, H. Krakauer and L. Schwartz, Phys. Rev. B 12 (1975) 4090.
[16] P. Prelovšek, Phys. Rev. Lett. 40 (1978) 1596.
[17] P. Prelovšek, Phys. Rev. B18 (1978) 3657.
[18] B. Kramer, A. MacKinnon and D. Weaire, Phys. Rev. B 23 (1981) 6357.
[19] G.D. Smith, Numerical solution of partial differential equations (Clarendon, Oxford, 1985).
[20] A. Goldberg, H.M. Schey and J.I. Schwartz, Am. J. Phys. 35 (1967) 177.
[21] S.E. Koonin, Computational Physics (Benjamin-Cummings, Melo Park California, 1986).
[22] A. Askar and A.S. Cakmak, J. Chem. Phys. 68 (1978) 2794.
[23] R.J. Ribin, J. Chem. Phys. 70 (1979) 4811.
[24] D. Kosloff and R. Kosloff, J. Comp. Phys. 52 (1983) 35.
[25] S. Lie and F. Engel, Theorie der Transformationsgruppen (Teubner, Leipzig, 1888).

[26] H.F. Trotter, Proc. Am. Math. Soc. 10 (1959) 545.
[27] M. Suzuki, Prog. Theor. Phys. 56 (1976) 1454.
[28] M. Suzuki, Commun. Math. Phys. 51 (1976) 183.
[29] M. Suzuki, Commun. Math. Phys. 57 (1977) 193.
[30] A.J. Chorin, T.J.R. Hughes, M.F. McCracken and J.E. Marsden, Communications on Pure and Applied Mathematics 31 (Wiley, New York, 1978) p. 205.
[31] H. De Raedt and A. Lagendijk, Phys. Rep. 127 (1985) 233.
[32] M. Suzuki, J. Stat. Phys. 43 (1986) 833.
[33] M. Suzuki, Progress in Quantum Field Theory, eds. H. Ezawa and S. Kamefuchi, (Elsevier, Amsterdam, 1986).
[34] H. DeRaedt and B. De Raedt, Phys. Rev. A 28 (1983) 3575.
[35] M. Suzuki, J. Math. Phys. 26 (1985) 601.
[36] K. Fujiwara, J. Phys. Soc. Japan 14 (1961) 1513.
[37] J.M. Cowley and A.F. Moodie, Proc. Phys. Soc. 70 (1957) 486.
[38] P. Goodman and A.F. Moodie, Acta Crystallogr. Sec. A 30 (1974) 280.
[39] H. De Raedt, Europhys. Lett. 3 (1987) 139.
[40] M. Barma and B.S. Shastry, Phys. Lett. 61A (1977) 15.
[41] M.D. Feit, J.A. Fleck Jr. and A. Steiger, J. Comput. Phys. 47 (1982) 412.
[42] P.W. Anderson, Phys. Rev. 109 (1958) 1493.
[43] M. Az'bel, Sov. Phys. Jetp. 19 (1966) 634.
[44] S. Aubry, Solid State Sci. 8 (1978) 264.
[45] M. Az'bel, Phys. Rev. Lett. 43 (1979) 1954.
[46] G. André and S. Aubry, Ann. Israel Phys. Soc. 3 (1980) 133.
[47] B. Simon, Advan. Appl. Math. 3 (1982) 463.
[48] J.B. Sokoloff, Phys. Rep. 126 (1985) 189.
[49] E.H. Lieb and D.C. Mattis, Mathematical Physics in One Dimension, (Academic, New York, 1966).
[50] S. John, H. Sompolinsky and M.J. Stephen, Phys. Rev. B 21 (1983) 5592.
[51] E. Akkermans and R. Maynard, Phys. Rev. B 32 (1984) 7850.
[52] J. Canisius and J.L. Van Hemmen, J. Phys. C 18 (1985) 4873.
[53] C.A. Condat and T.R. Kirkpatrick, Phys. Rev. B 32 (1985) 495.
[54] M.P. Van Albada and A. Lagendijk, Phys. Rev. Lett. 54 (1985) 2242.
[55] P.E. Wolf and G. Maret, Phys. Rev. Lett. 55 (1985) 2696.
[56] S. Etemad, R. Thompson and M.J. Andrejco, Phys. Rev. Lett. 57 (1986) 575.
[57] M. Az'bel, Phys. Rev. B 28 (1983) 4106.
[58] S. Jon, Phys. Rev. Lett. 53 (1984) 2169.
[59] P.W. Anderson, Philos. Mag. B 52 (1985) 505.
[60] For an extensive list of references on the Aubry and related models see ref. [42]..
[61] I thank S. Das Sarma for bringing this to may attention.
[62] A. MacKinnon, J. Phys. C 13 (1980) L1031.
[63] P. Erdös and R.C. Herndon, Adv. Phys. 31 (1982) 65.
[64] T. Schneider, M.P. Soerensen, A. Politi and M. Zannetti, Phys. Rev. Lett. 56 (1986) 2341.
[65] T. Scheider, A. Politi and D. Würtz, Z. Phys. B 66 (1987) 469.
[66] K.S. Dy and T.C. Ma, J. Phys. C 15 (1982) 6971.
[67] D.J. Thouless, Phys. Rep. 13 (1974) 93.
[68] N.F. Mott and E.A. Davis, Electronic Processes in Non-Crystalline Materials, (Clarendon, Oxford, 1979).
[69] Y. Nagaoka and H. Fukayama, Anderson Localization, eds. Y. Nagaoka, and H. Fukayama (Springer, Berlin, 1982).
[70] B. Kramer, G. Bergmann and Y. Bruynseraede, Localization, Interaction, and Transport Phenomena, eds. B. Kramer, G. Bergmann and Y. Bruynseraede (Springer, Berlin, 1984).
[71] P.A. Lee and T.V. Ramakrishnan, Rev. Mod. Phys. 57 (1985) 287.
[72] A. Möbius, J. Phys. C 19 (1986) L147.
[73] W. Götze, Solid State Commun. 27 (1978) 1393.
[74] W. Götze, J. Phys. C 12 (1979) 1279.

[75] W. Götze, P. Prelovšek and P. Wölfle, Solid State Commun. 30 (1979) 369.
[76] D. Vollhardt and P. Wölfle, Phys. Rev. Lett. 45 (1980) 842.
[77] D. Vollhardt and P. Wölfle, Phys. Rev. B 22 (1980) 4666.
[78] F. Wegner, Z. Phys. B 25 (1976) 327.
[79] F. Wegner, Z. Phys. B 35 (1979) 207.
[80] E. Abrahams, P.W. Anderson, D.C. Licciardello and T.V. Ramakrishnan, Phys. Rev. Lett. 42 (1979) 673.
[81] P.W. Anderson, D.J. Thouless, E. Abrahams and D.S. Fisher, Phys. Rev. B 22 (1980) 3519.
[82] K. Schönhammer and W. Brenig, Phys. Lett. A 42 (1973) 447.
[83] D. Weaire and V. Srivastava, Amorphous and Liquid Semiconductors, ed. W. Spear (CICL, Univ. of Edinburgh, 1977).
[84] S. Yoshino and M. Okazaki, J. Phys. Soc. Japan 43 (1977) 415.
[85] P.A. Lee, Phys. Rev. Lett. 42 (1979) 1492.
[86] J. Stein and U. Krey, Z. Phys. B 37 (1980).
[87] H. Scher, Ph.D. thesis, Harvard University, 1983.
[88] A. MacKinnon and B. Kramer, Z. Phys. B 53 (1983) 1.
[89] A.D. Zdetsis, C.M. Soukoulis, E.N. Economou and G.S. Grest, Phys. Rev. B 32 (1985) 7811.
[90] E.N. Economou, C.M. Soukoulis and A.D. Zdetsis, Phys. Rev. B 30 (1984) 1686.
[91] I am grateful to W. Götze for pointing out this to me.
[92] K. Ishizuka, and N. Uyeda, Acta Crystallogr. Sec. A 33 (1977) 740.
[93] E. Huntley, W.M. Pickering and A.S. Zinober, J. Comp. Phys. 27 (1978) 256.
[94] D. Van Dyck, J. Microsc. 119 (1980) 141.
[95] P.G. Self, J. Microsc. 127 (1982) 293.
[96] R. Kilaas and R. Gronsky, Ultramicroscopy 11 (1983) 289.
[97] W. Coene and D. Van Dyck, Ultramicroscopy 15 (1984) 41.
[98] W. Coene and D. Van Dyck, Ultramicroscopy 15 (1984) 287.
[99] J. Dudek, Z. Phys. A 292 (1979) 205.
[100] P. Bonche, H. Flocard, P.H. Heenen, S.J. Krieger and M.S. Weiss, Nucl. Phys. A 443 (1985) 39.
[101] D. Baye and P.H. Heenen, J. Phys. A 19 (1986) 2041.
[102] M. Abramowitz and I.A. Stegun, Handbook of Mathematical Functions, (Dover, New York, 1964).
[103] W. Coene, private communication.
[104] J.H. Wilkinson, The Algebraic Eigenvalue Problem, (Clarendon, Oxford, 1965).
[105] R. Blankenbecler and R.L. Sugar, Phys. Rev. D 27 (1983) 1304.
[106] H. Betsuyaku and T. Yokota, Prog. Theor. Phys. 75 (1986) 808.
[107] H. Betsuyaku and T. Yokota, Phys. Rev. B 33 (1986) 6505.
[108] D.M. Ceperley and M.H. Kalos, Monte Carlo Methods in Statistical Physics, ed. K. Binder (Springer, Berlin, 1979).
[109] K.E. Schmidt and M.H. Kalos, Applications of the Monte Carlo Methods in Statistical Physics, ed. K. Binder (Springer, Berlin, 1984).
[110] J.H. Hetherington, Phys. Rev. A 30 (1984) 2713.
[111] A. Weinstein and W. Stenger, Methods of Intermediate Problems for Eigenvalues (Academic, New York, 1972).
[112] T. Kato, Perturbation Theory for Linear Operators (Springer, Berlin, 1976).
[113] S. Shore and L.M. Sander, Phys. Rev. B 12 (1975) 1546.
[114] H.G. Reik, H. Nusser and L.A. Amarante Ribeiro, J. Phys. A 15 (1982) 3491.
[115] S. Benk and E. Sigmund, J. Phys. C 18 (1984) 533.
[116] H.G. Reik, N. Klenner and H. Nusser, J. Phys. A 18 (1985) 1697.
[117] C. Durst, E. Sigmund, P. Reineker and A. Scheuing, J. Phys. C 16 (1986) 2701.
[118] M. Vanhimbeeck, H. De Raedt and D. Schoemaker, Phys. Rev. B (Preprint).