Computer Physics Communications 39 (1986) 313–332
North-Holland, Amsterdam

MULTIDIMENSIONAL INTERPOLATION AND DIFFERENTIATION BASED ON AN ACCELERATED SINC INTERPOLATION PROCEDURE

R.H. BISSELING, R. KOSLOFF

Department of Physical Chemistry and The Fritz Haber Research Center for Molecular Dynamics, The Hebrew University, Jerusalem 91904, Israel

and

D. KOSLOFF

Department of Geophysics and Planetary Science, Tel Aviv University, Tel Aviv 69978, Israel

Received 17 July 1985

A multidimensional interpolator based on convolution with the sinc function is developed. The interpolator is global in nature, with all sampled data contributing to the computation. Preprocessing of the data by partial summation accelerates convergence of the actual interpolation, when repeatedly interpolating the same data. Modification of the interpolation procedure gives an efficient method for numerical differentiation. The method is intended for bandlimited functions with finite support. The interpolator was tested for a class of problems related to molecular dynamics, including interpolation of 1D, 2D and 3D Gaussian wave packets on a grid; integration of classical trajectories on an interpolated potential; and the transfer of a sampled wave function from a polar grid to a Cartesian grid and back. It was found that the interpolator is very accurate for Gaussian-like wave functions, and that even for functions which are not bandlimited, such as the Morse potential, a reasonable accuracy can be obtained.

1. Introduction

A wide class of physical and chemical problems requires not only an accurate but also a rapid interpolator. Data on functions is often gathered at discrete and equally spaced sampling points. By interpolation, the value of the function is retrieved at any arbitrary point. In many situations, the sampling interval between points is relatively large so that the task of generating an accurate interpolator is challenging. In such cases interpolation methods based on approximation by polynomials or piecewise polynomial functions like splines, break down because they fail to represent the shorter wavelengths of the function. In multidimensions, the shortcomings of polynomial type interpolators become more severe. This study is concerned with the construction of a multidimensional interpolator based on a global approach, which performs well on sparsely sampled data. The advantages of such a global approach are that all of the known values of the function on the sampling points are utilized to obtain the interpolated value.

The interpolation scheme chosen was based on a convolution with the sinc function [1,2]. Because of the slow decay of this function a partial summation procedure was used [3] to accelerate the convergence of the interpolator. A multidimensional interpolator was then developed. An extension of this interpolation will give an option of differentiation.

The present study was initiated in order to be able to transform a quantum mechanical wave function represented on a Cartesian grid to a polar grid [4]. Because this required extensive interpolation, a rapid procedure was needed.

0010-4655/86/$03.50

The development of the interpolation procedure opened new applications, such as simulation of reactive scattering by the method of classical trajectories. In this simulation the input needed for integrating the equations of motion consisted of the potential energy surface and its derivatives. In many cases this potential is supplied on grid points. An accurate and efficient procedure is then required to find the derivatives between these sampling points.

In section 2 the details of the sinc method for interpolation and differentiation are described. In section 3 examples of the use of the sinc method are given, such as the interpolation of Gaussian wave packets; the direct calculation of a classical trajectory by interpolation of the value of the potential and its derivatives; and the transfer of a quantum mechanical wave function from a Cartesian grid to a polar grid. In section 4 the conclusions are presented.

2. Description of the method

2.1. The sinc interpolator

The Whittaker–Kotel'nikov–Shannon sampling theorem [1,2] states that a bandlimited function is fully specified if the function values are given in a discrete, sufficiently dense set of equally spaced sampling points. This implies that, given the values on such a set of points, the value in a point between the sampling points can be interpolated with any desired accuracy.

The *Fourier transform* of a function f: $R \to C$ is defined as the function $\hat{f}$: $R \to C$ with

$$\hat{f}(s) = \int_{-\infty}^{\infty} f(x)\, \mathrm{e}^{-2\pi i s x}\, \mathrm{d}x. \tag{2.1}$$

The function f is called *bandlimited* if there exists an s_0 such that $\hat{f}(s) = 0$ for $|s| > s_0$. The smallest such value s_0 is sometimes referred to as the *Nyquist* or *folding frequency*. If the function values $f(x)$ are known in points $x = n\mathrm{d}x$ with sampling distance $\mathrm{d}x \leq 1/2s_0$ then, according to the sampling theorem,

$$f(t\,\mathrm{d}x) = \sum_{n=-\infty}^{\infty} f(n\,\mathrm{d}x)\, \mathrm{sinc}(t-n), \tag{2.2}$$

where the function *sinc* is defined as $\mathrm{sinc}(x) = \sin \pi x / \pi x$ for $x \neq 0$ and $\mathrm{sinc}(0) = 1$. For convenience the sampling points in this section are assumed to be at distance $\mathrm{d}x = 1$. In that case the previous expression is valid if $s_0 \leq \frac{1}{2}$. The expression can then be rewritten in the form

$$f(m+\delta) = \sum_{n=-\infty}^{\infty} f(n)\, \mathrm{sinc}(m+\delta-n) = \sum_{n=-\infty}^{\infty} f(n) \frac{\sin(\pi(m+\delta-n))}{\pi(m+\delta-n)}, \tag{2.3}$$

where m is an integer, and δ a real number in the closed segment [0,1]. The expression (2.3) is a convolution of f and the sinc function.

2.2. Accelerated convergence by partial summation

Summation by parts is the finite difference analogue of integration by parts of a definite integral and is expressed in the formula [3]

$$\sum_{n=0}^{N-1} F(n)\Delta G(n) = [F(N)G(N) - F(0)G(0)] - \sum_{n=0}^{N-1} G(n+1)\Delta F(n), \tag{2.4}$$

where the finite difference operator Δ is defined by $\Delta F(n) = F(n+1) - F(n)$.

Lanczos [5] and, more recently, Rosenbaum and Boudreaux [6] have used partial summation to speed up convergence in a spectral interpolation formula. The treatment of interpolation in this paper is similar to their work except that in the present case interpolation is done in the time domain and not in the frequency domain.

Application of partial summation, eq. (2.4), to accelerate convergence in eq. (2.3) is done as follows: Suppose f is a bandlimited function with a finite support, with $f(i) = 0$ for $i < 0$ and for i large enough. For a given fixed integer m and a fixed $\delta \in (0,1)$ define

$$F(n) = 1/(m - n + \delta) \tag{2.5}$$

and

$$G(n) = \sum_{i=0}^{n-1} (-1)^i f(i). \tag{2.6}$$

Then

$$\Delta F(n) = 1/(m - n + \delta)(m - n - 1 + \delta) \tag{2.7}$$

and

$$\Delta G(n) = (-1)^n f(n). \tag{2.8}$$

After simple algebraic manipulation eq. (2.3) can be rewritten as

$$f(m+\delta) = \sum_{n=0}^{\infty} f(n) \frac{\sin(\pi(m-n+\delta))}{\pi(m-n+\delta)} = (-1)^m \frac{\sin \pi\delta}{\pi} \sum_{n=0}^{\infty} \frac{(-1)^n f(n)}{m-n+\delta}$$

$$= (-1)^m \frac{\sin \pi\delta}{\pi} \sum_{n=0}^{\infty} F(n)\Delta G(n). \tag{2.9}$$

Consequently partial summation, according to eq. (2.4), can be applied with $N \to \infty$. The term $F(N)G(N)$ disappears because $\lim_{N\to\infty} F(N) = 0$ and $G(N)$ is a constant function for large enough N. Since $G(0) = 0$ the term $F(0)G(0) = 0$. As a result

$$f(m+\delta) = (-1)^{m+1} \frac{\sin \pi\delta}{\pi} \sum_{n=0}^{\infty} \frac{G(n+1)}{(m-n+\delta)(m-n-1+\delta)}, \tag{2.10}$$

where

$$G(n+1) = \sum_{i=0}^{n} (-1)^i f(i). \tag{2.11}$$

The infinite sum in eq. (2.10) converges faster than the original sum in eq. (2.3) because of the quadratic denominator.

In conclusion, an efficient interpolation of a function f at many points $m + \delta$ can be performed by preprocessing the sampled data to compute the alternating partial sums of eq. (2.11), storing the results and using them to compute $f(m + \delta)$ by eq. (2.10) for every point $m + \delta$.

Applying this method for interpolation near the grid points gives poor results, because of the singularities of eq. (2.10) for $\delta = 0$ and $\delta = 1$. Instead, use is made of a first order Taylor approximation $f(m + \delta) = f(m) + \delta f'(m)$ for $\delta \approx 0$, and $f(m + \delta) = f(m + 1) + (\delta - 1)f'(m + 1)$ for $\delta \approx 1$. The derivatives $f'(m)$ and $f'(m + 1)$ are computed by the method for differentiation on grid points described in subsection 2.5.

2.3. Double partial summation

Partial summation can be repeated a number of times to effect a further acceleration of convergence. As an example, the procedure of double partial summation will be outlined. The same procedure which speeded up convergence in eq. (2.3), resulting in eq. (2.10), can be applied again. Define

$$FF(n) = 1/(m-n+\delta)(m-n-1+\delta) \tag{2.12}$$

and

$$GG(n) = \sum_{i=0}^{n} G(i). \tag{2.13}$$

Then

$$\Delta FF(n) = 2/(m-n+\delta)(m-n-1+\delta)(m-n-2+\delta) \tag{2.14}$$

and

$$\Delta GG(n) = G(n+1). \tag{2.15}$$

Eq. (2.10) can now be rewritten as

$$f(m+\delta) = (-1)^{m+1}\frac{\sin \pi\delta}{\pi}\sum_{n=0}^{\infty} FF(n)\Delta GG(n). \tag{2.16}$$

Here partial summation, eq. (2.4), can be applied with $N \to \infty$. Note that $\lim_{N\to\infty} FF(N)GG(N) = 0$, since for large N the factor $GG(N)$ grows linearly with N and $FF(N)$ falls off quadratically. Since $GG(0) = 0$ the term $FF(0)GG(0) = 0$. As a result

$$f(m+\delta) = (-1)^{m}\frac{2\sin \pi\delta}{\pi}\sum_{n=0}^{\infty}\frac{GG(n+1)}{(m-n+\delta)(m-n-1+\delta)(m-n-2+\delta)}, \tag{2.17}$$

where

$$GG(n+1) = \sum_{i=0}^{n+1}\sum_{j=0}^{i-1}(-1)^{j} f(j). \tag{2.18}$$

At the initial cost of an additional calculation of partial sums, the convergence is speeded up by an extra factor of the order $1/(m-n)$.

The three equations (2.3), (2.10) and (2.17) have the form of a convolution of f with $\sin \pi x/\pi x$, resp. G with $\sin \pi x/\pi x(x-1)$, resp. GG with $\sin \pi x/\pi x(x-1)(x-2)$. These functions are shown in fig. 1.

2.4. Multidimensional interpolation

The one-dimensional sampling theorem can be generalized to higher dimensions. For a bandlimited function f of k variables, sampled at integer grid points $(n_1,\ldots,n_k)$, the sampling formula is, in analogy with eq. (2.3),

$$f(m_1+\delta_1,\ldots,m_k+\delta_k) = \sum_{n_1=-\infty}^{\infty}\cdots\sum_{n_k=-\infty}^{\infty} f(n_1,\ldots,n_k)\,\mathrm{sinc}(m_1+\delta_1-n_1)\cdots\mathrm{sinc}(m_k+\delta_k-n_k) \tag{2.19}$$

for integer $m_1,\ldots,m_k$ and $\delta_1,\ldots,\delta_k \in [0,1]$.

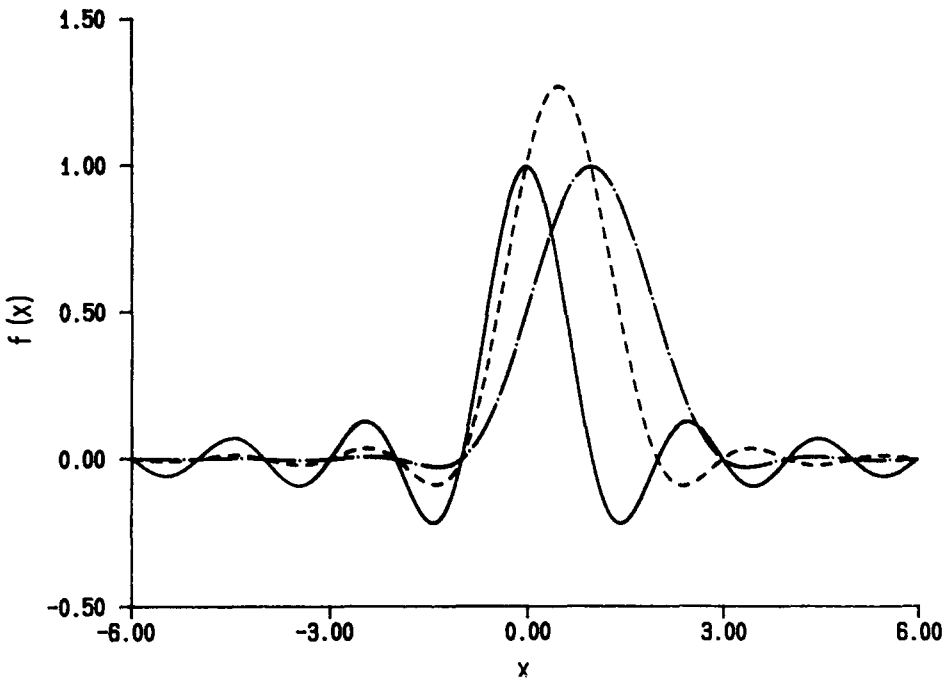

Fig. 1. The sinc function (solid line) and the functions $-\sin \pi x/\pi x(x-1)$ (dashed line) and $\sin \pi x/\pi x(x-1)(x-2)$ (dash-dotted line). Interpolation with 0, 1, 2, times acceleration is a convolution with these functions, respectively.

Suppose f has a finite support with $f(n_1,\ldots,n_k)=0$ if one of the n_j's is negative or large enough. Following the same procedure as in eq. (2.9), we rewrite eq. (2.19) as

$$f(m_1+\delta_1,\ldots,m_k+\delta_k) = C\sum_{n_1=0}^{\infty}\cdots\sum_{n_k=0}^{\infty}\frac{(-1)^{n_1+\cdots+n_k}f(n_1,\ldots,n_k)}{(m_1-n_1+\delta_1)\cdots(m_k-n_k+\delta_k)}$$
$$= C\sum_{n_1=0}^{\infty}\frac{(-1)^{n_1}}{(m_1-n_1+\delta_1)}\left(\cdots\sum_{n_k=0}^{\infty}\frac{(-1)^{n_k}f(n_1,\ldots,n_k)}{(m_k-n_k+\delta_k)}\right)\cdots\Bigg), \tag{2.20}$$

where the constant $C=(-1)^{m_1+\cdots+m_k}\pi^{-k}\sin\pi\delta_1\cdots\sin\pi\delta_k$. Applying partial summation for the kth sum gives

$$f(m_1+\delta_1,\ldots,m_k+\delta_k) = -C\sum_{n_1=0}^{\infty}\frac{(-1)^{n_1}}{(m_1-n_1+\delta_1)}$$
$$\times\left(\cdots\sum_{n_{k-1}=0}^{\infty}\frac{(-1)^{n_{k-1}}}{(m_{k-1}-n_{k-1}+\delta_{k-1})}\right.$$
$$\left.\times\left(\sum_{n_k=0}^{\infty}\frac{G_1(n_1,\ldots,n_k)}{(m_k-n_k+\delta_k)(m_k-n_k-1+\delta_k)}\right)\right)\cdots\Bigg), \tag{2.21}$$

where

$$G_1(n_1,\ldots,n_k)=\sum_{i_k=0}^{n_k}(-1)^{i_k}f(n_1,\ldots,n_{k-1},i_k) \tag{2.22}$$

and hence

$$f(m_1+\delta_1,\ldots,m_k+\delta_k) = -C\sum_{n_k=0}^{\infty}\frac{1}{(m_k-n_k+\delta_k)(m_k-n_k-1+\delta_k)}$$
$$\times\left[\left(\sum_{n_1=0}^{\infty}\frac{(-1)^{n_1}}{(m_1-n_1+\delta_1)}\left(\cdots\sum_{n_{k-1}=0}^{\infty}\frac{(-1)^{n_{k-1}}G_1(n_1,\ldots,n_k)}{(m_{k-1}-n_{k-1}+\delta_{k-1})}\right)\right)\cdots\right]. \tag{2.23}$$

The expression inside the square brackets of eq. (2.23) has the form of the right-hand-side of eq. (2.20) (with $k-1$ variables instead of k) and hence the procedure above can be repeated. After performing this procedure k times, defining successively functions $G_1, G_2,\ldots,G_k$, the interpolation formula obtained is

$$f(m_1+\delta_1,\ldots,m_k+\delta_k) = (-1)^k C\sum_{n_1=0}^{\infty}\cdots\sum_{n_k=0}^{\infty}\frac{1}{(m_1-n_1+\delta_1)(\bar{m}_1-n_1-1+\delta_1)}$$
$$\cdots\frac{1}{(m_k-n_k+\delta_k)(m_k-n_k-1+\delta_k)}G_k(n_1,\ldots,n_k), \tag{2.24}$$

where

$$G_k(n_1,\ldots,n_k) = \sum_{i_1=0}^{n_1}\cdots\sum_{i_k=0}^{n_k}(-1)^{i_1+\cdots+i_k}f(i_1,\ldots,i_k) \tag{2.25}$$

and

$$C = (-1)^{m_1+\cdots+m_k}\pi^{-k}\sin\pi\delta_1\cdots\sin\pi\delta_k. \tag{2.26}$$

Eqs. (2.24)–(2.26) are the multidimensional analogue of eqs. (2.10)–(2.11). As in the one-dimensional case, f is interpolated efficiently by first preparing alternating partial sums by eq. (2.25), storing them and using the results to compute $f(m_1+\delta_1,\ldots,m_k+\delta_k)$ by eq. (2.24) for the points $(m_1+\delta_1,\ldots,m_k+\delta_k)$.

2.5. *Differentiation*

Functions that can be interpolated efficiently by the sinc method can also be differentiated efficiently in a similar way. Two cases of differentiation will be considered in this subsection: (i) on grid points; (ii) between grid points.

The class of functions to be differentiated is the same as before: bandlimited functions with finite support and with function values that are known in equidistant grid points. Here the grid distance is supposed to be one, and f is supposed to be zero for $i<0$ or i large.

The differentiation formula is provided by differentiation of the sinc interpolation expression: Differentiation of eq. (2.3) gives

$$f'(m+\delta) = \sum_{n=-\infty}^{\infty} f(n)\,\mathrm{sinc}'(m-n+\delta), \tag{2.27}$$

where m is an integer, and δ a real number in the segment [0,1]. Here

$$\mathrm{sinc}'(x) = \frac{\cos\pi x}{x} - \frac{\sin\pi x}{\pi x^2} \quad \text{for} \quad x\neq 0 \tag{2.28}$$

and

$$\operatorname{sinc}'(0) = 0. \tag{2.29}$$

(i). The derivative of f in grid points m, $f'(m)$, is computed from the known values $f(n)$, as follows: Setting $\delta = 0$ in eq. (2.27) gives

$$\begin{aligned} f'(m) &= \sum_{n=-\infty}^{\infty} f(n) \operatorname{sinc}'(m-n) \\ &= \sum_{n \neq m} f(n) \left[\frac{\cos(\pi(m-n))}{m-n} - \frac{\sin(\pi(m-n))}{\pi(m-n)^2} \right] + f(m) \operatorname{sinc}'(0) \\ &= \sum_{n \neq m} f(n) \frac{(-1)^{m-n}}{m-n} = (-1)^m \sum_{\substack{n=0 \\ n \neq m}}^{\infty} \frac{(-1)^n f(n)}{m-n}. \end{aligned} \tag{2.30}$$

This expression has the by now familiar form which allows partial summation. Care has to be taken to sum over all $n \neq m$. This is done by defining

$$F(n) = \begin{cases} 1/(m-n) & \text{if } n \neq m, \\ 0 & \text{if } n = m, \end{cases} \tag{2.31}$$

and defining $G(n)$ as usual (eq. (2.6)). Then

$$\Delta F(n) = \frac{1}{(m-n)(m-n-1)} \quad \text{for} \quad n \neq m-1,\ m \tag{2.32}$$

and

$$\Delta F(m-1) = \Delta F(m) = -1. \tag{2.33}$$

Eq. (2.30) can now be rewritten as

$$\begin{aligned} f'(m) &= (-1)^m \sum_{n=0}^{\infty} F(n) \Delta G(n) = (-1)^{m+1} \sum_{n=0}^{\infty} G(n+1) \Delta F(n) \\ &= (-1)^{m+1} \left[\sum_{\substack{n=0 \\ n \neq m-1,\ m}}^{\infty} \frac{G(n+1)}{(m-n)(m-n-1)} - G(m) - G(m+1) \right], \end{aligned} \tag{2.34}$$

where

$$G(n+1) = \sum_{i=0}^{n} (-1)^i f(i). \tag{2.35}$$

An efficient differentiation of the function f at many points m can be performed by first computing the partial sums in eq. (2.35), storing the results and using them to compute $f'(m)$ by eq. (2.34) for every point m. As in the case of interpolation it is possible to repeat the acceleration procedure.

(ii). The function f can be differentiated in a point $m + \delta$ between the grid points n (i.e., with $\delta \neq 0,1$). One way to proceed is to combine eqs. (2.27) and (2.28) and then apply partial summation to speed up

convergence. Another, equivalent, way is to differentiate the accelerated result of partial summation, eq. (2.10). This gives

$$f'(m+\delta) = \frac{\partial}{\partial\delta}(-1)^{m+1}\frac{\sin\pi\delta}{\pi}\sum_{n=0}^{\infty}\frac{G(n+1)}{(m-n+\delta)(m-n-1+\delta)}$$
$$= (-1)^{m+1}\left[\cos\pi\delta\sum_{n=0}^{\infty}\frac{G(n+1)}{(m-n+\delta)(m-n-1+\delta)}\right.$$
$$\left.+\frac{\sin\pi\delta}{\pi}\sum_{n=0}^{\infty}\frac{G(n+1)(1-2(m-n+\delta))}{(m-n+\delta)^2(m-n-1+\delta)^2}\right], \tag{2.36}$$

where the partial sums $G(n)$ are defined as before, eq. (2.35).

This procedure can also be applied on the doubly accelerated formula eq. (2.17). This results in

$$f'(m+\delta) = (-1)^m\left[2\cos\pi\delta\sum_{n=0}^{\infty}\frac{GG(n+1)}{(m-n+\delta)(m-n-1+\delta)(m-n-2+\delta)}\right.$$
$$\left.-\frac{2\sin\pi\delta}{\pi}\sum_{n=0}^{\infty}\frac{GG(n+1)\big(3(m-n+\delta)^2-6(m-n+\delta)+2\big)}{(m-n+\delta)^2(m-n-1+\delta)^2(m-n-2+\delta)^2}\right], \tag{2.37}$$

where $GG(n+1)$ is defined in eq. (2.18).

For the special case of midpoint differentiation ($\delta = \frac{1}{2}$) the formulae (2.36) and (2.37) provide even faster differentiators. This is because $\cos\pi\delta = 0$ for $\delta = \frac{1}{2}$, leaving only the second sum in the formula, thereby speeding up convergence by an extra factor of the order $1/(m-n)$.

Note that instead of differentiating the accelerated interpolation formula eq. (2.10) we can integrate it, in a similar way, obtaining a formula for numerical integration of a function.

2.6. Generalization

The sinc interpolation formula, eq. (2.3), is a particular case of the formula given by Schwartz [7] for the interpolation of analytic functions.

$$f(x) = \sum_n f(x_n)\frac{u(x)}{(x-x_n)}\frac{1}{u'(x_n)}. \tag{2.38}$$

Here the analytic function f is sampled at the grid points x_n (not necessarily at equal distances) and it is approximated using an analytic function $u(x)$ with simple zeros at the grid points. Taking $u(x) = \sin\pi x$ and $x_n = n$ in eq. (2.38) reduces the expression to eq. (2.3).

The method of partial summation to accelerate convergence can also be applied in the more general case. This results in

$$f(x) = \sum_n G(n+1)\frac{u(x)}{(x-x_n)(x-x_{n+1})}, \tag{2.39}$$

where

$$G(n+1) = (x_n - x_{n+1})\sum_{i=0}^{n}\frac{f(x_i)}{u'(x_i)}. \tag{2.40}$$

The sampled data are preprocessed to calculate the values $G(n+1)$ and after that, in the actual interpolation, the expression (2.39) is evaluated.

2.7. Error analysis

It is an idealization to assume that a function is bandlimited and has a finite support. Since there exists an uncertainty relation $\Delta x \Delta s \geq 1/4\pi$, (cf. ref. [2], chap. 8) between the width of a Fourier integrable function and the width of its Fourier transform, both assumptions are only approximately correct. This gives rise to two kinds of errors, as outlined below.

The error analysis performed here is due to Schwartz [7], who gave the analysis for the non-accelerated case. There are two competing errors (neglecting round-off): the truncation error ϵ_T caused by assuming finite support and cutting off the infinite sums, and the approximation error ϵ_A caused by the assumption of bandlimitedness. Optimal efficiency is obtained when both errors are about equal.

The approximation error is equal to

$$\epsilon_A = \frac{1}{2\pi i} \int_\Gamma \frac{f(z)}{z-x} \frac{\sin \pi x}{\sin \pi z} \, dz, \tag{2.41}$$

(cf. [7]) where the contour Γ in the complex plane encloses all the grid points.

The effect of the acceleration procedure is to reduce the truncation error ϵ_T.

For Gaussian wave functions, $f(x) = e^{-ax^2}$, Schwartz found an exponential decrease of the error, as $e^{-\pi N/2}$, where for a given number of grid points N the grid distance dx was chosen such that $\epsilon_A \approx \epsilon_T$. The acceleration introduced here changes the relation between the two errors, by decreasing ϵ_T and increasing ϵ_A. For a given accuracy, acceleration decreases the computational effort.

3. Examples

3.1. Gaussian wave packets

The Gaussian wave packet [8] is an important example of a quantum mechanical wave function. It describes the probability of a free particle being present in a certain point of space at a given time. A normalized Gaussian wave packet in dimension one has the form

$$\psi_G(x) = \left(\frac{1}{2\pi(\Delta x)^2} \right)^{1/4} e^{ikx} e^{-(x/2\Delta x)^2}, \tag{3.1}$$

where x is the space coordinate, k the momentum, and Δx the width of the wave packet. Recently, a basis of Gaussian wave packets has been used by Heller et al. in semiclassical time dependent wave packet propagations with many degrees of freedom [9,10]. Gaussian wave packets have also been used in full quantum simulations of dynamical processes such as the $H^+ + H_2$ reaction [11], and atom–surface scattering for the He–W system [12].

The Gaussian wave packet possesses properties which permit successful application of sinc interpolation and differentiation. The Fourier transform of a Gaussian function is again a Gaussian. Since the tails of a Gaussian function fall off rapidly, both the Gaussian and its Fourier transform have, approximately, a finite support. This implies that in a good approximation Gaussian wave functions are bandlimited functions with a finite support. These are precisely the properties which enable the use of the sinc method.

Table 1
Accuracy and computation time of interpolation of the one-dimensional Gaussian wave packet $f(x) = e^{-x^2/2}$

Total number of grid points N	Number of points M used in the interpolation	Single acceleration		Double acceleration	
		maximum absolute error	CPU-time (ms)	maximum absolute error	CPU-time (ms)
16	16	0.147081	2.9	0.2117930	3.6
32	16	0.003602	3.4	0.0097140	4.2
	32	0.002436	5.3	0.0090885	6.8
64	16	0.001501	1.8	0.0002495	2.1
	32	0.000354	3.1	0.0000254	3.7
	64	0.000086	5.7	0.0000029	6.6
128	16	0.001424	1.7	0.0002341	2.0
	32	0.000336	2.9	0.0000239	3.6
	64	0.000081	5.9	0.0000027	6.7
	128	0.000020	10.9	0.0000003	13.1
256	16	0.001287	1.8	0.0002469	2.2
	32	0.000332	3.2	0.0000236	3.7
	64	0.000080	5.7	0.0000027	7.0
	128	0.000020	11.0	0.0000003	13.2
	256	0.000005	20.8	0.0000002	25.0

The accuracy and efficiency of the sinc method were checked by interpolating and differentiating one-, two- and three-dimensional Gaussian wave packets. For simplicity the momentum was chosen as $k = 0$ and the width as $\Delta x = 1/\sqrt{2}$, and the function was renormalized giving the form

$$\psi_G(x) = e^{-x^2/2} \tag{3.2}$$

in dimension one and, e.g.,

$$\psi_G(x, y, z) = e^{-(x^2+y^2+z^2)/2} \tag{3.3}$$

in dimension three.

Table 1 presents the results for interpolation of a one-dimensional Gaussian wave packet. The wave packet was entered into an equidistant grid with N grid points, numbered $0, 1, \ldots, N-1$, at distance dx. While N was varied, the length of the grid Ndx was kept constant at $Ndx = 32$. The wave packet was shifted and its peak was placed in the center of the grid. The wave function was then interpolated in a set of points $m + \delta$ ($m = 0, 1, \ldots, N-2$ and $\delta = 0.5$), and the results were compared with the exact values, giving the absolute errors. Comparison of the maximum absolute error, which is shown in the table, with the maximum function value, $f(0) = 1$, gives a measure of the accuracy of the interpolation. The CPU-time shown is the computation time necessary to perform an interpolation in one point $m + \delta$ on a VAX 11/750 computer. This time does not take into account the time needed for preparation of the partial sums (e.g. 8 ms for 256 points). Since this preparation is done only once, before the actual interpolations, its computation time is negligible.

Examining the results in table 1, it is clear that there are three causes of errors: undersampling, truncation of the sum and machine round-off.

For a small number of grid points (e.g. $N = 16$ or 32) the function is undersampled, since it has frequencies outside the band represented on the grid. In this range, the error ϵ_A (cf. subsection 2.7) is

Table 2
Accuracy and efficiency of differentiation of the one-dimensional Gaussian wave packet $f(x) = e^{-x^2/2}$. The computation is twice accelerated by partial summation

Total number of grid points N	Number of points M used in the differentiation	$\delta = 0$ maximum absolute error	$\delta = 0.25$ maximum absolute error	$\delta = 0.25$ CPU-time (ms)	$\delta = 0.5$ maximum absolute error
64	16	0.0019563	0.0010617	4.0	0.0002132
	32	0.0001767	0.0001110	6.9	0.0000102
	64	0.0000189	0.0000126	12.8	0.0000006
128	16	0.0036695	0.0019926	3.9	0.0003991
	32	0.0003329	0.0002092	6.8	0.0000191
	64	0.0000358	0.0000239	12.5	0.0000011
	128	0.0000043	0.0000040	24.4	0.0000003
256	16	0.0075659	0.0042253	4.1	0.0007535
	32	0.0006576	0.0004137	7.1	0.0000378
	64	0.0000709	0.0000474	13.0	0.0000021
	128	0.0000083	0.0000075	24.3	0.0000007
	256	0.0000015	0.0000075	48.2	0.0000007

dominant. Note that an extra acceleration *increases* the total error. This is explained by the fact that $\epsilon_T \ll \epsilon_A$, and hence the decrease in ϵ_T caused by extra partial summation is irrelevant. On the other hand, the undersampling error ϵ_A is enlarged, because of the accumulation of undersampling errors in each partial sum term. This effect sets a limit to the number of partial accelerations that is worthwhile to perform for each given set of data.

If N is large enough ($N \geq 64$), increasing N further does not improve accuracy. Accuracy for such N is mainly determined by the number M of terms that are included in the computation of the sum of eq. (2.10), resp. (2.17), and by the number of accelerations applied. In this range the error ϵ_T is dominant. Doubly accelerated interpolation, eq. (2.17), is much more accurate than singly accelerated interpolation, eq. (2.10). The error approximately decreases with M as M^{-2} (single acceleration), M^{-3} (double acceleration), respectively.

When the error becomes very small the precision of the computer (0.6×10^{-7}) becomes the dominant factor. The error is then determined by the accumulation of round-off errors.

The CPU-time grows linearly with M, with not more than 20% additional CPU time for double acceleration.

Table 2 shows the results for doubly accelerated differentiation of a one-dimensional Gaussian wave packet. The wave function was entered in the grid as above. The function was differentiated in a set of points $m + \delta (m = 0, 1, \ldots, N-2)$ for three different cases: in grid points ($\delta = 0$), in midpoints ($\delta = 0.5$), and in points which are neither grid points nor midpoints (e.g. $\delta = 0.25$). The maximum absolute error given in the table should be compared with the maximum value of the derivative $|f'(\pm 1)| \approx 0.607$.

The differentiation in midpoints is clearly very accurate, its error decreasing rapidly with M as M^{-4}, until the accuracy limit of the computer is reached. The error in the other cases ($\delta = 0$ and $\delta = 0.25$) grows with M as M^{-3}. The error in all three cases grows proportionally with N. This can be explained as follows: Formulae (2.34), (2.36), (2.37) are valid for grid distance one. For grid distance $\mathrm{d}x$ the expressions on the right-hand-side should be divided by $\mathrm{d}x$. This also divides the error by $\mathrm{d}x$, or multiplies it with a factor proportional to N, since $N\,\mathrm{d}x$ was kept constant.

The table shows the CPU-time needed for differentiation in the case $\delta = 0.25$ which represents the general case. The special case of midpoint differentiation is much more accurate than the general case, and

Table 3
Accuracy and efficiency of interpolation of the two-dimensional Gaussian wave packet $f(x, y) = e^{-(x^2+y^2)/2}$. Interpolation is performed in midpoints, i.e., points $(m_1 + \delta_1, m_2 + \delta_2)$ exactly in between grid points ($\delta_1 = \delta_2 = 0.5$). The computation is accelerated once by partial summation

Total number of grid points $N = N_1 \times N_2$	Number of points $M = M_1 \times M_2$ used in the interpolation	Maximum absolute error	CPU-time (ms)
64×64	8×8	0.006807	9
	16×16	0.001455	33
	32×32	0.000343	121
	32×64	0.000343	382
	64×64	0.000083	462
128×128	8×8	0.008806	9
	16×16	0.001449	32
	32×32	0.000333	124
	64×64	0.000081	476

it needs about 25% less CPU-time, because only one sum has to be computed. The differentiation on grid points is somewhat less accurate than the general case but its computation is about 50% faster, because of the simpler code.

Tables 3 and 4 show the results for singly accelerated interpolation of a two- and three-dimensional Gaussian wave packet. The interpolation was performed in conditions similar to those in table 1. The length and width (and height) of the grid were kept constant at $N_1\,dx = N_2\,dy\ (= N_3\,dz) = 32$. The accuracy was obtained by comparing the maximum absolute error with the maximum of the wave function $f(0, 0) = 1$ and $f(0, 0, 0) = 1$, respectively.

Examining the results, it is found that the accuracy obtained is approximately the same as in the equivalent one-dimensional case (cf. table 1). The results for an $N \times N$ two-dimensional grid using $M \times M$-point interpolation are very similar to the results for an N-point one-dimensional grid using M-point interpolation. The error is determined by the maximum of the errors in each direction (e.g., compare in table 3 the $M = 32 \times 32$ case with the $M = 32 \times 64$ case). The error decreases as $(\min(M_i))^{-2}$. The CPU-time is proportional to M, the total number of points used in the interpolation. Each additional dimension multiplies the CPU cost with a constant factor of about 1.4.

Table 4
Accuracy and efficiency of interpolation of the three-dimensional Gaussian wave packet $f(x, y, z) = e^{-(x^2+y^2+z^2)/2}$. Interpolation is performed in midpoints, i.e., points $(m_1 + \delta_1, m_2 + \delta_2, m_3 + \delta_3)$ exactly in between grid points ($\delta_1 = \delta_2 = \delta_3 = 0.5$). The computation is accelerated once by partial summation

Total number of grid points $N = N_1 \times N_2 \times N_3$	Number of points $M = M_1 \times M_2 \times M_3$ used in the interpolation	Maximum absolute error	CPU-time (ms)
$16 \times 16 \times 16$	$8 \times 8 \times 8$	0.1915	84
	$16 \times 16 \times 16$	0.1522	643
$32 \times 32 \times 32$	$8 \times 8 \times 8$	0.0075	86
	$16 \times 16 \times 16$	0.0042	548

3.2. Classical trajectories

The classical trajectory method is one of the most frequently used simulations of elementary molecular dynamics [13,14]. In this method the coordinates and momenta of the atoms involved are integrated by using Hamilton's equations of motion. In order to apply the method the potential energy surface has to be supplied. The most accurate potentials are those supplied by ab initio calculations. If they are not available semi-empirical potentials are used.

Ab initio calculations provide the value of the potential and its derivatives at discrete grid points. Present trajectory methods fit this data to a continuous functional form in order to be able to integrate the trajectories. This fitting procedure is difficult and is a main source of error. Some use of spline interpolation for these potentials has been reported but the discontinuity of high order derivatives in this procedure causes trouble in the use of high order integration procedures. Semi-empirical potentials may also become numerically expensive to use in obtaining the values of the potential and its derivatives at the integration points. For example, the DIM potential [15] requires a matrix diagonalization procedure each time the potential is calculated.

In situations when a large batch of trajectories is needed, computation time can be saved by calculating and storing the potential on grid points, for later use when integrating the trajectories.

Two methods exist for obtaining the derivatives of the potential needed for integrating the trajectory, application of which depends on the existence of the potential derivatives on the sampling points. If these derivatives are supplied one can use the interpolator to obtain the values of the derivatives along the trajectory. If only the value of the potential is supplied the interpolator can be used to supply numerical derivatives on the trajectory path.

In general the potential energy function is a difficult function to interpolate because of its strong singularities arising when the interatomic distances become small. Nevertheless, it will be shown that this problem can be overcome.

The equations of motion of the classical trajectory are Hamilton's equations:

$$\frac{\partial q}{\partial t} = \frac{\partial H}{\partial p} = \frac{p}{m}, \quad \frac{\partial p}{\partial t} = -\frac{\partial H}{\partial q} = -\frac{\partial V}{\partial q}. \tag{3.4}$$

A fourth order Runge–Kutta fixed step integrator was used to integrate these equations.

As a first demonstration the Morse oscillator was used to show the various applications of the interpolating procedure for classical trajectories.

The Morse potential is

$$V(q) = D(1 - \mathrm{e}^{-\alpha q})^2 - D, \tag{3.5}$$

where α and D are constants. The analytical derivative of the Morse potential is

$$\partial V/\partial q = 2D\alpha(1 - \mathrm{e}^{-\alpha q})\,\mathrm{e}^{-\alpha q}. \tag{3.6}$$

First, this exact derivative was used for the integration. The time step was chosen so that the analytical solution [16] for the amplitude coincided with the integrated solution up to six digits for 1000 integration steps. This result was then used to check the interpolation.

The Morse potential grows exponentially for negative q and becomes asymptotically zero for large positive q. Because of this exponential growth the Morse potential is not a bandlimited function with a finite support. This problem was solved by starting the grid at positive q where the potential is small. Starting at negative q causes the propagation of large errors through the grid by the partial summation procedure.

Table 5
Classical trajectory of the one-dimensional Morse oscillator. The parameters are $D = 0.1$, $\alpha = 1.0$, initial momentum $p(0) = 0$, initial amplitude $q(0) = -0.6$, and mass $m = 1.0$. The total number of grid points is $N = 800$

	Number of points M used in interpolation or differentiation	Average error of interpolation or differentiation	Relative error after k integration steps	
			$k = 100$	$k = 1000$
Interpolation	15	0.00017	0.00012	0.014
	50	0.0000033	0.0000061	0.00037
Differentiation	15	0.11	0.028	1.7
	50	0.00026	0.0019	0.0044

For the calculation a grid of $N = 800$ points was used with $\Delta q = 0.0045$, extending from $q = -1.2$ to $q = 2.4$. On this grid the potential and its derivative were calculated on the grid points and stored for interpolation. The trajectory was then integrated using both the interpolated derivative (cf. subsection 2.3) and the numerical derivative (cf. subsection 2.5). A double partial summation was used for both cases.

Table 5 compares the results of the different methods. Fig. 2 displays the amplitude of the Morse oscillator obtained by the different methods. Examining fig. 2 and table 5 one finds that the interpolation procedure for obtaining the derivatives converges more rapidly than the differentiation procedure. The results show that it is possible to integrate with sufficient accuracy classical trajectories for many integration steps, despite the fact that the Morse potential is not bandlimited and despite the fact that errors in classical trajectory integration accumulate.

A more realistic example is the integration of the molecular system of H_3^+. This integration is a three body three-dimensional calculation. By using conservation of momentum in the center of mass system the remaining problem consists of a 12-dimensional phase space. Since the molecular forces and the potential are functions of the three interatomic distances, a three-dimensional interpolation procedure was needed. For this system the DIM potential [15] was chosen. The same Runge–Kutta integrator as above was used with a time step of 0.1 atomic time units.

First, a trajectory was run with the original DIM potential and afterwards it was compared to trajectories in which the derivatives were calculated by interpolation. Three derivatives were needed and these were calculated and stored on a three-dimensional grid for later interpolation. In order to obtain a more bandlimited function for the interpolation, the derivatives of the potential were multiplied by a *taper function* T:

$$\frac{\partial V'(R_1, R_2, R_3)}{\partial R_i} = \frac{\partial V(R_1, R_2, R_3)}{\partial R_i} T(R_1, R_2, R_3), \tag{3.7}$$

where R_1, R_2 and R_3 are the interatomic distances and:

$$T(R_1, R_2, R_3) = \frac{\alpha}{(R_1 - \beta)^8 + \alpha} \frac{\alpha}{(R_2 - \beta)^8 + \alpha} \frac{\alpha}{(R_3 - \beta)^8 + \alpha}, \tag{3.8}$$

where α and β are positive constants. After the interpolation the derivatives were recovered by inverting the procedure. Table 6 summarizes the parameters used for the grid and the taper function.

Table 7 compares the results of the trajectories calculated by interpolation to the trajectory calculated by using the original DIM potential and derivatives. Fig. 3 displays the trajectory as a projection on the R_1R_2 plane. In constructing the 3D interpolation it was found that the main consideration of computer

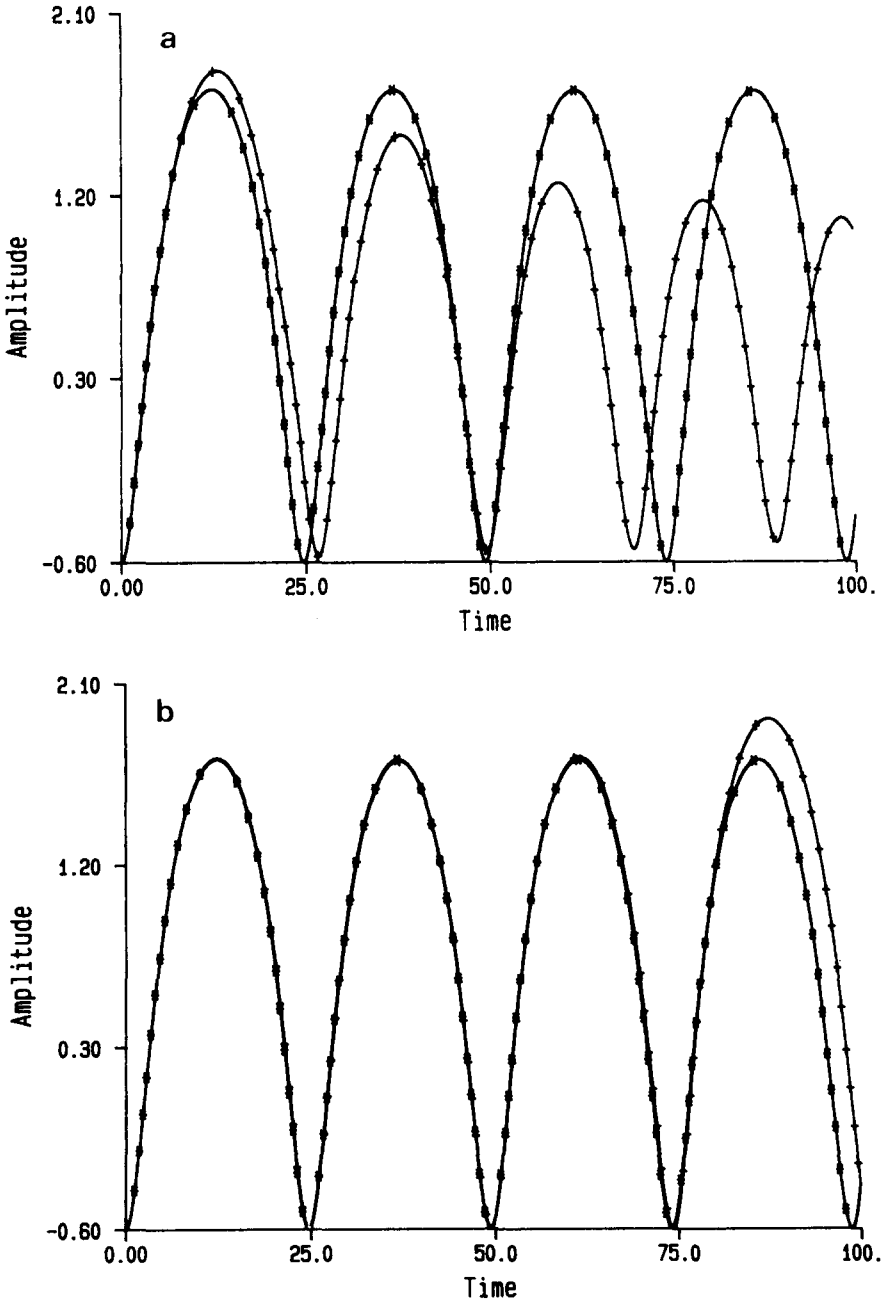

Fig. 2. Classical trajectory of the one dimensional Morse oscillator as a function of time. (a) The analytical trajectory (solid line), the trajectory computed by interpolation of the derivative (×----× line) and the trajectory computed by numerical differentiation (+----+ line). The number of points used in the calculation was $M = 15$. (b) The same as for (a), but with $M = 50$ points.

efficiency was storage. On the VAX 11/750 computer the six grids of derivatives and partial sums of size $60 \times 60 \times 60$ exhausted the maximum available memory size for a program. Also, the method of storage was not optimal for trajectory calculations, because the partial sum coefficients of the 3D grid were not

Table 6
Parameters of grid and taper function for the representation of the DIM potential of the three-dimensional H_3^+-system

Grid size	$N =$	$40\times40\times40$	$60\times60\times60$
Grid distance	$dR_1 = dR_2 = dR_3 =$	0.02	0.015
Starting point of grid	$R_{1,0} = R_{2,0} = R_{3,0} =$	0.61	0.59
Taper constants	$\alpha =$	0.0002	0.0002
	$\beta =$	1.01	1.04

Table 7
Accuracy of the sinc interpolation method for classical trajectories of the three-dimensional H_3^+ system

Number of grid points N	Number of points M used in the interpolation	Average error of interpolation	Relative error after k integration steps		
			$k = 100$	$k = 1000$	$k = 1900$
$40\times40\times40$	$8\times8\times8$	0.0011	0.0012	0.005	–
$60\times60\times60$	$8\times8\times8$	0.00023	0.00058	0.002	0.009
	$12\times12\times12$	0.00013	0.00020	0.003	0.002

stored consecutively in the memory of the computer. Therefore it is not surprising that the calculation by interpolation took longer than the direct calculations using the DIM potential.

As a conclusion, efficient storage is a key factor in using interpolation schemes for classical trajectories. A larger grid would make it possible to represent a larger fraction of the potential, including the dissociation plateau. This would permit the elimination of the taper function by starting the computation of the partial sums from the dissociation plateau.

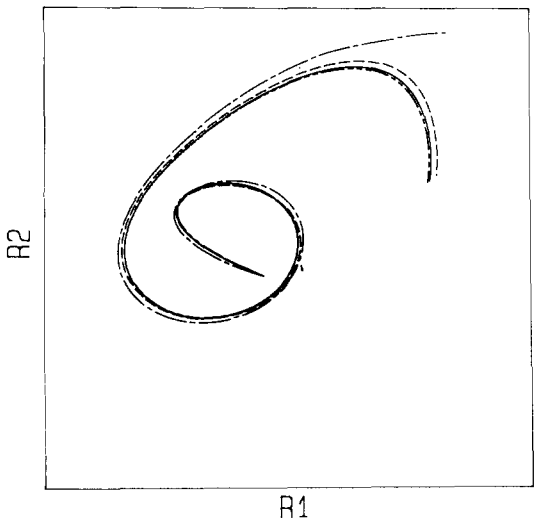

Fig. 3. Projection on the R_1R_2 plane of a classical trajectory for the H_3^+ system computed by different methods: The classical trajectory for the original potential (solid line). The classical trajectory using the interpolated derivatives of the potential for: $N = 40\times40\times40$ and $M = 8\times8\times8$ (– – – – line); $N = 60\times60\times60$ and $M = 8\times8\times8$ (dashed line); $N = 60\times60\times60$ and $M = 16\times16\times16$ (– – – line).

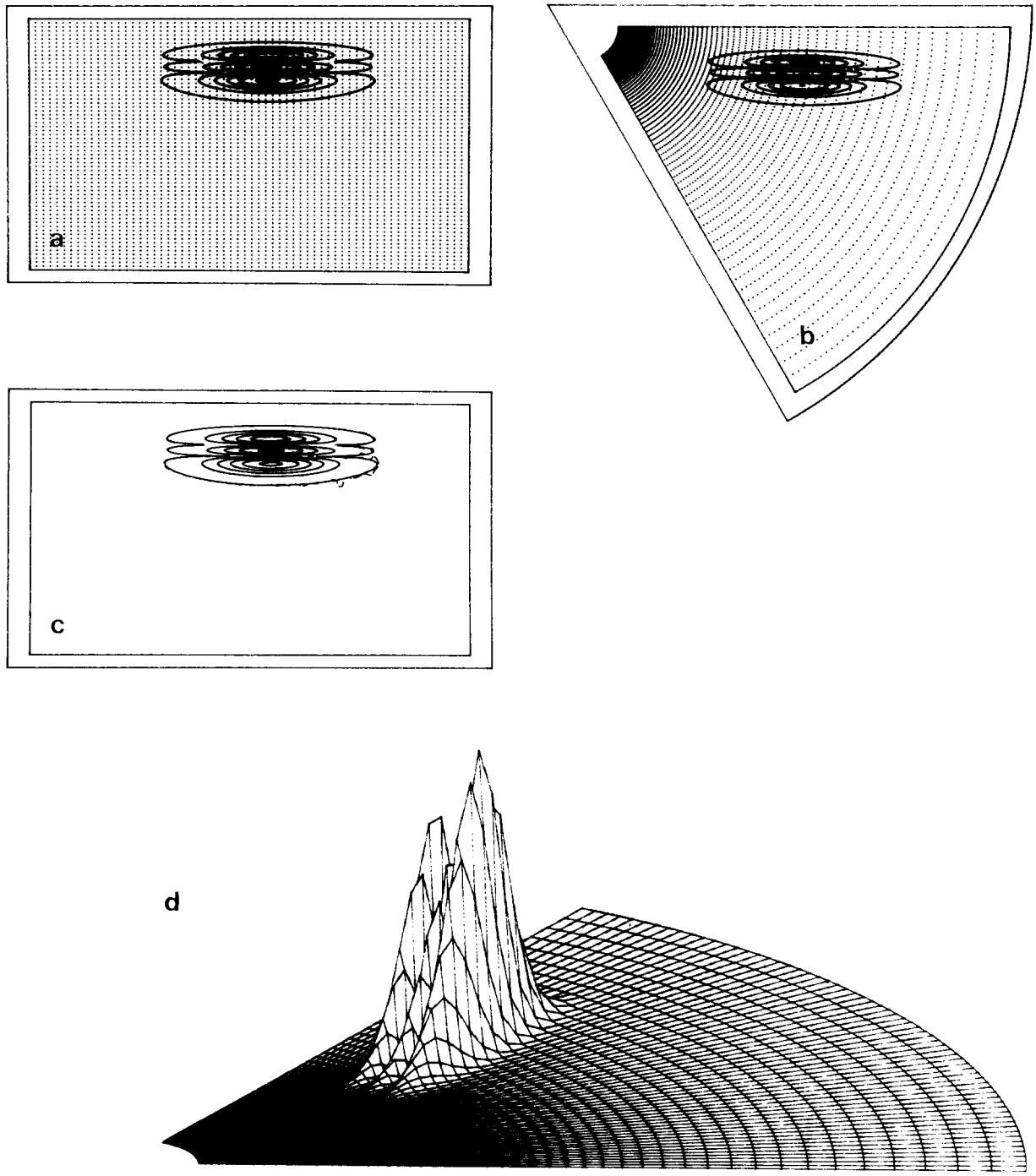

Fig. 4. (a) The original wave function ψ on an $N = 64 \times 64$ Cartesian equidistant grid. (b) The wave function obtained by interpolation into an $N = 64 \times 64$ polar grid, with equal distance in the angular direction and exponential scaling in the radial direction. (c) The original wave function ψ (solid line), compared with the wave function ψ', obtained by transferring ψ to the polar grid and back to the Cartesian grid. The interpolation used $M = 8 \times 8$ points (dash–dotted line, undistinguishable from the original) or $M = 4 \times 4$ points (dashed line). (d) Perspective view of (b).

3.3. Grid to grid transfer

The transfer of a function between (partly) overlapping grids with different geometries can be done efficiently by using the sinc interpolator. As an example the transferral of a wave function from a Cartesian grid to a polar grid will be shown. The need for such a transferral occurred [4] when a wave packet, which was computed on a Cartesian grid, had to be entered into a grid with polar coordinates to be propagated in time. (The wave packet propagation simulated the 2D collinear hydrogen exchange reaction $H + H_2 \rightarrow H_2 + H$.)

The geometries of the grids are as follows: The Cartesian grid is defined by

$$x_i = i\,dx \quad \text{for} \quad i = 0, 1, \ldots, N_x - 1, \qquad y_j = j\,dy \quad \text{for} \quad j = 0, 1, \ldots, N_y - 1, \tag{3.9}$$

and the wave function ψ on the grid is given by the values $\psi_{ij} \equiv \psi(x = x_i,\ y = y_j)$. The polar grid is defined by

$$r_k = r_0\, e^{k\,dr} \quad \text{for} \quad k = 0, 1, \ldots, N_r - 1, \qquad \phi_l = l\,d\phi \quad \text{for} \quad l = 0, 1, \ldots, N_\phi - 1, \tag{3.10}$$

with exponentially increasing grid distance in the radial direction. The wave function on the polar grid is represented by the values $\phi^{kl} \equiv \psi(r = r_k,\ \phi = \phi_l) = \psi(x = r_k \cos \phi_l,\ y = r_k \sin \phi_l)$.

The wave function is transferred from the Cartesian to the polar grid by interpolating the values ψ^{kl} in the Cartesian points $(x = r_k \cos \phi_l,\ y = r_k \sin \phi_l)$ from the known values ψ_{ij} in the Cartesian grid points $(x_i,\ y_j)$. It is also possible to transfer back to the Cartesian system by interpolating the wave function values ψ_{ij} in the polar points $(r = (x_i^2 + y_j^2)^{1/2},\ \phi = \arctan(y_j/x_i))$ from the known values in the polar grid points $(r_k,\ \phi_l)$.

Fig. 4 shows (a) a wave packet ψ on the Cartesian grid; (b) the same function transferred to a polar grid; and (c) the result ψ' of transferring ψ to a polar grid and back, compared with the original function ψ. Fig. 4d shows the wave function on the polar grid in a perspective view. The wave function represents a collinear system of an H atom approaching an H_2 molecule in the $v = 2$ vibrational mode. The wave function ψ is normalized such that $\int\int |\psi(x, y)|^2\, dx\, dy = 1$.

Table 8
Accuracy of the repeated transferral back and forth of a wave function from an $N = 64 \times 64$ Cartesian grid to an $N = 64 \times 64$ polar grid

Number of points M used in the interpolation	Number of transferrals back and forth	Relative error	Overlap
8×8	1	0.0097	0.9979
	2	0.0172	0.9962
	5	0.0456	0.9911
16×16	1	0.0017	1.002
	2	0.0035	1.0005
	3	0.0053	1.0007
	4	0.0070	1.0010
	5	0.0088	1.0013
	10	0.0175	1.0025
	20	0.0346	1.0049
	30	0.4092	1.0071
	40	7.9300	1.0081

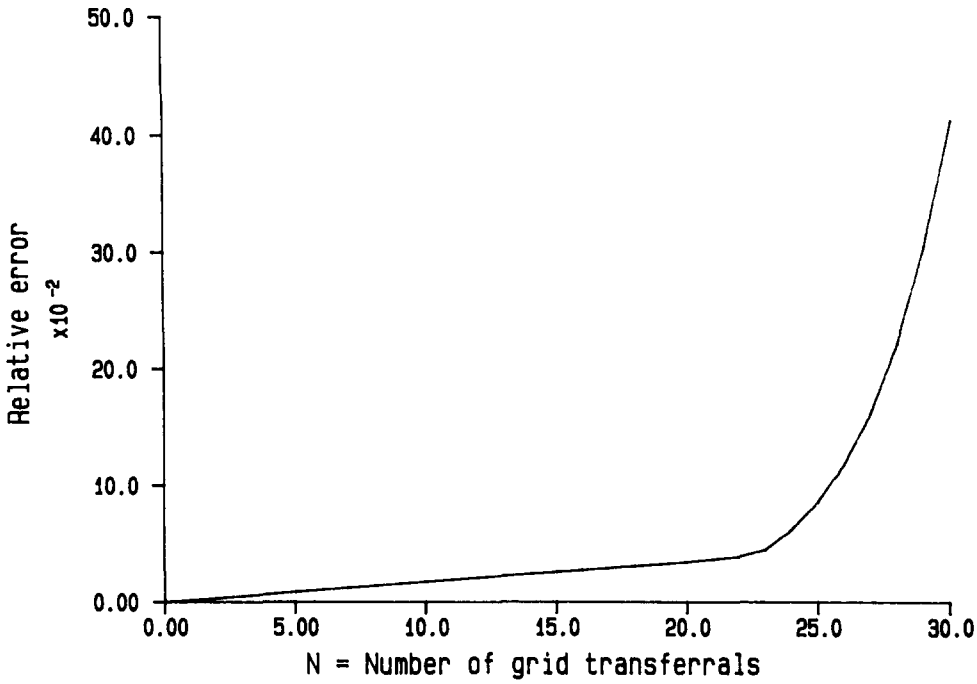

Fig. 5. The increase in error after many grid transferrals back and forth. The wave function is the same as in fig. 4, except for the vibrational mode, which is $v = 0$ in this case.

Fig. 4a and b clearly show the different geometries of the grids. Grid parameters are $N_x = N_y = N_r = N_\phi = 64$, $\mathrm{d}x = 0.188$, $\mathrm{d}y = 0.108$, $\mathrm{d}r = 0.04$, $\mathrm{d}\phi = 0.0166$ and $r_0 = 0.8$. The comparison in fig. 4c shows that the contour lines of the original and the transferred wave function are almost indistinguishable, except for the lowest contour line which represents a value of 3% of the maximum value of $|\psi|^2$. A comparison of ψ and ψ' gives a measure of the accuracy of a (double) grid transferral. Two criteria for the error are useful: the maximum of the difference $|\psi - \psi'|$ over the grid (divided by max $|\psi|$ to normalize) and the overlap $|\langle\psi|\psi'\rangle| = |\int\int\psi^*(x, y)\psi'(x, y)\,\mathrm{d}x\,\mathrm{d}y|$ which should be unity for perfect transferral. A repeated transferral back and forth shows the error propagation. Results are shown in table 8 and fig. 5. The error after one transferral back and forth for the $N = 64 \times 64$ grid is comparable to the error shown in table 3. It is found that the relative error grows linearly until a point where it starts growing out of bounds exponentially.

4. Conclusion

This work used a global approach to interpolation. Such an approach assumes that the interpolated function and its derivatives up to a high order are continuous. As a result, all sampled data contribute to the reconstruction of the interpolated value. It has been shown that this method is easily adaptable to multidimensional interpolation. When using such a global approach care should be taken of local errors in the initial data because such errors will be propagated through the whole grid. This restricts the choice of problems for which such an interpolator is suited.

The basic interpolation formula based on the sinc function assumes that the interpolated function is bandlimited and has a finite support, this being an idealization. Common interpolated functions are only

approximately bandlimited. Nevertheless, it has been demonstrated in the present work that a practical accurate interpolation can be obtained for functions which are very far from the restriction of refs. [1,2]. As an example the Morse potential varies a few orders of magnitude in the interpolated interval, is definitely not bandlimited and still good interpolated results were obtained.

An interesting issue is the acceleration of convergence by the partial summation procedure. By this procedure the global data is compressed to the vicinity of the interpolated point. It would seem that this acceleration procedure could be carried out indefinitely. A careful examination reveals that errors from the boundary are propagated to the interior. Because the interpolated functions are never strictly bandlimited, residual errors always exist at the boundary. This puts a practical limit to the number of accelerations by partial summation which can be used.

An important extension of the interpolation procedure is the calculation of numerical derivatives. As expected, derivatives are more sensitive to imperfections of the interpolated function. An important issue is the character of the eigenvalues of the approximate derivative operator on a finite grid. The analytic derivative operator has purely imaginary eigenvalues. Repeated use of the numerical differencing procedure revealed that the eigenvalues of the approximate derivative have a real component. This means that repeated use in a propagation scheme [17] leads to exponential divergence of the propagated solution. A similar situation has been found in the repeated use of the interpolator in the grid to grid transfer where after a repeated back and forth use the errors grow exponentially.

Accurate interpolation has many important applications in chemistry and physics. The accelerated multidimensional sinc interpolator was found to be of practical use in computations in these fields.

Acknowledgements

We would like to express our thanks to Prof. M. Cohen for stimulating discussions. The computations were carried out on the VAX 11/750 computer of the Fritz Haber Research Center for Molecular Dynamics at The Hebrew University of Jerusalem. This work was partially supported by the Israel Academy of Sciences. The Fritz Haber Research Center is supported by the Minerva Gesellschaft für die Forschung, mbH, Munich, Fed. Rep. Germany.

References

[1] D.P. Petersen and D. Middleton, Informat. Contr. 5 (1962) 279.
[2] R.N. Bracewell, The Fourier Transform and its Applications, 2nd ed. (McGraw-Hill, New York, 1978).
[3] K.S. Miller, An Introduction to the Calculus of Finite Differences and Difference Equations (Henry Holt, New York, 1960) chapt. 1.
[4] R. Bisseling and R. Kosloff, J. Comput. Phys. 59 (1985) 136.
[5] C. Lanczos, Applied Analysis (Prentice Hall, Englewood Cliffs, NJ, 1956) chapt. IV, § 20.
[6] J.H. Rosenbaum and G.F. Boudreaux, Geophys. 46 (1981) 1667.
[7] C. Schwartz, J. Math. Phys. 26 (1985) 411.
[8] C. Cohen-Tannoudji, B. Diu and F. Laloë, Quantum mechanics, vol. I (John Wiley, New York, 1977) p. 61.
[9] E.J. Heller, Acc. Chem. Res. 14 (1982) 368.
[10] G. Drolshagen and E.J. Heller, J. Chem. Phys. 79 (1983) 2072.
[11] R. Kosloff and D. Kosloff, J. Chem. Phys. 79 (1983) 1823.
[12] A.T. Yinnon and R. Kosloff, Chem. Phys. Lett. 102 (1983) 216.
[13] R.N. Porter and L.M. Raff, in: Dynamics of Molecular Collisions, part B, ed. W.H. Miller (Plenum Press, New York, 1976) p. 1.
[14] D.G. Truhlar and J.T. Muckerman, in: Atom–Molecule Collision Theory, ed. R.B. Bernstein (Plenum Press, New York, 1979) p. 505.
[15] R.K. Preston and J.C. Tully, J. Chem. Phys. 54 (1971) 4297.
[16] W.C. Demarcus, Am. J. Phys. 46 (1978) 733.
[17] D. Kosloff, to be published.