UPPSALA
UNIVERSITET

PROJECT REPORT

Optimization of finite difference coefficients for quantum molecular dynamics

Marzieh Farzamfar and Cong Wang

Project in Computational Science: Report

January 2012

DEPARTMENT OF INFORMATION TECHNOLOGY

Optimization of finite difference coefficients for quantum molecular dynamics

Marzieh Farzamfar, Cong Wang

January 2012

Abstract

Quantum molecular dynamics explains the time-evolution of a chemical system by solving the Schrödinger equation, it provides a complete information at an atomic level. The numerical approximation of the solution of the time-dependent Schrödinger equation is discussed in this work. The main purpose is, given a set of grid points, to find a method that minimizes the discretization error. Different methods have been suggested in various quantum molecular studies. In this study we focus on the spectral differences method, since it can solve differential equations by accelerating the summation of the numerical derivatives to produce a matrix representation with two important properties, such as exponential convergence and sparsity. We study spectral differences method when it`s weight is generated by Gegenbauer polynomial and compare it with three other methods.

1. Introduction

Quantum Molecular Dynamics

It is very important for scientists to know what happens to materials under extreme conditions. An accurate look is needed at what happens to individual atoms and molecules during experiments. Simulations based on quantum molecular dynamics make it possible to view experimental activity as it happens. Classical molecular dynamics is concerned with the classical motion of atoms interacting with a given potential, but the interesting chemistry and physics of many molecules take place at the atomic and subatomic level. Quantum molecular dynamics focuses on the interactions between atoms and electrons. The properties of the materials which cannot be measured easily by experiments are predicted by quantum simulations. Quantum molecular dynamic simulations are able to provide accurate information on the properties of materials in extreme conditions (high temperature or high pressure) which are not easy or even impossible to achieve experimentally.

The equation that molecular properties (chemical properties, physical properties and structural properties of molecule) are derived from is the Schrödinger equation, $i\hbar\frac{\partial}{\partial t}\psi(X,t) = \hat{H}\psi(X,t)$ where ψ is the wave function, and $\hat{H}$ is the Hamiltonian operator, which includes terms for potential and kinetic energy. Nuclei are much heavier than electrons and they move slowly so in molecular systems we can assume that electrons are moving in a field of fixed nuclei. We use this assumption in Schrödinger equation to values of effective electronic energy which is dependent on relative nuclear coordinates. When the nuclei are moved to new coordinates, molecular energies are calculated again, this is the description of potential energy surface for the molecule. In addition, structural and dynamic data from molecular dynamics provide accurate insights into the binding affinities, mobility and stability of proteins and nucleic acids that we are not able to get from static model.

The starting point for our project is the solution of the time-dependent Schrödinger equation in one dimension on one potential energy surface. The wave function of a system evolves in time according to the time-dependent Schrödinger equation:

$$i\hbar\frac{\partial}{\partial t}\psi(X,t) = -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial X^2}\psi(X,t) + V(X)\psi(X,t).$$

In this equation, the state of a quantum mechanical system is completely specified by the wave function $\psi(X,t)$ that depends on space and time, and m is the mass of the particle, $\hbar$ is the reduced Planck constant, and V(X) is a real function representing the potential energy surface of the system. We look for numerical solutions for $\psi(X,t)$,which contain all the necessary information about the dynamic system. In the Schrödinger representation, the wave function is time dependent but the operators are time independent. In our study we first use finite difference methods for space discretization and then apply the Crank-Nicolson method and Lanczos iteration algorithm for the time evolution. Next, we implement the spectral differences method to approximate derivatives accurately on a given grid points by using Gegenbauer weight polynomial. Computing the discretization error

for the different methods is considered. As a reference solution we use that, obtained by the Sinc-DVR method to discretize in space and Lanczos iterations for the time discretization.

Each method that we implement for spatial discretization generates a differentiation matrix D which approximates $\frac{\partial^2}{\partial X^2}\psi(X,t)$, and $T = -\frac{h^2}{2m}\frac{\partial^2}{\partial X^2}\psi(X,t)$. So, the sparsity of the matrix D is important, since it is related to the time of our implementation and also the accuracy of the solution is dependent on the approximation of D matrix, In fact with better approximation of D we will have more accurate solution.

1. Solving the equation

We start with Crank-Nicolson scheme for time discretization and second order central finite difference method for spatial discretization.

The finite difference discretization of the time-dependent Schrodinger equation is rather straightforward. We discretize the spatial part of the equation, as follows,

$$\left.\frac{d^2\psi(x)}{dx^2}\right|_{x=x_n} = \frac{\psi_{x_{n+1}} - 2\psi_{x_n} + \psi_{x_{n-1}}}{h^2}.$$

By using Crank-Nicolson Scheme to construct time discretization:

$$i\hbar\frac{\psi_{x_n}^{t_{j+1}} - \psi_{x_n}^{t_j}}{b} = \frac{1}{2}\left|\begin{array}{l}\left[-\frac{\hbar^2}{2mh^2}\left(\psi_{x_{n+1}}^{t_{j+1}} - 2\psi_{x_n}^{t_{j+1}} + \psi_{x_{n-1}}^{t_{j+1}}\right) + V_{x_n}\psi_{x_n}^{t_{j+1}}\right] + \\ \left[-\frac{\hbar^2}{2mh^2}\left(\psi_{x_{n+1}}^{t_j} - 2\psi_{x_n}^{t_j} + \psi_{x_{n-1}}^{t_j}\right) + V_{x_n}\psi_{x_n}^{t_j}\right]\end{array}\right|$$

Denote $f_{x_n} = \frac{b}{2\hbar}V_{x_n} \quad and \quad g = \frac{b\hbar}{4mh^2}$

Then, by substitution we obtain the fully discretized system :

$$\begin{aligned}\psi_{x_n}^{t_{j+1}} + if_{x_n}\psi_{x_n}^{t_{j+1}} + 2ig\psi_{x_n}^{t_{j+1}} - ig\psi_{x_{n+1}}^{t_{j+1}} - ig\psi_{x_{n-1}}^{t_{j+1}} \\ = \psi_{x_n}^{t_j} - if_{x_n}\psi_{x_n}^{t_j} - 2ig\psi_{x_n}^{t_j} + ig\psi_{x_{n+1}}^{t_j} + ig\psi_{x_{n-1}}^{t_j}\end{aligned}$$

$$\left(1 + if_{x_n} + 2ig\right)\psi_{x_n}^{t_{j+1}} - ig\psi_{x_{n+1}}^{t_{j+1}} - ig\psi_{x_{n-1}}^{t_{j+1}} = \left(1 + if_{x_n} + 2ig\right)\psi_{x_n}^{t_j} + ig\psi_{x_{n+1}}^{t_j} + ig\psi_{x_{n-1}}^{t_j}.$$

In matrix form we have:

$$\left(I-\frac{\Delta t}{2i\hbar}H\right)\begin{pmatrix}\psi_1\\ \psi_2\\ \vdots\\ \psi_{N-1}\\ \psi_N\end{pmatrix}^{t_{j+1}} = \left(I+\frac{\Delta t}{2i\hbar}H\right)\begin{pmatrix}\psi_1\\ \psi_2\\ \vdots\\ \psi_{N-1}\\ \psi_N\end{pmatrix}^{t_j}$$

$$H = T + V \ , T = Coeff * D$$

$$\frac{1}{h^2}D = \frac{d^2}{dx^2}, \ Coeff = -\frac{\hbar^2}{2mh^2}$$

We denote $C = Coeff.I$, where I is the identity matrix of proper size. So, the matrix representation of one dimension Hamiltonian in discrete space is given by

$$H = \begin{pmatrix} V-2C & C & 0 & 0 & 0 & C \\ C & V-2C & C & 0 & 0 & \cdots \\ 0 & C & V-2C & C & 0 & \vdots \\ \vdots & 0 & & \ddots & & \\ \cdots & \vdots & 0 & C & V-2C & C \\ C & 0 & 0 & 0 & C & V-2C \end{pmatrix}$$

2. Implementation of Sinc-DVR scheme

One of the most useful methods to estimate unknown values which are within the range of a discrete set of points, is interpolation. For an unknown function $g(x)$ with given data points $\{(x_i, g(x_i))\}$, interpolation can estimate the value $g(x)$. Here $f(x)$ is the interpolant of $g(x)$ and it is defined as $f(x) = \sum_{n=1}^{N} c_n \phi_n(x)$. It is usually convenient to introduce a set of basis functions $C_j(x)$.
In fact $C_j(x)$ are cardinal functions and satisfy $C_j(x_i) = \delta_{ij}$, referred to as the cardinality condition. Now the interpolation of $g(x)$ is represented by

$$f(x) = \sum_{j=1}^{N} g(x_j)C_j(x).$$

In second method we focus on interpolation based on *sinc functions*. The sinc function is defined by

$$sinc(u) = \begin{cases} \dfrac{\sin \pi u}{\pi u} & if\ u \neq 0, \\ 1 & if\ u = 0. \end{cases}$$

The integral form of the sinc function has the following form:

$$sinc(u) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \exp(-i\omega u)\, d\omega.$$

We can derive the orthogonality of translated sinc functions

$$\int_{-\infty}^{\infty} sinc(u)sinc(u-j)du = \frac{1}{2\pi}\int_{-\pi}^{\pi} \exp(-ij\omega)\, d\omega = \delta_{0j}\,.$$

j is an integer and, the Sinc interpolating functions is defined by

$$C_j(x;h) = sinc\left(\frac{[x-x_j]}{h}\right), j = 0, \pm 1, \pm 2 \quad \text{and} \quad C_j(x;h) = \delta_{ij} = \begin{cases} 1, i = j \\ 0, i \neq j \end{cases}$$

While the basis functions are produced by the Sinc functions, the numerical solution is computed as

$$\psi(x,t) = \sum_{j=0}^{M-1} \psi(x_j,t) sinc(\frac{x-x_j}{h})$$

$$\frac{\partial^2}{\partial x^2}\psi(x,t) = \sum_{j=0}^{M-1} \psi(x_j,t) C_j{}^{(2)}(x)$$

$$C_j{}^{(2)}(x) = \frac{\frac{\pi}{h}\sin(\frac{\pi}{h}(x-x_j))}{x-x_j} - \frac{2\cos(\frac{\pi}{h}(x-x_j))}{(x-x_j)^2} + \frac{2\sin(\frac{\pi}{h}(x-x_j))}{\frac{\pi}{h}(x-x_j)^3}$$

$$k = |i-j| \text{ if } \quad k \neq 0$$

If $i = j$ then $x \longrightarrow x_j$

$$C_j^{(2)}(x_j) = -\frac{\pi^2}{3h^2}$$

The differentiation weights for the second derivative are:

$$D_{ij} = \begin{cases} \frac{-\pi^2}{3h^2} & k = 0 \\ \frac{2(-1)^{k+1}}{h^2k^2} & k \neq 0 \end{cases}$$

The matrix representation of H by applying the Sinc-DVR method takes the following form

Let $Coeff = -\frac{\hbar^2}{2m}$, and denote $C = Coeff.I$.

$$H=\begin{bmatrix} -\frac{\pi^2}{3h^2}.C+V & \frac{2(-1)^{k+1}}{h^2k^2}.C & \frac{2(-1)^{k+1}}{h^2k^2}.C & \dots & \frac{2(-1)^{k+1}}{h^2k^2}.C & \frac{2(-1)^{k+1}}{h^2k^2}.C \\ \frac{2(-1)^{k+1}}{h^2k^2}.C & -\frac{\pi^2}{3h^2}.C+V & \frac{2(-1)^{k+1}}{h^2k^2}.C & \cdots & \frac{2(-1)^{k+1}}{h^2k^2}.C & \frac{2(-1)^{k+1}}{h^2k^2}.C \\ \frac{2(-1)^{k+1}}{h^2k^2}.C & \frac{2(-1)^{k+1}}{h^2k^2}.C & -\frac{\pi^2}{3h^2}.C+V & \ddots & \frac{2(-1)^{k+1}}{h^2k^2}.C & \frac{2(-1)^{k+1}}{h^2k^2}.C \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{2(-1)^{k+1}}{h^2k^2}.C & \frac{2(-1)^{k+1}}{h^2k^2}.C & \frac{2(-1)^{k+1}}{h^2k^2}.C & \dots & -\frac{\pi^2}{3h^2}.C+V & \frac{2(-1)^{k+1}}{h^2k^2}.C \\ \frac{2(-1)^{k+1}}{h^2k^2}.C & \frac{2(-1)^{k+1}}{h^2k^2}.C & \frac{2(-1)^{k+1}}{h^2k^2}.C & \dots & \frac{2(-1)^{k+1}}{h^2k^2}.C & -\frac{\pi^2}{3h^2}.C+V \end{bmatrix}$$

The matrix H is dense, which imposes a high computational cost and memory requirements.

3. Implementation of Truncated Sinc-DVR for generating D matrix:

In all of the spectral difference methods, the expression of the mth derivative of the solution is truncated to a finite sum ω in order to generate a banded matrix and replace the derivatives of the $sinc$ cardinal sum-accelerated functions $\Delta_k^{(m)}$ with sum accelerated derivatives $\tilde{\Delta}_k^{(m)}$ as follows

$$u^{(m)}(x_i) = u(x_i)\tilde{\Delta}_0^{(m)} + \sum_{k=1}^{n}(u(x_i+kh)+u(x_i-kh))\tilde{\Delta}_k^{(m)}.$$

The simplest approach for obtaining a matrix D with a bandwidth 2w+1 is to truncate the sum to ω without modifying the derivatives such that $\tilde{\Delta}_k^{(m)} = \Delta_k^{(m)}$.

Boyd [1] has shown that this truncated Sinc method generates a very poor approximation of the derivative.

$$k = |i-j|, \qquad D_{ij} = \begin{cases} (D_{ij})_{sinc}, & k \le \omega \\ 0, & k > \omega \end{cases}$$

As an example, bandwidth = (2 ω +1) , 2 ω +1=3 $\Longrightarrow$ ω =1,

$$D=\begin{bmatrix} D_{ij} & D_{ij} & 0 & 0 & D_{ij} \\ D_{ij} & D_{ij} & D_{ij} & 0 & 0 \\ 0 & D_{ij} & \ddots & \ddots & 0 \\ 0 & 0 & \ddots & D_{ij} & D_{ij} \\ D_{ij} & 0 & 0 & D_{ij} & D_{ij} \end{bmatrix}$$

Bandwidth=5, ω =2

$$D=\begin{bmatrix} D_{ij} & D_{ij} & D_{ij} & 0 & 0 & D_{ij} & D_{ij} \\ D_{ij} & D_{ij} & D_{ij} & D_{ij} & 0 & 0 & D_{ij} \\ D_{ij} & D_{ij} & D_{ij} & \ddots & \ddots & 0 & 0 \\ 0 & D_{ij} & D_{ij} & \ddots & D_{ij} & D_{ij} & 0 \\ 0 & 0 & \ddots & D_{ij} & D_{ij} & D_{ij} & D_{ij} \\ D_{ij} & 0 & 0 & D_{ij} & \ddots & D_{ij} & D_{ij} \\ D_{ij} & D_{ij} & 0 & 0 & D_{ij} & D_{ij} & D_{ij} \end{bmatrix}$$

Of course, it is possible to generate the matrix D with this method by choosing different size of bandwidth; we have shown the matrix D here for the case the bandwidth is 3 and 5. In our program the matrix D that is computed with Sinc method and is truncated to a finite bandwidth 2 ω +1 with a periodic boundary condition.

4. Implementation of finite differences

When using finite differences, the derivatives are approximated as linear combinations of the grid values of the corresponding function, based on truncated Taylor extensions, namely, for some functions $u(x)$, we use

$$\frac{d^m u(x)}{dx^m}\Big|_{x=x_0} \approx \sum_{k=0}^{N} \alpha_k^{(m)} u_k.$$

Since for each grid point, the set of coefficients $\{\alpha_k\}$ is different, therefore in the more general form we have

$$u_j^{(m)} = \sum_{k=0}^{N} \alpha_{jk}^{(m)} u_k.$$

In matrix form

$$\boldsymbol{u}^{(m)} = \boldsymbol{D}^{(m)}\boldsymbol{u}$$

$$\begin{pmatrix} u_0^{(m)} \\ u_1^{(m)} \\ \vdots \\ u_{N-1}^{(m)} \\ u_N^{(m)} \end{pmatrix} = \begin{pmatrix} \alpha_{00}^{(m)} & \alpha_{01}^{(m)} & \cdots & \alpha_{0N}^{(m)} \\ \alpha_{10}^{(m)} & \alpha_{11}^{(m)} & \cdots & \alpha_{1N}^{(m)} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{N-1,0}^{(m)} & \alpha_{N-1,1}^{(m)} & \alpha_{N-1,N-1}^{(m)} & \alpha_{N-1,N}^{(m)} \\ \alpha_{N0}^{(m)} & \alpha_{N1}^{(m)} & \cdots & \alpha_{NN}^{(m)} \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix},$$

where $\boldsymbol{D}^{(m)}$ is the *differentiation* matrix

On a periodic grid, the second order central differentiation matrix for the second derivative is

$$D^{(2)} = \begin{pmatrix} -2 & 1 & 0 & 0 & \cdots & 0 & 1 \\ 1 & -2 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -2 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -2 & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 & -2 & 1 \\ 1 & 0 & \cdots & 0 & 0 & 1 & -2 \end{pmatrix}.$$

We can construct this matrix by generating the finite difference weights for each grid point and then introducing these weights in row i. The algorithm that is used to generate finite differences weights is the algorithm that Fornberg [2] is introduced.

❖ **Algorithm for computing weights for finite differences**

In [2], Fornberg determines the weights in finite difference formulas at the grid points. Let $\{\alpha_0, \alpha_1, \dots, \alpha_N\}$ be distinct real numbers, we define $\omega_n(x) := \prod_{k=0}^{n}(x - \alpha_k)$, the polynomial

$F_{n,v} := \frac{\omega_n(x)}{\acute{\omega}_n(\alpha_v)(x-\alpha_v)}$ is one of the minimal degree , $\begin{cases} x = \alpha_v \,, F_{n,v} = 1 \\ x = \alpha_k, F_{n,v} = 0 \\ 0 \le k \le n \\ k \ne v \end{cases}$.

Let $p(x) := \sum_{v=0}^{n} F_{n,v}(x) f(\alpha_v)$ be the Lagrange polynomial for an arbitrary function $F(x)$. In fact, the weights show the changes of the value of $\left[\frac{d^m p(x)}{dx^m}\right]_{x=0}$ in $f(\alpha_v)$. The changes in each $f(\alpha_v)$ affect only one term in $p(x)$. So, we find

$$\delta_{n,v}^{m} = \left[\frac{d^m}{dx^m} F_{n,v}(x)\right]_{x=0} \quad \Longrightarrow \quad F_{n,v}(x) = \sum_{m=0}^{n} \frac{\delta_{n,v}^{m}}{m!} x^m \,,$$

$$F_{n,v} := \frac{\omega_n(x)}{\acute{\omega}_n(\alpha_v)(x-\alpha_v)} \Rightarrow \quad \omega(x) = (x - \alpha_n)\omega_{n-1}(x) \implies \acute{\omega}_n(x) = (x - \alpha_n)\omega_{n-1}^{\prime}(x) + \omega_{n-1}(x)$$

$$F_{n,v}(x) = \frac{x - \alpha_n}{\alpha_v - \alpha_n} F_{n-1,v}(x)$$

$$F_{n,n}(x) = \frac{\omega_{n-1}(x)}{\omega_{n-1}(\alpha_n)} = \frac{\omega_{n-2}(\alpha_{n-1})}{\omega_{n-1}(\alpha_n)}(x - \alpha_{n-1}) F_{n-1,n-1}(x) \qquad (n > 1).$$

Now we substitute $F_{n,v}(x) = \sum_{m=0}^{n} \frac{\delta_{n,v}^{m}}{m!} x^m$ in to the above equation, and the desired relations between the weights are obtained:

$$\delta_{n,v}^{m} = \frac{1}{\alpha_n - \alpha_v}\left(\alpha_n \delta_{n-1,v}^{m} - m\delta_{n-1,v}^{m-1}\right),$$

$$\delta_{n,n}^{m} = \frac{\omega_{n-2}(\alpha_{n-1})}{\omega_{n-1}(\alpha_n)}(\mathrm{m}\delta_{\mathrm{n-1,n-1}}^{\mathrm{m-1}} - \alpha_{\mathrm{n-1}}\delta_{\mathrm{n-1,n-1}}^{\mathrm{m}}),$$

$$\sum_{\upsilon=0}^{n} \delta_{n,\upsilon}^{m} = \begin{cases} 1, & m = 0, \\ 0, & m > 0. \end{cases}$$

As is seen from the latter formulas, this algorithm is able to generate the coefficients of the approximation at a grid point for derivatives different order and also with different order of accuracy. This is very important to solve partial differential equations in science problems. Generating the differentiation matrix is possible when we have the finite difference weights. We can write a general approximation for the m-th derivative at a grid point i , using a stencil of n+1 neighboring nodes , as

$$u_i^{(m)} = \frac{1}{\Delta x^m}\sum_{j=0}^{n} \alpha_j u_j, \begin{cases} m\ is\ order\ of\ derivative, \\ n\ is\ order\ of\ accuracy\ . \end{cases}$$

Once we have determined the weights that we use to approximate the m-th derivatives at each grid point, so now for a grid of N+1points we have;

$$u_i^{(m)} = \sum_{j=0}^{n} d_{ij}^{(m)} u_j,$$

$$u^{(m)} = D^{(m)} u.$$

Here, D is again the differentiation matrix which is a sparse matrix and needs lower computational cost and memory requirements.

6. Implementation of the Spectral differences approximation

Earlier articles [2] [3] show that, for solving the Schrödinger equation, spectral methods are a good choice since, in comparison with finite differences, they are global. This means that, in finite differences, computation at any point depends only on information at neighboring points but in spectral methods it depends on the whole domain of computation.

Spectral difference methods use sum-acceleration of the second derivative in the kinetic energy to generate a Hamiltonian which is sparse and spectrally accurate at the same time. Actually, for the spectral differences method with equally spaced grid points, both sparsity and rapid convergence are achievable, which is especially useful in chemical physics. It can be used to establish important molecular information by solving Schrodinger equations. Grid implementation of spectral methods is an accurate method for solving differential equations. In fact the Sinc-DVR method is a suitable method for solving chemical problems since it is able to approximate well functions that decay exponentially. As mentioned in Section 3, the representation of expanded wave function with a Sinc basis function on an infinite grid is

$$\psi(x) = \sum_{j=-\infty}^{\infty} \psi(x_j) C_j(x,h),$$

$$C_j(x,h) = sinc\left[\frac{x-x_j}{h}\right],$$

$$sinc(x) = \frac{\sin(\pi x)}{\pi x}.$$

In the Schrödinger equation, we need to approximate the second derivative of the wave function:

$$\psi^{(m)}(x_j) = \psi(x_j)\Delta_0{}^{(m)} + \sum_{k=1}^{\infty} [\psi(x_i + kh) + (-1)^m \psi(x_i - kh)]\Delta_k^{(m)}.$$

Here, N is number of points and the h is grid spacing and we obtain

$$\Delta_k^{(m)} = C_j^{(m)}(x_i, h), k = j - i.$$

Spectral differences have both spectral accuracy and sparsity features by accelerating the convergence of the sum in the above equation. This acceleration is gained by truncating the sum at $m \leq N$ and we use modified derivative weights ($\tilde{\Delta} = w\Delta$) so the accelerated approximation of the second derivative becomes

$$\psi^{(2)}(x_j) = \psi(x_j)\tilde{\Delta}_0{}^{(2)} + \sum_{j=1}^{m} [\psi(x_i + kh) + \psi(x_i - kh)]\tilde{\Delta}_k^{(2)}.$$

Boyd [1] has suggested two methods for sum-acceleration, including the Euler method and the Finite difference method. We want to have the derivatives in the kinetic energy with high accuracy. For a free particle with no external force V(x)=0 the Schrodinger equation contains only the kinetic energy operator. So we have $\frac{d^2}{dx^2}\psi(x) = -k^2\psi(x)$ and the eigensolution of the wave function in this

condition (free particle) is $\psi(x) = e^{ikx} = \exp(ikx)$. We use this function as a numerical approximation in Schrödinger equation so the result is $\psi^{(2)}(x_j) = \tilde{\Delta}_0{}^{(2)} + 2\sum_{j=1}^{m} \exp(ik(jh))\tilde{\Delta}_j^{(2)}$.

The sum-acceleration of a second derivative should minimize the least-squares error in the approximated frequencies. We denote the frequency, for the second derivative in Schrodinger equation by $\varepsilon(K)$ and by using Fourier transformation, the frequency error is defined to be

$$\varepsilon(K) = \psi^{(2)} + k^2.$$

Actually, the frequency error measures the error at each frequency k for the free particle.

$$\varepsilon(K) = \tilde{\Delta}_0^{(2)} + 2\sum_{j=1}^{m} \tilde{\Delta}_j{}^2 \cos(jhK) + k^2,$$

Where δ_j are the second derivative weights that are to be optimized for the best sum acceleration. We can express the least-squares minimization problem in this formula:

$$I = \int_{-\pi}^{\pi} \omega(K)\varepsilon(K)^2\, dK$$

$$Derivative\ approximation\ \ \psi(x) = \sum_{j=-n}^{n} \delta_j \psi(x + jh)$$

$$I = \int_{-\pi}^{\pi} \omega(k) \left(K^2 - \sum_{j=-n}^{n} \Delta_j^{(2)} e^{j\,jk}\right)^2 dK.$$

We define the inner product as

$$(u, v) \equiv \int_{-\pi}^{\pi} \omega(k)u(K)v(K)dK.$$

Where $u(K)\ and\ v(K)$ are two arbitrary functions.

Let $\varphi_j(K) = \cos(jK)$

In the expressions, $\omega(K)$ is a frequency weight function and the scaled frequency is the highest frequency that the grid can capture and so $K_{max} = \pm\frac{\pi}{h}$, and grid spacing is denoted by h . We apply different spectral differences techniques, obtained by different choices of the weight function $\omega(K)$.

For a given function, the optimum weights (δ_j) are those, that minimize the frequency error in least-squares setting:

$$\min_\delta I(\delta) = \int_{-\infty}^{\infty} \varepsilon(K)^2 dx,$$

$$\min_\delta I(\delta) = \int_{-\infty}^{\infty} \left(\frac{\partial^2 u}{\partial x^2} - \sum_{j=-w}^{w} \delta_j u(x + jh)\right)^2 dx.$$

Applying Fourier transform to the above formula we get

$$\varepsilon(k) = -\frac{K^2}{h^2} - \delta_0 - 2\sum_{j=1}^{w} \delta_j \cos(jK)$$

$$\min_{\delta} J(\delta) = \int_{-\pi}^{\pi} \omega(k)\varepsilon(K)^2 \, dK$$

With two arbitrary functions $u(K) and\ v(K)$, we define the inner product, and then we can compute the weighted least squares solution by solving the linear system:

$\boldsymbol{G\delta = \chi}$ G is an square matrix $\boldsymbol{G_{ij} = 2(\varphi_i, \varphi_j)}$ and $\boldsymbol{\delta\ and\ \chi}$ are vectors

$$\chi_j = -\frac{1}{h^2}(K^2, \varphi_j)$$

$$\delta_j = \delta_j^{(2)}$$

We use the Gegenbauer polynomials for $\omega(K)$, these polynomials are orthogonal on the interval $[-\pi, \pi]$ with respect the following weight:

$$\omega(K) = (1 - \frac{1}{\pi^2}K^2)^{\alpha}$$

By solving this linear system in Matlab, we find that the spectral differences (Gegenbauer polynomials) method is appropriate but only when the number of grid points is less than 64, and the reason of this problem is the matrix G is ill-conditioned. Therefore, we consider dealing with this ill-conditioning by analytical techniques and high-precision floating point arithmetics.

Analytical Gegenbauer spectral differences

For solving the linear system $G\delta = \chi$ with analytical methods; we implement numeric method and symbolic method in Maple.

To solve the arising system of linear equations, we use Maple's function nt , which compute the integrals within the linear system.

Symbolic Method:

Gegenbauer polynomials are able to approximate any arbitrary function (K) , where the value of K is in the interval $[-1,1]$ according to $\omega(K, \alpha) = (1 - K^2)^{\alpha}$. Gegenbauer polynomials $G_i^{\alpha}(K)$ are orthogonal with respect to $\omega(K)$, so if we expand the error function with Gegenbauer then the coefficients of the expansion are given by the integral I_2:

$$\mathrm{I}_2 = \int_{-1}^{1} \omega(\mathrm{K})\mathrm{G}_\mathrm{i}^{\alpha}(\mathrm{K})\varepsilon(\mathrm{K})\,\mathrm{dK} = 0\,, \mathrm{i} = 0, \dots, \mathrm{n}.$$

Gegenbauer polynomials are linear combinations of $\{1, K, K^2, \dots, K^n\}$ so we can write the integration of I_1 in the form:

$$\mathrm{I}_1 = \int_{-1}^{1} \omega(\mathrm{K})\mathrm{K}^{\mathrm{i}}\varepsilon(\mathrm{K})\,\mathrm{dK} = 0\,, \mathrm{i} = 0, \dots, \mathrm{n}.$$

With these definitions , we estimate the value of the integrals I_1 and I_2 as

$$\mathrm{I}_2(\mathrm{i}, \mathrm{j}, \alpha) = \int_{-1}^{1} \cos(\mathrm{j}\pi\mathrm{K})\,\mathrm{K}^{2\mathrm{i}}(1 - \mathrm{K}^2)^{\alpha}\mathrm{dK},$$

$$\mathrm{I}_1(\mathrm{i}, \alpha) = \int_{-1}^{1} \mathrm{K}^{2\mathrm{i}}(1 - \mathrm{K}^2)^{\alpha}\mathrm{dK}.$$

These integrals can be computed exactly in Maple, using gamma functions and the generalized hyper-geometric function

$$\mathrm{I}_1(\mathrm{i}, \alpha) = \frac{\Gamma(\alpha + 1)\Gamma(1/2 + \mathrm{i})}{\Gamma(3/2 + \alpha + \mathrm{i})}$$

$$\mathrm{I}_2(\mathrm{i}, \mathrm{j}, \alpha) = \mathrm{I}_1(\mathrm{i}, \alpha)\,{}_1\mathrm{F}_2\left(\tfrac{1}{2} + \mathrm{i}, \left[\tfrac{1}{2}, \tfrac{3}{2} + \alpha + \mathrm{i}\right], -\tfrac{1}{4}\mathrm{j}^2\pi^2\right).$$

Finally, the derivative weights for Gegenbauer spectral method is the solution of the linear equation:

$$\tilde{\Delta}_0^{(2)} I_1(i, \alpha) + 2\sum_{j=0}^{n} \tilde{\Delta}_j^{(2)} I_2(i, j, \alpha) + \pi^2 I_1(i + 1, \alpha) = 0, \qquad i = 0, \dots, n$$

For the implementation of the symbolic method we have used $I_1 and\ I_2$ above as build-in functions. The theoretical explanation of the symbolic method is described below.

$$\chi_j = -\frac{1}{h^2}(K^2, \varphi_j)$$

$through\ the\ definition\ of\ \left(K^2, \varphi_j\right), the\ above\ formula\ is\ expanded\ to$

$$\chi_j = -\tfrac{1}{h^2}\int_{-\pi}^{\pi}(1 - \tfrac{1}{\pi^2}K^2)^{\alpha}K^2\cos(jK)dK.$$

Let $K = m\pi$ and $m \in [-1,1]$, then the formula is transformed to

$$\chi_j = -\tfrac{1}{h^2}\int_{-\pi}^{\pi}(1 - m^2)^{\alpha}(m\pi)^2\cos(jm\pi)dm\pi.$$

After changing the variable under the integral from $m\pi$ to m, we obtain

$$\chi_j = -\tfrac{\pi^3}{h^2}\int_{-\pi}^{\pi}(1 - m^2)^{\alpha}(m)^2\cos(jm\pi)dm.$$

Then use I_2 to instead the integral part of the above function,

$$\chi_j = -\frac{\pi^3}{h^2} I_2(1, j, \alpha) \quad j = 0, \dots, \omega$$

Apply the same idea for the operator G_{ij} as follows:

$$G_{ij} = 2\int_{-\pi}^{\pi}(1 - \tfrac{1}{\pi^2}K^2)^{\alpha}\cos(iK)\cos(jK)dK \ .$$

$Let\ K = m\pi$ and get the integral of m with the interval [-1,1],

$$G_{ij} = 2\pi \int_{-1}^{1}(1 - m^2)^{\alpha}\cos(i\pi m)\cos(j\pi m)dm$$

By applying the formula $\cos\theta + \cos\beta = 2\cos(\frac{\theta+\beta}{2})\cos(\frac{\theta-\beta}{2})$ to replace $\cos(i\pi m)\cos(j\pi m)$, we arrive at

$G_{ij} = \pi\int_{-1}^{1}(1 - m^2)^{\alpha}\cos\big((i + j)\pi m\big)dm + \pi\int_{-1}^{1}(1 - m^2)\cos((i - j)\pi m)dm \qquad i, j = 0, \dots, \omega.$

Denote next $+j = p\ and\quad i - j = q$.

When $j = 0 \Rightarrow G_{ij} = \frac{\pi}{2}I_2(0, i, \alpha) + \frac{\pi}{2}I_2(0, i, \alpha) = \pi I_2(0, i, \alpha)$

When $j \neq 0 \ \Rightarrow G_{ij} = \ \pi I_2(0, p, \alpha) + \pi I_2(0, q, \alpha)$

Then the linear system is defined as

$\delta_0 G_{i0} + \sum_{j=0}^{\omega}\big(\delta_j G_{ij}\big) - \chi_j = 0 \qquad i = 0, \dots, \omega$

The so-obtained linear system can be easily solved with high precision floating point arithmetic in Maple. We have used $igits = 50$. However, we got the similar result in Matlab implementation, in fact the Gegenbauer method still produces larger errors than the high-order finite difference method for meshes with more than 64 grid points.

N / Method	128	256	512	1024	2048
Sinc - DVR	6.462650	9.590642	18.03130	63.46217	292.3984
Trun - Sinc	3.003596	3.900185	5.306299	9.415642	33.40159
Finite differences	2.988733	3.536411	4.724180	6.855668	21.54972
Gegenbauer	2.936982	3.466816	4.574816	6.849255	21.29761

Table1:Execution time (seconds) for different methods with respect to the number of grid points

In all methods the equation $\psi^{n+1} = (D + V)\psi^n$ is solved. As already has mentioned, D is an N*N matrix, and ψ is a N-vector. In Sinc-DVR method, the matrix D is a full matrix. In the truncated Sinc-DVR, the finite differences and Gegenbauer method, D is a sparse matrix. The multiplication time when D is a full matrix is of order $O(N^2)$ while in the case D is a sparse matrix it reduces to $O(N)$. The latter difference influences the execution time significantly and the experiments when D is dense run much slower. This conclusion is confirmed by the results in the following table. We see that the Sinc-DVR method is the most time-consuming method. Also, to see the difference in execution time we use larger number of points. When the number of points is 2048, the execution time of Sinc-DVR method is about 10 times larger than the execution time of the three other methods.

N / Method	256	128	64	32	16
Sinc - DVR	3.735e-013	7.338e-013	3.344e-008	1.806e-003	0.1268
Trun - Sinc	2.266	3.845e-001	8.649e-002	2.428e-002	0.1268
Finite differences	5.124e-009	1.176e-006	1.999e-004	1.437e-002	0.1511
Analytical	4.209e-005	1.142e-005	2.62e-005	1.031e-002	0.1466
Gegenbauer	3.092e-005	8.561e-006	2.926e-005	1.031e-002	0.1466

Table2: Computation of Error for different methods with respect to the number of grid points

In Table 2, the discretization error is computed with respect to the number of grid-points. As we can see, the Sinc-DVR method is the most accurate method. Gegenbauer spectral differences method has also good accuracy but when the number of grid points is more than 64 then the finite difference method becomes more accurate due to the ill-conditioning of the matrix G.

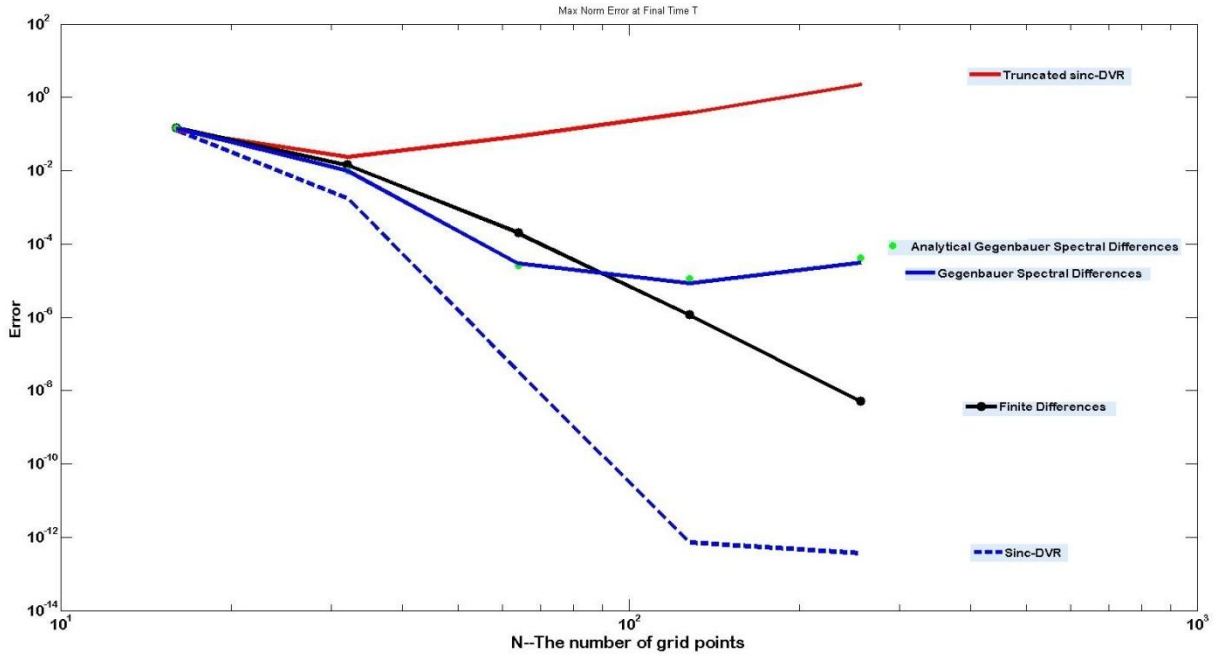

Figure1: The figure shows the Error as a function of N for five different methods: Sinc-DVR, Truncated Sinc-DVR, Finite differences, Analytical spectral differences and Gegenbauer Spectral differences.

The order of accuracy for the different methods is shown in Figure 1. The Sinc-DVR method is exponentially convergent. On the other hand, the finite difference method has forth order of accuracy however is not exponentially accurate. Gegenbaur spectral differences method is exponentially convergence only for less than 64 grid points and the truncated Sinc-DVR method is worse with respect to accuracy.

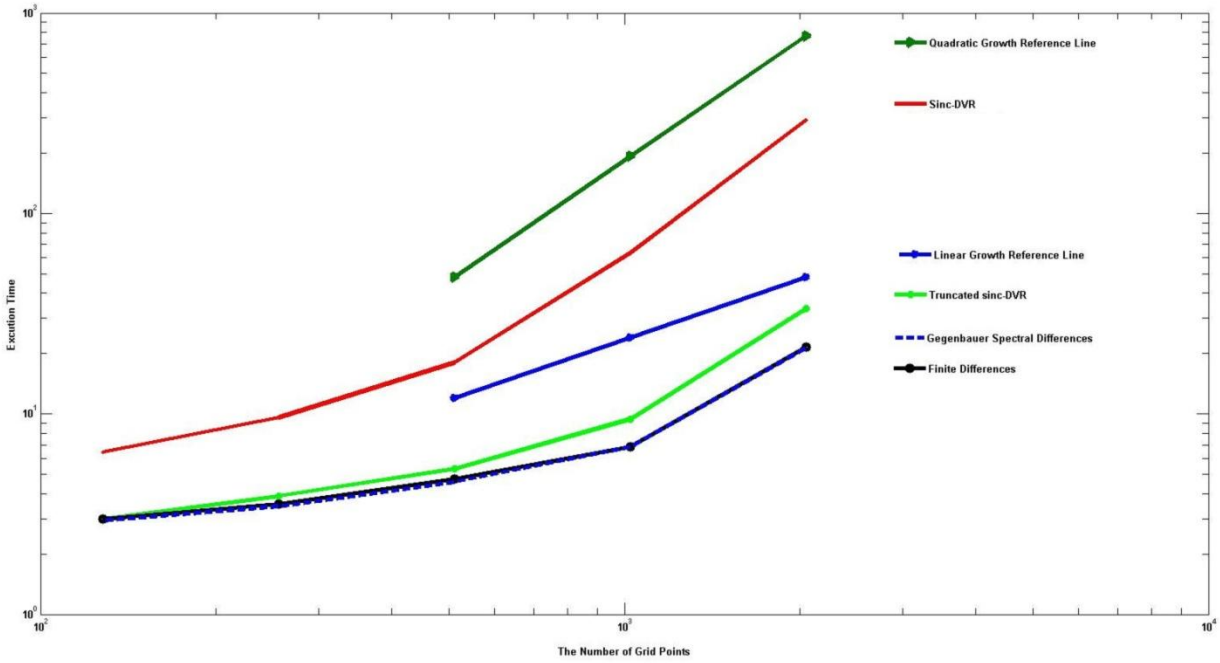

Figure 2: the figure shows the execution time of different methods as a function of grid points.

As we can see in Figure 2, the execution time for Sinc-DVR method is growing quadratically with respect to the number of grid points, however for the truncated Sinc-DVR, the finite differences and Gegenbauer methods the execution time is increasing linearly.

Reference

[1] J. P. Boyd, "Sum-accelerated pseudospec ral methods: finite differences and sech-weighted differences," 1994.

[2] B. Fornberg, "Generation of finite difference formulas on arbitrarily spaced Grids," *Mathematics of computation ,* vol. 51, no. 184, pp. 699-706, 1988.

[3] D. A. Mazziotti, "Spectral difference methods for solving the differential equations of chemical physics," vol. 117, 2002.

[4] D. A. Mazziotti, "Spectral difference methods for solving differential equations," Chemical Physics letters, Cambridge, 1999.