ORIGINAL RESEARCH
published: 28 September 2020
doi: 10.3389/fphy.2020.00390

Numerical Solutions of Quantum Mechanical Eigenvalue Problems

Asif Mushtaq[1]*, *Amna Noreen*[2] *and Kåre Olaussen*[3]

[1] Seksjon for Matematikk, Nord Universitet, Bodø, Norway, [2] Bodin videregående skole, Nordland fylkeskommune, Bodø, Norway, [3] MJAU, Trondheim, Norway

A large class of problems in quantum physics involve solution of the time independent Schrödinger equation in one or more space dimensions. These are boundary value problems, which in many cases only have solutions for specific (quantized) values of the total energy. In this article we describe a Python package that "automagically" transforms an analytically formulated Quantum Mechanical eigenvalue problem to a numerical form which can be handled by existing (or novel) numerical solvers. We illustrate some uses of this package. The problem is specified in terms of a small set of parameters and selectors (all provided with default values) that are easy to modify, and should be straightforward to interpret. From this the numerical details required by the solver is generated by the package, and the selected numerical solver is executed. In all cases the spatial continuum is replaced by a finite rectangular lattice. We compare common stensil discretizations of the Laplace operator with formulations involving Fast Fourier (and related trigonometric) Transforms. The numerical solutions are based on the NumPy and SciPy packages for Python 3, in particular routines from the **`scipy.linalg`**, **`scipy.sparse.linalg`**, and **`scipy.fftpack`** libraries. These, like most Python resources, are freely available for Linux, MacOS, and MSWindows. We demonstrate that some interesting problems, like the lowest eigenvalues of anharmonic oscillators, can be solved quite accurately in up to three space dimensions on a modern laptop—with some patience in the 3-dimensional case. We demonstrate that a reduction in the lattice distance, for a fixed the spatial volume, does not necessarily lead to more accurate results: A smaller lattice length increases the spectral width of the lattice Laplace operator, which in turn leads to an enhanced amplification of the numerical noise generated by round-off errors.

Keywords: numpy array, FFT (fast fourier transform), quantum mechanics, python classes, eigenvalue problems, sparse SciPy routines, Schrödinger equations

OPEN ACCESS

Edited by:
Jesus Martin-Vaquero,
University of Salamanca, Spain

Reviewed by:
Kazuharu Bamba,
Fukushima University, Japan
Jan Sladkowski,
University of Silesia of Katowice, Poland

****Correspondence:***
Asif Mushtaq
asif.mushtaq@nord.no

Specialty section:
This article was submitted to Mathematical and Statistical Physics, a section of the journal Frontiers in Physics

Received: *30 November 2019*
Accepted: *11 August 2020*
Published: *28 September 2020*

Citation:
Mushtaq A, Noreen A and Olaussen K (2020) Numerical Solutions of Quantum Mechanical Eigenvalue Problems. Front. Phys. 8:390. doi: 10.3389/fphy.2020.00390

1. INTRODUCTION

The Schrödinger equation has been a central part of "modern" physics for almost a century. When interpreted broadly, it can be formulated in a multitude of ways [1]. Here we mainly restrict our discussion to the non-relativistic, time independent form,

$$\left[-\Delta_{\mathbf{q}} + V(\mathbf{q})\right]\psi(\mathbf{q}) = E\psi(\mathbf{q}). \tag{1}$$

This constitutes an eigenvalue problem for E (there are many cases where the operator defined by Equation (1) allows for a continuous spectrum of E-values, but this will not directly influence the treatment of finite discretizations of such systems). In Equation (1), $\mathbf{q}$

denotes the configuration space coordinate for a system of one or more particles in one or more spatial dimensions, and $\Delta_{\mathbf{q}}$ is a Laplace operator on this configuration space. $V(\mathbf{q})$ is the interaction potential, and E the eigenvalue parameter, interpreted as an allowed energy for the quantum system.

Despite its appearance as a single-particle equation, Equation (1) can also be used to model N-particle systems, with $\mathbf{q} = (\mathbf{r}_1, \ldots, \mathbf{r}_N)$ and $\Delta_{\mathbf{q}} = (c_1\Delta_1, \ldots, c_N\Delta_N)$. Here each Δ_k is an ordinary flat space Laplace operator, and c_k is a numerical coefficient inversely proportional to the mass m_k of particle k; this mass may differ from particle to particle. By a suitable scaling of each coordinate $\mathbf{r}_k$, one can mathematically transform all c_k to (for instance) unity. But such transformations may obscure physical interpretations of the coordinates, and make mathematical formulations more error-prone.

How to solve eigenvalue problems like (1)? Fortunately for the rapid initial development of quantum mechanics, for many important physical cases [like the hydrogen atom [2, 3] and harmonic oscillators [4]] it could be reduced to a set of one-dimensional eigenvalue problems, through the separation of variables method. Moreover, the resulting one-dimensional problems could all be solved exactly by analytic methods. The origins for such fortunate states of affairs can invariably be traced to an enhanced set of symmetries. However, not every system of physical interest enjoy a high degree of symmetry. Even most one-dimensional problems of the form (1) have no known analytic solution. A popular and much investigated system is the anharmonic oscillator,

$$\left[-\frac{d^2}{dx^2} + \mu x^2 + \varepsilon x^4\right]\psi(x) = E\psi(x). \tag{2}$$

This model has often functioned as a theoretical laboratory [5, 6], for instance to investigate the behavior and properties of perturbative [7, 8] and other [9–12] expansions, and alternative solution methods [13–15].

It this article we describe some attempts to simplify *numerical* solutions of eigenvalue problems like (1). Our approach relies on standard numerical algorithms, already coded and freely available through Python packages like **`numpy`** [16] and **`scipy`** [17, 18]. The main aim is to automatize the transformation of (1) to function calls accepted by the numerical eigenvalue solvers. Within the above class of models, the problem is completely defined by the coefficient vector $(c_1, c_2, \ldots, c_N)$ and the real function $V(\mathbf{q})$. In principle, this should be the only user input required for a numerical solution.

In practice some additional decisions must be made, like how a possibly infinite configuration space should be reduced to a region of finite extent, how the boundaries of this region should be treated, and how this region should be further approximated by a finite lattice. Other options involve selection of numerical approaches, like whether dense or iterative sparse matrix solvers should be used. Such decisions have consequences for many "trivial" details of the numerical programs, but they can be provided in the form of parameters and selectors, automatically implemented without further tedious and error-prone human intervention. Even many of the decisions indicated above may ultimately by delegated to artificial intelligence systems, but this is beyond our current scope.

2. AVAILABLE PYTHON PROCEDURES FOR NUMERICAL SOLUTION

Numerical approaches to problems like those above are in principle straightforward: The operator

$$\mathrm{H} = \mathrm{T} + \mathrm{V}$$

defined by Equation (1) is approximated by a finite real symmetric matrix

$$M_{\mathbf{H}} = M_{\mathbf{T}} + M_{\mathbf{V}}$$

where we have introduced the symbol $\mathbf{T} = -\Delta_{\mathbf{q}}$. For densely defined matrices $M_{\mathbf{H}}$ there are several standard numerical eigenvalue solvers available, like **`eig`** and **`eigvals`** in the **`scipy.linalg`** package. A $10^4 \times 10^4$ matrix of double precision numbers requires 800 Mb of storage space; this is indicative of the problem magnitudes that can be handled by dense matrix methods on (for instance) modern laptops. That is, such computers have more than enough memory for numerical treatment of one-dimensional problems, and usually also sufficient memory for two-dimensional ones.

For higher-dimensional problems one may utilize the sparse nature of $M_{\mathbf{H}}$ to find solutions through iterative procedures, like the **`eigsh`** eigenvalue solver in the **`scipy.sparse.linalg`** package. This solver does not require any explicit matrix construction of $M_{\mathbf{H}}$, only a **`LinearOperator`** function that returns the vector $M_{\mathbf{H}}\psi$ for any input vector ψ. In the representations we consider, $M_{\mathbf{V}}$ is always diagonal, and $M_{\mathbf{T}}$ can be made diagonal by a Fast Fourier Transform (FFT), or some of its discrete trigonometric variants. This opens the possibility it to handle non-sparse matrix problems, where $\mathbf{T}$ is replaced by more general expressions of $F(\mathbf{T})$, by the same procedures. For instance functions F that involves fractional and/or inverse powers of its arguments.

3. REQUIRED PARAMETERS AND SELECTORS

In this section we describe the additional quantities that a user must input for a full specification of the numerical problem. They assume that configuration space has been modeled by a rectangular point lattice, with a selection of possible boundary conditions.

3.1. Lattice Shape

The most basic quantity of the numerical model is the discrete lattice approximating the relevant region of configuration space. For rectangular approximations this is defined by the **`shape`** parameter, a Python *tuple*,

$$\texttt{shape} = \left(s_0, s_1, \ldots, s_{d-1}\right), \tag{3}$$

where each s_k is a positive integer specifying the number of lattice points in the k'th direction, and d is the (effective) dimension of configuration space. For models with continuous symmetries (for instance rotational ones) the effective dimension may be chosen smaller than the physical one, by separation of variables. Likewise, discrete symmetries may can used to reduce the size of configuration space that this lattice must approximate.

In Python programs, quantities like the wave function ψ and the potential V are defined as floating point NumPy arrays of shape **`shape`**.

3.2. Edge Lengths and Offsets

The geometric extent of the selected region is specified by its edge lengths **`xe`**. This is a NumPy array of positive floating point numbers,

$$\texttt{xe} = \left[e_0, e_1, \ldots, e_{d-1}\right]. \tag{4}$$

A secondary quantity, derived from **`xe`** and **`shape`** is the elementary lattice cell,

$$\texttt{dx} = \texttt{xe}/\texttt{shape} = \left[e_0/s_0, e_1/s_1, \ldots, e_{d-1}/s_{d-1}\right]. \tag{5}$$

The absolute positioning of the region, with respect to some fixed coordinate system, is specified by a NumPy array of floating point numbers,

$$\texttt{xo} = \left[x_0, x_1, \ldots, x_{d-1}\right]. \tag{6}$$

This is defined as the position of the "lower left" corner of the selected region. The placement of the lattice points within the region still needs to be specified, as will be discussed below.

3.3. Boundary Conditions

The restriction to finite regions of space requires imposition of boundary conditions. For regions of rectangular shape (generalized to arbitrary dimensions), as considered here, the perhaps simplest choice is *periodic* boundary conditions in each directions. This may be viewed as a topological property of configuration space itself. Other boundary conditions are really properties of functions defined on this space, as specifications of how the functions should be extended beyond the boundary. Two natural choices are *symmetric* and *anti-symmetric* extensions. With a lattice approximation a further distinction can be made, related to how the lattice points are positioned relative to the boundary.

In this connection, it is natural to consider the cases handled by the trigonometric cousins of the fast Fourier transform (FFT). In the one-dimensional case the extension may be symmetric or anti-symmetric with respect to a boundary, which is situated either (i) at a lattice point, or (ii) midway between two lattice points. Thus, at each boundary there is 2×2 matrix of possibilities, as indicated by **Table 1**.

TABLE 1 | Individual boundary conditions covered by standard discrete trigonometric transforms (**DCT** and **DST**).

Function extension	Symmetric	Anti-symmetric
Boundary at lattice point	"S"	"A"
Boundary midway between points	"s"	"a"

With two boundaries there are altogether $4 \times 4 = 16$ possibilities. However, the routines in **`scipy.fftpack`** (**`dct`** and **`dst`** of types I–IV) only implement cases where both options come from the same row of **Table 1**. With the periodic extension **`P`** in addition, one ends up with a set of nine possibilities in each direction:

$$\mathcal{B} = \left\{ \texttt{'PP'}, \texttt{'SS'}, \texttt{'SA'}, \texttt{'AS'}, \texttt{'AA'}, \texttt{'ss'}, \texttt{'sa'}, \texttt{'as'}, \texttt{'aa'} \right\}. \tag{7}$$

Hence, the numerical model must be further specified by a Python tuple of two-character strings, defining the selected boundary condition in all directions,

$$\mathbf{bc} = \left(b_0, b_1, \ldots, b_{d-1}\right) \tag{8}$$

with each $b_k \in \mathcal{B}$ (or in an enlarged set of possibilities).

3.4. Lattice Positions. Dual Lattice Squared Positions

When **`bc`** is given, one may automatically calculate the positions of all lattice points

$$\boldsymbol{xlat} = \left(\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_{d-1}\right), \tag{9}$$

provided **`shape`**, **`xe`**, and **`xo`** are also known. In Equation (9), the property ***xlat*** is a tuple of one-dimensional arrays. For illustration, consider the case of a 3-dimensional lattice of shape (s_x, s_y, s_z). Then ***xlat*** is a Python tuple $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$, where $\mathbf{X}$ is a **`numpy`** array of shape $(s_x, 1, 1)$, $\mathbf{Y}$ is a **`numpy`** array of shape $(1, s_y, 1)$, and $\mathbf{Z}$ is a **`numpy`** array of shape $(1, 1, s_z)$. These are all one-dimensional arrays, but their shape information implies that (for instance) the Python expression $\mathbf{X}{*}\mathbf{Y}$ automatically evaluates to a **`numpy`** array of shape $(s_x, s_y, 1)$.

A Python function $V(x, y, z)$, defined by an expression that can involve "standard" functions, may then be evaluated on the complete lattice by the short and simple expression $V(*\boldsymbol{xlat})$. When V depends on all its arguments, the result will be a **`numpy`** array of shape (s_x, s_y, s_z).

In general, when Fourier transforming a periodic function $f(\boldsymbol{x})$, where $\boldsymbol{x}$ takes values on some discrete lattice, the result becomes another periodic function $\tilde{f}(\boldsymbol{k})$, where $\boldsymbol{k}$ takes values on another discrete lattice (the dual lattice/reciprocal space). Modulo an overall scaling, a set of $\boldsymbol{k}$-values (labeling the points of some complete, minimal subdomain to be extended by periodicity) can be defined such that $f(\boldsymbol{x} + \boldsymbol{a})$ transforms to $\mathrm{e}^{-\mathrm{i}\boldsymbol{k}\cdot\boldsymbol{a}}\tilde{f}(\boldsymbol{k})$. A natural choice for that minimal domain is, in physicists language, the *first Brillouin zone* (this choice may still leave a somewhat arbitrary selection of boundary points to be included). On this subdomain of the dual lattice, derivatives can be defined as the multiplication operators $-\mathrm{i}\boldsymbol{k}$. But these operators must still be extended to the full dual lattice by periodicity. The common stensil expressions for lattice derivatives correspond to the lowest Fourier components of the (periodically extended) multiplication operator $-\mathrm{i}\boldsymbol{k}$.

For the other (discrete trigonometric) transformations a complication arises, because a derivation also induces a transposition of the boundary conditions in $\mathcal{B}$. However, *two* derivations in the same direction leave the boundary conditions unchanged, and hence can be represented as a multiplication operator $\boldsymbol{q}$ on the transformed functions. Let ∂_k be shorthand notation for $\partial/\partial x_k$. The previous conclusion implies that all operators of the form $F(\partial_0^2, \partial_1^2, \ldots, \partial_{d-1}^2)$ can be evaluated through multiplications and fast discrete transforms,

$$F(\partial_0^2, \partial_1^2, \ldots, \partial_{d-1}^2) = \mathcal{T}^{-1} F(q_0, q_1, \ldots, q_{d-1}) \mathcal{T}. \tag{10}$$

We have implemented code that performs $\mathcal{T}$ and $\mathcal{T}^{-1}$ through a sequence of discrete trigonometric or fast Fourier transforms, dependent on **bc** and the other parameters. Analogous to the arrays ***xlat*** of lattice positions (Equation 9), one may automatically calculate similar arrays of squared positions for reciprocal lattice,

$$\boldsymbol{qlat} = (\mathbf{Q}_0, \mathbf{Q}_1, \ldots, \mathbf{Q}_{d-1}). \tag{11}$$

3.5. Lattice Laplacian. Stensil Representations

Instead of relying on FFT type transforms, one may directly construct discrete approximations (stencils) of the Laplace operator, and similar differential operators. The simplest implementation of a lattice Laplacian in one dimension is obtained by use of the formula

$$\frac{d^2\psi}{dx^2}(x_n) \approx \frac{\psi(x_n + \delta x) - 2\psi(x_n) - \psi(x_n - \delta x)}{\delta x^2}, \tag{12}$$

where δx is the distance between nearest-neighbor lattice points. The formal discretizations error of this approximation is of order δx^2. By summing such expression in d orthogonal directions one finds the $(2d+1)$-stensil expression for the lattice Laplacian.

A more accurate approximation is the $(4d+1)$-stencil,

$$\boldsymbol{\Delta}\varphi(\boldsymbol{x_n}) \approx \sum_{k=0}^{d-1} \frac{-\varphi(\boldsymbol{x_n} + 2\boldsymbol{\delta}_k) + 16\varphi(\boldsymbol{x_n} + \boldsymbol{\delta}_k) - 30\varphi(\boldsymbol{x_n}) + 16\varphi(\boldsymbol{x_n} - \boldsymbol{\delta}_k) - \varphi(\boldsymbol{x_n} - 2\boldsymbol{\delta}_k)}{12|\boldsymbol{\delta}_k|^2}. \tag{13}$$

Here $\boldsymbol{\delta}_k$ denotes a vector of length $|\boldsymbol{\delta}_k|$ pointing in positive k-direction.

An arbitrary (short-range) position independent operator O can in general be represented by a stensil $s_O(\boldsymbol{b})$ such that

$$(O\psi)(\boldsymbol{x_n}) = \sum_{\boldsymbol{b}} s_O(\boldsymbol{b})\, \psi(\boldsymbol{x_{n-b}}). \tag{14}$$

When $\boldsymbol{n} - \boldsymbol{b}$ falls outside the lattice, the value of $\psi(\boldsymbol{x_{n-b}})$ must is interpreted according to the boundary conditions **`bc`**. This can again be automatized. We have implemented an algorithm for this, currently only for 5 of the 9 cases in $\mathcal{B}$ in each direction, but for an arbitrary number of directions.

The various ways to approximate the Laplace operator, or more generally the kinetic energy operator, is made available through the selector **ke**, whose value is currently limited to the set of options {'`2dplus1`', '`4dplus1`', '`fftk2`'}. The last of these options is discussed in section 5.

4. SIMPLE APPLICATIONS

In this section we will demonstrate some applications of our automatic code. The main requirement is that in each case only a set of parameters and selectors should be provided, with no coding required by the application itself. This should be sufficient to generate eigenvalues E_n as requested, and optionally also the associated eigenfunctions (an issue which we have not yet tested).

4.1. Example: One-Dimensional Harmonic Oscillator

Consider the eigenvalue problem of the one-dimensional harmonic oscillator,

$$-\psi_n''(x) + x^2\, \psi_n(x) = E_n\, \psi_n(x). \tag{15}$$

The eigenvalues are $E_n = 2n + 1$ for $n = 0, 1, \ldots$, and the extent of the wavefunction $\psi_n(x)$ can be estimated from the requirement that a classical particle of energy E_n is restricted to $x^2 \leq E_n$. A quantum particle requires a little more space than the classically restricted one.

For a numerical analysis we provide the parameters

`shape` $= (128,)$, **`bc`** $=$ ('`a`', '`a`'), **`xe`** $= (25,)$, **`xo`** $= (-12.5)$,
`V` $=$ **`lambda`** $x : x**2$,

selects the 3-stensil approximation for **T** (default choice), and the dense matrix solver **`eigvalsh`** (default choice). This instantly returns 128 eigenvalues as plotted in **Figure 1**. We may easily change **`shape`** to (1024), for a much better result. The potential for additional explorations, without any coding whatsoever, should be obvious.

For a better quantitative assessment of the accuracy obtained we plot some energy differences, $E_n^{(\mathrm{exact})} - E_n$, in **Figure 2**.

This *brute force* method leads to a dramatic increase in memory requirement with increasing lattice size. For a lattice with $N = 2^m$ sites, the matrix requires storage of 4^m double precision (8 byte) numbers. For $m = 13$ this corresponds to about $\frac{1}{2}$ Gb of memory, for $m = 14$ about 2 Gb. The situation becomes even worse in higher dimensions.

Assuming that we are only interested some of the lowest eigenvalues, an alternative approach is to calculate these by the iterative routine **`eigsh`** from **`scipy.sparse.linalg`**. This allows extension to larger lattices, as shown in **Figure 3**.

With a sparse eigenvalue solver the calculation becomes limited by available computation time, which in many cases is a

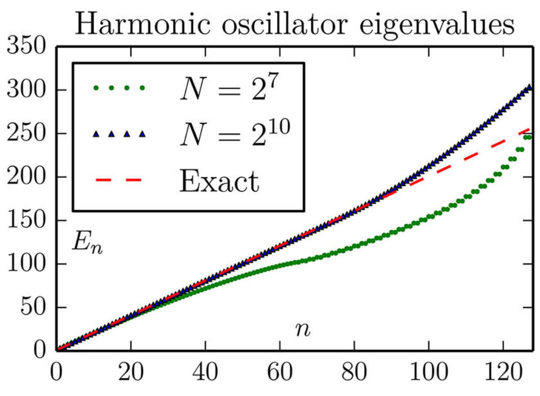

FIGURE 1 | The 128 lowest eigenvalues of Equation (15), computed with the standard 3-stensil approximation for the Laplace operator (here the kinetic energy *T*). The parameters are chosen to illustrate two typical effects: With the **bc**=(**a**, **a**) boundary conditions the harmonic oscillator potential is effectively changed to $V = \infty$ for $x \geq 12.5$, thereby modifying the behavior of extended (highly exited) states. The effect of this is to increase the eigenenergies of such states, to a behavior more similar to a particle-in-box. This is visible for $n \gtrsim 80$. The effect of using the 3-stensil approximation for *T* is to change the spectrum of this operator from k^2 to (the slower rising) $(2/\delta x)^2 \sin^2(k\delta x/2)$. This is visible in the sub-linear rise of the spectrum for $N = 2^7$.

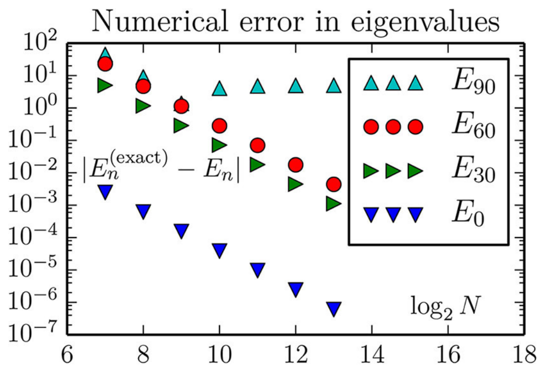

FIGURE 2 | The discretizations error of energy eigenvalues when using the standard 3-stensil approximation for the one-dimensional Laplace operator (here the kinetic energy *T*). There is no improvement in E_{90} beyond a certain lattice size *N*, because the corresponding oscillator state is too large for the geometric region. Hence, for improved accuracy of higher eigenvalues one should instead increase the **xe**, while maintaining $\mathbf{xo} = -\mathbf{xe}/2$. For the other states the improvement is consistent with the expectation of an error proportional to δx^2. This predicts an accuracy improvement of magnitude $2^{12} = 4,096$ when the number of lattice sites increases from $N = 2^7$ to $N = 2^{13}$ for a fixed geometry. The eigenvalues are computed by the dense matrix routine `eigvalsh` from `scipy.linalg`.

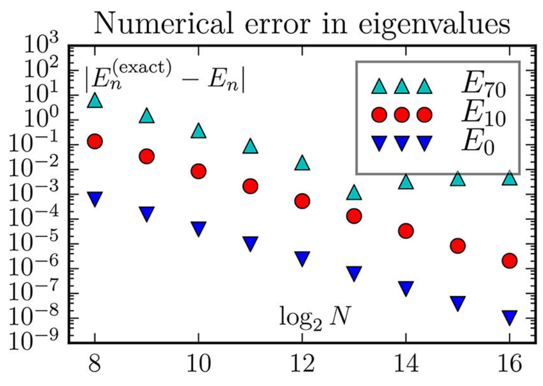

FIGURE 3 | The discretizations error computed by the routine `eigsh` from `scipy.sparse.linalg`. For a fixed lattice size the discretizations error is essentially the same as with dense matrix routines. However, with a memory requirement proportional to the lattice size (instead of its square) it becomes possible to go to much larger lattices. This figure also demonstrates (E_{70}) that the error can be limited by boundary effects instead of the finite discretization length δx.

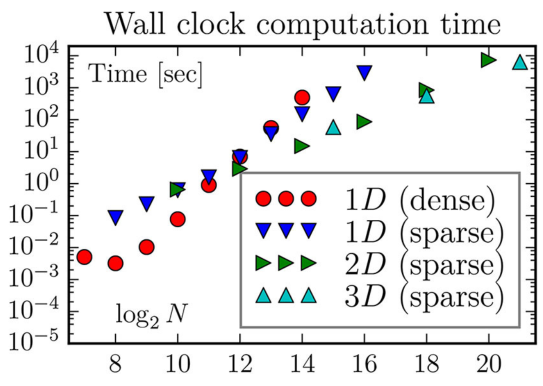

FIGURE 4 | The wall clock time used to find the lowest 128 eigenvalues, for various systems and methods. We have also used the dense matrix routine `eigvalsh` to compute the eigenvalues of a $2^7 \times 2^7$ ($N = 2^{14}$) two-dimensional lattice; not unexpected it takes the same time as for a 2^{14} one-dimensional lattice. Somewhat surprisingly, with `eigsh` it is much faster to find the eigenvalues for two-dimensional lattice than for a one-dimensional with the same number of sites, and somewhat faster to find the eigenvalues for a three-dimensional lattice than for the two-dimensional with the same number of sites.

much weaker constraint: With proper planning and organization of calculations, the relevant timescale is the time to analyze and publish results (i.e., weeks or months). The computation time is nevertheless of interest (it shouldn't be years). We have measured the wall clock time used to perform the computations for **Figures 2**, **3**, performed on a 2012 Mac Mini with 16 Gb of memory, and equipped with a parallelized `scipy` library. Hence, the `eigvalsh` and `eigsh` routines are running with four threads. The results are plotted in **Figure 4**.

Here we have used the `eigsh` routine in the most straightforward manner, using default settings for most parameters. This means, in particular, that the initial vector for the iteration (and the subsequent set of trial vectors) may not be chosen in a optimal manner for our category of problems. It is interesting to observe that `eigsh` works better for higher-dimensional problems. The (brief) `scipy` documentation [17] says that the underlying routines works best when computing eigenvalues of largest magnitude, which are of no physical interest for our type of problems. It is our experience that the

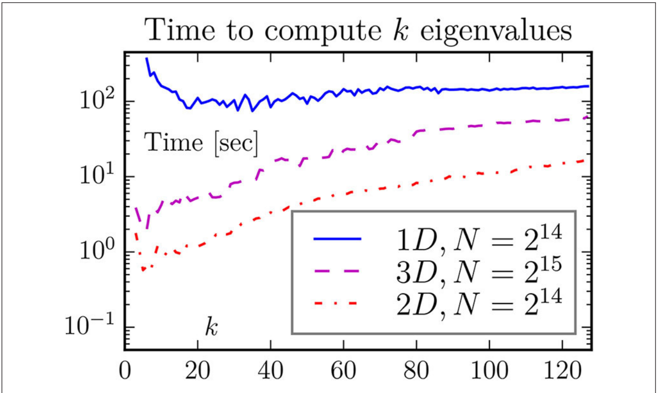

FIGURE 5 | One may think that it takes longer to compute more eigenvalues. This is not always the case when the number of eigenvalues is small, as demonstrated by this figure. The default choice of **eigsh** is to compute $k = 6$ eigenvalues. For our two- and three-dimensional problems this looks close to the optimal value, but it is too low for the one-dimensional problem.

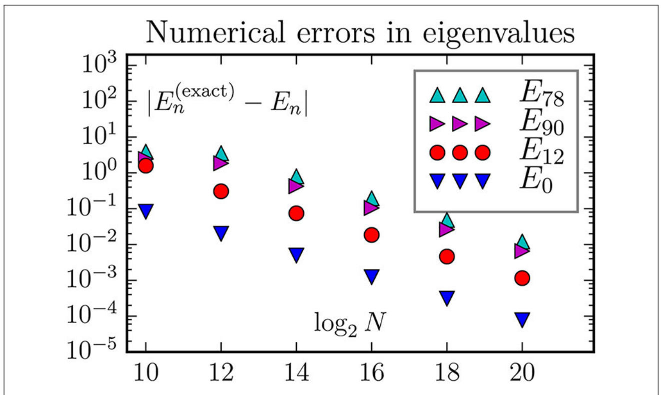

FIGURE 6 | The discretization error of energy eigenvalues when using the standard 5-stensil approximation for the two-dimensional Laplace operator. Exactly, the states E_{78} and E_{90} are the two edges of a 13-member multiplet with energy 26, and the state E_{12} is the middle member of a 5-member multiplet with energy 10. With the chosen parameters all states considered a well confined inside the geometric region; hence we do not observe any boundary correction effects.

suggested strategy, of using the *shift-invert* mode instead, does not work right out-of-the-box for problems of interesting size (i.e., where dense solvers cannot be used). We were somewhat surprised to observe that the computation time may *decrease* if the number of computed eigenvalues increases (cf. **Figure 5**).

4.2. Example: 2- and 3-Dimensional Harmonic Oscillators

The d-dimensional harmonic oscillator

$$\left[-\Delta + r^2\right] \psi_n(\mathbf{r}) = E_n\, \psi_n(\mathbf{r}), \tag{16}$$

has eigenvalues $E_n = (d + 2n)$, for $n = 0, 1, \ldots$. The degeneracy of the energy level E_n is $g_n = (n + 1)$ in two dimensions, and $g_n = \frac{1}{2}(n + 1)(n + 2)$ in three dimensions[1]. This degeneracy may be significantly broken by the numerical approximation. For a numerical solution we only have to change the previous parameters slightly:

$$\begin{aligned} &\textbf{shape} = (128,) * \text{dim},\ \textbf{bc} = ((\,'\mathtt{a}',\ '\mathtt{a}'),) * \text{dim}, \\ &\textbf{xe} = (25,) * \text{dim},\ \textbf{xo} = (-12.5,) * \text{dim}, && (17a) \\ &\textbf{V} = \textbf{lambda}\ x, y : x * *2 + y * *2 \quad (\text{dim} = 2), && (17b) \\ &\textbf{V} = \textbf{lambda}\ x, y, z : x * *2 + y * *2 + z * *2 \quad (\text{dim} = 3), && (17c) \end{aligned}$$

for dim = 2, 3.

As already discussed, the routine **eigsh** works somewhat faster in higher dimensions than in one dimension (for the same *total* number N of lattice points). The corresponding discretizations errors are shown in **Figures 6**, **7**.

The discretizations error continues to scale like δx^2. This means that a reduction of this error by a factor $2^2 = 4$ requires an increase in the number of lattice points by a factor 2^d in d dimensions. This means that is becomes more urgent

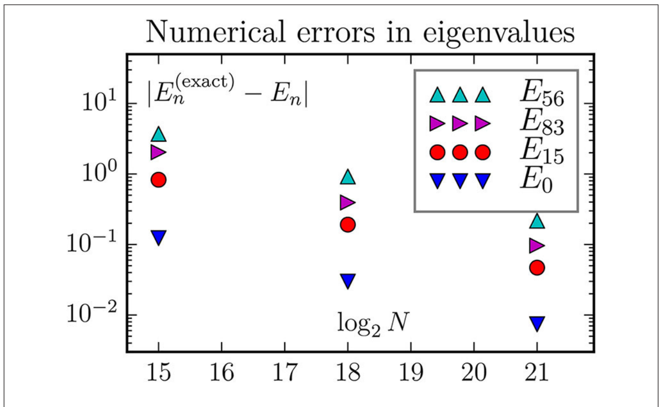

FIGURE 7 | The discretization error of energy eigenvalues when using the standard 7-stensil approximation for the three-dimensional Laplace operator. Exactly, the states E_{56} and E_{83} are the two edges of a 28-member multiplet with energy 15, and the state E_{15} is the middle member of a 10-member multiplet with energy 9.

to use a better representation of the Laplace operator in higher dimensions. Fortunately, as we shall see in the next sections, better representations are available for our type of problems.

5. FFT CALCULATION OF THE LAPLACE OPERATOR

One improvement is to use the reflection symmetry of each axis ($x \to -x$, $y \to -y$, etc.) to reduce the size of the spatial domain. This reduces δx by a half, without changing the number of lattice points.

A much more dramatic improvement is to use some variant of a Fast Fourier Transform (FFT): After a Fourier transformation, $\psi(\mathbf{r}) \to \tilde{\psi}(\mathbf{k})$, the Laplace operator turns into

[1]The general formula is $g_n = \binom{d-1+n}{d-1}$.

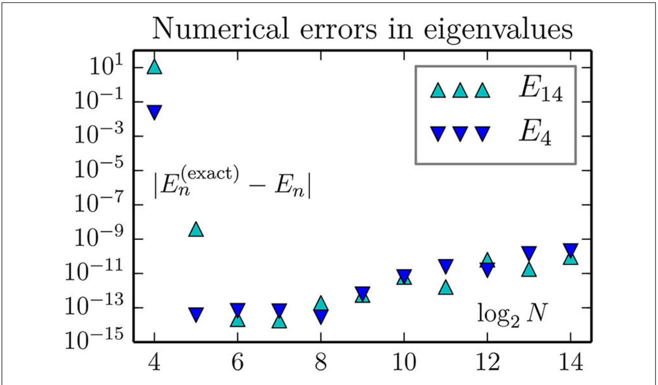

FIGURE 8 | With a FFT representation of the Laplace operator the discretization error drops exceptionally fast with $\delta x \propto N^{-1}$. When it becomes "small enough" the effect of numerical roundoff becomes visible; the latter leads to an *increase* in error with δx. The results in this figure is for a one-dimensional lattice, but the behavior is the same in all dimensions. The lesson is that we should make δx "small enough" (which in general may be difficult to determine *a priori*), but not smaller. It may also be possible to rewrite the eigenvalue problem to a form with less amplification of roundoff errors.

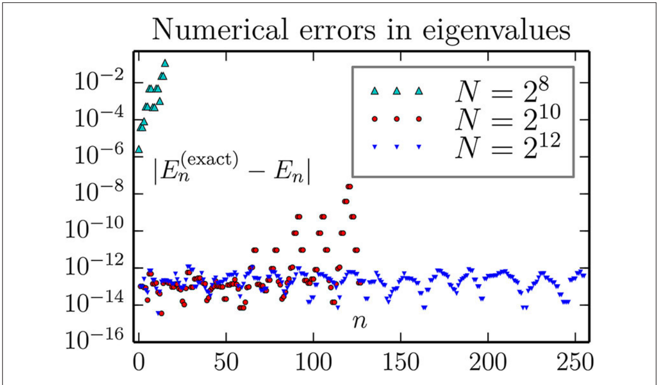

FIGURE 10 | Accuracy of computed eigenvalues for a 2*D* oscillator, using the FFT approximation for kinetic energy *T*. As can be seen, a large number of the lowest eigenvalues can be computed to an absolute accuracy in the range 10^{-14}–10^{-12} with a lattice of size $2^6 \times 2^6$. We observe not improvement by going to $2^7 \times 2^7$ lattice, but no harm either (except for an increase in the wall clock execution time from about 3 to 30 s for each combination of boundary conditions).

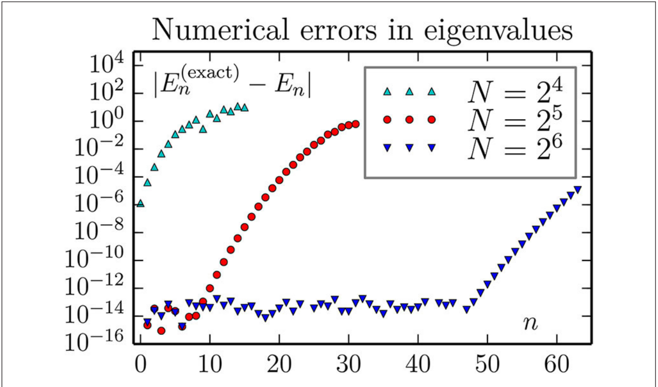

FIGURE 9 | Accuracy of computed eigenvalues for a 1*D* oscillator, using the FFT approximation for kinetic energy *T*. This figure may suggest that an increase in the number of lattice size *N* will lead to a accurate treatment of states with higher *n*. Our findings are that this is *not* the case: The results for $N = 2^7$ and $N = 2^8$ have essentially the same behavior as for $N = 2^6$.

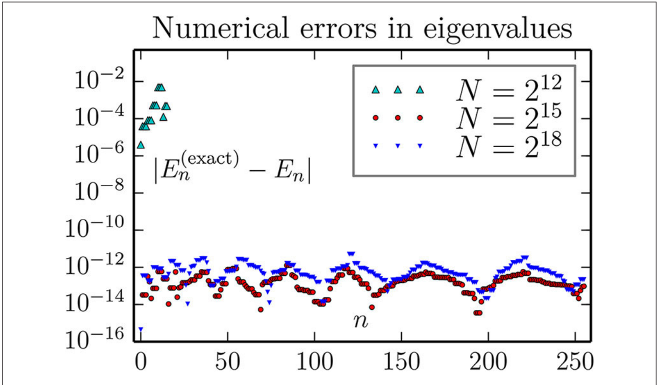

FIGURE 11 | Accuracy of computed eigenvalues for a 3*D* oscillator, using the FFT approximation for kinetic energy *T*. As can be seen, a large number of the lowest eigenvalues can be computed to an absolute accuracy in the range 10^{-14} to 10^{-12} with lattice of size $2^6 \times 2^6 \times 2^6$. We observe no improvement by going to $2^7 \times 2^7 \times 2^7$ lattice, but no harm either (except for an increase in the wall clock execution time from about 6 to 95 min for each combination of boundary conditions).

multiplication operator,

$$(-\Delta\psi)(\boldsymbol{r}) \to \boldsymbol{k}^2\,\tilde{\psi}(\boldsymbol{k}).$$

This means that application of the Laplace operator can be represented by (i) a Fourier transform, followed by (ii) multiplication by $\boldsymbol{k}^2$, and finally (iii) an inverse Fourier transform. Essentially the same procedure works for the related trigonometric transforms.

For rectangular lattices, these options can also be implemented as practical procedures, due to the existence of efficient and accurate[2] algorithms for discrete Fourier and trigonometric transforms. The time to perform the above procedure is not very much longer than the corresponding stensil operations. The benefit is that the Laplace operator becomes exact on the space of functions which can be represented by the modes included in the discrete transform.

We have coded this FFT-type representation of the Laplace operator for the various types of boundary conditions listed in **Table 1**. This possibility can be chosen as an option for the kinetic energy selector, **ke**. The obtainable accuracy through this option increases dramatically, as illustrated in **Figures 8–11**. As shown in **Figure 8**, a decrease of the lattice length δx does not necessarily

[2]The error of a back-and-forth FFT is a few times the numerical accuracy, i.e., in the range 10^{-14} to 10^{-15}. with double precision numbers. However, when an error of this order is multiplied by $\boldsymbol{k}^2$ it can be amplified by several orders of magnitude. Hence, the range of $\boldsymbol{k}^2$-values should not be chosen significantly larger than required to represent $\psi(\boldsymbol{r})$ to sufficient accuracy.

TABLE 2 | The 10 lowest eigenvalues of the quantum anharmonic oscillator, calculated to high precision by the method described in [14], from the Schrödinger equation $\left(-\frac{d^2}{d\xi^2}+\xi^4\right)\psi_n(\xi)=\varepsilon_n\psi_n(\xi)$.

n	ε_n
1	1.060 362 090 484 182 899 647 046 016 693
2	3.799 673 029 801 394 168 783 094 188 513
3	7.455 697 937 986 738 392 156 591 347 186
4	11.644 745 511 378 162 020 850 373 281 371
5	16.261 826 018 850 225 937 894 954 430 385
6	21.238 372 918 235 940 024 149 711 113 589
7	26.528 471 183 682 518 191 813 828 183 681
8	32.098 597 710 968 326 634 272 106 438 332
9	37.923 001 027 033 985 146 516 378 551 910
10	43.981 158 097 289 730 785 318 113 752 827

The eigenfunctions obey the (anti-)symmetry property, $\psi_n(\xi)=(-1)^{n-1}\,\psi_n(-\xi)$.

lead to a more accurate result. We attribute this to an enhanced amplification of roundoff errors.

It might be that the harmonic oscillator systems are particularly favorable for application of the FFT representation. One important feature is that the Fourier components of the harmonic oscillator wave functions vanishes exponentially fast, like $e^{-\boldsymbol{k}^2/2}$, with increasing wave-numbers $\boldsymbol{k}^2$. This feature is shared with all eigenfunctions of polynomial potential Schrödinger equations, but usually with different powers of $\boldsymbol{k}$ in the exponent, which quantitatively leads to a somewhat different behavior. Furthermore, the onset of exponential decay will occur for larger values of $\boldsymbol{k}^2$ for the more excited states (i.e., with larger eigenvalue numbers).

For systems with singular wavefunctions the corresponding Fourier components may vanish only algebraically with $\boldsymbol{k}^2$. An equally dramatic increase in accuracy cannot be expected for such cases.

6. ANHARMONIC OSCILLATORS

Our general setup allows for any computable potential, by simply changing the definition of the function assigned to **V** (This does not mean that every potential will lead to a successful calculation of eigenvalues). For demonstration and comparison purposes, like here, one encounters the problem that the exact answers are no longer known. This makes it more difficult to assess the accuracy and other qualities of the code. As an example where some instructive comparisons are possible, we consider the two-dimensional anharmonic oscillator obeying the Schrödinger equation,

$$\frac{1}{2}\left(-\frac{d^2}{dx^2}-\frac{d^2}{dy^2}+x^4+6\,x^2y^2+y^4\right)\Psi_E(x,y)=E\,\Psi(x,y). \tag{18}$$

By construction, this problem has separable solutions of the form

$$\Psi_E(x,y)=\psi_m(\xi)\,\psi_n(\eta),\ \text{with}\ \ \xi=(x+y)/\sqrt{2},\ \ \eta=(x-y)/\sqrt{2}, \tag{19}$$

TABLE 3 | The 22 lowest eigenvalues E of the two-dimensional quantum anharmonic oscillator, as defined by the Schrödinger equation $\frac{1}{2}\left(-\frac{d^2}{dx^2}-\frac{d^2}{dy^2}+x^4+6x^2y^2+y^4\right)\Psi_E(x,y)=E\,\Psi_E(x,y)$, displayed to 30 decimals accuracy.

(P_x, P_y)	Comp	E
(S,S)	$\varepsilon_1+\varepsilon_1$	2.120 724 180 968 365 799 294 092 033 385
(S,A)	$\varepsilon_1+\varepsilon_2$	4.860 035 120 285 577 068 430 140 205 205
(A,S)	$\varepsilon_1+\varepsilon_2$	4.860 035 120 285 577 068 430 140 205 205
(S,S)	$\varepsilon_2+\varepsilon_2$	7.599 346 059 602 788 337 566 188 377 025
(S,S)	$\varepsilon_1+\varepsilon_3$	8.516 060 028 470 921 291 803 637 363 878
(A,A)	$\varepsilon_1+\varepsilon_3$	8.516 060 028 470 921 291 803 637 363 878
(S,A)	$\varepsilon_2+\varepsilon_3$	11.255 370 967 788 132 560 939 685 535 698
(A,S)	$\varepsilon_2+\varepsilon_3$	11.255 370 967 788 132 560 939 685 535 698
(S,A)	$\varepsilon_1+\varepsilon_4$	12.705 107 601 862 344 920 497 419 298 064
(A,S)	$\varepsilon_1+\varepsilon_4$	12.705 107 601 862 344 920 497 419 298 064
(S,S)	$\varepsilon_3+\varepsilon_3$	14.911 395 875 973 476 784 313 182 694 372
(S,S)	$\varepsilon_2+\varepsilon_4$	15.444 418 541 179 556 189 633 467 469 884
(A,A)	$\varepsilon_2+\varepsilon_4$	15.444 418 541 179 556 189 633 467 469 884
(S,S)	$\varepsilon_1+\varepsilon_5$	17.322 188 109 334 408 837 542 000 447 077
(A,A)	$\varepsilon_1+\varepsilon_5$	17.322 188 109 334 408 837 542 000 447 077
(S,A)	$\varepsilon_3+\varepsilon_4$	19.100 443 449 364 900 413 006 964 628 557
(A,S)	$\varepsilon_3+\varepsilon_4$	19.100 443 449 364 900 413 006 964 628 557
(S,A)	$\varepsilon_2+\varepsilon_5$	20.061 499 048 651 620 106 678 048 618 897
(A,S)	$\varepsilon_2+\varepsilon_5$	20.061 499 048 651 620 106 678 048 618 897
(S,A)	$\varepsilon_1+\varepsilon_6$	22.298 735 008 720 122 923 796 757 130 281
(A,S)	$\varepsilon_1+\varepsilon_6$	22.298 735 008 720 122 923 796 757 130 281
(S,S)	$\varepsilon_4+\varepsilon_4$	23.289 491 022 756 324 041 700 746 562 742

This equation is separable in terms of two identical one-dimensional problems, with eigenvalues ε_m as listed in ***Table 2****. Hence each eigenvalues E is composed of two eigenvalues ε_m, ε_n as indicated in the second column. The reflection parities (P_x, P_y) listed in the first column indicate how the wavefunctions $\Psi_E(x,y)$ can be chosen symmetric (S) or anti-symmetric (A) under the reflections $x\to -x$ or $y\to -y$.*

where the factors ψ obey a one-dimensional equation,

$$\left(-\frac{d^2}{d\xi^2}+\xi^4\right)\psi_m(\xi)=\varepsilon_m\,\psi_m(\xi), \tag{20}$$

and $E=\varepsilon_m+\varepsilon_n$. As mentioned in the introduction, equations like the latter have been quite intensely studied in the literature. Accurate values for the even parity eigenvalues of Equation (20) can for instance be found in Table 1 of [9]. In **Table 2**, we list the 10 lowest eigenvalues to 30 decimals precision, calculated by the very-high-precision method described in [14]. Hence, for practical purposes all ε_m of interest can be considered exactly known. This means that the eigenvalues E of Equation (18) can also be considered exactly known. We list the 22 lowest ones of them in **Table 3**. These are the values we want to compare against the standard solution methods. The latter make no use of the separability property of the problem, which anyway is destroyed by the lattice approximation.

The first column of **Table 3** associates a symmetry classification (P_x, P_y) to each eigenvalue E, or rather to its corresponding eigenfunction $\Psi_E(x,y)$. Since Equation (18) are

TABLE 4 | Numerical calculations of the lowest eigenvalues of the two-dimensional quantum anharmonic oscillator, by various approximations and lattice sizes.

(P_x, P_y)	**Stensil (2^{10} x 2^{10})**	**"FFT" (2^4 x 2^4)**	**"FFT" (2^5 x 2^5)**	**"FFT" (2^7 x 2^7)**
(S, S)	$2.120\,\underline{5}74\,864\,327$	$2.12\underline{1}\,724\,631\,908$	$2.120\,724\,180\,968$	$2.120\,724\,180\,96\underline{9}$
(S, A)	$4.85\underline{9}\,463\,304\,350$	$4.86\underline{3}\,978\,042\,739$	$4.860\,035\,120\,2\underline{7}6$	$4.860\,035\,120\,286$
(A, S)	$4.85\underline{9}\,463\,304\,350$	$4.86\underline{3}\,978\,042\,731$	$4.860\,035\,120\,2\underline{6}9$	$4.860\,035\,120\,28\underline{9}$
(S, S)	$7.59\underline{7}\,839\,625\,245$	$7.5\underline{8}0\,886\,360\,302$	$7.599\,346\,06\underline{4}\,273$	$7.599\,346\,059\,59\underline{9}$
(S, S)	$8.5\underline{1}4\,505\,169\,411$	$8.\underline{4}43\,877\,132\,728$	$8.516\,060\,03\underline{3}\,426$	$8.516\,060\,028\,46\underline{7}$
(A, A)	$8.5\underline{1}4\,700\,940\,122$	$8.\underline{4}66\,735\,662\,572$	$8.516\,060\,02\underline{4}\,420$	$8.516\,060\,028\,46\underline{7}$
(S, A)	$11.2\underline{5}2\,295\,795\,135$	$11.\underline{0}91\,953\,034\,554$	$11.255\,371\,0\underline{2}7\,420$	$11.255\,370\,967\,79\underline{2}$
(A, S)	$11.2\underline{5}2\,295\,795\,137$	$11.\underline{0}91\,953\,034\,552$	$11.255\,371\,0\underline{2}7\,446$	$11.255\,370\,967\,7\underline{8}4$
(S, A)	$12.7\underline{0}2\,160\,201\,238$	$12.71\underline{3}\,861\,518\,776$	$12.705\,107\,6\underline{0}5\,729$	$12.705\,107\,601\,8\underline{6}8$
(A, S)	$12.7\underline{0}2\,160\,201\,248$	$12.71\underline{3}\,861\,518\,777$	$12.705\,107\,6\underline{0}5\,757$	$12.705\,107\,601\,86\underline{1}$
(S, S)	$14.90\underline{5}\,839\,565\,650$	$1\underline{6}.827\,495\,880\,048$	$14.911\,396\,\underline{4}13\,962$	$14.911\,395\,875\,97\underline{0}$
(S, S)	$15.43\underline{8}\,616\,444\,914$	$1\underline{7}.044\,711\,067\,731$	$15.444\,418\,\underline{9}09\,471$	$15.444\,418\,541\,17\underline{8}$
(A, A)	$15.43\underline{9}\,522\,886\,891$	$1\underline{4}.126\,665\,759\,659$	$15.444\,41\underline{8}\,063\,518$	$15.444\,418\,541\,17\underline{5}$
(S, S)	$17.31\underline{6}\,965\,583\,271$	$1\underline{8}.162\,997\,853\,055$	$17.322\,188\,\underline{1}95\,788$	$17.322\,188\,109\,3\underline{3}7$
(A, A)	$17.31\underline{7}\,047\,769\,535$	$1\underline{6}.740\,653\,634\,905$	$17.322\,187\,\underline{9}29\,414$	$17.322\,188\,109\,32\underline{8}$
(S, A)	$19.0\underline{9}1\,567\,414\,142$	$1\underline{8}.071\,825\,773\,508$	$19.100\,44\underline{2}\,397\,522$	$19.100\,443\,449\,36\underline{0}$
(A, S)	$19.0\underline{9}1\,567\,414\,151$	$1\underline{8}.071\,825\,773\,501$	$19.100\,44\underline{2}\,397\,503$	$19.100\,443\,449\,36\underline{1}$
(S, A)	$20.0\underline{5}3\,053\,266\,697$	$20.\underline{2}44\,253\,292\,135$	$20.061\,49\underline{6}\,254\,183$	$20.061\,499\,048\,64\underline{8}$
(A, S)	$20.0\underline{5}3\,053\,266\,716$	$20.\underline{2}44\,253\,292\,132$	$20.061\,49\underline{6}\,254\,153$	$20.061\,499\,048\,64\underline{8}$
(S, A)	$22.29\underline{0}\,449\,617\,012$	$2\underline{2}.809\,096\,276\,441$	$22.298\,734\,\underline{8}48\,064$	$22.298\,735\,008\,720$
(A, S)	$22.29\underline{0}\,449\,617\,033$	$2\underline{2}.809\,096\,276\,438$	$22.298\,734\,\underline{8}48\,071$	$22.298\,735\,008\,71\underline{8}$
(S, S)	$23.2\underline{7}6\,097\,201\,666$	$\underline{3}5.427\,997\,419\,504$	$23.289\,48\underline{6}\,014\,610$	$23.289\,491\,022\,74\underline{9}$

The accuracy obtained is indicated by an underscore of the first inaccurate position (when taking roundoffs into account). The first column list the symmetry types (reflection parities) of the associated wavefunction.

invariant under reflections,

$$\mathcal{P}_x : x \to -x \quad \text{or} \quad \mathcal{P}_y : y \to -y,$$

all eigenfunctions can be constructed to transform symmetrically (S) or anti-symmetrically (A) under such reflections. For $m < n$, such a construction is

$$\Psi^{(\pm)}_{mn}(x, y) = \frac{1}{\sqrt{2}}\left[\psi_m(\xi)\,\psi_n(\eta) \pm \psi_n(\xi)\,\psi_m(\eta)\right]. \qquad (21a)$$

For $m = n$ there is only one possibility,

$$\Psi_{mm}(x, y) = \psi_m(\xi)\,\psi_m(\eta). \qquad (21b)$$

By use of the properties that

$$\psi_m(-\xi) = (-1)^{m-1}\psi(\xi), \quad \mathcal{P}_x : (\xi, \eta) \to -(\eta, \xi), \quad \text{and} \quad \mathcal{P}_y : (\xi, \eta) \to (\eta, \xi),$$

we find that

$$\begin{aligned}\Psi^{(\pm)}_{mn}(-x, y) &= \pm(-1)^{m+n}\Psi^{(\pm)}_{mn}(x, y), \quad \text{and} \\ \Psi^{(\pm)}_{mn}(x, -y) &= \pm\Psi^{(\pm)}_{mn}(x, y).\end{aligned} \qquad (22)$$

and further that $\Psi_{mm}(-x, y) = \Psi_{mm}(x, -y) = \Psi_{mm}(x, y)$. The conclusion is that in an exact calculation the states Ψ_{mn} will be double degenerate when $m \neq n$, with parities (P_x, P_y) equal to (S, S) and (A, A) when m, n are both even or both odd, otherwise with parities (S, A) and (A, S). The states Ψ_{mm} are singlets with parities (S, S). The first column of **Table 3** is constructed according to these rules.

Table 4 displays the results of some standard numerical solutions to Equation (18), "automagically" generated in the same way as the previous treatments of the harmonic (linear) oscillators. In the second column we show the results of using the minimal 5-point stensil approximation of the Laplace operator on a $1,024 \times 1,024$ lattice (approximating the whole space). The resulting numerical problem is solved with the **`eigsh`** sparse solver. The numerical accuracy is indicated by an underscore of the first inaccurate position, when taking proper roundoffs into account: The exact and numerical results are rounded off to the same number of digits, and compared; the underscore indicates the first position where the results differ.

As can be seen, the results are less than impressive, taking into account the amount computational work invested. One straightforward improvement is to utilize the reflection symmetries of the problem to reduce the magnitude of the problem (with the same lattice cell size δx^2) by a factor 4, or to reduce the lattice cell size δx^2 (with the same problem magnitude) by a factor 4. Another option is to use a higher order stensil approximation like (13). However, as already discussed in section 5, an even better option (for this class of problems) is to use a FFT type of approximation of the Laplace operator. The resulting eigenvalues are listed in columns 3–5, for various lattice sizes approximating the upper right quadrant ($x \geq 0,\ y \geq 0$) of space. For each lattice size the problem must be solved 4 times, with symmetric (S) and anti-symmetric (A) boundary conditions at the axes $x = 0$ and $y = 0$.

By symmetry under interchange, $x \leftrightarrow y$, we expect the (S, A) and (A, S) to give identical results (as long as the lattice approximation respects this symmetry). As can be seen, the numerical results satisfy the symmetry within a numerical accuracy of *few* $\times 10^{-12}$, regardless how close the results are to the exact values. The degeneracy of states with (S, S), respectively, (A, A) symmetry cannot be deduced in the same way from the lattice approximated problem. In the infinite space formulation the problem is separable, which in turn implies this degeneracy. However, the lattice approximation introduces boundaries that are non-factorizable in the (ξ, η)-coordinates. This means that the problem is no longer separable in the lattice approximation. As a result the degeneracy of the (S, S) and (A, A) energies are split by a much larger amount, of the same order as the difference between exact and numerical results. (In this case, the lattice problem could be made separable by a rotation of the lattice orientation by 45 degrees.)

We observe that even a $2^4 \times 2^4$ lattice with in the "FFT approximated" Laplace operator provide almost equally accurate results as a $2^{10} \times 2^{10}$ lattice with the 5-stensil approximation. The results from a $2^5 \times 2^5$ lattice seems more than sufficient for practical purposes (say compared to experimental obtainable accuracy), with little to be gained by further decrease of the lattice length δx.

The computation times for the "FFT approximation" are about 0.06, 0.8, and 75 s for respectively 16×16, 32×32, and 128×128 lattice sizes. For the same number of lattice points, the 5-stensil formulation may lead to somewhat shorter computation times. But this is completely offset by the need to work with a much larger number of lattice points: The computation time for the $1,024 \times 1,024$ stensil approximation was about 30 min.

The Python package described in this paper is available at [19].

DATA AVAILABILITY STATEMENT

The raw data [19] supporting the conclusions of this article will be made available by the authors, without undue reservation.

AUTHOR CONTRIBUTIONS

All authors listed have made a substantial, direct and intellectual contribution to the work, and approved it for publication.

ACKNOWLEDGMENTS

An early version of this work was presented at the IAENG International MultiConference of Engineers and Computer Scientists, Hong Kong March 18–20, 2015, as documented in [20].

AM would like to thank *Mathematics Teaching and Learning Research Group* within *The department of mathematics, Bodø, Nord University* for partial support.

REFERENCES

1. Dirac PAM. The fundamental equations of quantum mechanics. *Proc R Soc A*. (1925) **109**:642–53. doi: 10.1098/rspa.1925.0150
2. Schrödinger E. An undulatory theory of the mechanics of atoms and molecules. *Phys Rev*. (1926) **28**:1049–70. doi: 10.1103/PhysRev.28.1049
3. Pauli W. Über die Wasserstoffspektrum vom Standpunkt der neuen Quantenmechanik. *Zeitsch Phys*. (1926) **36**:336–63.
4. Heisenberg W. Über quantentheoretische Umdeutung kinematischer und mechanischer Beziehungen [Quantum-theoretical reinterpretation of kinematic and mechanical relations]. *Zeitsch Phys*. (1925) **33**:879–93.
5. Bender CM, Wu TT. Analytic structure of energy levels in a field-theory model. *Phys Rev Lett*. (1968) **21**:406–9. doi: 10.1103/PhysRevLett.21.406
6. Bender CM, Wu TT. Anharmonic oscillator. *Phys Rev*. (1969) **184**:1231–60. doi: 10.1103/PhysRev.184.1231
7. Bender CM, Wu TT. Large-order behavior of perturbation theory. *Phys Rev Lett*. (1971) **27**:461–5. doi: 10.1103/PhysRevLett.27.461
8. Bender CM, Wu TT. Anharmonic oscillator. II. A study of perturbation theory in large order. *Phys Rev D*. (1973) **7**:1620–36. doi: 10.1103/PhysRevD.7.1620
9. Bender CM, Olaussen K, Wang PS. Numerological analysis of the WKB approximation in large order. *Phys Rev D*. (1977) **16**:1740–8. doi: 10.1103/PhysRevD.16.1740
10. Zinn-Justin J, Jentschura UD. Multi-instantons and exact results I: conjectures, WKB expansions, and instanton interactions. *Ann Phys*. (2004) **313**:197–267. doi: 10.1016/j.aop.2004.04.004
11. Zinn-Justin J, Jentschura UD. Multi-instantons and exact results II: specific cases, higher-order effects, and numerical calculations. *Ann Phys*. (2004) **313**:269–325. doi: 10.1016/j.aop.2004.04.003
12. Noreen A, Olaussen K. Quantum loop expansion to high orders, extended borel summation, and comparison with exact results. *Phys Rev Lett*. (2013) **111**:040402. doi: 10.1103/PhysRevLett.111.040402
13. Janke W, Kleinert H. Convergent strong-coupling expansions from divergent weak-coupling perturbation theory. *Phys Rev Lett*. (1995) **75**:2787–91. doi: 10.1103/PhysRevLett.75.2787
14. Mushtaq A, Noreen A, Olaussen K, Øverbø I. Very-high-precision solutions of a class of Schrödinger type equations. *Comput Phys Commun*. (2011) **182**:1810–3. doi: 10.1016/j.cpc.2010.12.046
15. Noreen A, Olaussen K. High precision series solutions of differential equations: ordinary and regular singular points of second order ODE's. *Comput Phys Commun*. (2012) **183**:2291–7. doi: 10.1016/j.cpc.2012.05.015
16. Van Der Walt S, Colbert SC, Varoquaux G. The NumPy array: a structure for efficient numerical computation. *Comput Sci Eng*. (2011) **13**:22–30. doi: 10.1109/MCSE.2011.37
17. Jones E, Oliphant T, Peterson P. *SciPy: Open Source Scientific Tools for Python*. (2014). Available online at: http://www.scipy.org/
18. Oliphant TE. Python for scientific computing. *Comput Sci Eng*. (2007) **9**:10–20. doi: 10.1109/MCSE.2007.58
19. Mushtaq A, Noreen A, Olaussen A. *Python Package for Numerical Solutions of Quantum Mechanical Eigenvalue Problems* (2020). doi: 10.6084/m9.figshare.127655
20. Noreen A, Olaussen A. A python class for higher-dimensional schrödinger equations. In: *Proceedings of the International MultiConference of Engineers and Computer Scientists IMECS 2015*. Hong Kong (2015). p. 206–11.

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.