Solving the energies for a hindered methyl rotor potential with the *QRotor* Python package

Pablo Gila-Herranz

July 16, 2024

Contents

1 Theoretical background **1**
1.1 Hamiltonian for a hindered methyl rotor potential 1
1.2 Solving the time-independent Schrödinger equation with the Finite Difference Method 1

2 Calibration **2**
2.1 Energy convergence 2
2.2 Interpolation of custom potentials 4
2.3 Reproducing known results from titov2023 7

3 Preliminary results **7**
3.1 Isotope efects 7
3.2 Custom potential with a quasi-infinite barrier 8

4 Comment on why we need to solve the eigenvalues for every new B value **9**

References **10**

1 Theoretical background

1.1 Hamiltonian for a hindered methyl rotor potential

A hindered methyl rotor potential can be expressed as a function of the angle, φ. Thus its energies are given by the 1-dimensional time-independent Schrödinger equation,

$$H\Psi(\varphi) = E\Psi(\varphi)$$

The hamiltonian depends of the kinetic rotational energy and the potential energy,

$$H = -B\frac{d^2}{d\varphi^2} + V(\varphi) \tag{1}$$

The kinetic rotational energy depends on the inertia $I = mr^2$ of each hydrogen, as

$$B = \frac{\hbar^2}{2I} = \frac{\hbar^2}{2\sum_i m_i r_i^2} \tag{2}$$

In QRotor the potential can be introduced as a custom array. It can also be adjusted to a function such as follows, where the coefficients were previously obtained via electronic calculation methods [1],

$$V(\varphi) = c_0 + c_1\sin(3\varphi) + c_2\cos(3\varphi) + c_3\sin(6\varphi) + c_4\cos(6\varphi)$$

1.2 Solving the time-independent Schrödinger equation with the Finite Difference Method

The time-independent Schrödinger equation is a second-order differential equation. It can be solved numerically by discretizing it with the finite difference method [2]. This way, with a fine enough grid, the first derivative can be approximated as the slope,

$$\frac{d\Psi}{d\varphi} = \frac{\Psi(\varphi+\Delta\varphi) - \Psi(\varphi)}{\Delta\varphi}$$

Following the same procedure, the second derivative is

$$\frac{d^2\Psi}{d\varphi^2} = \nabla^2\Psi = \frac{\frac{\Psi(\varphi+\Delta\varphi)-\Psi(\varphi)}{\Delta\varphi} - \frac{\Psi(\varphi)-\Psi(\varphi-\Delta\varphi)}{\Delta\varphi}}{\Delta\varphi} = \frac{\Psi(\varphi+\Delta\varphi) - 2\Psi(\varphi) + \Psi(\varphi-\Delta\varphi)}{\Delta\varphi^2}$$

This second derivative can be expressed in matrix form as

$$\nabla^2 = \frac{1}{\Delta\varphi^2}\begin{bmatrix} -2 & 1 & 0 & \cdots & 0 & 0 \\ 1 & -2 & 1 & & 0 & 0 \\ 0 & 1 & \ddots & & & \vdots \\ \vdots & & & \ddots & 1 & 0 \\ 0 & 0 & & 1 & -2 & 1 \\ 0 & 0 & \cdots & 0 & 1 & -2 \end{bmatrix}$$

The multiplication of this operator and the wavefunction vector yields the second derivative at every grid point. To impose periodic boundary conditions, the first and last grid points are connected with an off-diagonal term,

$$\nabla^2 = \frac{1}{\Delta\varphi^2}\begin{bmatrix} -2 & 1 & 0 & \cdots & 0 & \mathbf{1} \\ 1 & -2 & 1 & & 0 & 0 \\ 0 & 1 & \ddots & & & \vdots \\ \vdots & & & \ddots & 1 & 0 \\ 0 & 0 & & 1 & -2 & 1 \\ \mathbf{1} & 0 & \cdots & 0 & 1 & -2 \end{bmatrix}$$

Finally, to build the potential energy operator, the energy at each grid point is set to equal the potential energy. This results in a diagonal matrix, with the potential energy at each point along the diagonal,

$$V(\varphi) = \begin{bmatrix} V(\varphi_1) & 0 & \cdots & 0 \\ 0 & V(\varphi_2) & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & V(\varphi_N) \end{bmatrix}$$

This way, the energy eigenvalues of the hindered methyl rotor can be obtained as the eigenvalues of the newly-constructed hamiltonian matrix from equation (1). QRotor solves this eigenvalue problem with the shift-inverted mode of the ARPACK package, provided through SciPy [3].

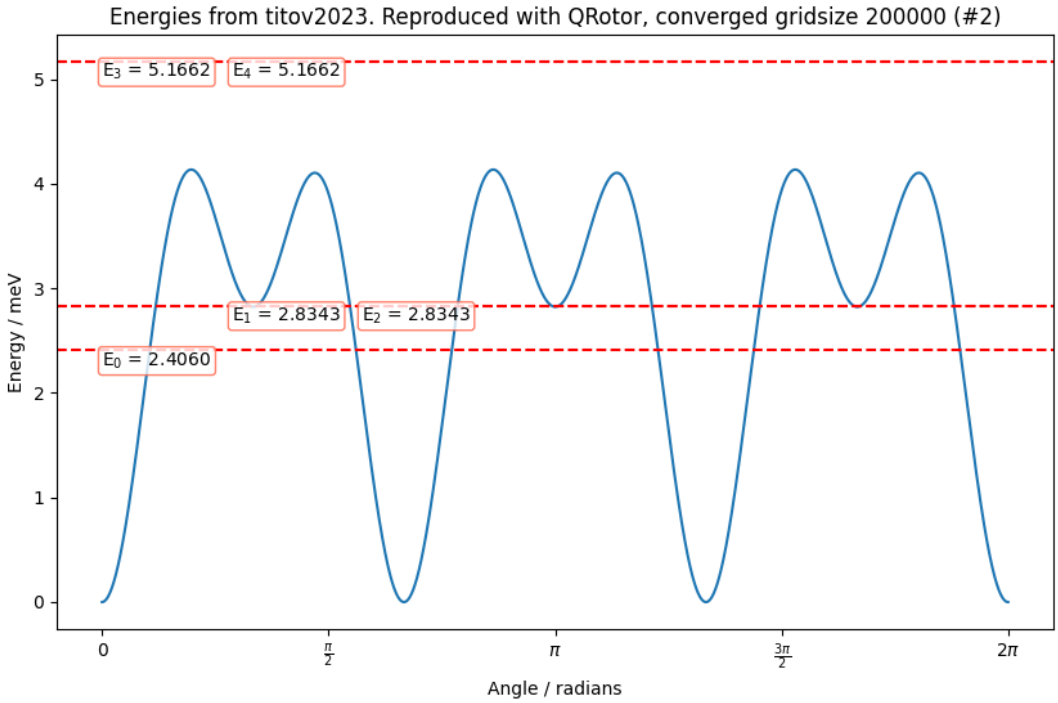
2 Calibration

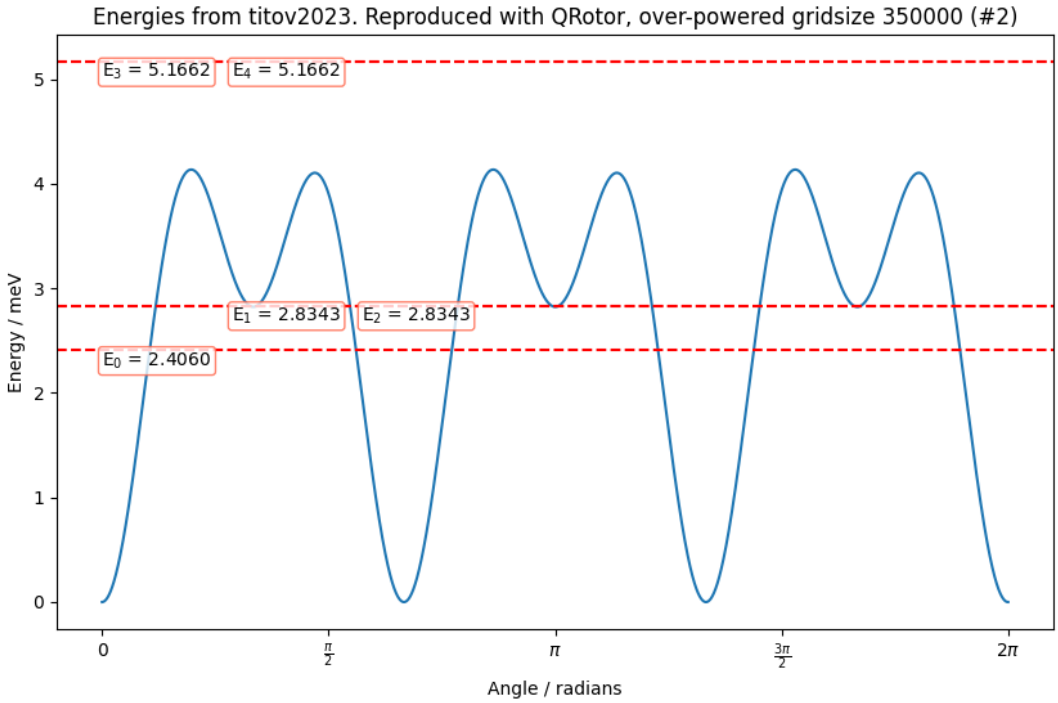
2.1 Energy convergence

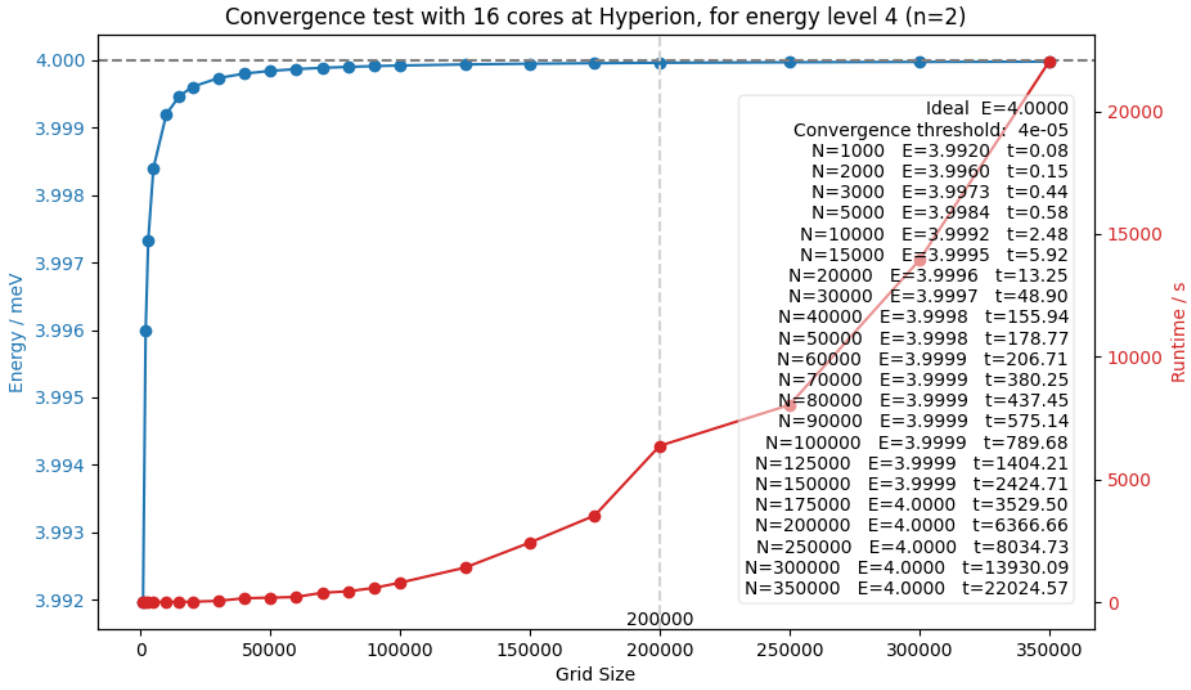
The convergence of the eigenvalues with the grid size was initially studied for a zero potential. These results were later confirmed for a known potential. The calculations were performed on DIPC's Hyperion cluster [4], to take advantage of the RAM memory required to diagonalize the largest Hamiltonian matrices.

A reasonable convergence of 3 decimals was observed for the energies with quantum numbers n=1 and n=2 with grids of 5,000 and 20,000 respectively. For a convergence of 4 decimals, for n=1 and n=2 the grids grow up to 50,000 and 200,000 respectively. The results from the convergence study are summarized in the following table. A comparison for a known potential [1] between a converged and an oversized grid is presented. Notice that the eigenvalues in the outputs are labeled without degeneration for computational purposes.

Quantum n. (n)	**Eigenvalue**	**Decimal precision**	**Gridsize**	**Runtime**
0	0	-	-	-
1	1, 2	3	5,000	0.6s
1	1, 2	4	**50,000**	3min
2	3, 4	3	20,000	13s
2	3, 4	4	**200,000**	1h 45min
3	5, 6	3	50,000	3min
4	7, 8	3	90,000	9min 30s
5	9	3	150,000	40min


Energies from titov2023. Reproduced with QRotor, converged gridsize 200000 (#2)
E3 = 5.1662
E4 = 5.1662
E1 = 2.8343
E2 = 2.8343
E0 = 2.4060
Energy / meV
Angle / radians


Energies from titov2023. Reproduced with QRotor, over-powered gridsize 350000 (#2)
E3 = 5.1662
E4 = 5.1662
E1 = 2.8343
E2 = 2.8343
E0 = 2.4060
Energy / meV
Angle / radians

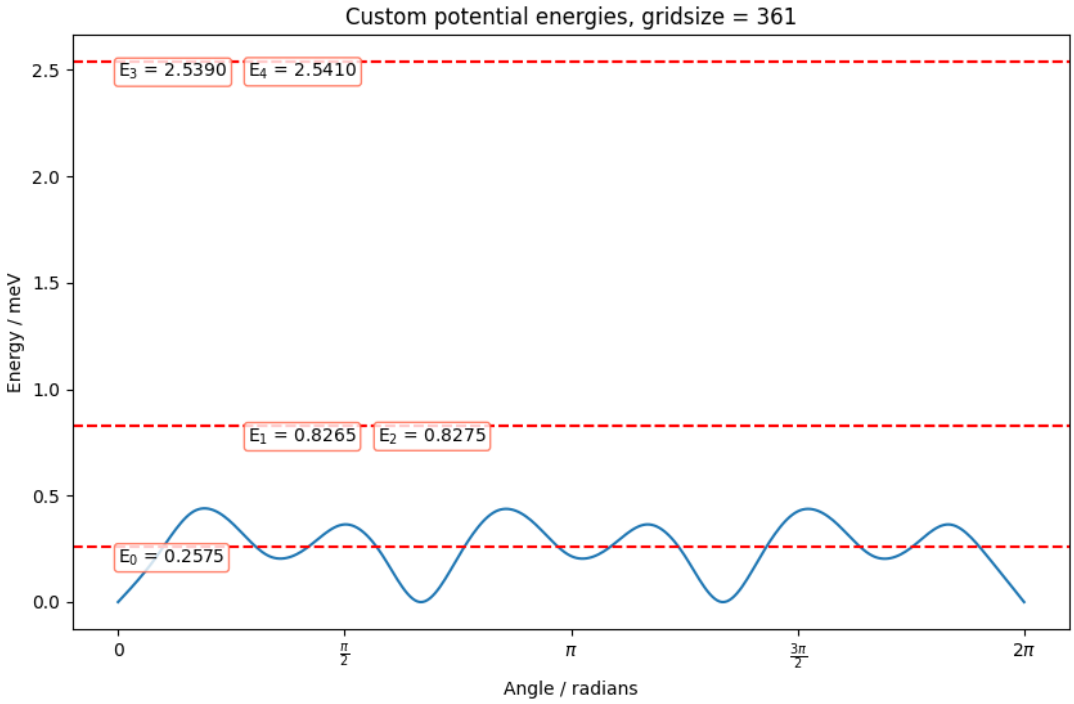


2.2 Interpolation of custom potentials

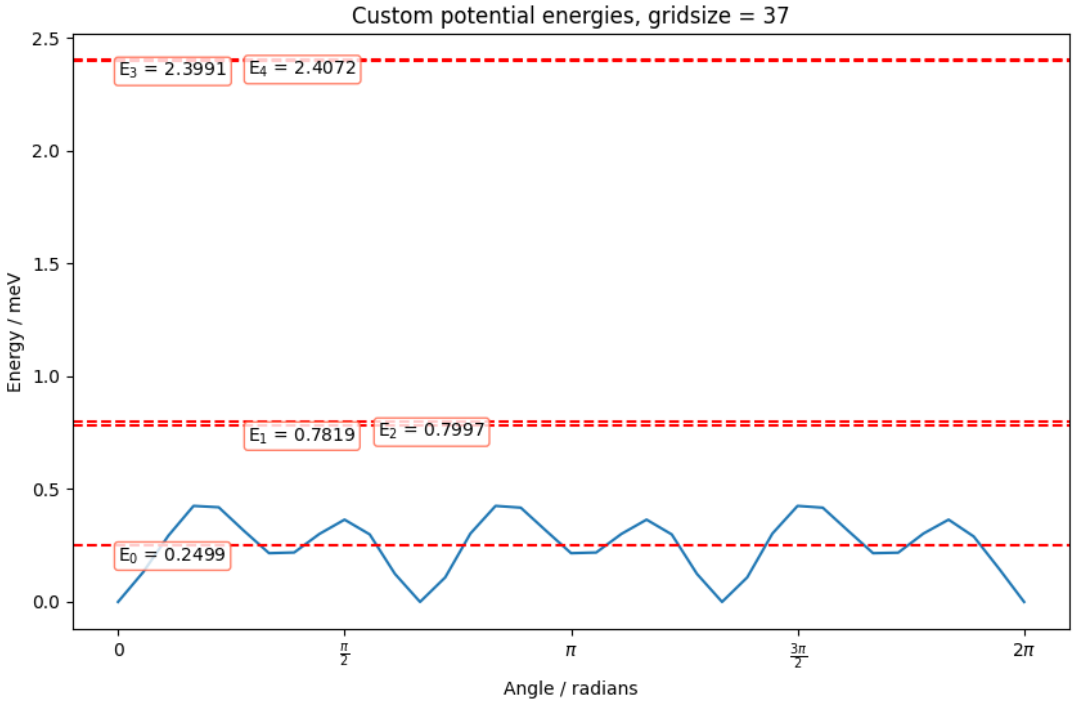
QRotor reads input potentials as `*.dat` files, with angle and energy information. This potential data can be previously calculated with electronic structure methods. The potential can then be interpolated to a bigger grid size to converge the energies. To do so, it takes advantage of SciPy's Cubic Splines, where the interpolated curve is made of points with matching first and second derivatives [5].

To validate the interpolation method, a test potential (provided by Félix) with 360 points was capped to 36 points. An interpolation to its previous size proved to match the original potential. Further interpolation to a grid of size 5000 showed the expected convergence between the original and the capped potential up to the 3rd decimal place. The results are shown in the following figures.

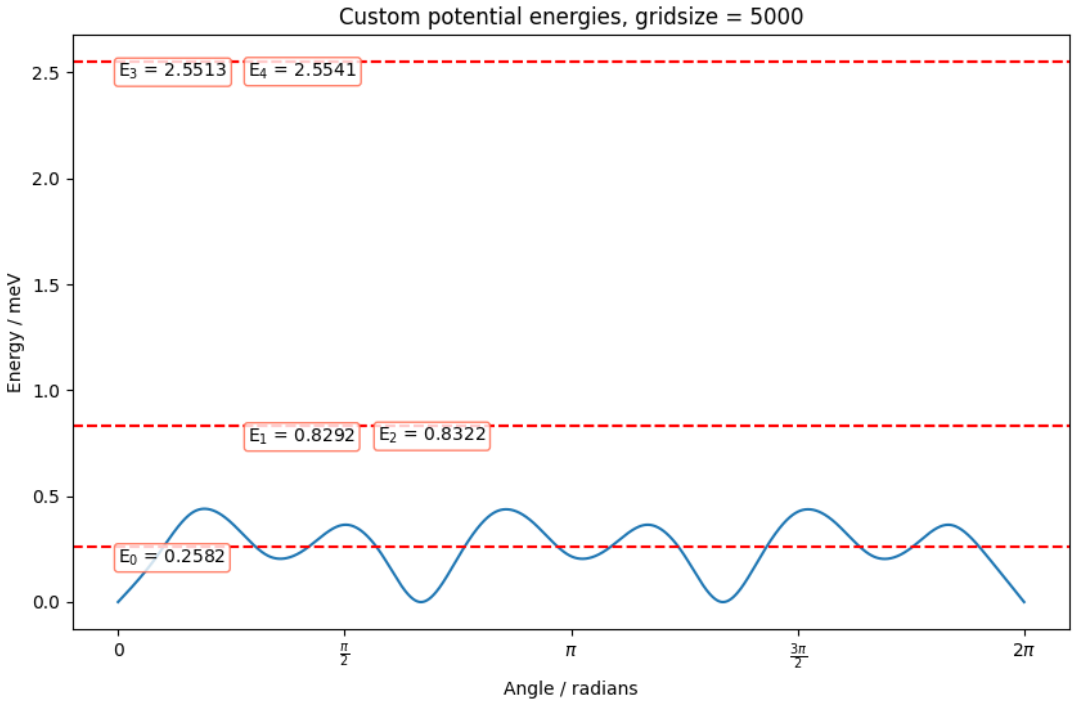
The original potential:



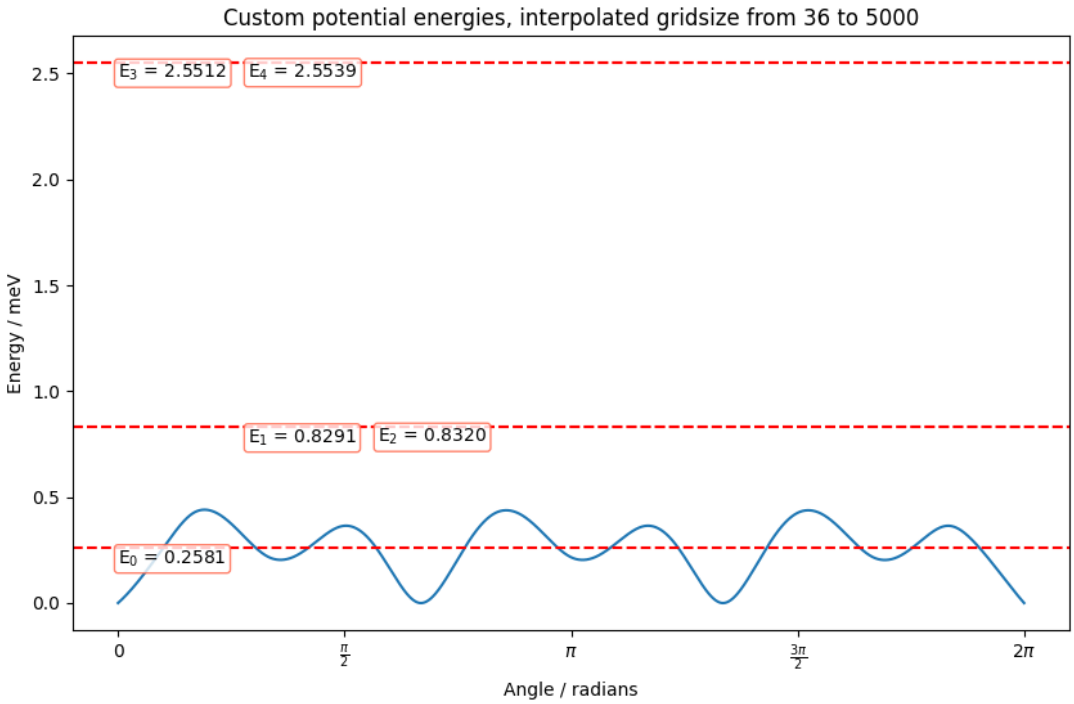
And the capped potential:



The original potential, interpolated to a grid of size 5000 to ensure a convergence up to the 3rd decimal position:



As expected, the interpolated capped potential converges up to the 3rd decimal position:

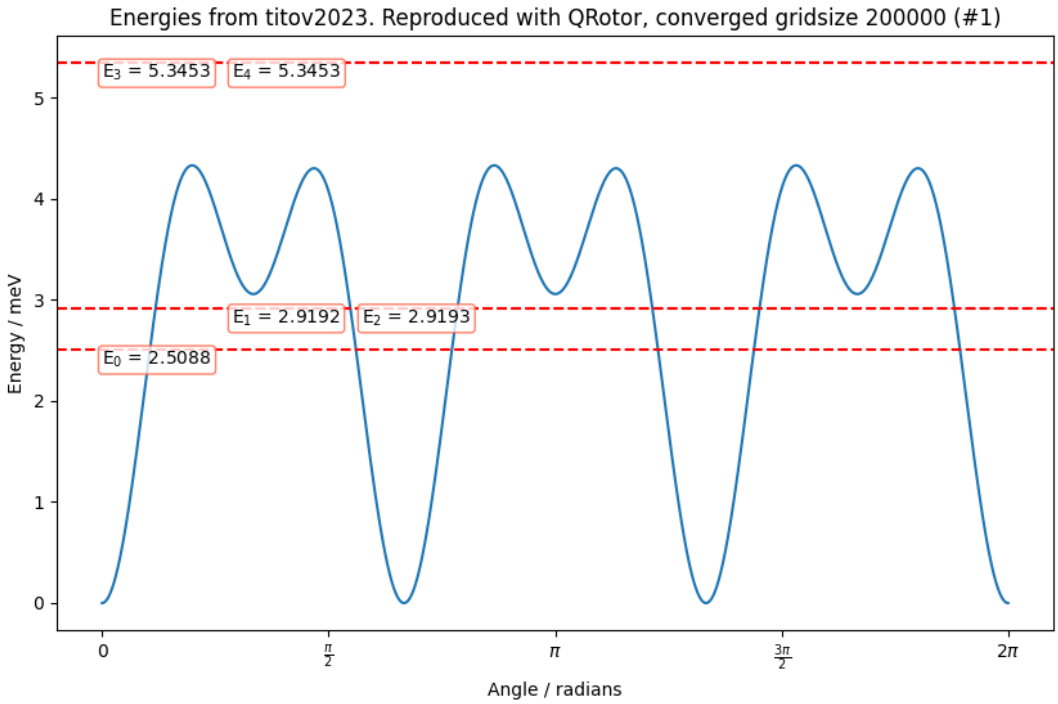


2.3 Reproducing known results from titov2023

The energies for a known potential (ZIF-8) from titov2023 [1] were calculated with a converged grid of size 200000. The results were compared with the ones from the original paper:

$$E_0 = 2.4871 \text{ meV},\ E_1 = 2.8969 \text{ meV},\ E_1' = 2.8971 \text{ meV},\ E_2 = 5.3214 \text{ meV},\ E_2' = 5.3222 \text{ meV}$$

The energies present a good match. The results from the first potential from the paper are shown in the following figure.

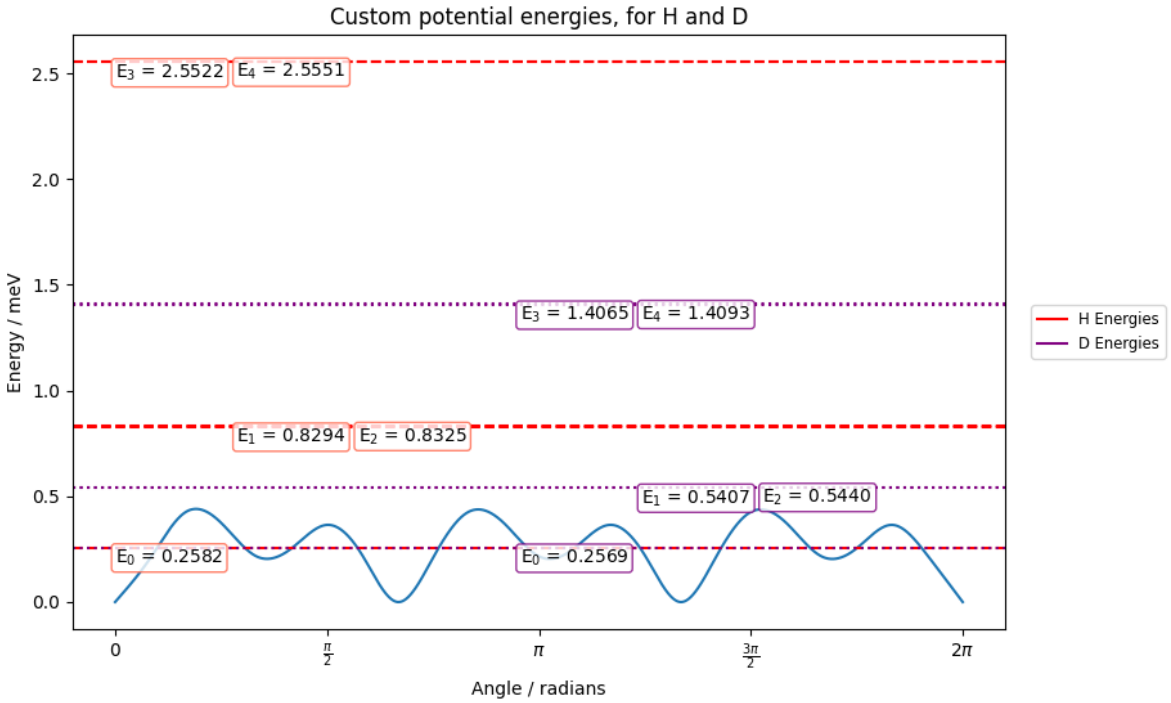


3 Preliminary results

3.1 Isotope efects

Taking a test potential given in meV, the energies were calculated for a regular methyl ($B = 0.645$) and a deuterated methyl ($B = 0.323$.)

It is expected from eq. (2) to observe a lower rotational energy as a result of an increase in the atomic masses. The isotopic effects on the energies of the hindered methyl rotor were studied by replacing the hydrogen mass with that of deuterium. The test potential was interpolated to a grid of size 200000 to ensure an optimal convergence. As expected, the results show a shift to lower energies for the deuterated counterpart.



3.2 Custom potential with a quasi-infinite barrier

Felix provided a potential in the eV range, with a huge inertia of of 301.54 amu Å^2, so that the rotational energy was a small value,

$$B = \frac{\hbar^2}{2I} = 6.93 \times 10^{-6}\text{eV}$$

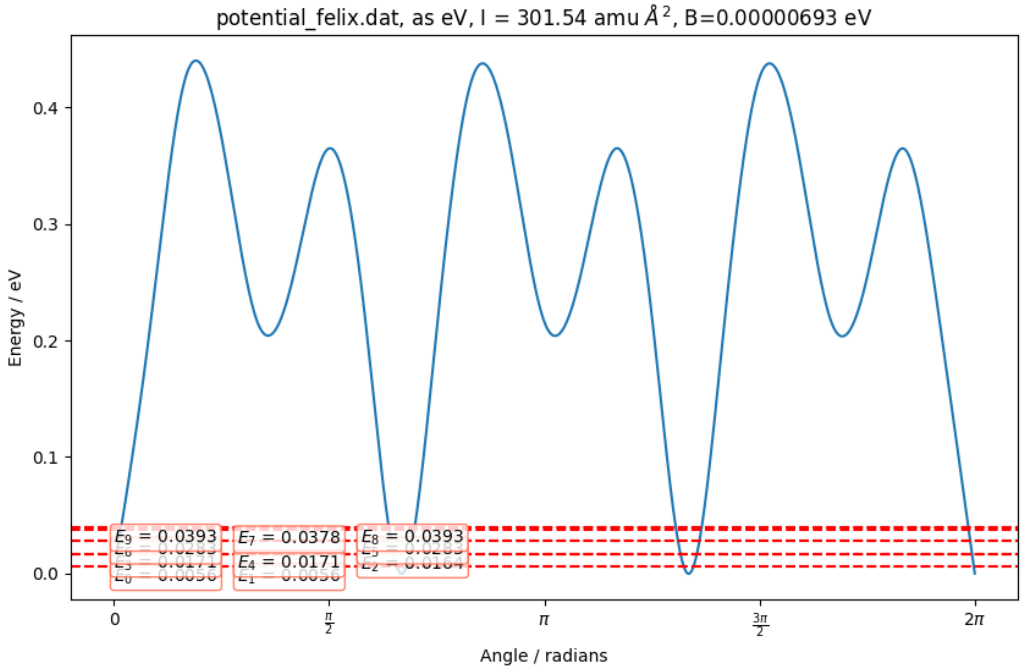
The result is the textbook harmonic potential for a rotor, which presents a degeneration in the ground energy level, since the rotation can be in both directions. The energy in this case would be given by

$$E = \frac{1}{2}k_\phi x^2$$

Felix told me something like

$$E = \sqrt{\frac{K_\phi}{I_{\text{something}}}}$$

The results are shown in the following figure.



4 Comment on why we need to solve the eigenvalues for every new B value

UPDATE: I should test with a reduced variable, $\rho = \frac{V_B}{B}$

Note that from the Hamiltonian eq. (1),

$$H = -B\frac{d^2}{d\varphi^2} + V(\varphi)$$

It can be deduced that B can theoretically be extracted by expressing the potential as a reduced variable, so that (TYPO: B SHOULD DIVIDE ALL VALUES)

$$\frac{H}{B} = \begin{bmatrix} +2+\frac{V_1}{B} & -1 & 0 & \cdots & 0 & -1 \\ -1 & +2+\frac{V_2}{B} & -1 & & 0 & 0 \\ 0 & -1 & \ddots & & & \vdots \\ \vdots & & & \ddots & -1 & 0 \\ 0 & 0 & & -1 & +2+\frac{V_{N-1}}{B} & -1 \\ -1 & 0 & \cdots & 0 & -1 & +2+\frac{V_N}{B} \end{bmatrix}$$

But this matrix **can not be solved numerically** until B is known. B cannot simply be set to 1 and then multiply the Hamiltonian eigenvalues by the desired B, since the diagonal terms can not be expressed as a reduced variable.

However, we know from the convergence criteria that it does not suppose any problem to just repeat the calculation for a new B value. In a regular PC, convergence up to 3 decimal places

can be achieved in less than a minute, which allows for a good iteration in our tests.

References

[1] K. Titov, M. R. Ryder, A. Lamaire, Z. Zeng, A. K. Chaudhari, J. Taylor, E. M. Mahdi, S. M. J. Rogge, S. Mukhopadhyay, S. Rudić, V. Van Speybroeck, F. Fernandez-Alonso, and J.-C. Tan, "Quantum tunneling rotor as a sensitive atomistic probe of guests in a metal-organic framework," vol. 7, no. 7, p. 073402.

[2] Q. Kong, T. Siauw, and A. Bayen, "Finite Difference Method," in *Python Programming And Numerical Methods: A Guide For Engineers And Scientists*, 1st ed. Elsevier. [Online]. Available: https://pythonnumericalmethods.berkeley.edu/notebooks/chapter23.03-Finite-Difference-Method.html

[3] Sparse eigenvalue problems with ARPACK — SciPy v1.13.0 Manual. [Online]. Available: https://docs.scipy.org/doc/scipy/tutorial/arpack.html#sparse-eigenvalue-problems-with-arpack

[4] Hyperion overview - DIPC Technical Documentation. [Online]. Available: https://scc.dipc.org/docs/systems/hyperion/overview/

[5] 1-D interpolation with Cubic Splines — SciPy v1.13.0 Manual. [Online]. Available: https://docs.scipy.org/doc/scipy/tutorial/interpolate/1D.html#cubic-splines