N-Particle DVR Implementation in One Dimension

Axel Fraud

October 27, 2025

1 Introduction

This project is based on the approach presented in the paper *"Discretized momentum-space methods for few-body problems"* by Dr. Sebastian König. The goal of the project was to understand and implement a one-dimensional, N-particle quantum system using Discrete Variable Representation (DVR), following the structure and techniques introduced in the paper.

DVR is a numerical method that makes it possible to represent quantum mechanical operators on a discrete grid. In this context, the DVR basis is constructed from plane-wave eigenfunctions of the momentum operator, assuming periodic boundary conditions in a box of length L. One of the key benefits of DVR is that local operators like the potential energy become nearly diagonal, which simplifies the calculation. The kinetic energy operator is not diagonal in this basis, but it can still be written exactly using an analytical expression.

I began by working through the derivations in the paper and implementing the two-particle version of the system. This helped me verify that I understood the basic structure of the method. I then extended the system to three particles and finally developed a general version of the code that works for any number of particles N. At each stage, I worked in relative coordinates and applied the minimum image convention to maintain periodicity.

The implementation closely follows the equations and structure from the paper. I used the exact expression for the kinetic energy matrix provided in Eq. (38), and I did not use the FFT trick often employed in spectral methods. This decision was made to stay consistent with the method described in the paper and to better understand the construction of the Hamiltonian from first principles.

The remainder of this report explains the full process step by step, including how the DVR grid is constructed, how the kinetic and potential operators are implemented, how the Hamiltonian is assembled, and how the eigenvalue problem is solved.

2 Parameter Setup and Grid Construction

The first step in building the DVR implementation is to define the physical and numerical parameters of the system. These parameters control the size of the domain, the resolution of the grid, the number of particles, and the strength and range of the interactions.

System Parameters

- N: Number of particles in the system. In this implementation, we first tested with $N = 2$ and $N = 3$, and then generalized to arbitrary N.
- n: Number of DVR points per coordinate axis. This sets the resolution of the grid.
- L: The length of the one-dimensional periodic box. All particle positions are confined to the interval $[-L/2, L/2)$ with periodic boundary conditions.
- m: Mass of each particle. The reduced mass $mu = m/2$ is used in the kinetic energy operator for pairwise interactions.
- V_0, R: Depth and range of the Gaussian pairwise interaction potential.

DVR Grid

The DVR grid is constructed using evenly spaced points over the domain $[-L/2, L/2)$. For n DVR points, the spacing between adjacent points is given by:

$$\Delta x = \frac{L}{n} \tag{1}$$

The set of grid points is then defined as:

$$x_k = -\frac{L}{2} + k\Delta x, \quad \text{for } k = 0, 1, \ldots, n-1 \tag{2}$$

This matches the DVR coordinate grid used in Eq. (35) of the paper, where the position values are uniformly spaced.

Momentum Grid

Although we do not use FFT in this implementation, the momentum grid is still used to construct the DVR transformation matrix. The momentum values are defined using the discrete Fourier frequencies consistent with periodic boundary conditions:

$$p_j = \frac{2\pi j}{L}, \quad \text{for } j = -\frac{n}{2}, \ldots, \frac{n}{2} - 1 \tag{3}$$

These values are used to define the plane-wave DVR basis, and they appear in the exponent of the transformation matrix $U_{kj} = \exp(ip_j x_k)/\sqrt{n}$.

Dimensionality of the System

When working with N particles in one dimension, the center-of-mass motion is factored out. As a result, the wavefunction is expressed in terms of $N - 1$ relative coordinates. Therefore, the full configuration space is defined on a grid of shape:

$$\text{Grid shape} = (n, n, \ldots, n) = n^{N-1} \tag{4}$$

The total Hilbert space dimension grows exponentially with $N-1$, so computational cost becomes significant as N increases.

Implementation Notes

All of the above parameters are defined at the start of the code. For example, a typical setup might use:

- `N = 3`
- `n_pts = 16`
- `L = 10.0`
- `mass = 1.0`
- `mu = mass / 2`
- `V0 = -5.0,  R = 1.0`

The arrays `x_vals` and `p_vals` are created using `numpy.linspace` and `numpy.fft.fftfreq` (the latter used only for defining p_j). These values form the basis for all subsequent steps in the DVR calculation.

3 DVR Basis and Transformation Matrix

The core of the DVR method is the transformation between momentum and position space using a discrete set of basis functions. In the implementation described by Dr. Sebastian König, the DVR basis is built from plane-wave eigenfunctions of the momentum operator in a periodic box. This construction leads to an explicit expression for the unitary transformation matrix, which is central to evaluating operators in the DVR basis.

Plane-Wave DVR Basis

In a periodic box of length L, the allowed momentum eigenvalues are:

$$p_j = \frac{2\pi j}{L}, \quad j = -\frac{n}{2}, \ldots, \frac{n}{2} - 1 \tag{5}$$

These momenta correspond to plane-wave states of the form:

$$\phi_j(x) = \frac{1}{\sqrt{L}} e^{ip_j x} \tag{6}$$

The DVR grid is constructed on position points x_k defined as in Section 2. To form the DVR transformation matrix, we evaluate the plane-wave functions at each grid point x_k.

DVR Transformation Matrix U

The transformation matrix U has components:

$$U_{kj} = \frac{1}{\sqrt{n}} \exp(ip_j x_k) \tag{35}$$

This is Equation (35) from the paper. The matrix U is unitary, satisfying:

$$U^\dagger U = UU^\dagger = I \tag{7}$$

This transformation is used to switch between coordinate and momentum representations of functions and operators. However, in this implementation, we do not need to explicitly use U to transform operators. Instead, we take advantage of the fact that in the DVR basis:

- The potential energy operator becomes approximately diagonal.
- The kinetic energy operator has a known closed-form expression (see Section 4).

Implementation Notes

In code, the matrix U can be constructed for verification purposes, though it is not required for computing the Hamiltonian. It can be useful for checking orthogonality or for transforming wavefunctions between representations. A typical implementation in Python is:

```
U = np.exp(1j * np.outer(x_vals, p_vals)) / np.sqrt(n_pts)
```

Here, `x_vals` and `p_vals` are the position and momentum grids defined earlier.

4 Potential Energy Operator

The potential energy in this system is modeled using pairwise Gaussian interactions between particles. The key advantage of using DVR is that local operators such as the potential energy operator become approximately diagonal in the DVR basis.

Pairwise Gaussian Potential

The potential energy between any two particles i and j is given by:

$$V(r_{ij}) = V_0 \exp\left(-\left(\frac{r_{ij}}{R}\right)^2\right) \tag{8}$$

Here:

- $V_0 < 0$ is the depth of the potential well.
- R is the range (width) of the interaction.
- r_{ij} is the relative distance between particles i and j.

Minimum Image Convention

Since the system is defined on a periodic box, we must ensure that the shortest possible distance between particles is used. To do this, we apply the minimum image convention:

$$r_{ij} = x_i - x_j - L \cdot \text{round}\left(\frac{x_i - x_j}{L}\right) \tag{9}$$

This wraps distances into the interval $[-L/2, L/2)$, ensuring proper behavior under periodic boundary conditions.

Diagonal Approximation in DVR

As discussed in Equation (40) of the paper, the potential operator becomes approximately diagonal in the DVR basis:

$$\langle \psi_k | \hat{V} | \psi_l \rangle \approx V(x_k)\delta_{kl} \tag{40}$$

This means we only need to compute the potential at each DVR grid point and assign it to the diagonal of the potential matrix.

Extension to Multi-Particle Systems

For an N-particle system, the total potential energy is the sum over all distinct pairs:

$$V_{\text{total}} = \sum_{i<j} V(r_{ij}) \tag{10}$$

In the DVR implementation, we construct the full grid over the $N-1$ relative coordinates, loop over all combinations of grid points, and compute the sum of Gaussian interactions for each configuration. The resulting values populate the diagonal of the potential energy matrix.

Implementation Notes

In code, we create the potential matrix by looping over all index combinations in the grid:

```
for idx in np.ndindex(grid_shape):
    coords = [x_vals[i] for i in idx]
    V = 0.0
    for i in range(len(coords)):
        for j in range(i + 1, len(coords)):
            r_ij = coords[i] - coords[j]
            r_ij = r_ij - L * np.round(r_ij / L)
            V += V0 * np.exp(- (r_ij / R)**2)
    V_diag[idx] = V
```

The result is then flattened and placed on the diagonal of the potential matrix.

5 Kinetic Energy Operator

Unlike the potential energy, the kinetic energy operator is not diagonal in the DVR basis. However, an exact closed-form expression exists for its matrix elements when using a plane-wave DVR with periodic boundary conditions. This expression is given in Equation (38) of the paper.

1D Kinetic Energy Matrix

For a one-dimensional grid with n DVR points, the matrix elements of the kinetic energy operator are:

$$T_{kl} = \begin{cases} \frac{\pi^2 n^2}{6\mu L^2}\left(1 + \frac{2}{n^2}\right), & k = l \\ \frac{(-1)^{k-l}\pi^2}{\mu L^2 \sin^2\left(\frac{\pi(k-l)}{n}\right)}, & k \neq l \end{cases} \tag{38}$$

Here:

- μ is the reduced mass.
- L is the length of the periodic box.
- n is the number of DVR points along one axis.

This expression is derived from the second derivative operator in momentum space and evaluated on the DVR grid. It does not rely on FFTs and is valid for periodic boundary conditions.

Higher-Dimensional Generalization

For an N-particle system, we eliminate center-of-mass motion and work in $N - 1$ relative coordinates. To construct the full kinetic energy operator in $N - 1$ dimensions, we use the Kronecker sum of the 1D kinetic energy matrix:

$$\hat{T}_{\text{total}} = T \otimes I \otimes \cdots \otimes I + I \otimes T \otimes \cdots \otimes I + \cdots + I \otimes \cdots \otimes T \tag{11}$$

This ensures that the kinetic energy operator acts on each coordinate direction independently, while preserving the tensor product structure of the DVR space.

Kronecker Construction

In code, the full kinetic energy matrix is built by looping over each coordinate axis and inserting the 1D kinetic energy operator at the appropriate position in the Kronecker product:

```
T_full = None
for i in range(N - 1):
    ops = [np.eye(n_pts) for _ in range(N - 1)]
    ops[i] = T_1d
    Ti = ops[0]
    for op in ops[1:]:
        Ti = np.kron(Ti, op)
    T_full = Ti if T_full is None else T_full + Ti
```

Each term in the sum corresponds to the kinetic energy acting along one coordinate axis in the full $(N-1)$-dimensional space.

Implementation Notes

- The 1D kinetic energy matrix is symmetric and real.
- The Kronecker sum preserves Hermiticity in the full space.
- The result is a dense matrix of size $n^{N-1} \times n^{N-1}$, which becomes computationally expensive as N increases.

This construction completes the kinetic energy part of the Hamiltonian. Together with the diagonal potential matrix, it forms the full Hamiltonian operator.

6 Hamiltonian Assembly and Sanity Checks

Once both the kinetic and potential energy operators have been constructed, we assemble the total Hamiltonian of the system. The Hamiltonian acts on the full DVR space and is defined as:

$$\hat{H} = \hat{T} + \hat{V} \tag{12}$$

Matrix Construction

The kinetic energy matrix $\hat{T}$ is already constructed as a full matrix using Kronecker sums (see Section 5). The potential energy matrix $\hat{V}$ is diagonal in the DVR basis and is constructed from a 1D array containing the potential energy at each grid point.

The diagonal potential is converted into a square matrix by placing its values along the diagonal:

```
V_matrix = np.diag(V_dvr)
H = T_dvr + V_matrix
```

Sanity Checks

Before solving the eigenvalue problem, we perform a few essential sanity checks:

- **Hermiticity:** We verify that the Hamiltonian is Hermitian:

  $$H = H^\dagger$$

  In code, this is checked using `np.allclose(H, H.conj().T)`.

- **Matrix shape:** The Hamiltonian matrix should be square with dimensions $n^{N-1} \times n^{N-1}$. This is confirmed by printing `H.shape`.
- **Potential scale:** The potential values (before placing them into the matrix) are inspected to ensure they are physically reasonable. For example, the minimum value should match approximately the depth V_0, and values far from the interaction region should approach zero.

Example Output

For a typical setup with $N = 3$ and $n = 16$, we expect:

- A Hamiltonian matrix of shape 256×256
- A symmetric matrix with real-valued diagonal and complex-valued (but conjugate symmetric) off-diagonal entries in T
- Potential values that range from approximately V_0 to 0

These checks confirm that the Hamiltonian is correctly assembled and ready for diagonalization in the next step.

7 Solving for Energies and Final Observations

Once the Hamiltonian matrix was assembled, the next step was to solve for the energy levels and wavefunctions by diagonalizing the matrix. This gave us the eigenvalues (the energy levels) and the corresponding eigenvectors (the wavefunctions written in the DVR basis).