QDynamics internal documentation

Duncan M. Freeman

January 20, 2024

DISCLAIMER: THIS IS A WORK IN PROGRESS PROJECT AND MAKES NO CLAIM OF SCIENTIFIC VALUE OR ACCURACY OF ANY KIND. DO NOT RELY ON THIS WORK!

With that out of the way, welcome!

This is documentation for my own purposes, to remember how this all fits together.

This project is a basic mockup of the Fewest Switches Surface Hopping Method, and follows:

Pedagogical Overview of the Fewest Switches Surface Hopping Method Amber Jain and Aarti Sindhu ACS Omega 2022 7 (50), 45810-45824 DOI: 10.1021/acsomega.2c04843

(todo: actual citation lol)

1 Overview

The simulation consists of a number of classically-modelled atomic nuclei, which are orbited by a cloud of electrons. The atomic nuclei are assumed to only respond to one energy state of the electron cloud (λ) at any given time.

The Hamiltonian of the electron system is:

$$H = \hat{T} + V(R)$$

1.1 Algorithm

1. Initialize runtime parameters:

   - (R, P): These are the position and momentum vectors of the atomic nuclei. They are modelled clasically. In code, they are called position and velocity, the mass being assumed constant for all nuclei.

   - λ: The current electronic state we are allowing the nuclei to observe. The corresponding energy eigenstate will be denoted Φ_λ.

   - c_j: The current electronic wavefunction parameters. It can also be represented as a vector $\vec{c}$. Note that $\psi(t)$ is a linear combination of energy eigenstates based on these coefficients:

     $$\psi(t = 0) = [\Phi_j]\vec{c}$$

     Or, in another notation, parameterized by time:

     $$|\psi(t)\rangle = \sum_j e^{-iE_j t/\hbar} c_j \, |\Phi_j\rangle$$

     This may be obtained by a delta function in energy space (setting an element of c_j to one), or by obtaining eigenvectors and solving for the coefficients given a desired wavefunction input ($[\Phi_j]^{-1}\psi_0 = \vec{c}$).

(Main loop begins here)

2. Calculate new energy eigenbasis

   - Recalculate energy eigenbasis; $H\Phi_j = E_j\Phi_j$.

     Make sure to save the old eigenbasis for calculations futher down the line!

     Use an SVD algorithm to get Φ_j from H (eigenbasis in matrix representation hereby represented as $[\Phi_j]$).

     Note that if $\langle\Phi_j(t - \Delta T)|\Phi_j(t)\rangle < 0$, we should set $|\Phi_j(t)\rangle = -\,|\Phi_j(t)\rangle$.

3. Integrate classical motion using quantum-derived forces

- Calculate the force on the protons due to (single component λ of) the electron cloud and proton-proton interaction.

$$m\ddot{R} = F = -\langle \Phi_\lambda | \nabla_R H | \Phi_\lambda \rangle$$

- Integrate proton motion by a small time step.

4. Integrate quantum equations of motion

$$V_{kj} = \langle \Phi_k | H | \Phi_j \rangle$$

$$U_{kj} = \langle \Phi_k(t_0) | H | \Phi_j(t_0 + dt_c) \rangle$$

$$T_{kj} = \frac{1}{dt_c} \log(U)$$

$$\dot{c}_k = -\frac{i}{\hbar} \sum_j (V_{kj} - i\hbar T_{kj}) c_j$$

5. Potential surface hopping

The probability of a hop from current surface λ to another surface k is:

$$P(\lambda \to k) = \frac{2\,\mathrm{Re}(T_{\lambda k} c_\lambda^* c_k)}{|c_\lambda|^2}$$

Call a random number r between 0 and 1.

If $\sum_{l=1}^{k-1} P(\lambda \to k) < r < \sum_{l=1}^{k} P(\lambda \to k)$, then we will hop to this new state k. It's possible and even likely to have no hops at all.

Handling a hop requires calculating the time derivative coupling matrix:

$$d_{\lambda k} = \frac{\langle \Phi_\lambda | \nabla_R H | \Phi_k \rangle}{E_k - E_\lambda}$$

Determine the coefficient γ to conserve total energy.

$$a = \sum_n d_{\lambda k}^{n2}$$

$$b = \sum_n v_n \cdot d_{\lambda k}^n$$

$$c = E_k - E_\lambda$$

If $b^2 - 4ac > 0$, calculate $\gamma = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$. The $\pm$ there should have the same sign as b.

On the other hand, if If $b^2 - 4ac < 0$, $\gamma = b/a$.

Now we update the velocities:

$$v_n' = v_n - \gamma d_{\lambda k}^n$$

6. Display to user

7. Goto 2

1.2 Basis

2 User interface

2.1 Requirements

Very important:

- Start/stop animation
- Set time step parameters for quantum and classical subsystems
- View any eigenstate ψ_n and know its corresponding energy eigenvalue E_n, as well as the current state coefficient c_j.
- View positions of nuclei and their direction vectors (!)
- Know which eigenstate is currently active (radio button!)
- Edit position and momentum variables for each nucleus (mass and velocity set seperately)
- Display total energy as a graph, and energy residing in each subsystem
- Toggle between probability (black and white) and color-coded real valued wave function
- View forcefield due to quantum subsystem
- In the event of a hop, add an option to explicitly pause and display this!
- View potential function

Less important:

- View $\psi(t)$, the current total wavefunction of the electron
- Edit electric potential parameters for each nucleus individually

2.2 Implementation

Scalar field visualzation mode; this controls the "background" image. Can be set to show any of the individual energy eigenstates. There should be a toggle to have it always show the current energy eigenstate no matter what. Can also show the combined potential surface due to the atomic nuclei. If displaying an energy eigenstate, it can show the probability instead of the signed value of the wavefunction. Can also show the combined wavefunction $\psi(t)$.