CyberTraining project:

Implementing a numerical efficient formulation of FSSH in Libra

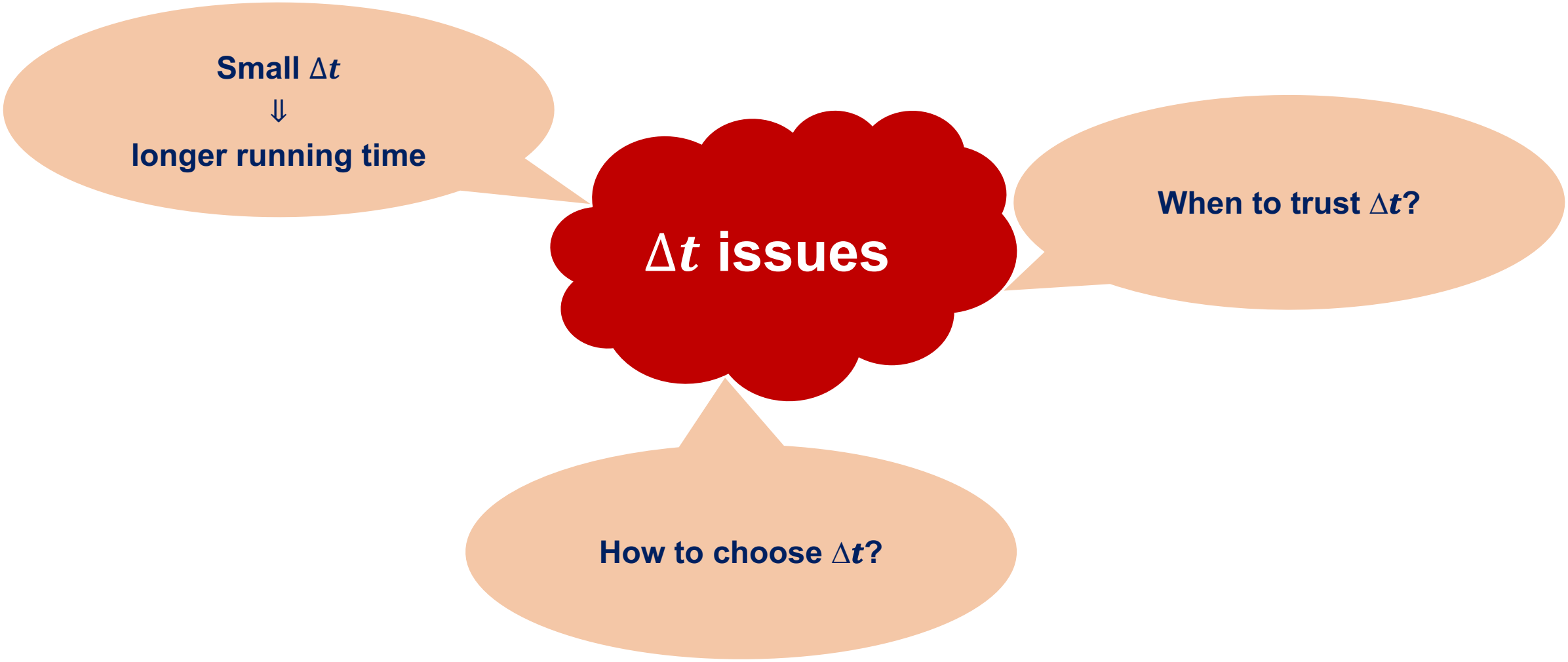
Leonardo Araujo | Leonardo.Araujo@tum.de | Department of Mathematics | Technical University of Munich

Agenda

- Motivation & Goal
- Formulation of a new hop probability
- Numerical experiment
- Time & workflow overview

Motivation & Goal

Motivation: issues of the time step size Δt for dynamics



Motivation: FSSH procedure from time t to $t + \Delta t$

1) Propagate classical coordinates from t to $t + \Delta t$ at active state m:

$$\boxed{\begin{aligned}\dot{\boldsymbol{x}} &= \boldsymbol{p} \\ \dot{\boldsymbol{p}} &= \boldsymbol{F}_{\boldsymbol{m}}^{\boldsymbol{adi}}(\boldsymbol{x})\end{aligned}}$$

2) Propagate TDSE of the electronic amplitudes from t to $t + \Delta t$:

$$\boxed{i\varepsilon \dot{\boldsymbol{c}}^{\boldsymbol{adi}}(\boldsymbol{t}) = \boldsymbol{H}^{\boldsymbol{adi}}(\boldsymbol{t})\boldsymbol{c}^{\boldsymbol{adi}}(\boldsymbol{t})}$$

with $\boldsymbol{H}_{\boldsymbol{m,n}}^{\boldsymbol{adi}}(\boldsymbol{t}) = \boldsymbol{\delta}_{\boldsymbol{m,n}}\boldsymbol{E}_{\boldsymbol{m}}(\boldsymbol{x}) - i\varepsilon \boldsymbol{d}^{\boldsymbol{m,n}}(\boldsymbol{x})^{\boldsymbol{T}}\boldsymbol{p}$

3) Hop to state n with probability:

$$\boxed{\boldsymbol{P}_{\boldsymbol{m,n}}^{\boldsymbol{FSSH}}(\boldsymbol{t}, \boldsymbol{t} + \Delta \boldsymbol{t}) = \frac{2\Delta t\, Re\left(\boldsymbol{\rho}_{\boldsymbol{n,m}}^{\boldsymbol{adi}}\right)\boldsymbol{d}^{\boldsymbol{m,n}}(\boldsymbol{x})^{\boldsymbol{T}}\boldsymbol{p}}{\boldsymbol{\rho}_{\boldsymbol{m,m}}^{\boldsymbol{adi}}}}$$

with $\boldsymbol{\rho}_{\boldsymbol{n,m}}^{\boldsymbol{adi}} = \boldsymbol{c}_{\boldsymbol{n}}^{\boldsymbol{adi}}\overline{\boldsymbol{c}_{\boldsymbol{m}}^{\boldsymbol{adi}}}$

Issue: energy conservation is not enough as indicator

1) Propagate classical coordinates from t to $t + \Delta t$ at active state m:

$$\dot{\boldsymbol{x}} = \boldsymbol{p}$$
$$\dot{\boldsymbol{p}} = \boldsymbol{F}_{\boldsymbol{m}}^{\boldsymbol{adi}}(\boldsymbol{x})$$

2) Propagate TDSE of the electronic amplitudes from t to $t + \Delta t$:

$$\boldsymbol{i\varepsilon \dot{c}^{adi}(t) = H^{adi}(t) c^{adi}(t)}$$

with $\boldsymbol{H_{m,n}^{adi}(t) = \delta_{m,n} E_m(x) - i\varepsilon d^{m,n}(x)^T p}$

3) Hop to state n with probability:

$$\boldsymbol{P_{m,n}^{FSSH}(t, t + \Delta t) = \frac{2\Delta t \, Re\left(\rho_{n,m}^{adi}\right) d^{m,n}(x)^T p}{\rho_{m,m}^{adi}}}$$

with $\boldsymbol{\rho_{n,m}^{adi} = c_n^{adi} \overline{c_m^{adi}}}$

Energy conservation only indicates how well step 1) is propagated but is no indicator of the convergence of the other steps.

Issue: typical time step size hierarchy

1) Propagate classical coordinates from t to $t + \Delta t$ at active state m:

$$\dot{\boldsymbol{x}} = \boldsymbol{p}$$
$$\dot{\boldsymbol{p}} = \boldsymbol{F}_{\boldsymbol{m}}^{\boldsymbol{adi}}(\boldsymbol{x})$$

2) Propagate TDSE of the electronic amplitudes from t to $t + \Delta t$:

$$i\varepsilon \dot{\boldsymbol{c}}^{\boldsymbol{adi}}(\boldsymbol{t}) = \boldsymbol{H}^{\boldsymbol{adi}}(\boldsymbol{t})\boldsymbol{c}^{\boldsymbol{adi}}(\boldsymbol{t})$$

with $\boldsymbol{H}_{\boldsymbol{m,n}}^{\boldsymbol{adi}}(\boldsymbol{t}) = \boldsymbol{\delta}_{\boldsymbol{m,n}}\boldsymbol{E}_{\boldsymbol{m}}(\boldsymbol{x}) - i\varepsilon \boldsymbol{d}^{\boldsymbol{m,n}}(\boldsymbol{x})^{\boldsymbol{T}}\boldsymbol{p}$

3) Hop to state n with probability:

$$\boldsymbol{P}_{\boldsymbol{m,n}}^{\boldsymbol{FSSH}}(\boldsymbol{t}, \boldsymbol{t} + \Delta \boldsymbol{t}) = \frac{2\Delta t\, Re\left(\boldsymbol{\rho}_{\boldsymbol{n,m}}^{\boldsymbol{adi}}\right)\boldsymbol{d}^{\boldsymbol{m,n}}(\boldsymbol{x})^{\boldsymbol{T}}\boldsymbol{p}}{\boldsymbol{\rho}_{\boldsymbol{m,m}}^{\boldsymbol{adi}}}$$

with $\boldsymbol{\rho}_{\boldsymbol{n,m}}^{\boldsymbol{adi}} = \boldsymbol{c}_{\boldsymbol{n}}^{\boldsymbol{adi}}\overline{\boldsymbol{c}_{\boldsymbol{m}}^{\boldsymbol{adi}}}$

Step 2) and 3) lead to decreasing Δt due to the exploding behavior of $d^{m,n}$ at intersections

Δt

General Goal

Find numerical efficient formulations and methods of step 2) and 3) such that:

FSSH boosts up and converges faster

Step 2) and 3) become less sensitive then step 1)

Energy conservation can be used as a convergence indicator for FSSH

Goal of this particular presentation

1) Propagate classical coordinates from t to $t + \Delta t$ at active state m:

$$\dot{x} = p$$
$$\dot{p} = F_m^{adi}(x)$$

2) Propagate TDSE of the electronic amplitudes from t to $t + \Delta t$:

$$i\varepsilon \dot{c}^{adi}(t) = H^{adi}(t) c^{adi}(t)$$

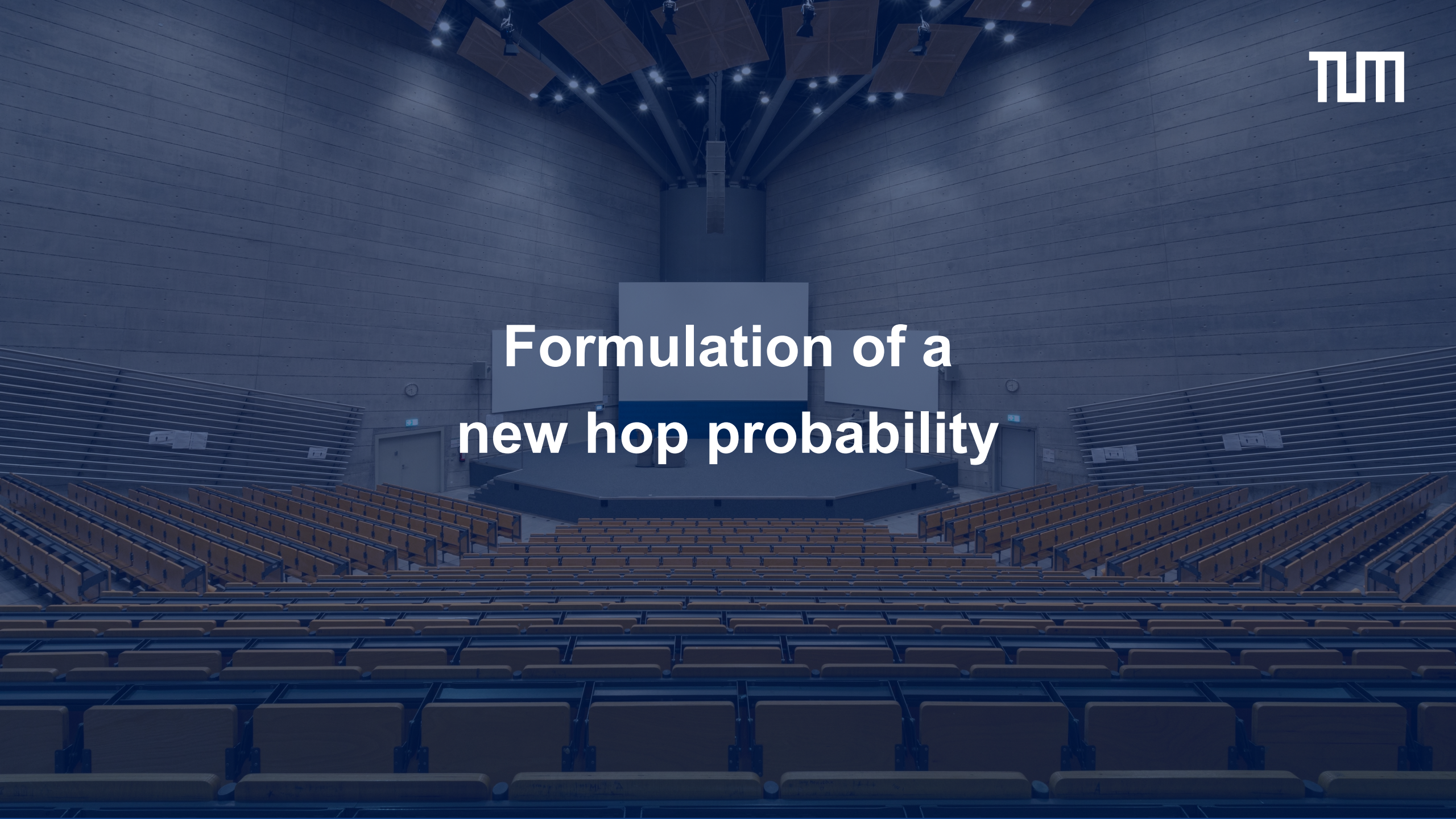
with $H_{m,n}^{adi}(t) = \delta_{m,n} E_m(x) - i\varepsilon d^{m,n}(x)^T p$

Find a numerical efficient formulation of $P_{m,n}^{FSSH}$

3) Hop to state n with probability:

$$P_{m,n}^{FSSH}(t, t + \Delta t) = \frac{2\Delta t \, Re\left(\rho_{n,m}^{adi}\right) d^{m,n}(x)^T p}{\rho_{m,m}^{adi}}$$

with $\rho_{n,m}^{adi} = c_n^{adi} \overline{c_m^{adi}}$



Formulation of a new hop probability

Derivation of the hop probability

$$P_{m,out}(t, t+\Delta t) = \frac{\rho^{adi}_{m,m}(t) - \rho^{adi}_{m,m}(t+\Delta t)}{\rho^{adi}_{m,m}(t)}$$

Derivation of the hop probability: FSSH (1/3)

FSSH

$$P_{m,out}(t, t+\Delta t) = \frac{\rho^{adi}_{m,m}(t) - \rho^{adi}_{m,m}(t+\Delta t)}{\rho^{adi}_{m,m}(t)}$$

$$\frac{\rho^{adi}_{m,m}(t) - \rho^{adi}_{m,m}(t+\Delta t)}{\rho^{adi}_{m,m}(t)} \overset{\text{FEM}}{\approx} \frac{\dot{\rho}^{adi}_{m,m}(t)\,\Delta t}{\rho^{adi}_{m,m}(t)}$$

Derivation of the hop probability: FSSH (2/3)

FSSH

$$P_{m,out}(t, t+\Delta t) = \frac{\rho^{adi}_{m,m}(t) - \rho^{adi}_{m,m}(t+\Delta t)}{\rho^{adi}_{m,m}(t)}$$

$$\frac{\rho^{adi}_{m,m}(t) - \rho^{adi}_{m,m}(t+\Delta t)}{\rho^{adi}_{m,m}(t)} \overset{\text{FEM}}{\approx} \frac{\dot{\rho}^{adi}_{m,m}(t)\,\Delta t}{\rho^{adi}_{m,m}(t)}$$

$$\frac{\dot{\rho}^{adi}_{m,m}(t)\,\Delta t}{\rho^{adi}_{m,m}(t)} = \sum_{n \neq m} \frac{2\Delta t\, Re\left(\rho^{adi}_{n,m}\right) d^{m,n}(x)^T p}{\rho^{adi}_{m,m}}$$

Derivation of the hop probability: FSSH (3/3)

FSSH

$$P_{m,out}(t, t+\Delta t) = \frac{\rho^{adi}_{m,m}(t) - \rho^{adi}_{m,m}(t+\Delta t)}{\rho^{adi}_{m,m}(t)}$$

$$\frac{\rho^{adi}_{m,m}(t) - \rho^{adi}_{m,m}(t+\Delta t)}{\rho^{adi}_{m,m}(t)} \overset{\approx}{\underset{\text{FEM}}{}} \frac{\dot{\rho}^{adi}_{m,m}(t)\,\Delta t}{\rho^{adi}_{m,m}(t)}$$

$$\frac{\dot{\rho}^{adi}_{m,m}(t)\,\Delta t}{\rho^{adi}_{m,m}(t)} = \sum_{n \neq m} \frac{2\Delta t\, Re\left(\rho^{adi}_{n,m}\right) d^{m,n}(x)^T p}{\rho^{adi}_{m,m}}$$

$$P_{m,out} \overset{!}{=} \sum_{n \neq m} P_{m,n}$$

$$P^{FSSH}_{m,n}(t, t+\Delta t) := \frac{2\Delta t\, Re\left(\rho^{adi}_{n,m}\right) d^{m,n}(x)^T p}{\rho^{adi}_{m,m}}$$

Derivation of the hop probability: FSSH2 (1/3)

FSSH

$$P_{m,out}(t, t+\Delta t) = \frac{\rho^{adi}_{m,m}(t) - \rho^{adi}_{m,m}(t+\Delta t)}{\rho^{adi}_{m,m}(t)}$$

FSSH2

$$\frac{\rho^{adi}_{m,m}(t) - \rho^{adi}_{m,m}(t+\Delta t)}{\rho^{adi}_{m,m}(t)} \overset{\text{FEM}}{\approx} \frac{\dot{\rho}^{adi}_{m,m}(t)\,\Delta t}{\rho^{adi}_{m,m}(t)}$$

$$\sum_{n} \rho^{adi}_{n,n}(t) = 1 \Leftrightarrow \rho^{adi}_{m,m}(t) = 1 - \sum_{n \neq m} \rho^{adi}_{n,n}(t)$$

$$\frac{\dot{\rho}^{adi}_{m,m}(t)\,\Delta t}{\rho^{adi}_{m,m}(t)} = \sum_{n \neq m} \frac{2\Delta t\, Re\left(\rho^{adi}_{n,m}\right) d^{m,n}(x)^T p}{\rho^{adi}_{m,m}}$$

$$P_{m,out} \overset{!}{=} \sum_{n \neq m} P_{m,n}$$

$$P^{FSSH}_{m,n}(t, t+\Delta t) := \frac{2\Delta t\, Re\left(\rho^{adi}_{n,m}\right) d^{m,n}(x)^T p}{\rho^{adi}_{m,m}}$$

Derivation of the hop probability: FSSH2 (2/3)

FSSH

$$P_{m,out}(t, t+\Delta t) = \frac{\rho^{adi}_{m,m}(t) - \rho^{adi}_{m,m}(t+\Delta t)}{\rho^{adi}_{m,m}(t)}$$

$$\frac{\rho^{adi}_{m,m}(t) - \rho^{adi}_{m,m}(t+\Delta t)}{\rho^{adi}_{m,m}(t)} \overset{\approx}{\underset{\text{FEM}}{}} \frac{\dot{\rho}^{adi}_{m,m}(t)\,\Delta t}{\rho^{adi}_{m,m}(t)}$$

$$\frac{\dot{\rho}^{adi}_{m,m}(t)\,\Delta t}{\rho^{adi}_{m,m}(t)} = \sum_{n \neq m} \frac{2\Delta t\, Re\left(\rho^{adi}_{n,m}\right) d^{m,n}(x)^T p}{\rho^{adi}_{m,m}}$$

$$P_{m,out} \overset{!}{=} \sum_{n \neq m} P_{m,n}$$

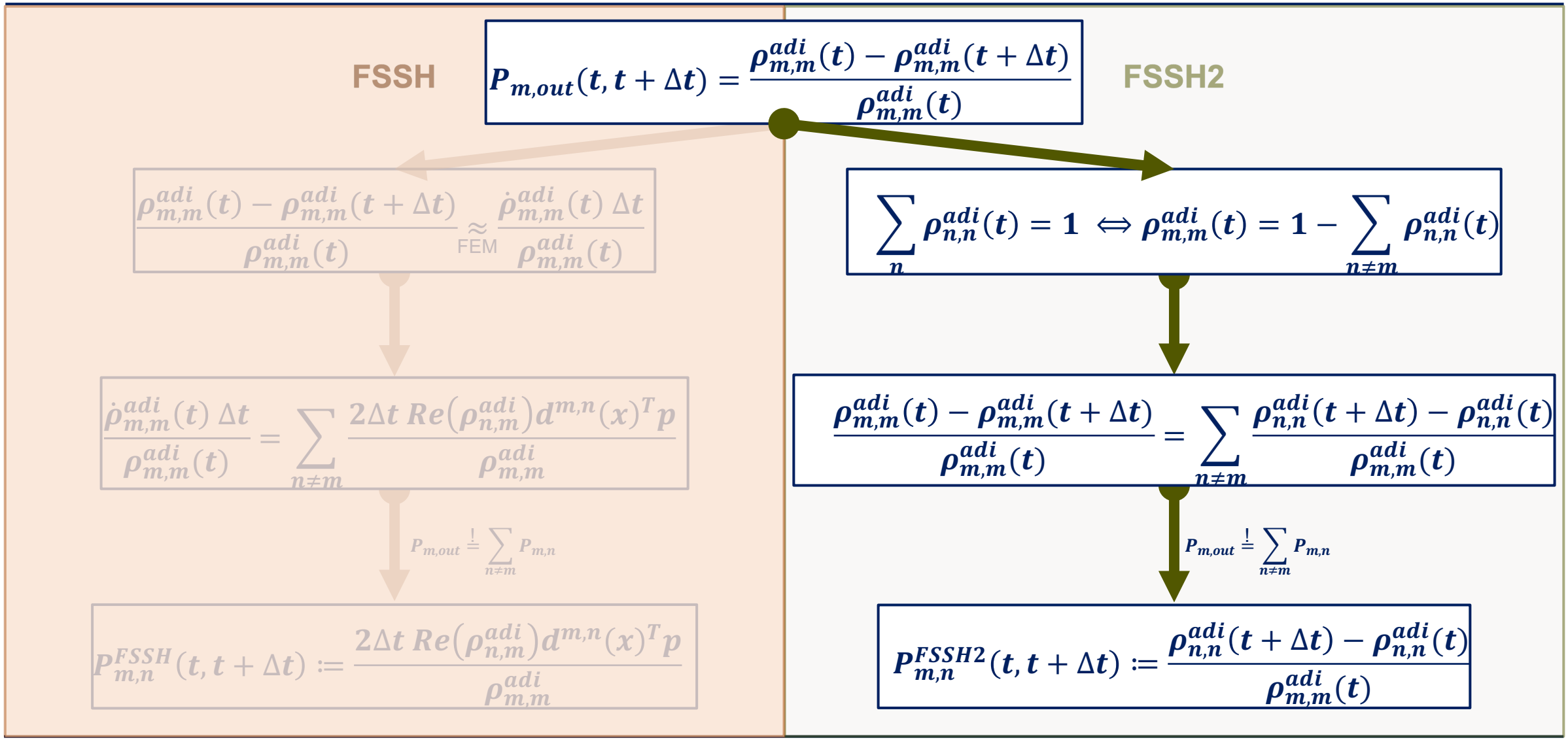
$$P^{FSSH}_{m,n}(t, t+\Delta t) := \frac{2\Delta t\, Re\left(\rho^{adi}_{n,m}\right) d^{m,n}(x)^T p}{\rho^{adi}_{m,m}}$$

FSSH2

$$\sum_n \rho^{adi}_{n,n}(t) = 1 \Leftrightarrow \rho^{adi}_{m,m}(t) = 1 - \sum_{n \neq m} \rho^{adi}_{n,n}(t)$$

$$\frac{\rho^{adi}_{m,m}(t) - \rho^{adi}_{m,m}(t+\Delta t)}{\rho^{adi}_{m,m}(t)} = \sum_{n \neq m} \frac{\rho^{adi}_{n,n}(t+\Delta t) - \rho^{adi}_{n,n}(t)}{\rho^{adi}_{m,m}(t)}$$

Derivation of the hop probability: FSSH2 (3/3)



Theoretical advantages of FSSH2 over FSSH

$$P_{m,n}^{FSSH}(t, t+\Delta t) := \frac{2\Delta t\, Re\left(\rho_{n,m}^{adi}\right) d^{m,n}(x)^T p}{\rho_{m,m}^{adi}}$$

vs.

$$P_{m,n}^{FSSH2}(t, t+\Delta t) := \frac{\rho_{n,n}^{adi}(t+\Delta t) - \rho_{n,n}^{adi}(t)}{\rho_{m,m}^{adi}(t)}$$

- **Numerical stable probability formula**
- **For a total of only two electronic states, we have $P_{m,n}^{FSSH2} = P_{m,out} \overset{\text{FEM}}{\approx} P_{m,n}^{FSSH}$**
- **FSSH2 can handle trivial crossings (depending on the TDSE propagator)**

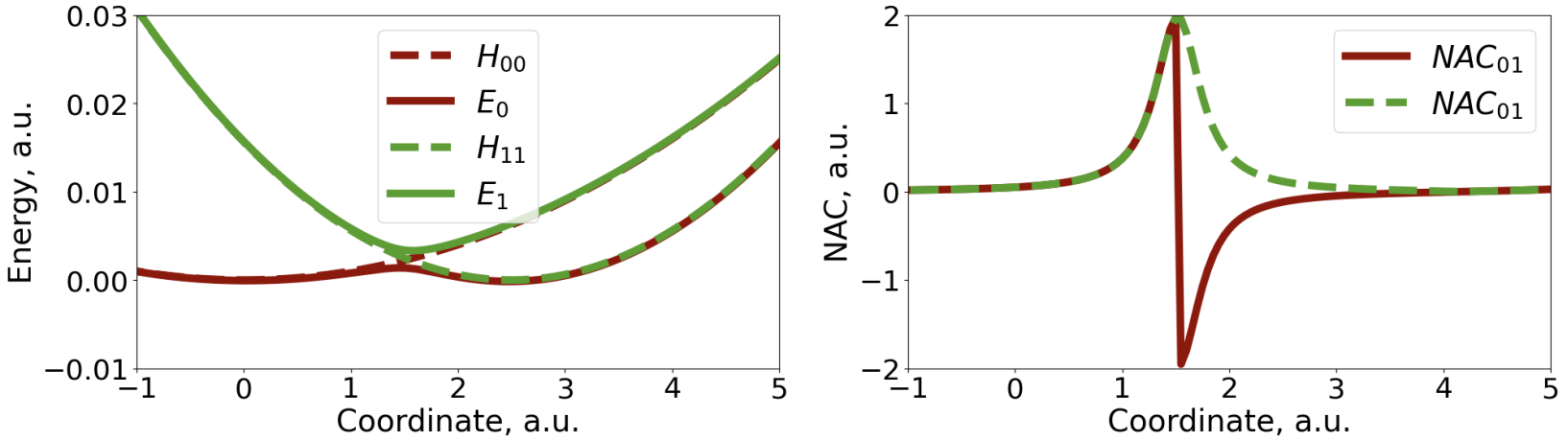
Numerical experiment

The Holstein model

For our simulations, we choose the one-dimensional Holstein model given by the following diabatic potential energy matrix for two electronic states,

$$V^{dia}(x) = \begin{pmatrix} U_1 + \frac{k_1}{2}(x - y_1)^2 & V \\ V & U_2 + \frac{k_2}{2}(x - y_2)^2 \end{pmatrix}.$$

We set: $U = \begin{pmatrix} 0 \\ -0.001 \end{pmatrix}$, $y = \begin{pmatrix} 0 \\ 0.5 \end{pmatrix}$, $k = \begin{pmatrix} 0.002 \\ 0.008 \end{pmatrix}$, and $V = 0.001$.



Initial wavefunction

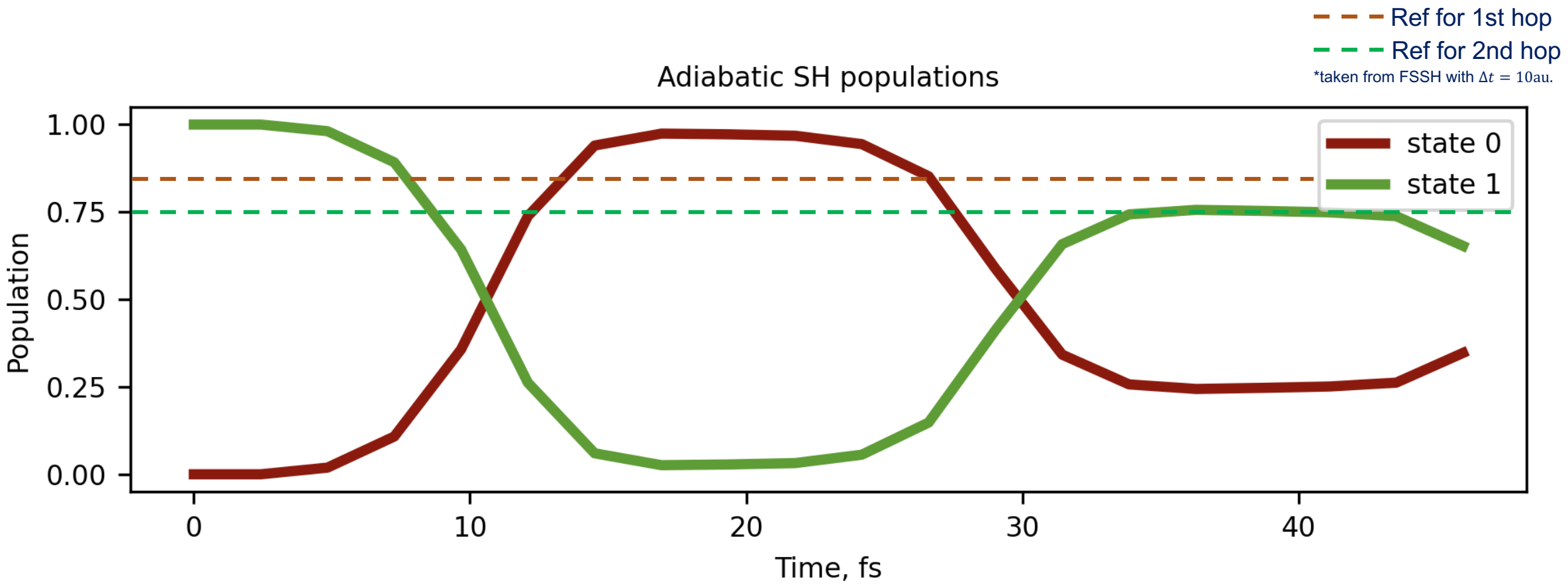
Gaussian wave packet centered at:

- $\begin{pmatrix} q \\ p \end{pmatrix} = \begin{pmatrix} -2 \\ 0 \end{pmatrix}$
- width matrix $\Sigma = \begin{pmatrix} \sqrt{\frac{1}{2\sqrt{0.01 \cdot \boldsymbol{M}}}} & 0 \\ 0 & \sqrt{\frac{\sqrt{0.01 \cdot \boldsymbol{M}}}{2}} \end{pmatrix}$
- located at the second diabatic state

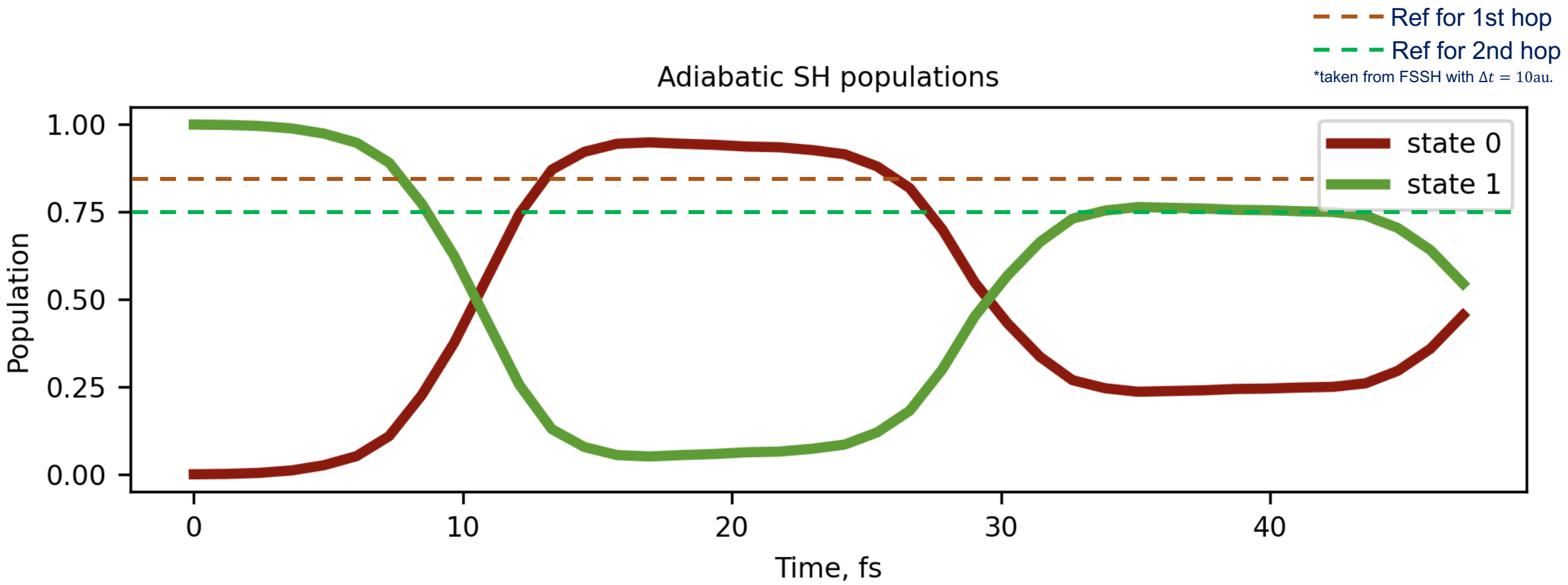
Further parameters

- **Mass M:** 2000au.
- **Simulation time:** 2000au. / ca. 50fs
- **Time step sizes Δt in au:** 100, 50, 20, 10
- **Methods:** FSSH, FSSH2, and GFSH (for Alexey)
- **Number of trajectories:** 1000 (5000 would be better to observe fewer deviations – but longer running time)

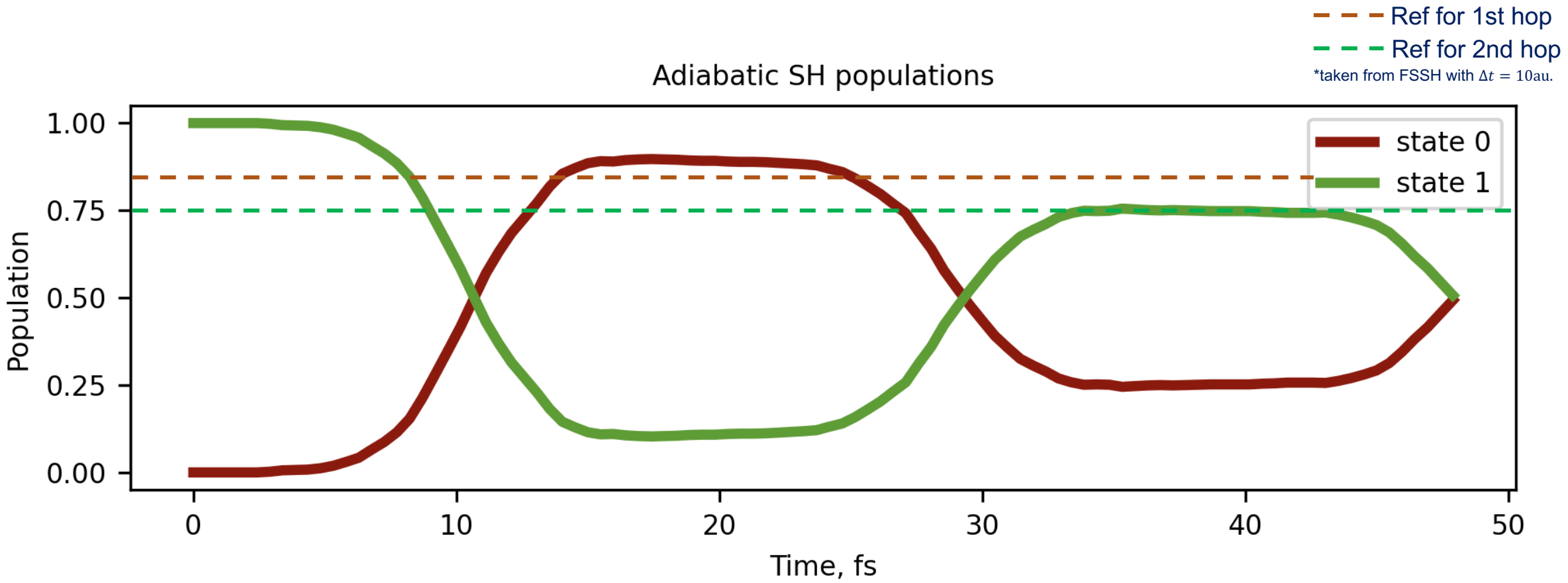
FSSH – $\Delta t = 100\text{au}$



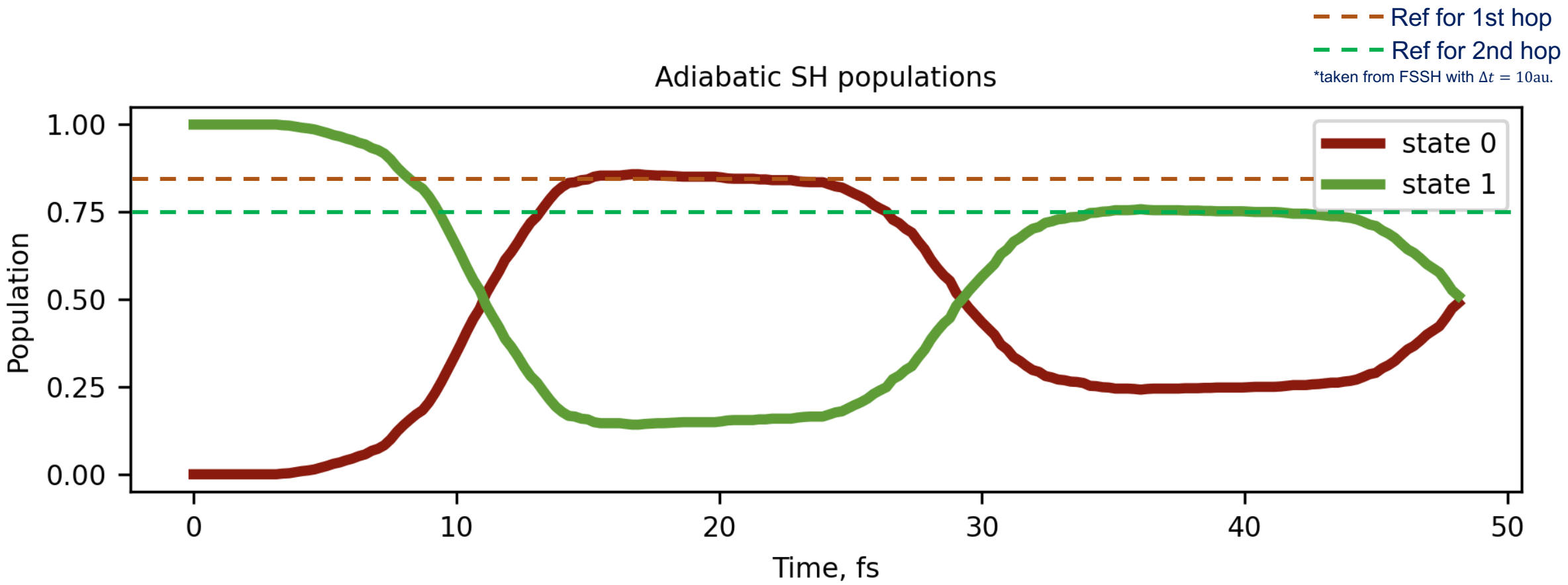
FSSH – $\Delta t = 50\text{au}$



FSSH – $\Delta t = 20\text{au}$

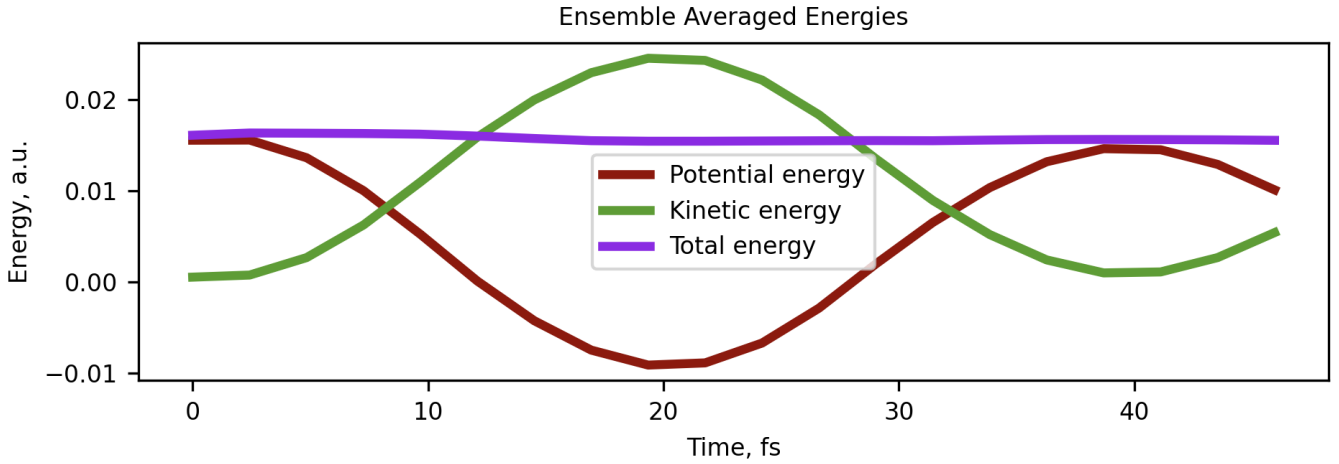


FSSH – $\Delta t = 10\text{au}$

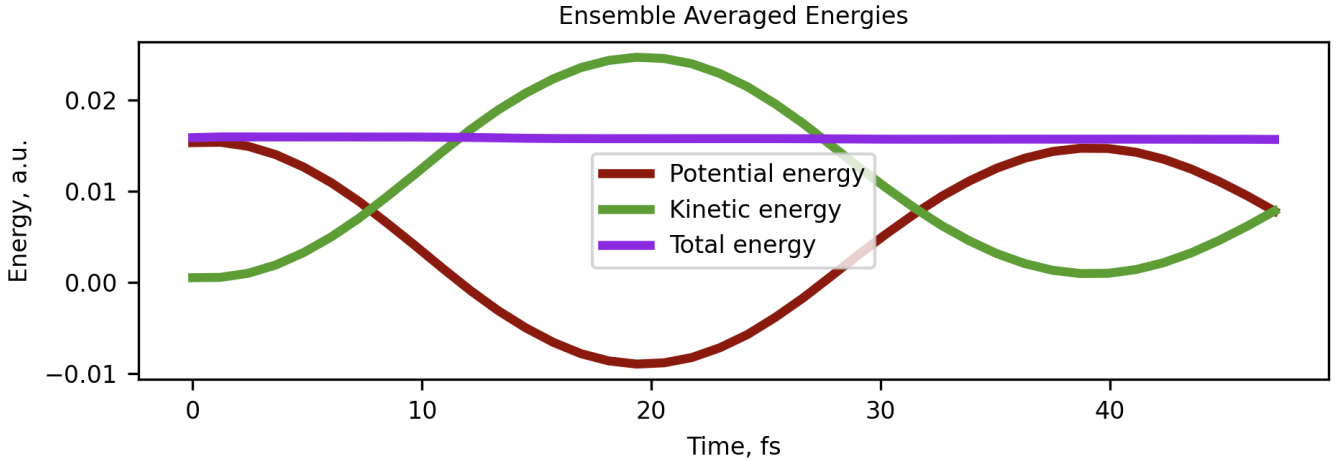


FSSH – energy conservation

$\Delta t = 100$:



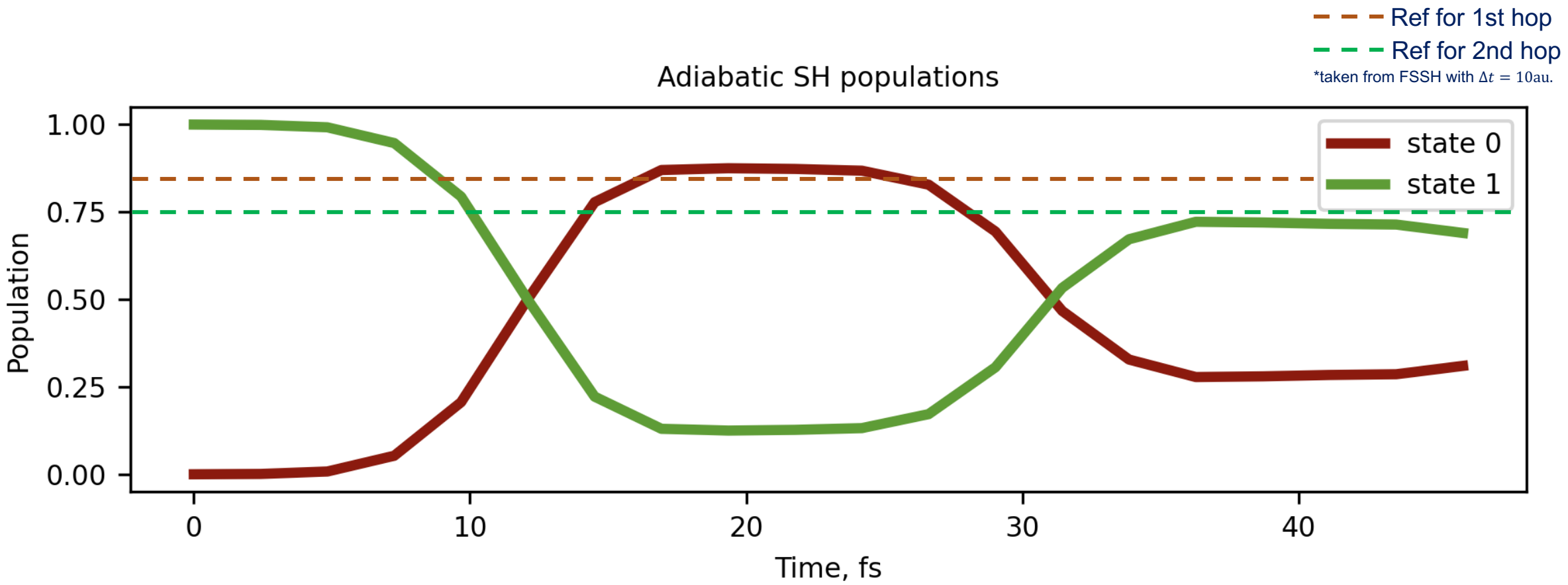
$\Delta t = 50$:



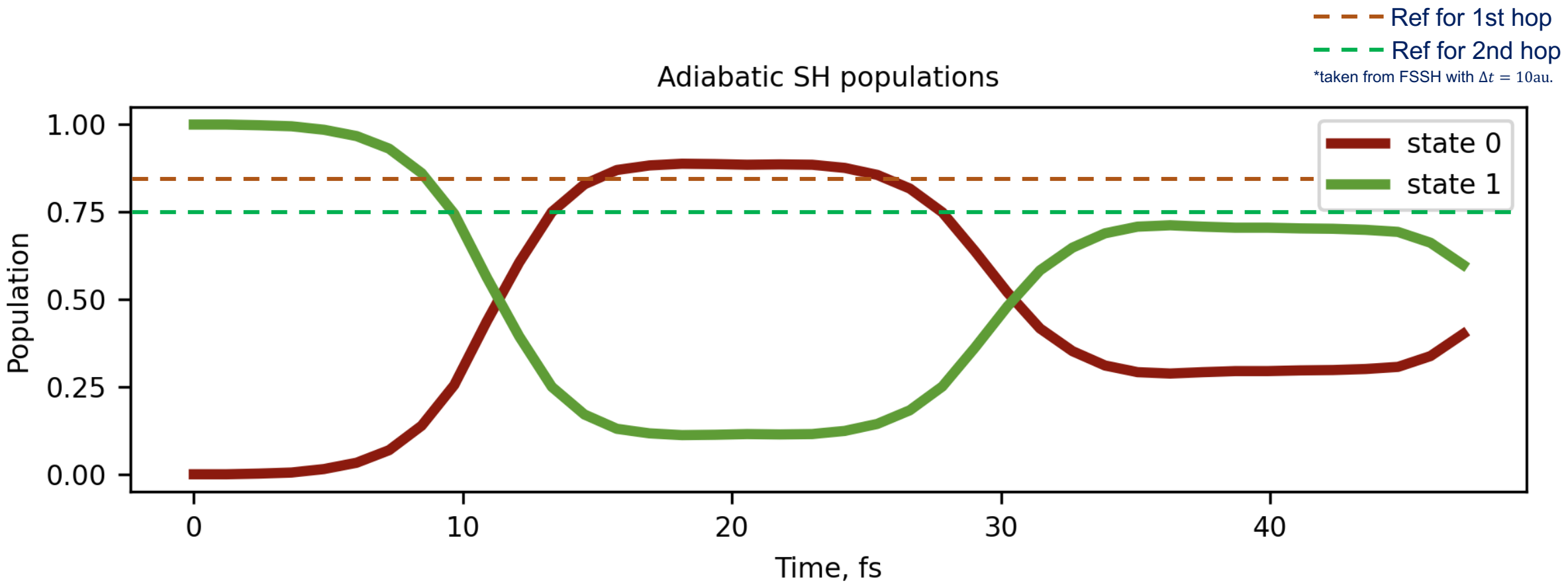
FSSH – observations

- Energy was already conserved at $\Delta t = 100\text{au}$.
- However, FSSH started only to converge at around $\Delta t = 10 \text{ to } 20\text{au}$.

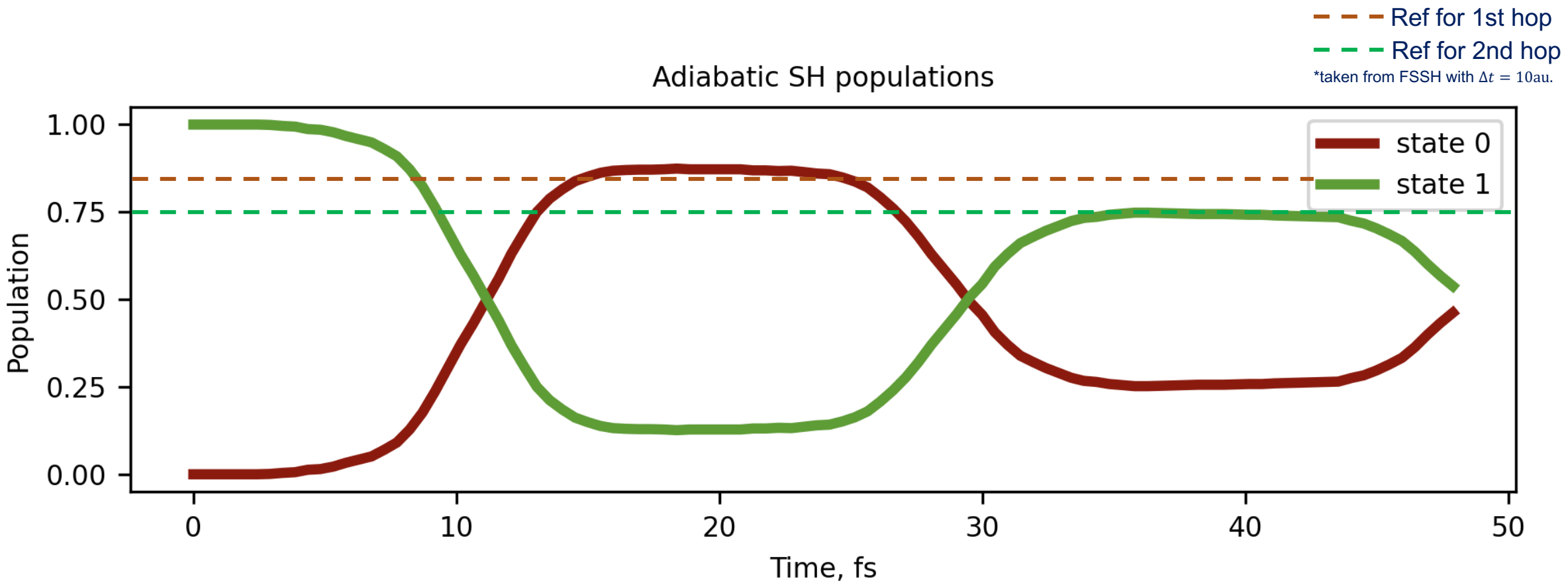
FSSH2 – $\Delta t = 100\text{au}$



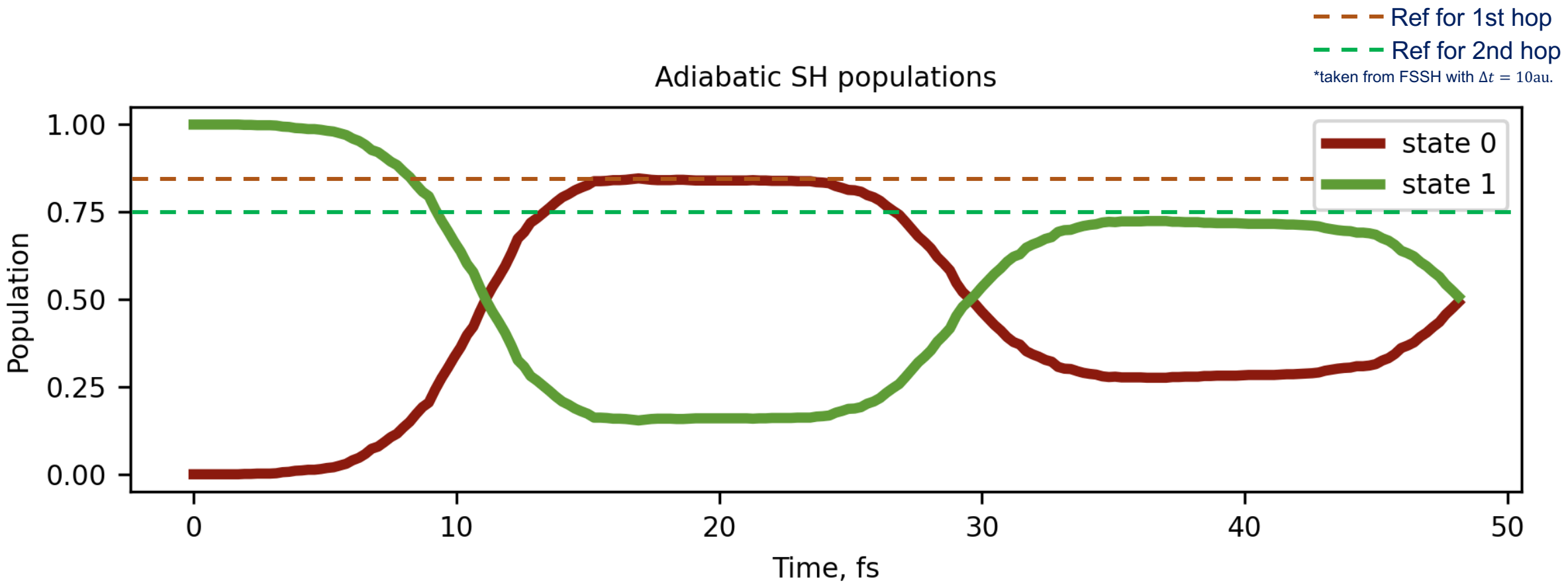
FSSH2 – $\Delta t = 50\text{au}$



FSSH2 – $\Delta t = 20\text{au}$

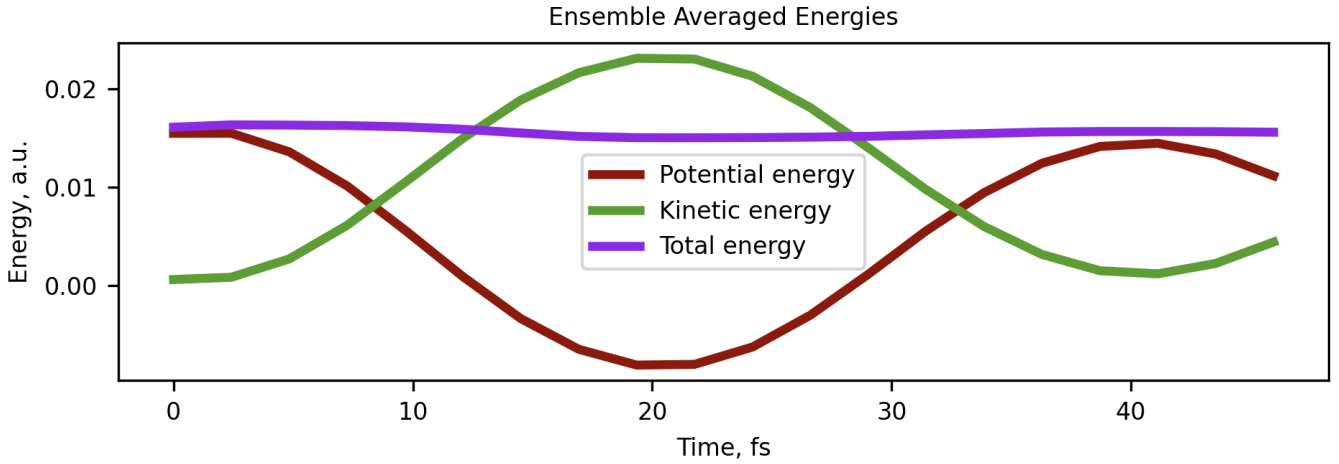


FSSH2 – $\Delta t = 10\text{au}$

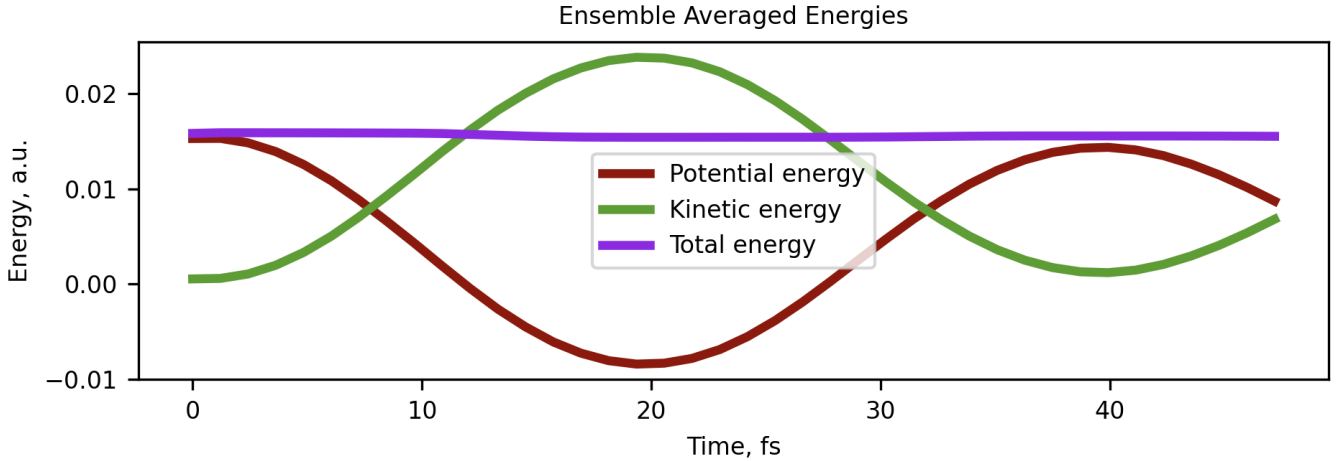


FSSH2 – energy conservation

$\Delta t = 100$:



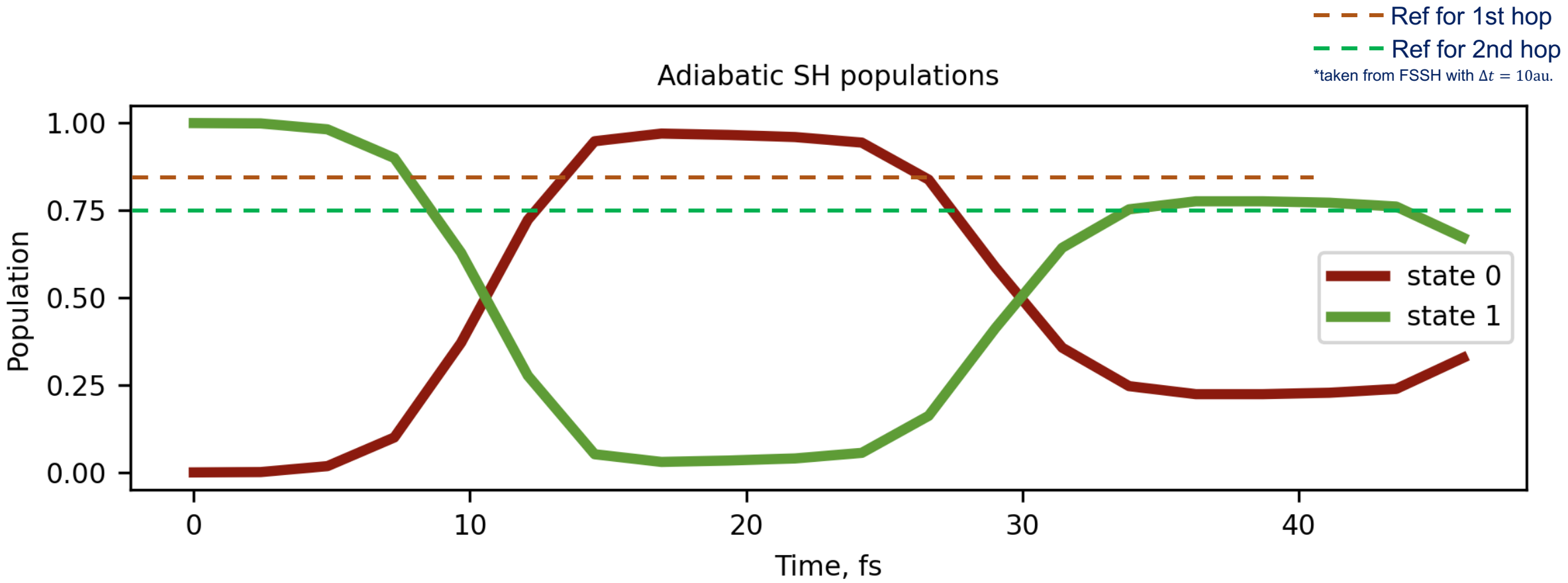
$\Delta t = 50$:



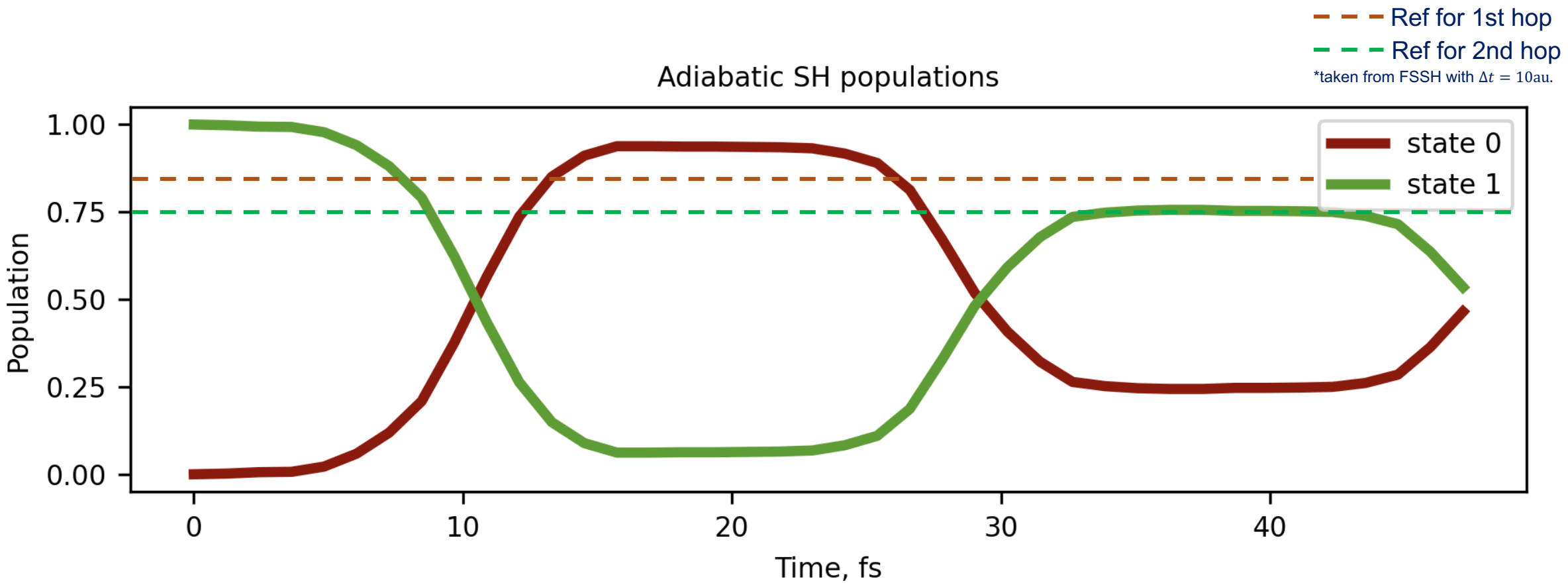
FSSH2 – observations

- Energy was already conserved at $\Delta t = 100\text{au}$.
- Accordingly, FSSH2 provided converged results at around $\Delta t = 100\text{au}$. (I could not try larger time step sizes since the system showed an error: division by zero)

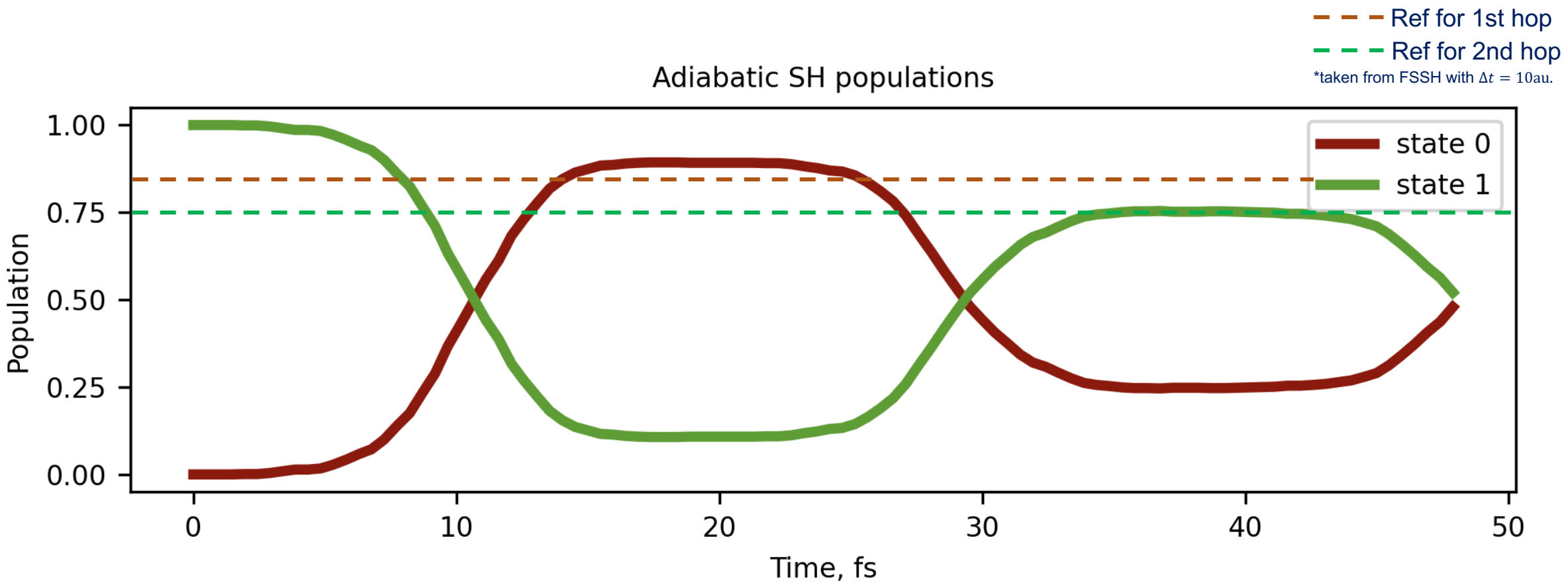
GFSH – $\Delta t = 100\text{au}$



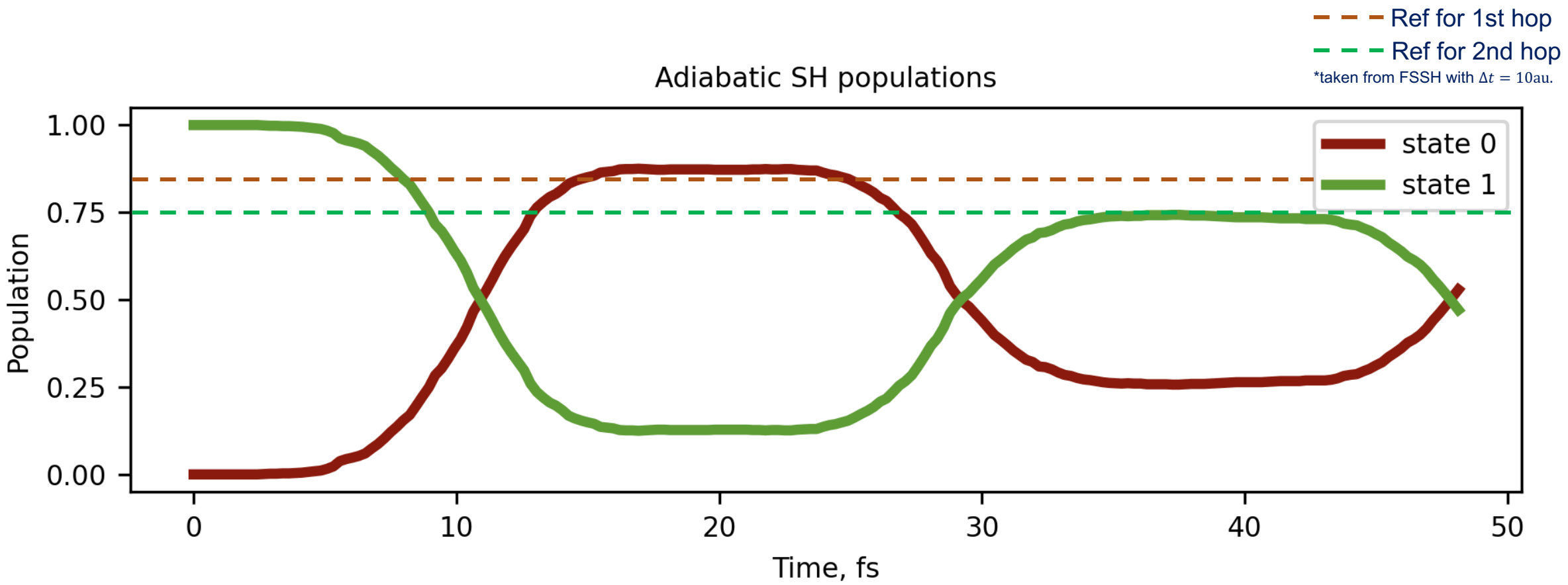
GFSH – $\Delta t = 50\text{au}$



GFSH – $\Delta t = 20\text{au}$

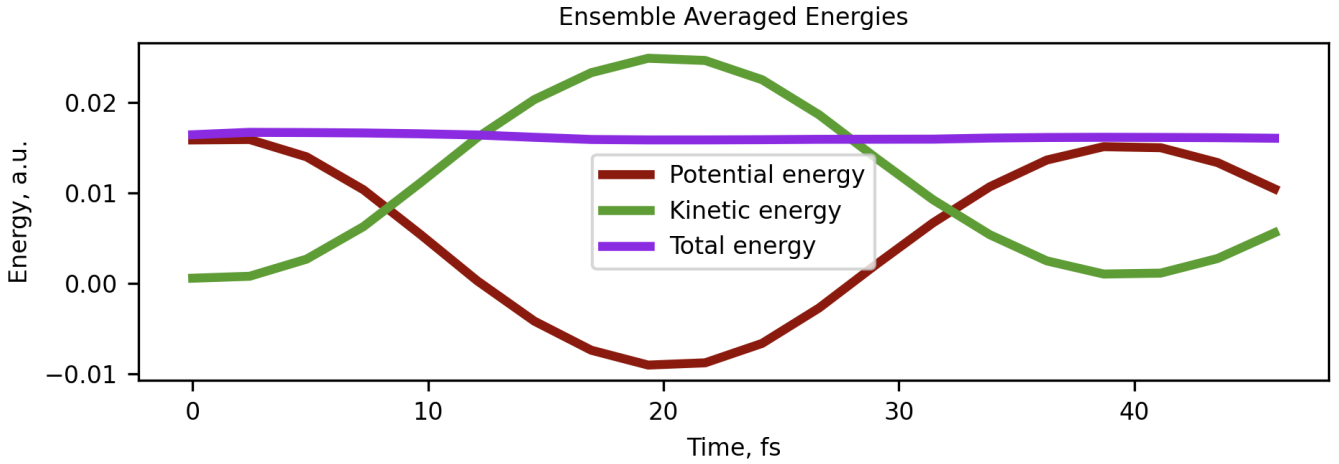


GFSH – $\Delta t = 10\text{au}$

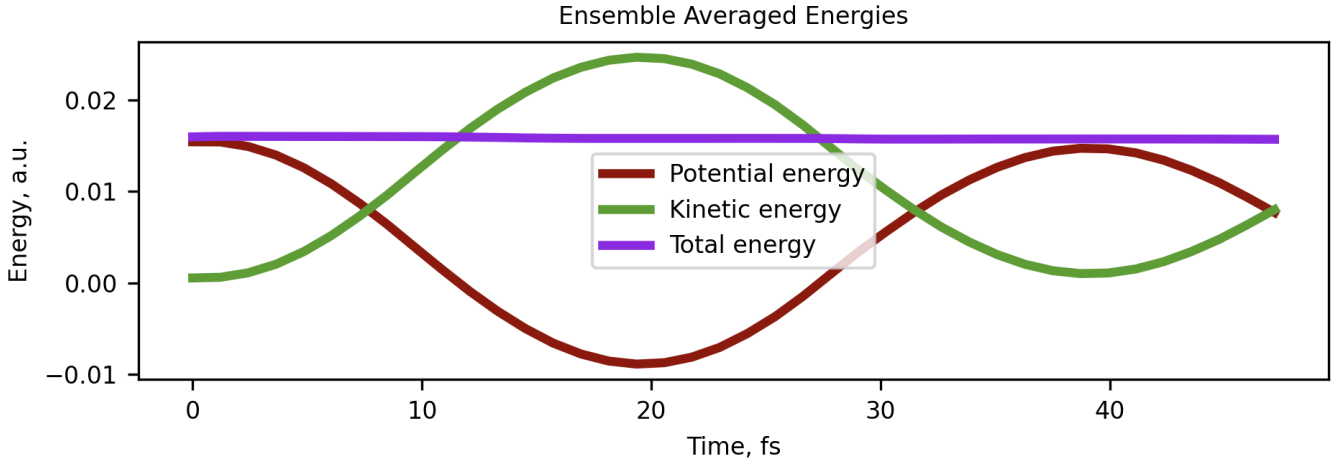


GFSH – energy conservation

$\Delta t = 100$:



$\Delta t = 50$:



GFSH – observations

- Same observations as for FSSH.

- It underperforms FSSH2, although their hop formulas are similar, because the implementation in Libra contains a Finite Element Method approximation of the original GFSH formula.

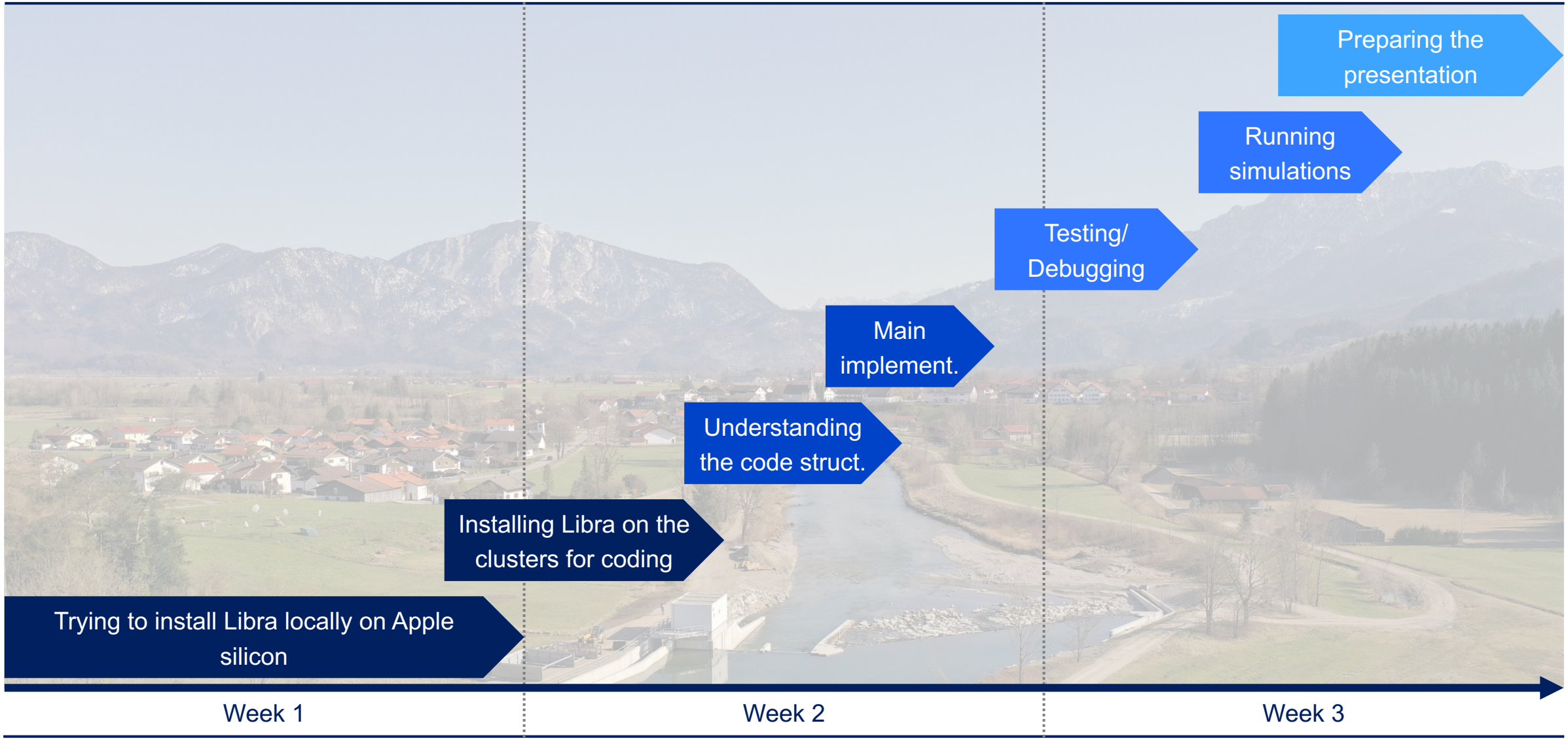
Conclusions

FSSH2 overperformed FSSH by significantly improving the populations for larger time steps and showing less sensitivity.

Energy conservation was no indicator of the convergence of FSSH2 but not of FSSH and GFSH.

Time & workflow overview

Time overview



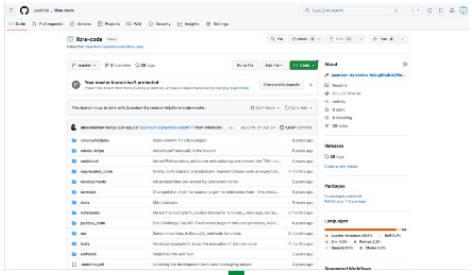
Workflow

CCR

Loc

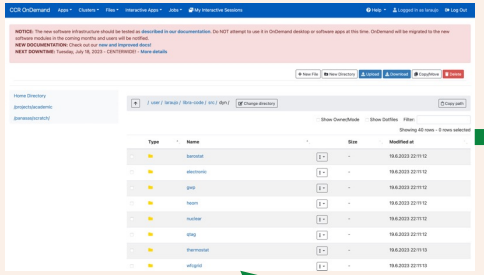
GitHub (Fork)

Pull from the Libra package



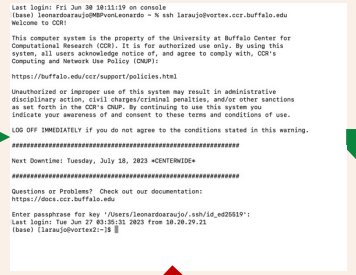
CCR OnDemand

Upload modifications



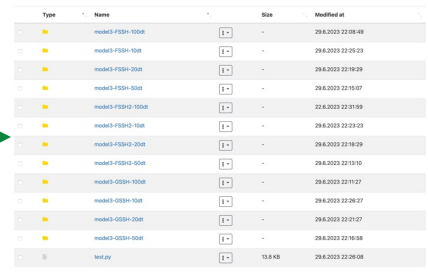
Terminal

Compile/test/simulate

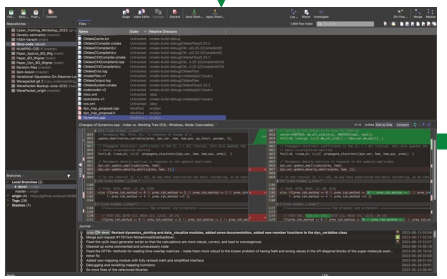


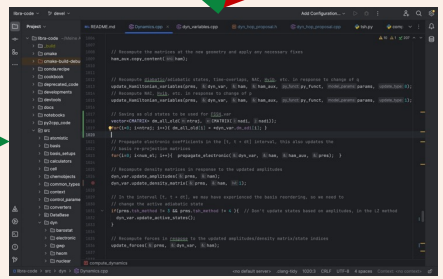
CCR OnDemand

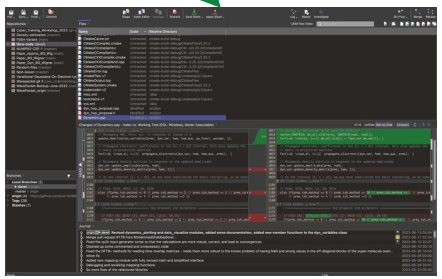
Download results



Implementation loop







SmartGit

Pull locally/observe modifications

CLion (C++ IDE)

Code/check syntax

SmartGit

Push modifications

Modified files: Dynamics.cpp

Saving densities of the previous time step:

```
1012  // Recompute diabatic/adiabatic states, time-overlaps, NAC, Hvib, etc. in response to change of q
1013  update_Hamiltonian_variables(prms, dyn_var, ham, ham_aux, py_funct, params, 0);
1014  // Recompute NAC, Hvib, etc. in response to change of p
1015  update_Hamiltonian_variables(prms, dyn_var, ham, ham_aux, py_funct, params, 1);
1016
1017  // Saving as old states to be used for FSSH_var
1018  vector<CMATRIX> dm_all_old(ntraj, CMATRIX(nadi, nadi));
1019  for(i=0; i<ntraj; i++){ dm_all_old[i] = *dyn_var.dm_adi[i]; }
1020
1021  // Propagate electronic coefficients in the [t, t + dt] interval, this also updates the
1022  // basis re-projection matrices
1023  for(i=0; i<num_el; i++){  propagate_electronic(dyn_var, ham, ham_aux, prms);  }
1024
```

Setting tsh_method=10 as FSSH2:

```
1165  // FSSH, GFSH, MSSH, LZ, ZN, DISH
1166  else if(prms.tsh_method == 0 || prms.tsh_method == 10 || prms.tsh_method == 1 || prms.tsh_method == 2 |
1167       || prms.tsh_method == 4 || prms.tsh_method ==5 ){
1168
1169
1170    vector<int> old_states(dyn_var.act_states);
1171
1172    //========================= Hop proposal and acceptance ==============================
1173
1174    // FSSH (0), FSSH_var (10), GFSH (1), MSSH (2), LZ(3), ZN (4)
1175    if(prms.tsh_method == 0 || prms.tsh_method == 10 || prms.tsh_method == 1 || prms.tsh_method == 2 || p
1176
1177      /// Compute hop proposal probabilities from the active state of each trajectory to all other states
1178      /// of that trajectory
1179      vector< vector<double> > g;
1180      g = hop_proposal_probabilities(prms, dyn_var, dm_all_old, ham, ham_aux);
1181
1182      // Propose new discrete states for all trajectories
1183      vector<int> prop_states( propose_hops(g, dyn_var.act_states, rnd) );
1184
```

Modified files: dyn_hop_proposal.h

Introducing a new method to compute hop probabilities with the previous densities as additional input:

```
66 vector< vector<double> > hop_proposal_probabilities(dyn_control_params& prms,
67       dyn_variables& dyn_var, nHamiltonian& ham, nHamiltonian& ham_prev);
68
69 vector< vector<double> > hop_proposal_probabilities(dyn_control_params& prms,
70       dyn_variables& dyn_var, vector<CMATRIX>& dm_all_old, nHamiltonian& ham, nHamiltonian& ham_prev);
71
72 //vector< vector<double> > hop_proposal_probabilities(dyn_control_params& prms,
73 //       dyn_variables& dyn_var, nHamiltonian& ham, vector<MATRIX>& prev_ham_dia);
74
```

Modified files: dyn_hop_proposal.cpp (1/2)

Introducing a new option in hop_proposal_probabilities(…) to compute FSSH2:

```
1284 if(prms.tsh_method == 0){ // FSSH
1285
1286 g[traj] = hopping_probabilities_fssh(prms, dm, Hvib, dyn_var.act_states[traj]);
1287
1288 }
1289 else if(prms.tsh_method == 10){ // FSSH_var
1290
1291     //CMATRIX& dm_old = *dyn_var_old.dm_adi[traj];
1292     CMATRIX dm_old = dm_all_old[traj];
1293
1294     g[traj] = hopping_probabilities_fssh_var(prms, dm, dm_old, Hvib, dyn_var.act_states[traj]);
1295
1296 }
1297 else if(prms.tsh_method == 1){ // GFSH
1298
1299 g[traj] = hopping_probabilities_gfsh(prms, dm, Hvib, dyn_var.act_states[traj]);
```

Modified files: dyn_hop_proposal.cpp (2/2)

Compute FSSH2 in a new method hopping_probabilities_fssh_var(…):

```
375         // Now calculate the hopping probabilities
376         i = act_state_indx;
377
378         sum = 0.0;
379         double a_ii_old = denmat_old.get(i,i).real();
380 //  cout<<"a_ii = "<<a_ii<<endl;
381
382         for(j=0;j<nstates;j++){
383
384             if(i!=j){
385             /**
386                   dc/dt = -(i/hbar) * Hvib * c
387                   (dc/dt)^+ = i/hbar * c^+ * Hvib^+
388
389                   rho = c * c^+
390
391                   P(i->j) = ( rho_jj - rho_old_jj ) / rho_ii
392
393                */
394
395                 double a_jj_old = denmat_old.get(j,j).real();
396                 double a_jj     = denmat.get(j,j).real();
397
398                 if(a_ii_old<1e-8){ g_ij = 0.0; }  // avoid division by zero
399                 else{
400                     g_ij = (a_jj - a_jj_old)/a_ii_old;  // This is a general case
401
402                     if(g_ij<0.0){  g_ij = 0.0; }
403
404                 }// else
405
406                 g[j] = g_ij;
407                 sum = sum + g_ij;
408             }
409             else{ g[j] = 0.0; }
410
411         }// for j
412
413         g[i] = 1.0 - sum;
414
415         return g;
```


Thank you for your attention!
Any questions?


Proooost!!