Modelling quantum states using normalizing flows

Álvaro Fernández, Yahya Saleh, Andrey Yachmenev, Jochen Küpper

Controlled Molecule Imaging Group

Center for Free-Electron Laser Science, Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany
Department of Physics, Universität Hamburg
Department of Chemistry, Universität Hamburg
Center for Ultrafast Imaging, Universität Hamburg

CUI THE HAMBURG CENTER FOR ULTRAFAST IMAGING

UHH Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

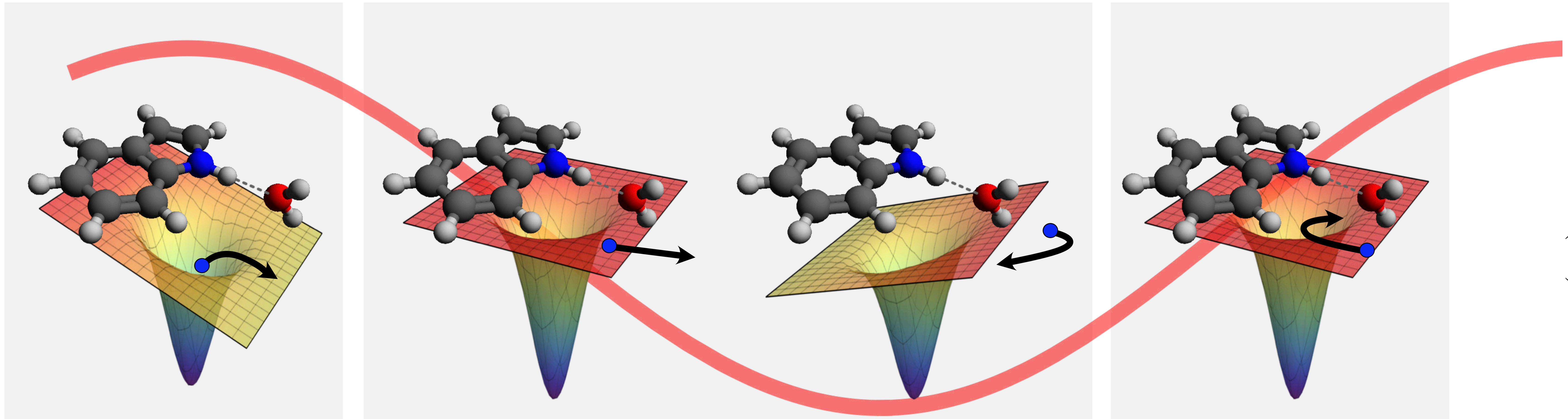
CFEL SCIENCE
Controlled Molecule Imaging

DESY.

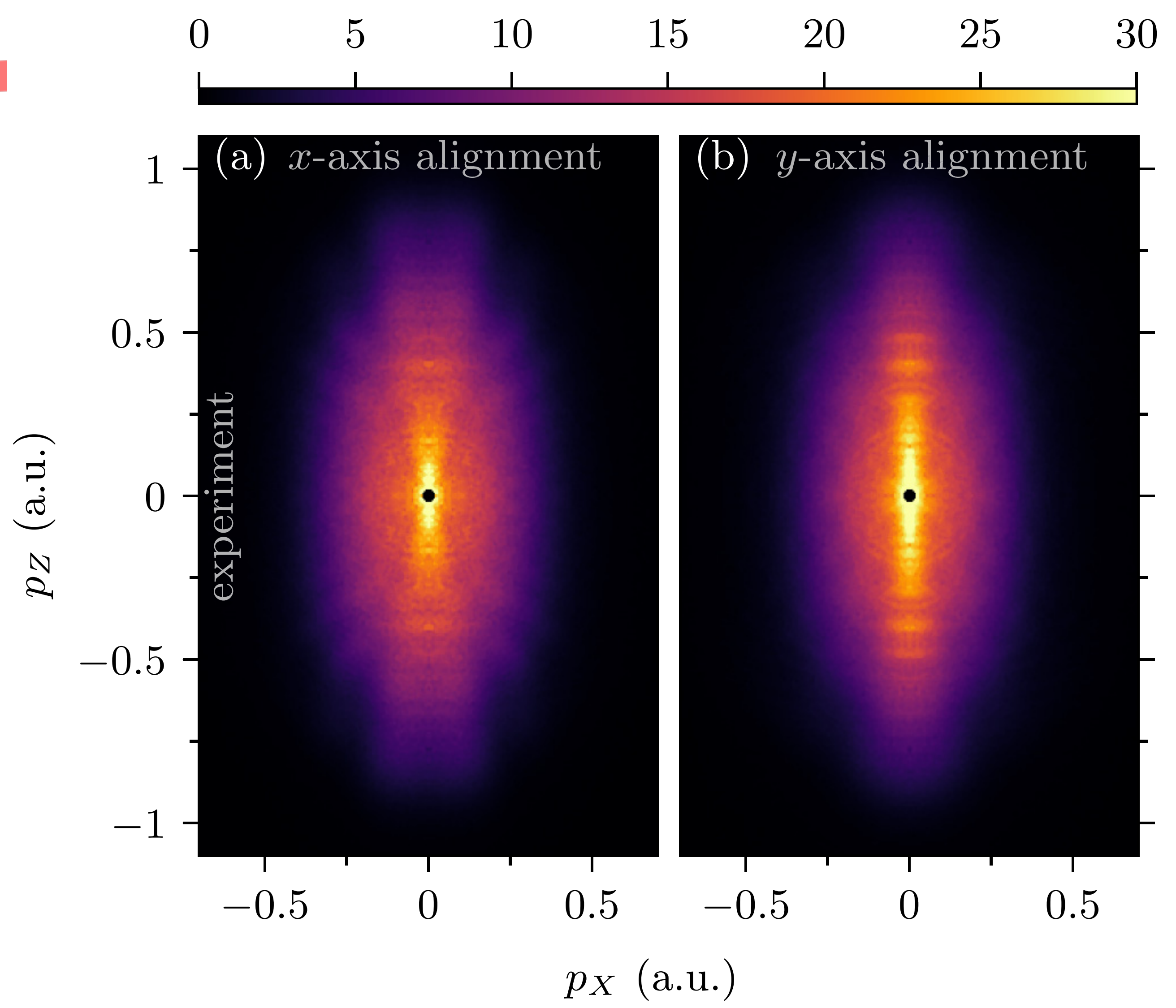
Objectives

- Create simulations for the LIED experiment.

Ultrafast imaging of dissociation dynamics with LIED



Ionisation | *Propagation* | *Rescattering*



Quantum | Classical | Classical → **solution of classical ODEs with neural networks, PAD features ↔ trajectory type**

Quantum | Quantum | Quantum → **solution of quantum ODEs with neural networks (LSTM, quantum flows, …)**

Objectives

- Create simulations for the LIED experiment.

Objectives

- Create simulations for the LIED experiment.
- Solve Time Dependent Schrödinger equation.

Objectives

- Create simulations for the LIED experiment.
- Solve Time Dependent Schrödinger equation.
- Solve Time Independent Schrödinger equation for many states.

Objectives

- Create simulations for the LIED experiment.
- Solve Time Dependent Schrödinger equation.
- Solve Time Independent Schrödinger equation for many states.

Time Independent Schrödinger Equation in Molecules

- Partial Derivative Eigenvalue equation:

$$H(x_1, x_2, \dots)\Psi_i(x_1, x_2, \dots) = [-\frac{1}{2}\Delta_i^2 + V(x_1, x_2, \dots)]\Psi_i(x_1, x_2, \dots) = E_i\Psi_i(x_1, x_2, \dots)$$

- Generally not solvable due to high degree of freedom.
- Need of numerical approximations.
  - Basis expansion:

$$\Psi_i(x_1, x_2, \dots) = \sum_{i_1, i_2, \dots = 0}^{N} c_{i_1, i_2, \dots}\phi_{i_1}(x_1)\phi_{i_2}(x_2)\dots$$

  - How can we improve this approximation?

Normalizing flows

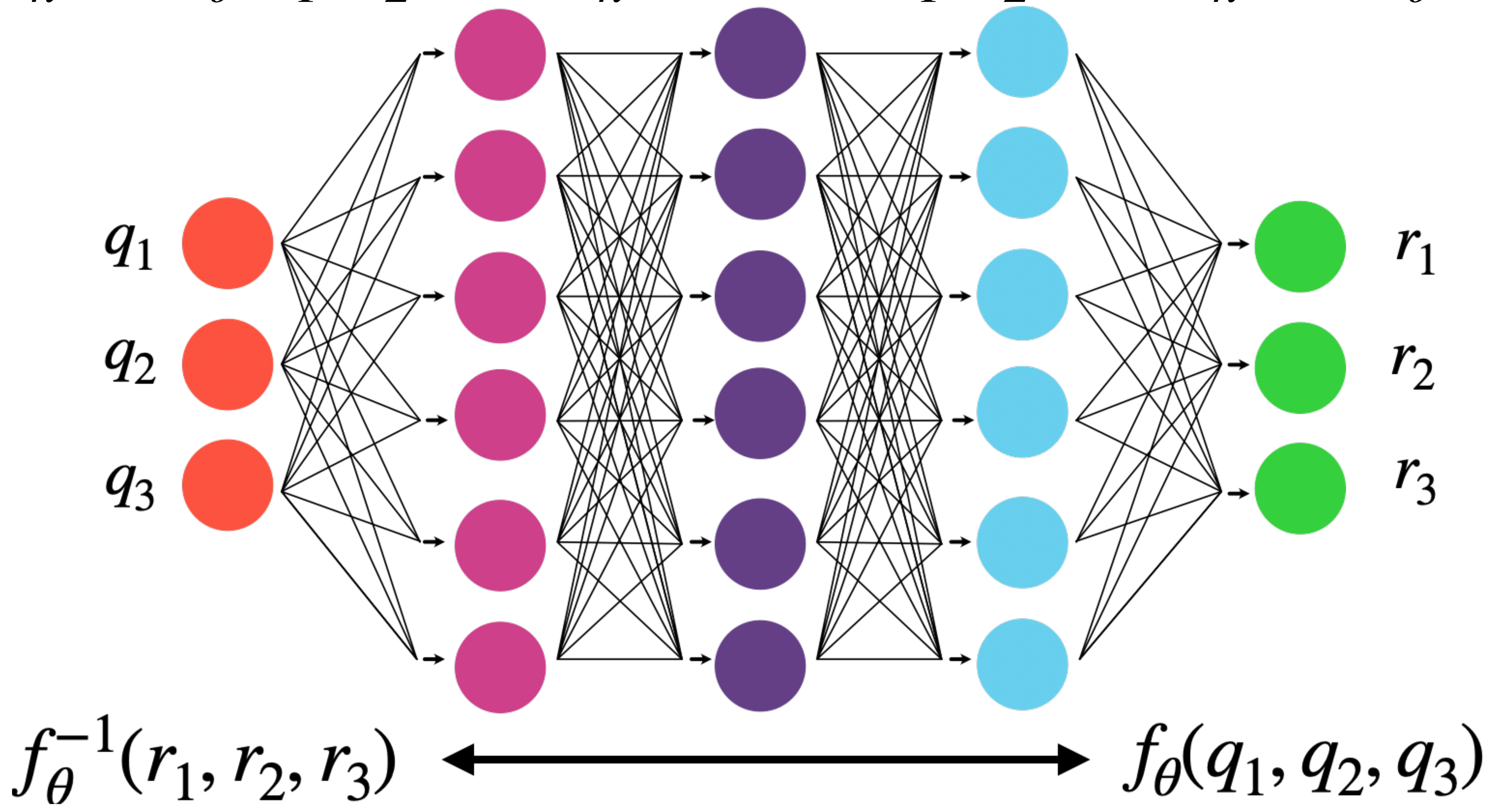
- Augment the basis by a flow of the intrinsic coordinates to a support space. Let this flow be a Neural Network:

$$(x_1, x_2, \ldots, x_n) = g_\theta(q_1, q_2, \ldots, q_n) \longrightarrow H(x_1, x_2, \ldots, x_n) = H_\theta(q_1, q_2, \ldots, q_n)$$

Normalizing flows

- Augment the basis by a flow of the intrinsic coordinates to a support space. Let this flow be a Neural Network:

$$(x_1, x_2, \ldots, x_n) = g_\theta(q_1, q_2, \ldots, q_n) \longrightarrow H(x_1, x_2, \ldots, x_n) = H_\theta(q_1, q_2, \ldots, q_n)$$



Normalizing flows

- Augment the basis by a flow of the intrinsic coordinates to a support space. Let this flow be a Neural Network:

$$(x_1, x_2, \ldots, x_n) = g_\theta(q_1, q_2, \ldots, q_n) \longrightarrow H(x_1, x_2, \ldots, x_n) = H_\theta(q_1, q_2, \ldots, q_n)$$

Normalizing flows

- Augment the basis by a flow of the intrinsic coordinates to a support space. Let this flow be a Neural Network:

$$(x_1, x_2, \ldots, x_n) = g_\theta(q_1, q_2, \ldots, q_n) \longrightarrow H(x_1, x_2, \ldots, x_n) = H_\theta(q_1, q_2, \ldots, q_n)$$

- This also impacts the basis functions of the expansion:

Normalizing flows

- Augment the basis by a flow of the intrinsic coordinates to a support space. Let this flow be a Neural Network:

$$(x_1, x_2, \ldots, x_n) = g_\theta(q_1, q_2, \ldots, q_n) \longrightarrow H(x_1, x_2, \ldots, x_n) = H_\theta(q_1, q_2, \ldots, q_n)$$

- This also impacts the basis functions of the expansion:

$$\phi_i^A(x) = \phi_i(g_\theta(x))\sqrt{|\det \frac{\partial g_\theta(x)}{\partial x}|}$$

Normalizing flows

- Augment the basis by a flow of the intrinsic coordinates to a support space. Let this flow be a Neural Network:

$$(x_1, x_2, \ldots, x_n) = g_\theta(q_1, q_2, \ldots, q_n) \longrightarrow H(x_1, x_2, \ldots, x_n) = H_\theta(q_1, q_2, \ldots, q_n)$$

- This also impacts the basis functions of the expansion:

$$\phi_i^A(x) = \phi_i(g_\theta(x))\sqrt{|\det \frac{\partial g_\theta(x)}{\partial x}|}$$

- Or in other words:

Normalizing flows

- Augment the basis by a flow of the intrinsic coordinates to a support space. Let this flow be a Neural Network:

$$(x_1, x_2, \ldots, x_n) = g_\theta(q_1, q_2, \ldots, q_n) \longrightarrow H(x_1, x_2, \ldots, x_n) = H_\theta(q_1, q_2, \ldots, q_n)$$

- This also impacts the basis functions of the expansion:

$$\phi_i^A(x) = \phi_i(g_\theta(x))\sqrt{|\det \frac{\partial g_\theta(x)}{\partial x}|}$$

- Or in other words:

$$\phi_i^A(q) = \phi_i(q)\sqrt{|\det \frac{\partial q}{\partial g_\theta^{-1}(q)}|}$$

Normalizing flows

- Augment the basis by a flow of the intrinsic coordinates to a support space. Let this flow be a Neural Network:

$$(x_1, x_2, \ldots, x_n) = g_\theta(q_1, q_2, \ldots, q_n) \longrightarrow H(x_1, x_2, \ldots, x_n) = H_\theta(q_1, q_2, \ldots, q_n)$$

- This also impacts the basis functions of the expansion:

$$\phi_i^A(x) = \phi_i(g_\theta(x))\sqrt{|\det \frac{\partial g_\theta(x)}{\partial x}|}$$

- Or in other words:

$$\phi_i^A(q) = \phi_i(q)\sqrt{|\det \frac{\partial q}{\partial g_\theta^{-1}(q)}|}$$

- So that we see that, by construction, our Neural Network needs to be invertible. We choose Invertible ResNet[1] as our architecture.

Normalizing flows

- Augment the basis by a flow of the intrinsic coordinates to a support space. Let this flow be a Neural Network:

$$(x_1, x_2, \ldots, x_n) = g_\theta(q_1, q_2, \ldots, q_n) \longrightarrow H(x_1, x_2, \ldots, x_n) = H_\theta(q_1, q_2, \ldots, q_n)$$

- This also impacts the basis functions of the expansion:

$$\phi_i^A(x) = \phi_i(g_\theta(x))\sqrt{|\det \frac{\partial g_\theta(x)}{\partial x}|}$$

- Or in other words:

$$\phi_i^A(q) = \phi_i(q)\sqrt{|\det \frac{\partial q}{\partial g_\theta^{-1}(q)}|}$$

- So that we see that, by construction, our Neural Network needs to be invertible. We choose Invertible ResNet[1] as our architecture.

[1] Invertible Residual Networks. Jens Behrmann, Will Grathwohl, Ricky T. Q. Chen, David Duvenaud, Joern-Henrik Jacobsen. arXiv:1811.00995

Normalizing flows

- We can use support space for performing integrals:

$$< \phi_i^A | \phi_j^A > = \int \phi_i^*(q)\phi_j(q)dq = \delta_{i,j}$$

$$< \phi_i^A | V(x) | \phi_j^A > = \int \phi_i^*(q)V(g_\theta^{-1}(q))\phi_j(q)dq$$

- Using many states variational principle, the eigenvalues of the Hamiltonian matrix ($\lambda_i(\theta)$) can only be bigger or equal to the real energies:

$$\sum_i^N \lambda_i(\theta) = \sum_i^N < \phi_i^A | H(\theta) | \phi_i^A > \geq \sum_i^N E_i$$

Normalizing flows

- We can use support space for performing integrals:

$$< \phi_i^A | \phi_j^A > = \int \phi_i^*(q)\phi_j(q)dq = \delta_{i,j}$$

$$< \phi_i^A | V(x) | \phi_j^A > = \int \phi_i^*(q)V(g_\theta^{-1}(q))\phi_j(q)dq$$

- Using many states variational principle, the eigenvalues of the Hamiltonian matrix ($\lambda_i(\theta)$) can only be bigger or equal to the real energies:

$$\sum_i^N \lambda_i(\theta) = \sum_i^N < \phi_i^A | H(\theta) | \phi_i^A > \geq \sum_i^N E_i$$

- This is the optimization problem we needed for the Neural Network:

Normalizing flows

- We can use support space for performing integrals:

$$< \phi_i^A | \phi_j^A > = \int \phi_i^*(q)\phi_j(q)dq = \delta_{i,j}$$

$$< \phi_i^A | V(x) | \phi_j^A > = \int \phi_i^*(q)V(g_\theta^{-1}(q))\phi_j(q)dq$$

- Using many states variational principle, the eigenvalues of the Hamiltonian matrix ($\lambda_i(\theta)$) can only be bigger or equal to the real energies:

$$\sum_i^N \lambda_i(\theta) = \sum_i^N < \phi_i^A | H(\theta) | \phi_i^A > \geq \sum_i^N E_i$$

- This is the optimization problem we needed for the Neural Network:

$$\mathscr{L}(\theta) = \sum_i^N < \phi_i^A | H(\theta) | \phi_i^A >$$

Normalizing flows

- We can use support space for performing integrals:

$$< \phi_i^A | \phi_j^A > = \int \phi_i^*(q)\phi_j(q)dq = \delta_{i,j}$$

$$< \phi_i^A | V(x) | \phi_j^A > = \int \phi_i^*(q)V(g_\theta^{-1}(q))\phi_j(q)dq$$

- Using many states variational principle, the eigenvalues of the Hamiltonian matrix ($\lambda_i(\theta)$) can only be bigger or equal to the real energies:

$$\sum_i^N \lambda_i(\theta) = \sum_i^N < \phi_i^A | H(\theta) | \phi_i^A > \geq \sum_i^N E_i$$

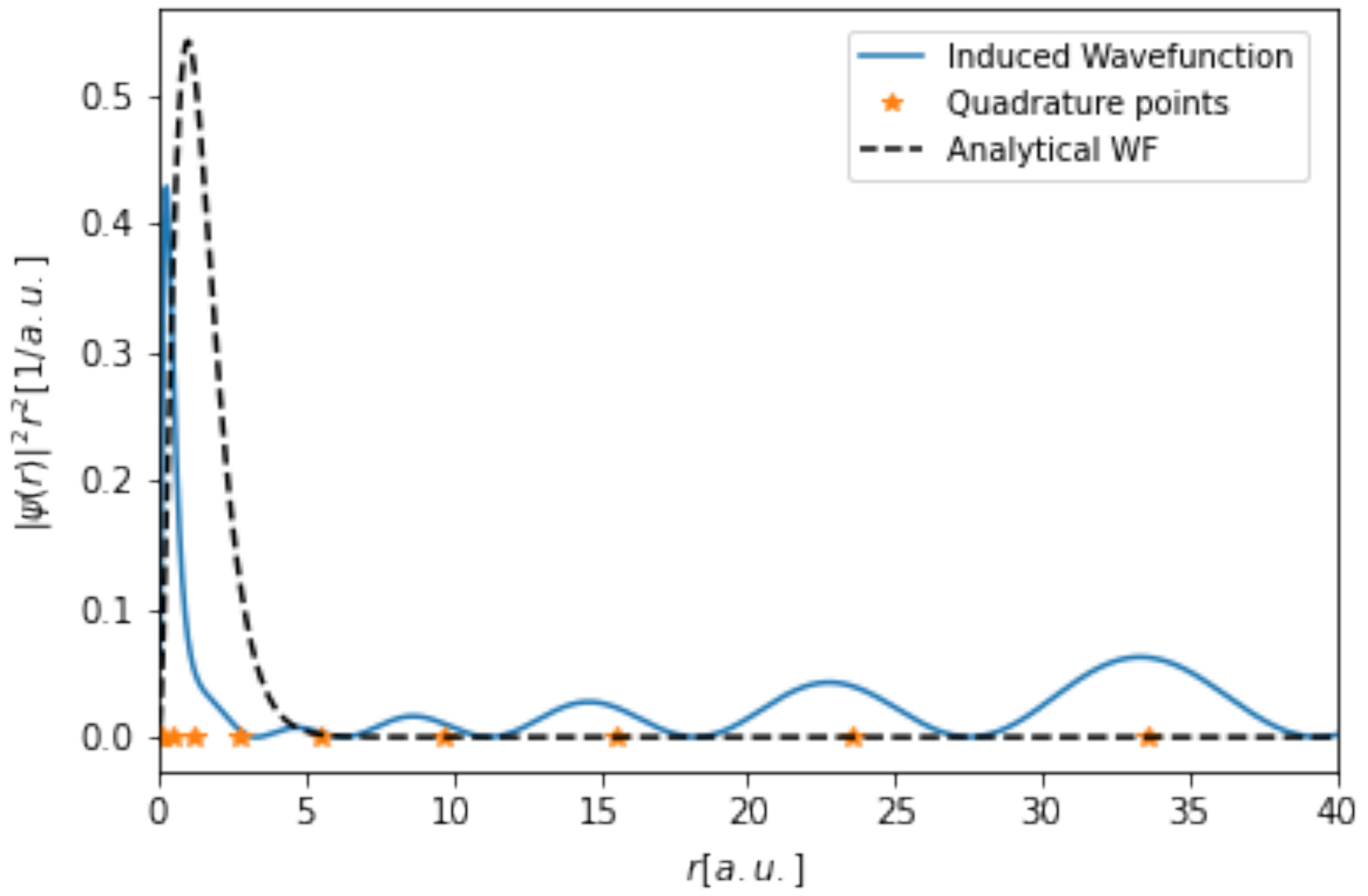
- This is the optimization problem we needed for the Neural Network:

$$\mathscr{L}(\theta) = \sum_i^N < \phi_i^A | H(\theta) | \phi_i^A >$$

- We can apply numerical approximations for integrals in support space. How?

Integrals using quadratures

- Very precise integrals. Memory usage scales approximately as N^{dim}, so very useful for low dimensional problems.
- Drawback: during training, wave function is only studied at quadrature points. Resulting eigenfunction is biased.



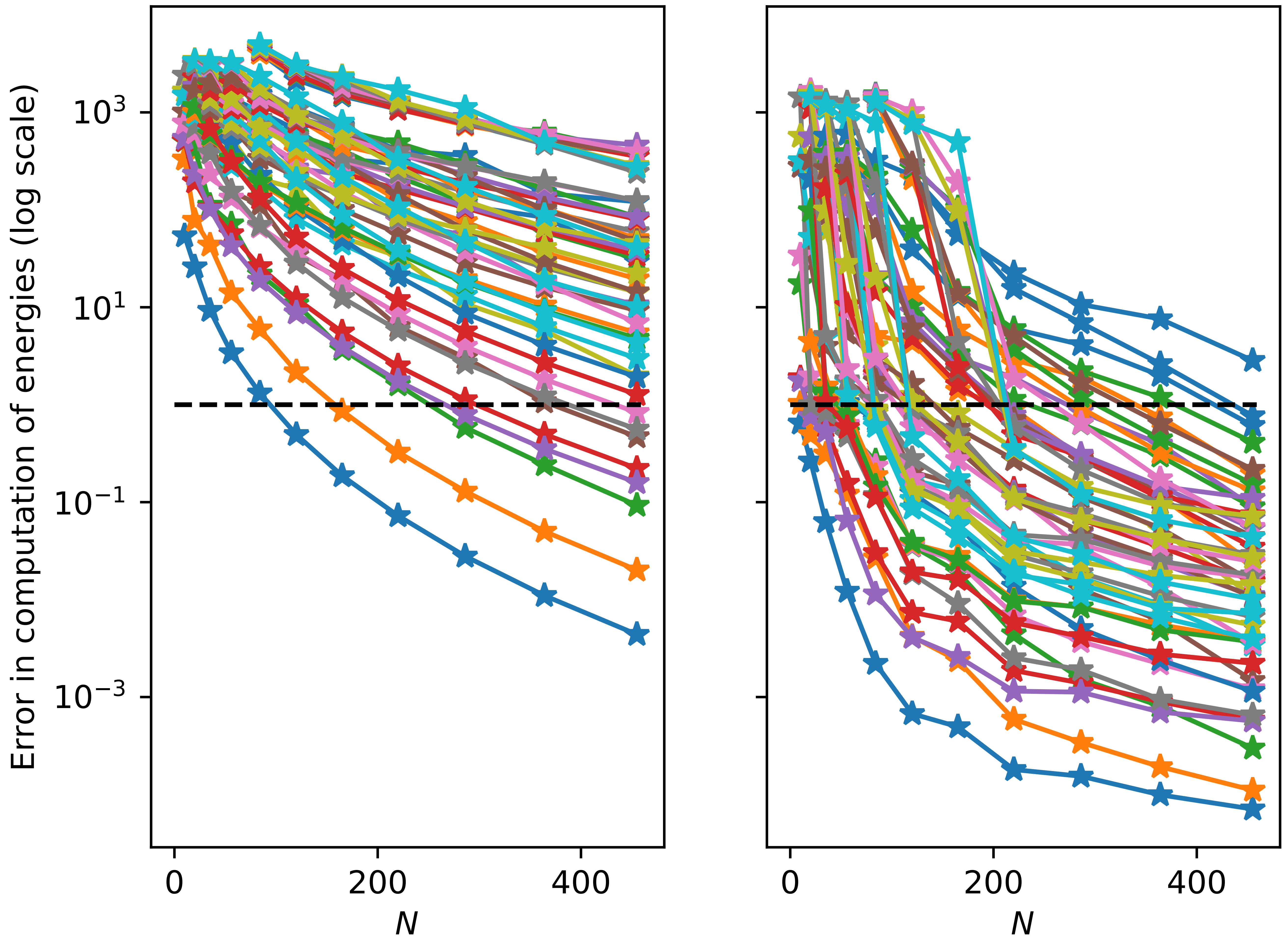
Integrals using Langevin MonteCarlo

- Starting from a point in space, create a Markov Chain that accepts points based on a density. Using $\rho_i(q) = |\phi_i^A(q)|^2$ as our density, we obtain a distribution of points $q_\alpha \sim |\phi_i^A(q)|^2$. Then any integral can be approximated by:

$$< \phi_i^A | f(q) | \phi_j^A > = \int \rho_i(q) \frac{f(q)\phi_j^A(q)}{\phi_i^A(q)} \simeq \frac{1}{M} \sum_{q_\alpha \sim |\phi_i^A(q)|^2}^{M} \frac{f(q_\alpha)\phi_j^A(q_\alpha)}{\phi_i^A(q_\alpha)}$$

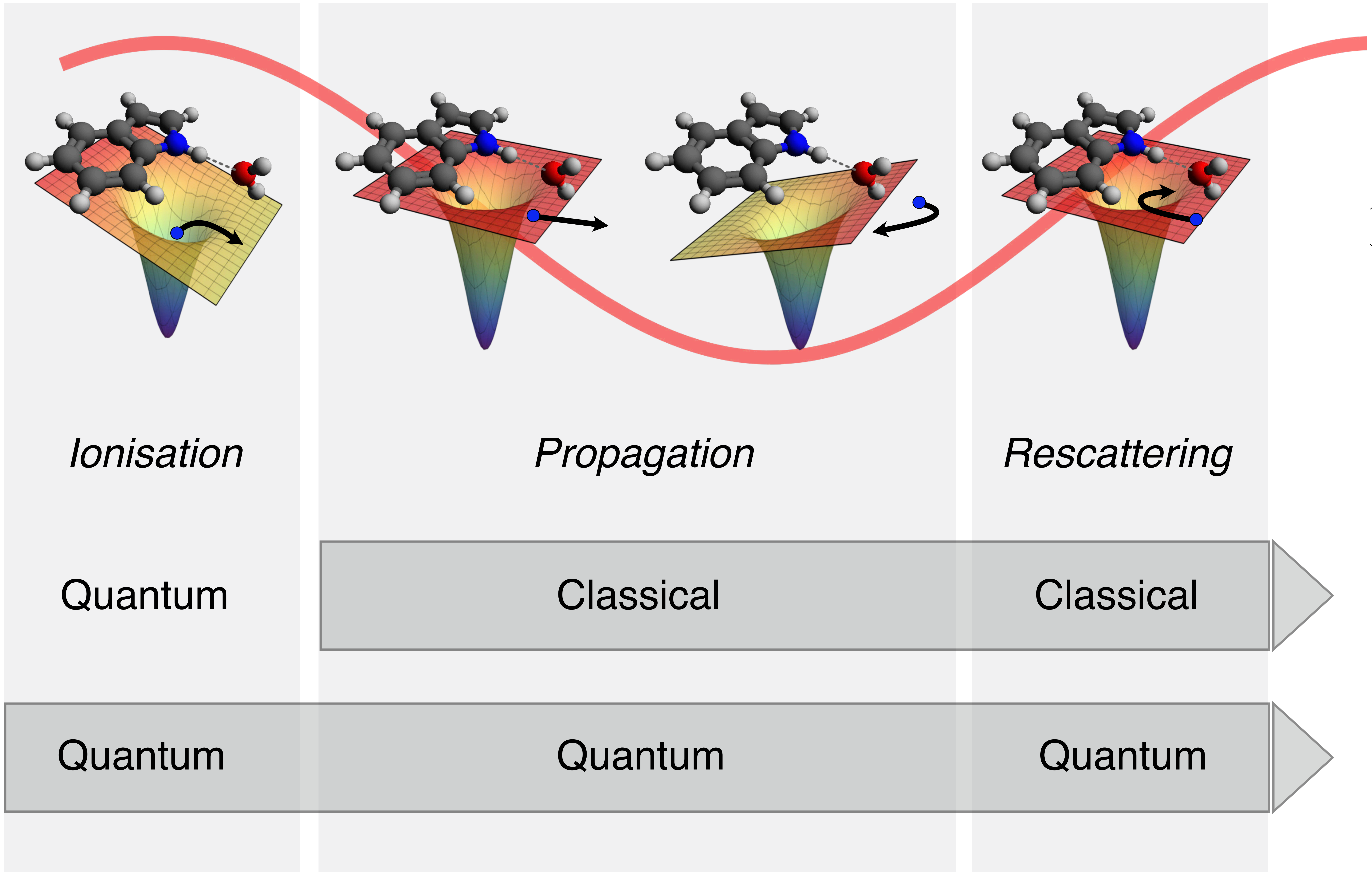
- The integral can cover all the space during training. To do so, choose M to be a big number and divide the training process in minibatches of m points.
- The memory usage scales now as $M \cdot dim$, of which only $m \cdot dim$ enter the training at the same time. Thus, it is more efficient for problems with many intrinsic degree of freedom.

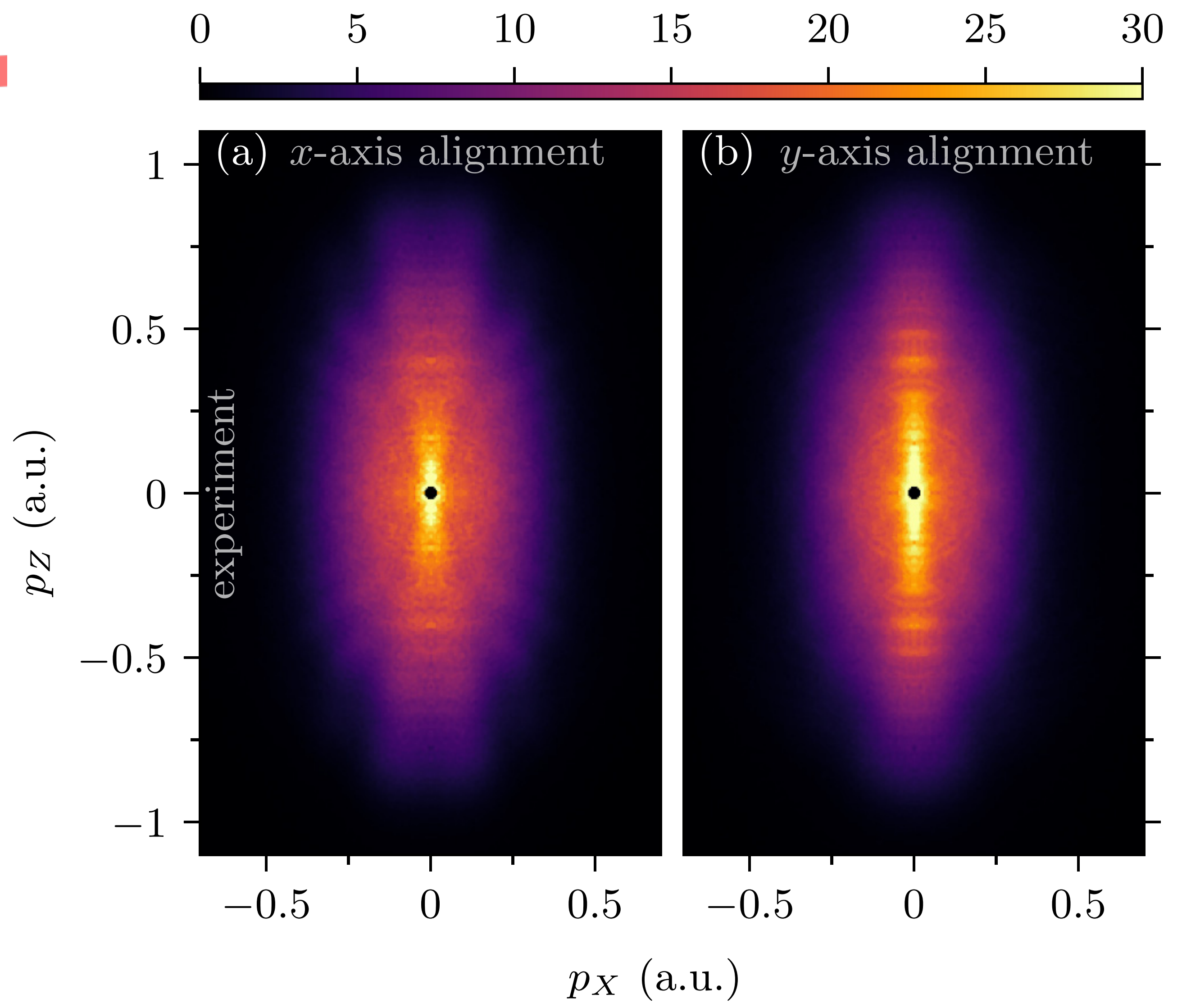
Results



- Outperform the convergence of classical basis expansion w.r.t. number of basis functions.
- Convergence for many excited states.
- Tested for vibrational and electronic states.

Ultrafast imaging of dissociation dynamics with LIED





solution of classical ODEs with neural networks,
PAD features ↔ trajectory type

solution of quantum ODEs with neural networks (LSTM, quantum flows, …)

Time Dependent Schrödinger Equation

- Partial Derivatives equation:

$$H(x_1, x_2, \ldots; t)\Psi(x_1, x_2, \ldots; t) = = i\partial_t \Psi(x_1, x_2, \ldots; t)$$

With boundary condition: $\Psi(x_1, x_2, \ldots; t = t_0) = \Psi_0(x_1, x_2, \ldots)$

- In the basis expansion approximation, initial condition is given by:

$$\Psi_0(x) = \sum_i^N c_i(t_0)\phi_i(x)$$

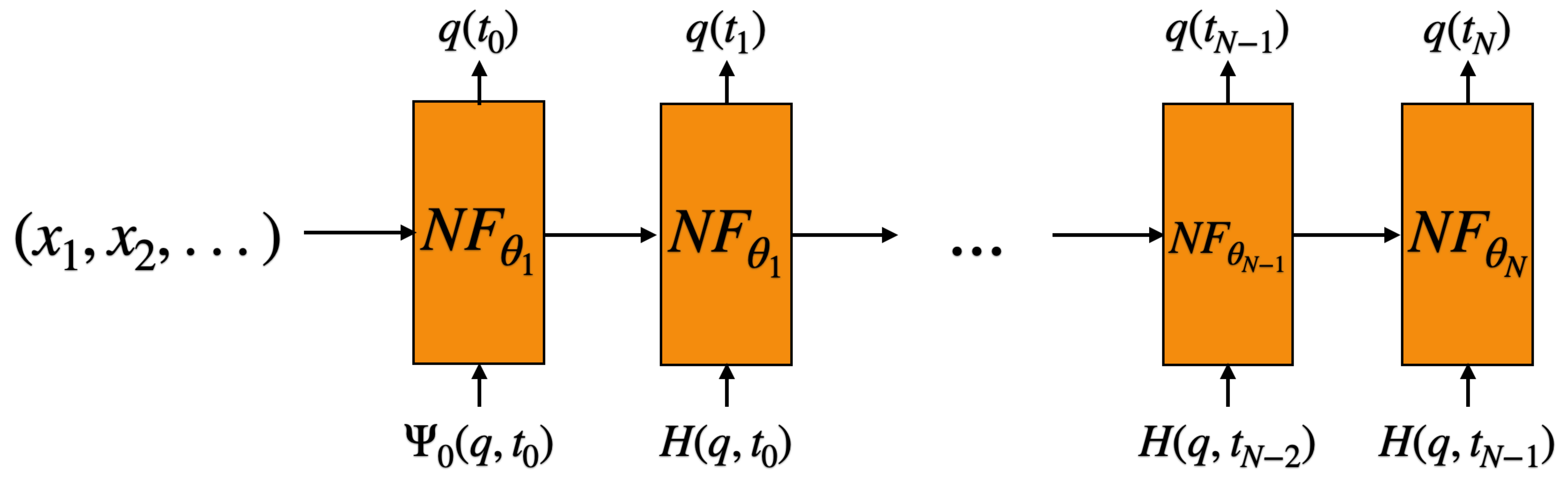
- Then, our approximation to the wave function is given by propagating the coefficients in small time steps at each time:

$$c_i(t) = \sum_j^N e^{<\phi_i(x)|H(x;t-\Delta t)|\phi_j(x)>} c_j(t - \Delta t)$$

- The approximation only holds for small time steps and a big number of basis functions.

Time Dependent Schrödinger Equation

- Can we use a similar strategy to solve a time dependent equation?
  - Hopefully! Using recurrent Neural Networks each for one time step:



  - We again create an augmented basis at each time:

$$\phi_i^A(q,t) = \phi_i(q)\sqrt{|\det \frac{\partial q}{\partial g_{\theta,t}^{-1}(q)}|}$$

  - And our approximation to the wave function:

$$\Psi^A(q,t) = \sum_i^N c_i(t)\phi_i^A(q,t)$$

Acknowledgments
CFEL Controlled Molecule Imaging Group

Jochen Küpper

Sebastian Trippel	**Amit Samanta**	**Andrey Yachmenev**
Hubertus Bromberger	Muhamed Amin	Álvaro Fernández
Wuwei Jin	Xuemei Cheng	Yahya Saleh
Dimitrios Koulentianos	Armando Estillore	Guang Yang
Matthew Robinson	Lukas Haas	
Mukhtar Singh	Jingxuan He	
Nidin Vadassery	Jannik Lübke	
	Surya Kiran Peravali	Karol Długołęcki
	Lena Worbs	Barbora Vagovič

CUI THE HAMBURG CENTER FOR ULTRAFAST IMAGING

Universität Hamburg DER FORSCHUNG | DER LEHRE | DER BILDUNG

DFG

EXC Center for Ultrafast Imaging

EXC Advanced Imaging of Matter

SPP 1840 Quantum Dynamics in Tailored Light Fields

MC-ITN MEDEA

MC-ITN ASPIRE

Alexander von Humboldt Stiftung/Foundation

Bundesministerium für Bildung und Forschung

HELMHOLTZ ASSOCIATION

erc COMOTION European Research Council Established by the European Commission

DESY.