Variational quantum Monte Carlo with neural network ansatz for open quantum systems

Alexandra Nagy

Prof. Vincenzo Savona
Laboratory of Theoretical Physics of Nanosystems
EPFL

GPU Day
July 11, 2019

Open quantum systems

EPFL

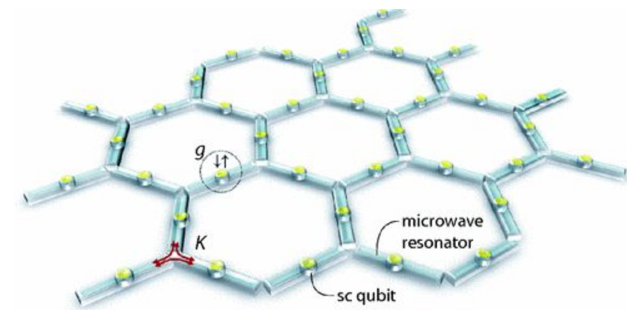
- Coupling to external environment
- Time evolution as Linbladian master equation

$$\dot{\hat{\rho}} = \underbrace{-\frac{i}{\hbar}\left[\hat{\mathcal{H}}, \hat{\rho}\right]}_{\text{unitary dynamics}} \underbrace{-\sum_j \frac{\gamma_j}{2}\left[\left\{\hat{K}_j^\dagger \hat{K}_j, \hat{\rho}\right\} - 2\hat{K}_j \hat{\rho} \hat{K}_j^\dagger\right]}_{\substack{\text{dissipative processes} \\ \hat{D} = \{\hat{K}_j\}}}$$

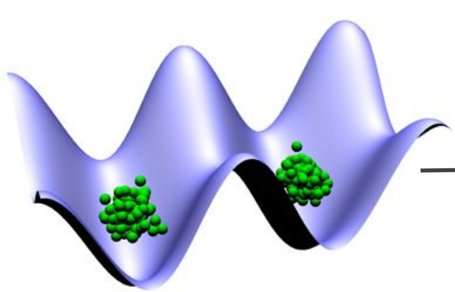
- Expressed as

$$\dot{\hat{\rho}} = \mathcal{L}(\hat{\rho})$$

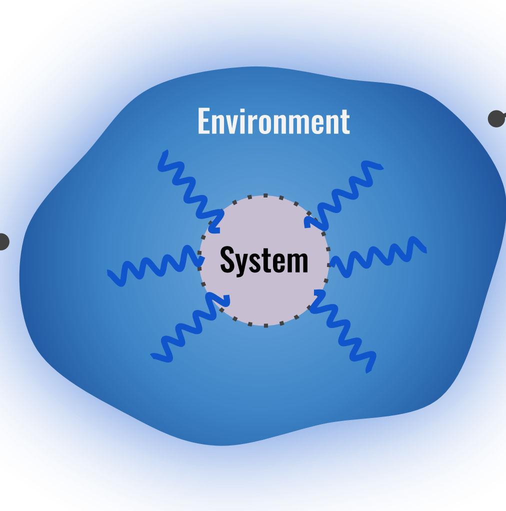
- Computational cost scales exponentially

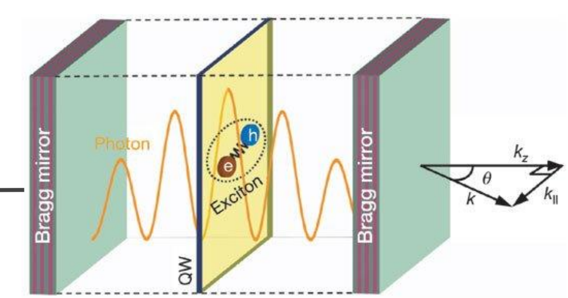


Koch et al., PRA (2010)



A. J. Daley, APS Viewpoint (2015)





Kasprzak et al., Nature (2006)

Lindbladian dynamics

- Formal solution

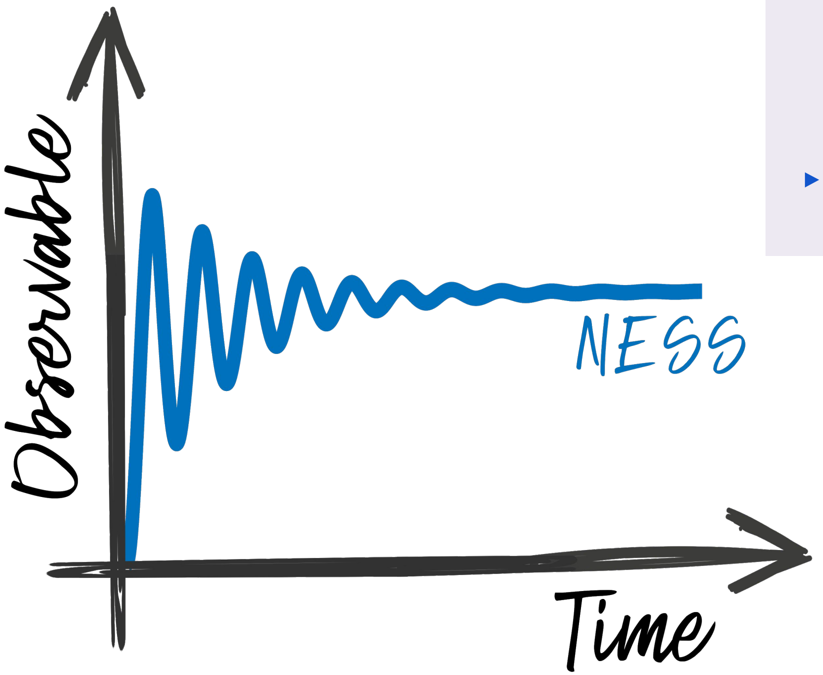
$$\hat{\rho}(t) = e^{\mathcal{L}t}\hat{\rho}(0)$$

- Long-time limit: **non-equilibrium steady state (NESS)**

$$\mathcal{L}(\hat{\rho}_{ss}) = \langle \rho_{ss}|\mathcal{L}|\rho_{ss}\rangle = 0$$

$$\mathcal{L}^\dagger\mathcal{L}(\hat{\rho}_{ss}) = \langle \rho_{ss}|\mathcal{L}^\dagger\mathcal{L}|\rho_{ss}\rangle = 0$$

- $\Longrightarrow$ Variational principle



What do we need?

- Variational ansatz for the density matrix
- Optimization method
- Stochastic sampling

What do we need?

- Variational ansatz for the density matrix
- Optimization method
- Stochastic sampling

Variational ansatz for the density matrix

- $\mathcal{H} = \{|\boldsymbol{\sigma}\rangle\} = \{|\sigma_1, \sigma_2, \ldots, \sigma_N\rangle\}$
- **Ansatz**: map $\rho_\chi(\boldsymbol{\sigma}, \boldsymbol{\eta})$
- Self-adjoint, positive semi-definite form

$$\rho_\chi(\boldsymbol{\sigma}, \boldsymbol{\eta}) = \sum_{j=1}^{J} p_j(\chi) \cdot \psi_j(\boldsymbol{\sigma}, \chi)\psi_j^*(\boldsymbol{\eta}, \chi)$$

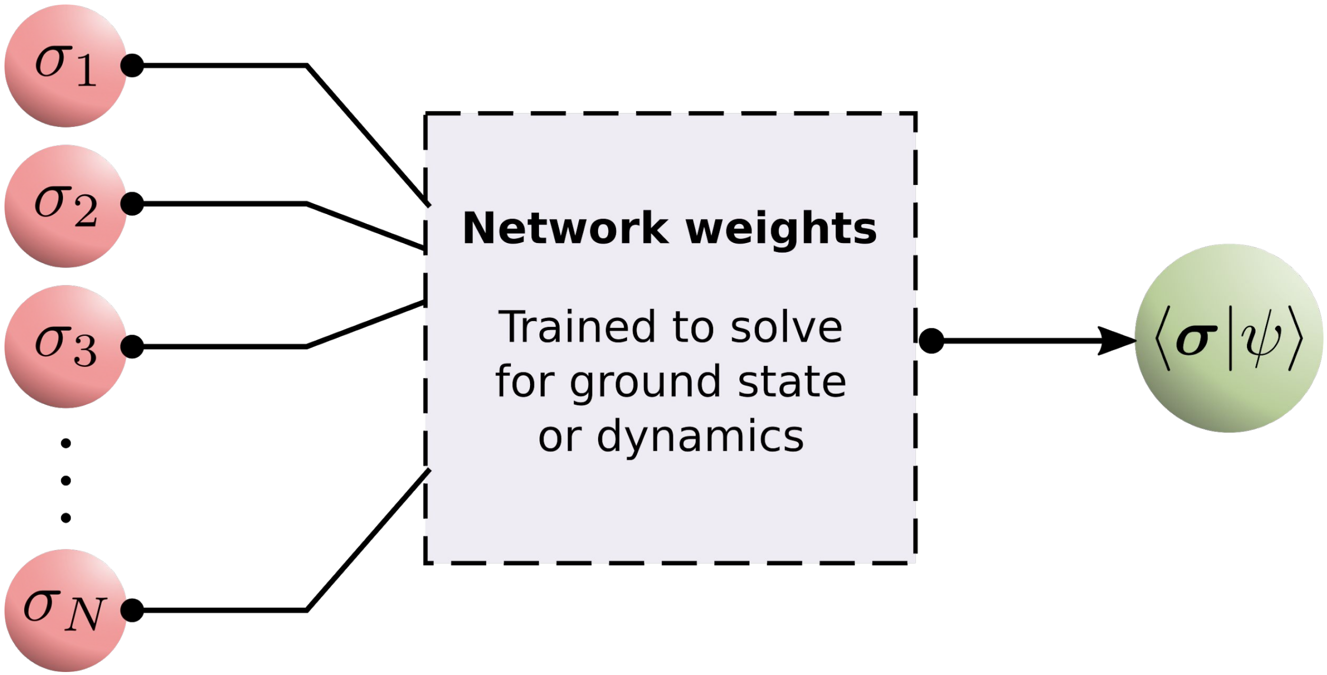
- What do we use for $\psi_j(\boldsymbol{\sigma}, \chi)$?
- Tensor networks? Jastrow wave-function?
- Neural network representation by Carleo and Troyer [Science 355, 602]

$$\begin{pmatrix} \rho_{1,1} & \rho_{1,2} & \cdots & \rho_{1,n} \\ \rho_{2,1} & \rho_{2,2} & \cdots & \rho_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{n,1} & \rho_{n,2} & \cdots & \rho_{n,n} \end{pmatrix}$$

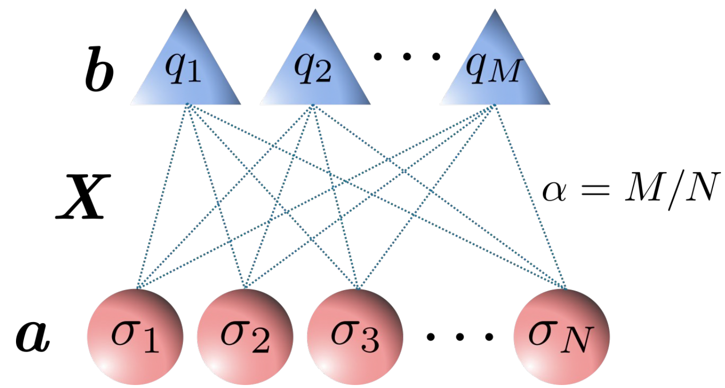
$$\downarrow$$

$$\rho_\chi(\boldsymbol{\sigma}, \boldsymbol{\eta})$$

Neural Network Quantum States



Restricted Boltzmann-machine



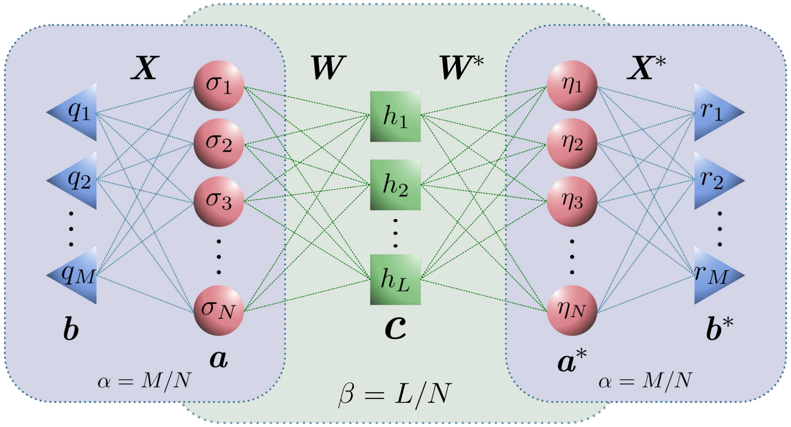
$$\psi(\boldsymbol{\sigma}, \chi) = \sum_{\{q\}} e^{\mathcal{H}(\boldsymbol{\sigma}, \{q\})} = \sum_{\{q\}} e^{\left(\sum_i a_i \sigma_i + \sum_m b_m q_m + \sum_{m,i} q_m \sigma_i X_{mi}\right)} = P(\boldsymbol{\sigma})$$

- The network is connected by Ising interactions
- The probability of a spin configuration is the Boltzmann-weight of the Ising Hamiltonian
- Connection to tensor networks
  [Cirac et al., PRX 8, 011006]

- Intrinsically non-local correlations
- No dimensionality constraints
- High accuracy (representability theorems)

Neural Network Density Matrix

$$\rho_\chi(\boldsymbol{\sigma},\boldsymbol{\eta}) = \sum_{j=1}^{J} p_j(\chi)\cdot\psi_j(\boldsymbol{\sigma},\chi)\psi_j^*(\boldsymbol{\eta},\chi)$$



$$\rho_\chi(\boldsymbol{\sigma},\boldsymbol{\eta}) = e^{\left(\sum_i a_i\sigma_i\right)} e^{\left(\sum_i a_i^*\eta_i\right)} \times \prod_{l=1}^{L}\cosh\left(c_l+\sum_i W_{li}\sigma_i+\sum_i W_{li}^*\eta_i\right)$$
$$\times\prod_{m=1}^{M}\cosh\left(b_m+\sum_i X_{mi}\sigma_i\right)\times\prod_{n=1}^{M}\cosh\left(b_n^*+\sum_i X_{ni}^*\eta_i\right)$$

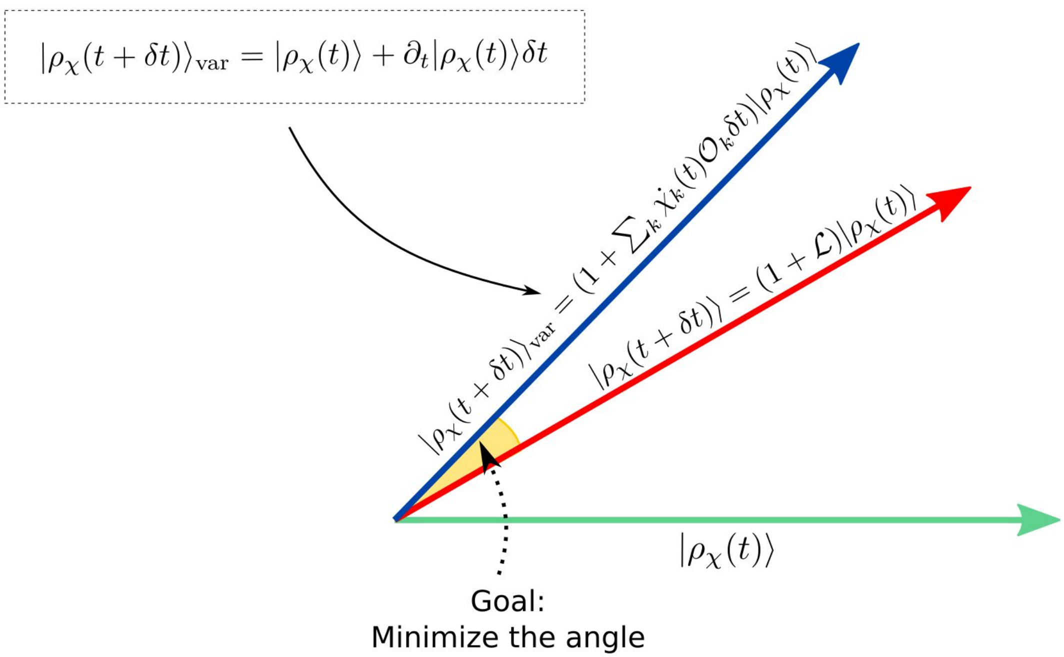
- The representative power depends on the number of hidden nodes and ancillary bits
- Numerically efficient, analytical derivatives

What do we need?

- Variational ansatz for the density matrix
- Optimization method
- Stochastic sampling

Stochastic Reconfiguration (SR) for open quantum systems

- Well adapted for Hamiltonian problems [Sorella et al., J. Chem. Phys. 127, 014105]
- Efficient and robust
- Accounts for the correlation between the variables



Stochastic Reconfiguration (SR) for open quantum systems

$$\partial_t \chi_k = \sum_{k'} S^{-1}_{kk'} F_{k'}$$

Linear system to be solved at each iteration step

Sparse solvers for large number of parameters [Choi et al., SIAM, **33**, 1810]

- Real time evolution

$$F_k = -\frac{\partial \langle\langle \mathcal{L} \rangle\rangle}{\partial \chi_k} = -\frac{\partial}{\partial \chi_k} \frac{\langle \rho_\chi | \mathcal{L} | \rho_\chi \rangle}{\langle \rho_\chi | \rho_\chi \rangle}$$

- Steady state

$$F_k = -\frac{\partial \langle\langle \mathcal{L}^\dagger \mathcal{L} \rangle\rangle}{\partial \chi_k} = -\frac{\partial}{\partial \chi_k} \frac{\langle \rho_\chi | \mathcal{L}^\dagger \mathcal{L} | \rho_\chi \rangle}{\langle \rho_\chi | \rho_\chi \rangle}$$

- SR = Steepest descent if S is the identity
- Euler approximation: $\chi(t+\delta t) = \chi(t) + \nu \cdot S^{-1}(t) F(t)$

$$O_k(\boldsymbol{\sigma}, \boldsymbol{\eta}) = \frac{1}{\rho_\chi(\boldsymbol{\sigma}, \boldsymbol{\eta})} \cdot \frac{\partial \rho_\chi(\boldsymbol{\sigma}, \boldsymbol{\eta})}{\partial \chi_k}$$

$$F_k = \langle\langle O_k^* \mathcal{L} \rangle\rangle - \langle\langle \mathcal{L} \rangle\rangle \langle\langle O_k^* \rangle\rangle$$

$$S_{kk'} = \langle\langle O_k^* O_{k'} \rangle\rangle - \langle\langle O_k^* \rangle\rangle \langle\langle O_{k'} \rangle\rangle$$

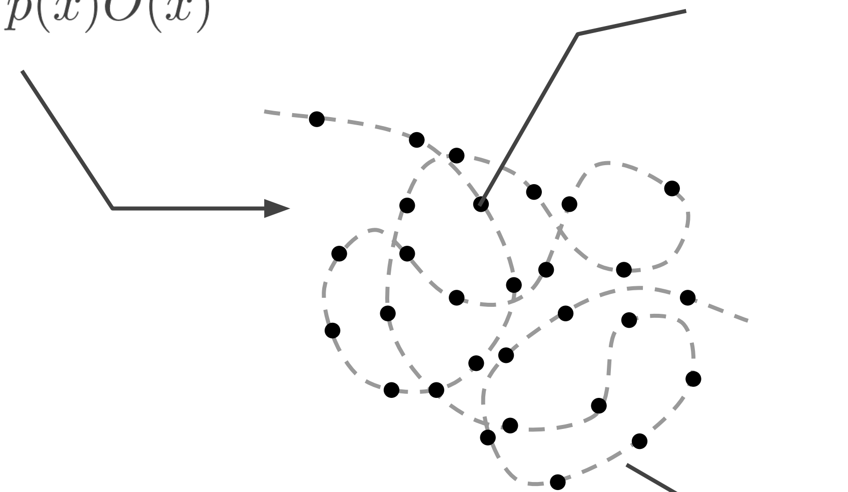
What do we need?

- Variational ansatz for the density matrix
- Optimization method
- Stochastic sampling

Stochastic sampling

$$\langle O \rangle = \sum_x p(x) O(x)$$

$$\mathcal{A}(X_i \to X_j) = \min\left(1, \frac{p(X_j) \cdot \mathcal{Q}(j|i)}{p(X_i) \cdot \mathcal{Q}(i|j)}\right)$$

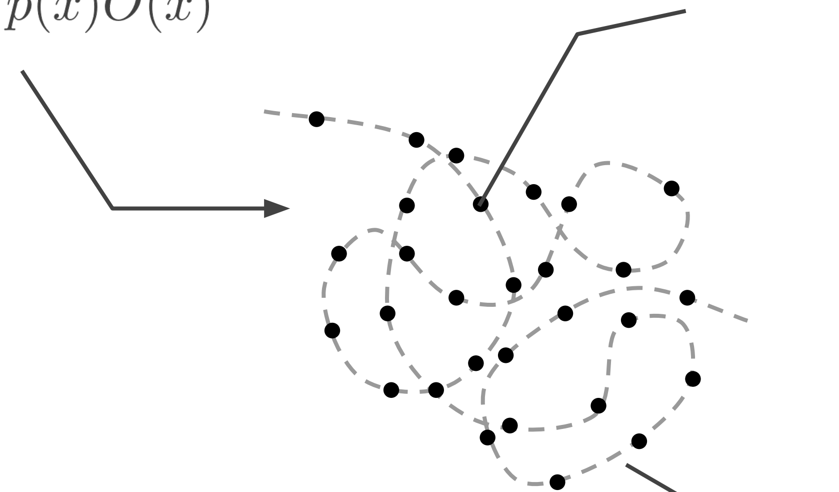


$$\langle O_{MC} \rangle = \frac{1}{N} \sum_i^N O(X_i)$$

Stochastic sampling

$$\langle O \rangle = \sum_x p(x) O(x)$$

$$\mathcal{A}(X_i \to X_j) = \min\left(1, \frac{p(X_j)\cdot \mathcal{Q}(j|i)}{p(X_i)\cdot \mathcal{Q}(i|j)}\right)$$



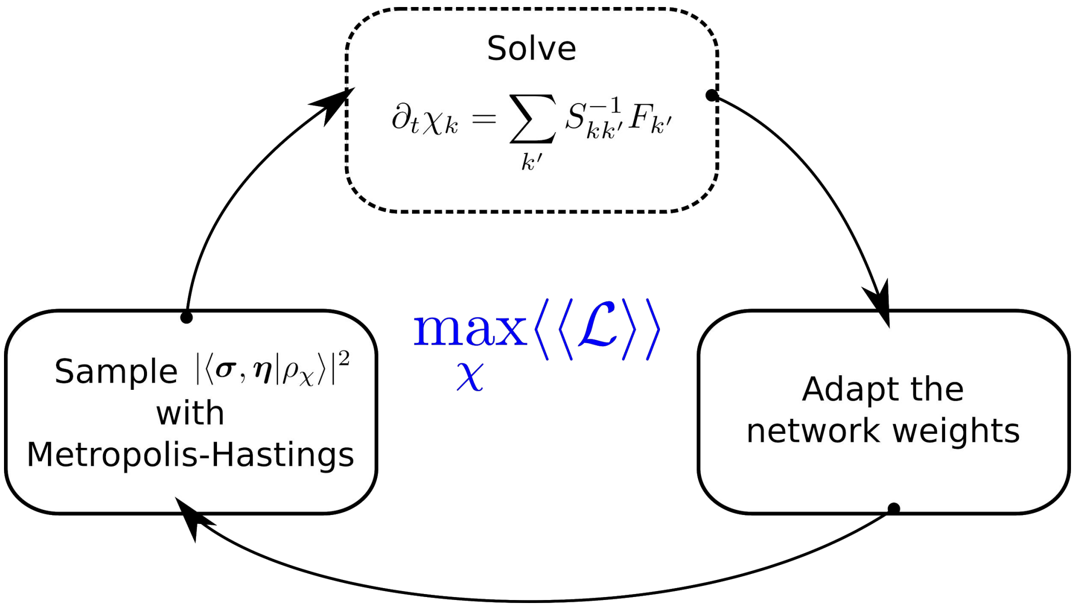
$$\langle O_{MC} \rangle = \frac{1}{N}\sum_i^N O(X_i)$$

- Let's rewrite the linear system for MCMC sampling

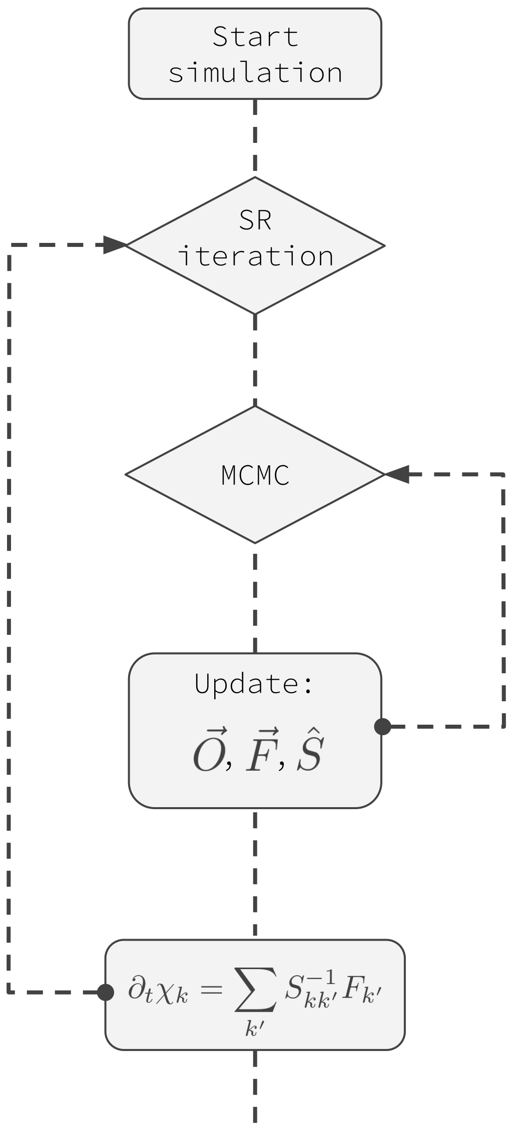
$$F_k = \sum_{\boldsymbol{\sigma},\boldsymbol{\eta}} |\langle \boldsymbol{\sigma},\boldsymbol{\eta}|\rho_\chi\rangle|^2 \cdot \left(\frac{\partial \ln\langle \boldsymbol{\sigma},\boldsymbol{\eta}|\rho_\chi\rangle}{\partial \chi_k}\right)^* \frac{\langle \boldsymbol{\sigma},\boldsymbol{\eta}|\mathcal{L}|\rho_\chi\rangle}{\langle \boldsymbol{\sigma},\boldsymbol{\eta}|\rho_\chi\rangle}$$
$$- \sum_{\boldsymbol{\sigma},\boldsymbol{\eta}} |\langle \boldsymbol{\sigma},\boldsymbol{\eta}|\rho_\chi\rangle|^2 \cdot \frac{\langle \boldsymbol{\sigma},\boldsymbol{\eta}|\mathcal{L}|\rho_\chi\rangle}{\langle \boldsymbol{\sigma},\boldsymbol{\eta}|\rho_\chi\rangle} \sum_{\boldsymbol{\sigma}',\boldsymbol{\eta}'} |\langle \boldsymbol{\sigma}',\boldsymbol{\eta}'|\rho_\chi\rangle|^2 \cdot \left(\frac{\partial \ln\langle \boldsymbol{\sigma}',\boldsymbol{\eta}'|\rho_\chi\rangle}{\partial \chi_k}\right)^*$$

What do we need?

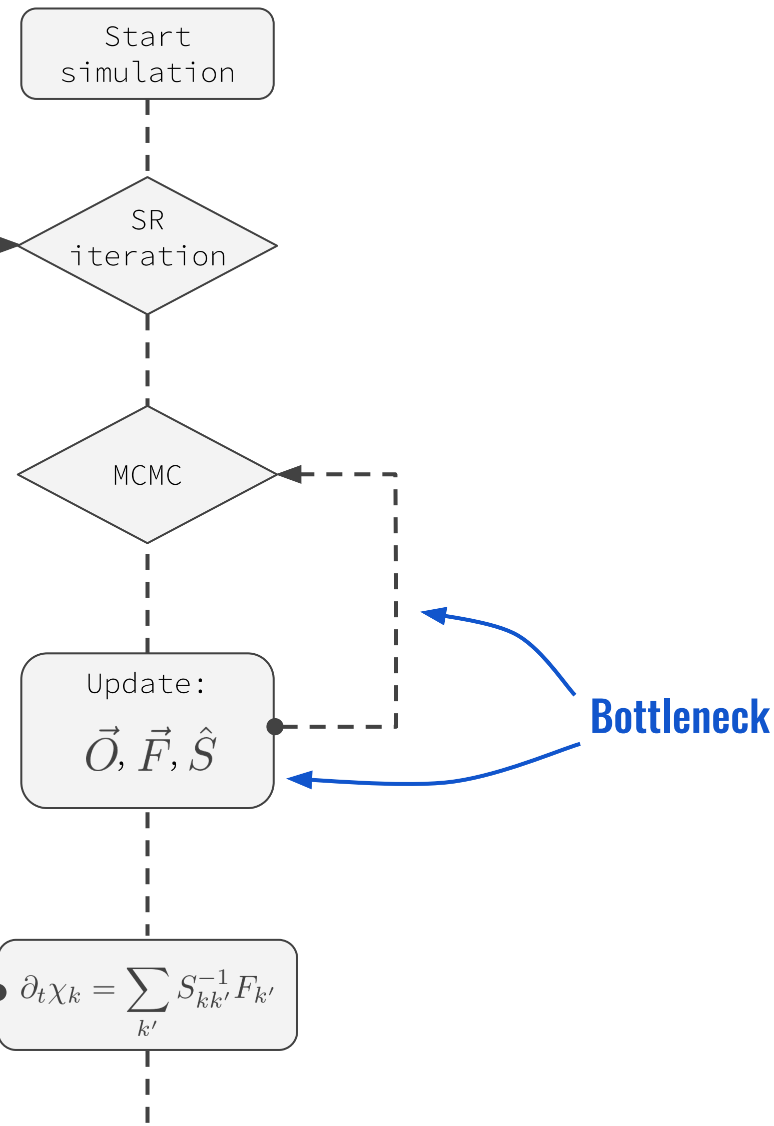
- Variational ansatz for the density matrix
- Optimization method
- Stochastic sampling



Implementation



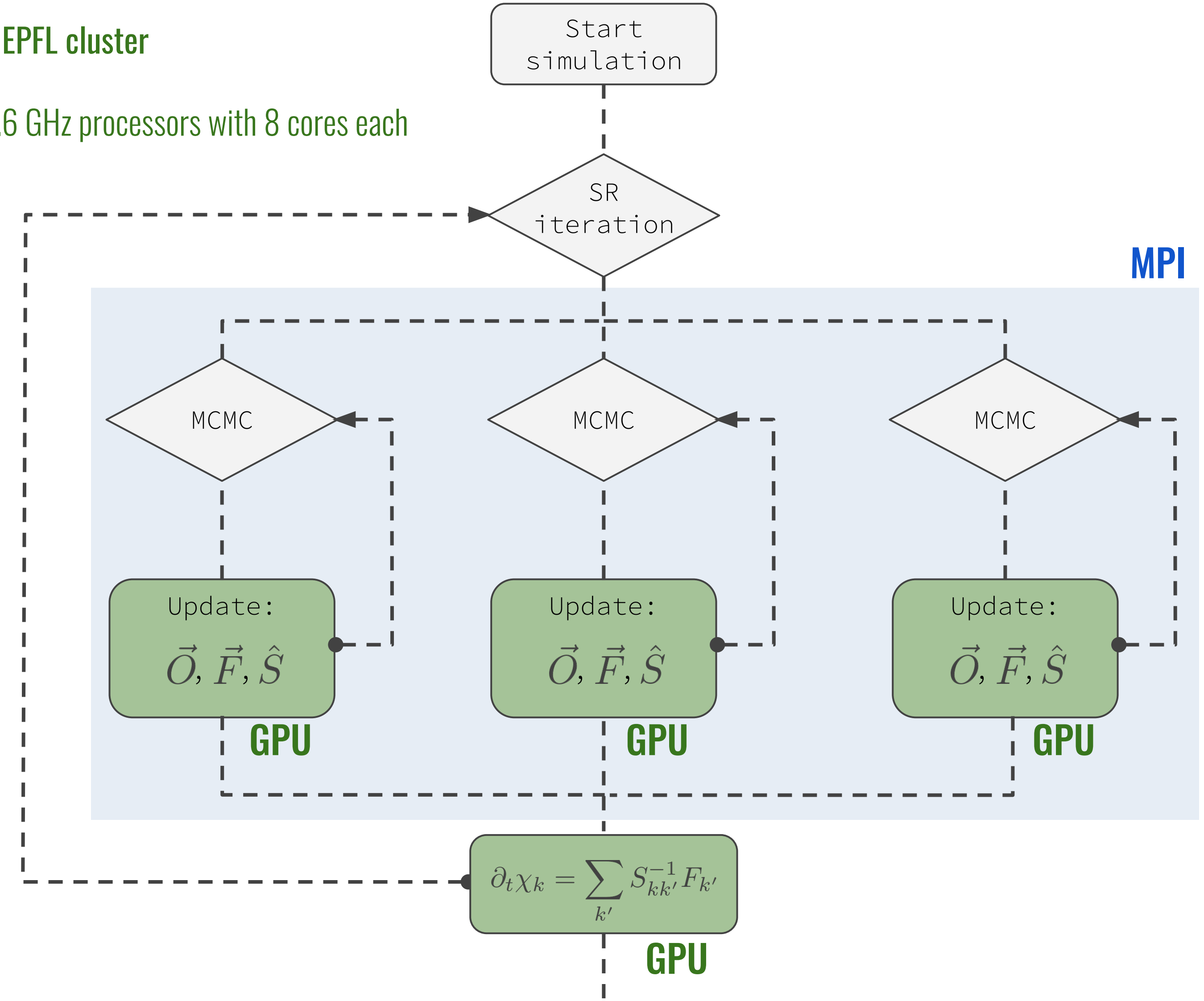
Implementation



Implementation

GPU SPEED UP: 20X

Python3.6 on EPFL cluster
Nodes:
2 Ivy Bridge 2.6 GHz processors with 8 cores each
4 K40 NVIDIA



Results: Spin lattices

XYZ Heisenberg model - 2D

$$\hat{\mathcal{H}} = \sum_{\langle i,j \rangle} \left(J_x \hat{\sigma}_i^x \hat{\sigma}_j^x + J_y \hat{\sigma}_i^y \hat{\sigma}_j^y + J_z \hat{\sigma}_i^z \hat{\sigma}_j^z \right)$$

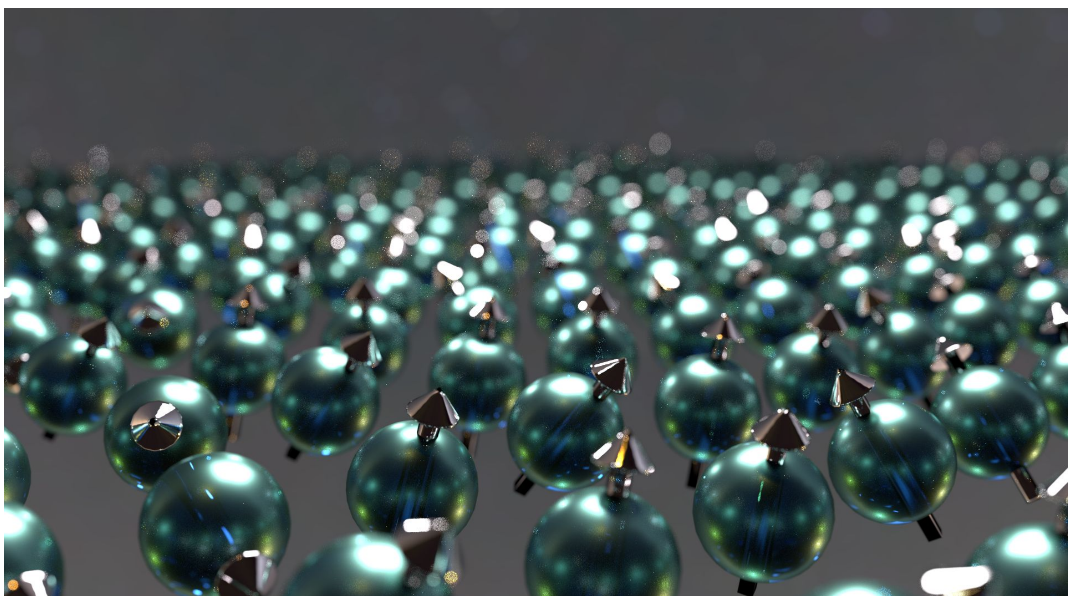
$$\hat{D} = \{\hat{\sigma}_i^-\}$$

Rydberg Ising model - 1D

$$\hat{\mathcal{H}} = \frac{U}{2} \sum_{\langle i,j \rangle} \hat{n}_i \hat{n}_j + \Omega \sum_i \hat{\sigma}_i^x$$

$$\hat{n}_i = \frac{1}{2}(1 + \hat{\sigma}_i^z)$$

$$\hat{D} = \{\hat{n}_i\}$$



Transverse field Ising model - 1D

$$\hat{\mathcal{H}} = J_z \sum_{\langle i,j \rangle} \hat{\sigma}_i^z \hat{\sigma}_j^z + h \sum_i \hat{\sigma}_i^x$$

$$\hat{D} = \{\hat{\sigma}_i^-\}$$

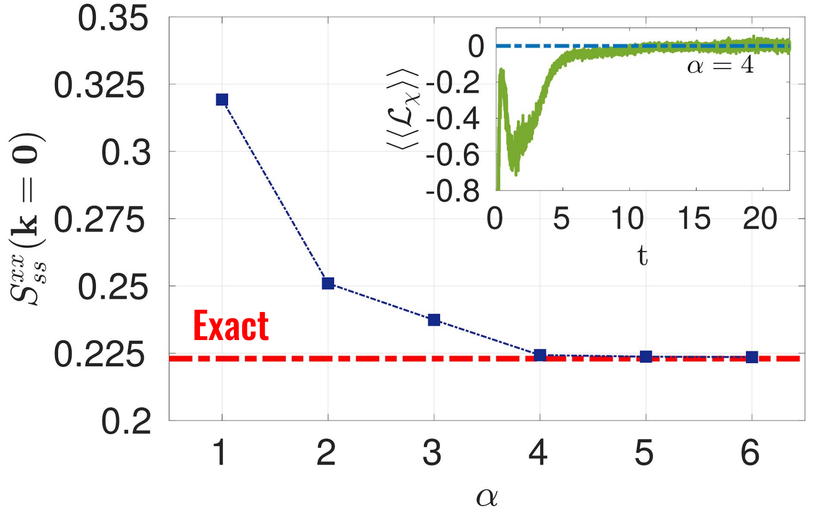
XYZ Heisenberg model - 2D - steady state

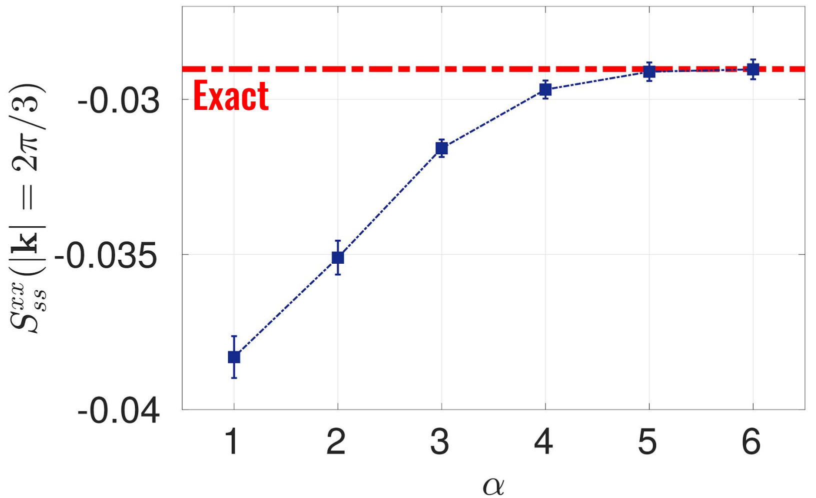
$$\hat{\mathcal{H}} = \sum_{\langle i,j \rangle} \left(J_x \hat{\sigma}_i^x \hat{\sigma}_j^x + J_y \hat{\sigma}_i^y \hat{\sigma}_j^y + J_z \hat{\sigma}_i^z \hat{\sigma}_j^z \right)$$

$$\hat{D} = \{ \hat{\sigma}_i^- \}$$

- Steady state spin structure factor

$$S_{ss}^{xx}(\mathbf{k}) = \frac{1}{N(N-1)} \sum_{\mathbf{j} \neq \mathbf{l}} e^{-i\mathbf{k}(\mathbf{j}-\mathbf{l})} \langle \hat{\sigma}_{\mathbf{j}}^x \hat{\sigma}_{\mathbf{l}}^x \rangle$$





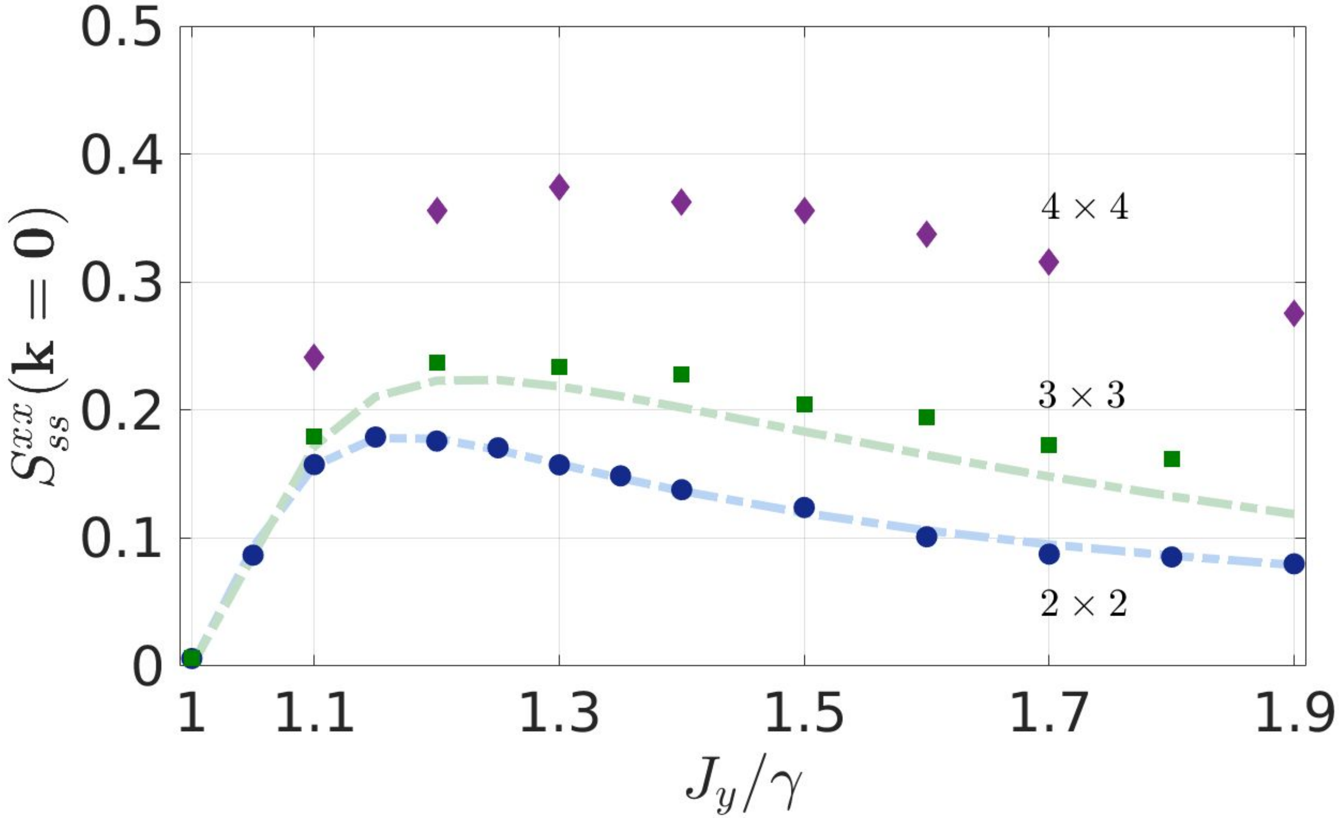
3×3 lattice

$J_x/\gamma = 0.9,\ J_y/\gamma = 1.2,\ J_z/\gamma = 1.0$

XYZ Heisenberg model - phase transition

$$\hat{\mathcal{H}} = \sum_{\langle i,j \rangle} \left(J_x \hat{\sigma}_i^x \hat{\sigma}_j^x + J_y \hat{\sigma}_i^y \hat{\sigma}_j^y + J_z \hat{\sigma}_i^z \hat{\sigma}_j^z \right)$$

$$\hat{D} = \{\hat{\sigma}_i^-\}$$



$J_x/\gamma = 0.9,\ J_z/\gamma = 1.0$

$\alpha = \beta = 3$

Rydberg Ising dynamics - 1D

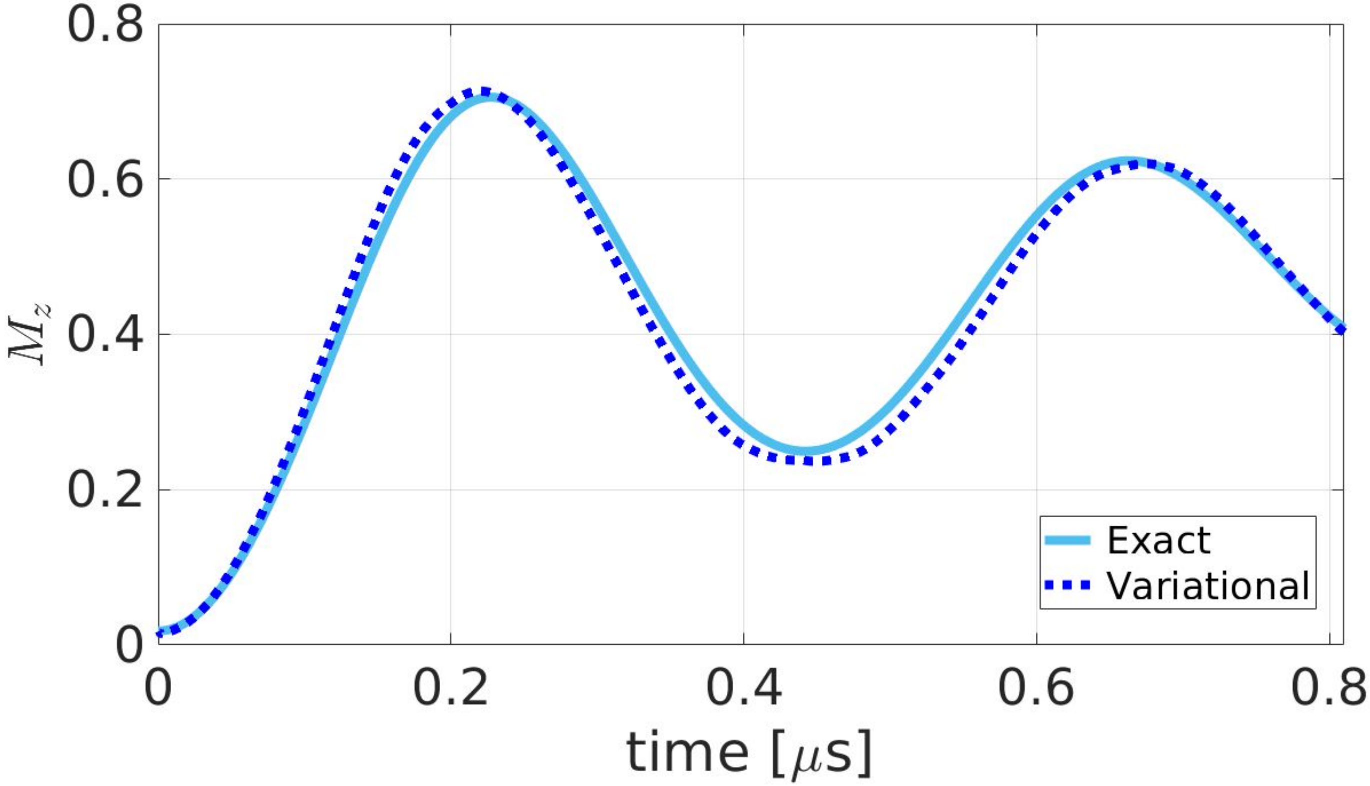
$$\hat{\mathcal{H}} = \frac{U}{2}\sum_{\langle i,j\rangle}\hat{n}_i\hat{n}_j + \Omega\sum_i\hat{\sigma}_i^x$$

$$\hat{n}_i = \frac{1}{2}(1+\hat{\sigma_i^z})$$

$$\hat{D} = \{\hat{n}_i\}$$

- Magnetization

$$M_z = \frac{1}{N}\sum_{i=1}^{N}\mathrm{Tr}(\hat{\rho}\hat{\sigma}_i^z)$$

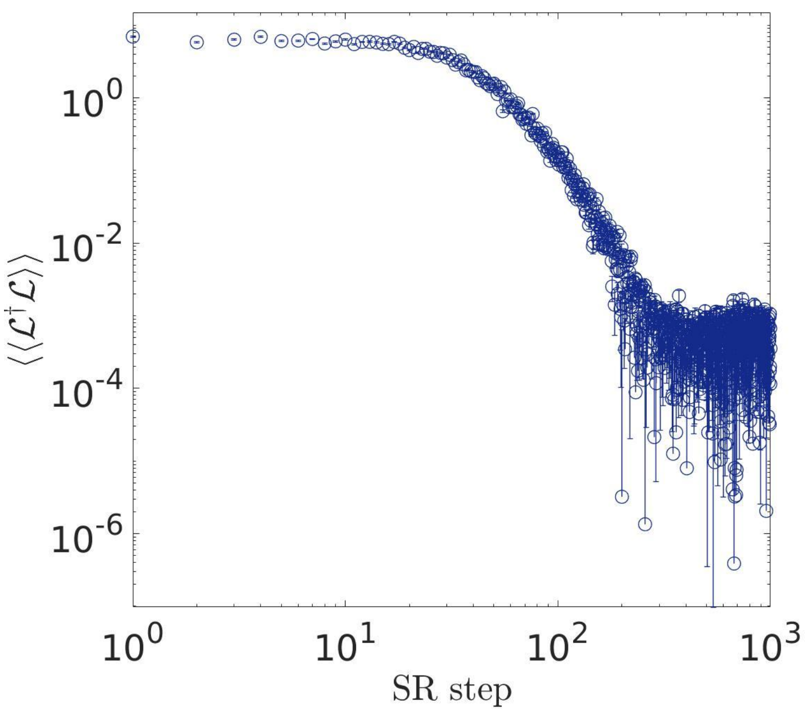


$$\alpha = \beta = 1, 6\text{ spins}, \frac{\Omega}{2\pi} = 0.9\text{ MHz}, \frac{U}{h} = 2.7\text{ MHz}$$

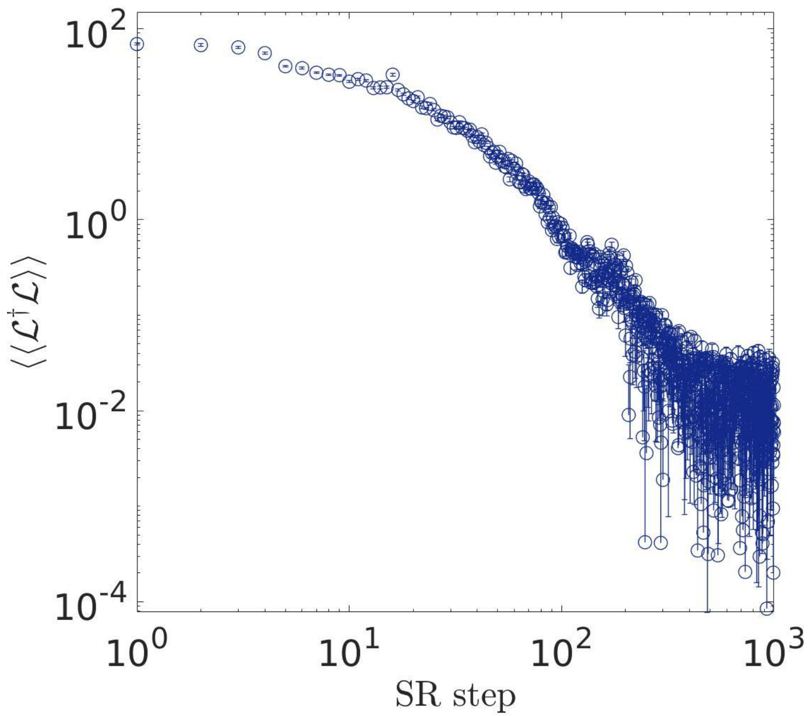
Transverse field Ising model - 1D

$$\hat{\mathcal{H}} = J_z \sum_{\langle i,j \rangle} \hat{\sigma}_i^z \hat{\sigma}_j^z + h \sum_i \hat{\sigma}_i^x$$

$$\hat{D} = \{\hat{\sigma}_i^-\}$$



5 spins



25 spins

$$\alpha = \beta = 1,\ J_z/\gamma = 0.5,\ h = 0.5$$

Conclusion

- Neural network ansatz for open quantum systems
- Efficient and robust optimization
- Adapted for many-core computing
- Results for different spin models
- Including phase transition and real time evolution
- Working version for bosons

Thank you for your attention!

https://doi.org/10.1103/PhysRevLett.122.250501

Featured

- PRL Viewpoint: https://physics.aps.org/articles/v12/74
- EPFL article: https://news.epfl.ch/news/simulating-quantum-systems-with-neural-networks/
- Engadget: https://www.engadget.com/2019/07/05/ai-simulates-quantum-systems/

Three flavours of machine learning

Supervised

- Labelled data

$$[(\mathbf{x_1}, \mathbf{y_1}), \ldots, (\mathbf{x_N}, \mathbf{y_N})]$$

- E.g. handwriting-recognition

Unsupervised

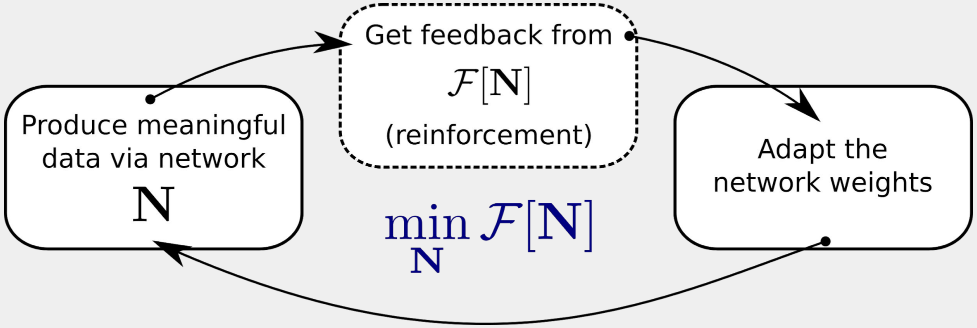
- Unlabelled data

$$[\mathbf{x_1}, \ldots, \mathbf{x_N}]$$

- E.g. cluster analysis

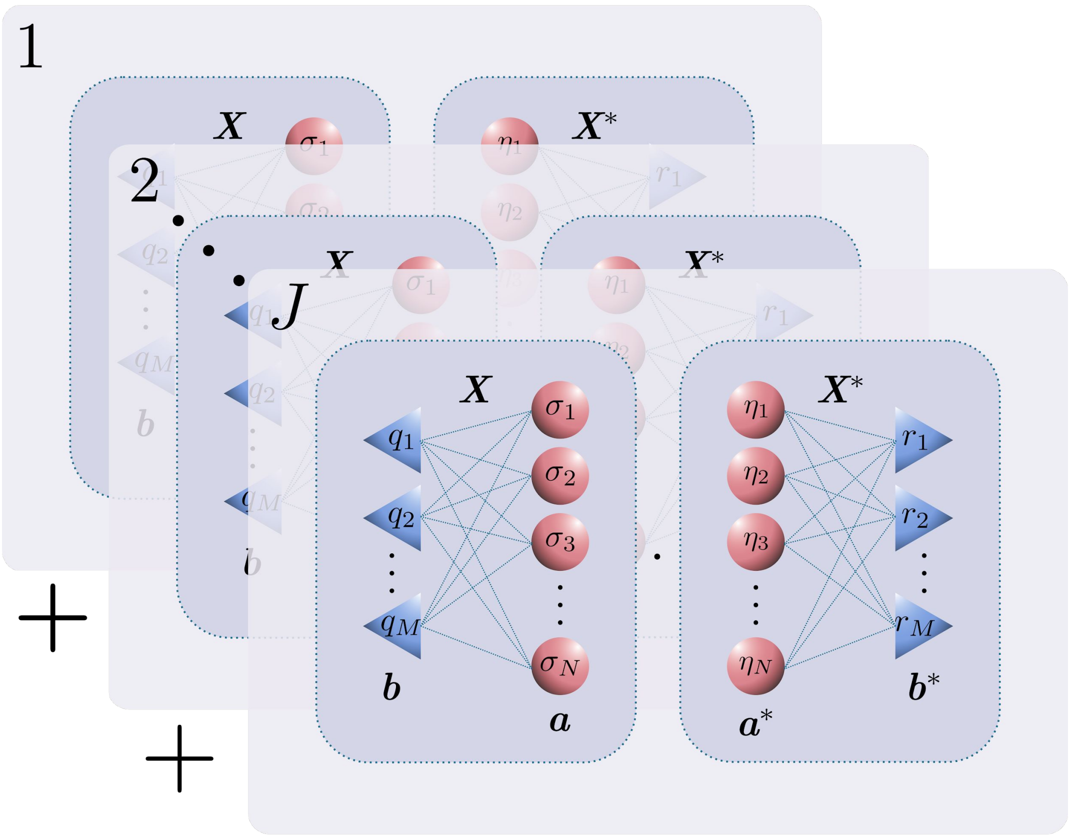
Reinforcement learning

- Generates data, gets feedback, solves the task



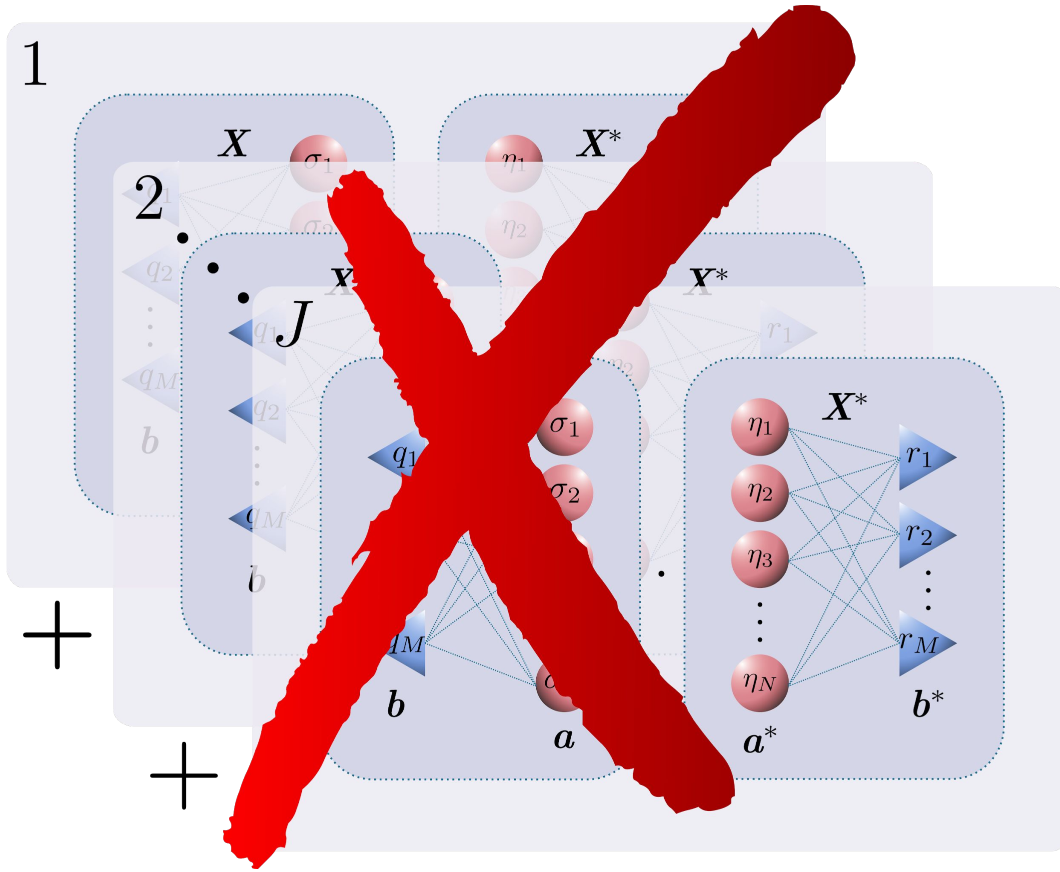
- E.g. playing go or Variational Monte Carlo

Neural Network Density Matrix



$$\rho_\chi(\boldsymbol{\sigma}, \boldsymbol{\eta}) = \sum_{j=1}^{J} p_j(\chi) \cdot \psi_j(\boldsymbol{\sigma}, \chi)\psi_j^*(\boldsymbol{\eta}, \chi)$$

Neural Network Density Matrix



$$\rho_\chi(\boldsymbol{\sigma},\boldsymbol{\eta}) = \sum_{j=1}^{J} p_j(\chi)\cdot\psi_j(\boldsymbol{\sigma},\chi)\psi_j^*(\boldsymbol{\eta},\chi)$$