Two lessons from deep learning

Lei Wang (王磊)

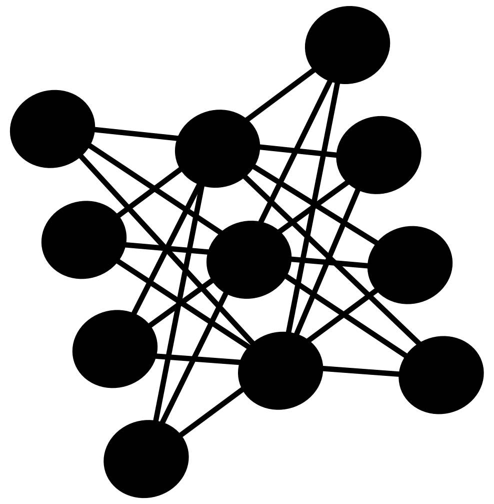
https://wangleiphy.github.io

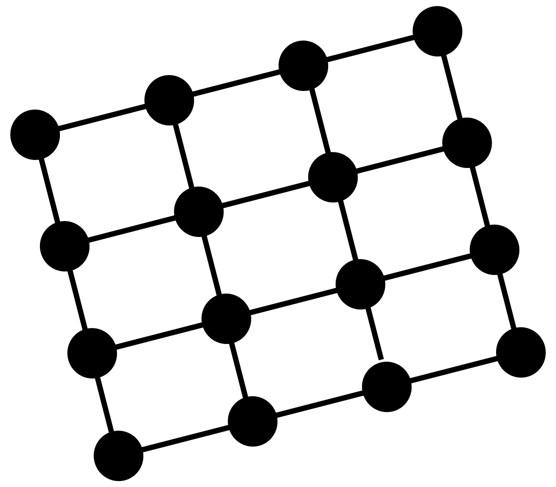
Institute of Physics

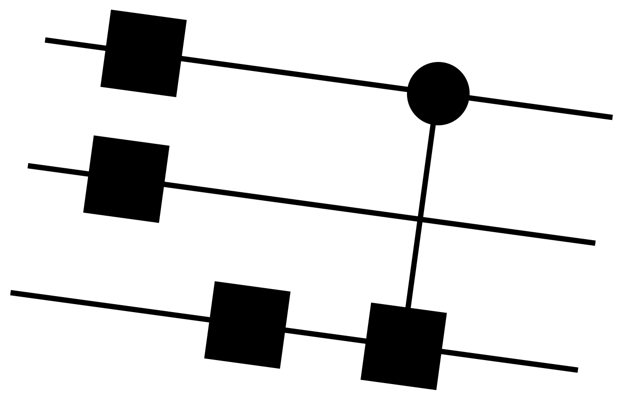
Chinese Academy of Sciences

Two lessons

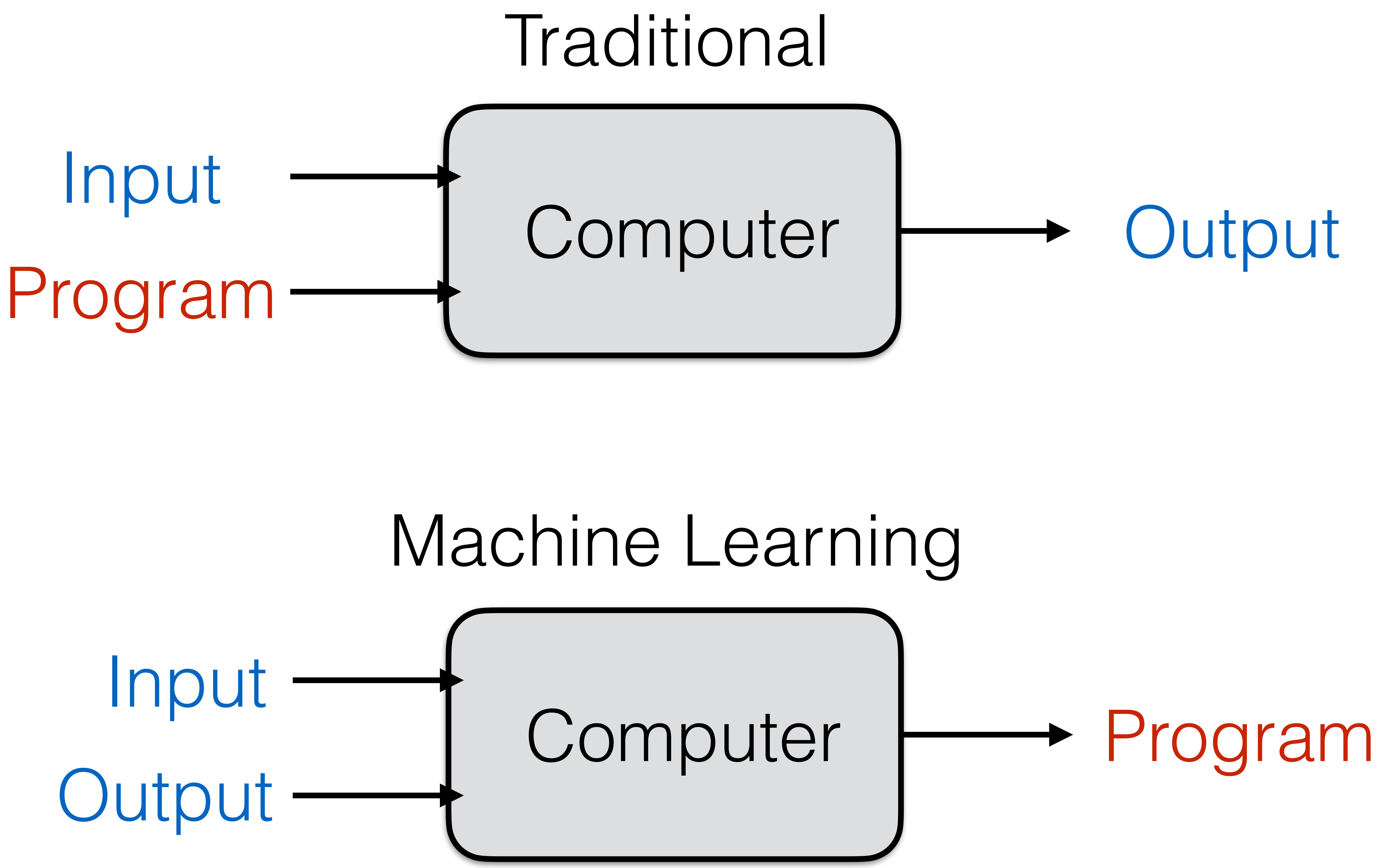
1. Differentiate programming
2. Representation learning







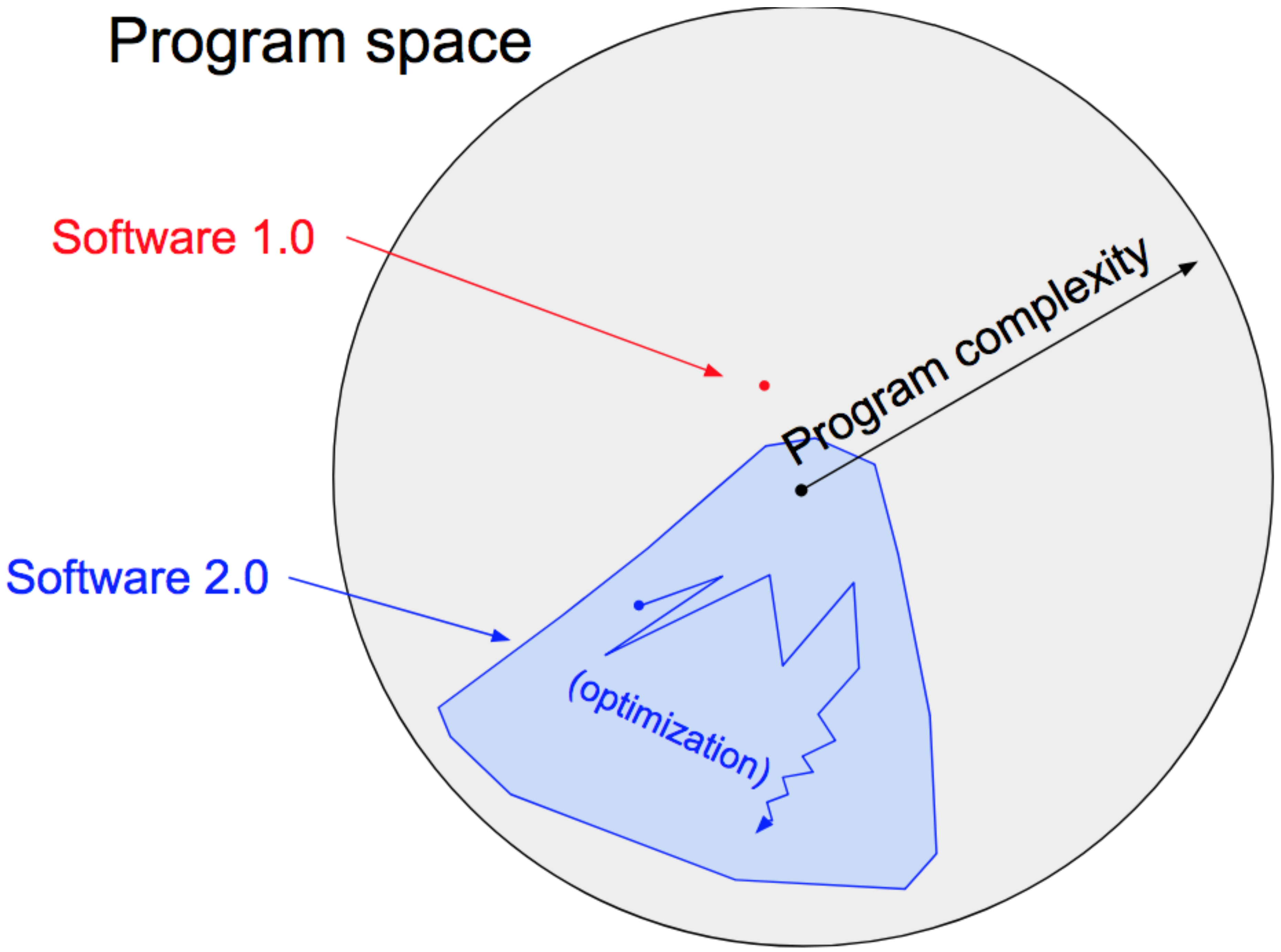
Differentiable Programming



Andrej Karpathy

Director of AI at Tesla. Previously Research Scientist at OpenAI and PhD student at Stanford. I like to train deep neural nets on large datasets.

https://medium.com/@karpathy/software-2-0-a64152b37c35



Writing software 2.0 by gradient search in the program space

Differentiable Programming

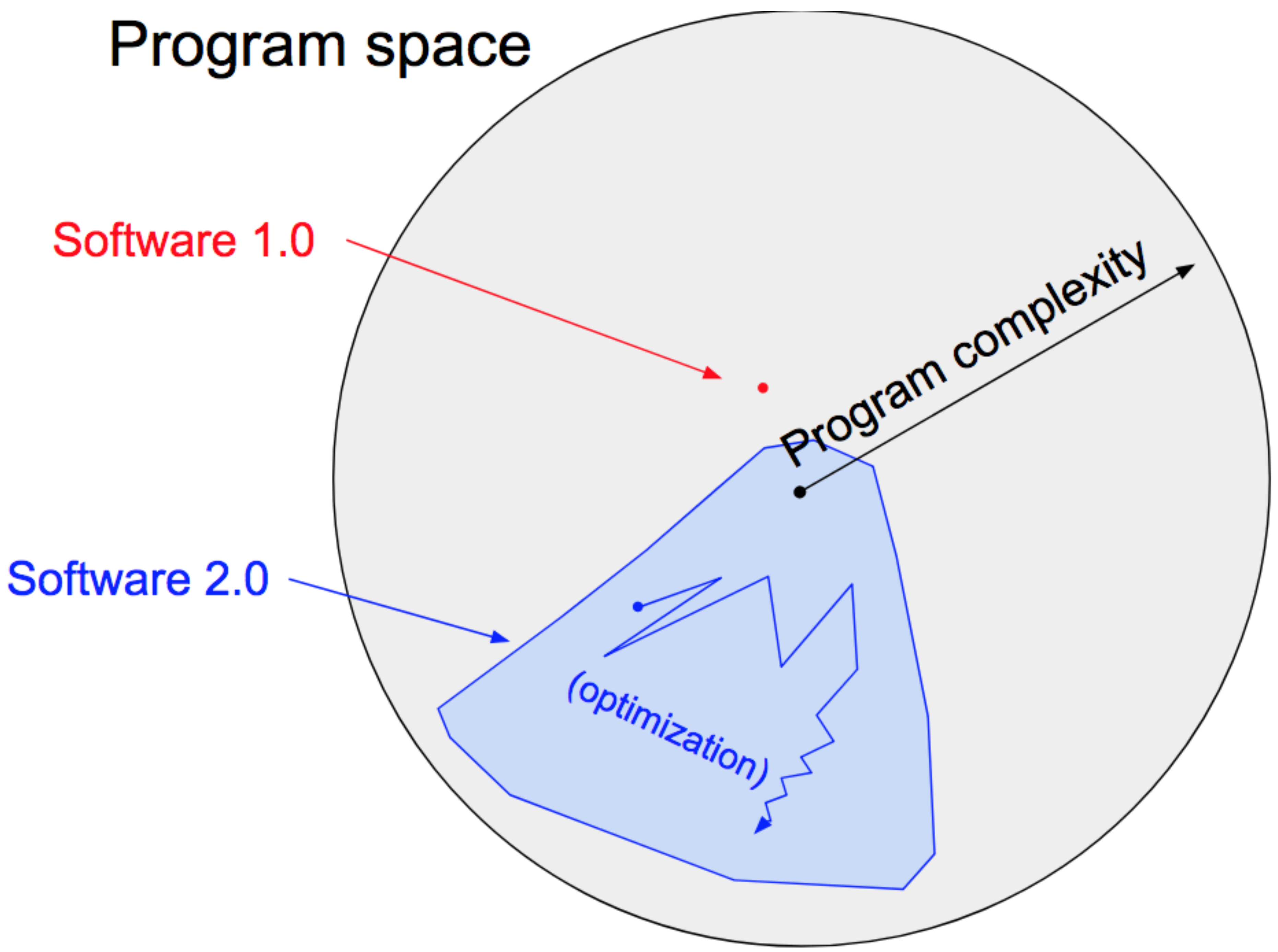
Benefits of Software 2.0

- Computationally homogeneous
- Simple to bake into silicon
- Constant running time
- Constant memory usage
- Highly portable & agile
- Modules can meld into an optimal whole
- **Better than humans**

Andrej Karpathy

Director of AI at Tesla. Previously Research Scientist at OpenAI and PhD student at Stanford. I like to train deep neural nets on large datasets.

https://medium.com/@karpathy/software-2-0-a64152b37c35

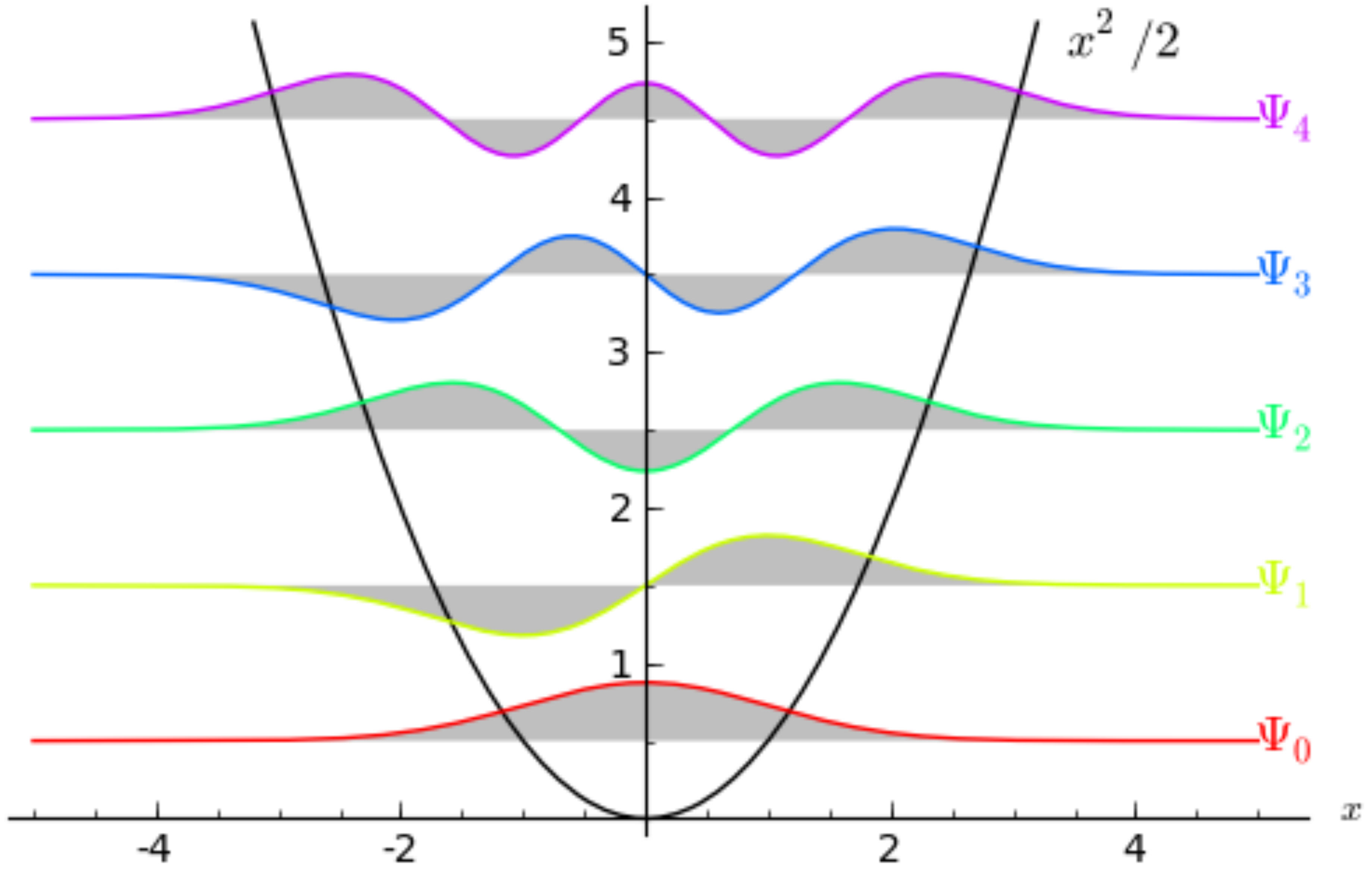


Writing software 2.0 by gradient search in the program space

Demo: Inverse Schrodinger Problem

Given ground state density, how to design the potential ?

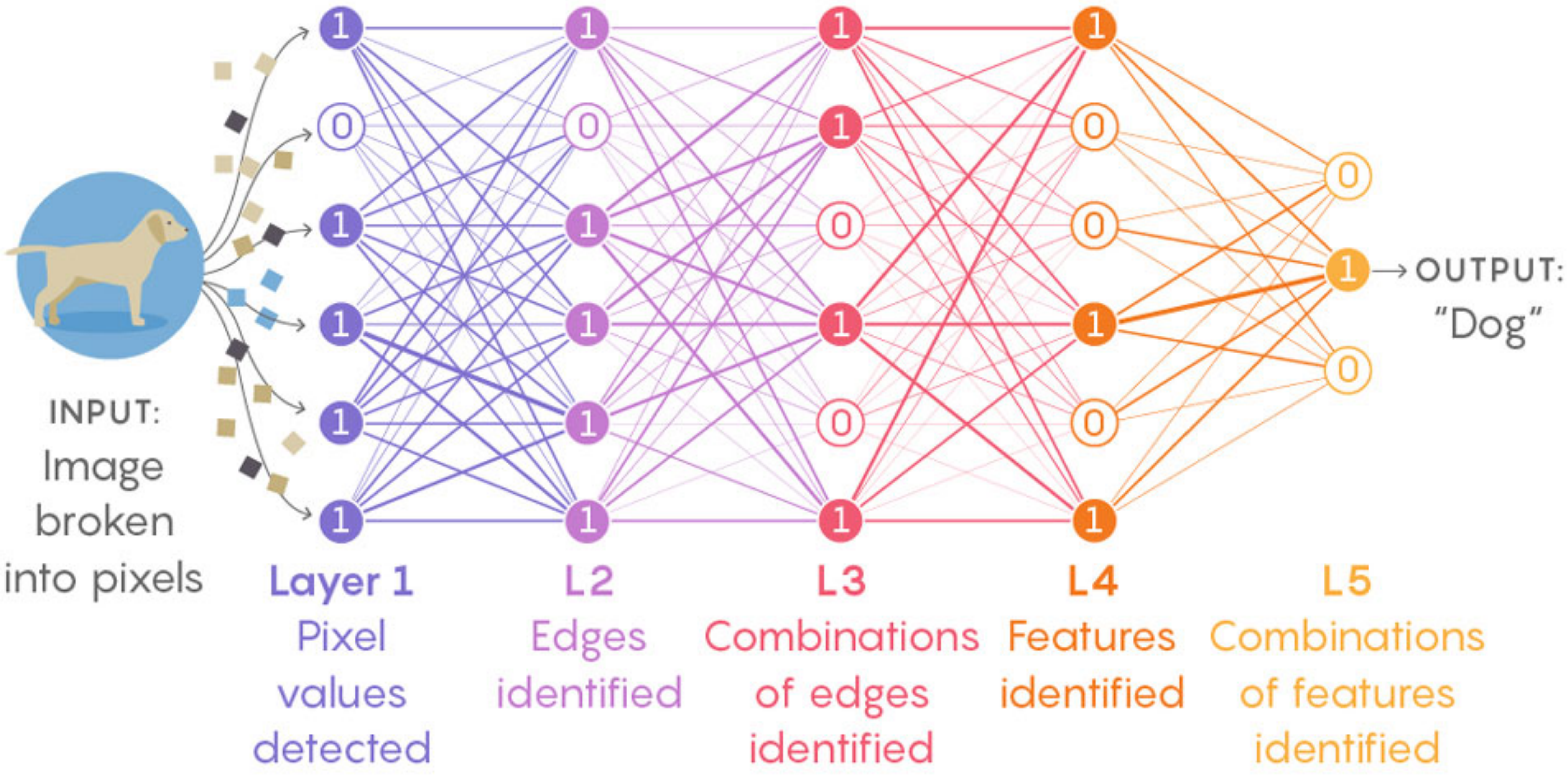
$$\left[-\frac{1}{2}\frac{\partial^2}{\partial x^2} + V(x)\right]\Psi(x) = E\Psi(x)$$



https://colab.research.google.com/drive/1e1NFA-E1Th7nN_9-DzQjAaglH6bwZtVU?usp=sharing

What is under the hood ?

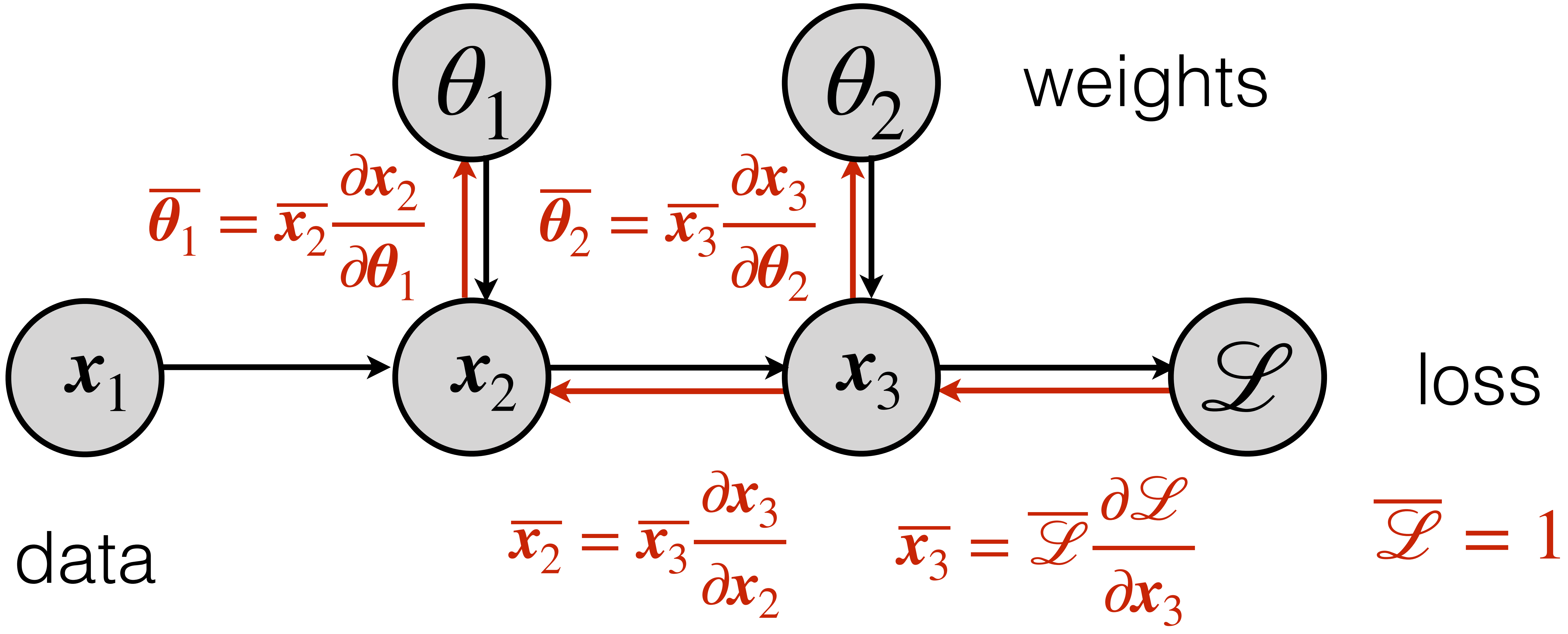
What is deep learning doing?



Deep learning composes differentiable components to a program e.g. a neural network, then optimizes it with gradients

Automatic differentiation on computation graph

“comb graph“

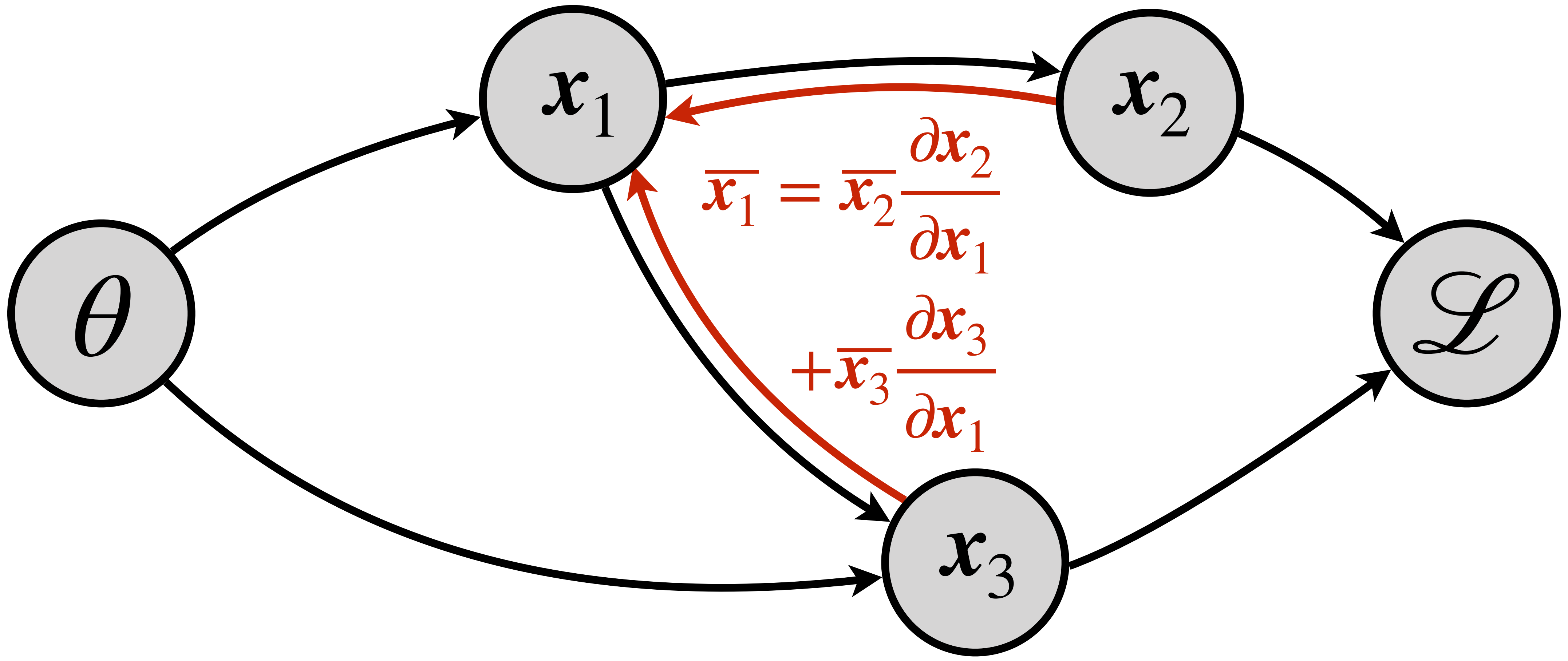


“adjoint variable” $\overline{x} = \frac{\partial \mathscr{L}}{\partial x}$

Pullback the adjoint through the graph

Automatic differentiation on computation graph

directed acyclic graph

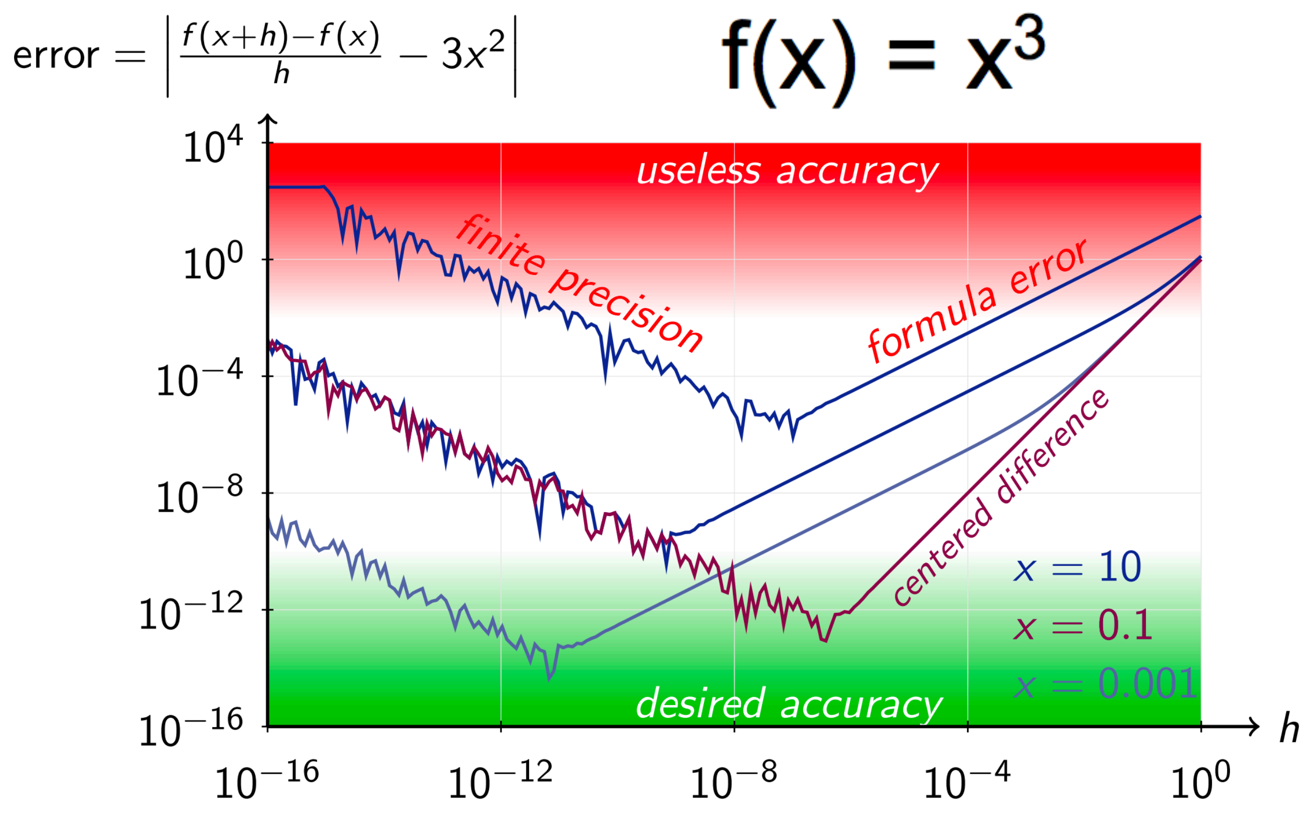


$$\overline{\boldsymbol{x}_i} = \sum_{j:\,\text{child of } i} \overline{\boldsymbol{x}_j}\frac{\partial \boldsymbol{x}_j}{\partial \boldsymbol{x}_i} \qquad \text{with} \quad \overline{\mathscr{L}} = 1$$

Message passing for the adjoint at each node

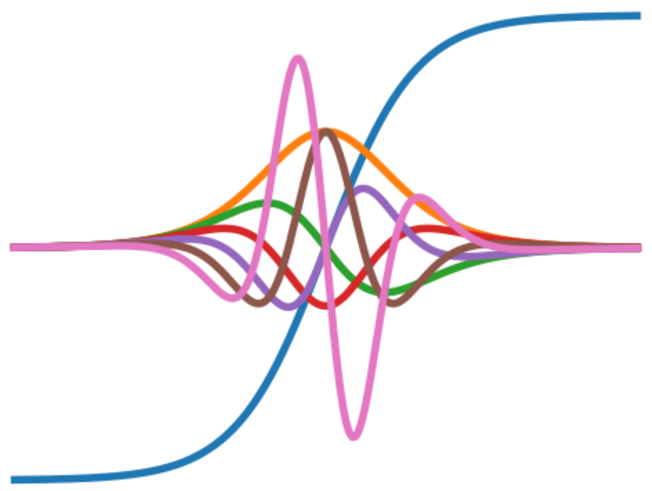
Advantages of automatic differentiation

- Accurate to the machine precision



- Same computational complexity as the function evaluation: Baur-Strassen theorem ’83

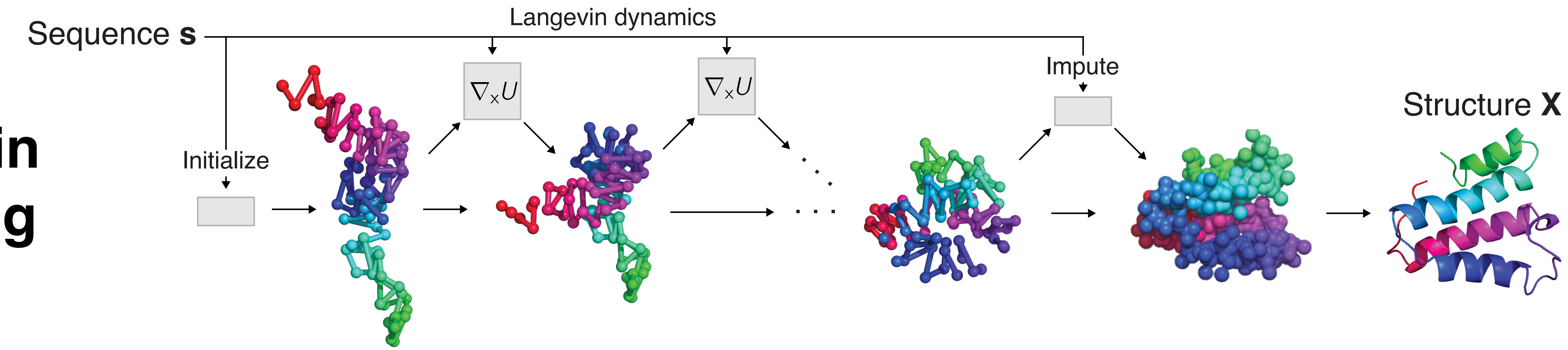
- Supports higher order gradients



```
>>> from autograd import elementwise_grad as egrad  # for functions that vectorize over inputs
>>> import matplotlib.pyplot as plt
>>> x = np.linspace(-7, 7, 200)
>>> plt.plot(x, tanh(x),
...          x, egrad(tanh)(x),                                     # first  derivative
...          x, egrad(egrad(tanh))(x),                              # second derivative
...          x, egrad(egrad(egrad(tanh)))(x),                       # third  derivative
...          x, egrad(egrad(egrad(egrad(tanh))))(x),                # fourth derivative
...          x, egrad(egrad(egrad(egrad(egrad(tanh)))))(x),         # fifth  derivative
...          x, egrad(egrad(egrad(egrad(egrad(egrad(tanh))))))(x))  # sixth  derivative
>>> plt.show()
```

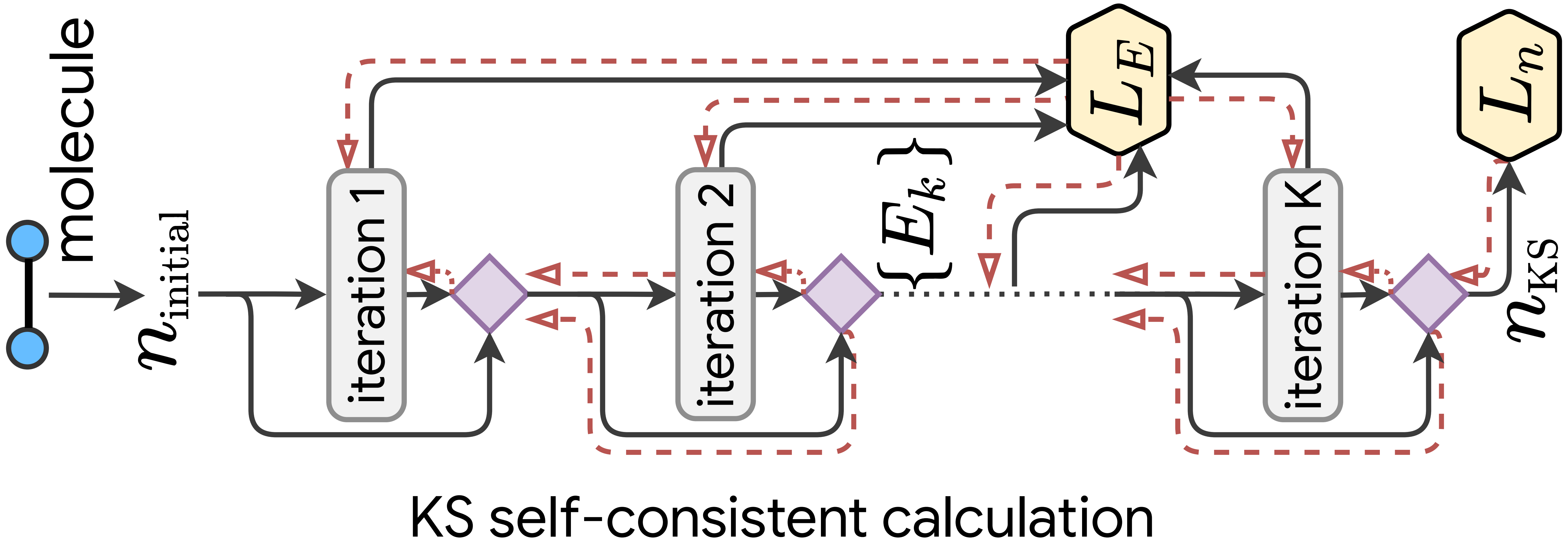
Computer simulation as a learnable model

Protein folding



Ingraham et al
ICLR '19

Density functional learning



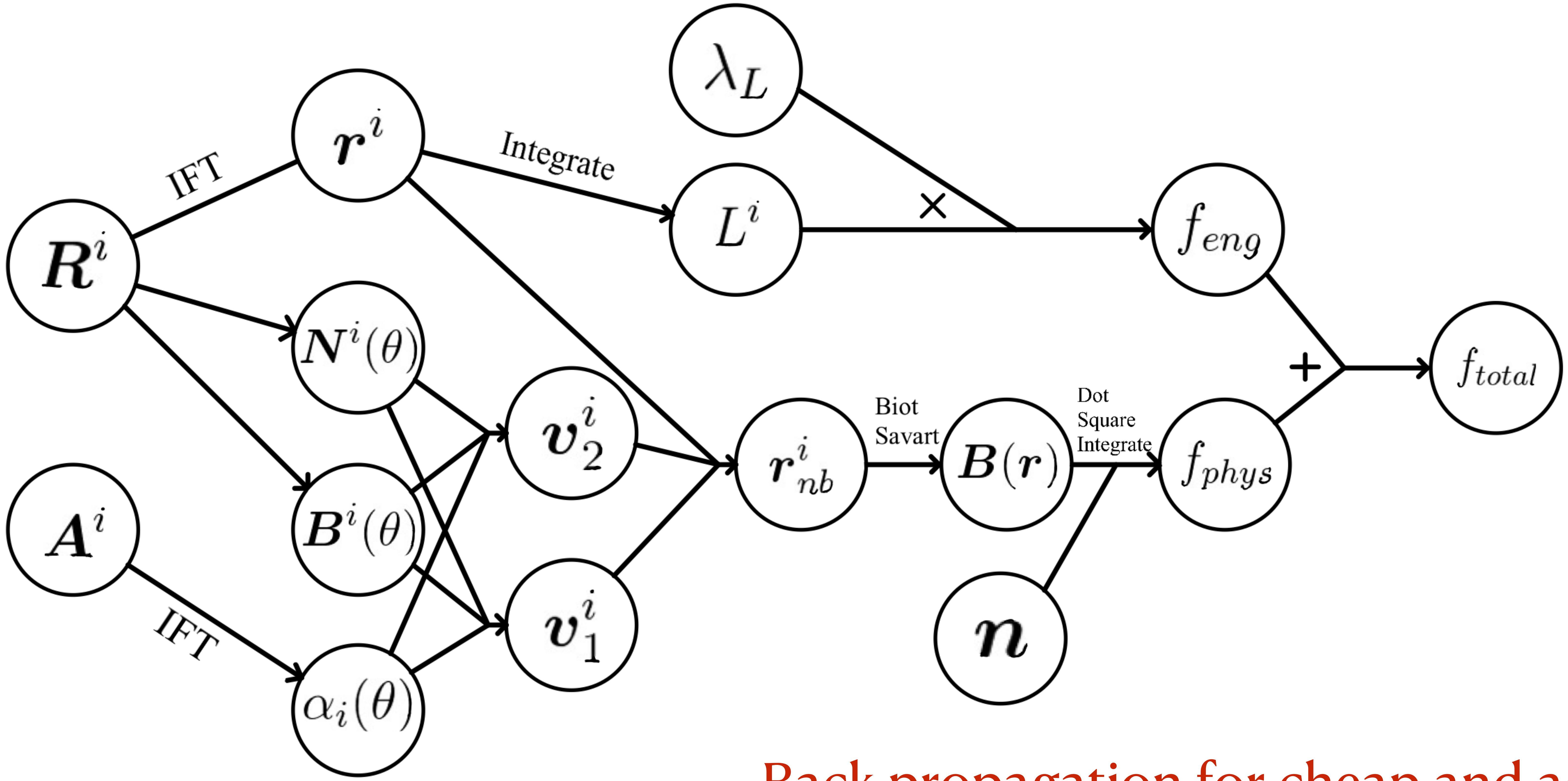
Li et al
PRL '21

Differentiate through the simulation to learn the model

Coil design in fusion reactors (stellarator)

Differentiable stellarator design

Coil parameters

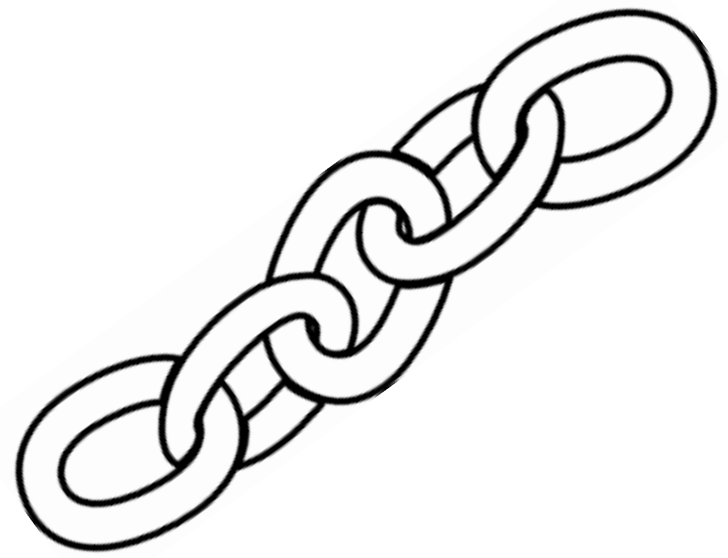


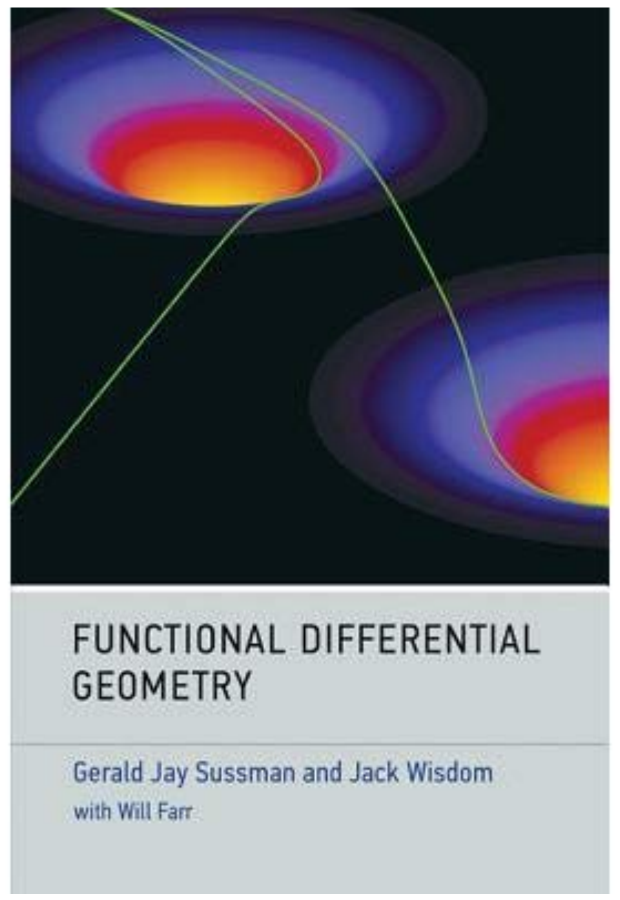
Total cost

Back propagation for cheap and accurate gradient

McGreivy et al 2009.00196

Differentiable programming is broader than training neural networks





Black magic box

Chain rule

Functional differential geometry

https://colab.research.google.com/github/google/jax/blob/master/notebooks/autodiff_cookbook.ipynb

Differentiating a general computer program (rather than neural networks) calls for deeper understanding of the technique

Reverse versus forward mode

$$\frac{\partial \mathscr{L}}{\partial \boldsymbol{\theta}} = \frac{\partial \mathscr{L}}{\partial \boldsymbol{x}_n} \frac{\partial \boldsymbol{x}_n}{\partial \boldsymbol{x}_{n-1}} \cdots \frac{\partial \boldsymbol{x}_2}{\partial \boldsymbol{x}_1} \frac{\partial \boldsymbol{x}_1}{\partial \boldsymbol{\theta}}$$

Reverse mode AD: Vector-Jacobian Product of primitives

- Backtrace the computation graph
- Needs to store intermediate results
- Efficient for graphs with large fan-in

Backpropagation = Reverse mode AD applied to neural networks

Reverse versus forward mode

$$\frac{\partial \mathscr{L}}{\partial \boldsymbol{\theta}} = \frac{\partial \mathscr{L}}{\partial \boldsymbol{x}_n} \frac{\partial \boldsymbol{x}_n}{\partial \boldsymbol{x}_{n-1}} \cdots \frac{\partial \boldsymbol{x}_2}{\partial \boldsymbol{x}_1} \frac{\partial \boldsymbol{x}_1}{\partial \boldsymbol{\theta}}$$

Forward mode AD: Jacobian-Vector Product of primitives

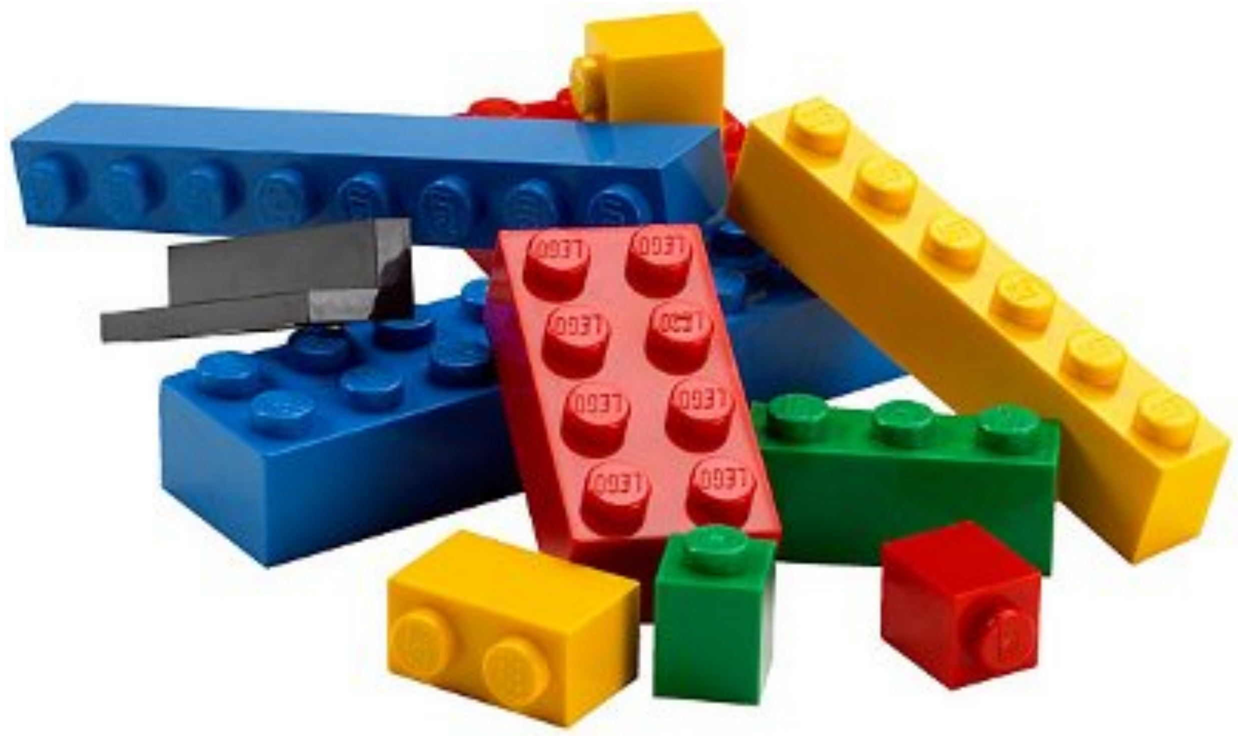
- Same order with the function evaluation
- No storage overhead
- Efficient for graph with large fan-out

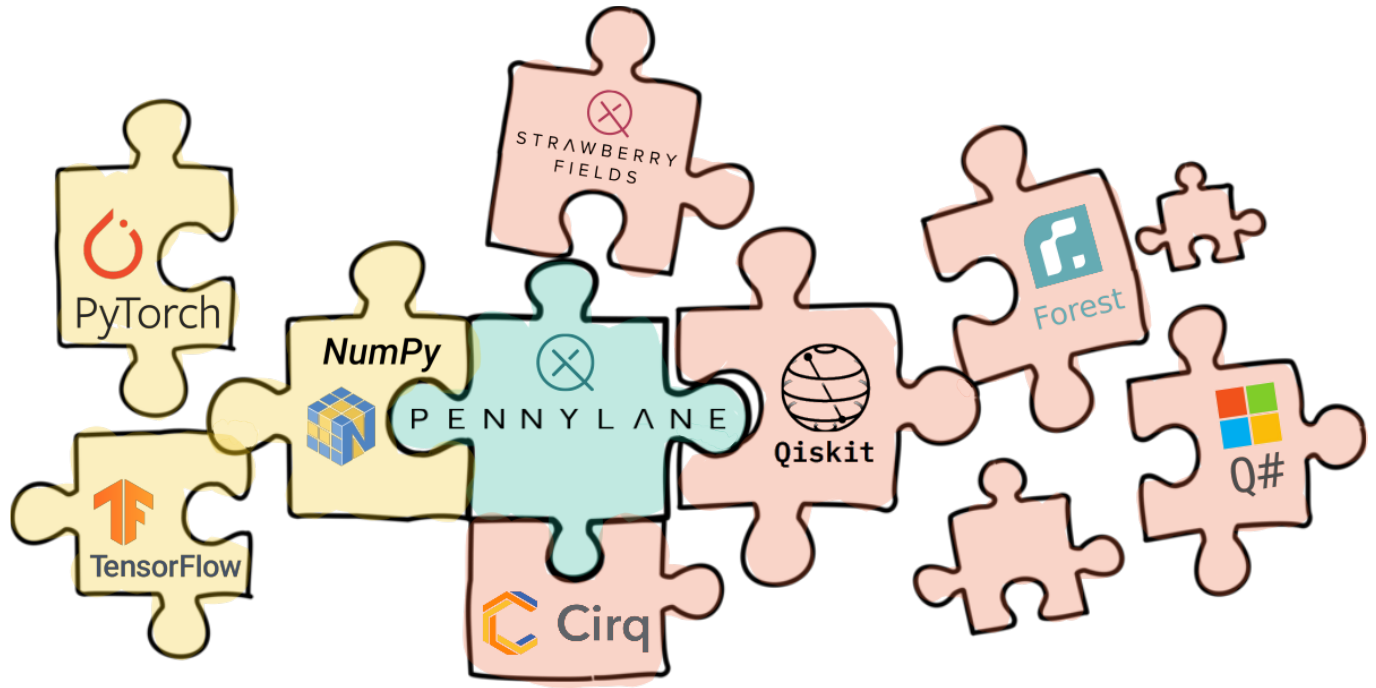
Less efficient for scalar output, but useful for higher-order derivatives

How to think about AD ?

- AD is modular, and one can control its granularity
- Benefits of writing customized primitives
  - Reducing memory usage
  - Increasing numerical stability
  - Call to external libraries written agnostically to AD

    (or, even a quantum processor)





https://github.com/PennyLaneAI/pennylane

Example of primitives

~200 functions to cover most of numpy in HIPS/autograd

https://github.com/HIPS/autograd/blob/master/autograd/numpy/numpy_vjps.py

Operators	+, −, *, /, (−), **, %, <, <=, ==, !=, >=, >
Basic math functions	exp, log, square, sqrt, sin, cos, tan, sinh, cosh, tanh, sinc, abs, fabs, logaddexp, logaddexp2, absolute, reciprocal, exp2, expm1, log2, log10, log1p, arcsin, arccos, arctan, arcsinh, arccosh, arctanh, rad2deg, degrees, deg2rad, radians
Complex numbers	real, imag, conj, angle, fft, fftshift, ifftshift, real_if_close
Array reductions	sum, mean, prod, var, std, max, min, amax, amin
Array reshaping	reshape, ravel, squeeze, diag, roll, array_split, split, vsplit, hsplit, dsplit, expand_dims, flipud, fliplr, rot90, swapaxes, rollaxis, transpose, atleast_1d, atleast_2d, atleast_3d
Linear algebra	dot, tensordot, einsum, cross, trace, outer, det, slogdet, inv, norm, eigh, cholesky, sqrtm, solve_triangular
Other array operations	cumsum, clip, maximum, minimum, sort, msort, partition, concatenate, diagonal, truncate_pad, tile, full, triu, tril, where, diff, nan_to_num, vstack, hstack
Probability functions	t.pdf, t.cdf, t.logpdf, t.logcdf, multivariate_normal.logpdf, multivariate_normal.pdf, multivariate_normal.entropy, norm.pdf, norm.cdf, norm.logpdf, norm.logcdf,

```python
# ----- Simple grads -----

defvjp(anp.negative, lambda ans, x: lambda g: -g)
defvjp(anp.abs,
    lambda ans, x : lambda g: g * replace_zero(anp.conj(x), 0.) / replace_zero(ans, 1.))
defvjp(anp.fabs,     lambda ans, x : lambda g: anp.sign(x) * g)  # fabs doesn't take complex numbers.
defvjp(anp.absolute, lambda ans, x : lambda g: g * anp.conj(x) / ans)
defvjp(anp.reciprocal, lambda ans, x : lambda g: - g / x**2)
defvjp(anp.exp,    lambda ans, x : lambda g: ans * g)
defvjp(anp.exp2,   lambda ans, x : lambda g: ans * anp.log(2) * g)
defvjp(anp.expm1,  lambda ans, x : lambda g: (ans + 1) * g)
defvjp(anp.log,    lambda ans, x : lambda g: g / x)
defvjp(anp.log2,   lambda ans, x : lambda g: g / x / anp.log(2))
defvjp(anp.log10,  lambda ans, x : lambda g: g / x / anp.log(10))
defvjp(anp.log1p,  lambda ans, x : lambda g: g / (x + 1))
defvjp(anp.sin,    lambda ans, x : lambda g: g * anp.cos(x))
defvjp(anp.cos,    lambda ans, x : lambda g: - g * anp.sin(x))
defvjp(anp.tan,    lambda ans, x : lambda g: g / anp.cos(x) **2)
defvjp(anp.arcsin, lambda ans, x : lambda g: g / anp.sqrt(1 - x**2))
defvjp(anp.arccos, lambda ans, x : lambda g:-g / anp.sqrt(1 - x**2))
defvjp(anp.arctan, lambda ans, x : lambda g: g / (1 + x**2))
defvjp(anp.sinh,   lambda ans, x : lambda g: g * anp.cosh(x))
defvjp(anp.cosh,   lambda ans, x : lambda g: g * anp.sinh(x))
defvjp(anp.tanh,   lambda ans, x : lambda g: g / anp.cosh(x) **2)
defvjp(anp.arcsinh, lambda ans, x : lambda g: g / anp.sqrt(x**2 + 1))
defvjp(anp.arccosh, lambda ans, x : lambda g: g / anp.sqrt(x**2 - 1))
defvjp(anp.arctanh, lambda ans, x : lambda g: g / (1 - x**2))
defvjp(anp.rad2deg, lambda ans, x : lambda g: g / anp.pi * 180.0)
defvjp(anp.degrees, lambda ans, x : lambda g: g / anp.pi * 180.0)
defvjp(anp.deg2rad, lambda ans, x : lambda g: g * anp.pi / 180.0)
defvjp(anp.radians, lambda ans, x : lambda g: g * anp.pi / 180.0)
defvjp(anp.square,  lambda ans, x : lambda g: g * 2 * x)
defvjp(anp.sqrt,    lambda ans, x : lambda g: g * 0.5 * x**-0.5)
```

Loop/Condition/Sort/Permutations are also differentiable

http://videolectures.net/deeplearning2017_johnson_automatic_differentiation/

Differentiable programming tools

HIPS/autograd

theano

Zygote

PyTorch

TensorFlow

SciML

Keras

NiLang

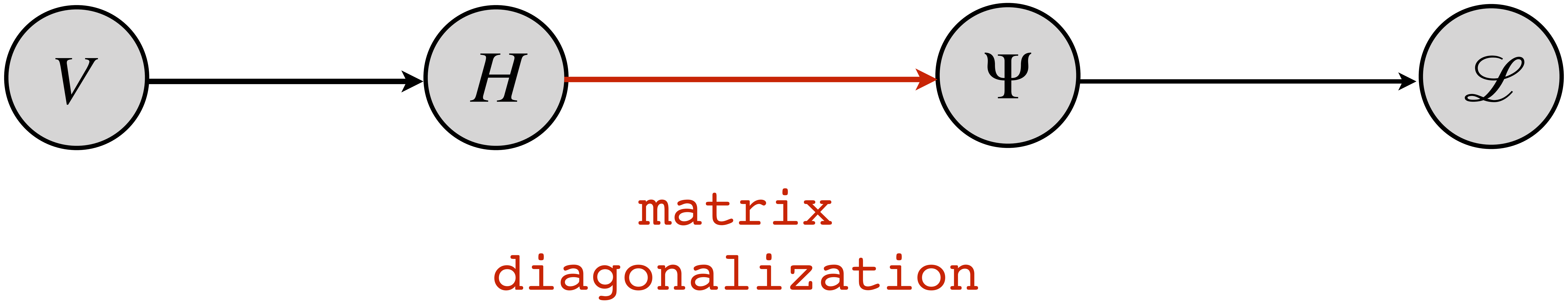
Differentiable Scientific Computing

- Many scientific computations (FFT, Eigen, SVD!) are differentiable
- ODE integrators are differentiable with O(1) memory
- Differentiable ray tracer and Differentiable fluid simulations
- Differentiable Monte Carlo/Tensor Network/Functional RG/Dynamical Mean Field Theory/Density Functional Theory/Hartree-Fock/Coupled Cluster/Gutzwiller/Molecular Dynamics…

Differentiate through domain-specific computational processes to solve learning, control, optimization and inverse problems

Differentiable Eigensolver

Inverse Schrodinger Problem



Useful for inverse Kohn-Sham problem, Jensen & Wasserman '17

Differentiable Eigensolver

$$H\Psi = \Psi E$$

Forward mode: What happen if $H \to H + dH$? Perturbation theory

Reverse mode: How should I change H given $\partial\mathscr{L}/\partial\Psi$ and $\partial\mathscr{L}/\partial E$? **Transposed perturbation theory!**

Hamiltonian engineering via differentiable programming

https://github.com/wangleiphy/DL4CSRC/tree/master/2-ising See also Fujita et al, PRB '18

$$H \longrightarrow |\Psi\rangle \longrightarrow \langle O\rangle$$

Automatic differentiation

Linear response theory

where $F_{i,j} = (d_j - d_i)^{-1}$ for $i \neq j$, and zero otherwise. Hence, the forward mode sensitivity equations are

$$\begin{aligned}\dot{D} &= I \circ (U^{-1}\dot{A}\,U),\\ \dot{U} &= U\left(F \circ (U^{-1}\dot{A}\,U)\right).\end{aligned}$$

In reverse mode, using the identity $\mathrm{Tr}(A\,(B\circ C)) = \mathrm{Tr}((A\circ B^T)\,C)$, we get

$$\begin{aligned}\mathrm{Tr}\left(\overline{D}^T \mathrm{d}D + \overline{U}^T \mathrm{d}U\right) &= \mathrm{Tr}\left(\overline{D}^T U^{-1}\mathrm{d}A\,U\right) + \mathrm{Tr}\left(\overline{U}^T U\,(F\circ(U^{-1}\mathrm{d}A\,U))\right)\\ &= \mathrm{Tr}\left(\overline{D}^T U^{-1}\mathrm{d}A\,U\right) + \mathrm{Tr}\left(\left((\overline{U}^T U)\circ F^T\right)\,U^{-1}\mathrm{d}A\,U\right)\\ &= \mathrm{Tr}\left(U\left(\overline{D}^T + (\overline{U}^T U)\circ F^T\right)\,U^{-1}\mathrm{d}A\right)\end{aligned}$$

and so

$$\overline{A} = U^{-T}\left(\overline{D} + F\circ(U^T\,\overline{U})\right)\,U^T.$$

An extended collection of matrix derivative results for forward and reverse mode algorithmic differentiation

https://people.maths.ox.ac.uk/gilesm/files/NA-08-01.pdf

same thing

In an independent-particle approximation the states are determined by the hamiltonian $\hat{H}_{\mathrm{eff}}$ in the effective Schrödinger equation (3.36). The change in the individual independent-particle orbitals, $\Delta\psi_i(\mathbf{r})$ to first order in perturbation theory, can be written in terms of a sum over the spectrum of the unperturbed hamiltonian $\hat{H}^0_{\mathrm{eff}}$ as [11,265,266],

$$\Delta\psi_i(\mathbf{r}) = \sum_{j\neq i}\psi_j(\mathbf{r})\frac{\langle\psi_j|\Delta\hat{H}_{\mathrm{eff}}|\psi_i\rangle}{\varepsilon_i - \varepsilon_j}, \qquad (3.61)$$

where the sum is over all the states of the system, occupied and empty, with the exception of the state being considered. Similarly, the change in the expectation value of an operator $\hat{O}$ in the perturbed ground state to lowest order in $\Delta\hat{H}_{\mathrm{eff}}$ can be written

$$\begin{aligned}\Delta\langle\hat{O}\rangle &= \sum_{i=1}^{\mathrm{occ}}\langle\psi_i + \delta\psi_i|\hat{O}|\psi_i + \delta\psi_i\rangle\\ &= \sum_{i=1}^{\mathrm{occ}}\sum_{j}^{\mathrm{empty}}\frac{\langle\psi_i|\hat{O}|\psi_j\rangle\langle\psi_j|\Delta\hat{H}_{\mathrm{eff}}|\psi_i\rangle}{\varepsilon_i - \varepsilon_j} + \mathrm{c.c.}\end{aligned} \qquad (3.62)$$

In (3.62) the sum over j is restricted to conduction states only, which follows from the fact that the contributions of pairs of occupied states i, j and j, i cancel in (3.62) (Exercise 3.21). Expressions Eqs. (3.61) and (3.62) are the basic equations upon which is built the theory of response functions (App. D) and methods for calculating static (Ch. 19) and dynamic responses (Ch. 20) in materials.

Richard Martin, Electronic structure

Density functional perturbation theory

$$Q_n = \left.\frac{d^n E}{d\lambda^n}\right|_{\lambda\to 0}$$

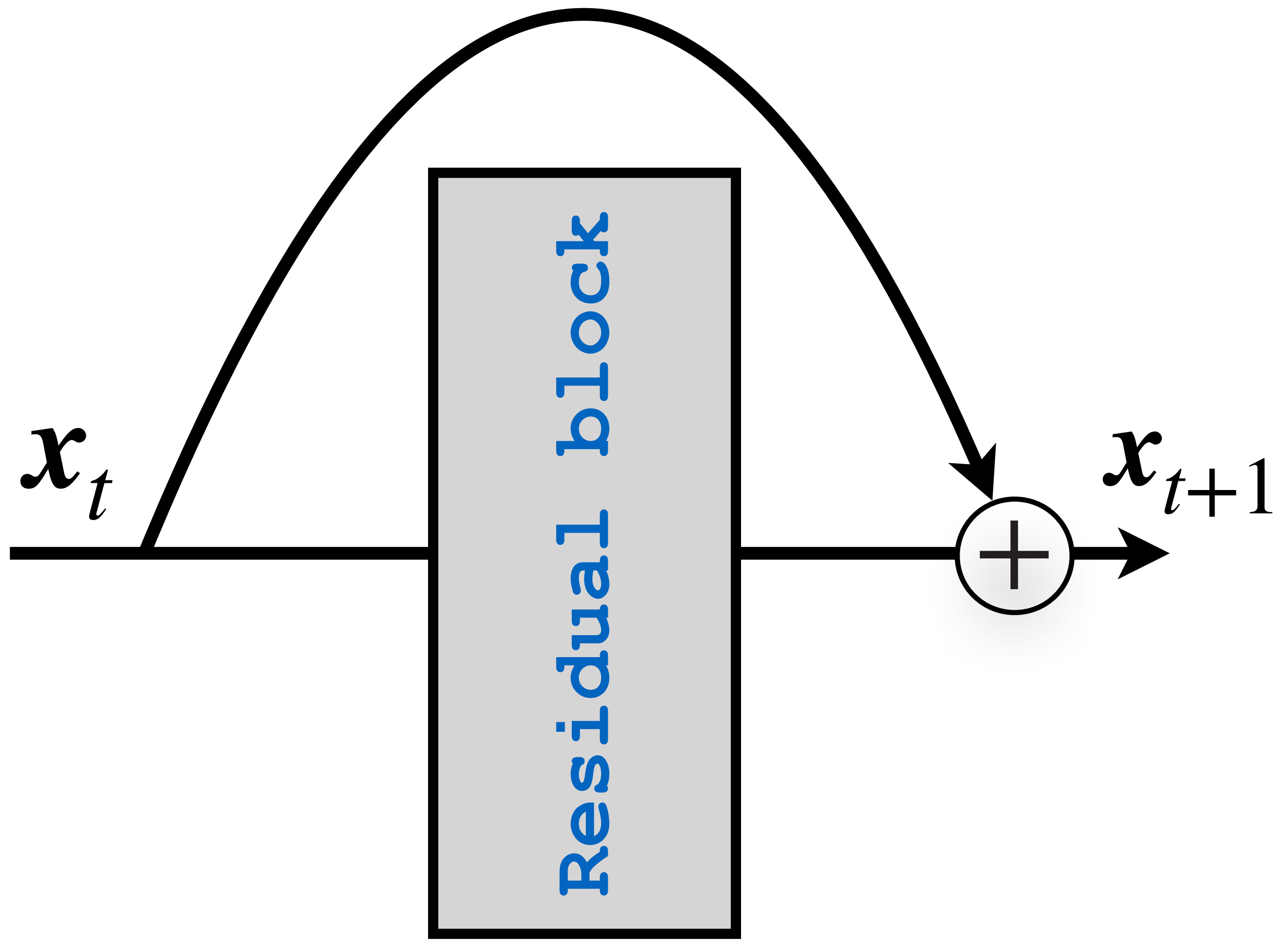
type of perturbation λ	order n	physical property Q
displacements of atoms	1	atomic force
$\delta\mathbf{R}$	2	force constants
	≥ 3	anharmonic force constants
homogeneous strain η	1	stress
	2	elastic constants
	≥ 3	higher order elastic constants
homogeneous electric field $\mathbf{E}$	1	dipole moment
	2	polarizability
$\delta\mathbf{R} + \eta$	2+1	Grüneisen parameter
$\delta\mathbf{R} + \mathbf{E}$	1+2	Raman scattering cross section

Baroni et al, RMP 2001

Differentiable DFT for a unified, flexible, and (very likely) more efficient framework

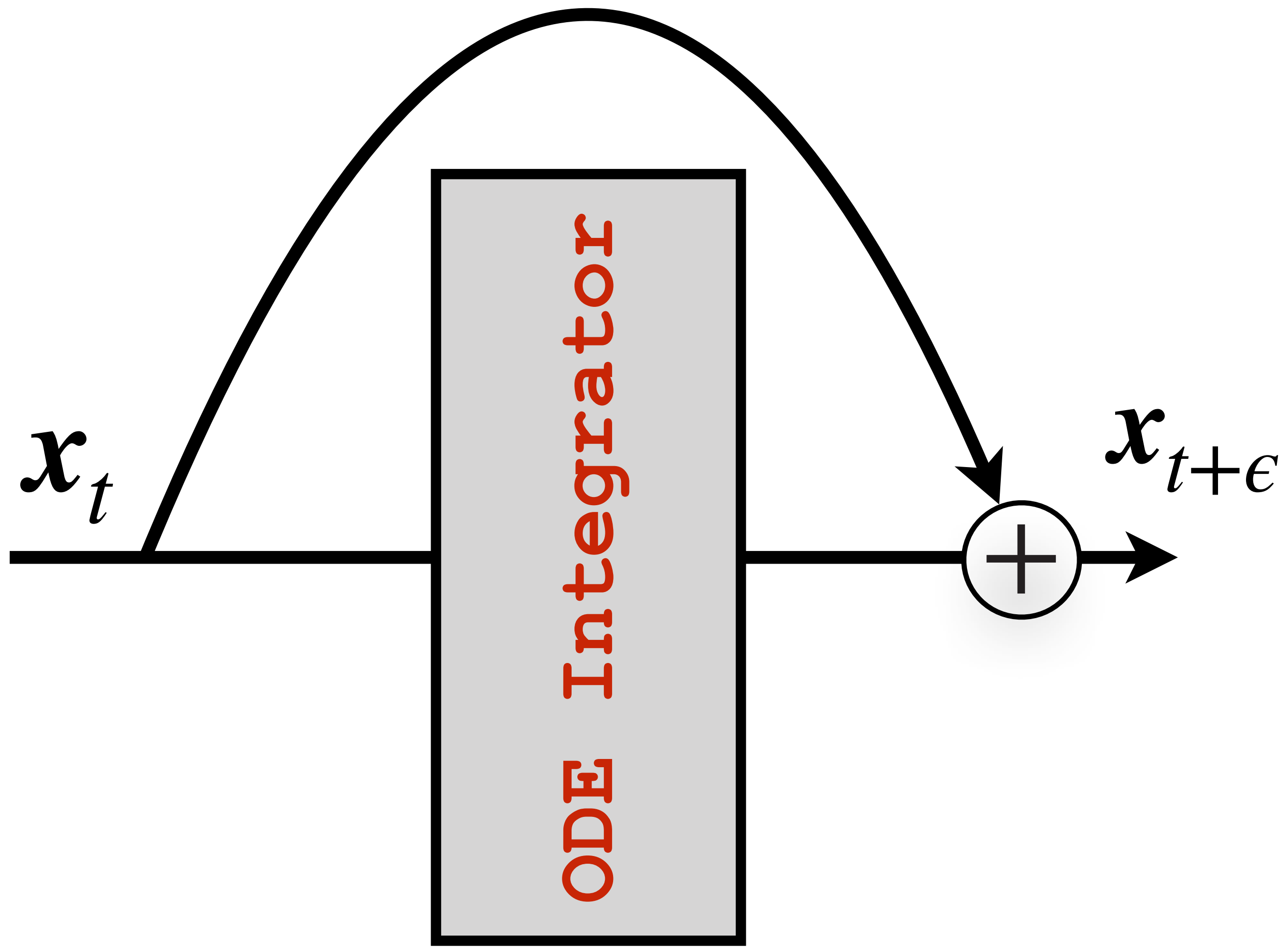
Neural Ordinary Differential Equations

Residual network



$$x_{t+1} = x_t + f(x_t)$$

ODE integration



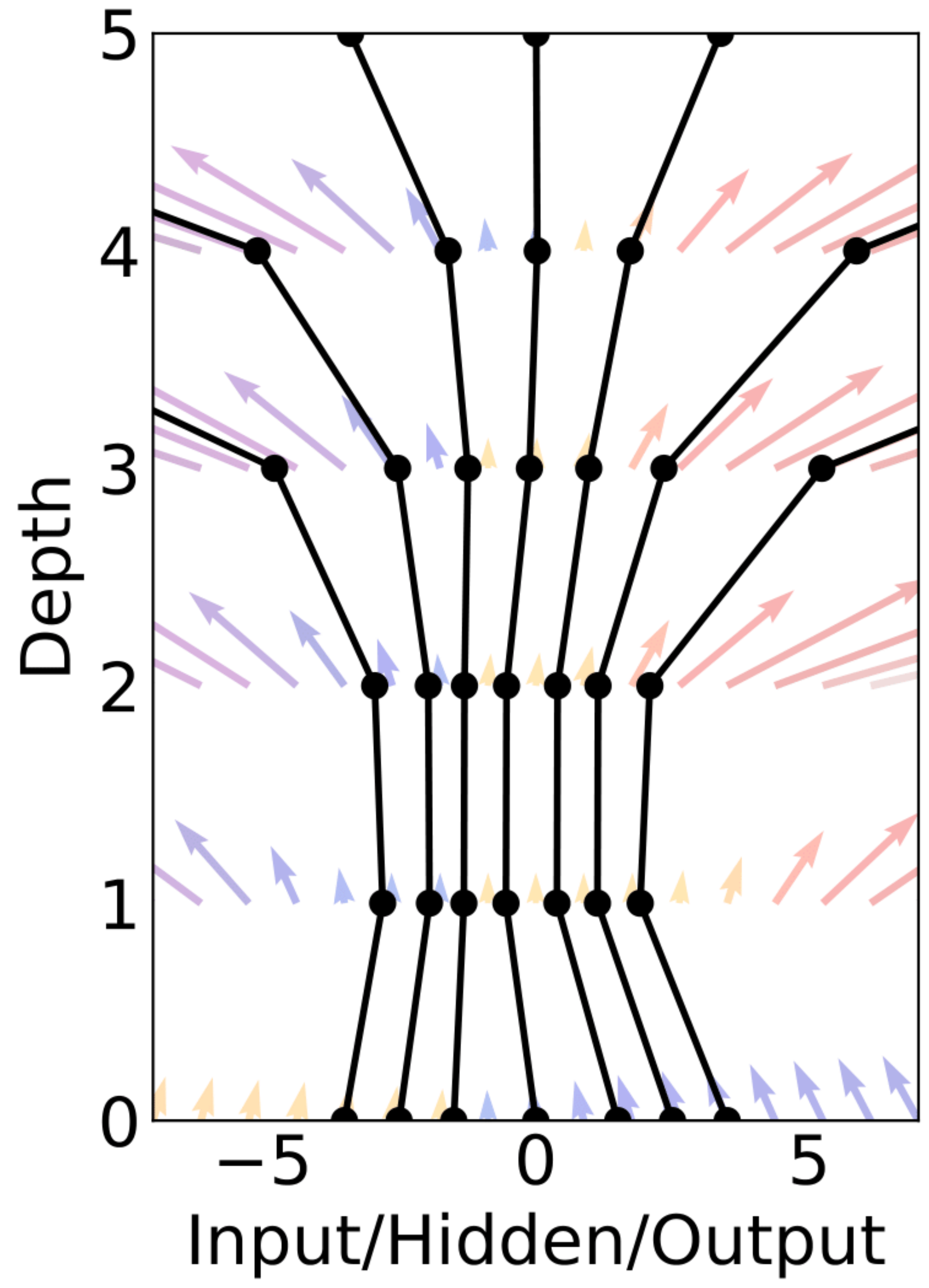
$$dx/dt = f(x)$$

Chen et al, 1806.07366 NIPS '18 Best paper award

cf Harbor el al 1705.03341
Lu et al 1710.10121, E 17'…

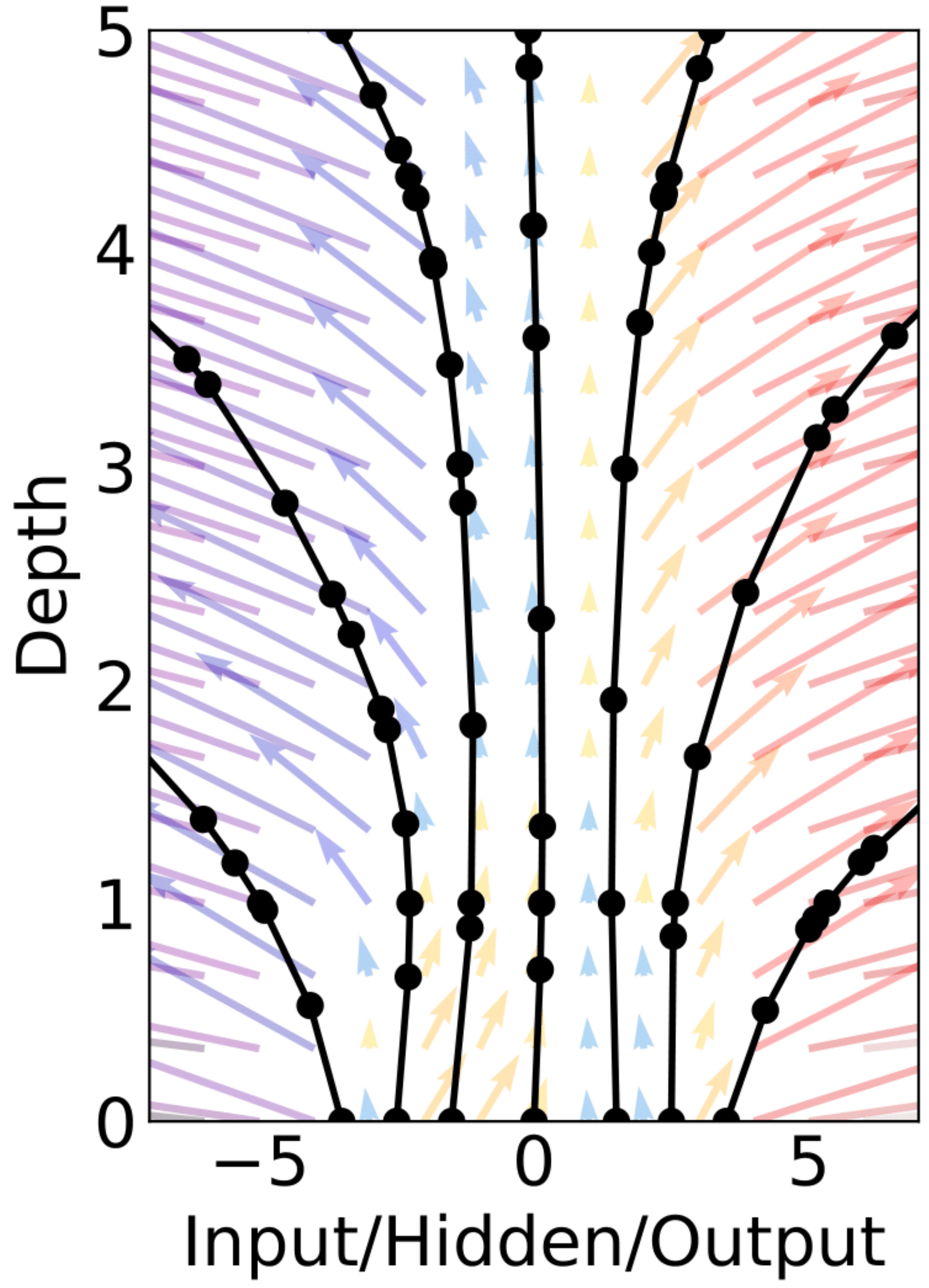
Neural Ordinary Differential Equations

Residual network



$$x_{t+1} = x_t + f(x_t)$$

ODE integration



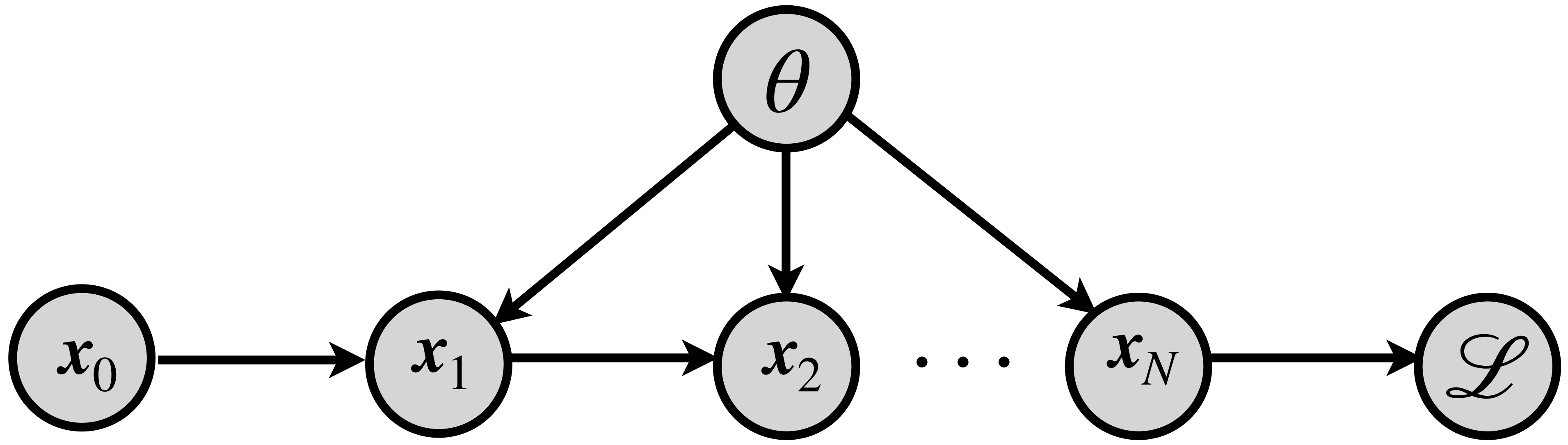
$$dx/dt = f(x)$$

Chen et al, 1806.07366 NIPS '18 Best paper award

cf Harbor el al 1705.03341
Lu et al 1710.10121, E 17'…

Backpropagate through ODE solver

$$\frac{d\boldsymbol{x}}{dt} = f(\boldsymbol{x}, \boldsymbol{\theta}, t)$$



Adjoint $\quad \overline{\boldsymbol{x}}(t) = \dfrac{\partial \mathscr{L}}{\partial \boldsymbol{x}(t)} \quad$ satisfies another ODE

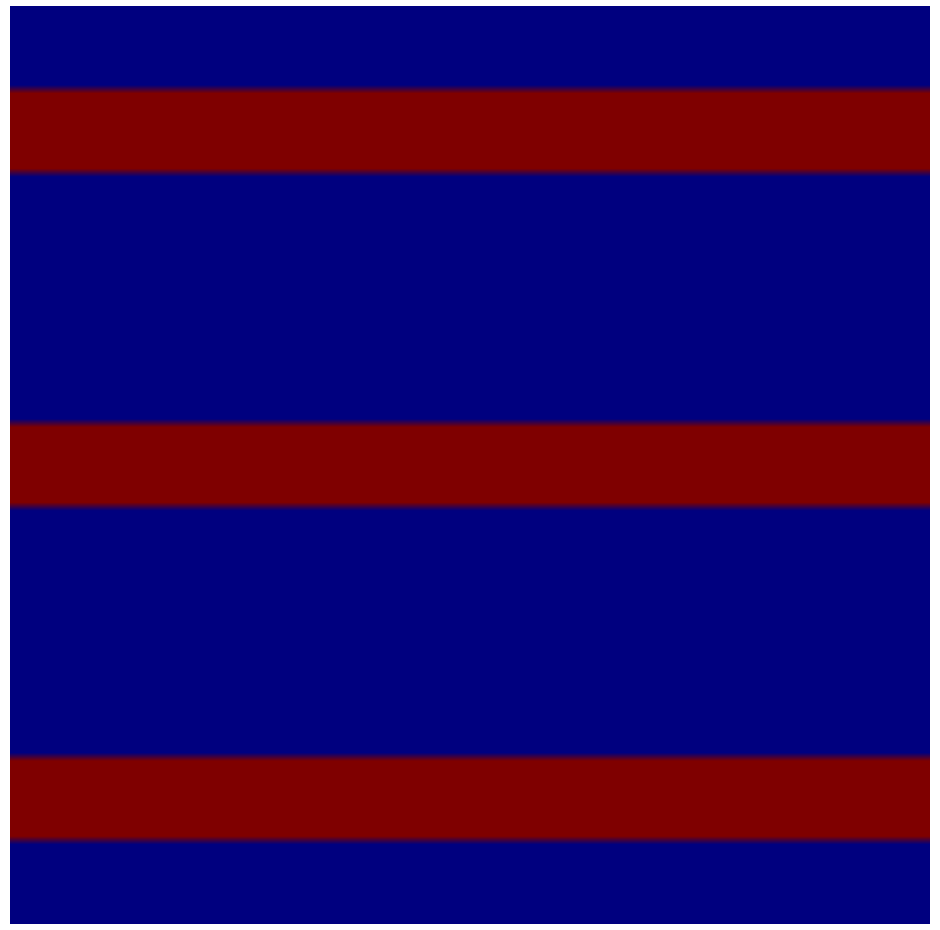
$$\frac{d\overline{\boldsymbol{x}}(t)}{dt} = -\,\overline{\boldsymbol{x}}(t)\frac{\partial f(\boldsymbol{x}, \boldsymbol{\theta}, t)}{\partial \boldsymbol{x}}$$

Gradient w.r.t. parameter

$$\frac{\partial \mathscr{L}}{\partial \boldsymbol{\theta}} = \int_0^T dt\, \overline{\boldsymbol{x}}(t)\frac{\partial f(\boldsymbol{x}, \boldsymbol{\theta}, t)}{\partial \boldsymbol{\theta}}$$

Exercise:
Derive this!

Why do we need Neural ODE ?



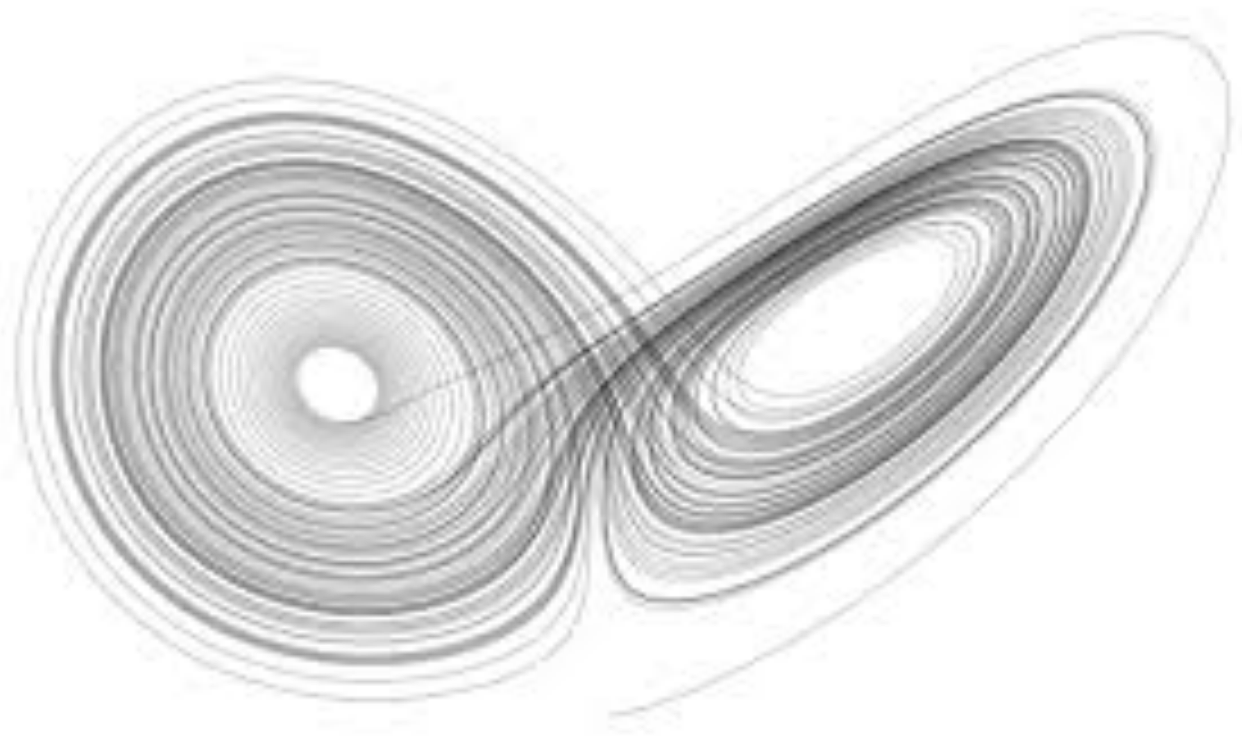
[Backpropagating through a fluid simulation]()

- Neural ODE has constant memory usage
- Works for black box ODE integrator
- Works with adaptive steps and implicit schemes

Differentiable ODE integrators

"Neural ODE" Chen et al, 1806.07366

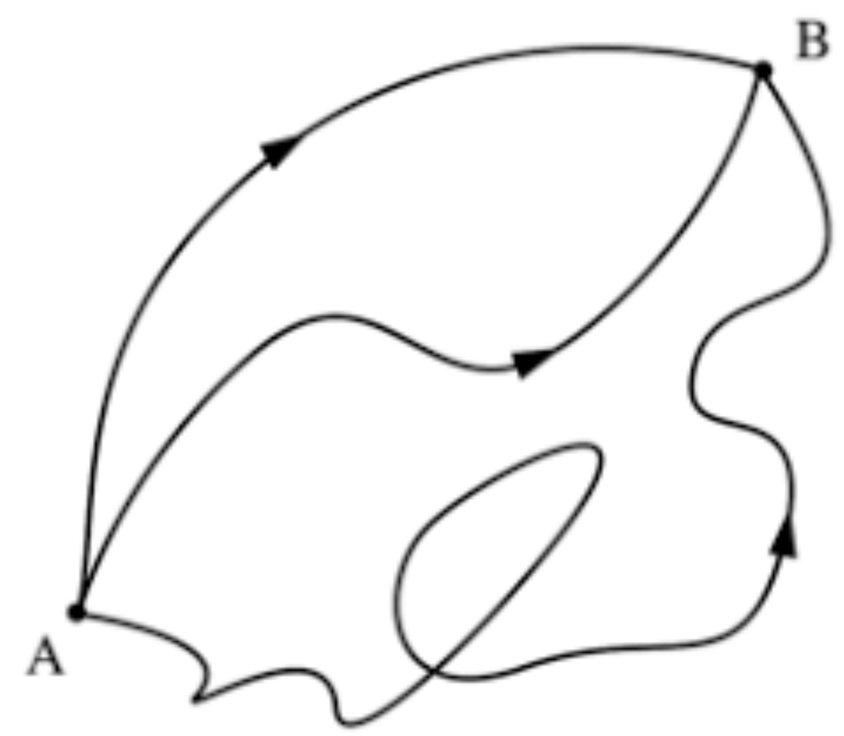
Dynamics systems



$$\frac{dx}{dt} = f_\theta(x, t)$$

Classical and quantum control

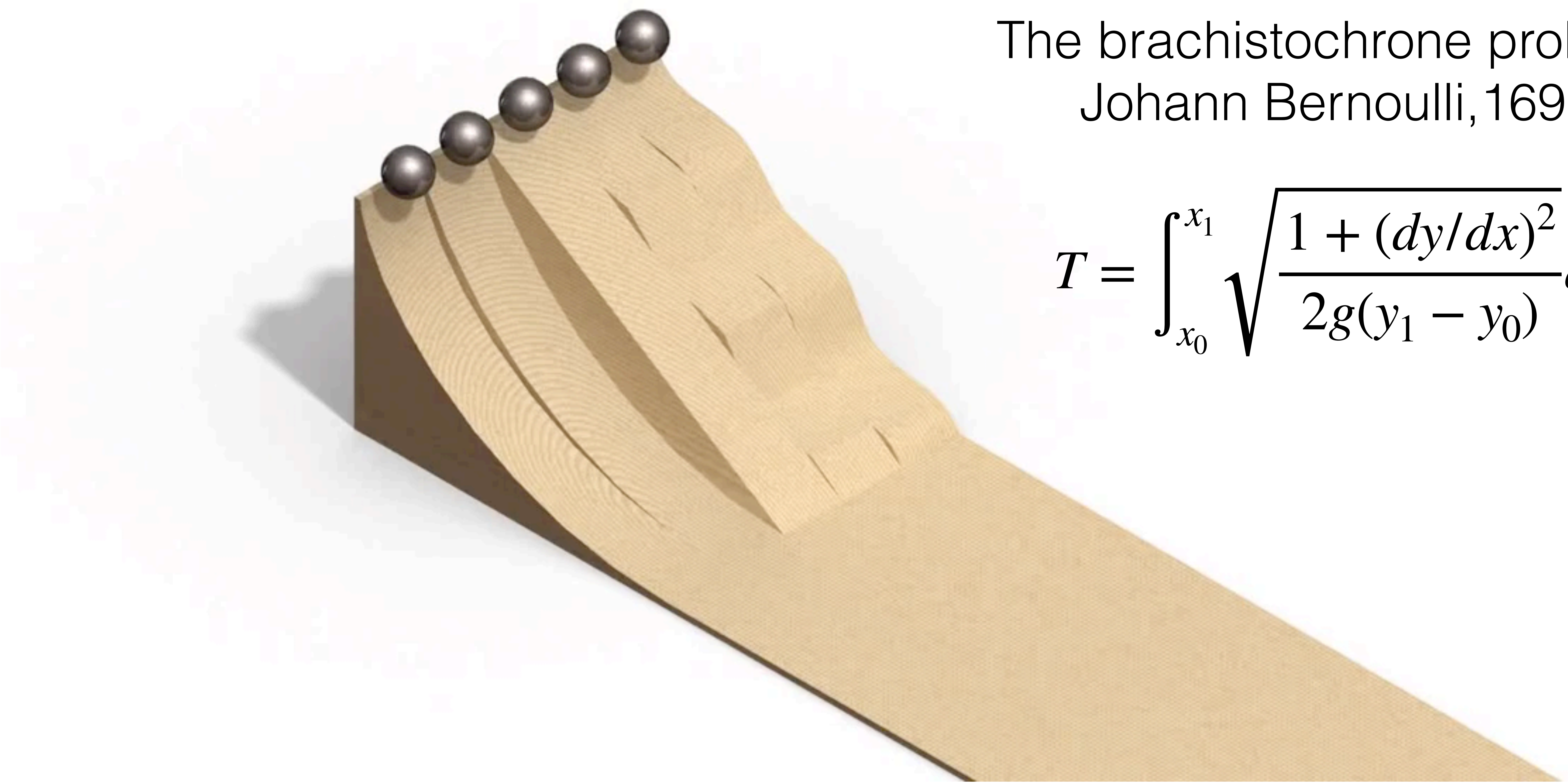
Principle of least actions



$$S = \int \mathscr{L}(q_\theta, \dot{q}_\theta, t)dt$$

Optics, (quantum) mechanics, field theory…

Differentiable functional optimization



The brachistochrone problem
Johann Bernoulli, 1696

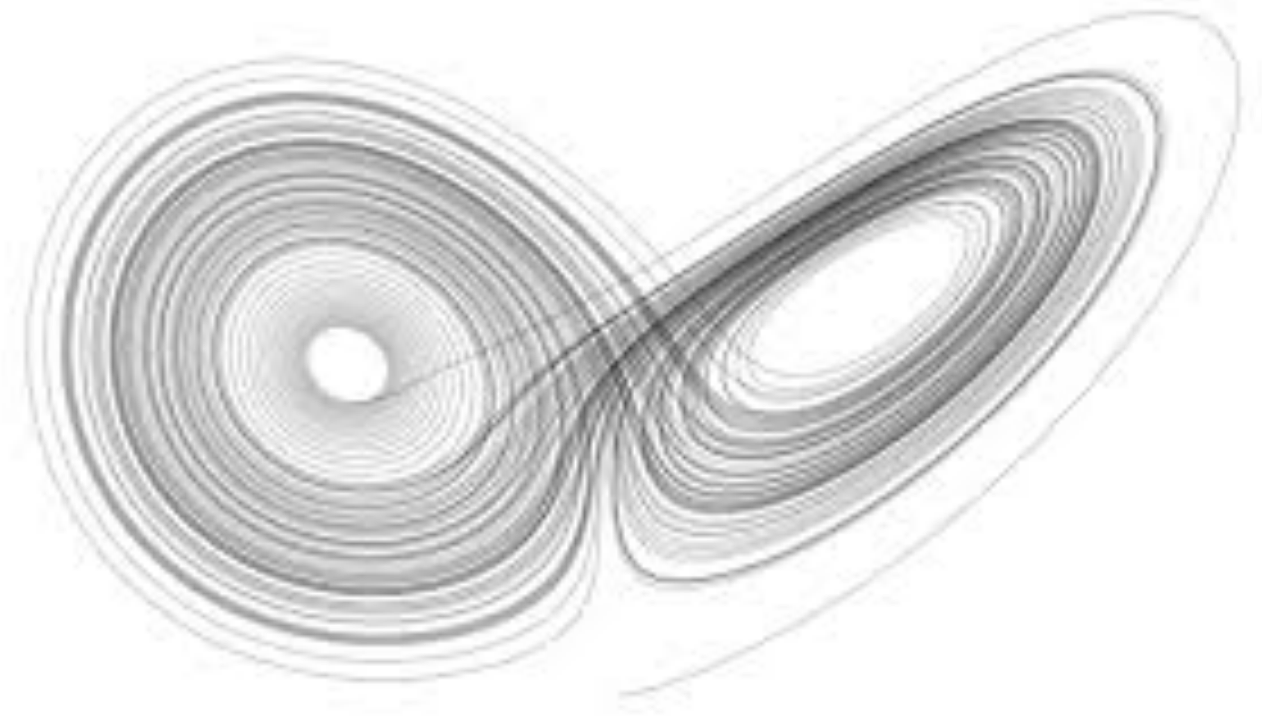
$$T = \int_{x_0}^{x_1} \sqrt{\frac{1 + (dy/dx)^2}{2g(y_1 - y_0)}} dx$$

https://github.com/QuantumBFS/SSSS/tree/master/1_deep_learning/brachistochrone

Differentiable ODE integrators

"Neural ODE" Chen et al, 1806.07366

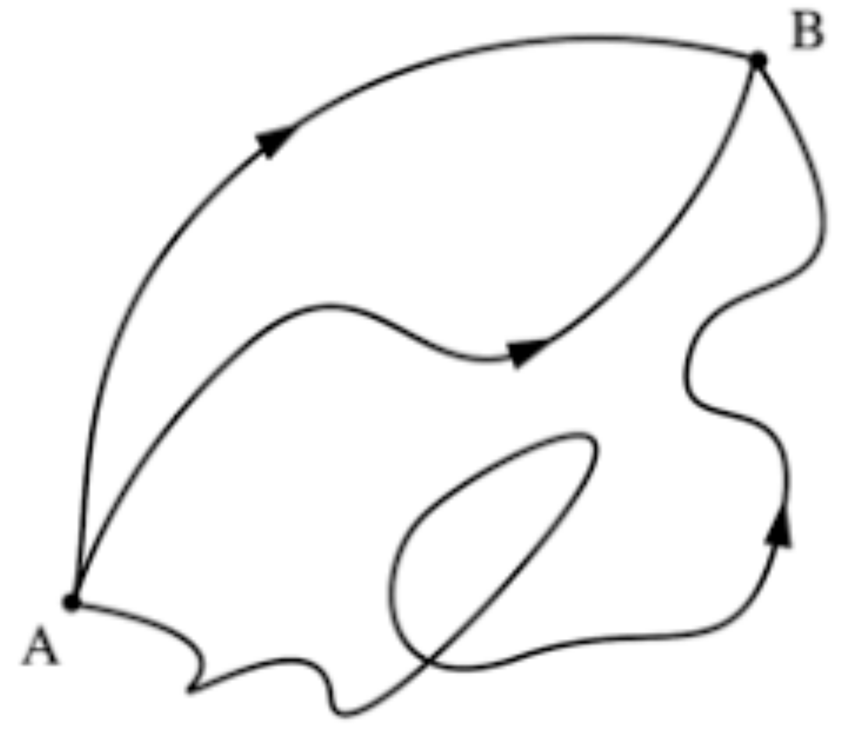
Dynamics systems



$$\frac{dx}{dt} = f_\theta(x, t)$$

Classical and quantum control

Principle of least actions

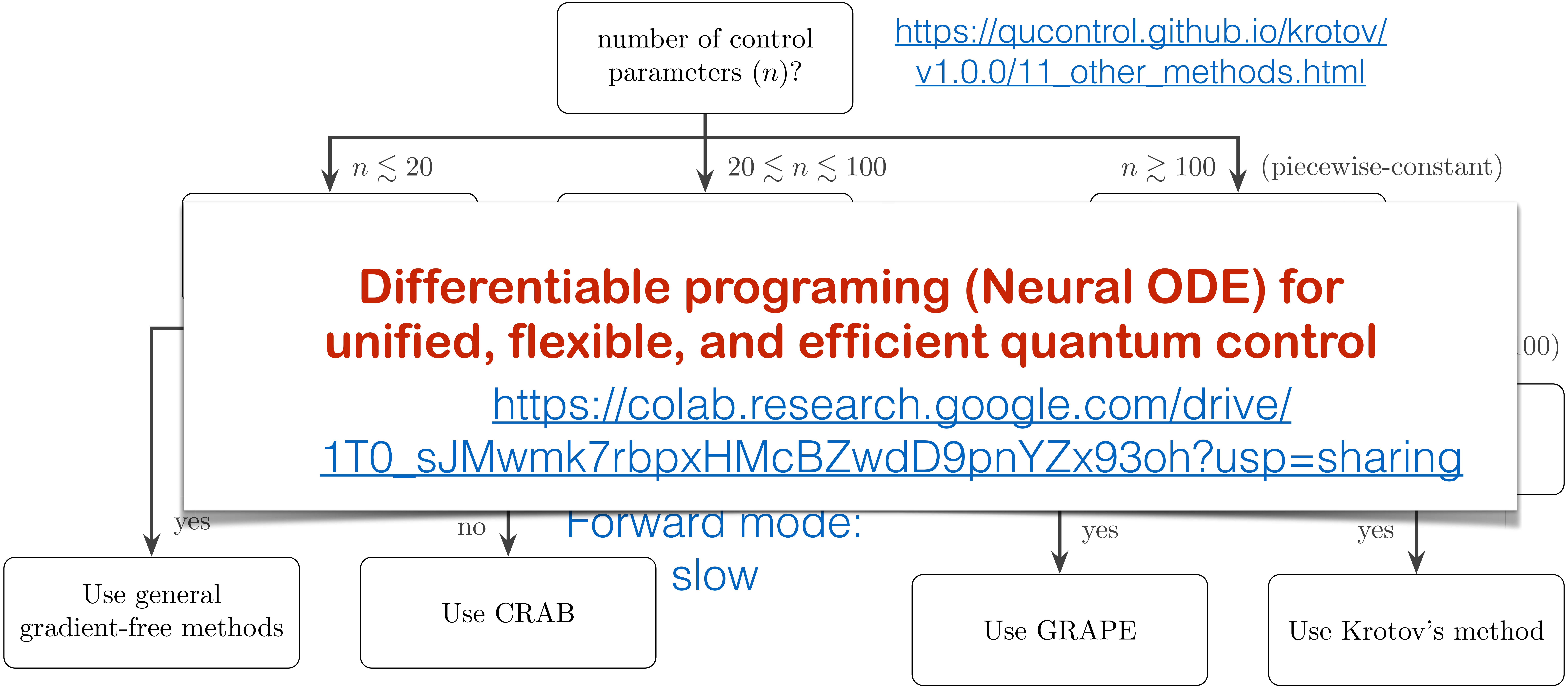


$$S = \int \mathscr{L}(q_\theta, \dot{q}_\theta, t)dt$$

Optics, (quantum) mechanics, field theory…

Quantum optimal control $i\frac{dU}{dt} = HU$

https://qucontrol.github.io/krotov/v1.0.0/11_other_methods.html



Differentiable programing (Neural ODE) for unified, flexible, and efficient quantum control

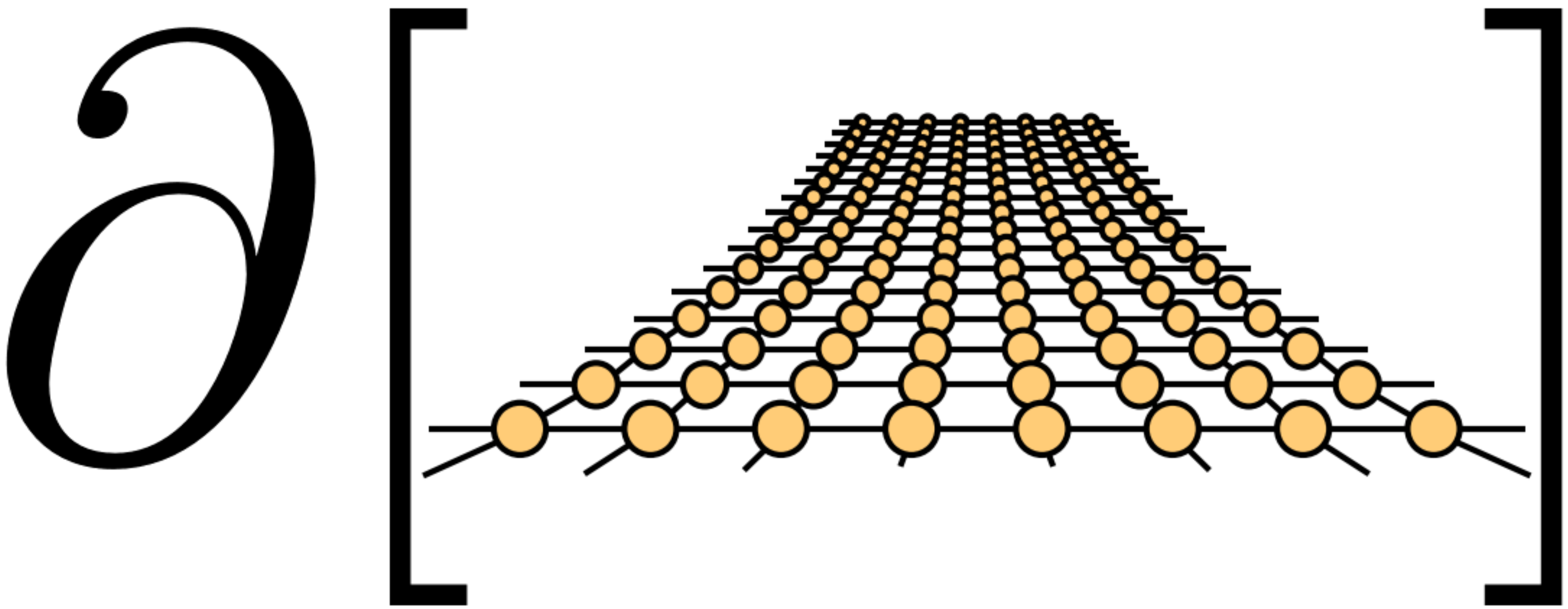
https://colab.research.google.com/drive/1T0_sJMwmk7rbpxHMcBZwdD9pnYZx93oh?usp=sharing

Forward mode: slow

No gradient: not scalable

Reverse mode w/ discretize steps: piesewise-constant assumption

Differentiable Programming Tensor Networks

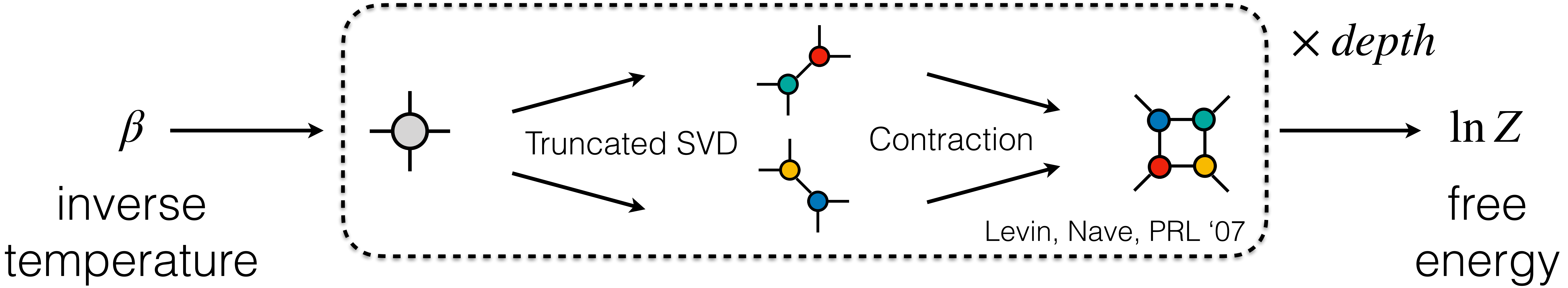


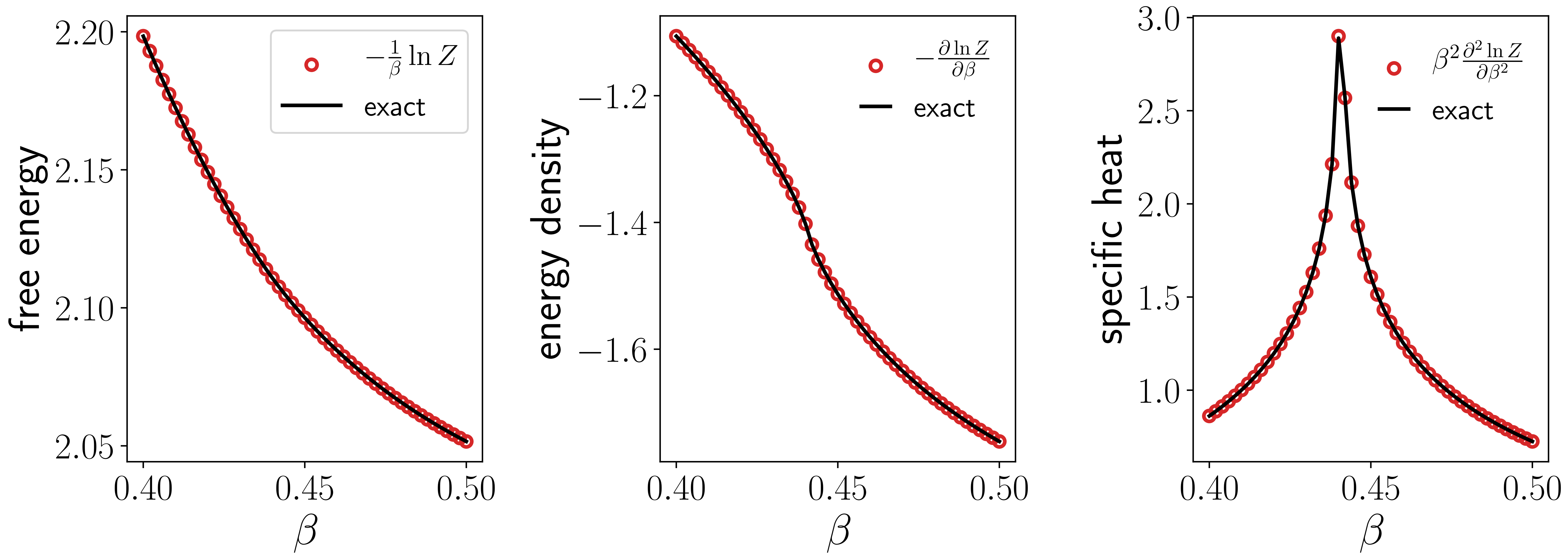
Liao, Liu, LW, Xiang, 1903.09650, PRX '19

https://github.com/wangleiphy/tensorgrad

Differentiate through tensor renormalization group

Computation graph





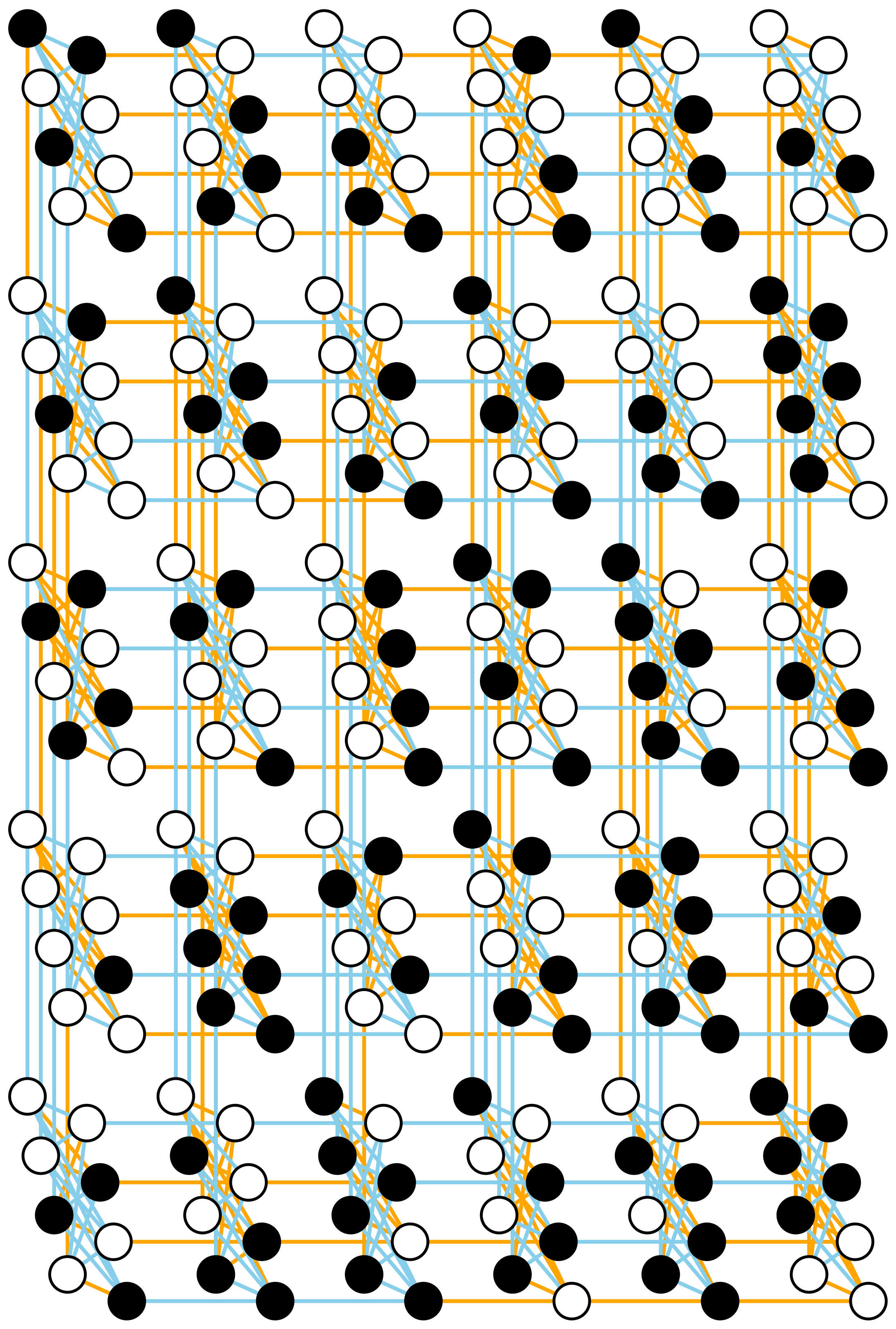
Compute physical observables as gradient of tensor network contraction

Exercise

- Implement a JAX or julia version of this

- Note: since we only compute gradient with respect a scalar β, forward model AD suffices.

Differentiable spin glass solver



couplings & fields → tensor network contraction → optimal energy

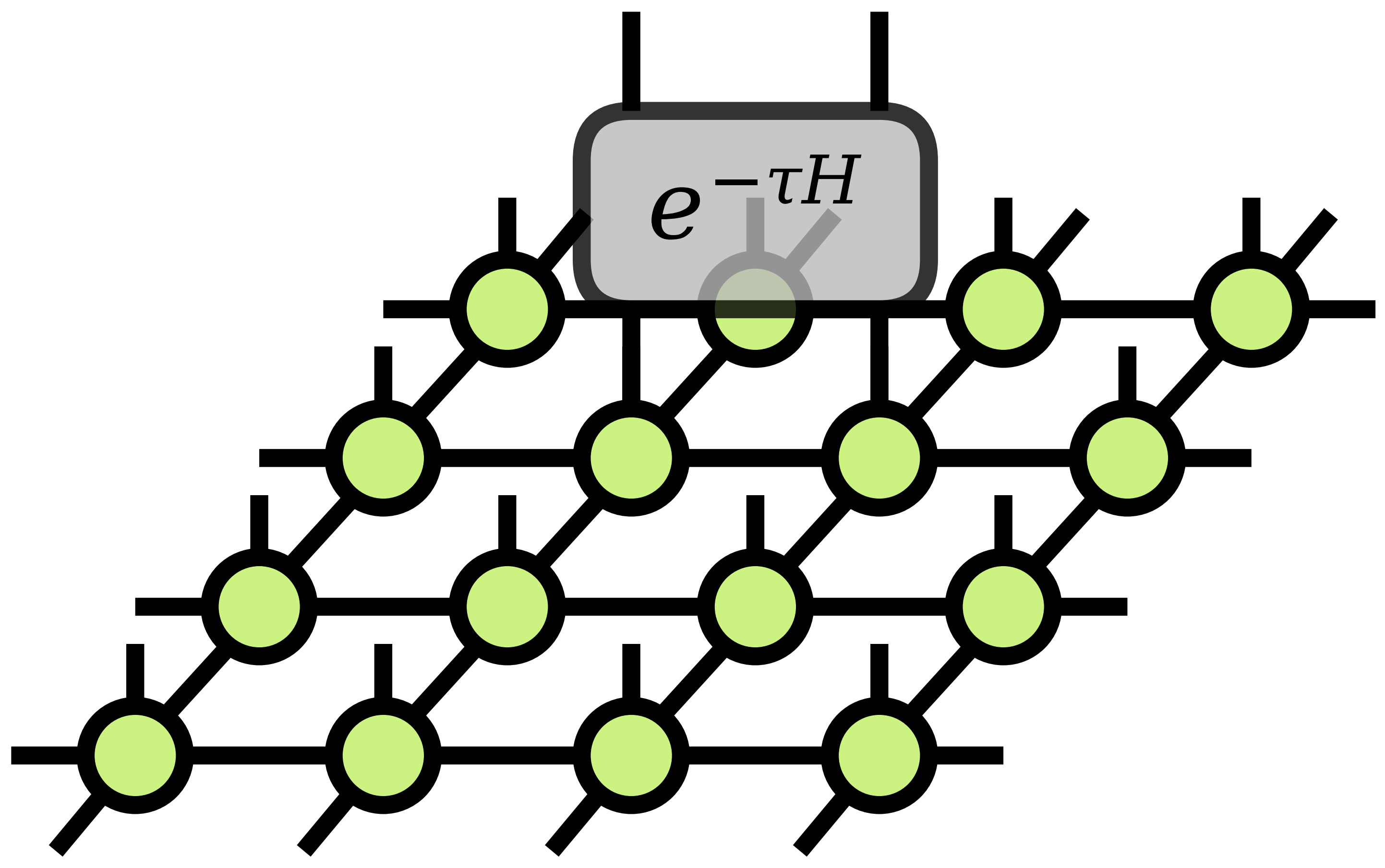
$$\text{optimal configuration} = \frac{\partial[\,\text{optimal energy}\,]}{\partial[\,\text{field}\,]}$$

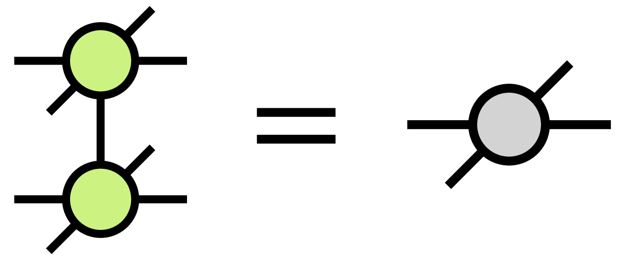
Liu, LW, Zhang, 2008.06888, PRL '21

https://github.com/TensorBFS/TropicalTensors.jl

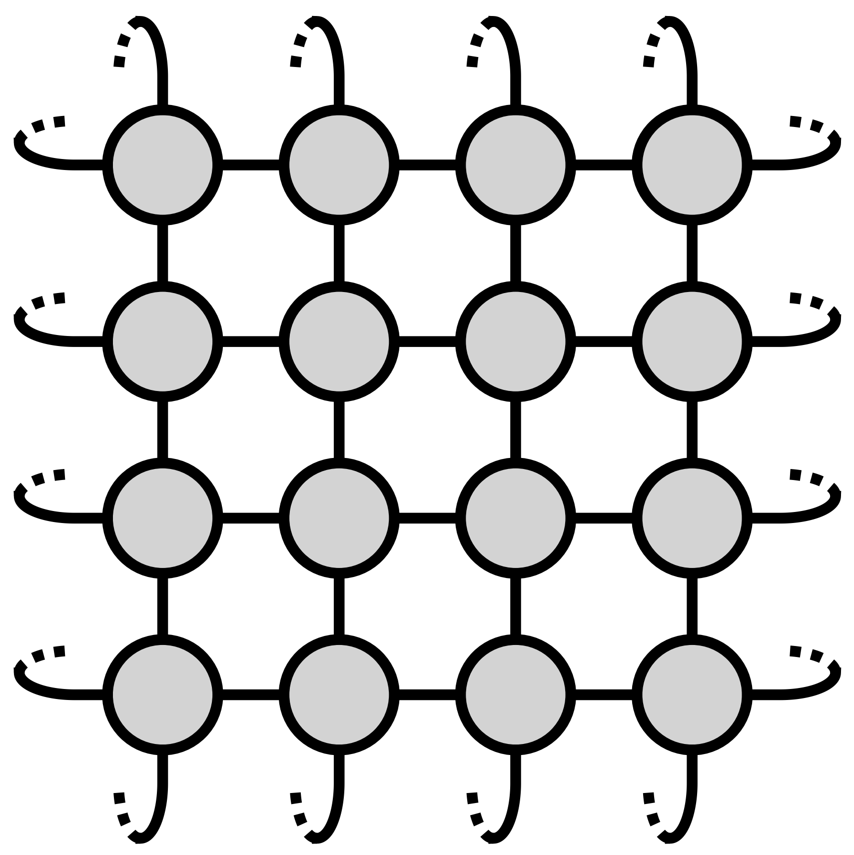
Tensor network quantum states

Optimization





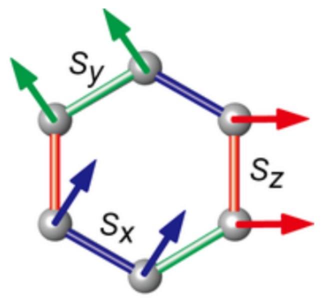
Contraction



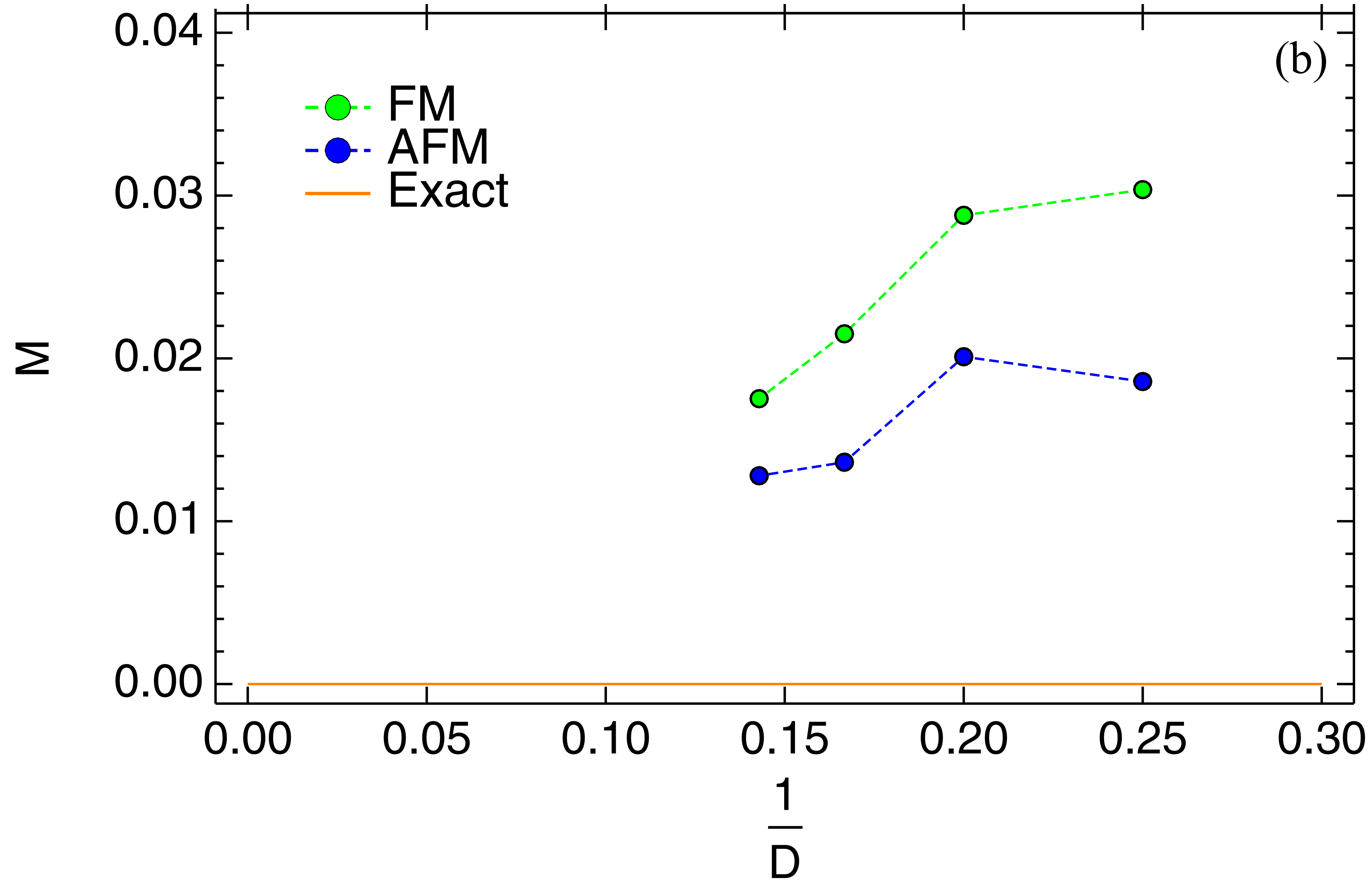
- Trotterized imaginary-time projection
- Update schemes: “simple”, “full” “cluster”, “faster full”…

- #P hard in general
- Approximated schemes: TRG, Boundary MPS, Corner transfer matrix RG

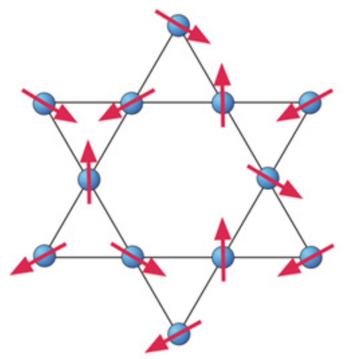
Representation v.s. Optimization: an eternal problem



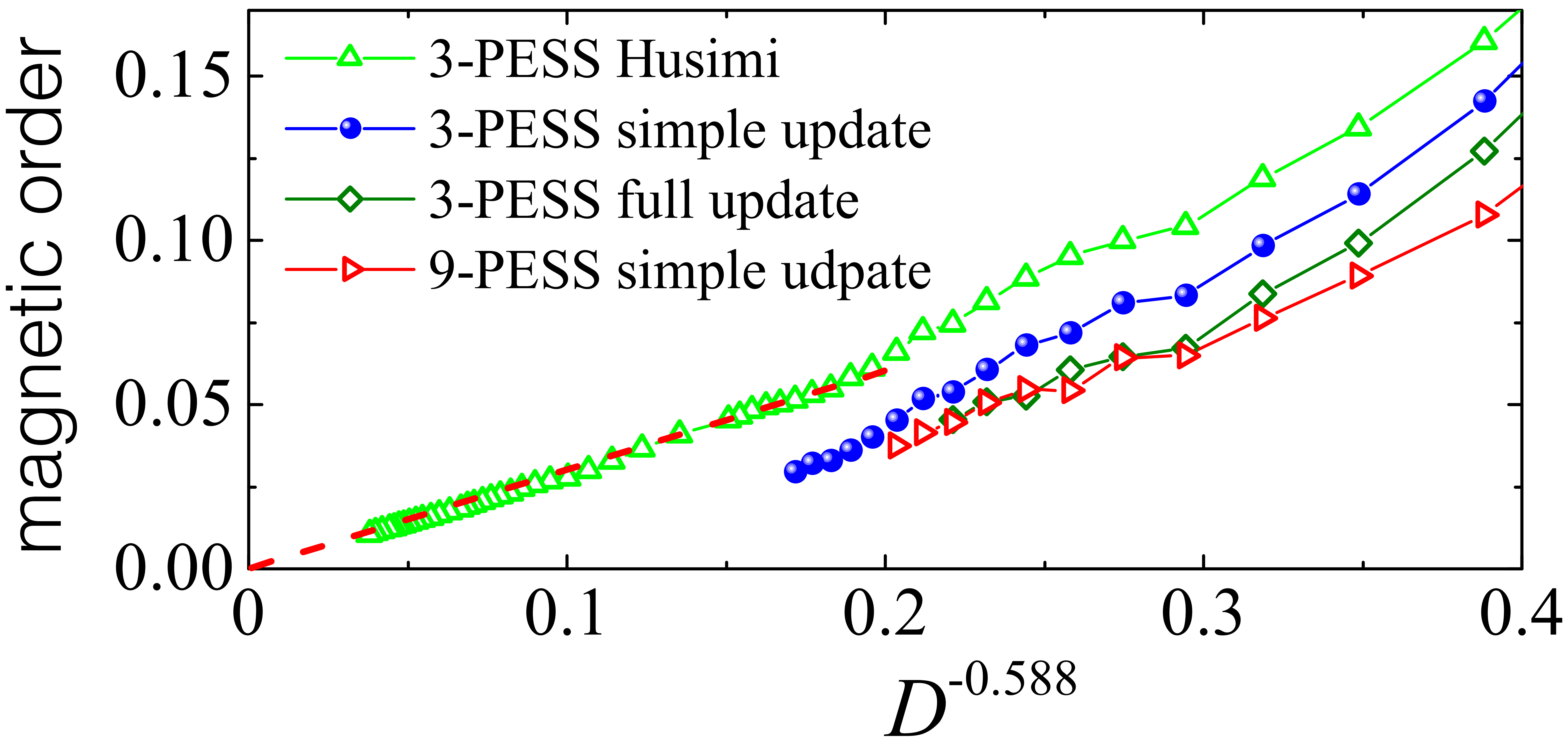
Kitaev Honeycomb model



Osorio, Corboz, Troyer, PRB ‘14



Kagome Heisenberg model

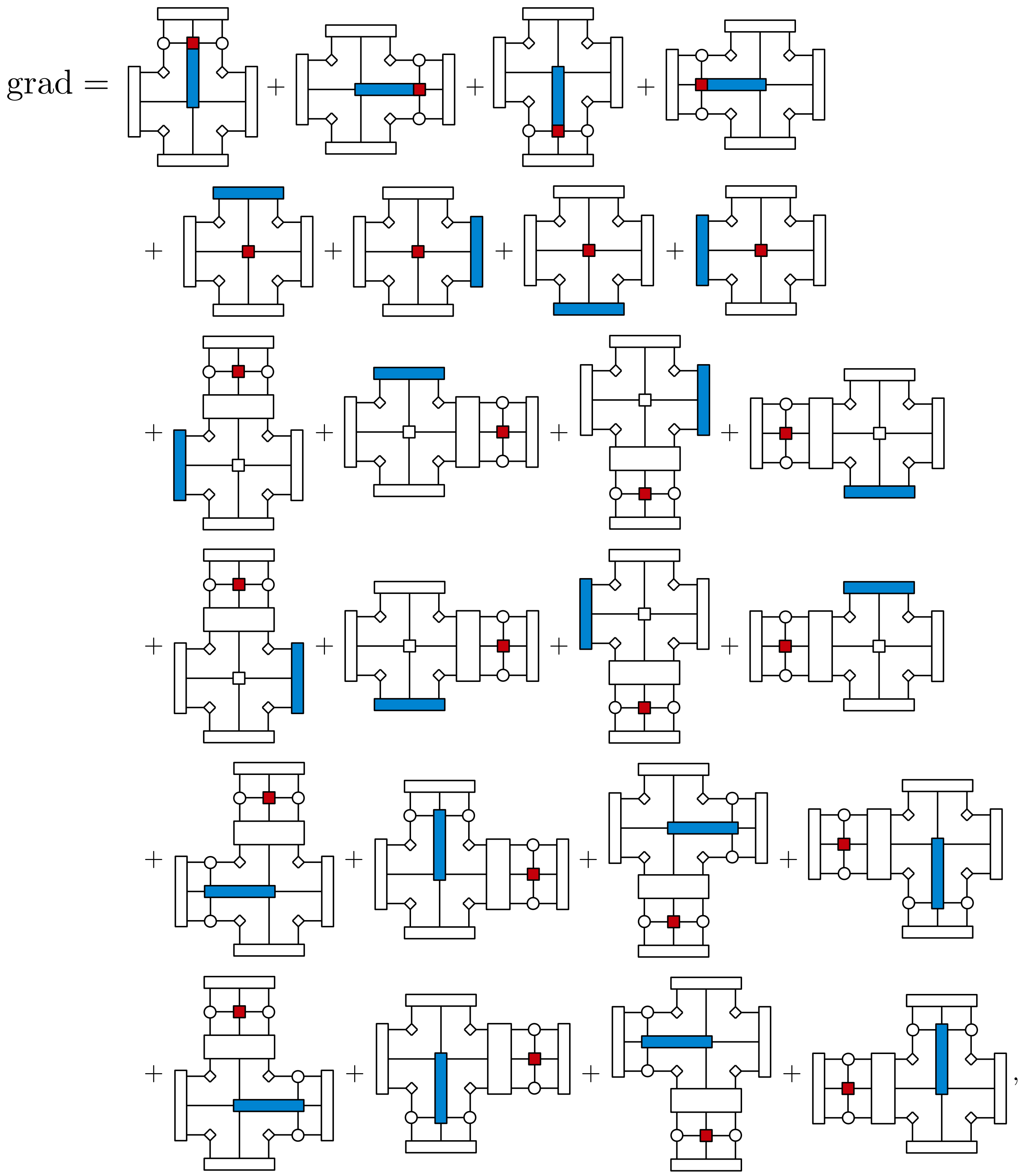


Liao et al, PRL ‘17

One typically finds ordered states
and tries hard to push up the bond dimensions

Differentiable tensor optimization

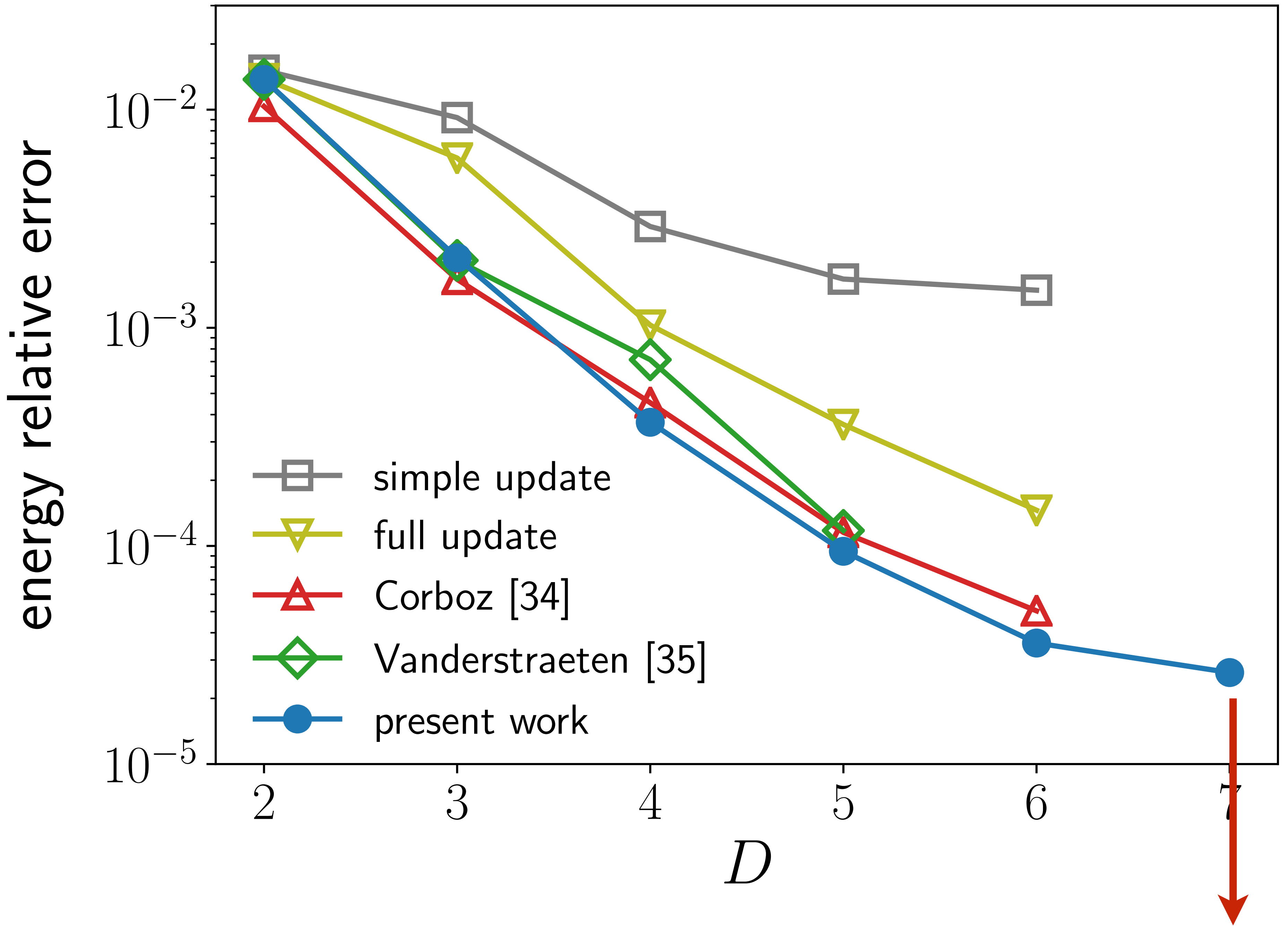
before…



Vanderstraeten et al, PRB '16

now, w/ differentiable programming

Liao, Liu, LW, Xiang, PRX '19

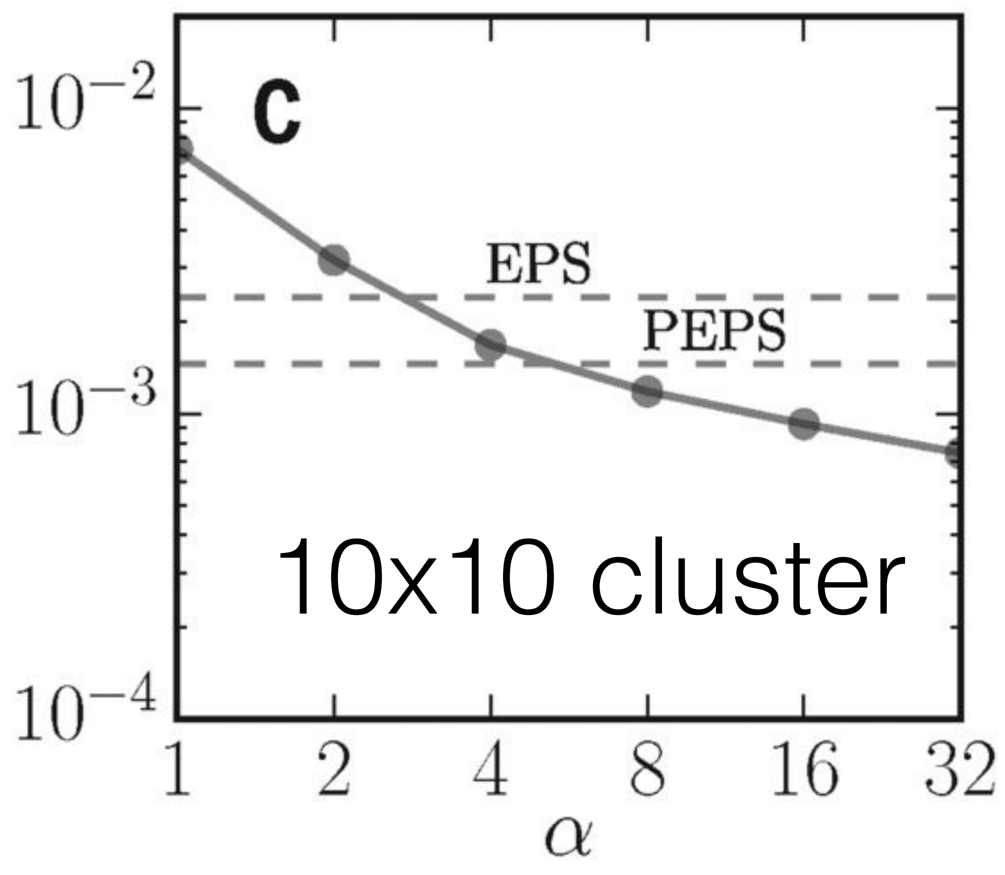


Lowest variational energy

https://github.com/wangleiphy/tensorgrad 1 GPU (Nvidia P100) week

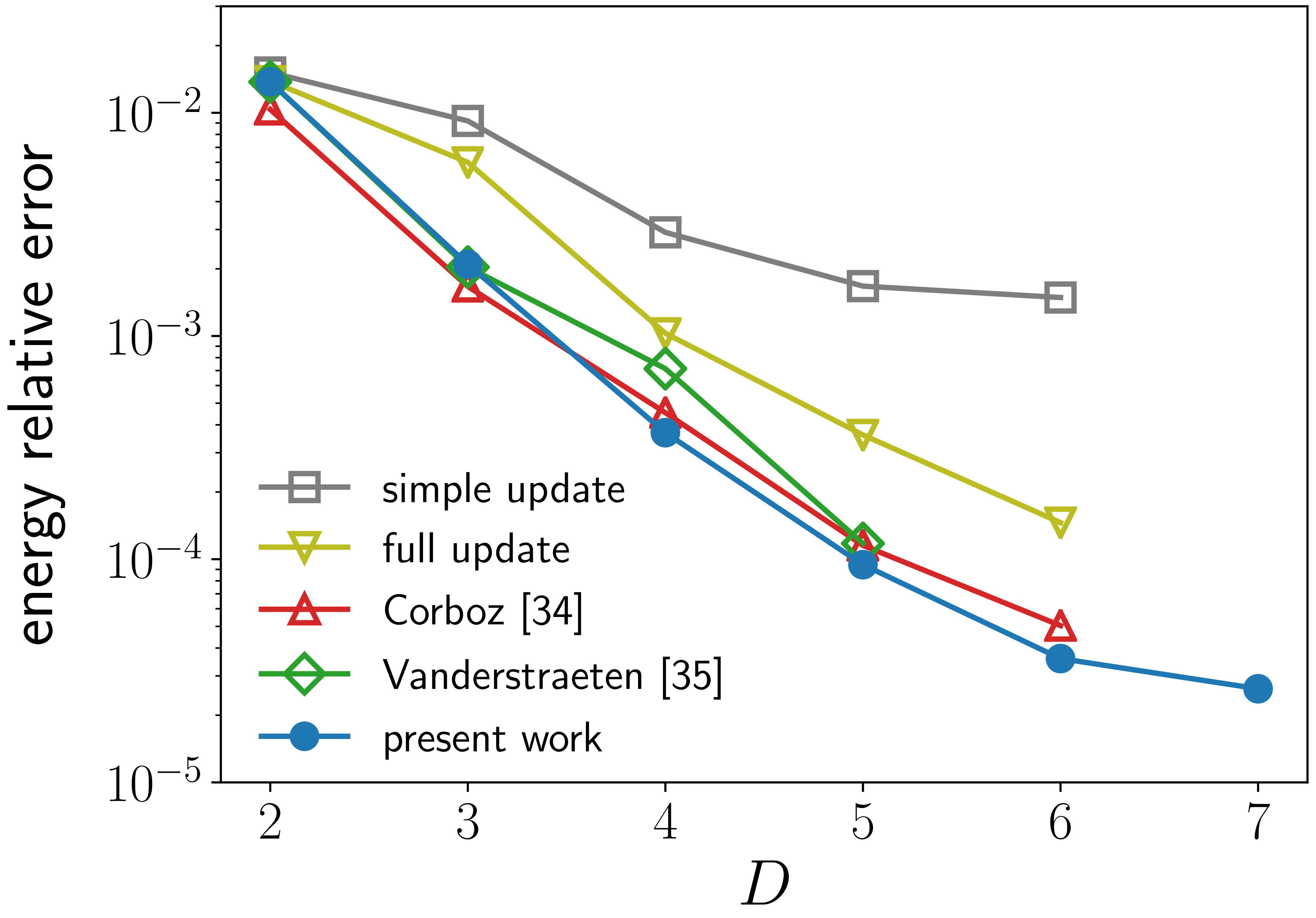
Differentiable tensor optimization

Finite size
Neural network



Carleo & Troyer, Science ‘17

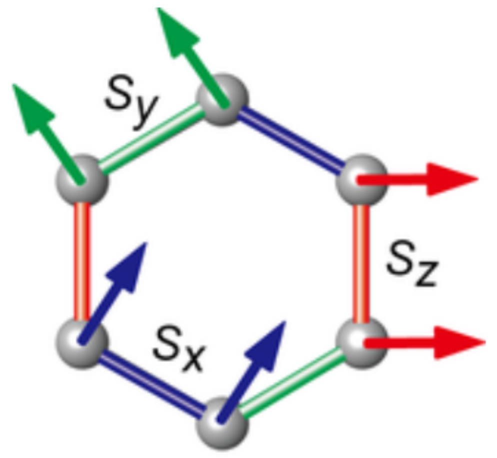
Infinite size
Tensor network



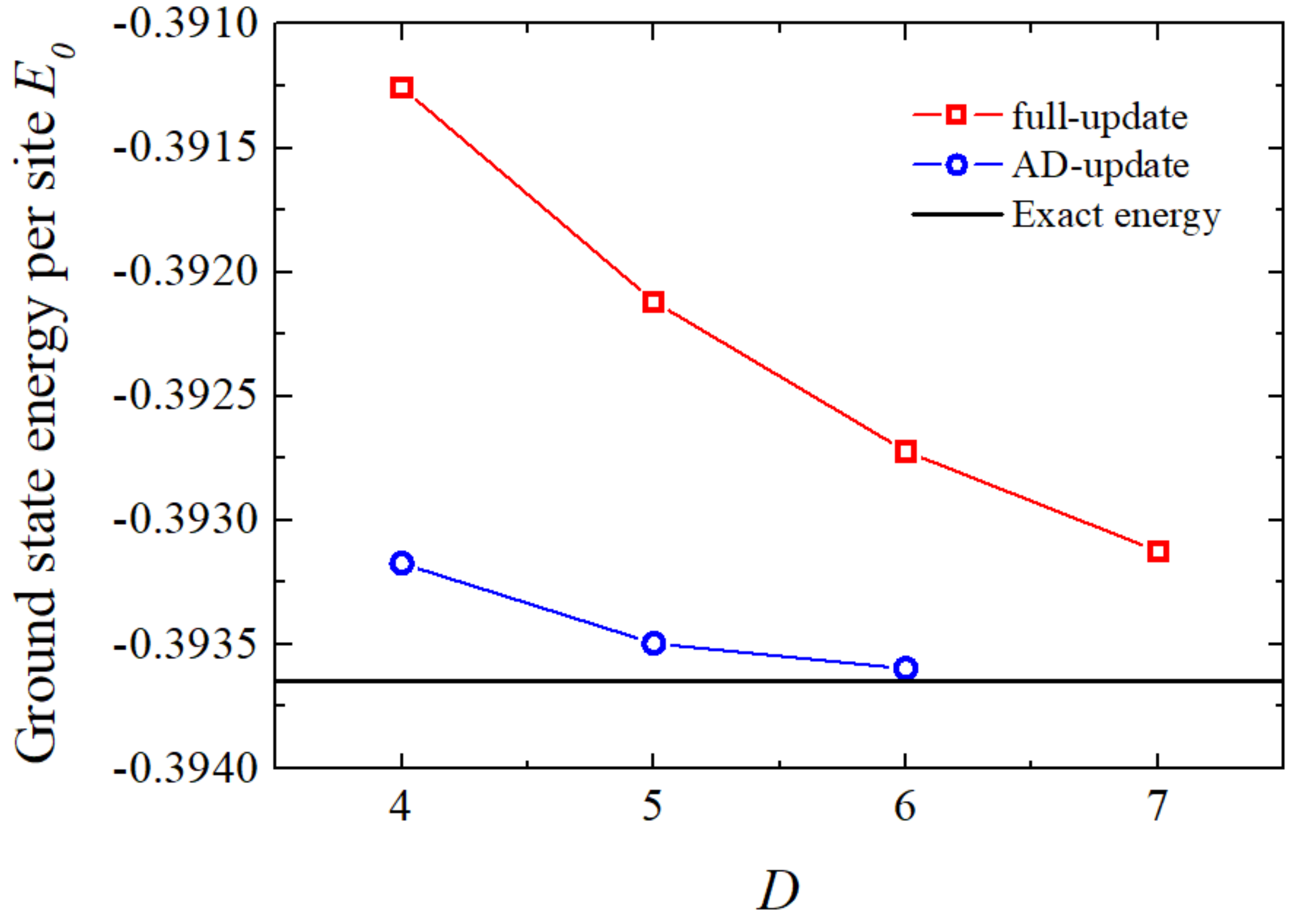
Liao, Liu, LW, Xiang, PRX ‘19

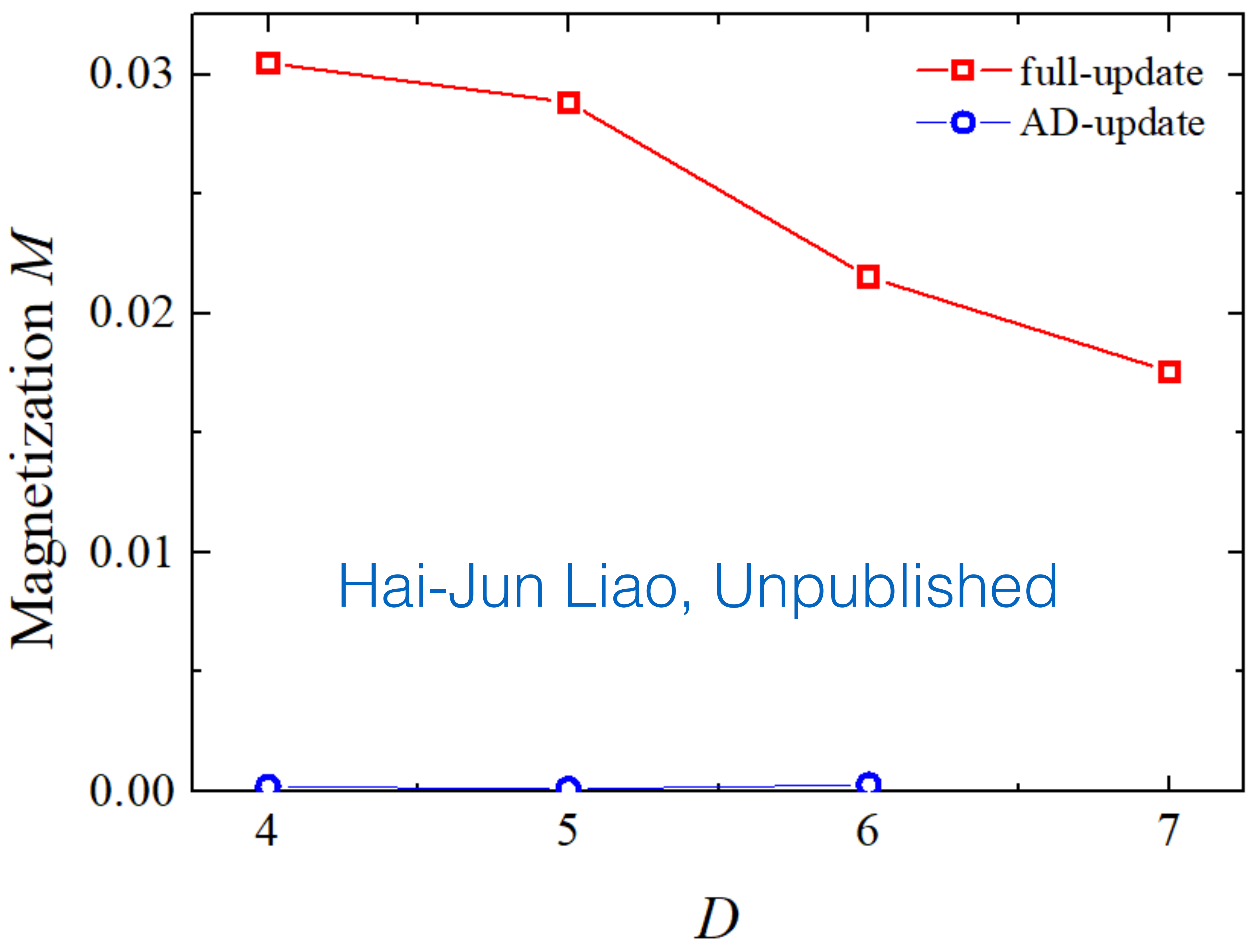
Further progress for challenging physical problems: frustrated magnets, fermions, thermodynamics …

Chen et al, ‘19
Xie et al, ’20
Tang et al ’20
…



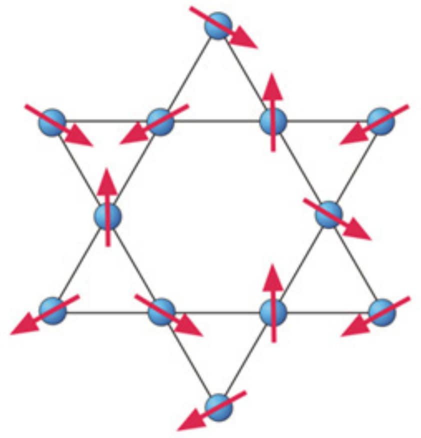
Kitaev honeycomb model



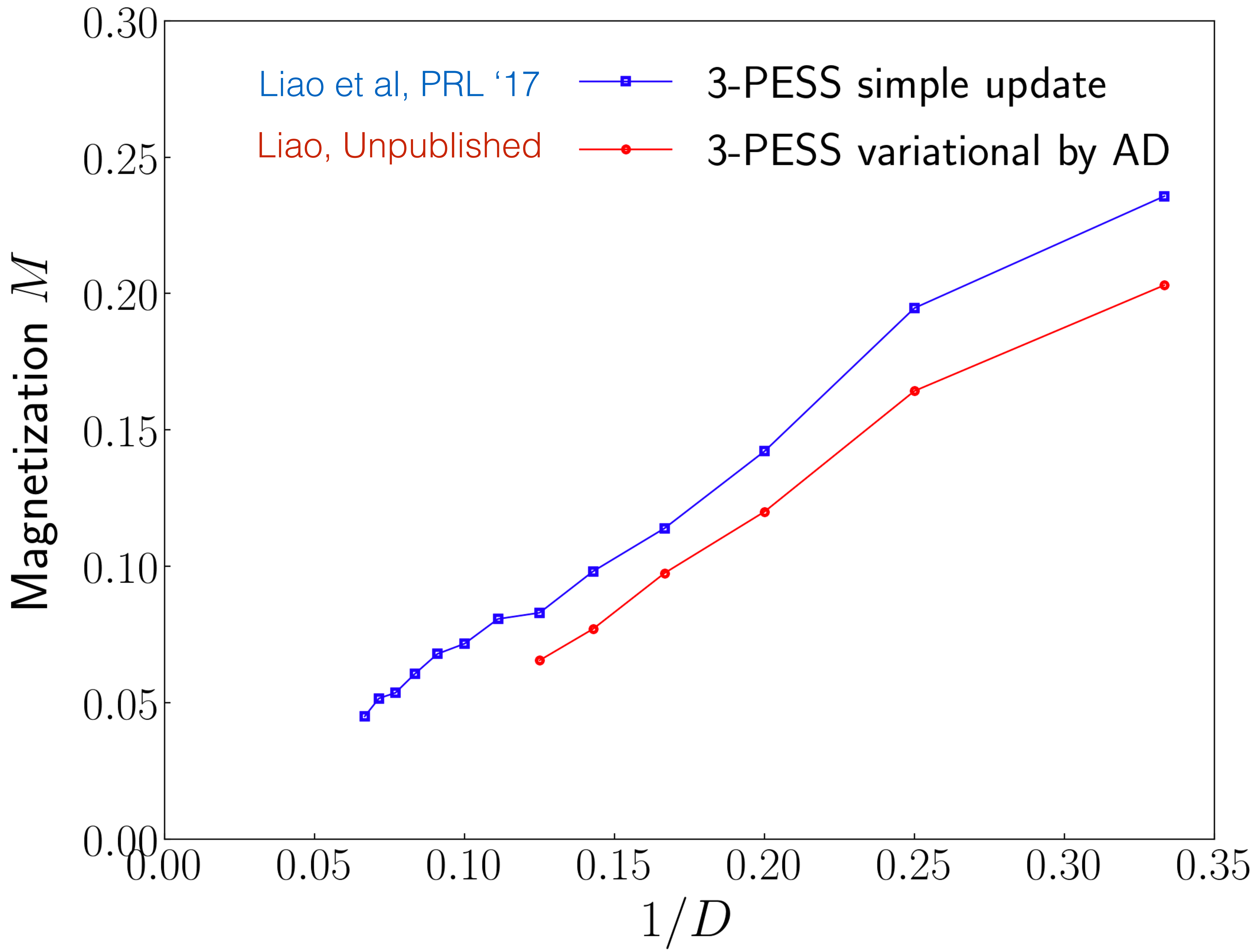


Reaches lower energy with fewer variational parameters
And substantially reduced magnetic order

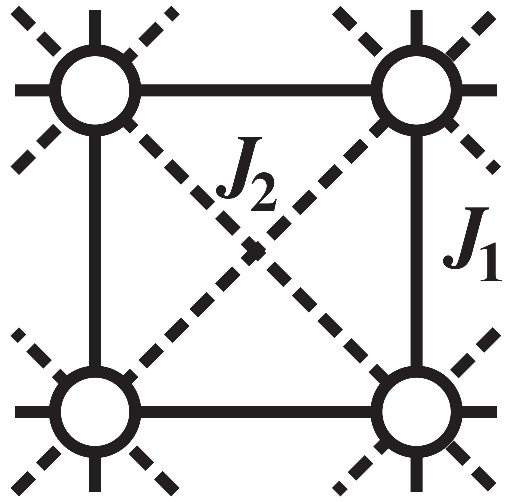
c.f. analytically constructed iPEPS, Lee et al, PRL '19



Kagome Heisenberg model

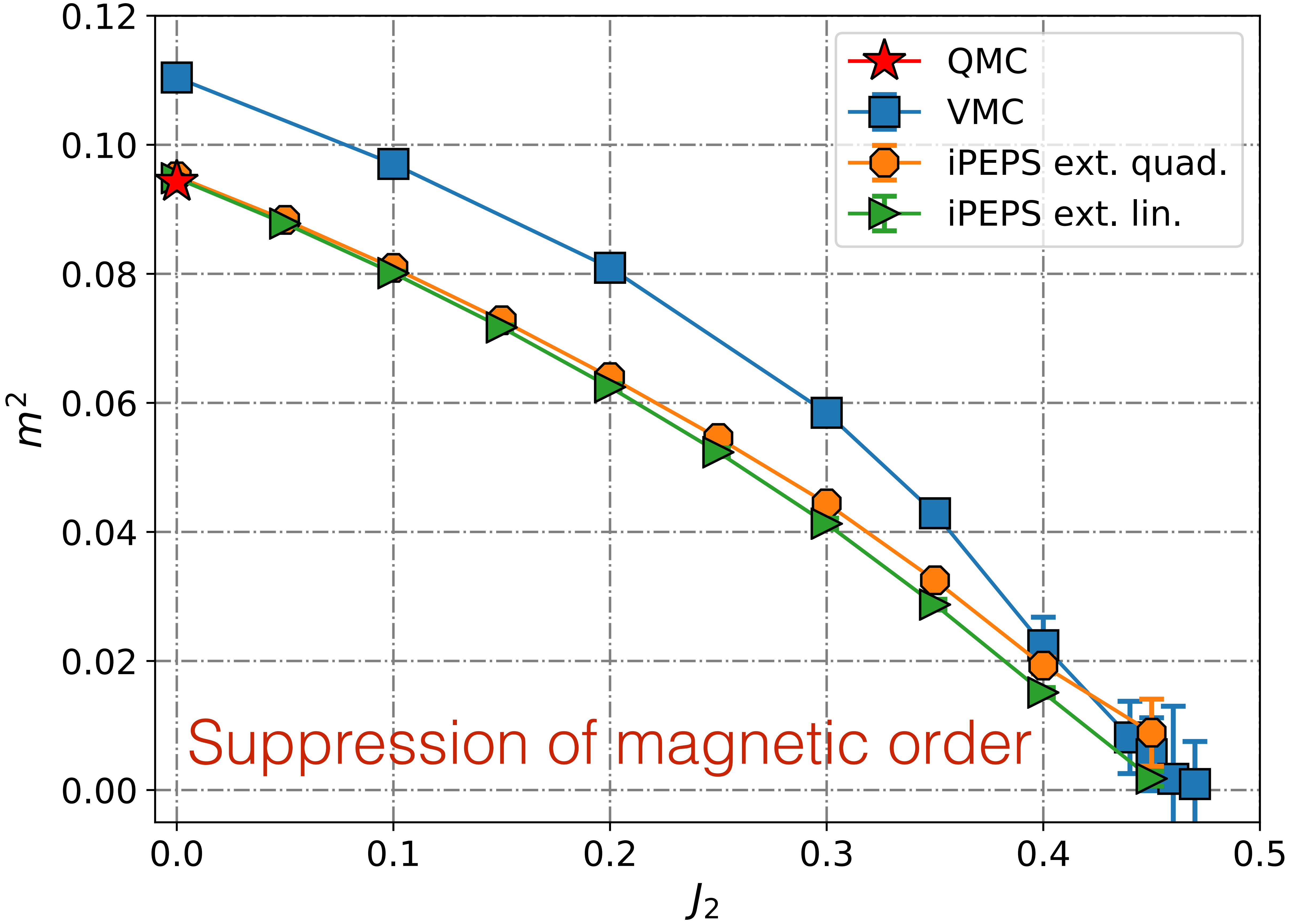


Lower energy, reduced magnetic order



Square lattice J_1-J_2 model

Hasik, Poiblanc, Becca, 2009.02313



U(1) symmetric tensor + correlation length extrapolation + automatic differentiation

See also DMRG level-spectroscopy (Wang, Sandvik '18)
RBM*pair product state VMC (Nomura, Imada '20)
finite PEPS VMC (Liu et al '20)
+ many many others

Nuts and Bolts

- Numerical stable backward through SVD

$$A \to UDV^T \qquad\qquad \bar{A} \xleftarrow{?} \bar{U}, \bar{D}, \bar{V}$$

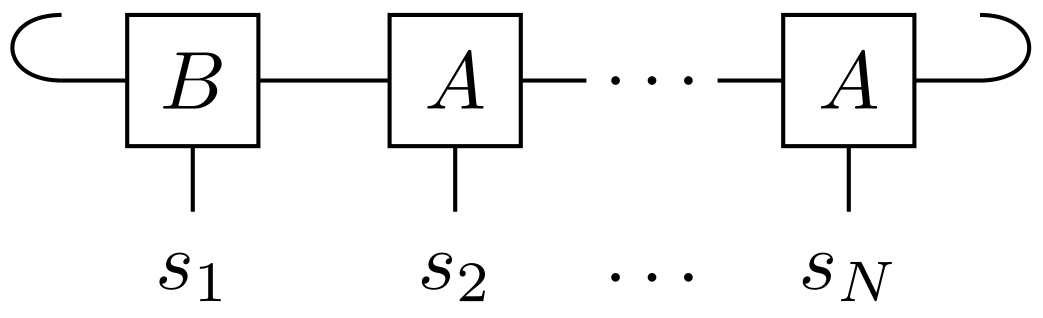
- Reduce memory via checkpointing or exploiting RG fixed point property

$$T_{i+1} = f(T_i, \theta) \xrightarrow{\textbf{Iterate}} T^* = f(T^*, \theta) \qquad\qquad \bar{\theta} = \overline{T^*}\left[1 - \frac{\partial f}{\partial T^*}\right]^{-1} \frac{\partial f}{\partial \theta}$$

Liao, Liu, LW, Xiang, PRX ‘19

More applications of differentiable tensor networks

$$|\Phi_k(B)\rangle = \sum_{j=0}^{N-1} \mathrm{e}^{-ikj}\hat{T}^j$$

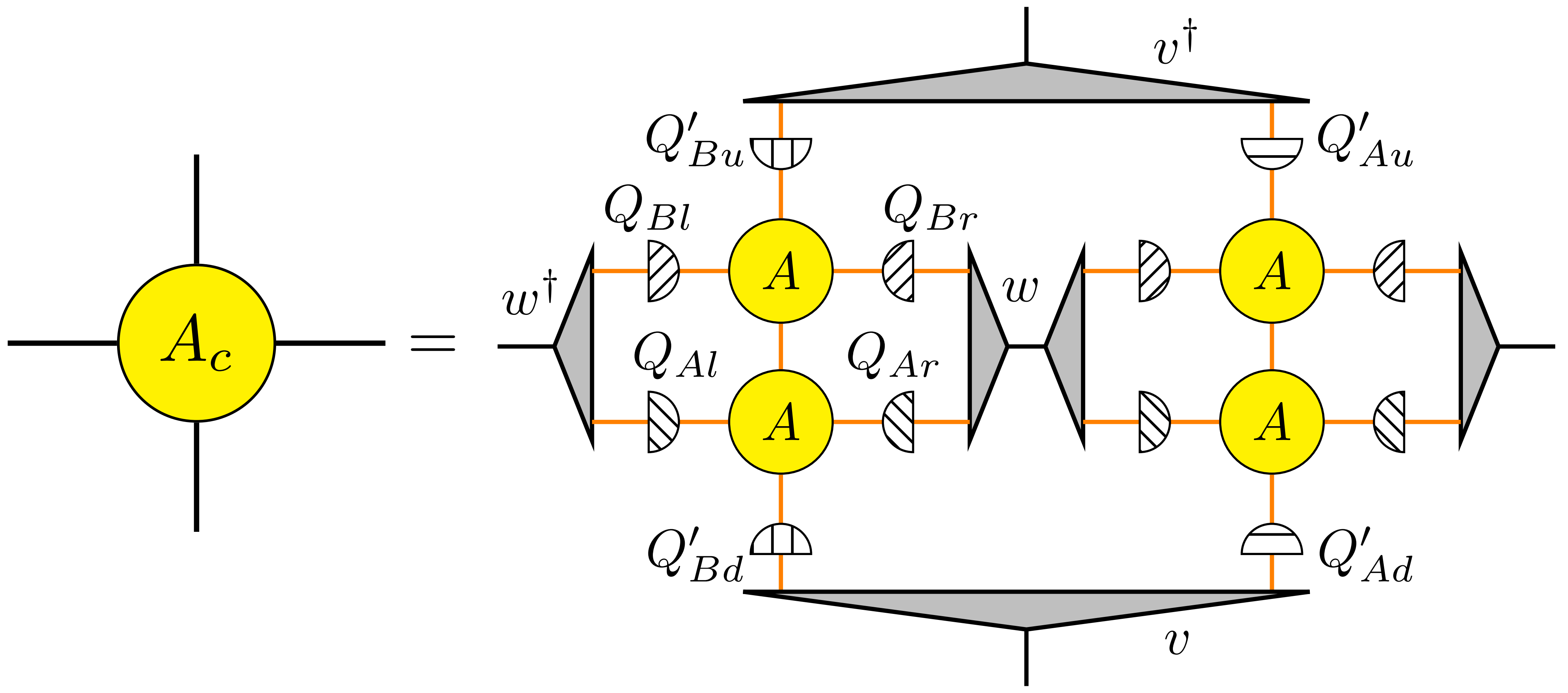


$$= \left.\frac{\partial|G(\lambda)\rangle}{\partial\lambda}\right|_{\lambda=0}$$

Generating function for
tensor diagrammatic summation
Tu et al, 2101.03935

$$\frac{\partial}{\partial B^\dagger}\left[\langle\Phi(B)_{\boldsymbol{k}}|\,\mathcal{H}\,|\Phi(B)_{\boldsymbol{k}}\rangle - \omega_{\boldsymbol{k}}(\langle\Phi(B)_{\boldsymbol{k}}\,|\Phi(B)_{\boldsymbol{k}}\rangle - 1)\right] = 0.$$

iPEPS excitations
Ponsioen et al, 2107.03399



Scaling dimension from
tensor RG fixed point
Lyu et al, 2102.08136

Gradients might appear more often than you've thought!

Primitives of differentiable Tensor networks, DFT, and scientific computing

Eigensolver/SVD

AD of complex-valued SVD, Wan and Zhang, 1909.02659

Degenerated eigenvalues: https://github.com/google/jax/issues/669

Dominant or truncated Eigensolver/SVD

https://math.mit.edu/~stevenj/18.336/adjoint.pdf

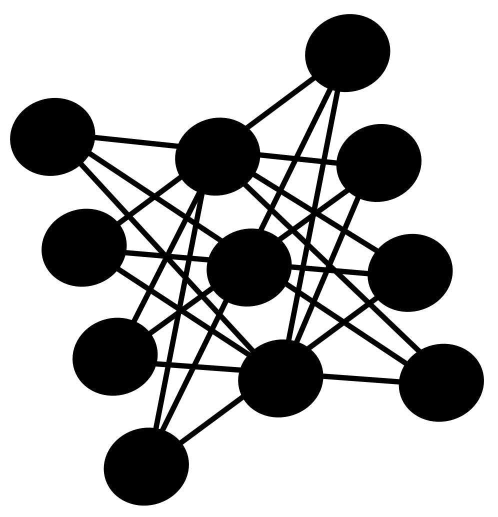
https://buwantaiji.github.io/2020/01/AD-of-truncated-SVD/

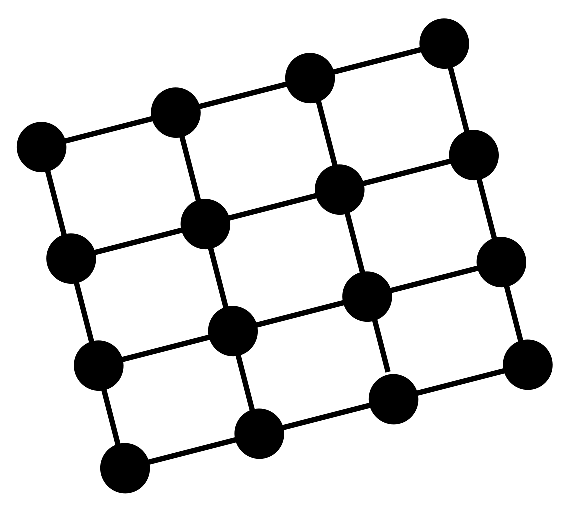
Xie, Liu, LW, PRB '20

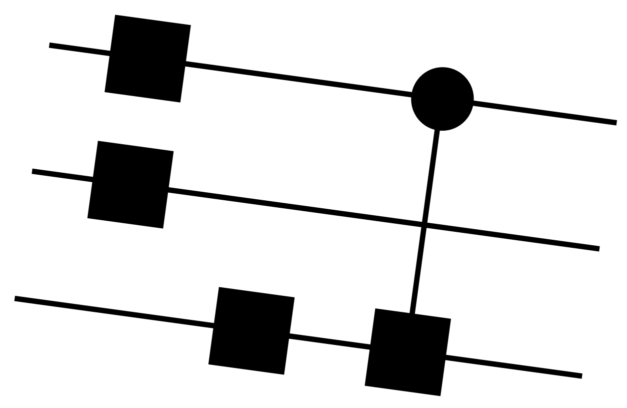
Fixed point iteration

http://implicit-layers-tutorial.org/implicit_functions/

Differentiable Programming Quantum Circuits

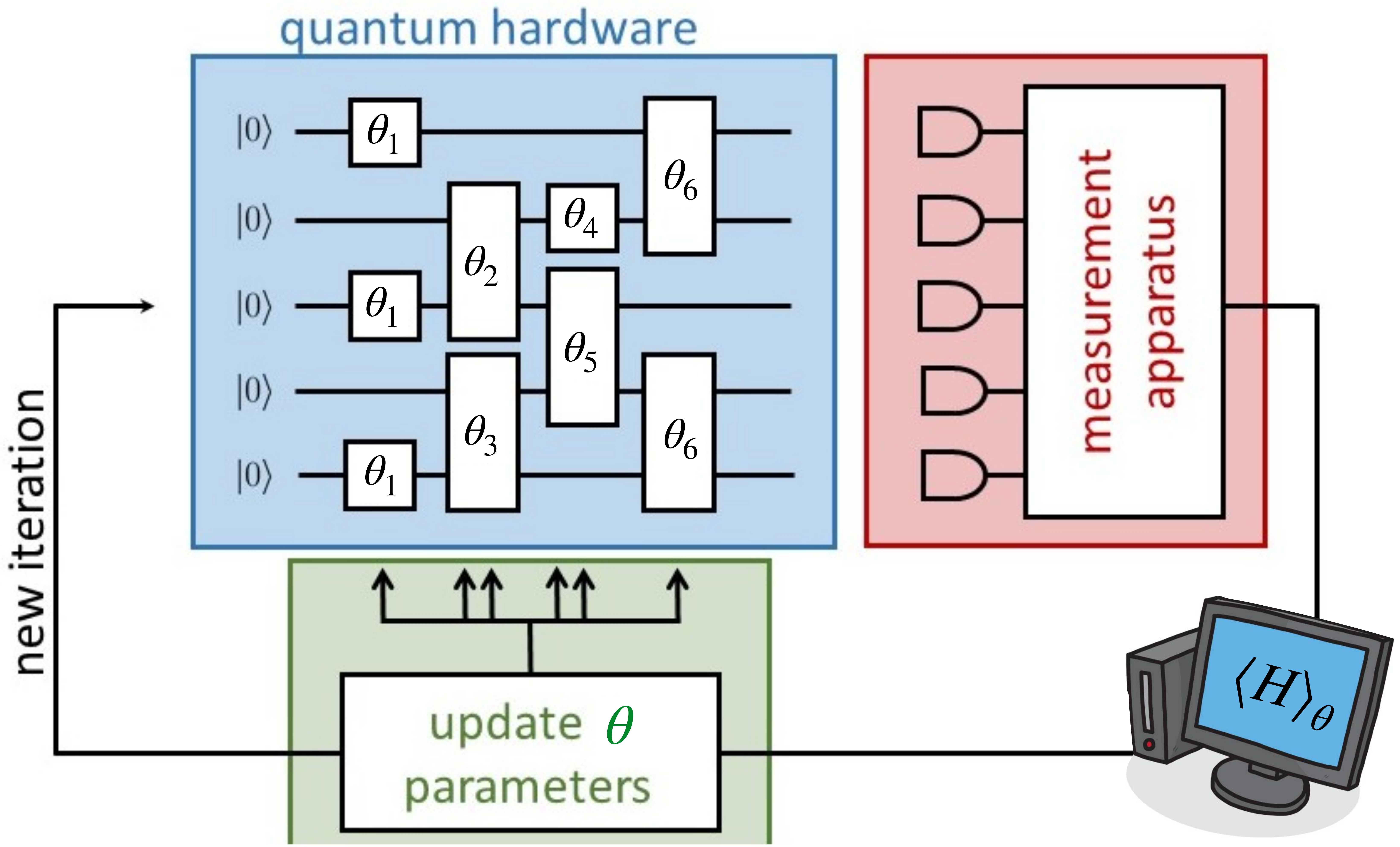






graphical models — tensor networks — quantum circuits

Variational quantum algorithms

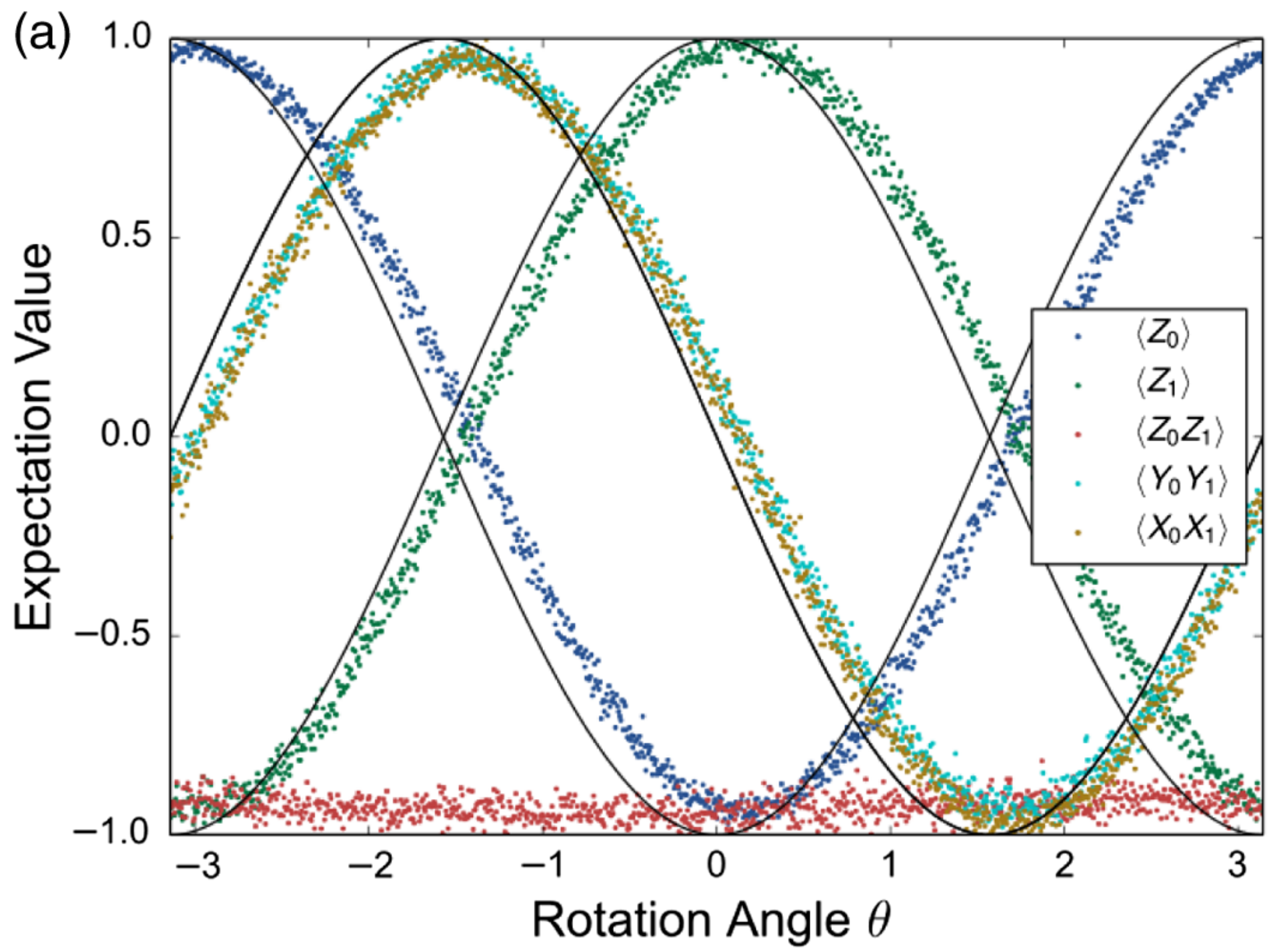


Peruzzo et al,
Nat. Comm. '13

Quantum circuit as a variational ansatz

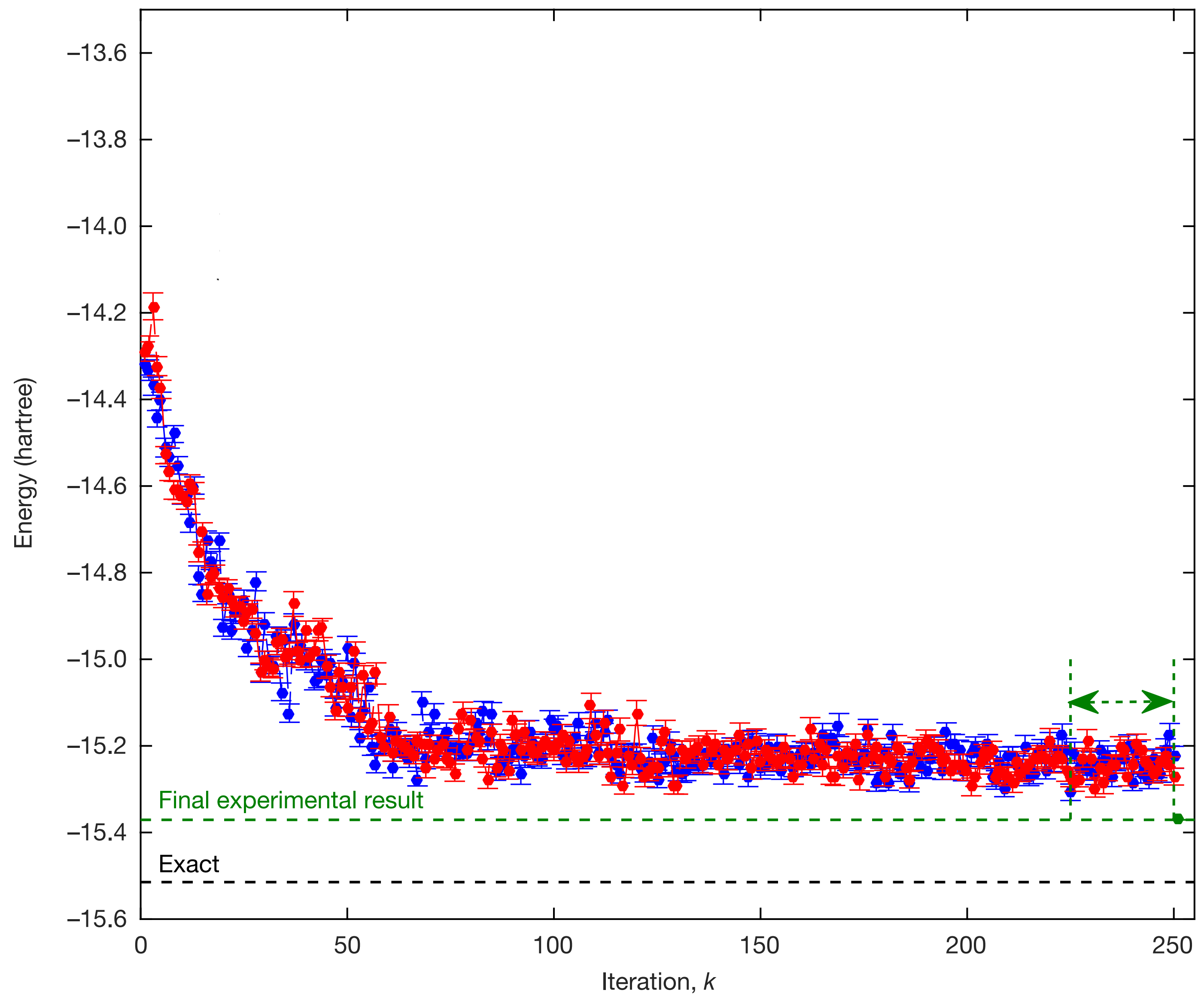
Optimize variational quantum circuits

Scan the single variational parameter



Google PRX '16

Stochastic perturbation of 30 variational parameters

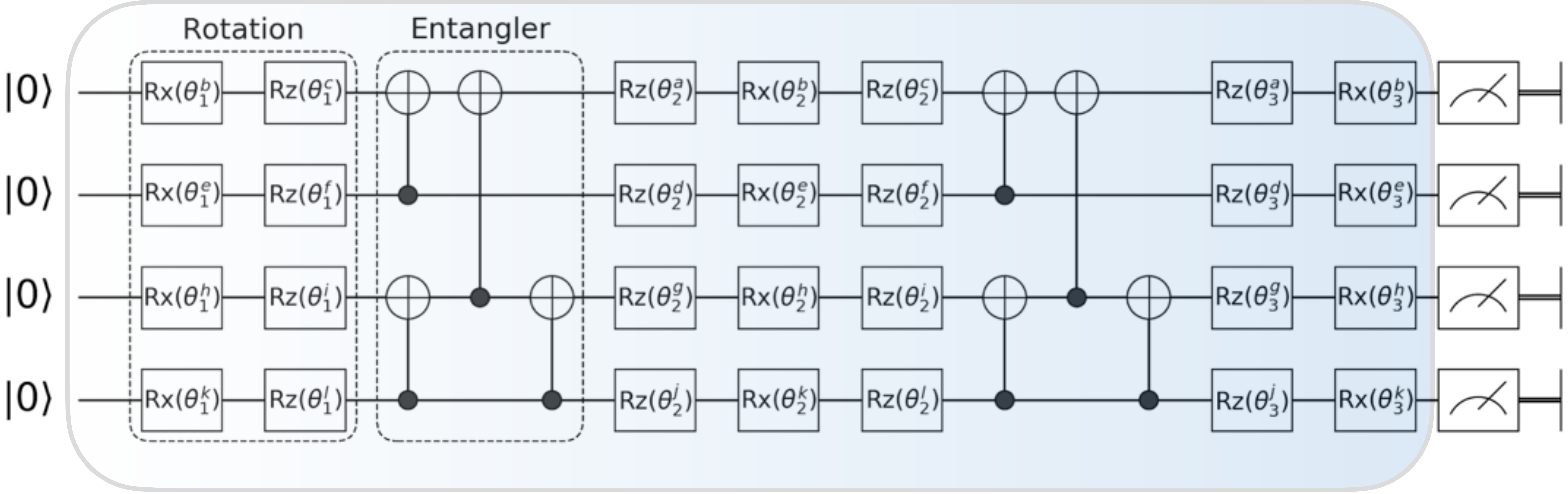


IBM Nature '17

Optimization with analytical gradient is essential for higher dimensions

Differentiable[1] quantum circuits

measure gradient on real device



Parametrized gate of the form

$e^{-\frac{i\theta}{2}\Sigma}$ with $\Sigma^2 = 1$

e.g., X, Y, Z, CNOT, SWAP…

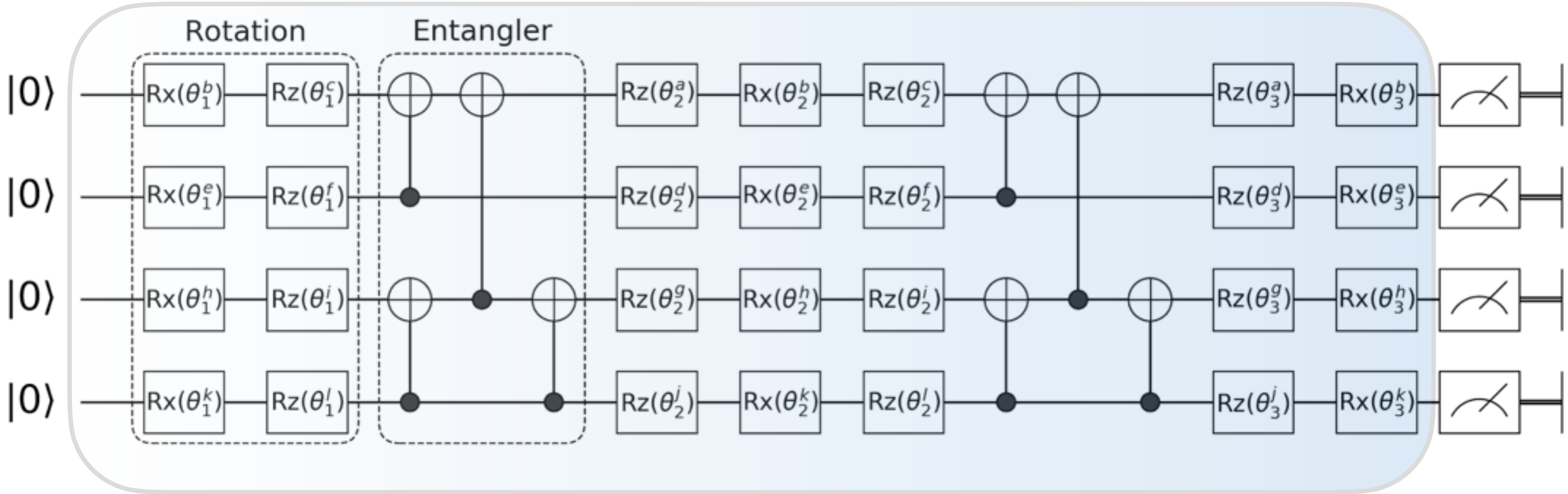
Li et al, PRL ’17, Mitarai et al, PRA ’18
Schuld et al, PRA ’19, Crooks, ’19…

$$\nabla \langle H \rangle_\theta = \left(\langle H \rangle_{\theta+\pi/2} - \langle H \rangle_{\theta-\pi/2} \right) / 2$$

Same complexity as forward mode automatic differentiation

Differentiable^2 quantum circuits

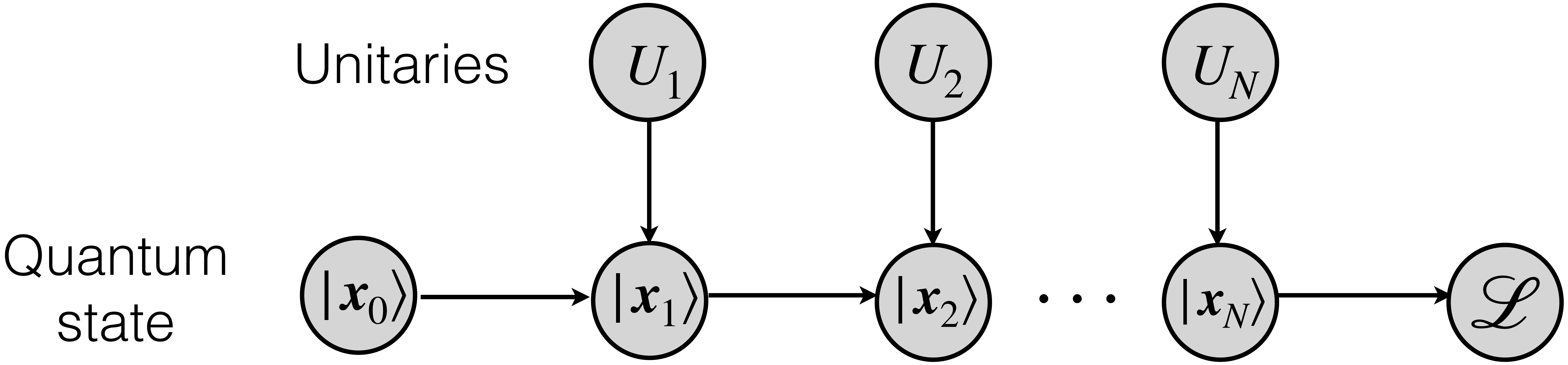
compute gradient in classical simulations



PENNYLANE TensorFlow Quantum

Unfortunately, forward mode is slow
Reverse mode is memory consuming

Quantum circuit computation graph



The same "comb graph" as the feedforward neural network, except that quantum computing is reversible

O(1) memory AD for reversible neural nets Gomez et al, 1707.04585 Chen et al, 1806.07366

Reversible AD for variational quantum circuits*

forward

$$U|x\rangle \rightarrow |y\rangle$$

backward

"uncompute" $|x\rangle \leftarrow U^\dagger |y\rangle$

adjoint for mat-vec multiply

$$\overline{|x\rangle} \leftarrow U^\dagger \overline{|y\rangle}$$

$$\overline{U} \leftarrow \overline{|y\rangle}\langle x|$$

All are in-place operations without caching

*GRAPE type algorithm on the level of circuits

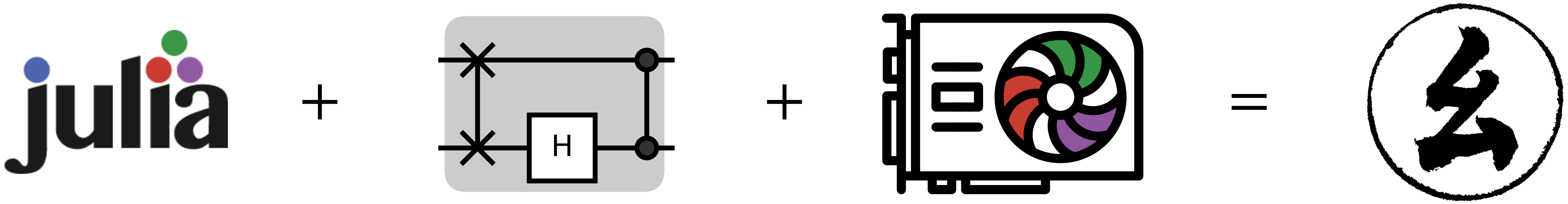
```
julia> using Yao, YaoExtensions

julia> n = 10; depth = 10000;

julia> circuit = dispatch!(
    variational_circuit(n, depth),
    :random);

julia> gatecount(circuit)
Dict{Type{#s54} where #s54 <:
    AbstractBlock,Int64} with 3 entries:
  RotationGate{1,Float64,ZGate} => 200000
  RotationGate{1,Float64,XGate} => 100010
  ControlBlock{10,XGate,1,1}    => 100000

julia> nparameters(circuit)
300010

julia> h = heisenberg(n);

julia> for i = 1:100
    _, grad = expect'(h, zero_state(n)=>
                                circuit)
    dispatch!(-, circuit, 1e-3 * grad)
    println("Step $i, energy = $(expect(
            h, zero_state(n)=>circuit))")
    end
```

Train a 10,000 layer, 300,000 parameter circuit on a laptop

https://yaoquantum.org/

Yao.jl: Extensible, Efficient Framework for Quantum Algorithm Design

https://yaoquantum.org/



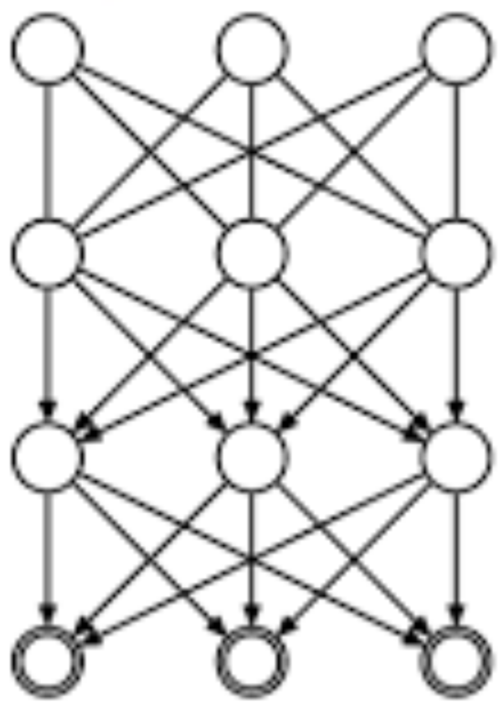
Xiu-Zhe Roger Luo (IOP, CAS → Waterloo & PI)
Jin-Guo Liu (IOP, CAS → Harvard & QuEra)

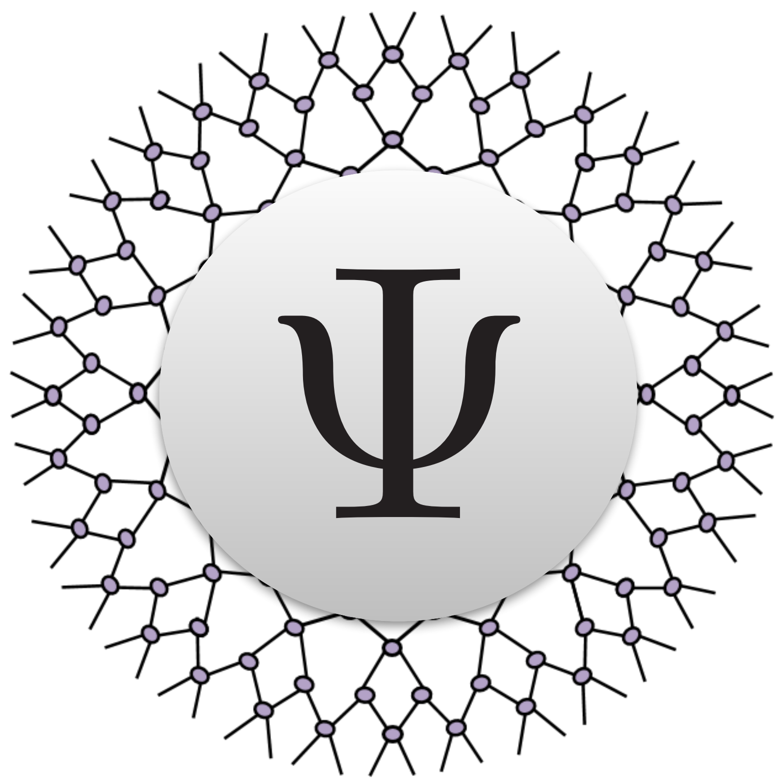
Features:

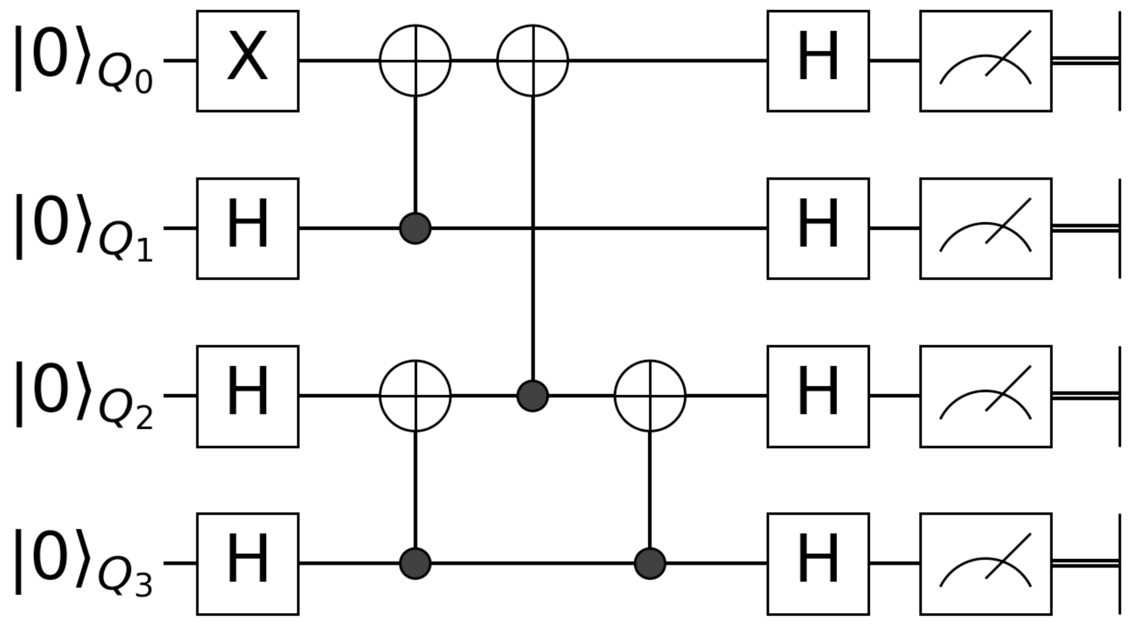
- Differentiable programming quantum circuits
- Batch parallelization with GPU acceleration
- Quantum block intermediate representation

❶ Differentiable Programming

$\partial\Big[$

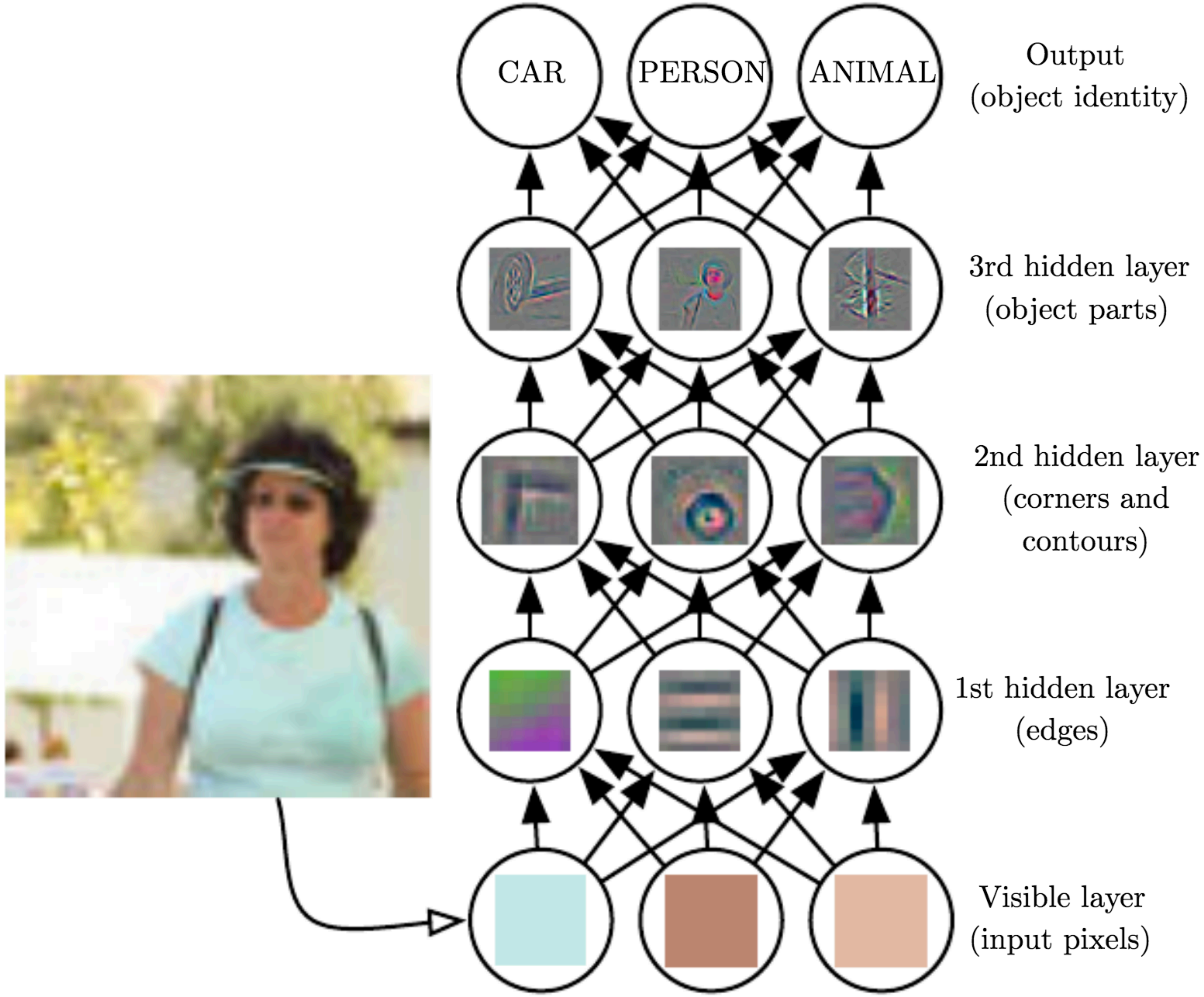




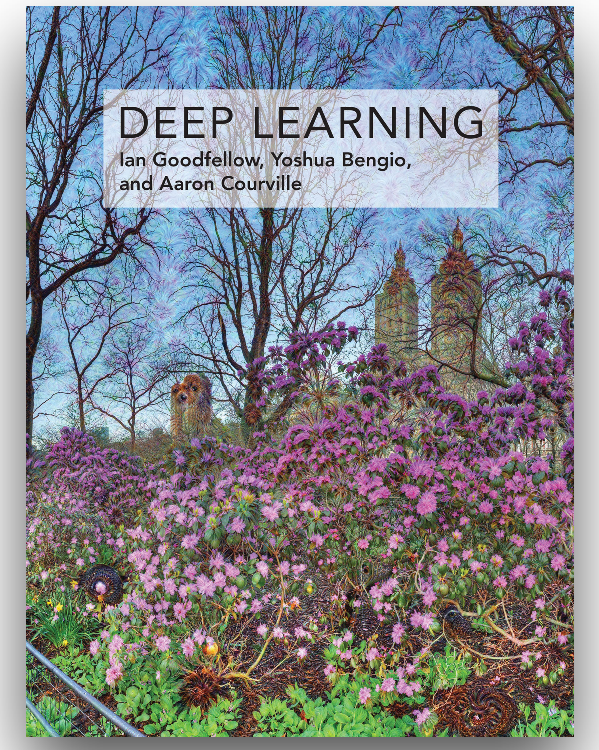


$\Big]$

② Representation Learning



Page 6
Figure 1.2



Goodfellow, Bengio, Courville, http://www.deeplearningbook.org/

Magic of learning representations

Neural style transfer

Gatys et al, 1508.06576

Latent space interpolation

Glow 1807.03039

https://blog.openai.com/glow/

Representation learning: what and how ?

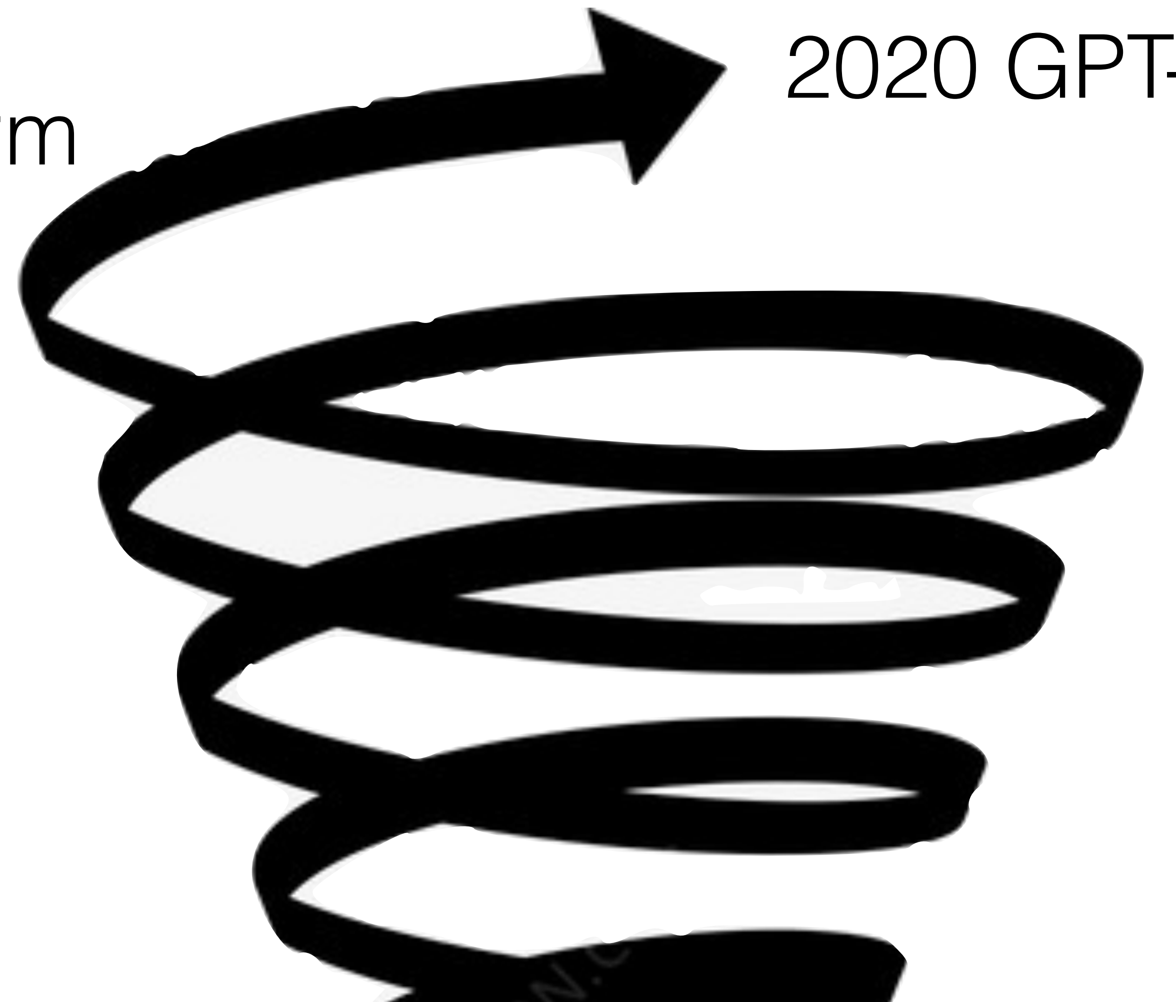
What is a good representation ?

1812.02230

Towards a Definition of Disentangled Representations

Irina Higgins*, David Amos*, David Pfau, Sebastien Racaniere,
Loic Matthey, Danilo Rezende, Alexander Lerchner
DeepMind

Generative **P**re-**T**raining appears to be a successful way in learning good representations



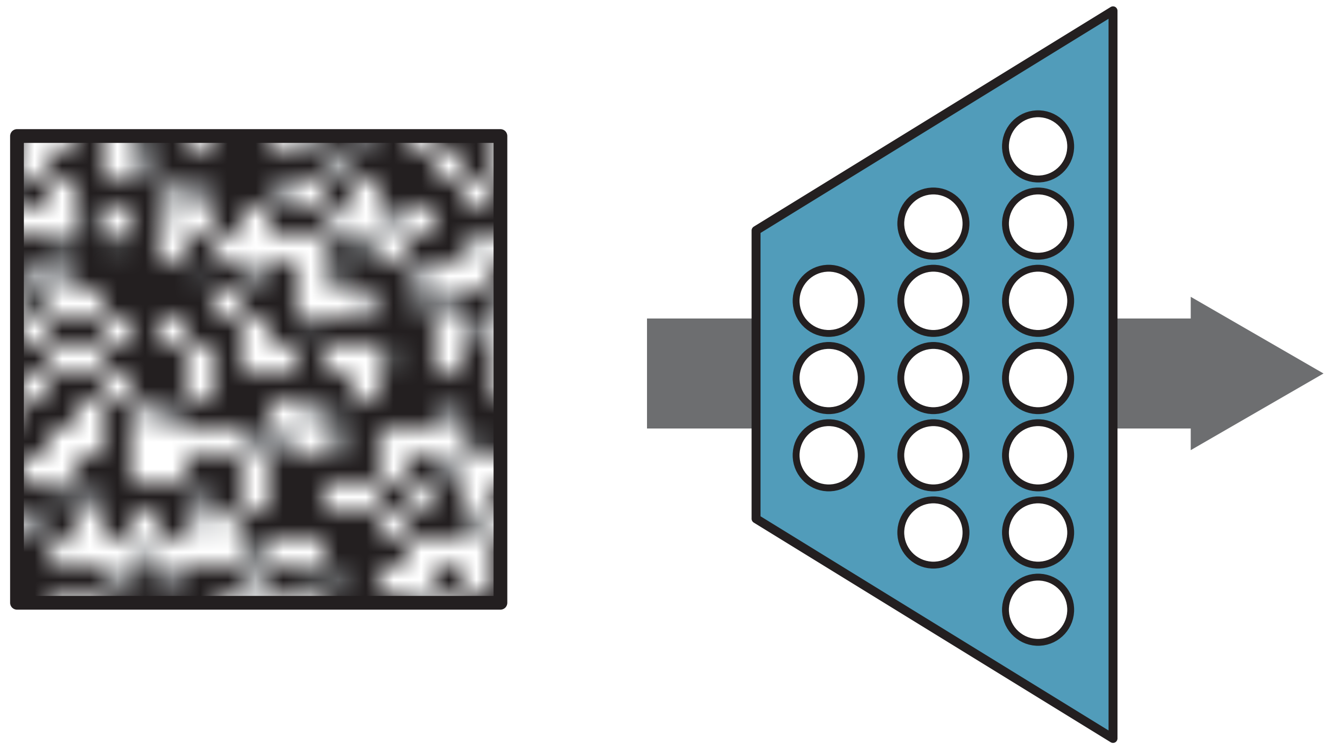
Discriminative learning | **Generative learning**

$y = f(\boldsymbol{x})$ or $p(y \mid \boldsymbol{x})$

$p(\boldsymbol{x}, y)$ or $p(\boldsymbol{x})$

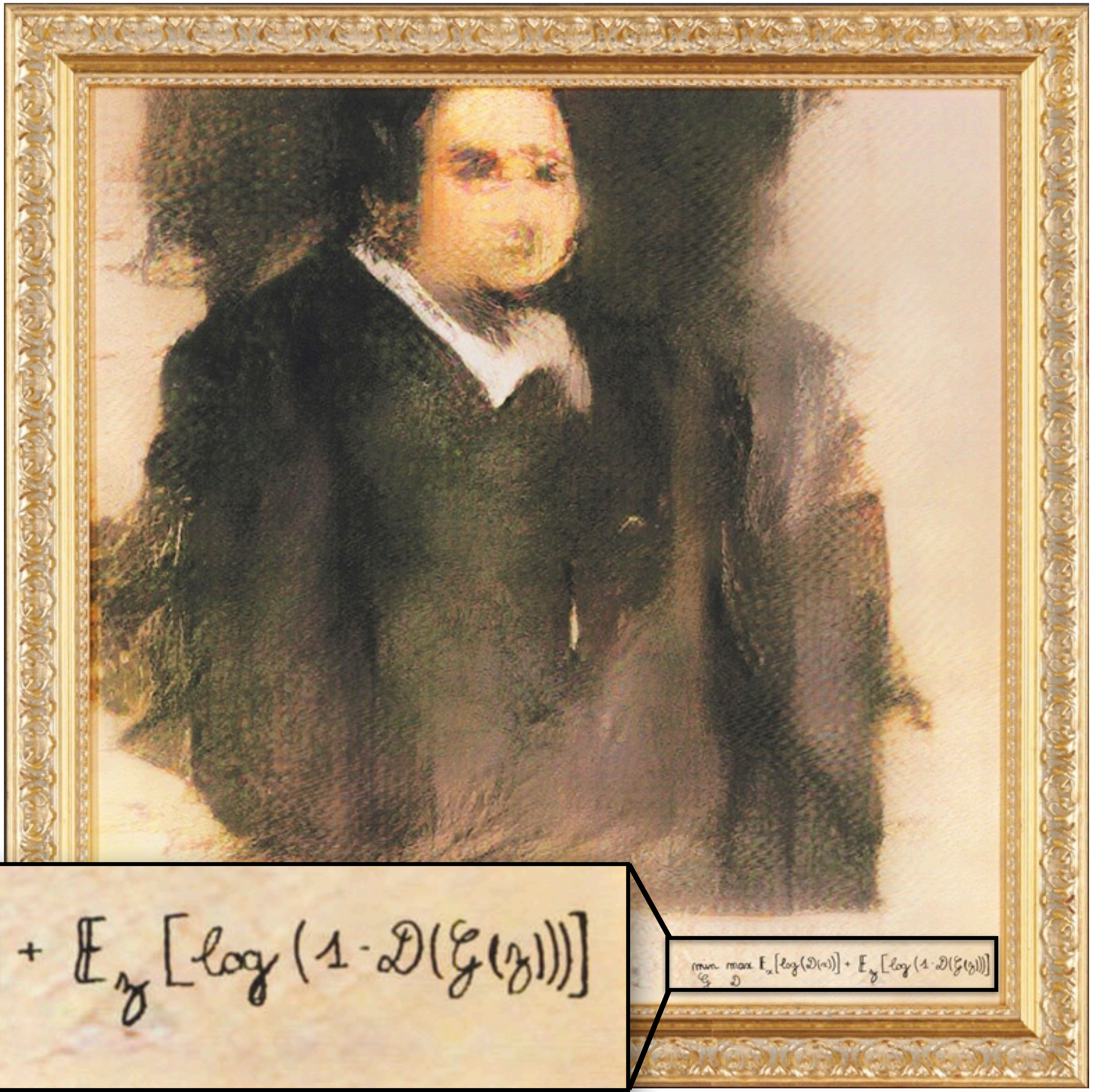
“What I can not create, I do not understand”

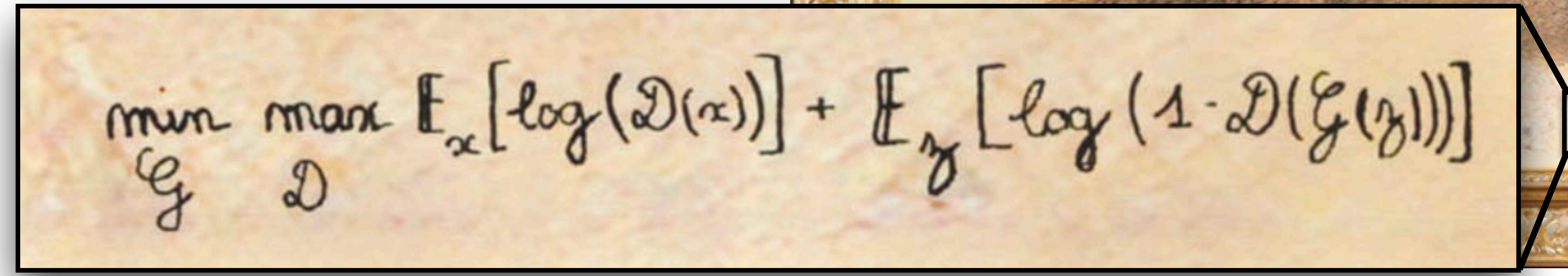
Generated Arts



Gaussian Noise

Generative Network



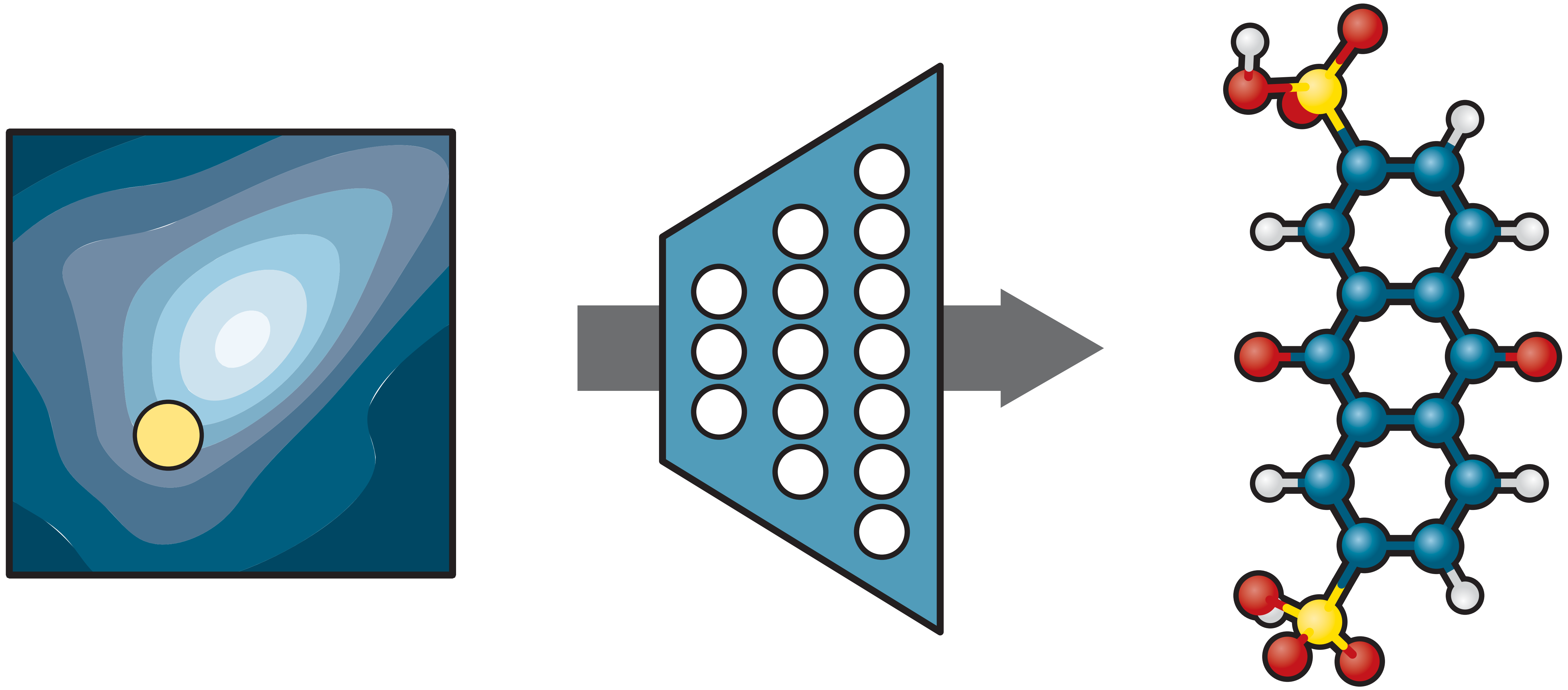


$432,500
25 October 2018
Christie's New York

https://www.christies.com/Features/A-collaboration-between-two-artists-one-human-one-a-machine-9332-1.aspx

Generating molecules

Latent attributes



Physical configurations

Simple Distributions

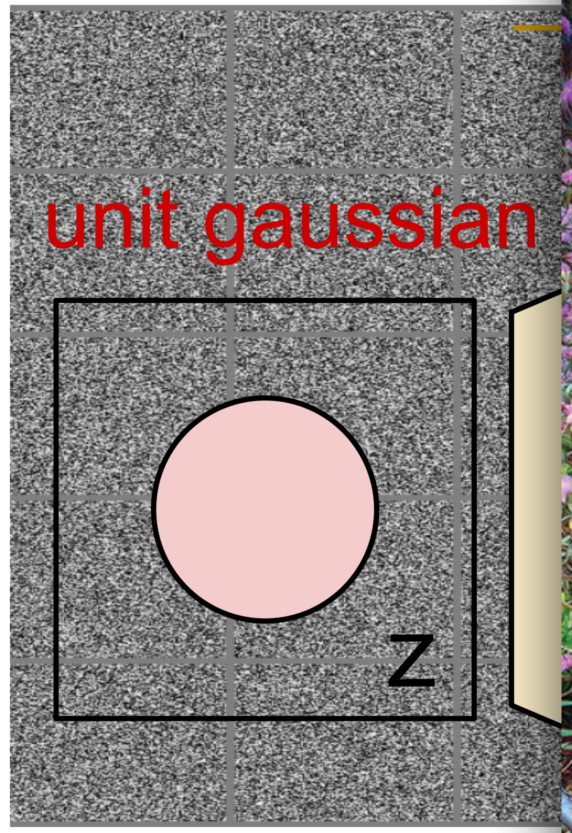
Generate $\rightleftarrows$ Inference

Complex Distribution

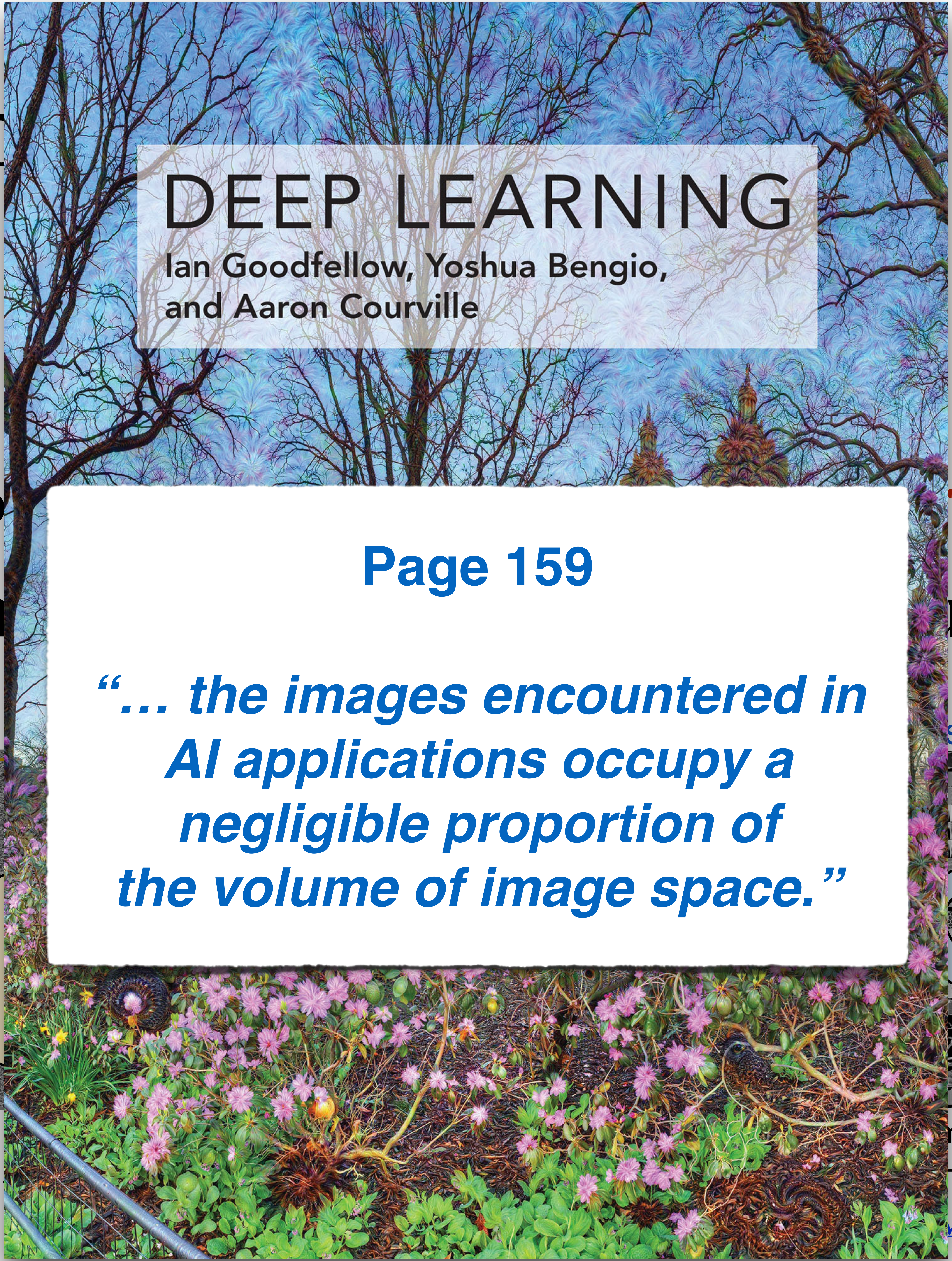
Sanchez-Lengeling & Aspuru-Guzik,
Inverse molecular design using machine learning:
Generative models for matter engineering, Science '18

Probab deling

How to om a
high-din ition ?



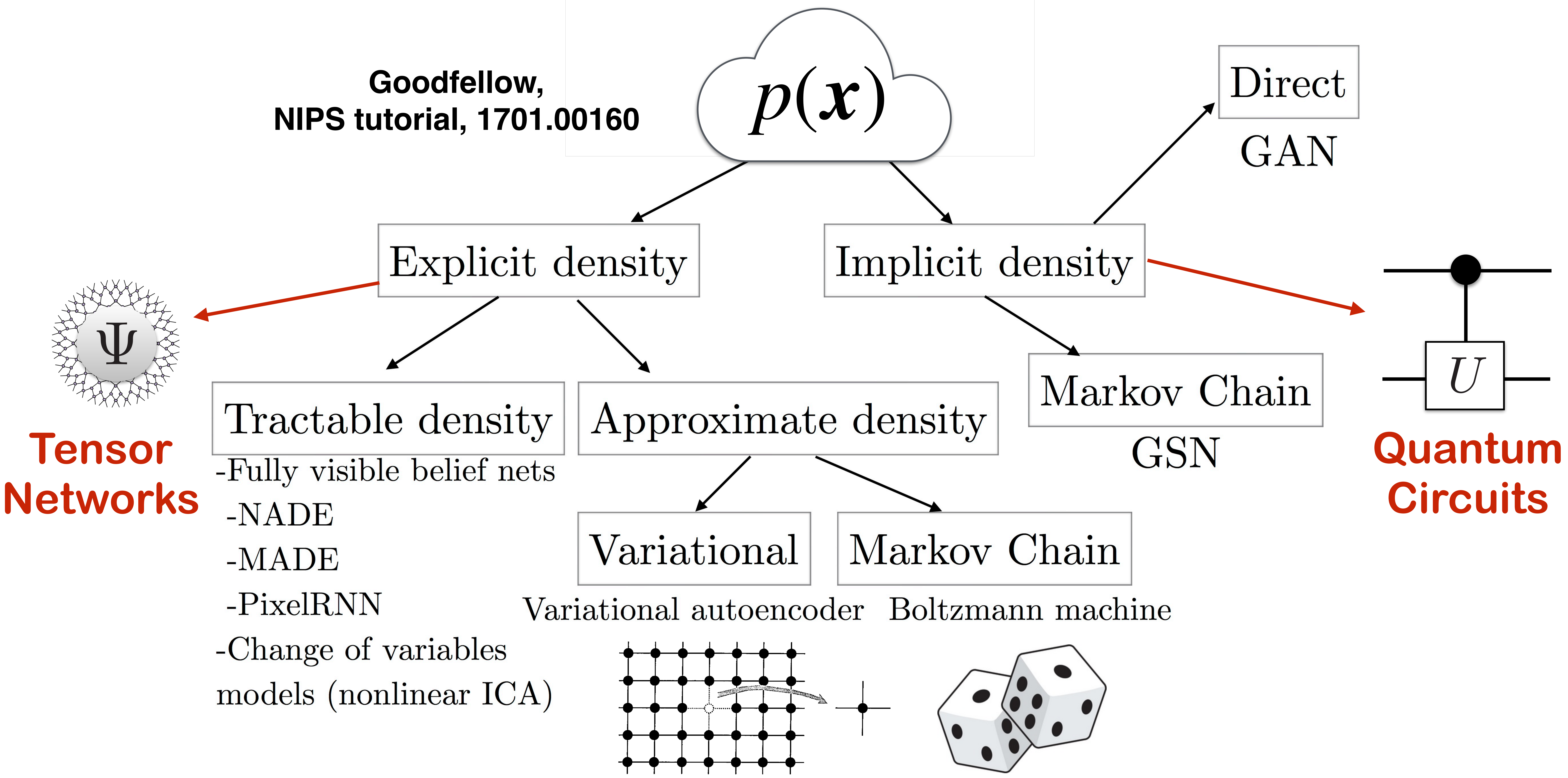
“random”



stribution

ve-models/

Generative models and their physics genes

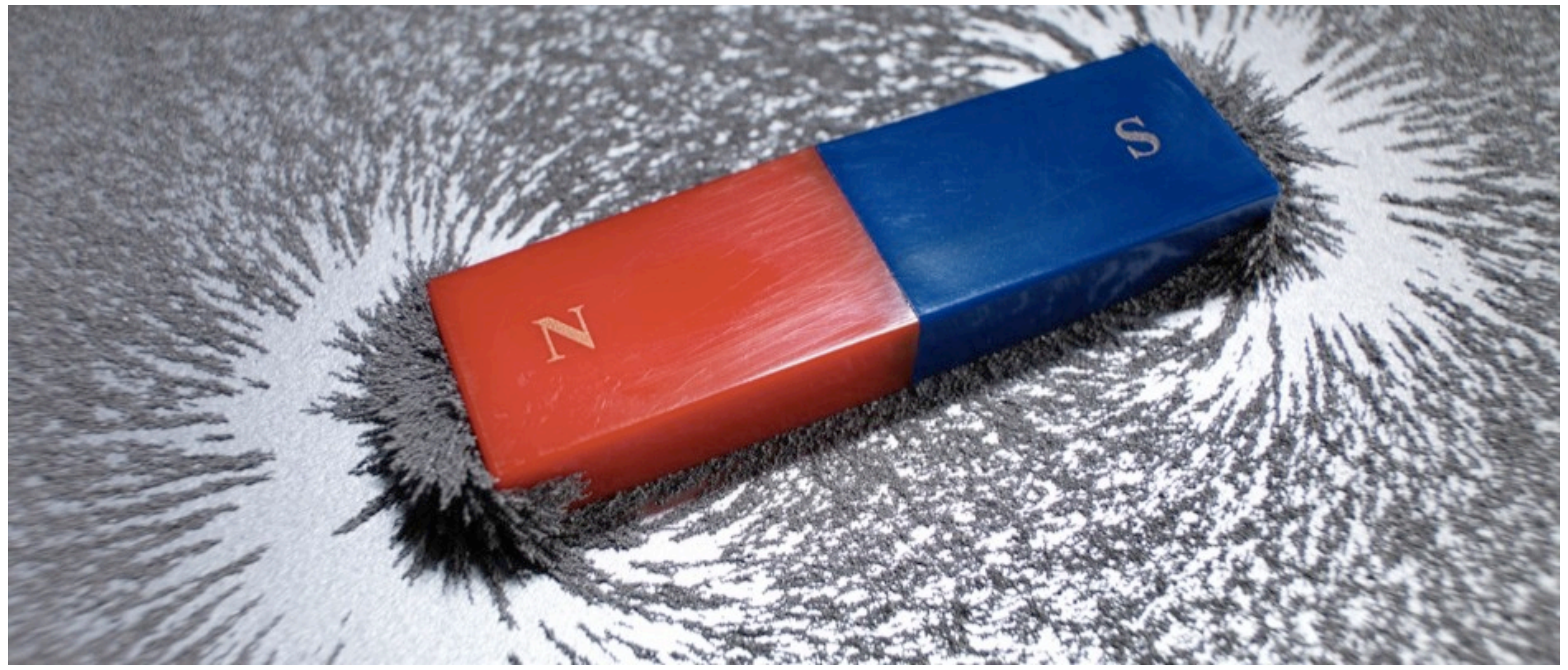


Generative modeling

Known: samples

Unknown: generating distribution

Statistical physics



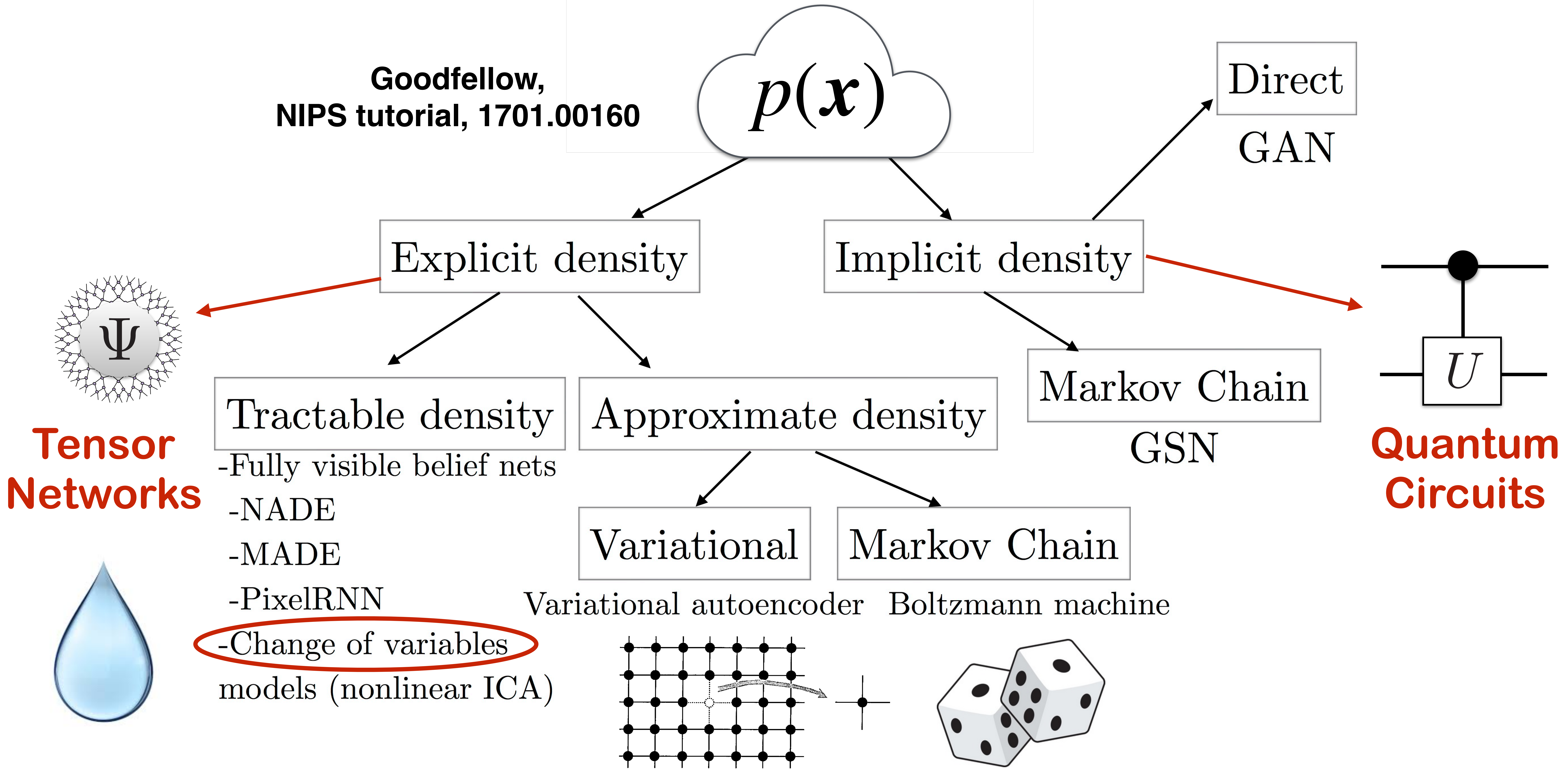
Known: energy function

Unknown: samples, partition function

Modern generative models for physics

Physics of and for generative modeling

Generative models and their physics genes



Lecture Note http://wangleiphy.github.io/lectures/PILtutorial.pdf

Generative Models for Physicists

Lei Wang*

Institute of Physics, Chinese Academy of Sciences
Beijing 100190, China

October 28, 2018

Abstract

Generative models generate unseen samples according to a learned joint probability distribution in the high-dimensional space. They find wide applications in density estimation, variational inference, representation learning and more. Deep generative models and associated techniques (such as differentiable programing and representation learning) are cutting-edge technologies physicists can learn from deep learning.

This note introduces the concept and principles of generative modeling, together with applications of modern generative models (autoregressive models, normalizing flows, variational autoencoders etc) as well as the old ones (Boltzmann machines) to physics problems. As a bonus, this note puts some emphasize on physics-inspired generative models which take insights from statistical, quantum, and fluid mechanics.

The latest version of the note is at http://wangleiphy.github.io/. Please send comments, suggestions and corrections to the email address in below.

CONTENTS

Generative modeling with normalizing flows

WaveNet 1609.03499 1711.10433

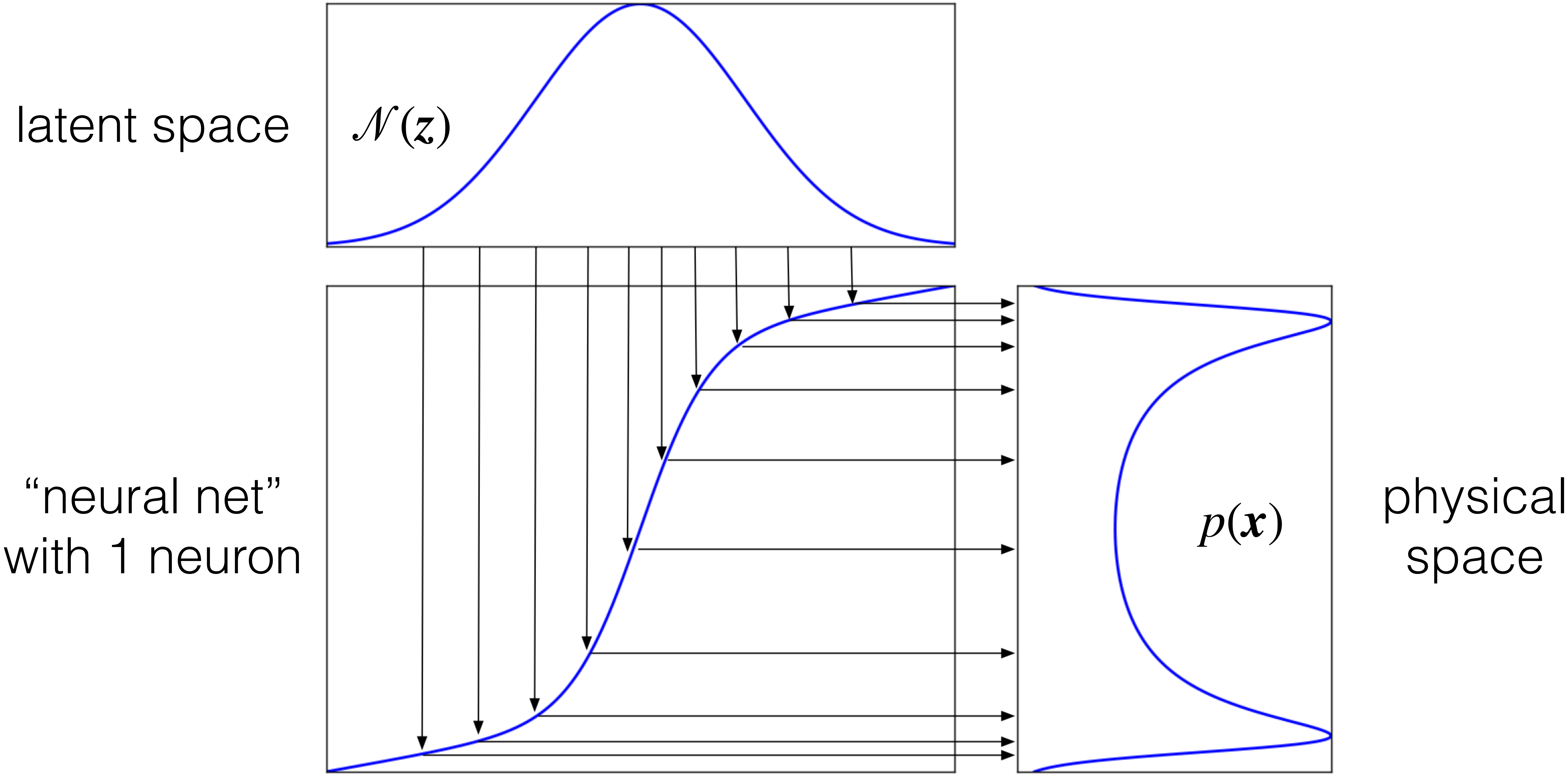
https://deepmind.com/blog/wavenet-generative-model-raw-audio/

https://deepmind.com/blog/high-fidelity-speech-synthesis-wavenet/

Glow 1807.03039

https://blog.openai.com/glow/

Normalizing flow in a nutshell



Normalizing Flows

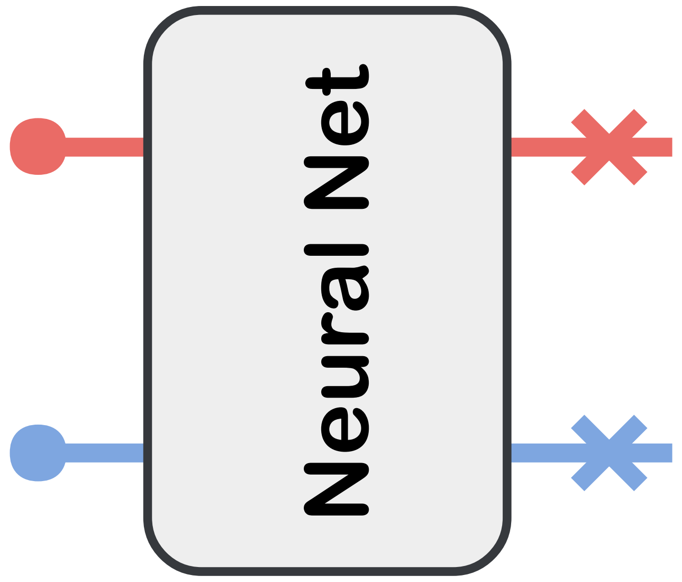
Change of variables $x \leftrightarrow z$ **with deep neural nets**

$$p(x) = \mathcal{N}(z)\left|\det\left(\frac{\partial z}{\partial x}\right)\right|$$

Review article 1912.02762
Tutorial https://iclr.cc/virtual_2020/speaker_4.html

composable, differentiable, and invertible mapping between manifolds

x



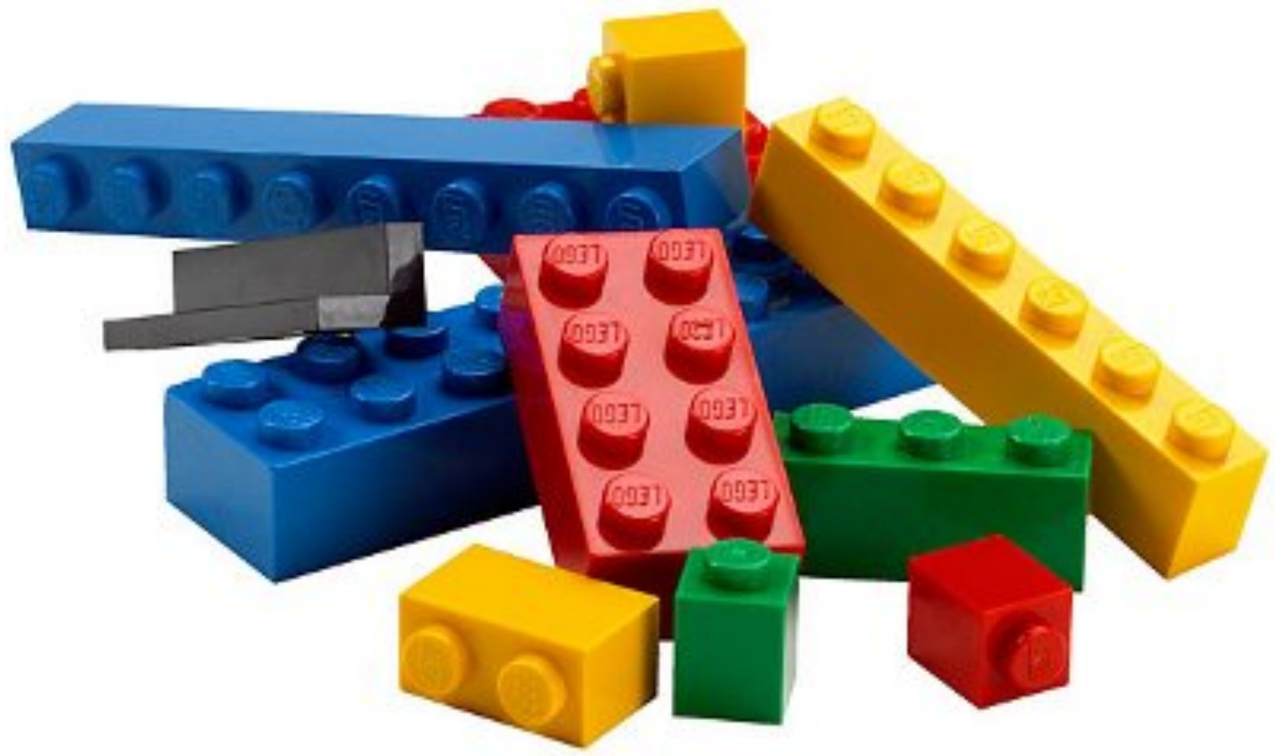
$z \sim \mathcal{N}(z)$

Learn probability transformations with normalizing flows

Got this name in Tabak & Vanden-Eijnden, Commun. Math. Sci. '10

Architecture design principle

Composability

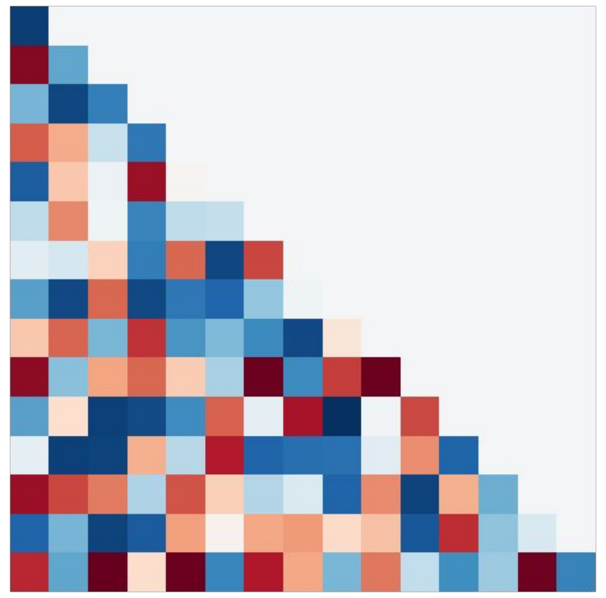


$$z = \mathcal{T}(\boldsymbol{x})$$

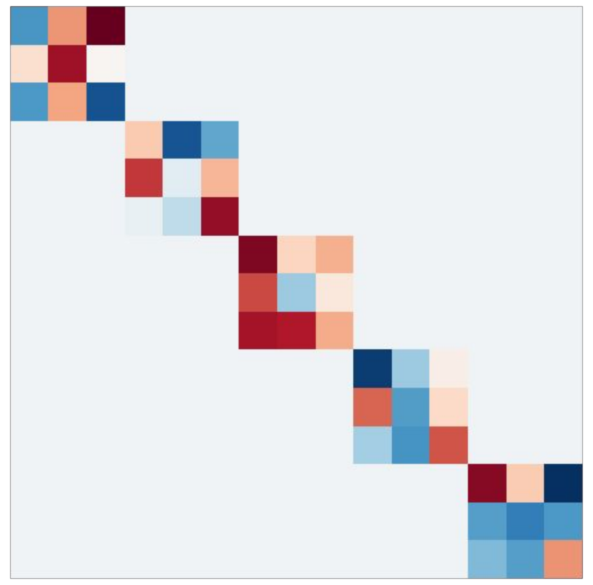
$$\mathcal{T} = \mathcal{T}_1 \circ \mathcal{T}_2 \circ \mathcal{T}_3 \circ \cdots$$

Balanced efficiency & inductive bias

$$\left|\det\left(\frac{\partial z}{\partial x}\right)\right|$$



Autoregressive



Neural RG

$$\frac{\partial \rho(\boldsymbol{x}, t)}{\partial t} + \nabla \cdot \left[\rho(\boldsymbol{x}, t)\boldsymbol{v}\right] = 0$$

Continuous flow

Example of a building block

Forward

arbitrary
neural nets

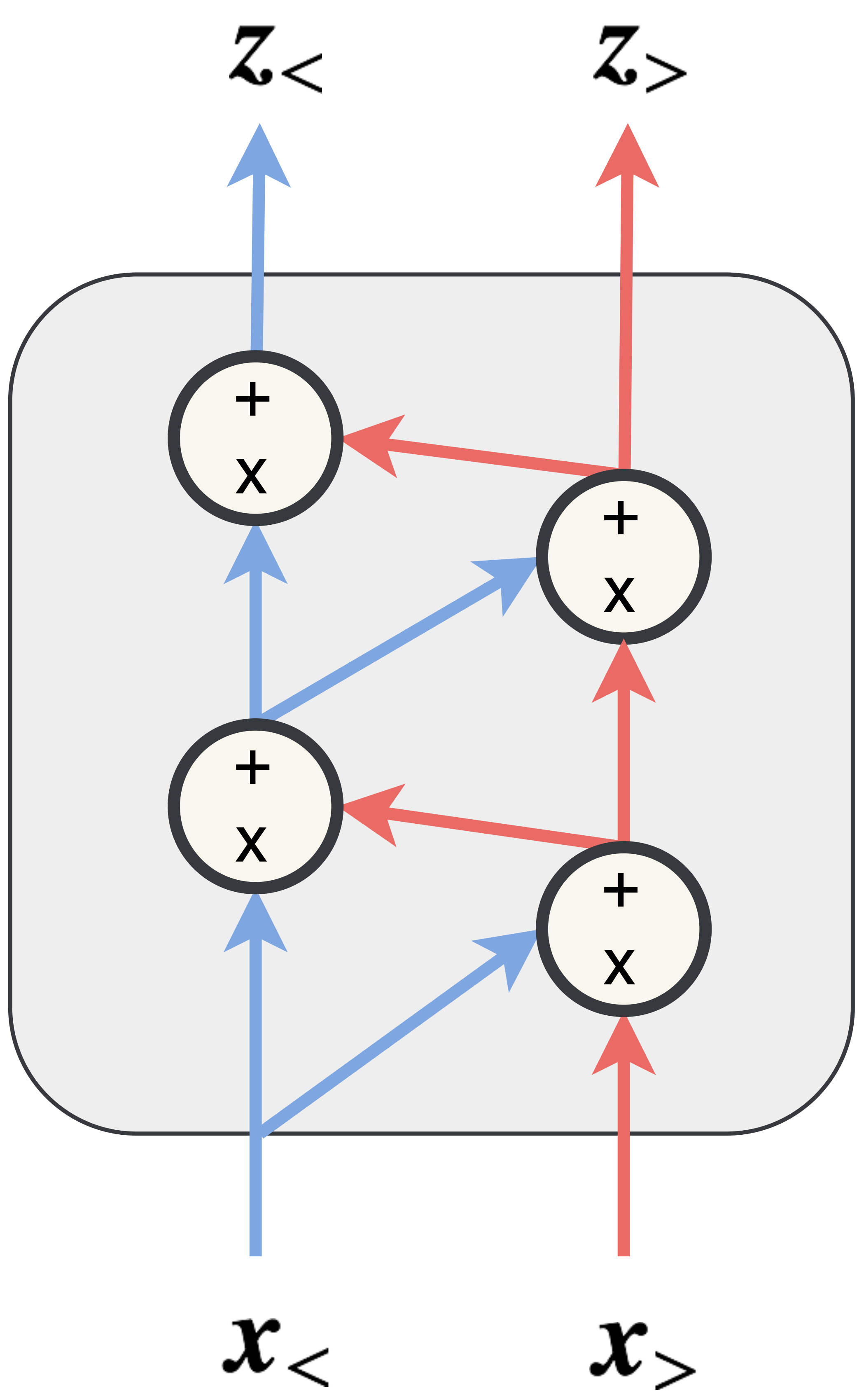
$$\begin{cases} \boldsymbol{x}_< = \boldsymbol{z}_< \\ \boldsymbol{x}_> = \boldsymbol{z}_> \odot e^{s(\boldsymbol{z}_<)} + t(\boldsymbol{z}_<) \end{cases}$$

Inverse

$$\begin{cases} \boldsymbol{z}_< = \boldsymbol{x}_< \\ \boldsymbol{z}_> = (\boldsymbol{x}_> - t(\boldsymbol{x}_<)) \odot e^{-s(\boldsymbol{x}_<)} \end{cases}$$

Log-Abs-Jacobian-Det

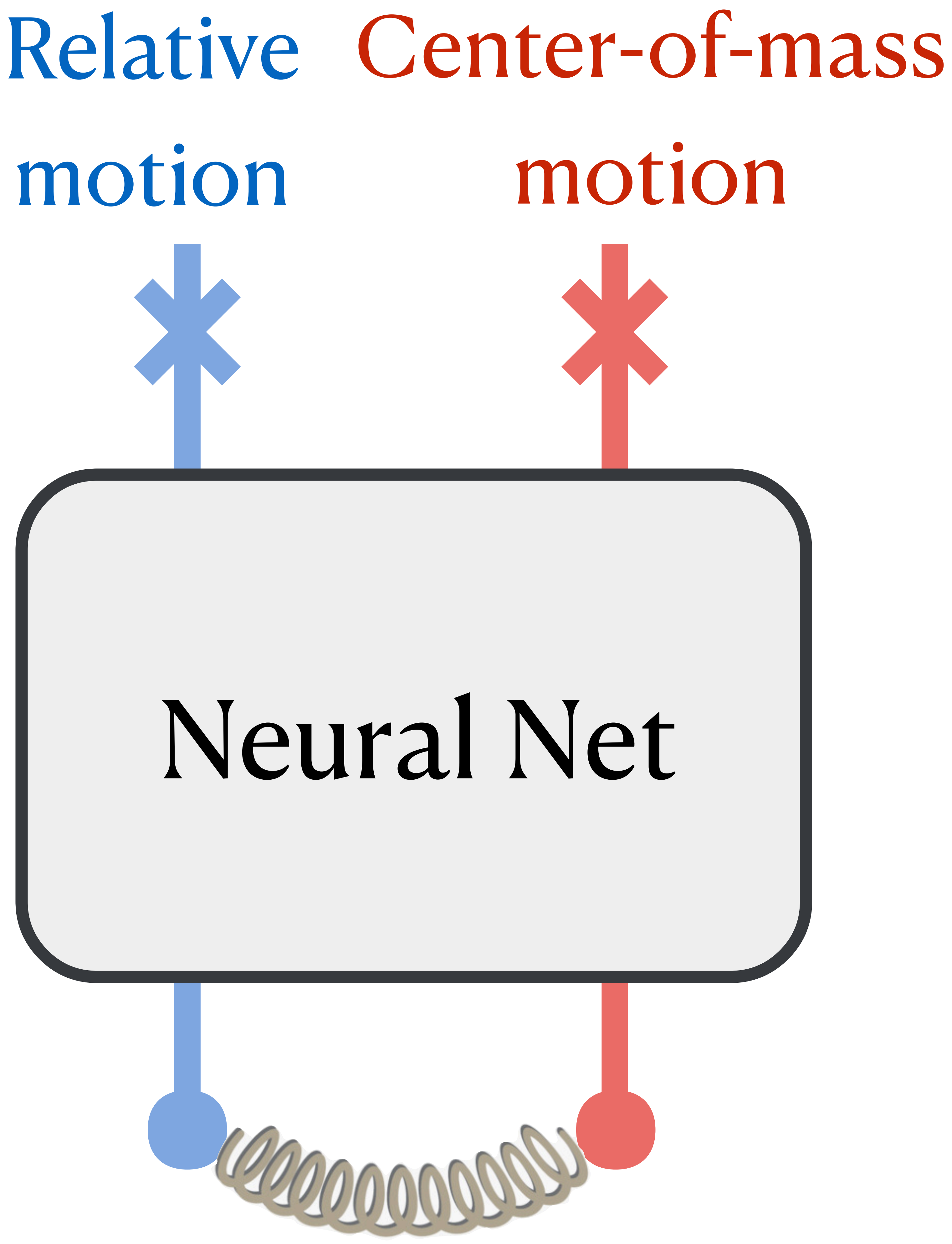
$$\ln \left| \det \left(\frac{\partial x}{\partial z} \right) \right| = \sum_i [s(\boldsymbol{z}_<)]_i$$



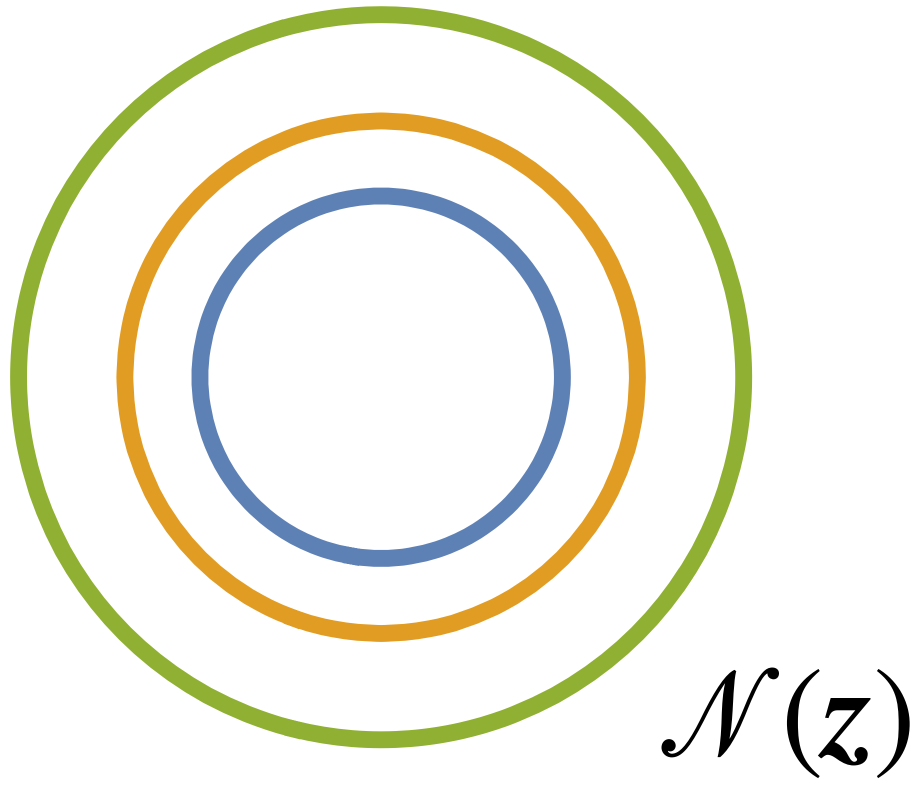
Real NVP, Dinh et al,1605.08803

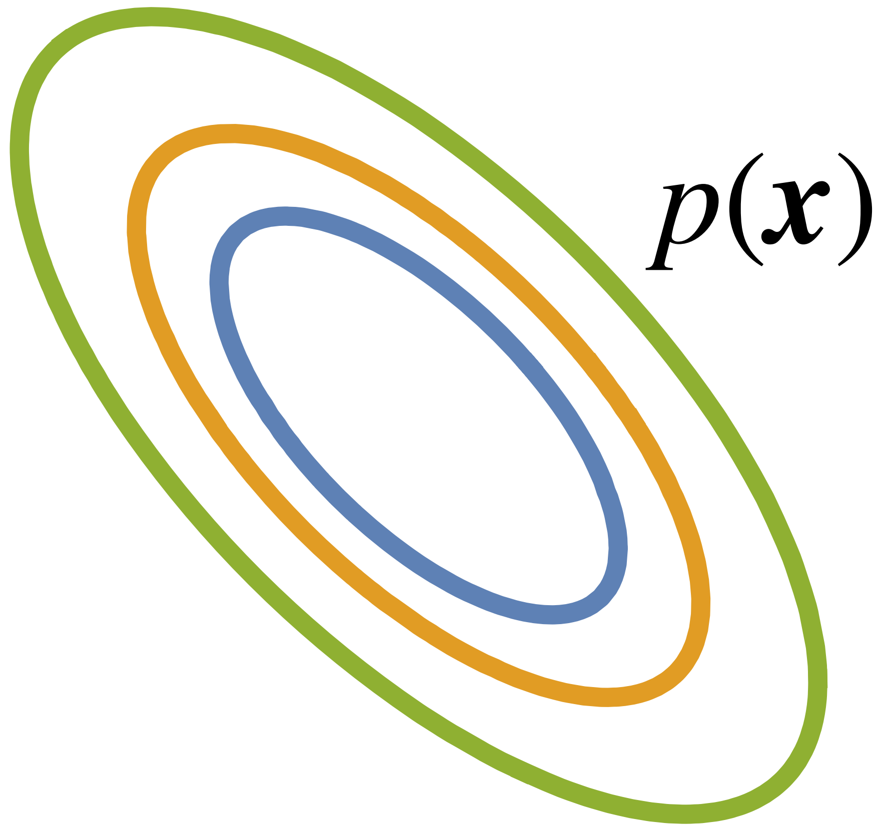
Turns out to have surprising connection Störmer–Verlet integration (later)

How it can be useful in physics ?



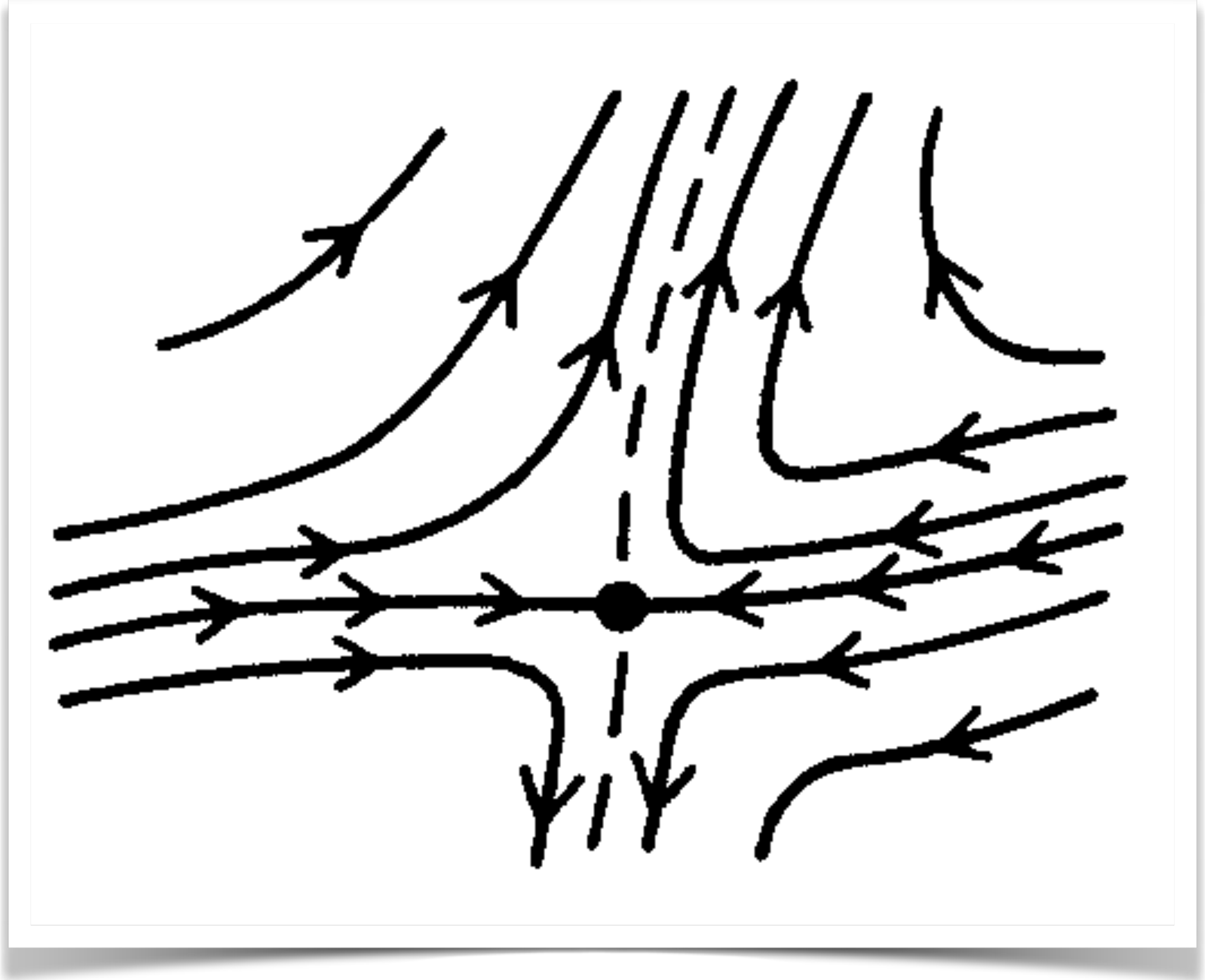
Coupled harmonic oscillator





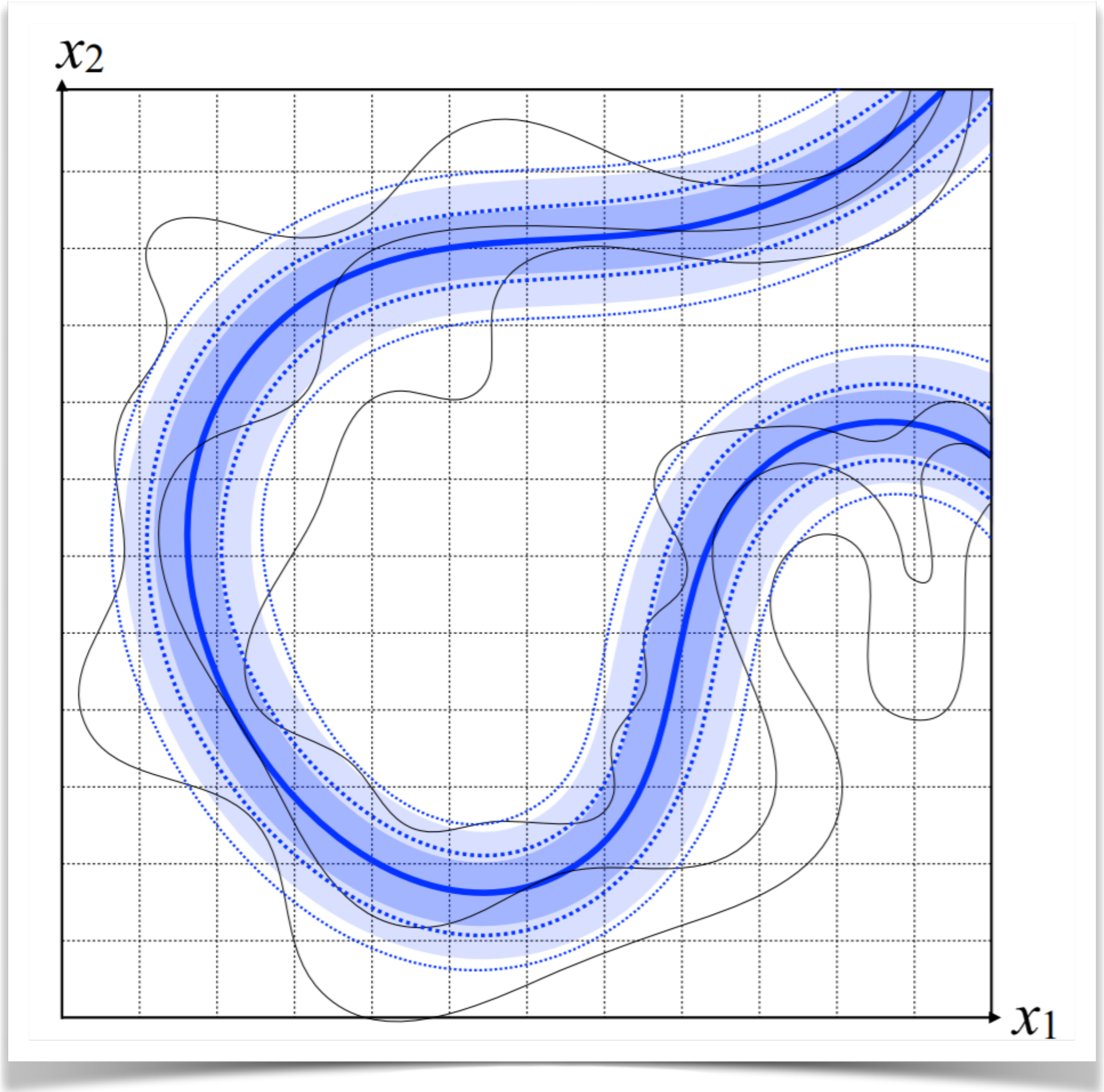
How it can be useful in physics ?

Renormalization group



Effective theory emerges upon transformation of the variables

Monte Carlo update

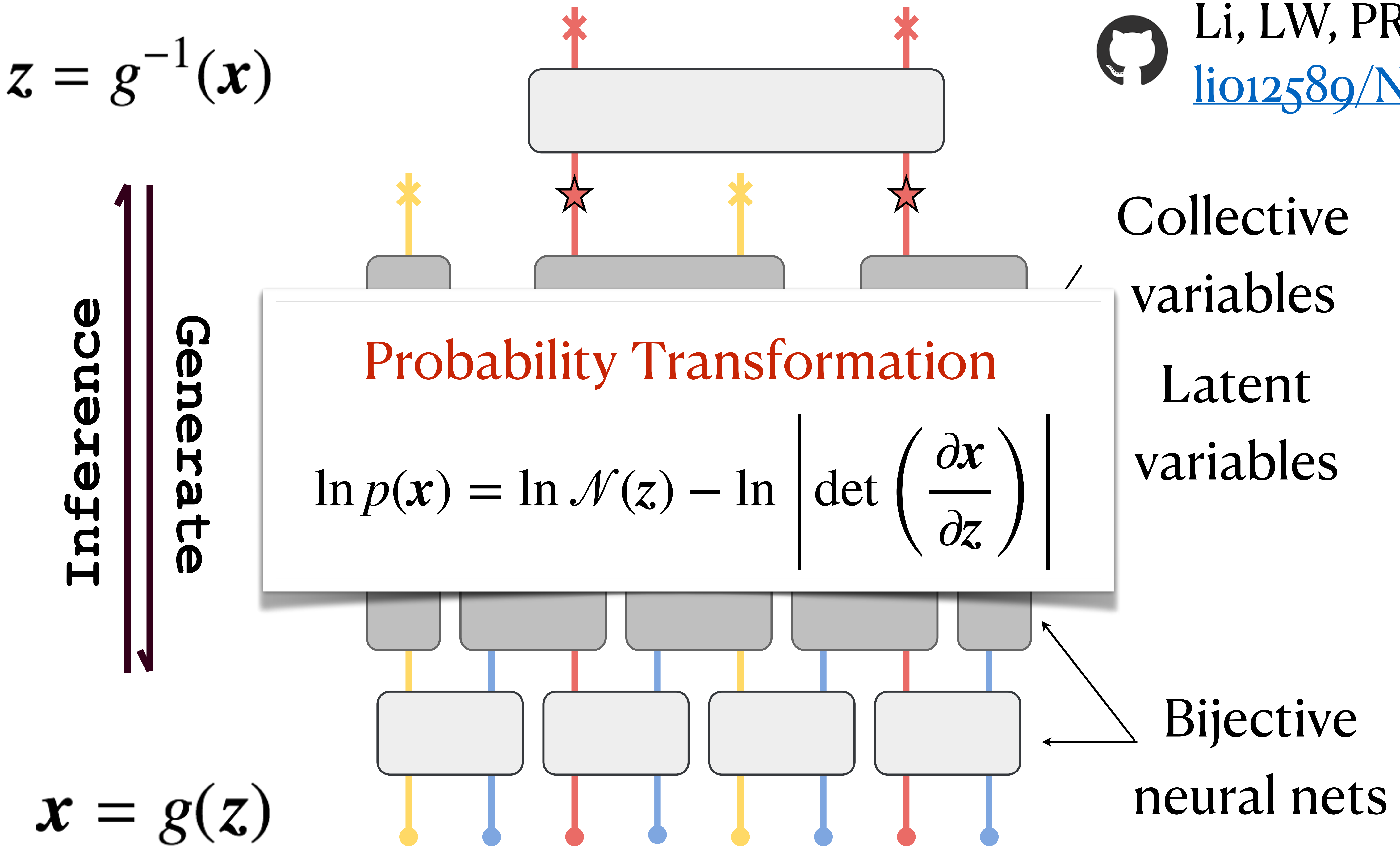


Physics happens on a manifold

Learn neural nets to unfold that manifold

Neural Network Renormalization Group

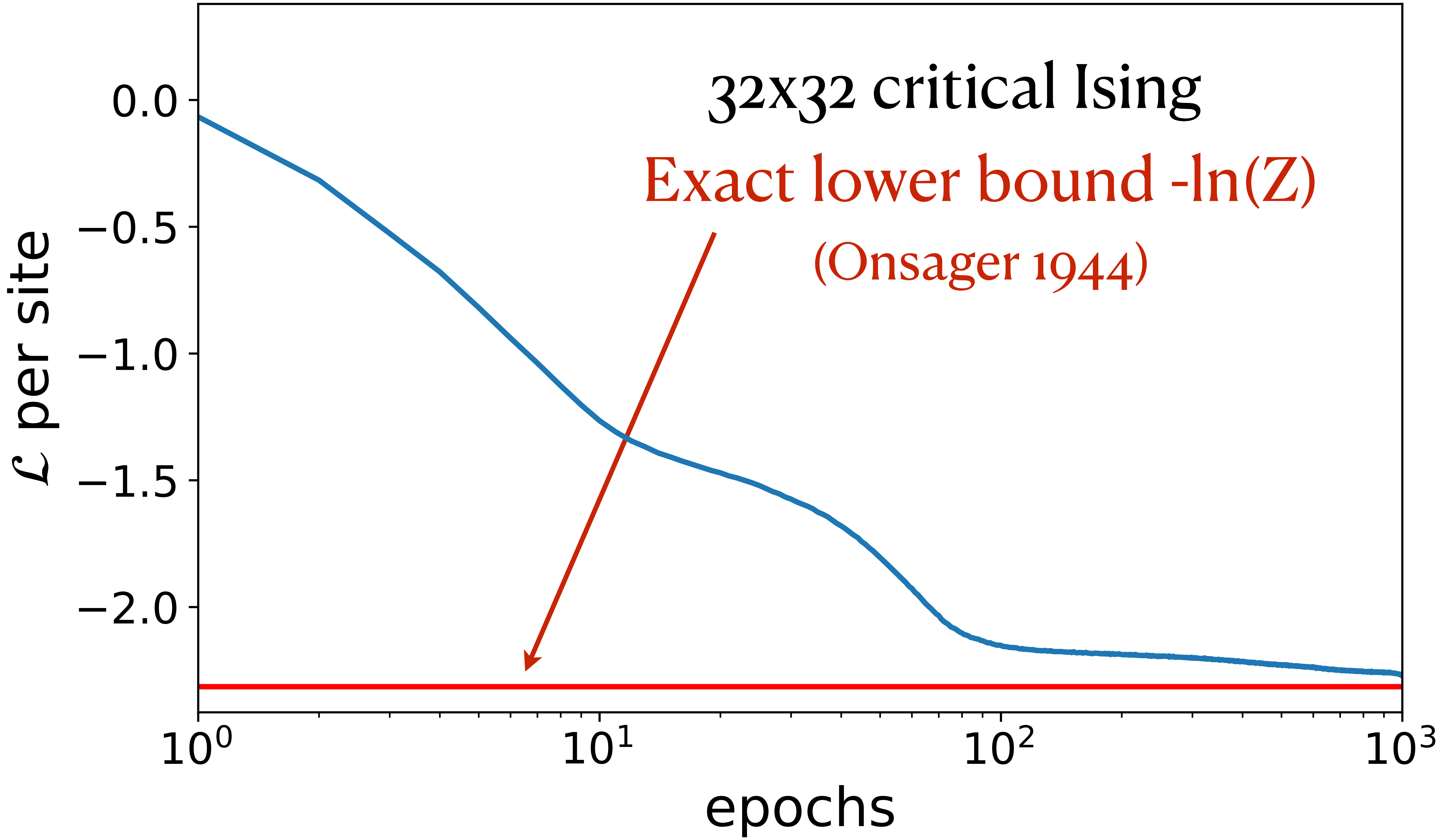
Li, LW, PRL '18
[li012589/NeuralRG]



Probability Transformation

$$\ln p(x) = \ln \mathcal{N}(z) - \ln \left| \det \left(\frac{\partial x}{\partial z} \right) \right|$$

Variational Loss

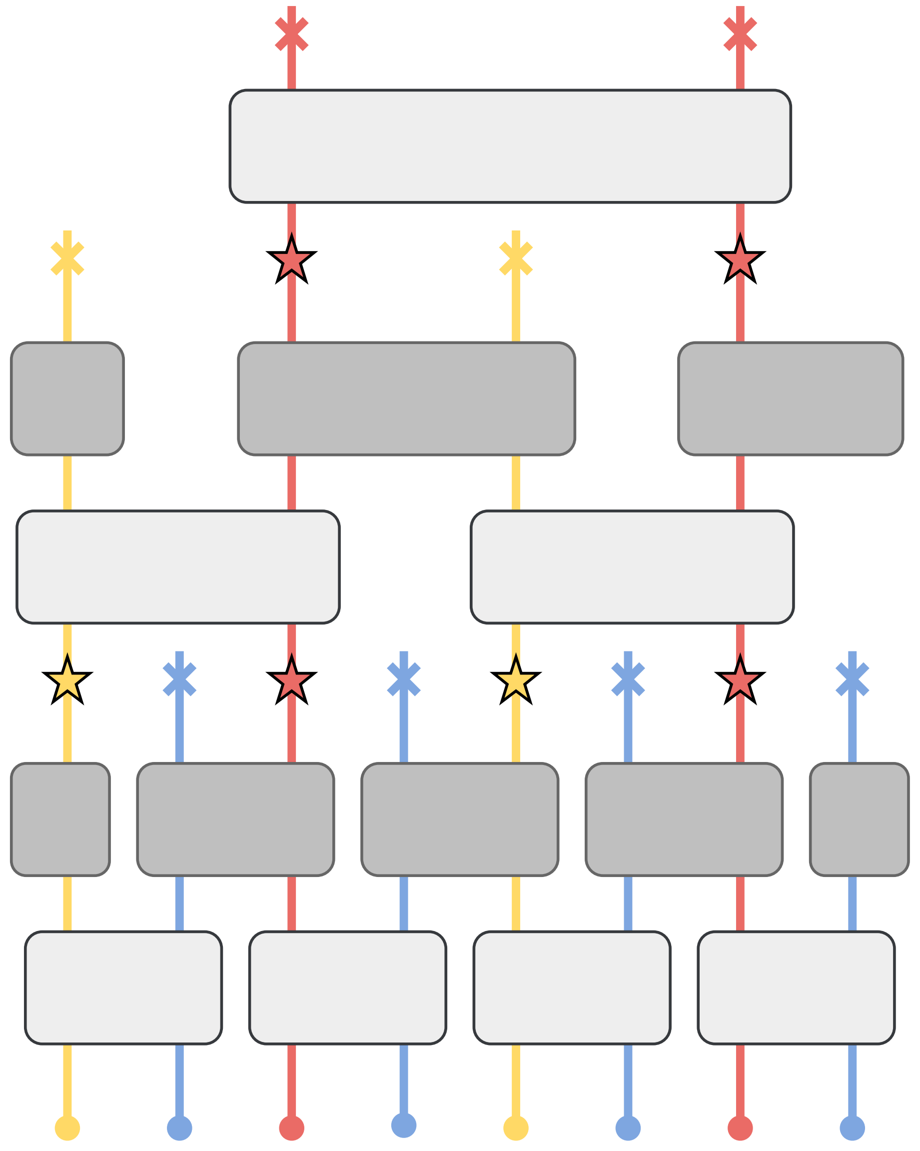


Training = Variational free energy calculation

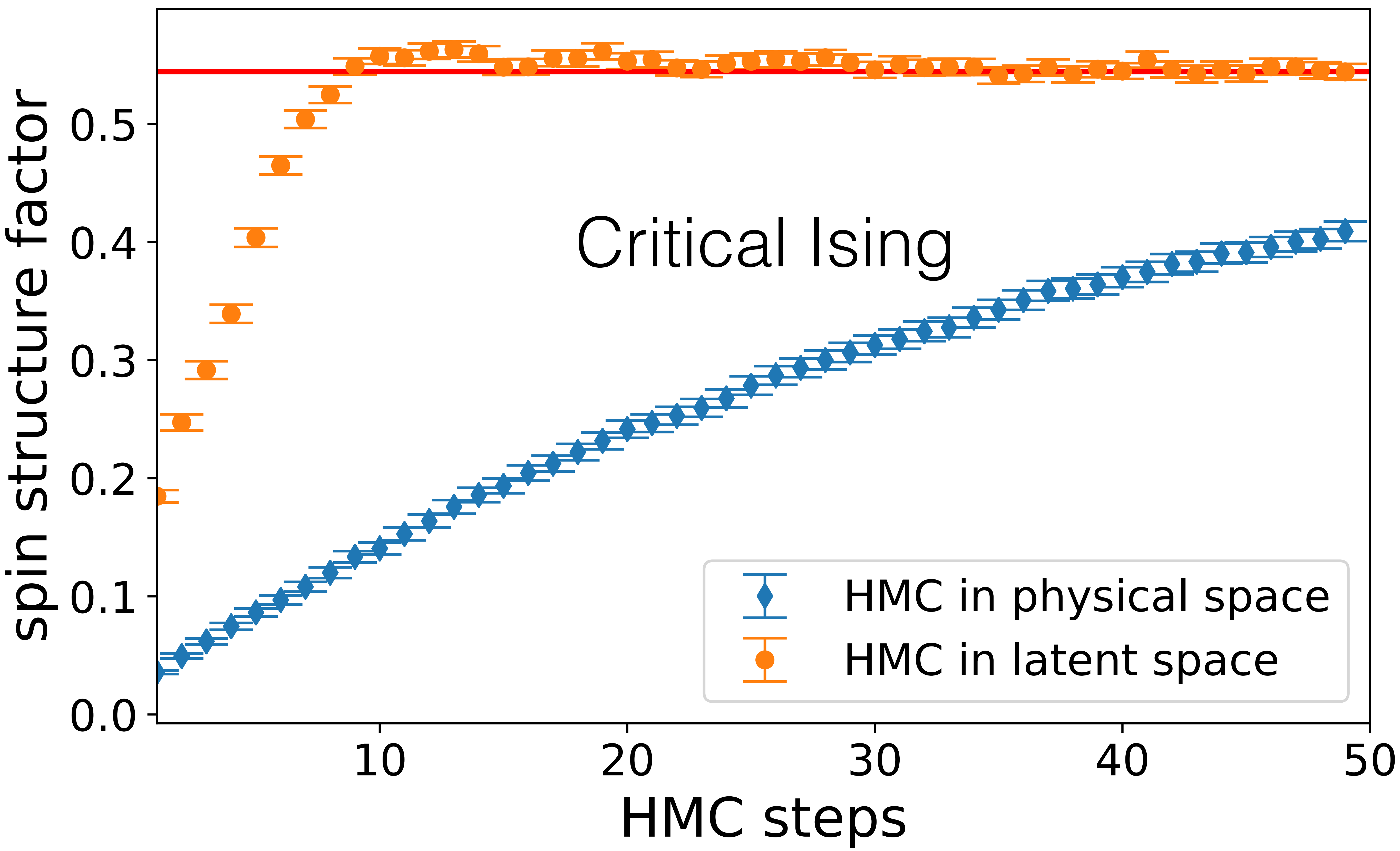
Sampling in the latent space

Latent space energy function

$E_{\mathrm{eff}}(z) = E(g(z)) + \ln p(g(z)) - \ln \mathcal{N}(z)$



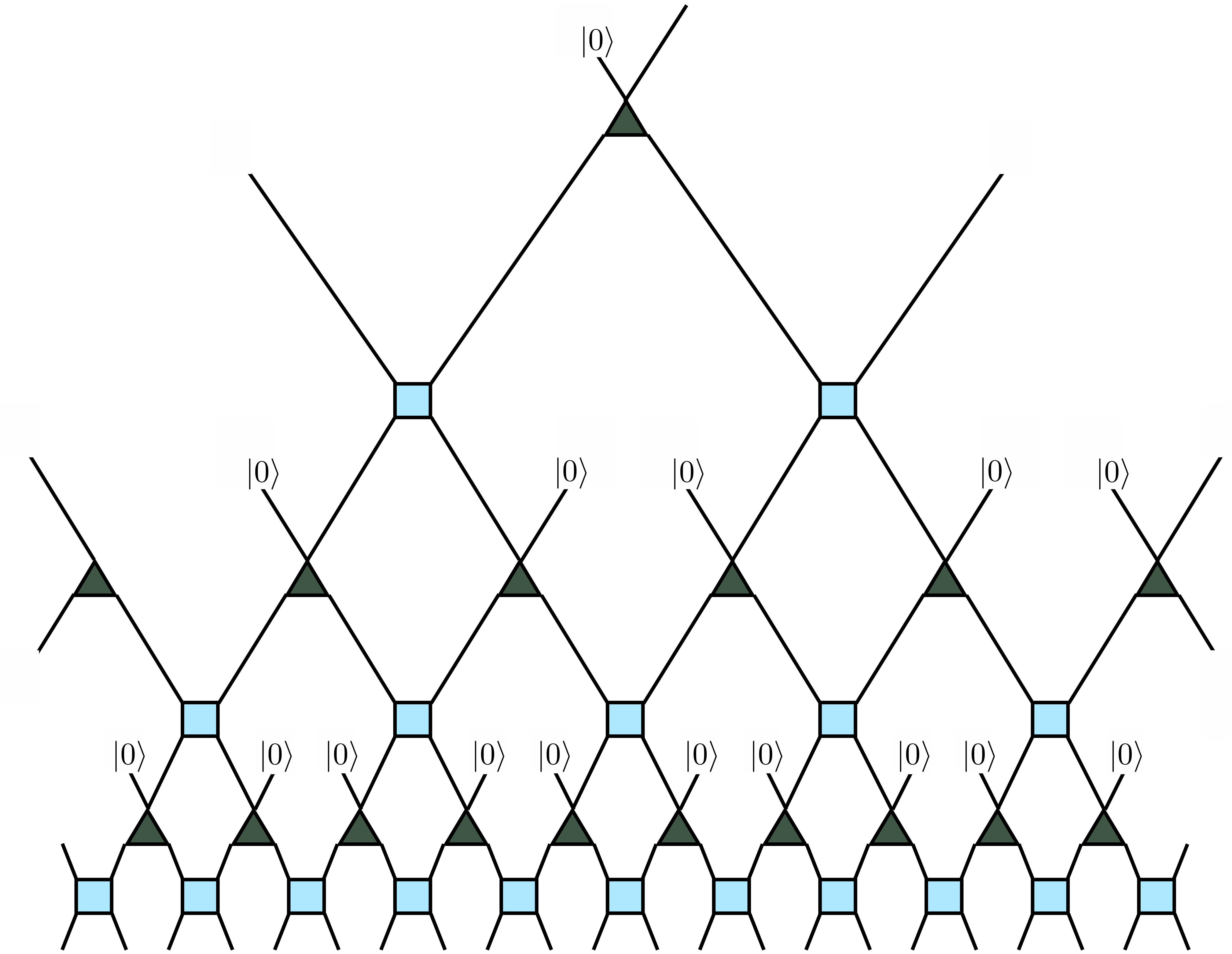
Physical energy function $E(\boldsymbol{x})$



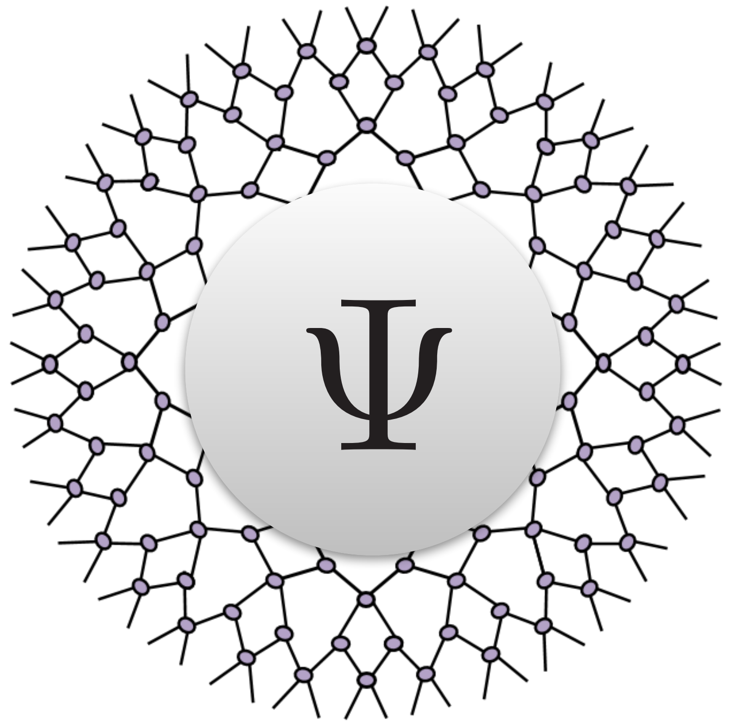
MC thermalizes faster (in the unit of MC steps) in the latent space

Other ways to close variational gap: neural importance sampling, Metropolis rejection of flow proposal ...

Quantum origin of the architecture

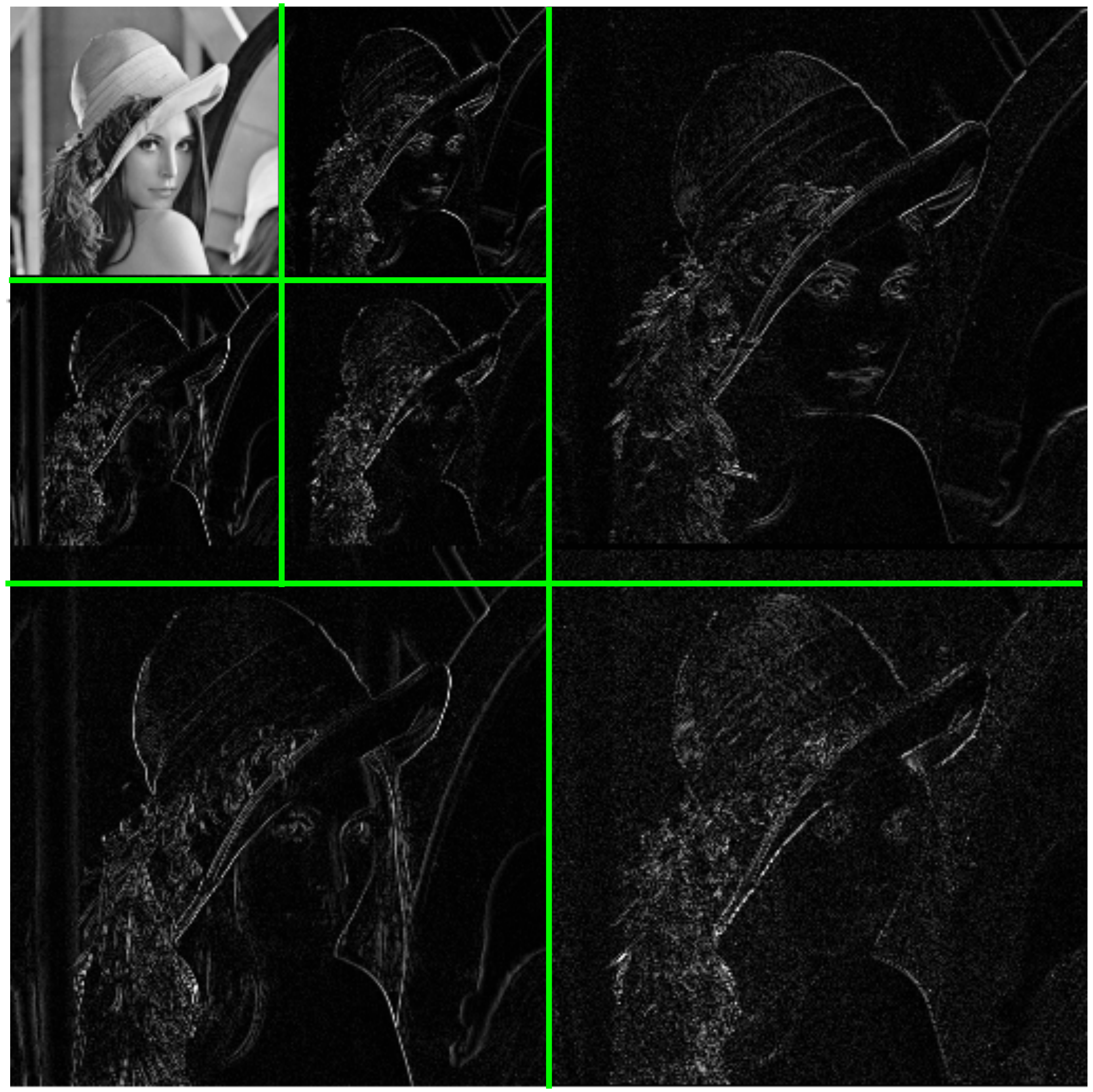


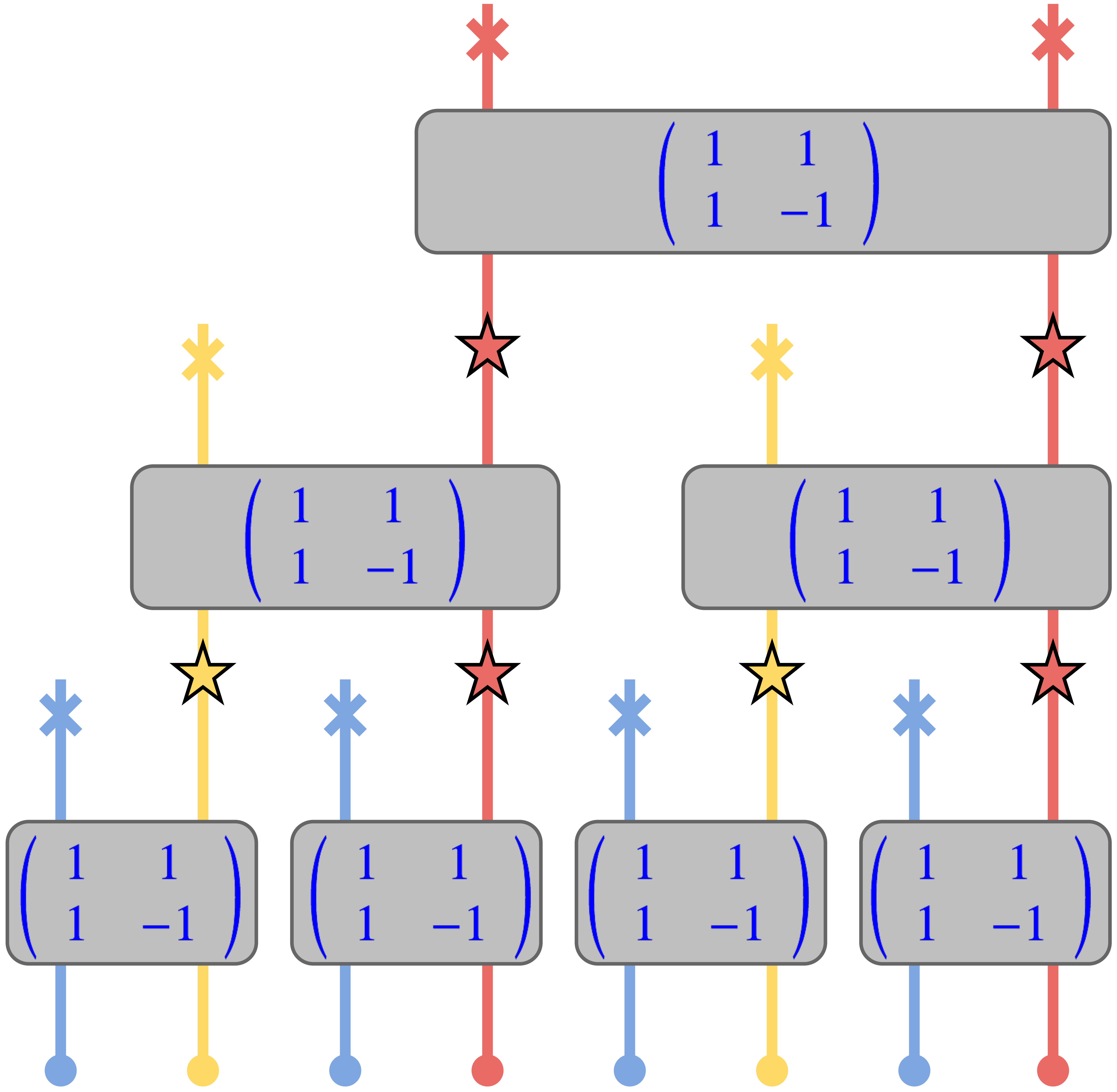
Entangled qubits



Multi-Scale
Entanglement
Renormalization
Ansatz

Connection to wavelets





Nonlinear & adaptive generalizations of wavelets

Guy, Wavelets & RG1999+ White, Evenbly, Qi, Wavelets, MERA, and holographic mapping 2013+

Continuous normalizing flows

$$\ln p(\boldsymbol{x}) = \ln \mathcal{N}(\boldsymbol{z}) - \ln \left| \det \left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}} \right) \right|$$

Consider infinitesimal change-of-variables Chen et al 1806.07366

$$\boldsymbol{x} = \boldsymbol{z} + \varepsilon \boldsymbol{v} \qquad\qquad \ln p(\boldsymbol{x}) - \ln \mathcal{N}(\boldsymbol{z}) = -\ln \left| \det \left(1 + \varepsilon \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{z}} \right) \right|$$

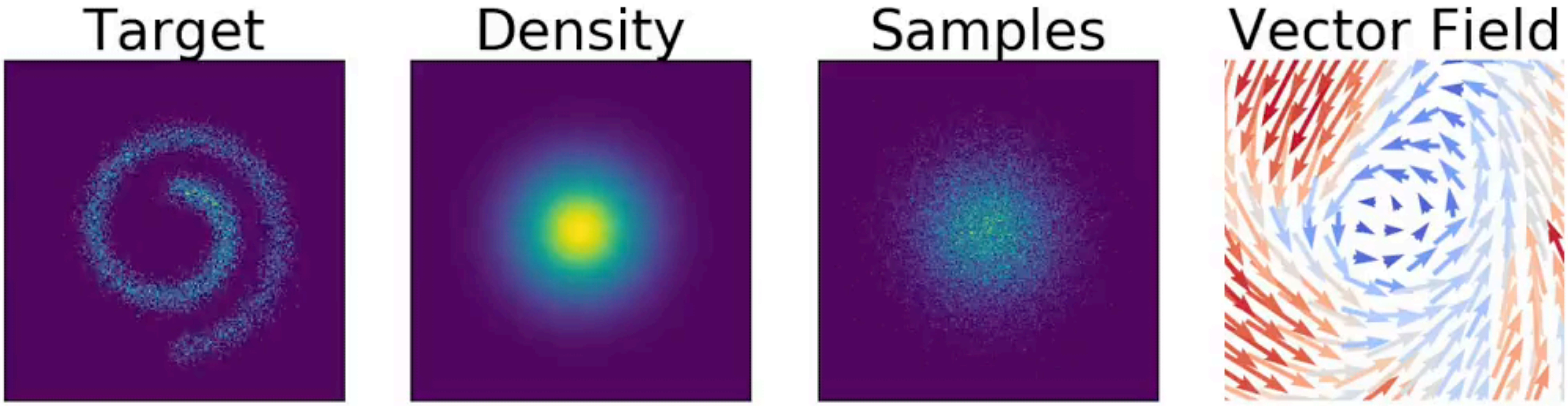
$\varepsilon \to 0$

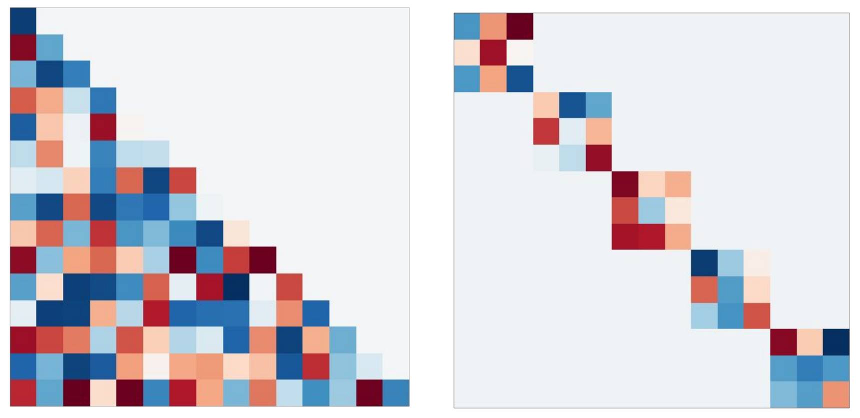
$t = T$ $\quad$ $t = 0$

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{v} \qquad\qquad \frac{d \ln \rho(\boldsymbol{x}, t)}{dt} = -\nabla \cdot \boldsymbol{v}$$

Continuous normalizing flows implemented with NeuralODE

Chen et al, 1806.07366, Grathwohl et al 1810.01367

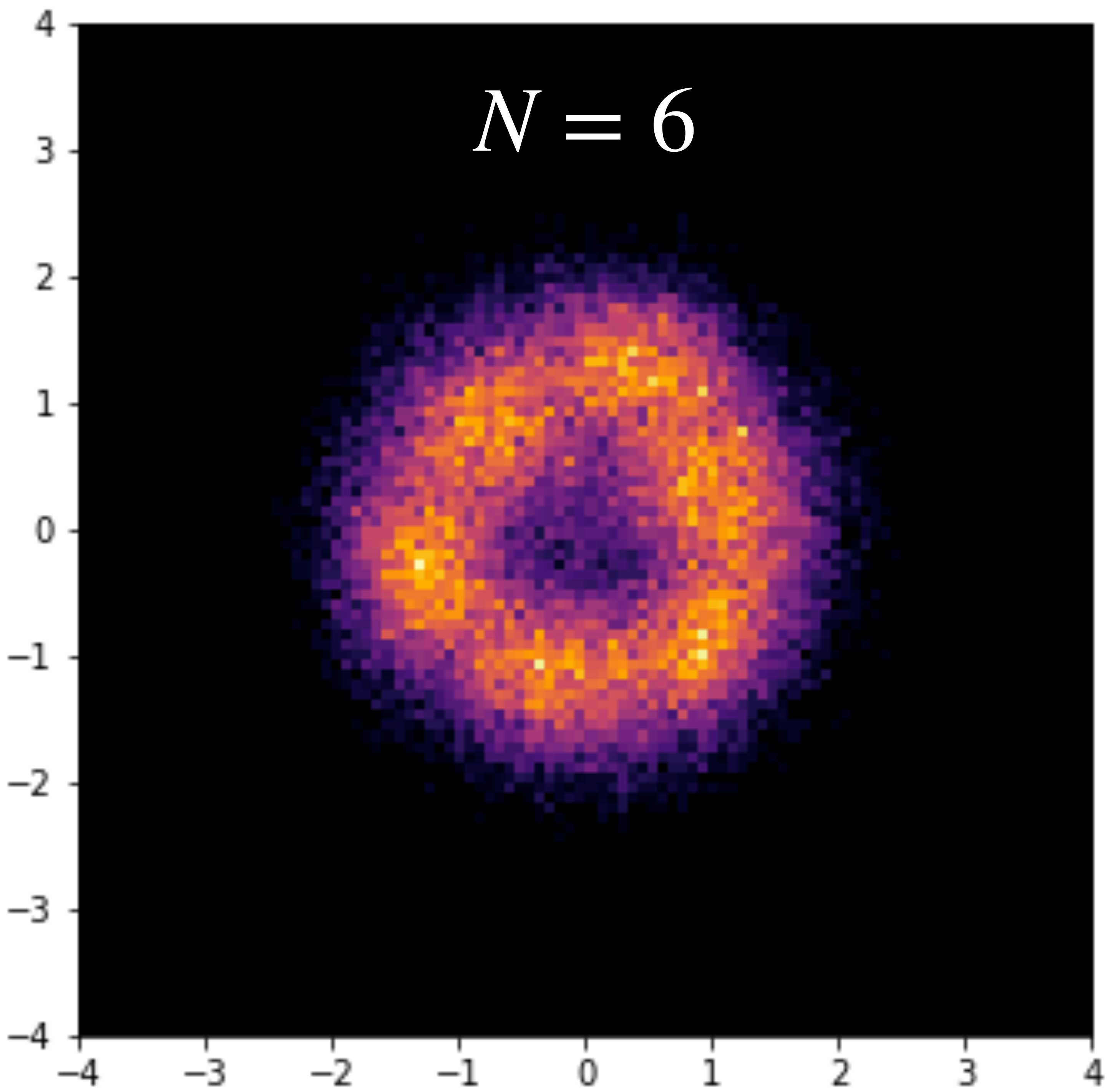


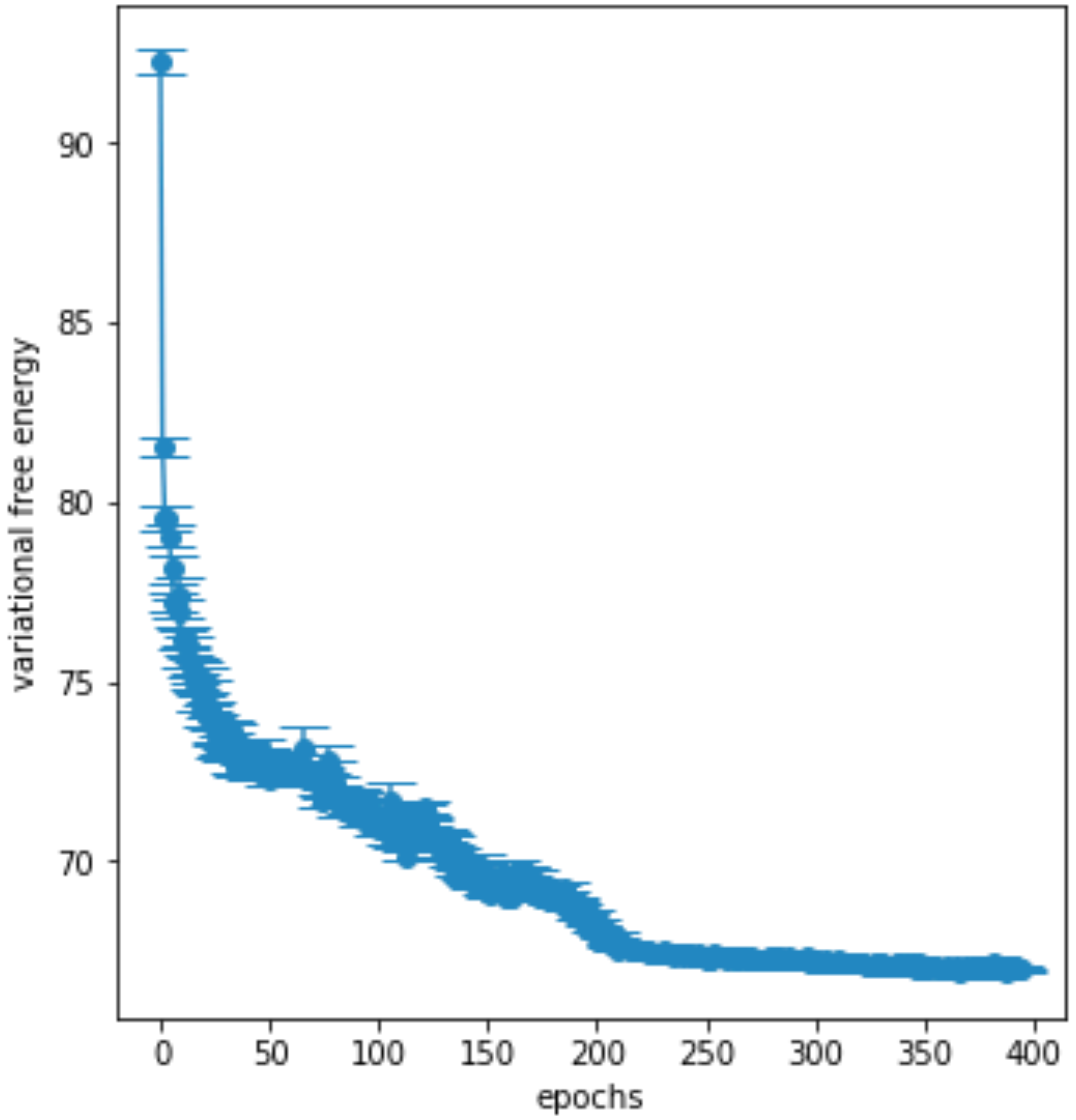


Continuous normalizing flows have no structural constraints on the transformation Jacobian

Demo: classical Coulomb gas in a parabolic trap

$$H = \sum_{i<j} \frac{1}{|\boldsymbol{x}_i - \boldsymbol{x}_j|} + \sum_i^N \frac{\boldsymbol{x}_i^2}{2} \qquad \boldsymbol{x} \sim e^{-\beta H}/Z$$





https://colab.research.google.com/drive/13wvsGtV4eTN4v6sXLf2Z52FCyCd6wQZP?usp=sharing

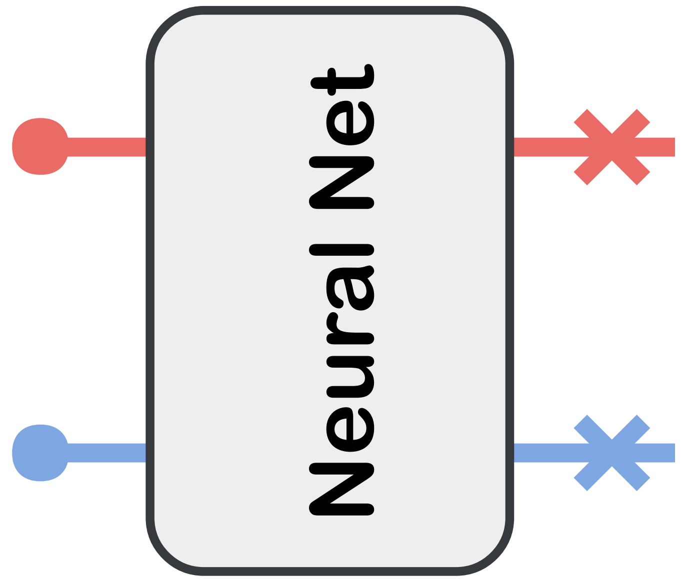
Two training approaches

Maximum likelihood estimation

"learn from data"

$$\mathscr{L} = -\mathbb{E}_{\boldsymbol{x}\sim\text{dataset}}\left[\ln p(\boldsymbol{x})\right]$$

$\boldsymbol{x} \sim \text{dataset}$



z

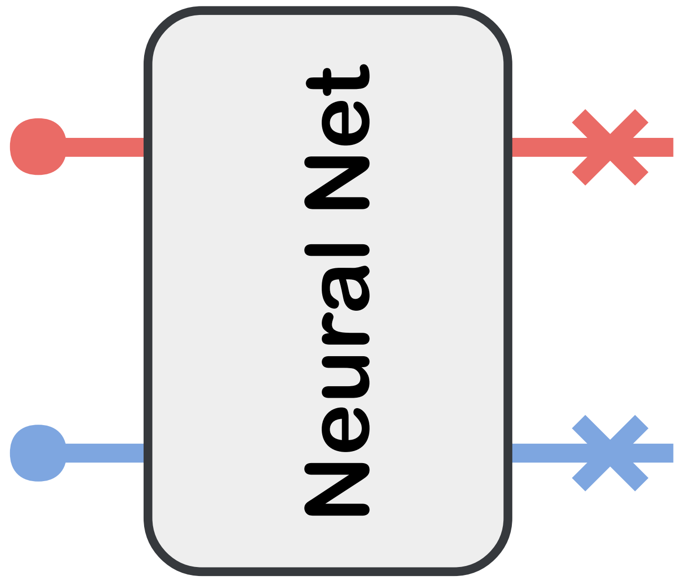
Sample from a given dataset

Variational calculation

"learn from Hamiltonian"

$$\mathscr{L} = \int d\boldsymbol{x}\, p(\boldsymbol{x})\left[\ln p(\boldsymbol{x}) + \beta H(\boldsymbol{x})\right]$$

$\boldsymbol{x}$



$z \sim \mathcal{N}(\mathbf{0}, \Sigma)$

Sample from your variational ansatz

Two training approaches

Maximum likelihood estimation

"learning from data"

$$\mathscr{L} = -\,\mathbb{E}_{\boxed{\boldsymbol{x}\sim\text{dataset}}}\left[\ln p(\boldsymbol{x})\right]$$

$$\mathbb{KL}(\pi\,||\,p) = \sum_{\boldsymbol{x}} \pi \ln \pi - \underbrace{\sum_{\boldsymbol{x}} \pi \ln p}_{\mathscr{L}}$$

Sample from a given dataset

Variational calculation

"learning from Hamiltonian"

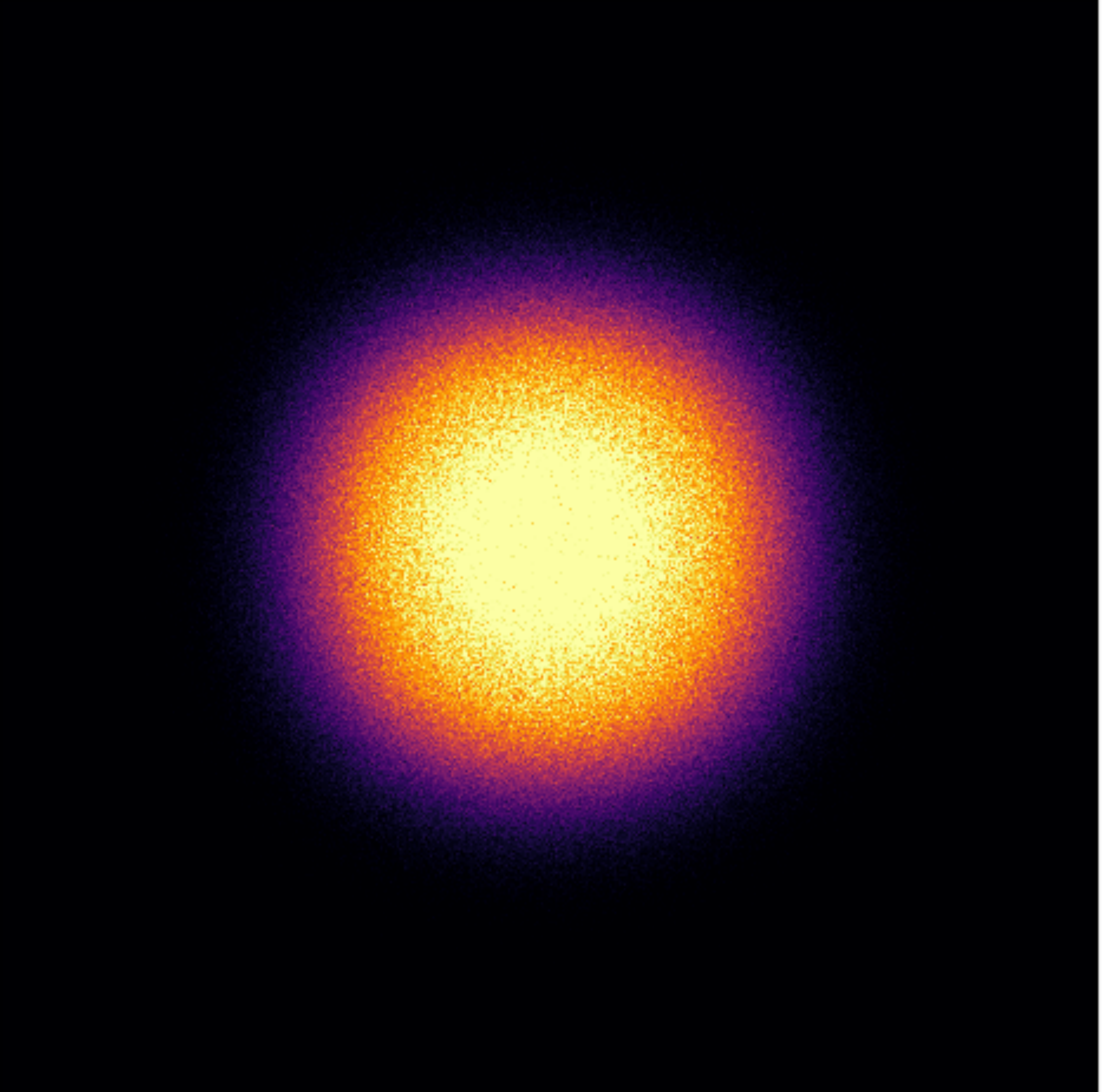
$$\mathscr{L} = \mathop{\mathbb{E}}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\left[\ln p(\boldsymbol{x}) + \beta \boxed{E(\boldsymbol{x})}\right]$$

$$\mathscr{L} + \ln Z = \mathbb{KL}\left(p\,||\,\frac{e^{-\beta E}}{Z}\right) \geq 0$$

Sample from your variational ansatz

Variational density matrices for quantum statistical mechanics

Classical	Quantum
Probability distribution p	Density matrix ρ
Kullback-Leibler divergence $\mathbb{KL}\left(p\,\|\,q\right)$	Quantum relative entropy $S\left(\rho\,\|\,\sigma\right)$
Variational free-energy $\mathscr{L} = \int d\boldsymbol{x}\, p(\boldsymbol{x})\left[\ln p(\boldsymbol{x}) + \beta H(\boldsymbol{x})\right]$	Variational free-energy $\mathscr{L} = \mathrm{Tr}(\rho \ln \rho) + \beta \mathrm{Tr}(H\rho)$



Fermi Flow: ab initio study of fermions at finite temperature

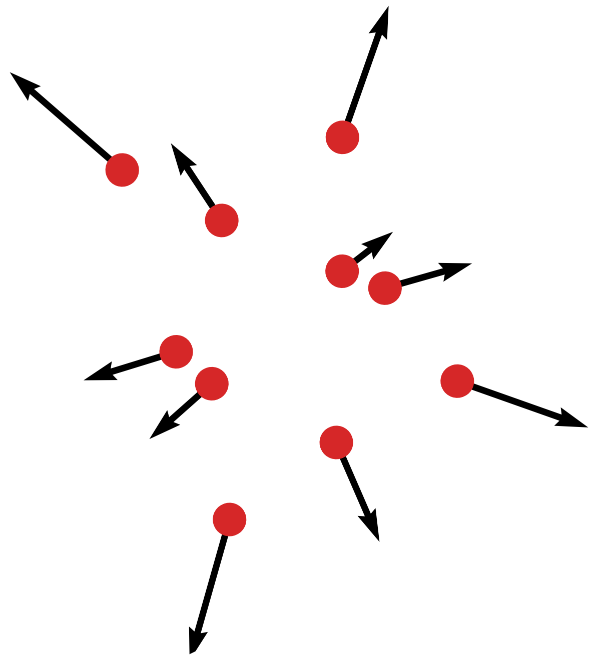
Xie, Zhang, LW, 2105.08644

https://github.com/buwantaiji/FermiFlow

Fluid physics behind flows

Zhang, E, LW 1809.10188
wangleiphy/MongeAmpereFlow

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{v}$$

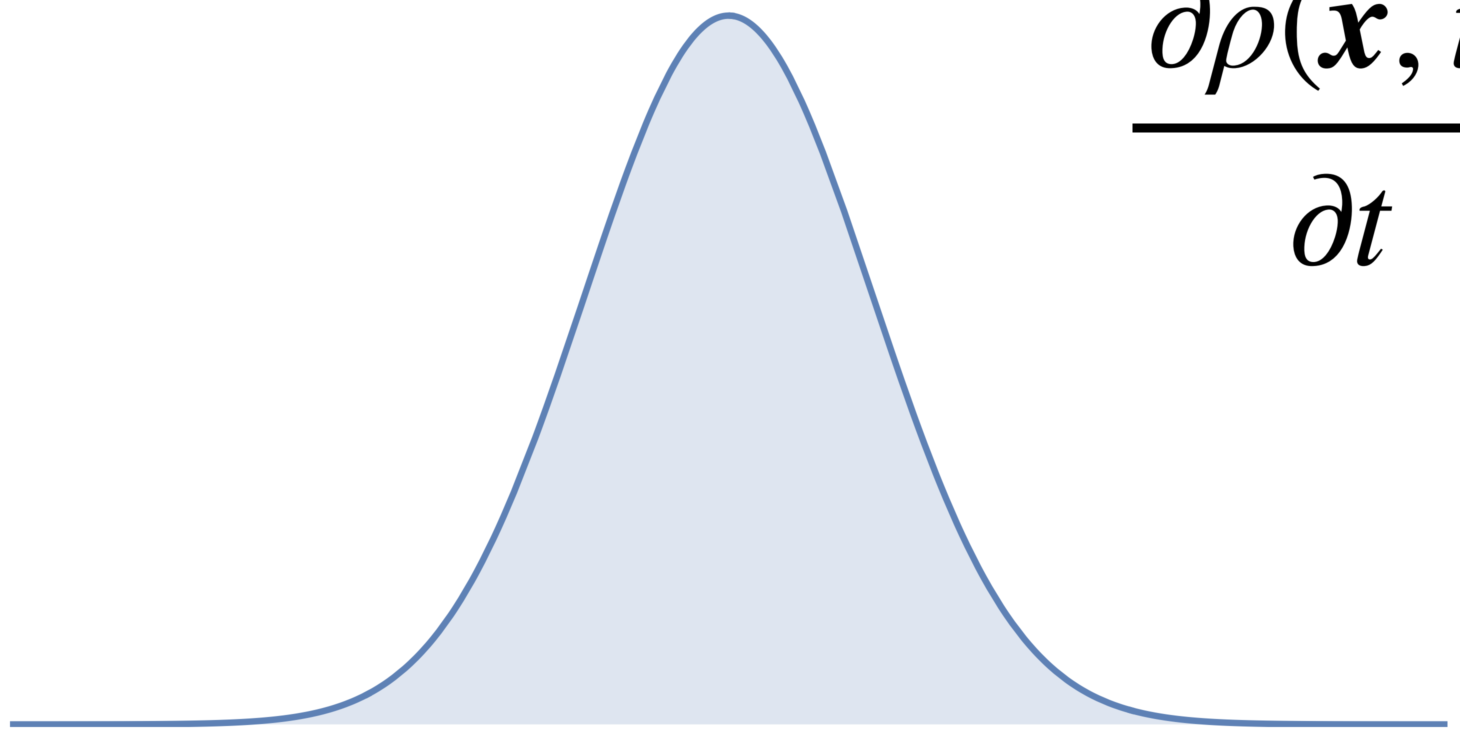


$$\frac{d \ln \rho(\boldsymbol{x}, t)}{dt} = -\nabla \cdot \boldsymbol{v}$$

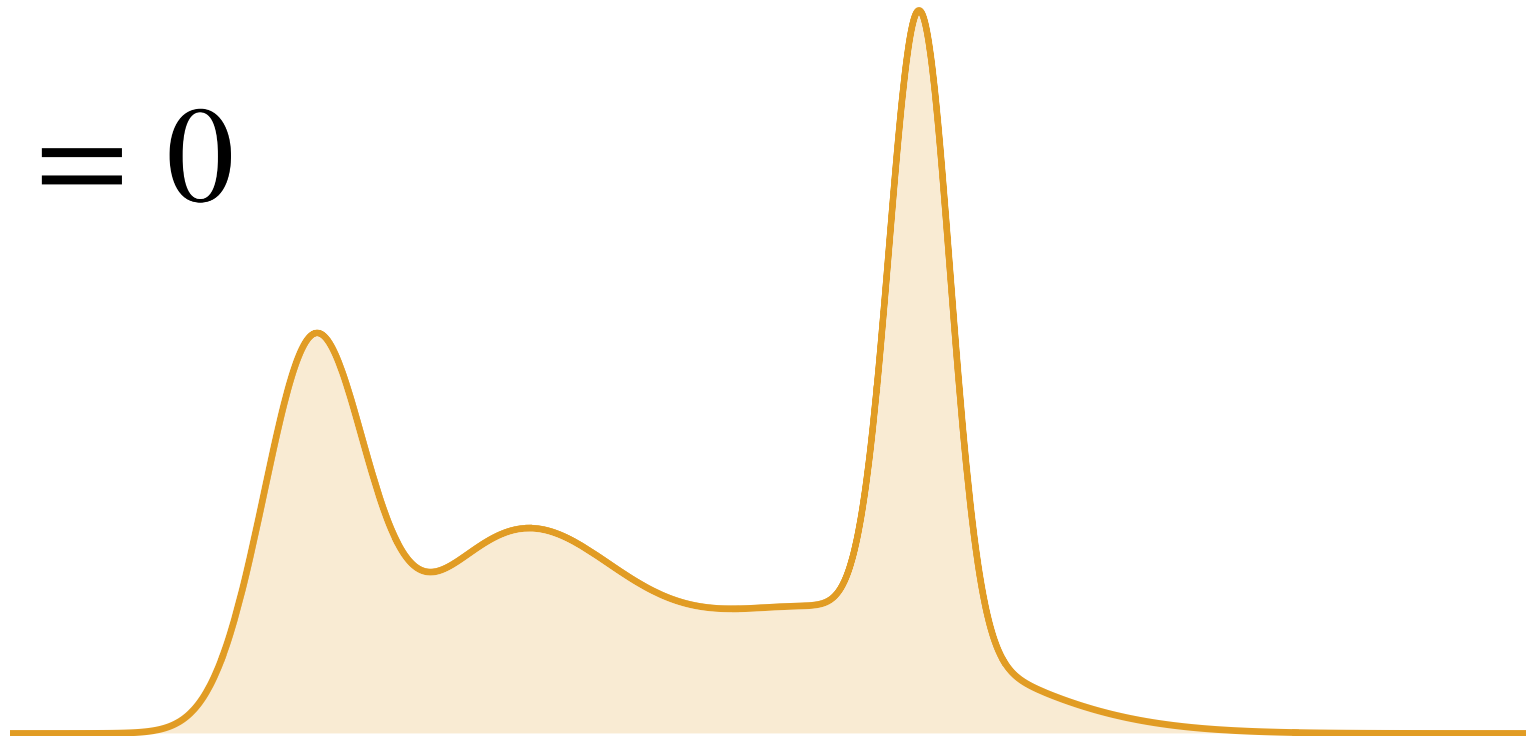
$$\frac{d}{dt} = \frac{\partial}{\partial t} + \boldsymbol{v} \cdot \nabla$$

"material derivative"

$$\frac{\partial \rho(\boldsymbol{x}, t)}{\partial t} + \nabla \cdot \left[\rho(\boldsymbol{x}, t)\boldsymbol{v}\right] = 0$$



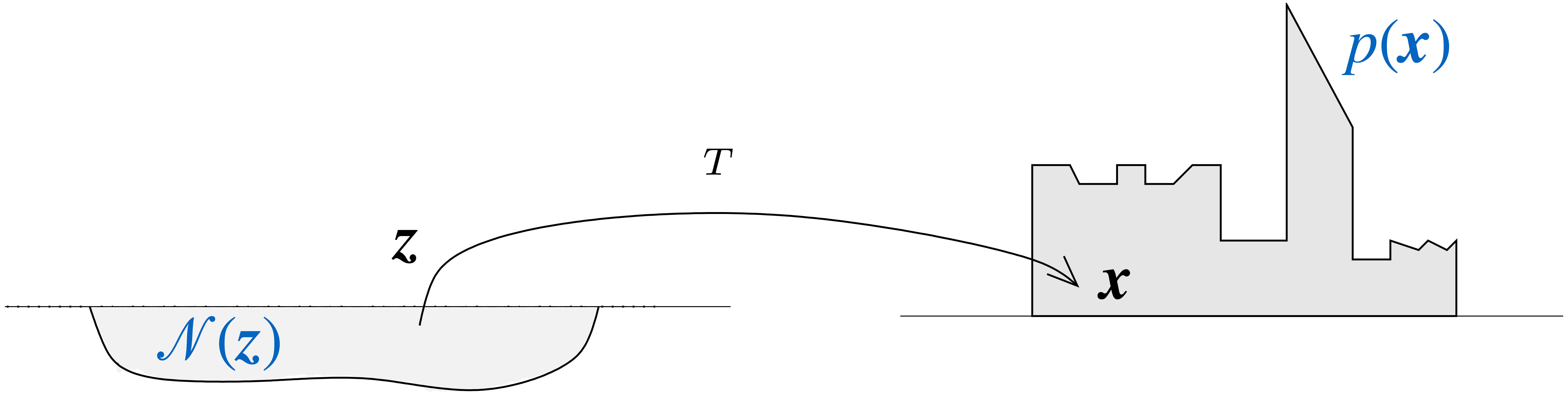
Simple density



Complex density

Optimal Transport Theory

Monge problem (1781): How to transport earth with optimal cost ?



Brenier theorem (1991)

Under certain conditions, the optimal map is

$$z \mapsto \boldsymbol{x} = \nabla u(z)$$

Monge | Kantorovich | Koopmans | Dantzig | Brenier | Otto | McCann | Villani | Figalli

Nobel Prize in Economics ’75

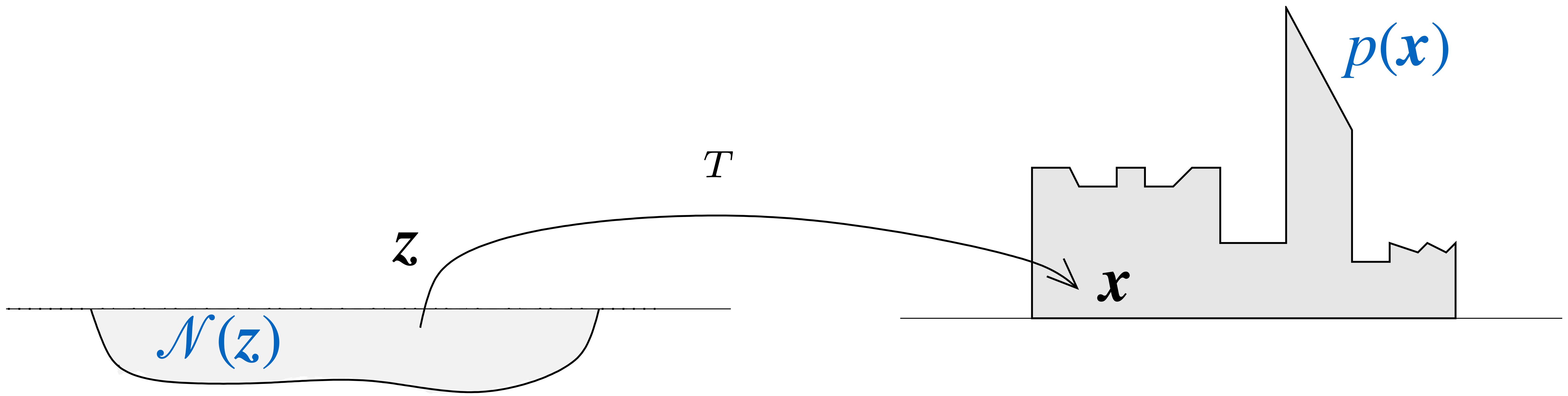
Fields Metal ’10

Fields Metal ’18

from Cuturi, Solomon NISP 2017 tutorial

Optimal Transport Theory

Monge problem (1781): How to transport earth with optimal cost ?



★ Brenier theorem (1991)

Under certain conditions
the optimal map is $\boxed{z \mapsto \boldsymbol{x} = \nabla u(z)}$

Monge-Ampère Equation

$$\boxed{\frac{\mathcal{N}(z)}{p(\nabla u(z))} = \det\left(\frac{\partial^2 u}{\partial z_i \partial z_j}\right)}$$

Monge-Ampère Flow

Zhang, E, LW 1809.10188

wangleiphy/MongeAmpereFlow

$$\frac{\partial \rho(\boldsymbol{x}, t)}{\partial t} + \nabla \cdot \left[\rho(\boldsymbol{x}, t) \nabla \varphi\right] = 0$$

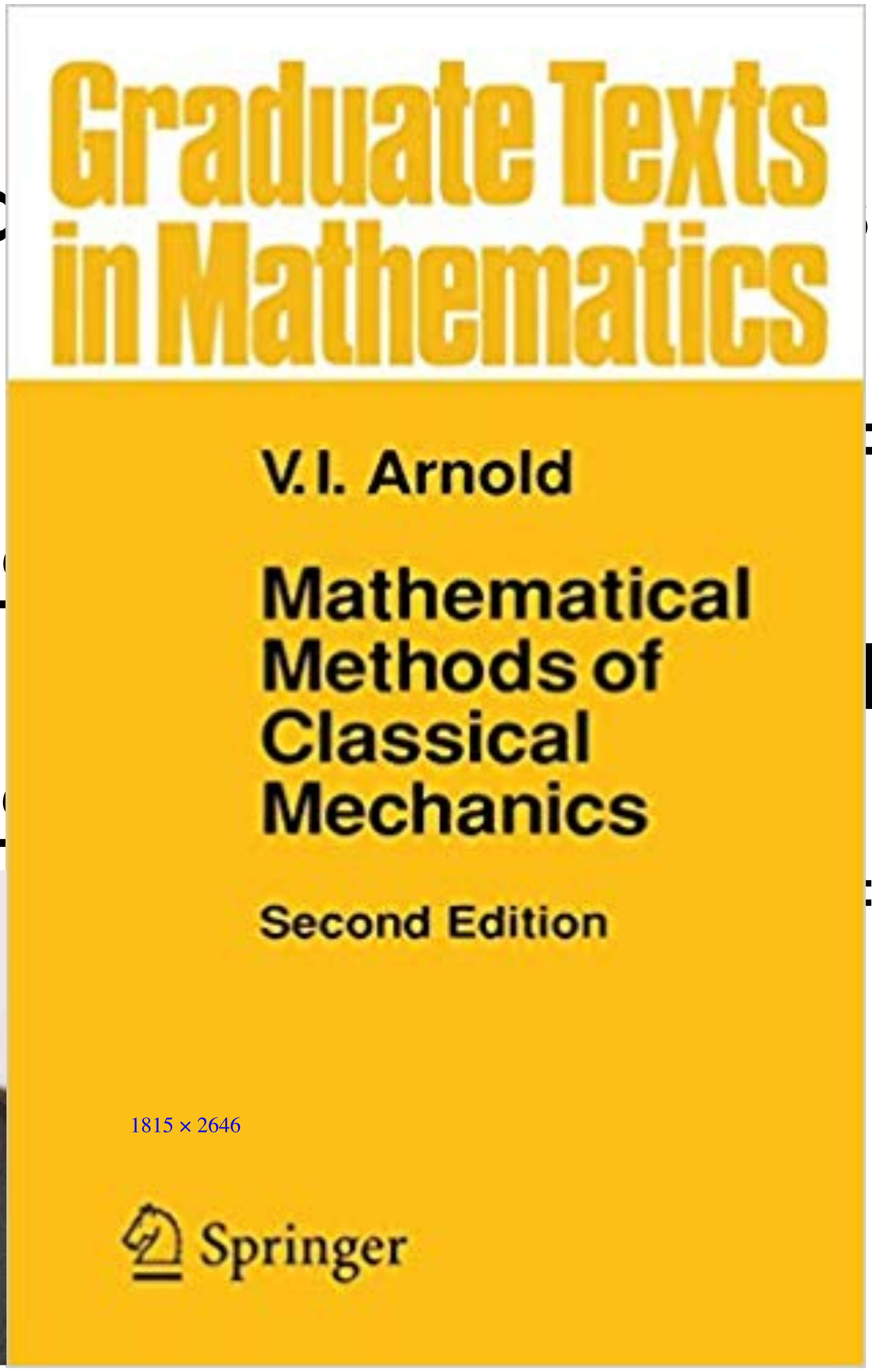
1. Drive the flow with an "irrotational" velocity field
2. Impose symmetry to the scalar valued potential for symmetric generative model

$$\varphi(g\,\boldsymbol{x}) = \varphi(\boldsymbol{x}) \quad \Longrightarrow \quad \rho(g\,\boldsymbol{x}) = \rho(\boldsymbol{x})$$

Flow in the phase space: Hamiltonian dynamics

Hamiltonian eq

$$\begin{cases} \dot{p} = - \\ \dot{q} = + \end{cases}$$

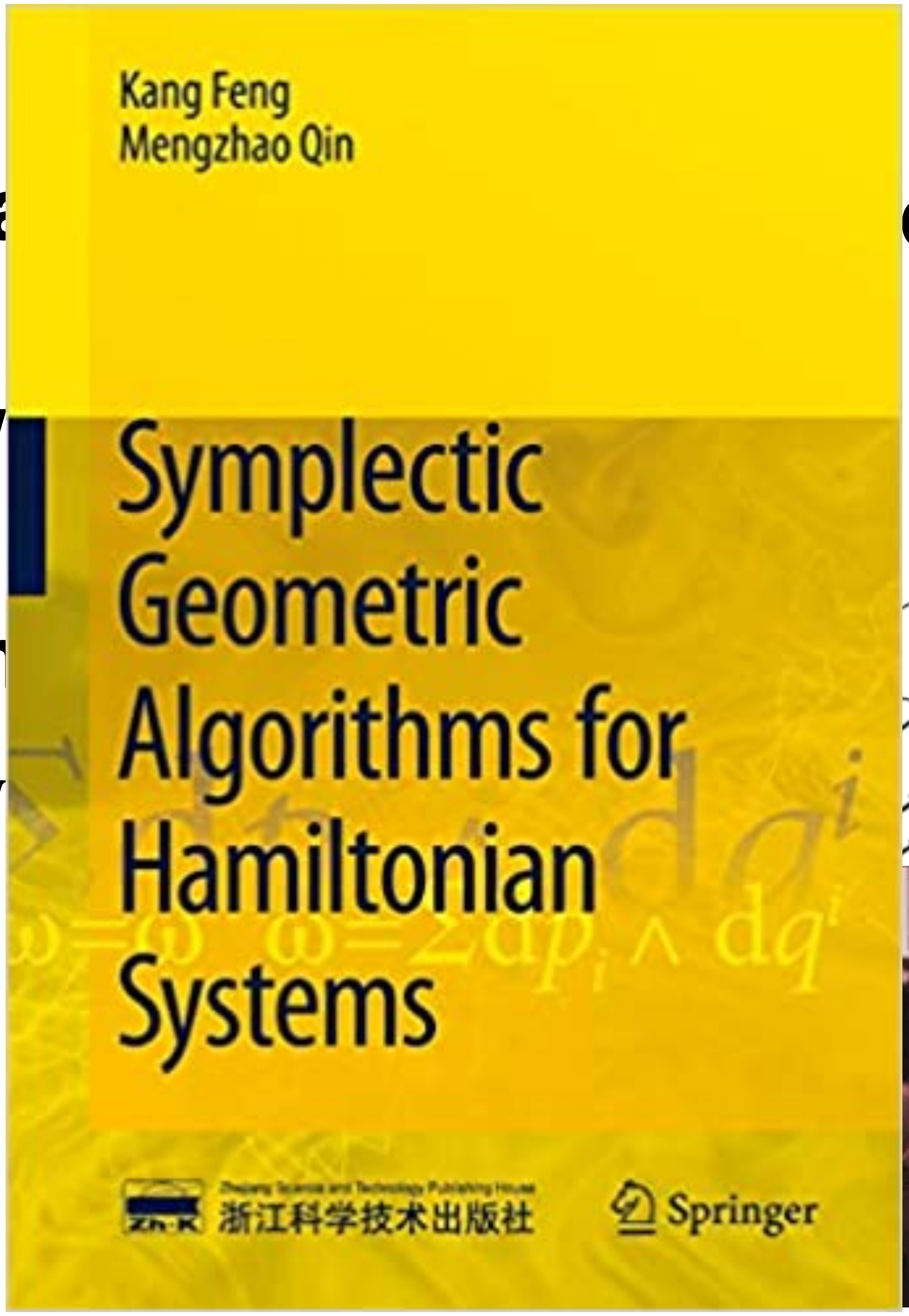


space va

$= (p, q$

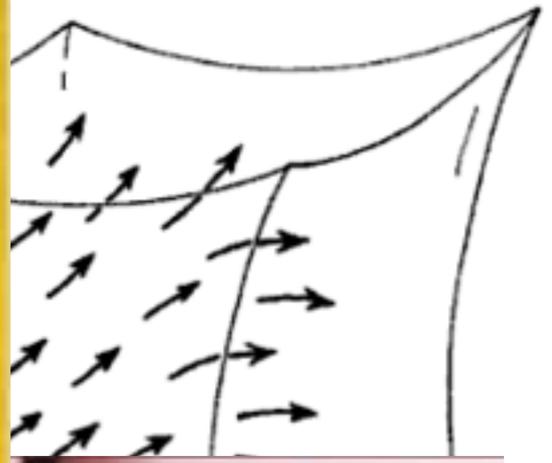
lectic m

$\begin{pmatrix} & I \\ -I & \end{pmatrix}$

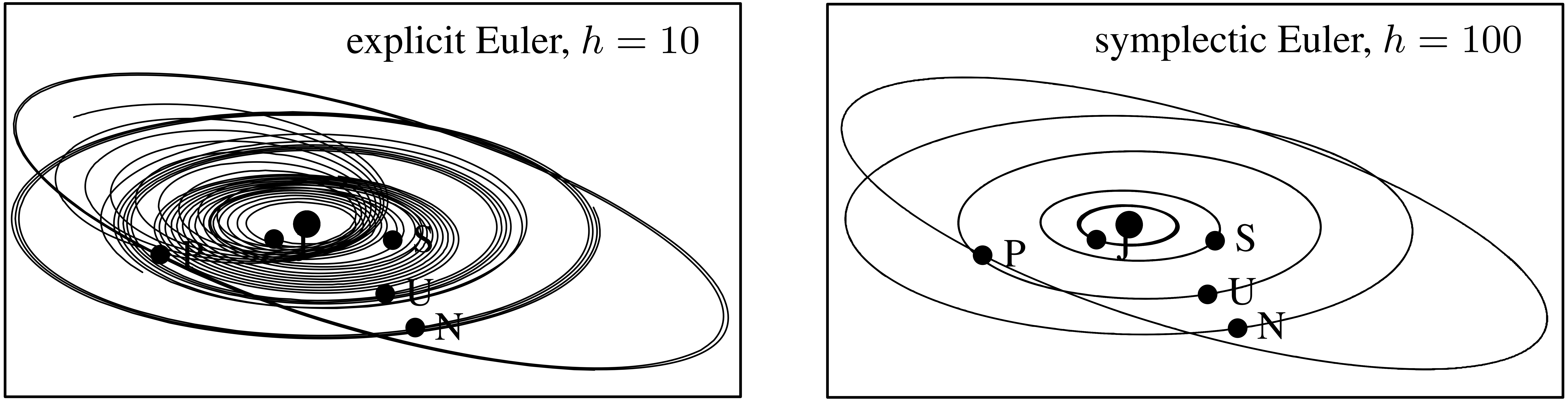


c gradient flow

$\nabla_x H(\boldsymbol{x}) J$



Symplectic Integrators



from Hairer et al, Geometric Numerical Integration

Canonical Transformations

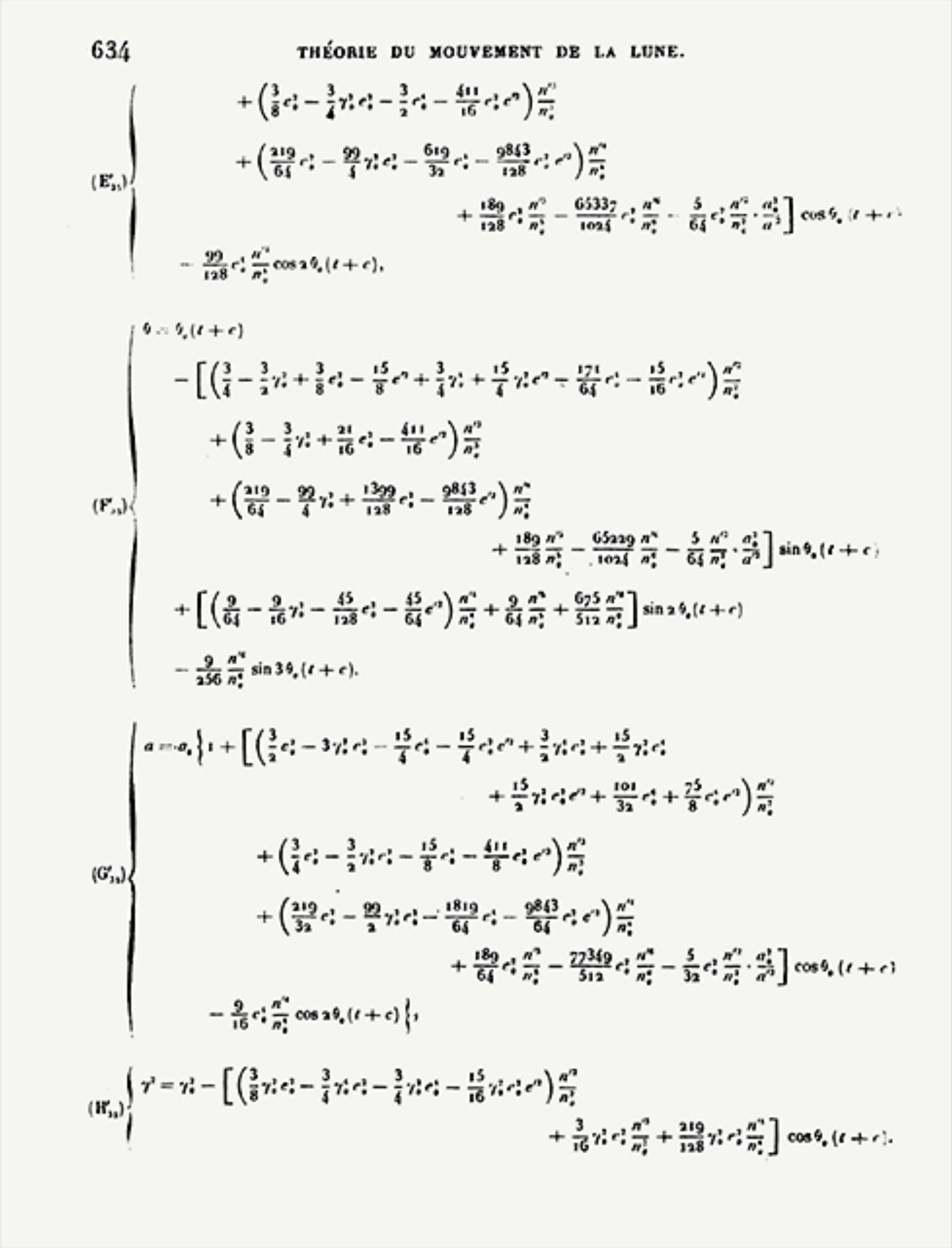
$$\boldsymbol{x} = (p, q) \xleftrightarrow{\text{Change of variables}} z = (P, Q)$$

which satisfies $\boxed{\left(\nabla_{\boldsymbol{x}} z\right) J \left(\nabla_{\boldsymbol{x}} z\right)^T = J}$ symplectic condition

one has $\dot{z} = \nabla_z K(z) J$ where $K(z) = H \circ \boldsymbol{x}(z)$

Preserves Hamiltonian dynamics in the "latent phase space"

Canonical transformation for Moon-Earth-Sun 3-body problem



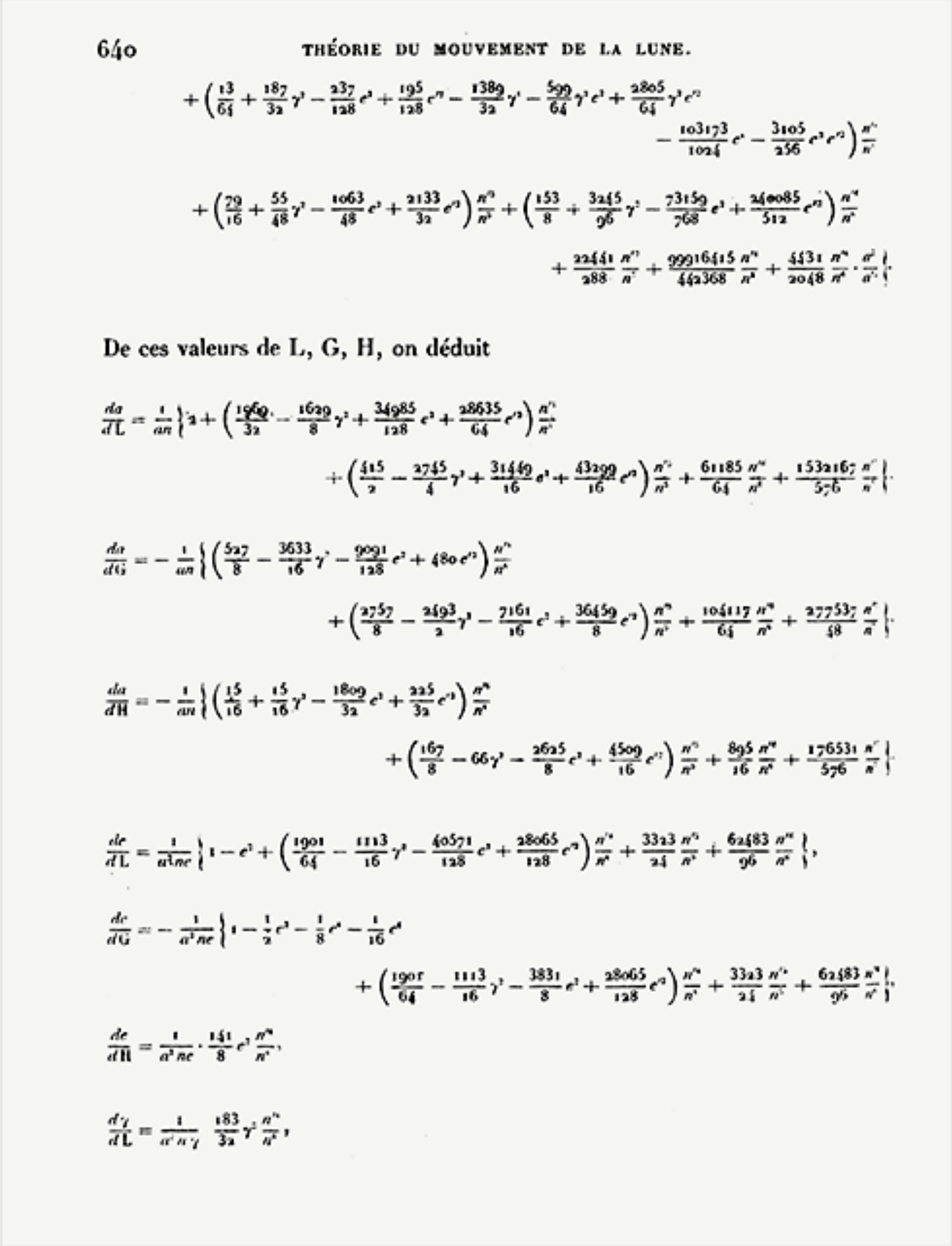
634 THÉORIE DU MOUVEMENT DE LA LUNE.

$$(E'_{11})\quad \begin{aligned} &+\left(\frac{3}{8}e_0^2-\frac{3}{4}\gamma_0^2e_0^2-\frac{3}{2}e_0^4-\frac{411}{16}e_0^2e'^2\right)\frac{n'^2}{n_0^2}\\ &+\left(\frac{219}{64}e_0^2-\frac{99}{4}\gamma_0^2e_0^2-\frac{619}{32}e_0^4-\frac{9843}{128}e_0^2e'^2\right)\frac{n'^3}{n_0^3}\\ &\qquad+\frac{189}{128}e_0^2\frac{n'^2}{n_0^2}-\frac{65337}{1024}e_0^2\frac{n'^4}{n_0^4}-\frac{5}{64}e_0^2\frac{n'^2}{n_0^2}\cdot\frac{a_0^2}{a'^2}\Big]\cos\theta_0(t+c)\\ &-\frac{99}{128}e_0^2\frac{n'^3}{n_0^3}\cos 2\theta_0(t+c), \end{aligned}$$

$$(F'_{11})\quad \begin{aligned} &\theta=\theta_0(t+c)\\ &-\Big[\left(\frac{3}{4}-\frac{3}{2}\gamma_0^2+\frac{3}{8}e_0^2-\frac{15}{8}e'^2+\frac{3}{4}\gamma_0^4+\frac{15}{4}\gamma_0^2e'^2-\frac{171}{64}e_0^4-\frac{15}{16}e_0^2e'^2\right)\frac{n'^2}{n_0^2}\\ &+\left(\frac{3}{8}-\frac{3}{4}\gamma_0^2+\frac{21}{16}e_0^2-\frac{411}{16}e'^2\right)\frac{n'^3}{n_0^3}\\ &+\left(\frac{219}{64}-\frac{99}{4}\gamma_0^2+\frac{1399}{128}e_0^2-\frac{9843}{128}e'^2\right)\frac{n'^4}{n_0^4}\\ &\qquad+\frac{189}{128}\frac{n'^3}{n_0^3}-\frac{65229}{1024}\frac{n'^5}{n_0^5}-\frac{5}{64}\frac{n'^2}{n_0^2}\cdot\frac{a_0^2}{a'^2}\Big]\sin\theta_0(t+c)\\ &+\Big[\left(\frac{9}{64}-\frac{9}{16}\gamma_0^2-\frac{45}{128}e_0^2-\frac{45}{64}e'^2\right)\frac{n'^4}{n_0^4}+\frac{9}{64}\frac{n'^5}{n_0^5}+\frac{675}{512}\frac{n'^6}{n_0^6}\Big]\sin 2\theta_0(t+c)\\ &-\frac{9}{256}\frac{n'^6}{n_0^6}\sin 3\theta_0(t+c). \end{aligned}$$

$$(G'_{11})\quad \begin{aligned} &a=a_0\Big\{1+\Big[\Big(\frac{3}{2}e_0^2-3\gamma_0^2e_0^2-\frac{15}{4}e_0^4-\frac{15}{4}e_0^2e'^2+\frac{3}{2}\gamma_0^4e_0^2+\frac{15}{2}\gamma_0^2e_0^4\\ &\qquad+\frac{15}{2}\gamma_0^2e_0^2e'^2+\frac{101}{32}e_0^6+\frac{75}{8}e_0^4e'^2\Big)\frac{n'^2}{n_0^2}\\ &+\left(\frac{3}{4}e_0^2-\frac{3}{2}\gamma_0^2e_0^2-\frac{15}{8}e_0^4-\frac{411}{8}e_0^2e'^2\right)\frac{n'^3}{n_0^3}\\ &+\left(\frac{219}{32}e_0^2-\frac{99}{2}\gamma_0^2e_0^2-\frac{1819}{64}e_0^4-\frac{9843}{64}e_0^2e'^2\right)\frac{n'^4}{n_0^4}\\ &\qquad+\frac{189}{64}e_0^2\frac{n'^3}{n_0^3}-\frac{77349}{512}e_0^2\frac{n'^5}{n_0^5}-\frac{5}{32}e_0^2\frac{n'^2}{n_0^2}\cdot\frac{a_0^2}{a'^2}\Big]\cos\theta_0(t+c)\\ &-\frac{9}{16}e_0^2\frac{n'^4}{n_0^4}\cos 2\theta_0(t+c)\Big\}, \end{aligned}$$

$$(H'_{11})\quad \begin{aligned} &\gamma^2=\gamma_0^2-\Big[\left(\frac{3}{8}\gamma_0^2e_0^2-\frac{3}{4}\gamma_0^4e_0^2-\frac{3}{4}\gamma_0^2e_0^4-\frac{15}{16}\gamma_0^2e_0^2e'^2\right)\frac{n'^2}{n_0^2}\\ &\qquad+\frac{3}{16}\gamma_0^2e_0^2\frac{n'^3}{n_0^3}+\frac{219}{128}\gamma_0^2e_0^2\frac{n'^4}{n_0^4}\Big]\cos\theta_0(t+c). \end{aligned}$$



640 THÉORIE DU MOUVEMENT DE LA LUNE.

$$\begin{aligned} &+\Big(\frac{13}{64}+\frac{187}{32}\gamma^2-\frac{237}{128}e^2+\frac{195}{128}e'^2-\frac{1389}{32}\gamma^4-\frac{599}{64}\gamma^2e^2+\frac{2805}{64}\gamma^2e'^2\\ &\qquad-\frac{103173}{1024}e^4-\frac{3105}{256}e^2e'^2\Big)\frac{n'^2}{n^2}\\ &+\left(\frac{79}{16}+\frac{55}{48}\gamma^2-\frac{1063}{48}e^2+\frac{2133}{32}e'^2\right)\frac{n'^3}{n^3}+\left(\frac{153}{8}+\frac{3245}{96}\gamma^2-\frac{73159}{768}e^2+\frac{240085}{512}e'^2\right)\frac{n'^4}{n^4}\\ &\qquad+\frac{22441}{288}\frac{n'^5}{n^5}+\frac{99916415}{442368}\frac{n'^6}{n^6}+\frac{4431}{2048}\frac{n'^2}{n^2}\cdot\frac{a^2}{a'^2}\Big\}. \end{aligned}$$

De ces valeurs de L, G, H, on déduit

$$\begin{aligned} \frac{da}{dL}&=\frac{1}{an}\Big\{2+\left(\frac{1969}{32}-\frac{1629}{8}\gamma^2+\frac{34985}{128}e^2+\frac{28635}{64}e'^2\right)\frac{n'^2}{n^2}\\ &\qquad+\left(\frac{415}{2}-\frac{2745}{4}\gamma^2+\frac{31449}{16}e^2+\frac{43299}{16}e'^2\right)\frac{n'^3}{n^3}+\frac{61185}{64}\frac{n'^4}{n^4}+\frac{153216 7}{576}\frac{n'^5}{n^5}\Big\}.\\ \frac{da}{dG}&=-\frac{1}{an}\Big\{\left(\frac{527}{8}-\frac{3633}{16}\gamma^2-\frac{9091}{128}e^2+480e'^2\right)\frac{n'^2}{n^2}\\ &\qquad+\left(\frac{2757}{8}-\frac{2493}{2}\gamma^2-\frac{7161}{16}e^2+\frac{36459}{8}e'^2\right)\frac{n'^3}{n^3}+\frac{104117}{64}\frac{n'^4}{n^4}+\frac{277537}{48}\frac{n'^5}{n^5}\Big\}.\\ \frac{da}{dH}&=-\frac{1}{an}\Big\{\left(\frac{15}{16}+\frac{15}{16}\gamma^2-\frac{1809}{32}e^2+\frac{225}{32}e'^2\right)\frac{n'^2}{n^2}\\ &\qquad+\left(\frac{167}{8}-66\gamma^2-\frac{2625}{8}e^2+\frac{4509}{16}e'^2\right)\frac{n'^3}{n^3}+\frac{895}{16}\frac{n'^4}{n^4}+\frac{176531}{576}\frac{n'^5}{n^5}\Big\}.\\ \frac{de}{dL}&=\frac{1}{a^2ne}\Big\{1-e^2+\left(\frac{1901}{64}-\frac{1113}{16}\gamma^2-\frac{40571}{128}e^2+\frac{28065}{128}e'^2\right)\frac{n'^2}{n^2}+\frac{3323}{24}\frac{n'^3}{n^3}+\frac{62483}{96}\frac{n'^4}{n^4}\Big\},\\ \frac{de}{dG}&=-\frac{1}{a^2ne}\Big\{1-\frac{1}{2}e^2-\frac{1}{8}e^4-\frac{1}{16}e^6\\ &\qquad+\left(\frac{1901}{64}-\frac{1113}{16}\gamma^2-\frac{3831}{8}e^2+\frac{28065}{128}e'^2\right)\frac{n'^2}{n^2}+\frac{3323}{24}\frac{n'^3}{n^3}+\frac{62483}{96}\frac{n'^4}{n^4}\Big\},\\ \frac{de}{dH}&=\frac{1}{a^2ne}\cdot\frac{141}{8}e^2\frac{n'^2}{n^2},\\ \frac{d\gamma}{dL}&=\frac{1}{a^2n\gamma}\ \frac{183}{32}\gamma^2\frac{n'^2}{n^2}, \end{aligned}$$

Charles Delaunay

↑

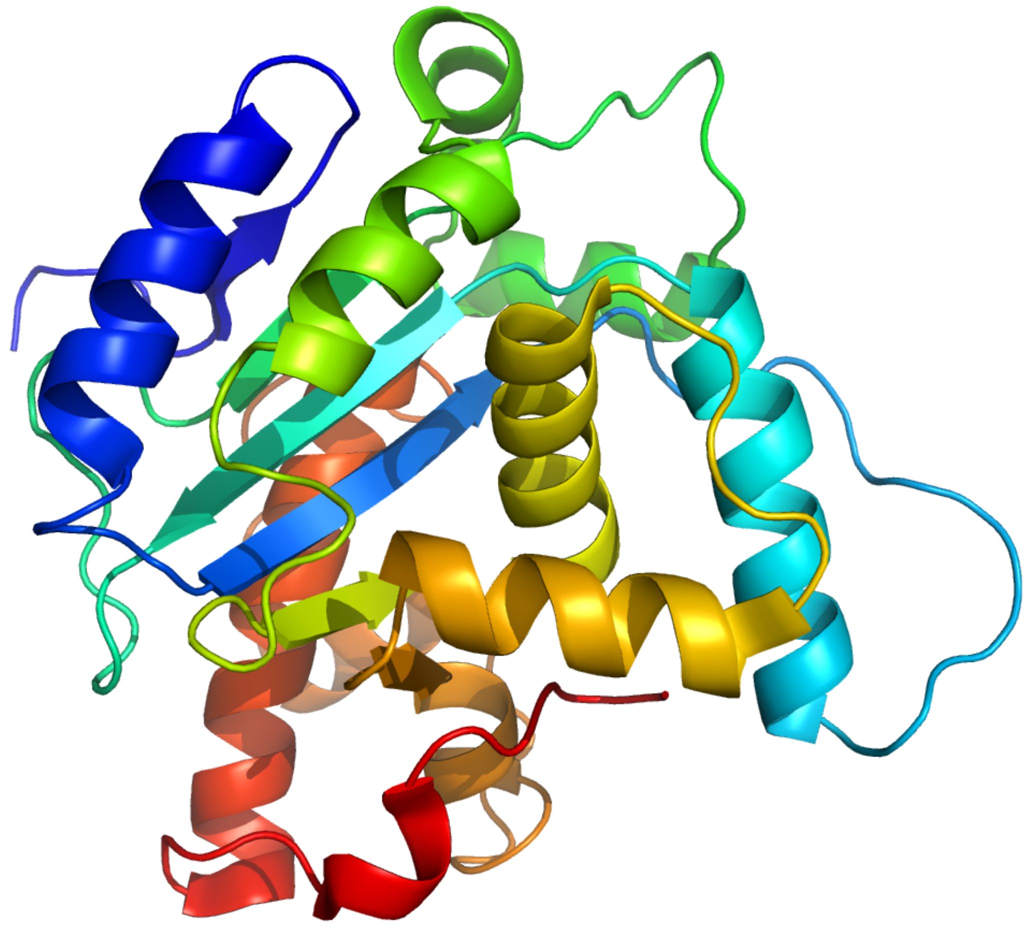
More than 1800 pages of this, ~20 years of efforts (1846-1867)

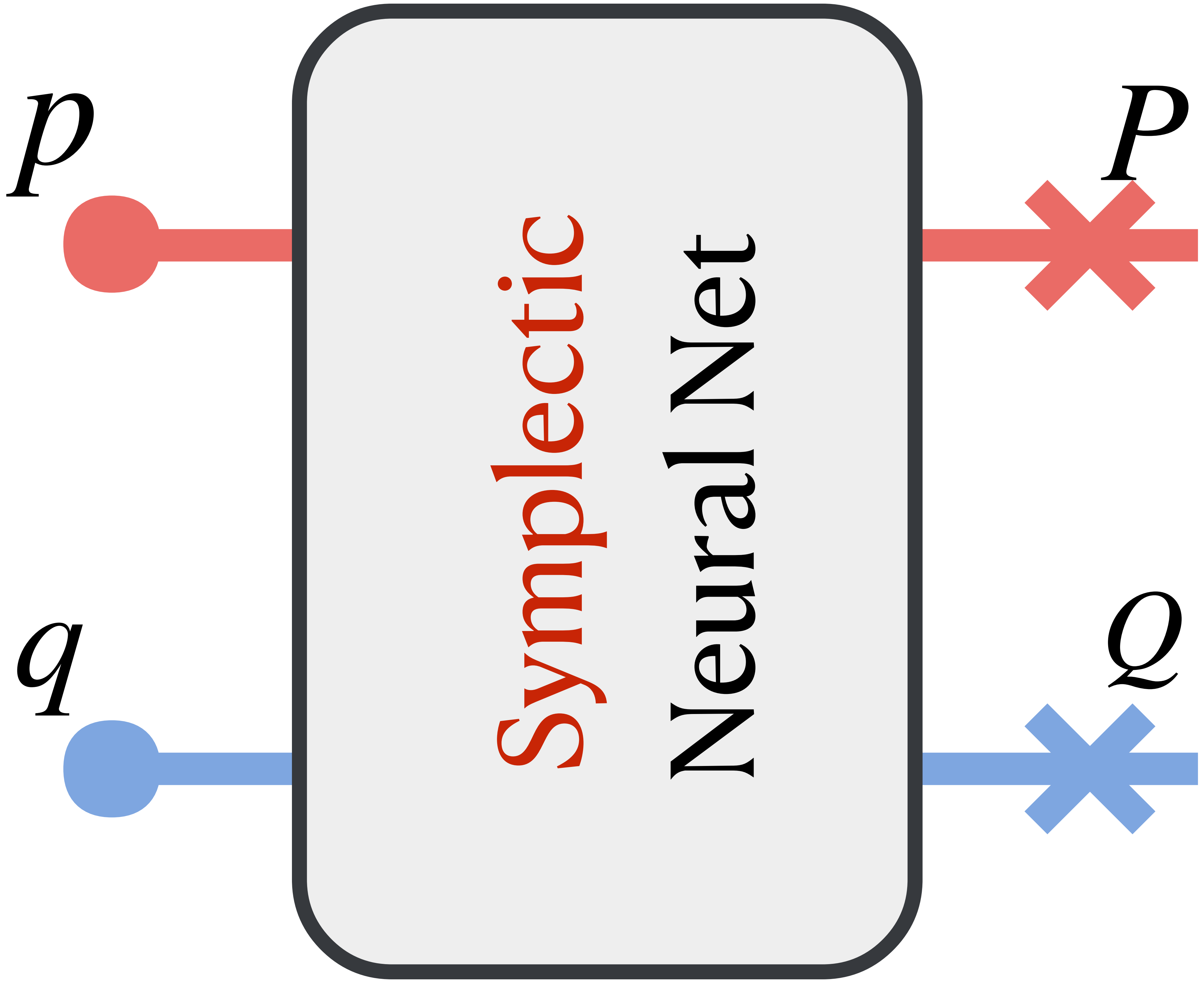
Neural Canonical Transformations

Li, Dong, Zhang, LW, PRX '20 li012589/neuralCT

$H(p, q)$

$$K(P, Q) = \sum_k \frac{P_k^2 + \omega_k^2 Q_k^2}{2}$$





physical space

latent space

Learn the network parameter and the latent harmonic frequency

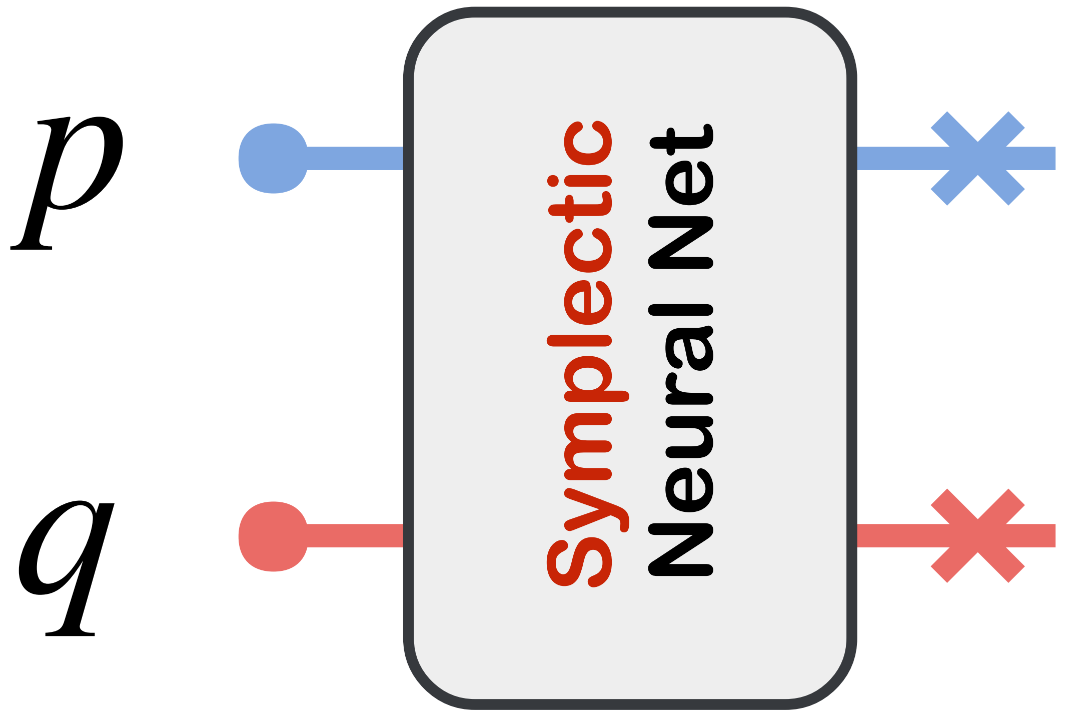
Symplectic primitives

- Linear transformation: Symplectic Lie algebra

- Continuous-time flow: Symplectic generating functions

Symplectic integrator of neural ODE, Chen et al 1806.07366

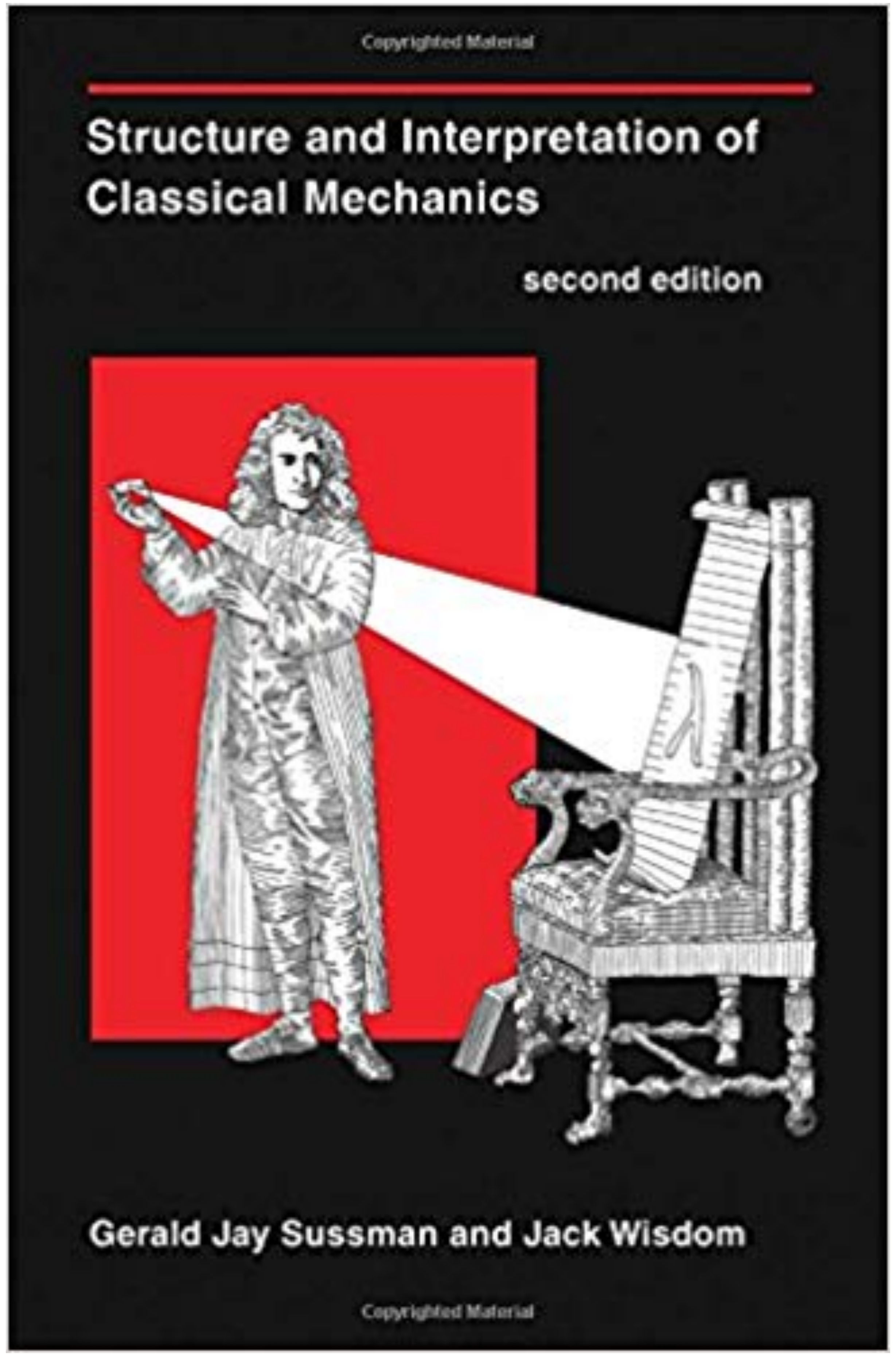
- **Neural point transformation**



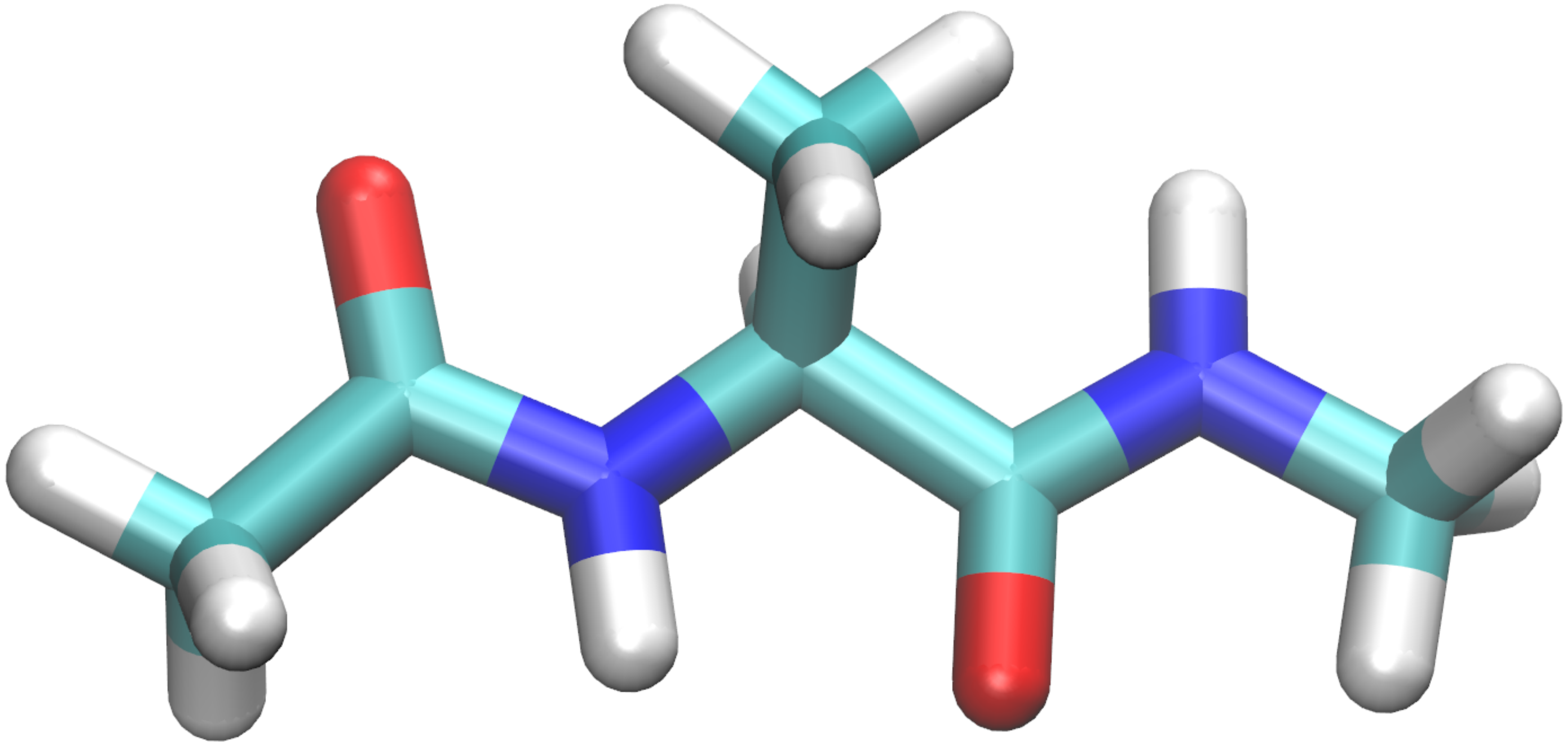
$$P = p\left(\nabla_Q q\right)$$

$$Q = f(q)$$

invertible neural net

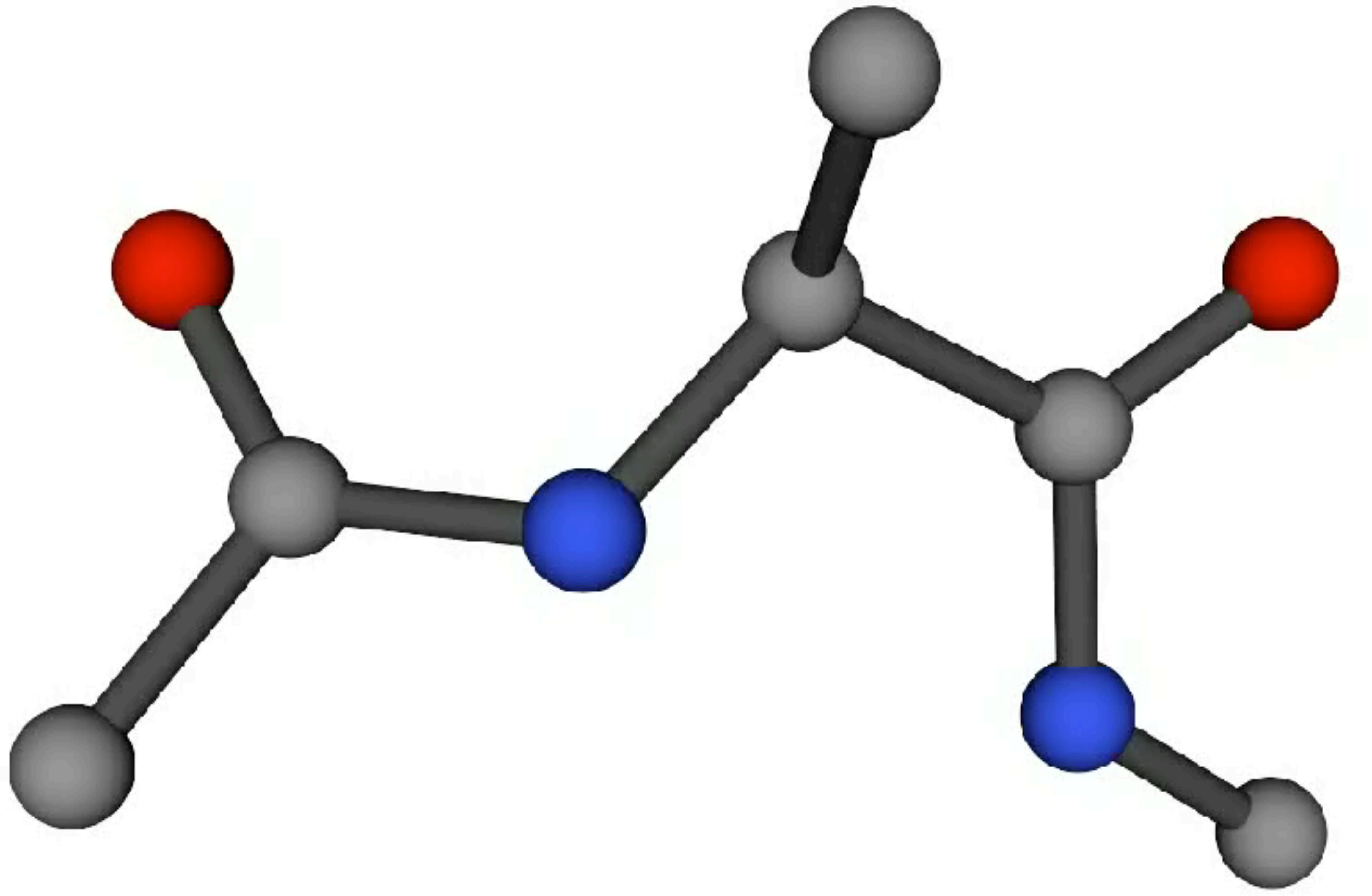


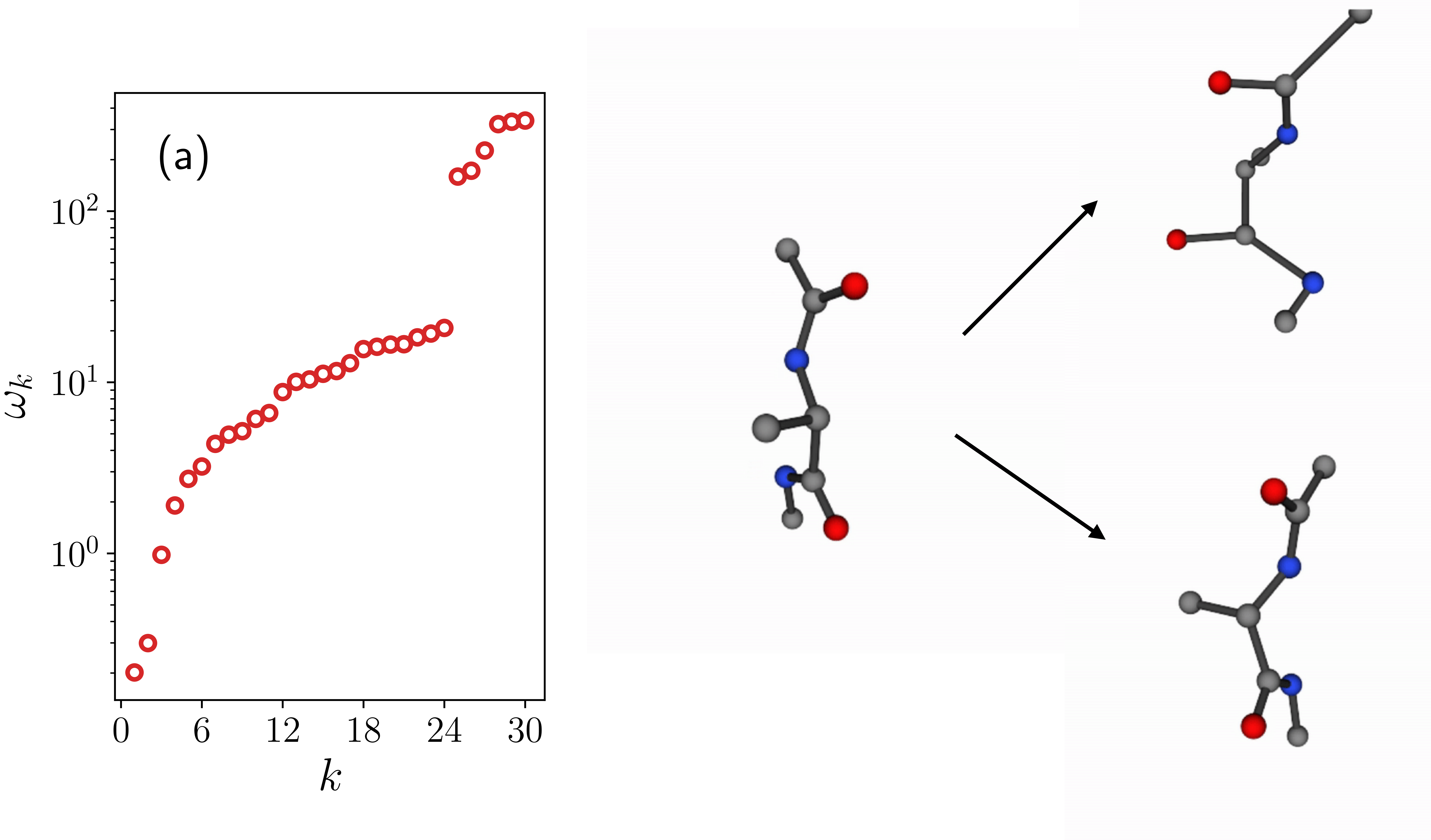
Application: identifying slow modes



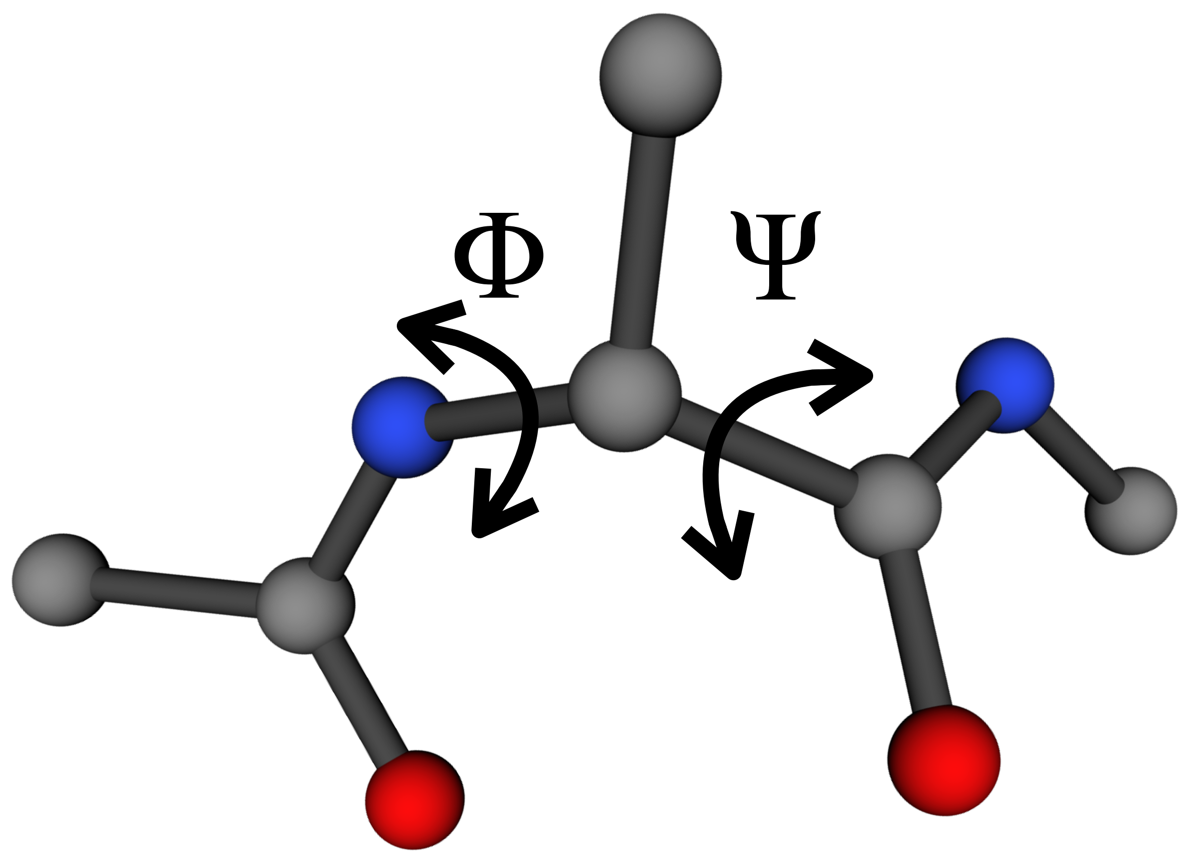
Data: 250 ns molecular dynamics simulation of alanine dipeptide at 300 K
https://markovmodel.github.io/mdshare/ALA2/#alanine-dipeptide

More than 3 hours of video ...

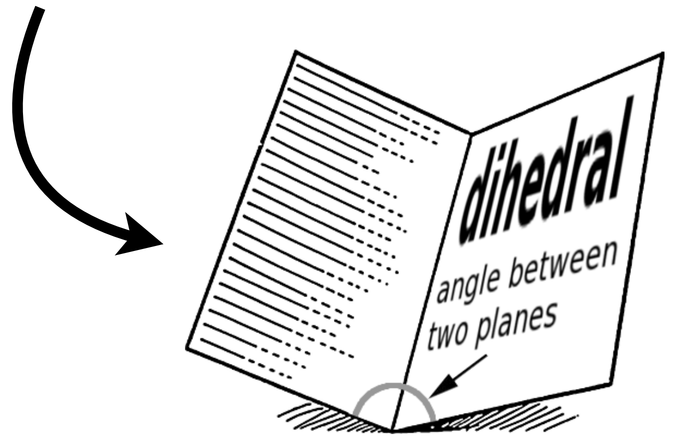


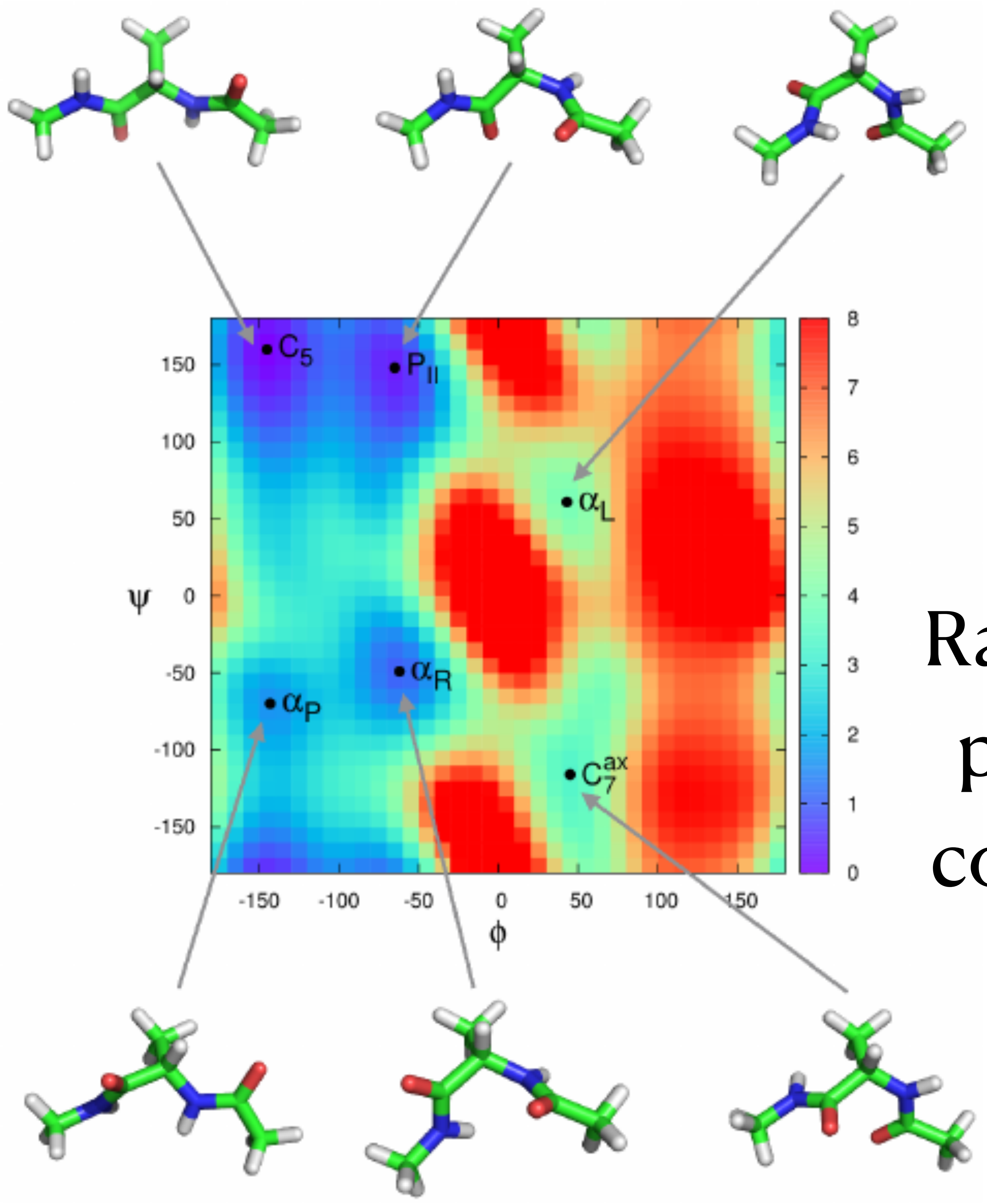


Neural canonical transformation decomposes nonlinear slow modes



slow motion of the two torsion angles





Ramachandran plot of stable conformations

Dimensional reduction to slow collective variables useful for control, prediction, enhanced sampling...

check the paper 1910.00024, PRX '20 for more examples & applications

“A Hamiltonian Extravaganza”

—Danilo J. Rezende@DeepMind

Sep 25 **ICLR 2020 paper submission deadline**

Sep 26 *Symplectic ODE-Net,* 1909.12077

Sep 27 *Hamiltonian Graph Networks with ODE Integrators*, 1909.12790

Sep 29 *Symplectic RNN*, 1909.13334

Sep 30 *Equivariant Hamiltonian Flows*, 1909.13739

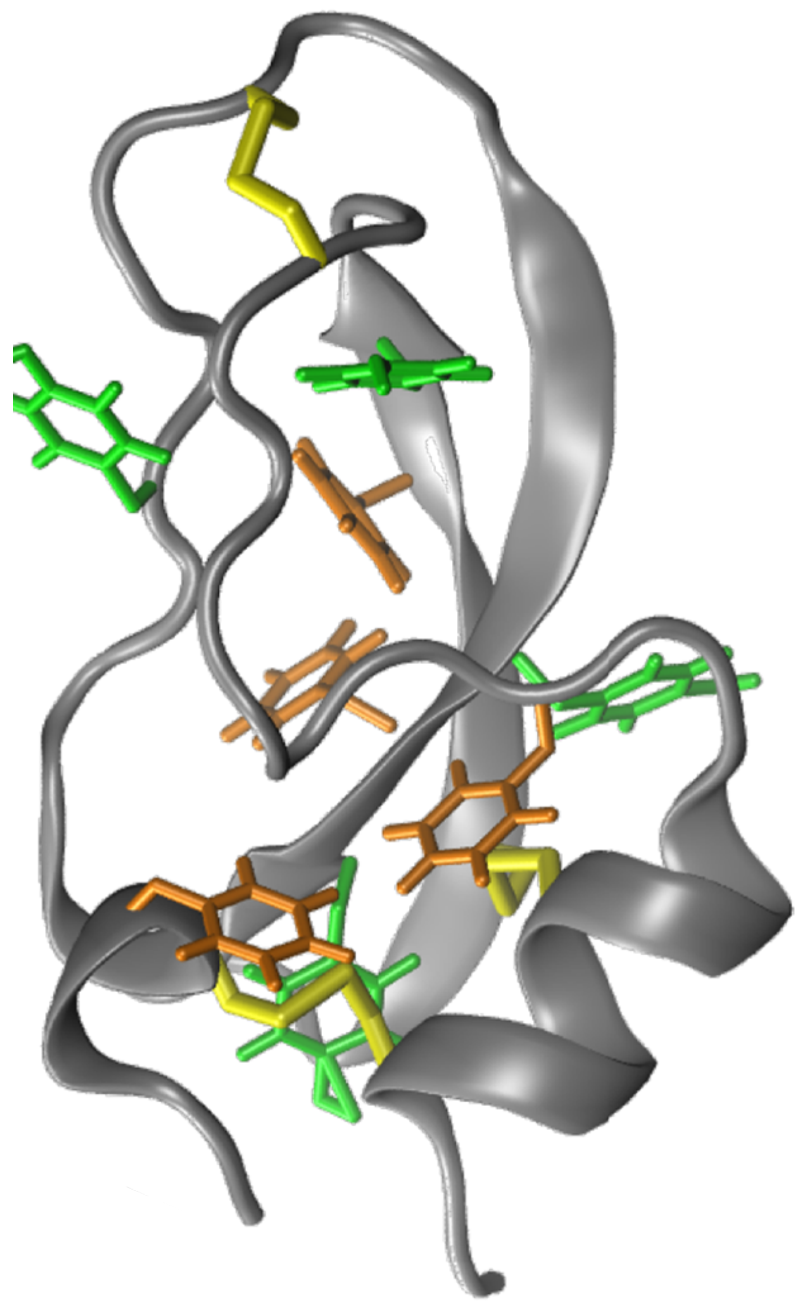
Hamiltonian Generative Network, 1909.13789 http://tiny.cc/hgn

Neural Canonical Transformation with Symplectic Flows, 1910.00024

See also Bondesan & Lamacraft, *Learning Symmetries of Classical Integrable Systems*,1906.04645

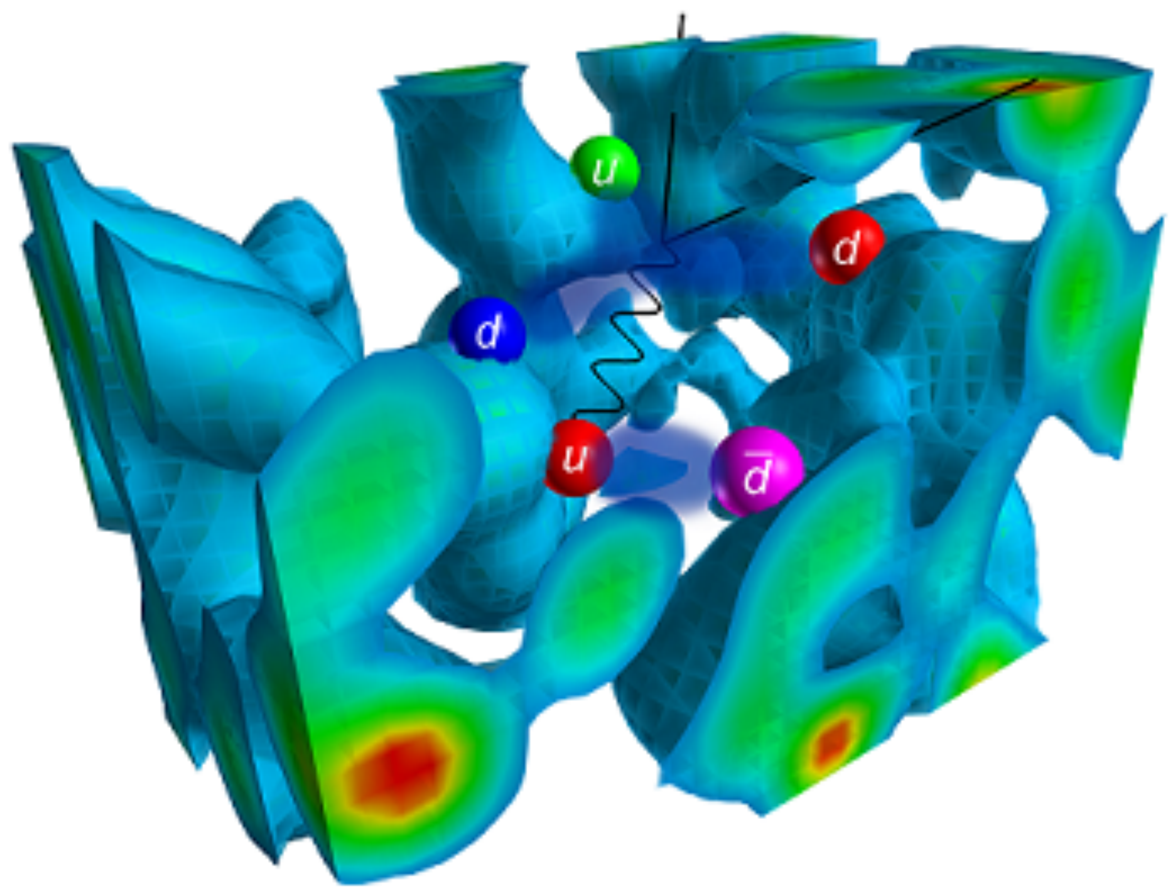
Other scientific applications of flow

Molecular simulation



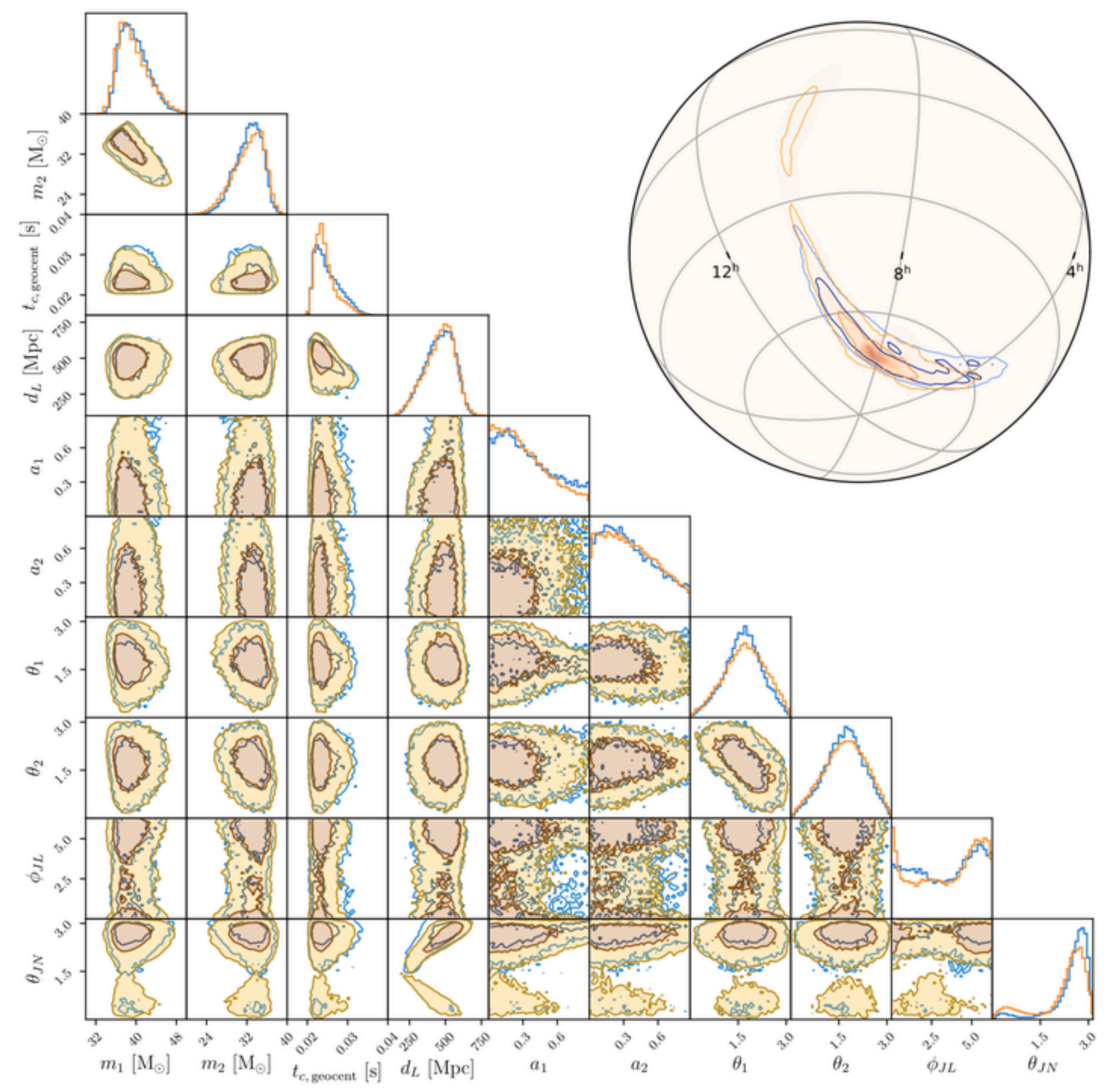
Noe et al, Science ‘19

Lattice field theory



Kanwar et al, PRL ‘20

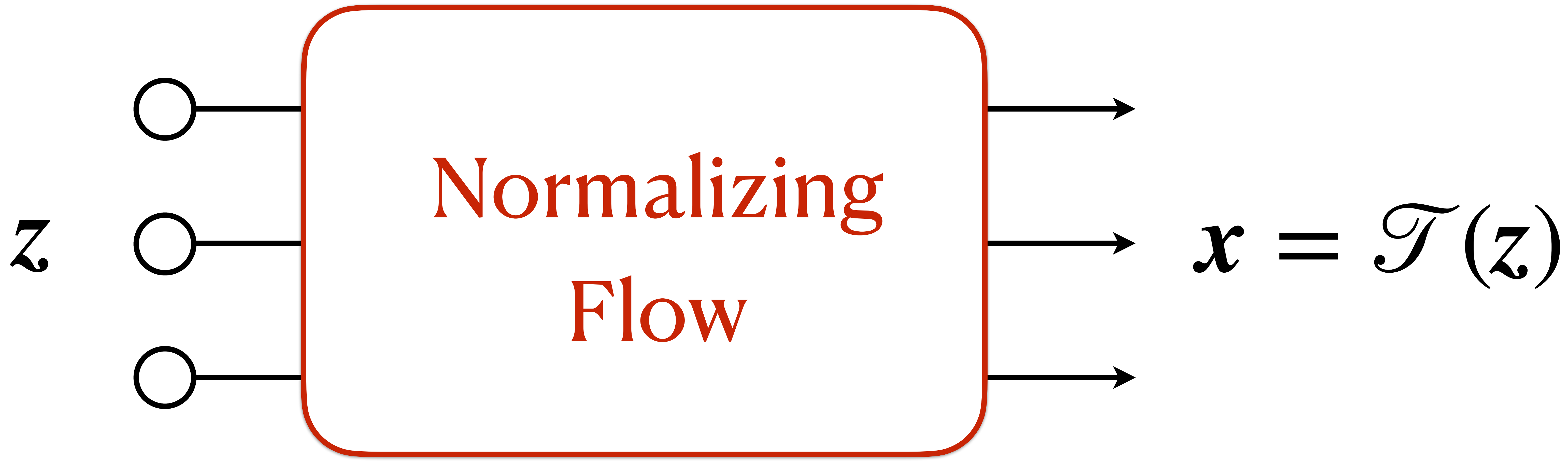
Gravitational wave detection



Green et al, MLST ‘21

Research questions on flow for science

Symmetries



Invariance

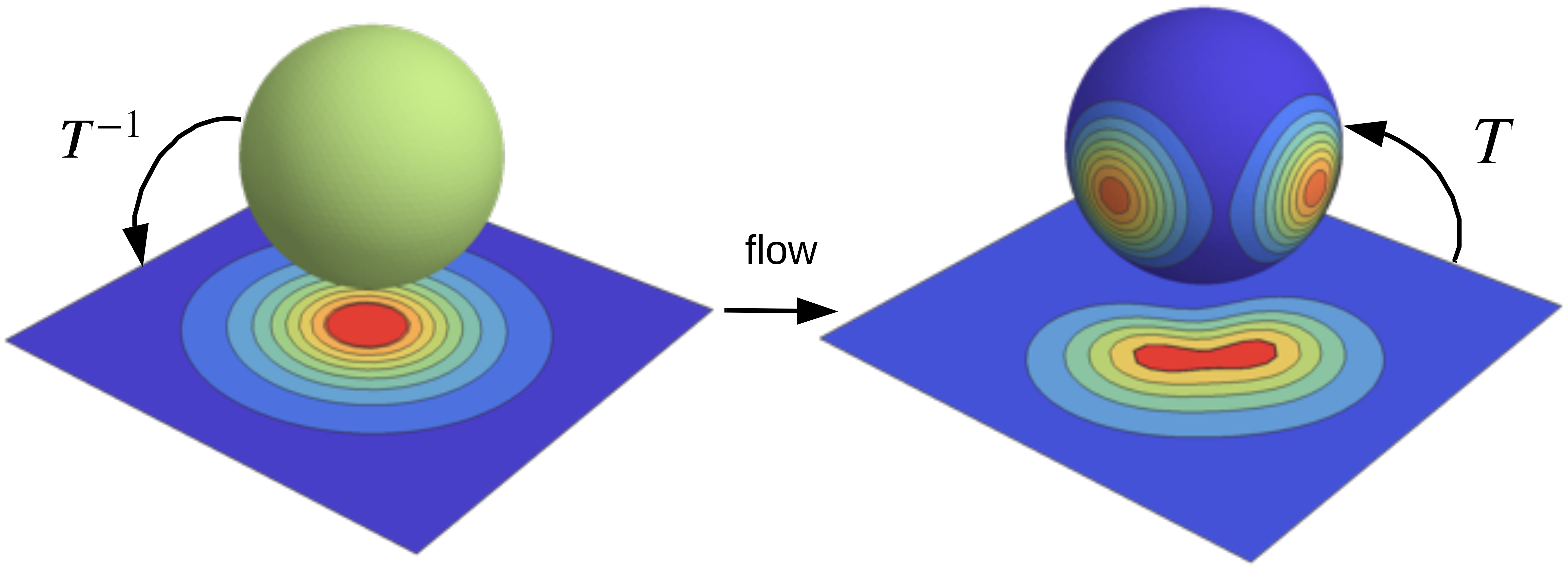
$$\rho(g\,\boldsymbol{x}) = \rho(\boldsymbol{x})$$

Equivariance

$$\mathcal{T}(g\,z) = g\mathcal{T}(z)$$

Spatial symmetries, permutation symmetries, gauge symmetries...

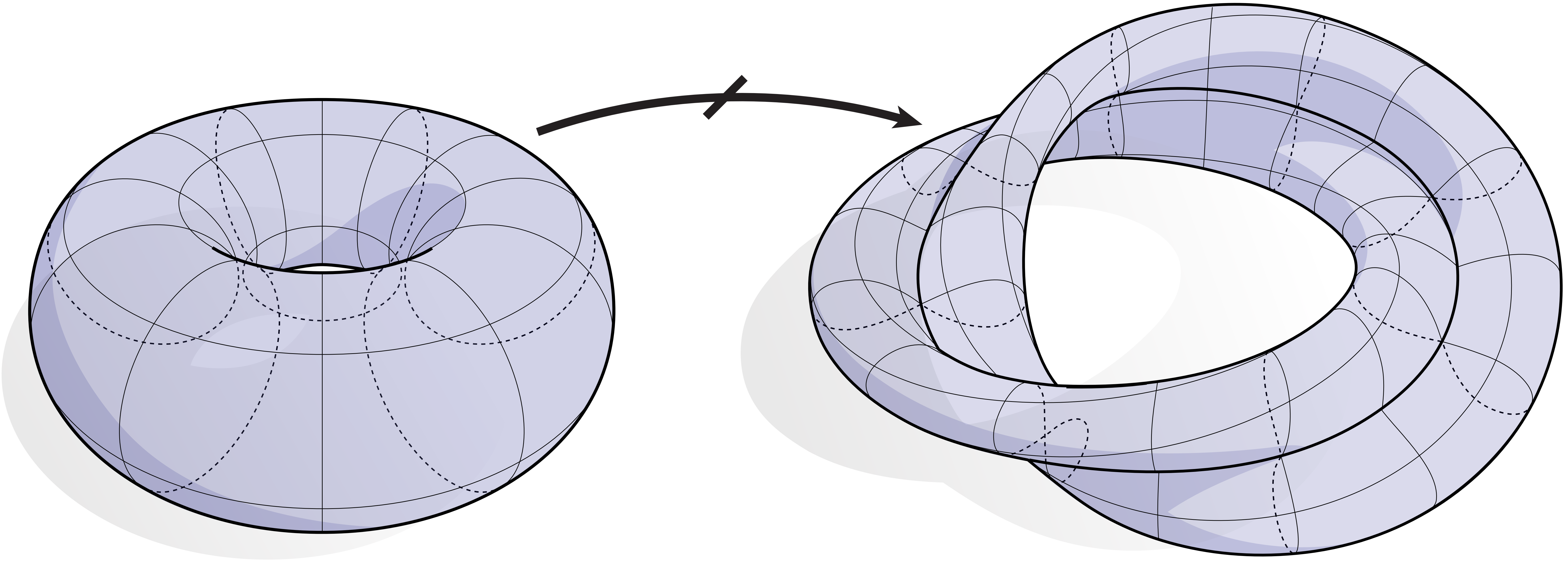
Flow on manifolds



Periodic variables, gauge fields, …

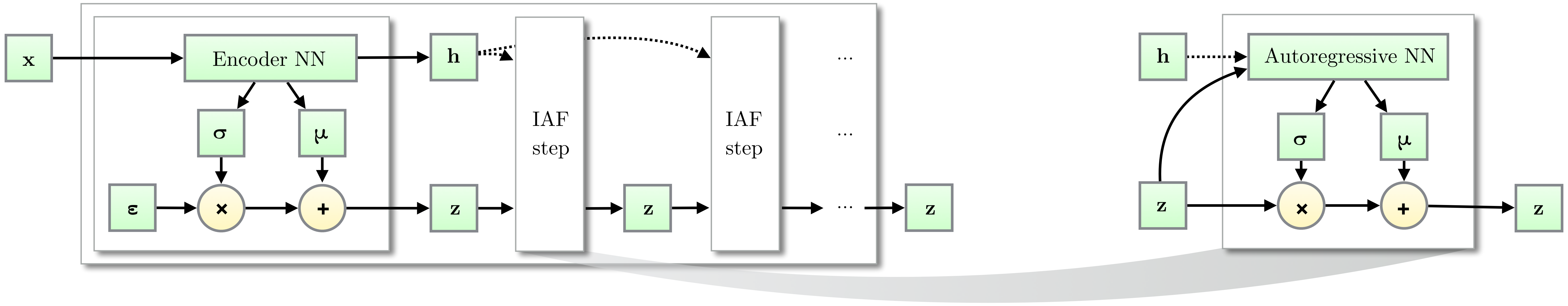
Gemici et al 1611.02304, Rezende et al, 2002.02428, Boyda et al, 2008.05456
Neural ODE on manifolds, Falorsi et al, 2006.06663, Lou et al, 2006.10254, Mathieu et al, 2006.10605

Obstructions

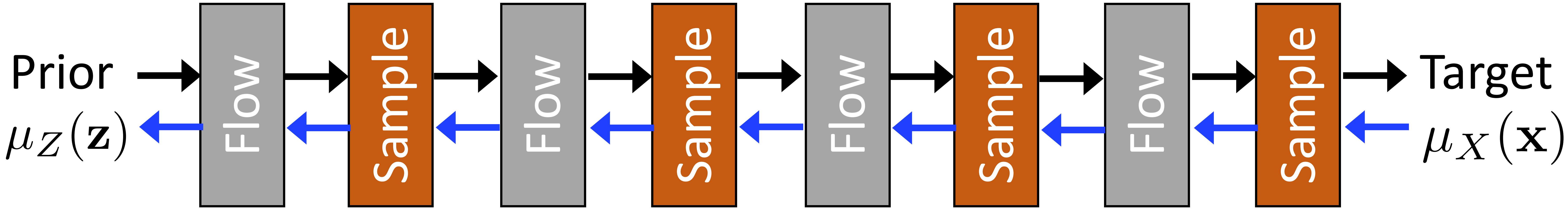


Dupont et al 1904.01681, Cornish et al, 1909.13833,
Zhang et al, 1907.12998, Zhong et al, 2006.00392, Ve´rine et al, 2107.07232

Mix with other approaches



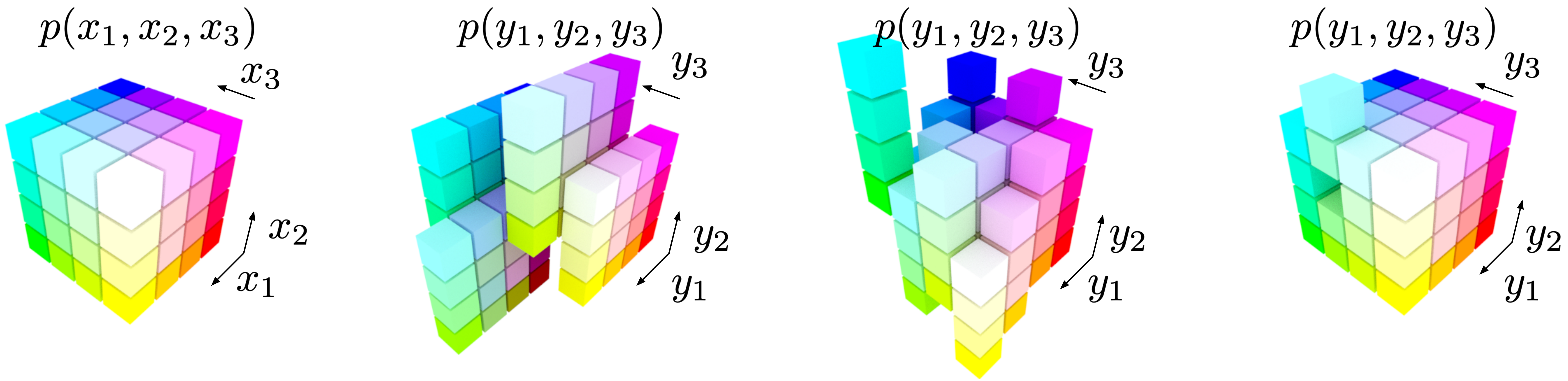
Flow in variational autoencoder Kingma et al, 1606.04934,...



Combining flow with Monte Carlo sampling Levy et al, 1711.09268, Wu et al 2002.06707, ...

Discrete flows

$$p(\boldsymbol{x}) = p(\boldsymbol{y} = \mathcal{T}(\boldsymbol{x}))$$



Tran et al, 1905.10347, Hoogeboom et al, 1905.07376, van den Berg 2006.12459

Learning representations by back-propagating errors

David E. Rumelhart[*], Geoffrey E. Hinton[†] & Ronald J. Williams[*]

Nature, 1986

Thank you!