Continuous Normalizing Flows for Lattice QCD

based on Trivializing Maps

Dr. Simone Bacchio

Computational Scientist

CaSToRC, The Cyprus Institute

A work in collaboration with

Pan Kessel, Stefan Schaefer, Lorenz Vaitl

Funded by
PRACE-6IP, WP8 "Forward Looking Software Solutions".
Grant agreement ID: 823767, Project name: LyNcs.

Generative Models

$$\mathbf{x} = f(\mathbf{z}) \longrightarrow \log p_{\mathbf{x}}(\mathbf{x}) = \log p_{\mathbf{z}}(\mathbf{z}) - \log \det \left| \frac{\partial f(\mathbf{z})}{\partial \mathbf{z}} \right|$$

➢ **First normalizing flows** arXiv:1505.05770

- Restrict functional form of *f* for simplified determinant
- Non-tractable analytic inverse of *f* → Not trainable on data

➢ **Autoregressive transformations** arXiv:1606.04934

- Use autoregressive models for lower-triangular Jacobian
- Expensive inverse of *f*, which requires *D* applications of *f*

➢ **Partitioned transformations** arXiv:1605.08803

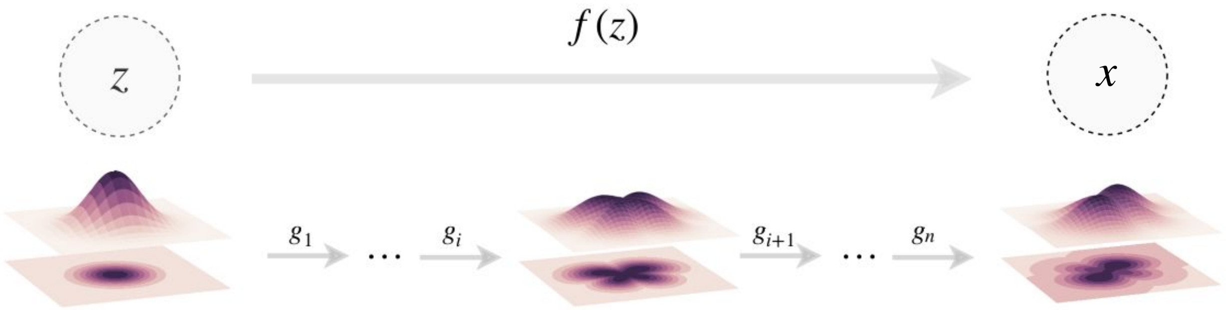
- Use partitioning and affine transformations for cheap *det* and inverse of *f*

➢ Cost of det ?

➢ Inverse of f?

From discrete to continuous

$$\mathbf{x} = f(\mathbf{z}) \longrightarrow \log p_{\mathbf{x}}(\mathbf{x}) = \log p_{\mathbf{z}}(\mathbf{z}) - \log \det \left| \frac{\partial f(\mathbf{z})}{\partial \mathbf{z}} \right|$$



$$\mathbf{h}_{t+1} = \mathbf{h}_t + f(\mathbf{h}_t, \theta_t) \xrightarrow{?} \frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \theta)$$

From discrete to continuous

$$\mathbf{x} = f(\mathbf{z}) \longrightarrow \log p_{\mathbf{x}}(\mathbf{x}) = \log p_{\mathbf{z}}(\mathbf{z}) - \log \det \left| \frac{\partial f(\mathbf{z})}{\partial \mathbf{z}} \right|$$

Neural Ordinary Differential Equations

arXiv:1806.07366

Ricky T. Q. Chen*, Yulia Rubanova*, Jesse Bettencourt*, David Duvenaud
University of Toronto, Vector Institute
{rtqichen, rubanova, jessebett, duvenaud}@cs.toronto.edu

$$\boxed{\frac{d\mathbf{z}}{dt} = f(\mathbf{z}(t), t)} \longrightarrow \log p(\mathbf{z}(t_1)) = \log p(\mathbf{z}(t_0)) - \int_{t_0}^{t_1} \mathrm{Tr}\left(\frac{\partial f}{\partial \mathbf{z}(t)} \right) dt$$

$$\longrightarrow \mathbf{x} = \mathbf{z}(t_1) = \mathbf{z}(t_0) + \int_{t_0}^{t_1} f(\mathbf{z}(t), t, \theta) dt$$

✓ *Tr cheaper*

✓ *No inverse required*

CNF for LFT

How to define $$\dot{U} \equiv \frac{dU}{dt} = f(U, t)$$???

where U is in $SU(N)$

ODEs on manifolds

$$\dot{U} = g(U)U$$ where $U \in$ Group, $\boxed{g \in \text{Algebra}}$

- g(U) must be element of the algebra
- Imposing Gauge invariance:

$$U_\mu(x) \to \Omega(x)U_\mu(x)\Omega^\dagger(x+\mu) \quad \Rightarrow \quad \boxed{g(U_\mu(x)) \to \Omega(x)g(U_\mu(x))\Omega^\dagger(x)}$$

- Strong constraints on g(U), **how to satisfy these properties?**

Lüscher's ansatz

$$g(U_\mu(x)) = \partial_{x,\mu}\tilde{S}(U)$$
$$\dot{U} = \left(\partial_{x,\mu}\tilde{S}(U)\right) U$$

arXiv:0907.5491

Trivializing maps, the Wilson flow and the HMC algorithm

Martin Lüscher

CERN, Physics Department, 1211 Geneva 23, Switzerland

where

$\tilde{S}(U) = \sum_i c_i W_i(U)$ and $W(U) = \sum_{x,\mu} \mathrm{Re}\,\mathrm{Tr}(U_\mu(x)\Sigma_\mu(x))$

Proof of properties

$$\partial_{x,\mu}\tilde{S}(U) = \sum_i c_i \sum_{y,\nu} \partial_{x,\mu}\mathrm{Tr}\big(U_\nu(y)\Sigma_\nu(y) + U_\nu^\dagger(y)\Sigma_\nu^\dagger(y)\big)$$

- **Is it element of the algebra?**

$$\begin{aligned} T_a\partial^a_{x,\mu}\mathrm{Tr}\big(U_\mu(x)\Sigma_\mu(x) + U_\mu^\dagger(x)\Sigma_\mu^\dagger(x)\big) &= T_a\mathrm{Tr}\big(T_aU_\mu(x)\Sigma_\mu(x) - \Sigma_\mu^\dagger(x)U_\mu^\dagger(x)T_a\big) \\ &\equiv T_a\mathrm{Tr}\big(T_a(U_\mu(x)\Sigma_\mu(x) - U_\mu^\dagger(x)\Sigma_\mu^\dagger(x))\big) \end{aligned}$$

$$M - M^\dagger = i\alpha_0 1 + \sum_b \alpha_b T_b \quad \Rightarrow \quad = T_a \sum_b \alpha_b \mathrm{Tr}(T_aT_b)$$

$$= -\frac{1}{2}T_a\sum_b \alpha_b\delta_{ab} = -\frac{1}{2}\sum_b \alpha_b T_b \qquad \square$$

Proof of properties

$$\partial_{x,\mu} \tilde{S}(U) = \sum_i c_i \sum_{y,\nu} \partial_{x,\mu} \mathrm{Tr}\left(U_\nu(y)\Sigma_\nu(y) + U_\nu^\dagger(y)\Sigma_\nu^\dagger(y)\right)$$

- **Does it transform as** $g(U_\mu(x)) \to \Omega(x) g(U_\mu(x)) \Omega^\dagger(x)$ **?**

$$\partial_{x,\mu} \mathrm{Tr}\left(U_\mu(x)\Sigma_\mu(x) + U_\mu^\dagger(x)\Sigma_\mu^\dagger(x)\right) = -\tfrac{1}{2}\left(U_\mu(x)\Sigma_\mu(x) - \Sigma_\mu^\dagger(x)U_\mu^\dagger(x) - i\alpha_0 1\right)$$

if $U_\mu(x)\Sigma_\mu(x)$ is a closed path, then

$$U_\mu(x) \to \Omega(x) U_\mu(x) \Omega^\dagger(x+\mu) \quad \Longrightarrow \quad \partial_{x,\mu} \tilde{S}(U) \to \Omega(x) \partial_{x,\mu} \tilde{S}(U) \Omega^\dagger(x)$$

$\square$

Lüscher's ansatz

$$g(U_\mu(x)) = \partial_{x,\mu}\tilde{S}(U)$$

$$\dot{U} = \left(\partial_{x,\mu}\tilde{S}(U)\right)U$$

arXiv:0907.5491

Trivializing maps, the Wilson flow and the HMC algorithm

Martin Lüscher

CERN, Physics Department, 1211 Geneva 23, Switzerland

Lüscher ansatz satisfies all properties, but...

?

- Does the force of any gauge invariant quantity satisfy the properties?
- Are there more generic approaches to define g(U)?
- Is it Lüscher ansatz good enough to define a CNF?

Lüscher's ansatz

$$g(U_\mu(x)) = \partial_{x,\mu}\tilde{S}(U)$$

$$\dot{U} = \left(\partial_{x,\mu}\tilde{S}(U)\right) U$$

arXiv:0907.5491

Trivializing maps, the Wilson flow and the HMC algorithm

Martin Lüscher

CERN, Physics Department, 1211 Geneva 23, Switzerland

$$\boxed{\frac{d\mathbf{z}}{dt} = f(\mathbf{z}(t), t)} \rightarrow \log p(\mathbf{z}(t_1)) = \log p(\mathbf{z}(t_0)) - \int_{t_0}^{t_1} \mathrm{Tr}\left(\frac{\partial f}{\partial \mathbf{z}(t)}\right) dt$$

$$\rightarrow \mathbf{x} = \mathbf{z}(t_1) = \mathbf{z}(t_0) + \int_{t_0}^{t_1} f(\mathbf{z}(t), t, \theta) dt$$

Reminder about CNF

Lüscher's ansatz

$$g(U_\mu(x)) = \partial_{x,\mu}\tilde{S}(U)$$

$$\dot{U} = \left(\partial_{x,\mu}\tilde{S}(U)\right)U$$

arXiv:0907.5491

Trivializing maps, the Wilson flow and the HMC algorithm

Martin Lüscher

CERN, Physics Department, 1211 Geneva 23, Switzerland

- Another result from his work: **Lüscher already discovered CNFs!**

$$\log p(U(t_1)) = \log p(U(t_0)) - \int_{t_0}^{t_1} \mathcal{L}\tilde{S}(U)dt$$

where $$\mathcal{L}\tilde{S}(U) = -\sum_{x,\mu,a}\partial^a_{x,\mu}\partial^a_{x,\mu}\tilde{S}(U)$$

Laplacian of action

$$\mathcal{L}\tilde{S}(U) = -\sum_{x,\mu,a} \partial^a_{x,\mu}\partial^a_{x,\mu}\tilde{S}(U)$$

$$= -\sum_i c_i \sum_{x,\mu,a} \sum_{y,\nu} \partial^a_{x,\mu}\partial^a_{x,\mu} \mathrm{ReTr}\big(U_\nu(y)\Sigma_\nu(y)\big)$$

For loops w/o repeated links

$$= -\sum_i c_i \sum_{x,\mu,a} \mathrm{ReTr}\big(T^a T^a U_\mu(x)\Sigma_\mu(x)\big)$$

Laplacian of action

$$\mathcal{L}\tilde{S}(U) = -\sum_{x,\mu,a} \partial^a_{x,\mu}\partial^a_{x,\mu}\tilde{S}(U)$$

$$= -\sum_i c_i \sum_{x,\mu,a}\sum_{y,\nu} \partial^a_{x,\mu}\partial^a_{x,\mu} \mathrm{ReTr}\big(U_\nu(y)\Sigma_\nu(y)\big)$$

For loops w/o repeated links

$$= -\sum_i c_i \sum_{x,\mu,a} \mathrm{ReTr}\big(T^a T^a U_\mu(x)\Sigma_\mu(x)\big)$$

Using the completeness relation

$$= \frac{N^2-1}{2N}\sum_i c_i \sum_{x,\mu} \mathrm{ReTr}\big(U_\mu(x)\Sigma_\mu(x)\big) = \boxed{\frac{N^2-1}{2N}\tilde{S}(U)}$$

$$\sum_a T^a_{\alpha\beta}T^a_{\gamma\delta} = -\frac{1}{2}\left(\delta_{\alpha\delta}\delta_{\beta\gamma} - \frac{1}{N}\delta_{\alpha\beta}\delta_{\gamma\delta}\right) \Rightarrow \sum_a T^a_{\alpha\beta}T^a_{\beta\delta} = -\frac{N^2-1}{2N}\delta_{\alpha\delta}$$

Our work: from Trivializing Maps to CNF

1. Time-dependence in the coefficients

$$\tilde{S}(U) = \sum_i c_i(t) W_i(U)$$

2. Training of the coefficients via minimization of the KL divergence
3. Calculation of the gradients via back-propagation
4. Generic implementation for any Wilson loop
5. …

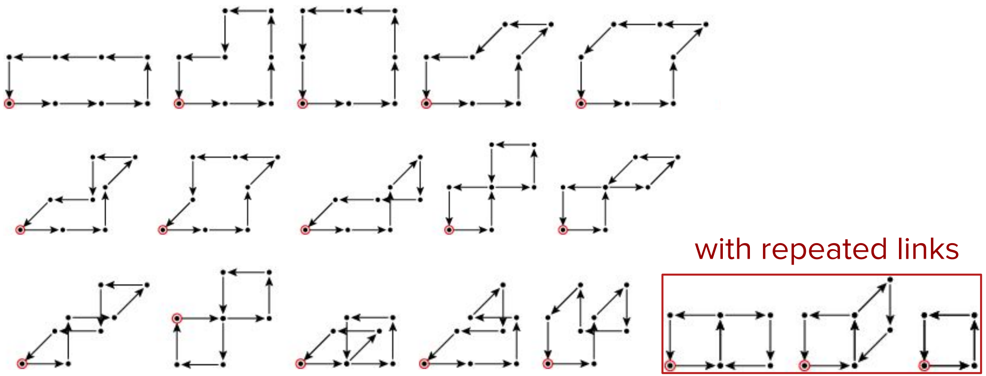
Mapping from uniform distribution:

$$L_{KL} = S_{\text{target}}(U_T) + \int_0^1 \mathcal{L}\tilde{S}(U_t)dt$$

Software

- Developed using Python and Lyncs-API
- Numpy implementation for S(N) in M-dimensions
- On GPU via Quda for SU(3) in 2/3/4-dimensions
- Logic for dealing with any-size closed loop

E.g. all unique geometries of length 8 in 3D

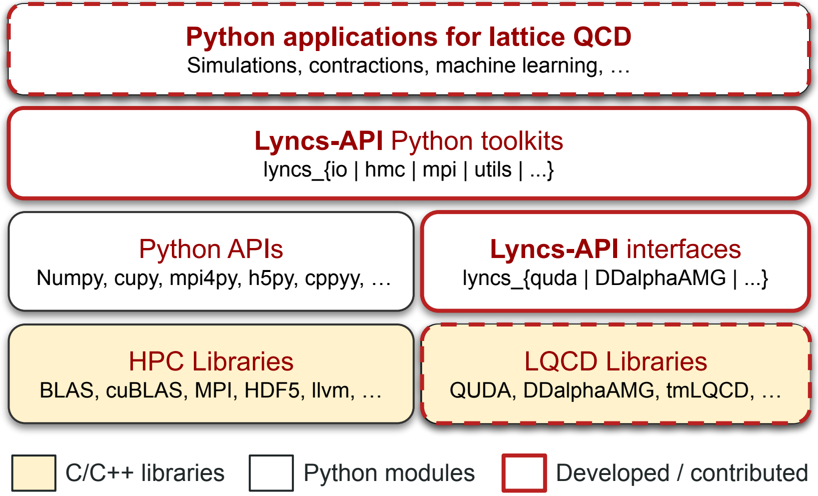


Lyncs-API

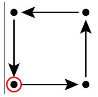
A Python API for Lattice QCD applications

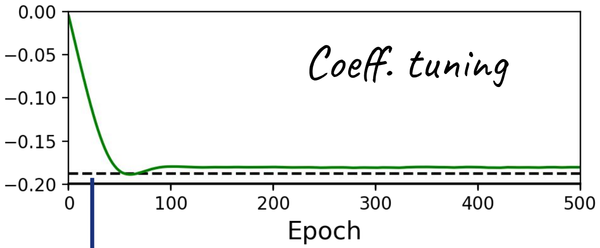
https://lyncs-api.github.io/ s.bacchio@gmail.com

Python ecosystem for Lattice QCD



First results





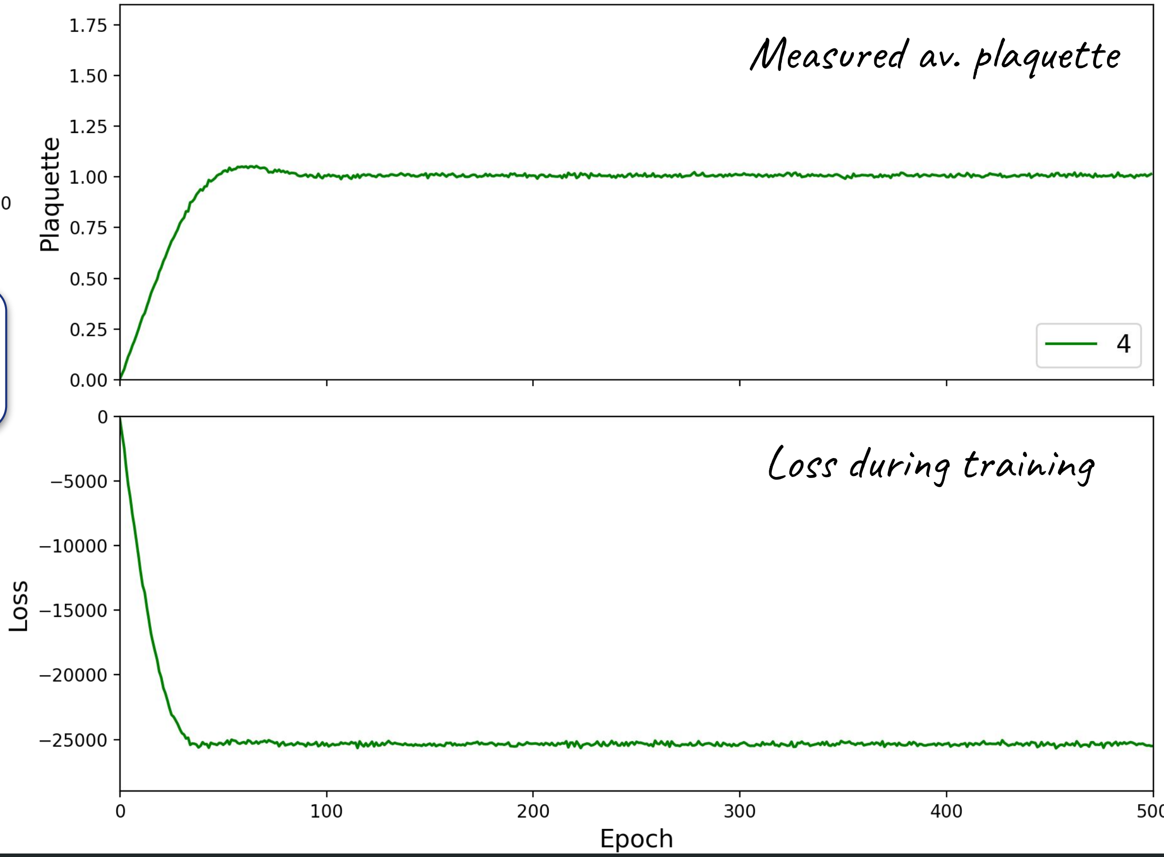
Wilson flow param.

$$\widetilde{S}^{(0)} = \left(-\frac{1}{32}\beta\right)\mathcal{W}_0 = \frac{3}{16}S_{\mathrm{w}}$$

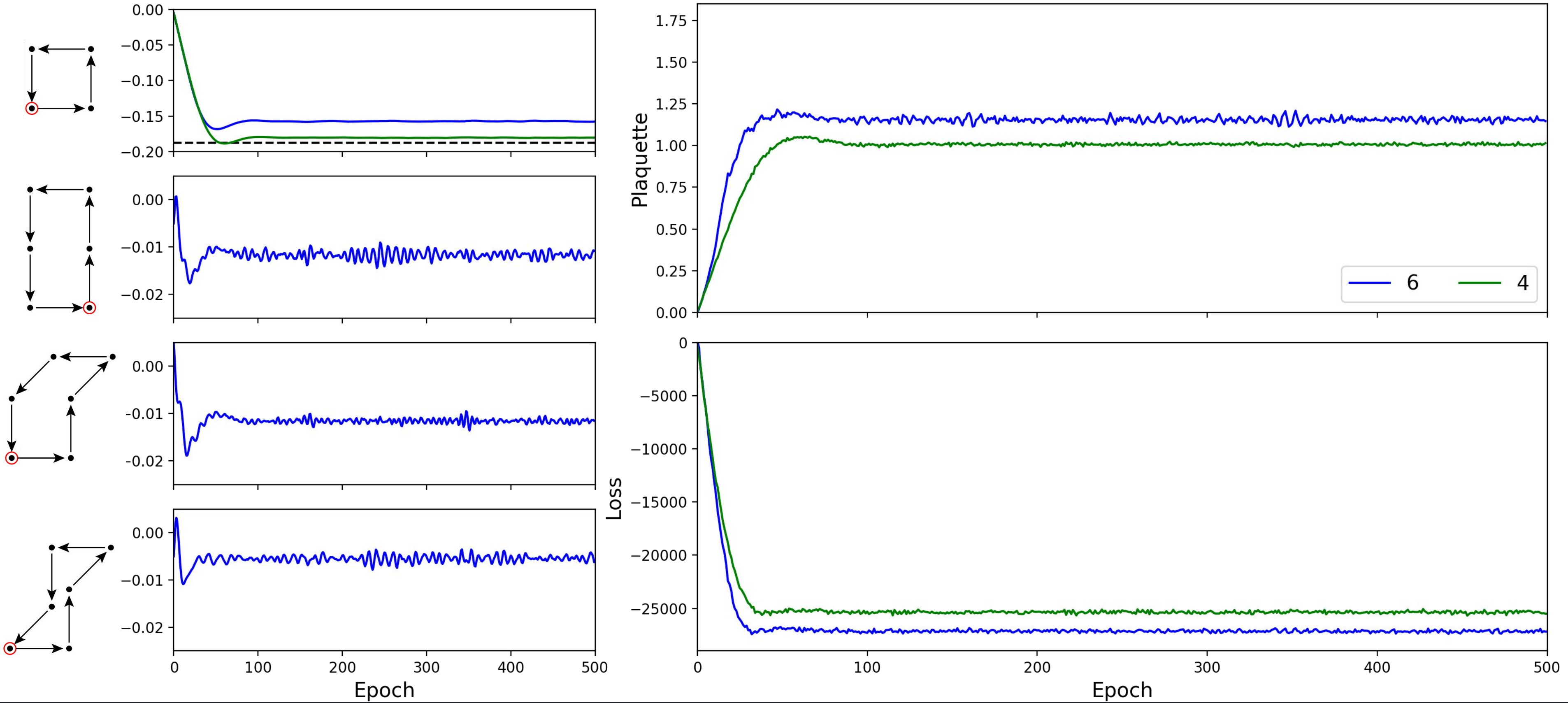
Setup

- Pure-gauge SU(3)
- Lattice size 16^4
- Coupling $\beta=6$
- No time depend.

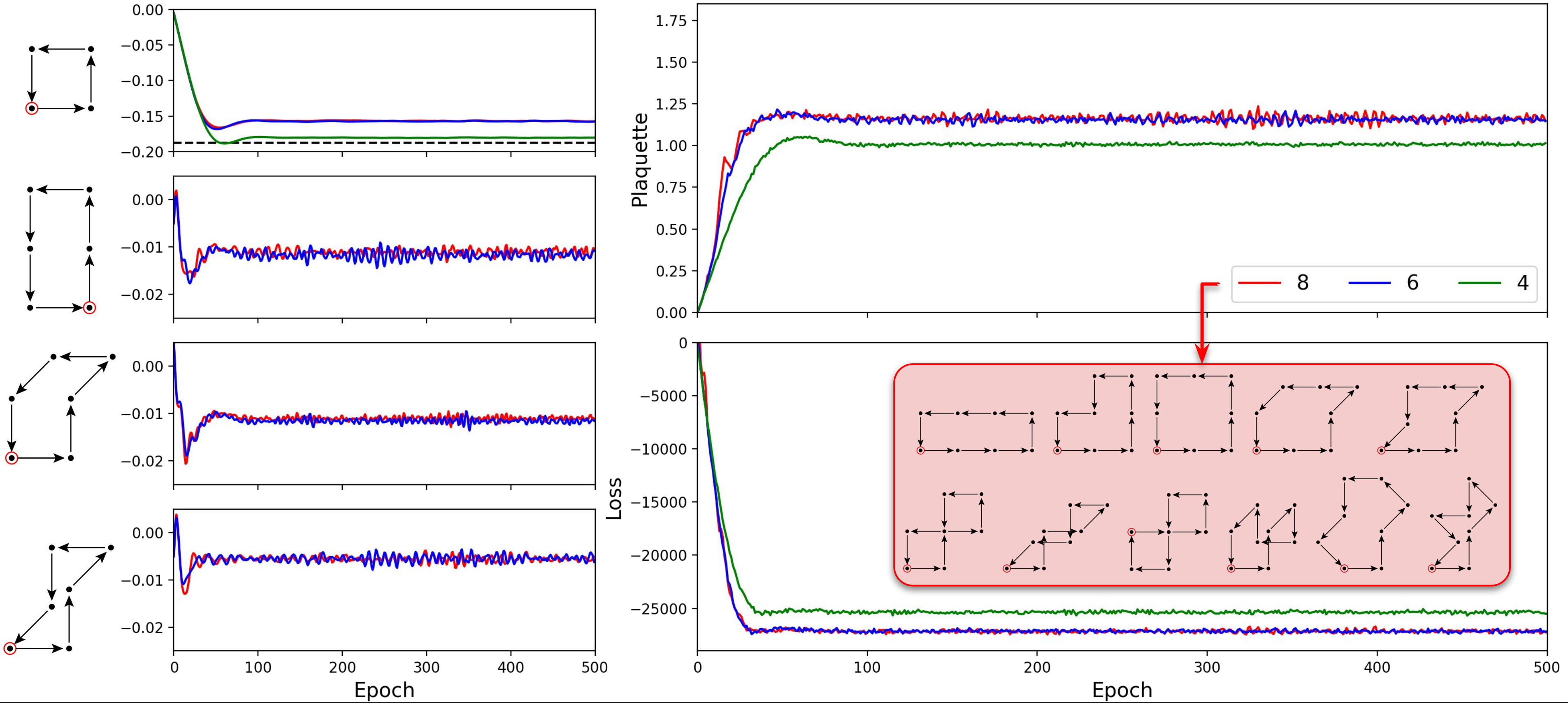
$$\tilde{S}(U) = \sum_i c_i(\cancel{t})W_i(U)$$



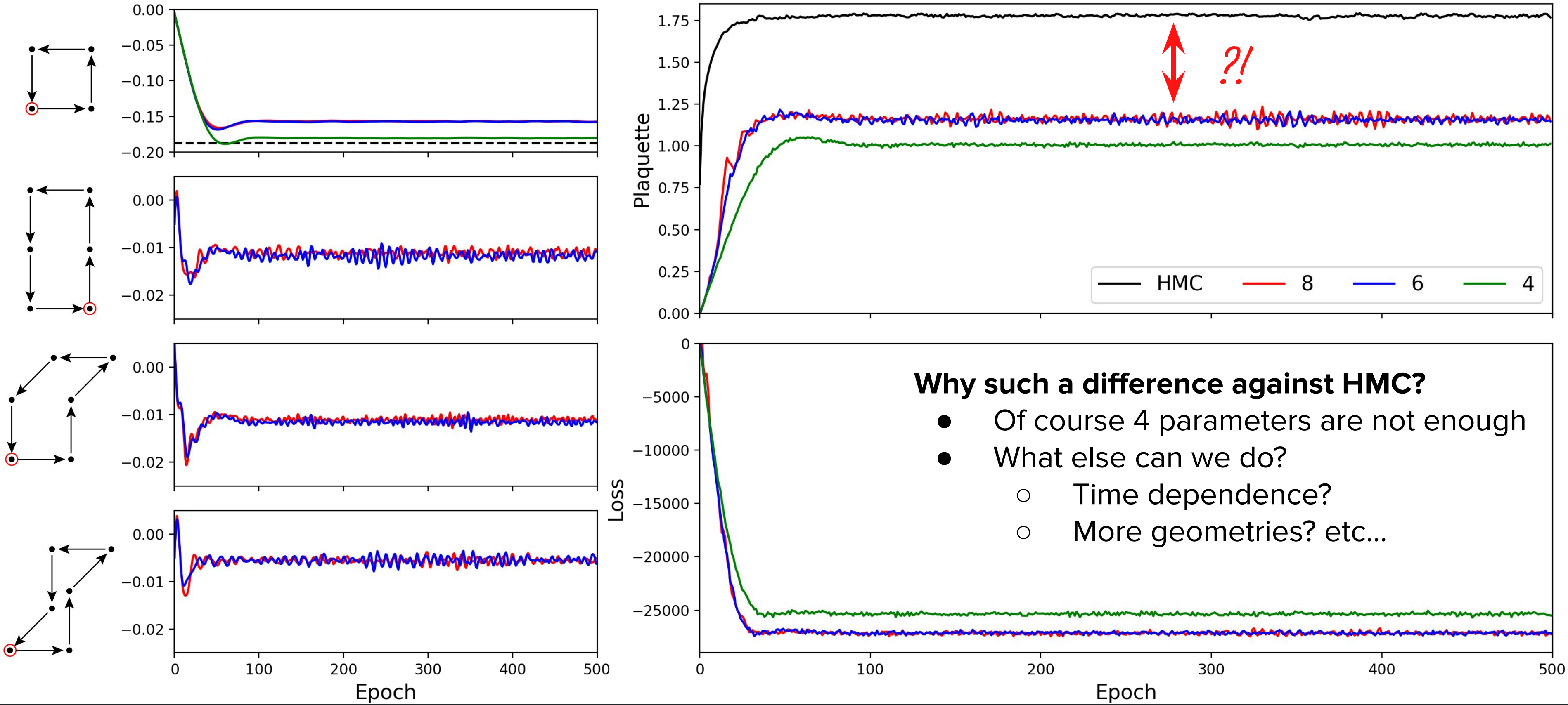
First results



First results



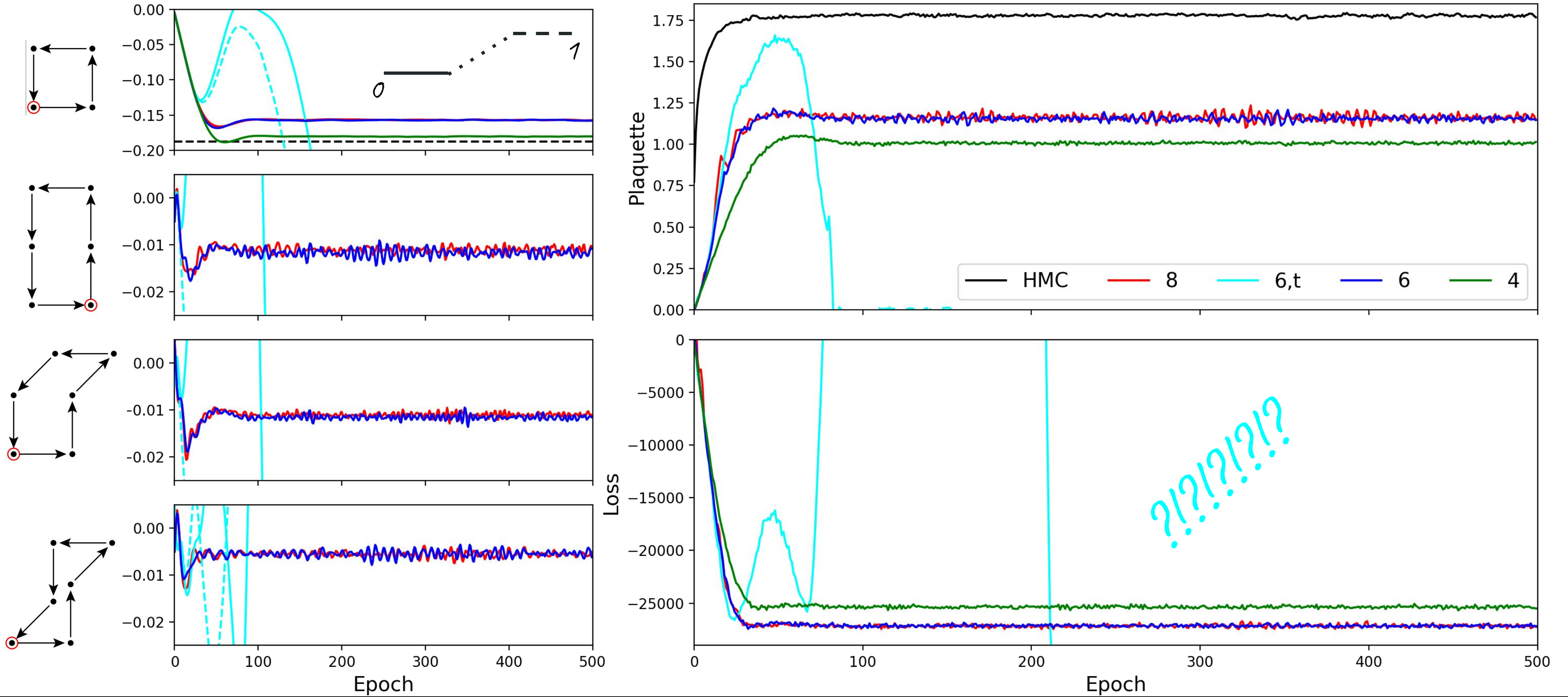
First results



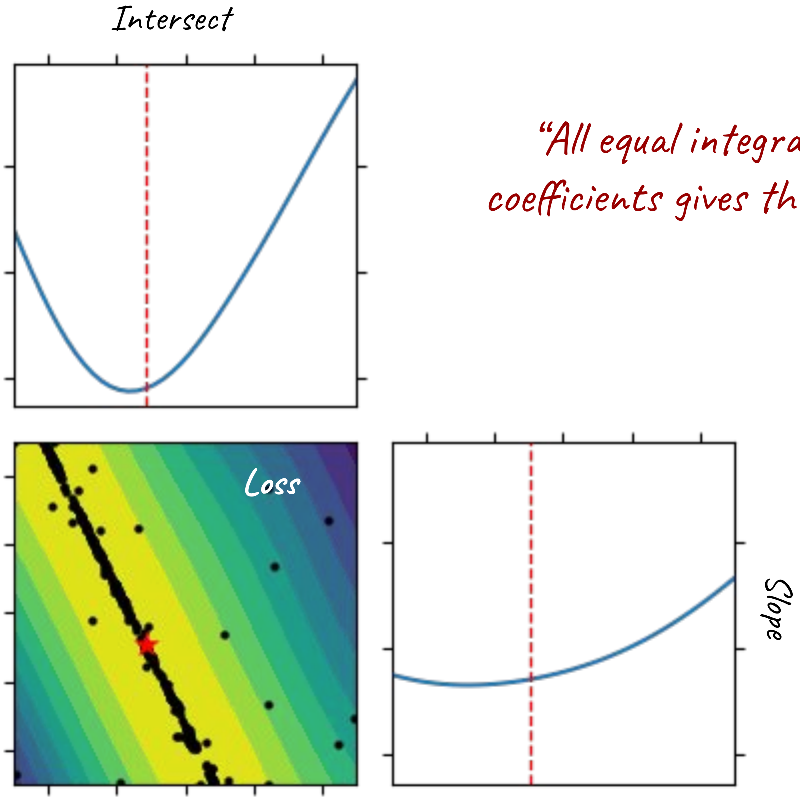
Why such a difference against HMC?

- Of course 4 parameters are not enough
- What else can we do?
    - Time dependence?
    - More geometries? etc...

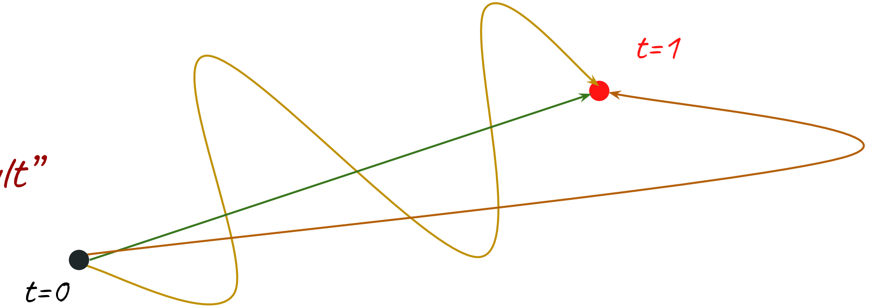
First results



Degeneracy of integral

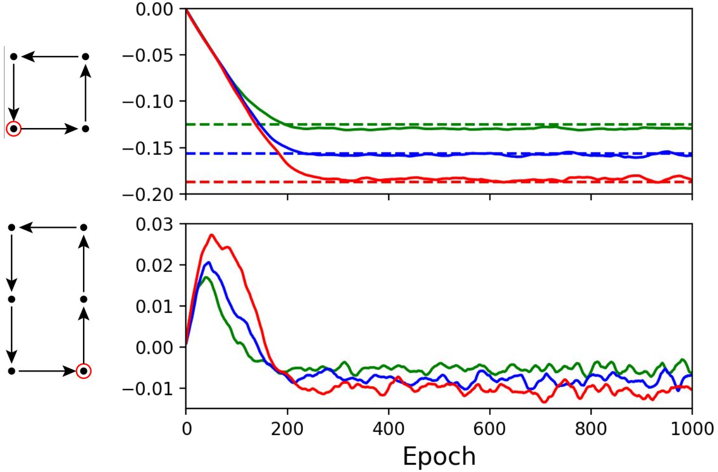


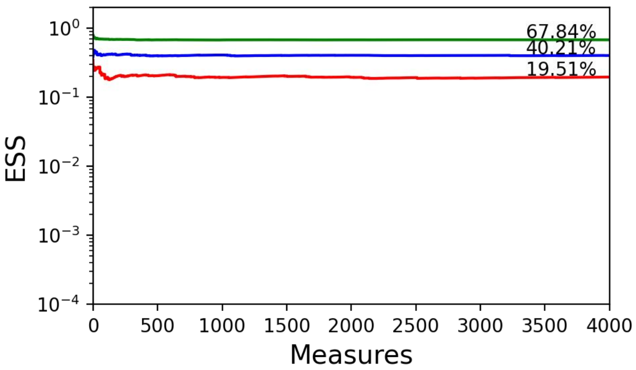
"All equal integrals over the coefficients gives the same result"

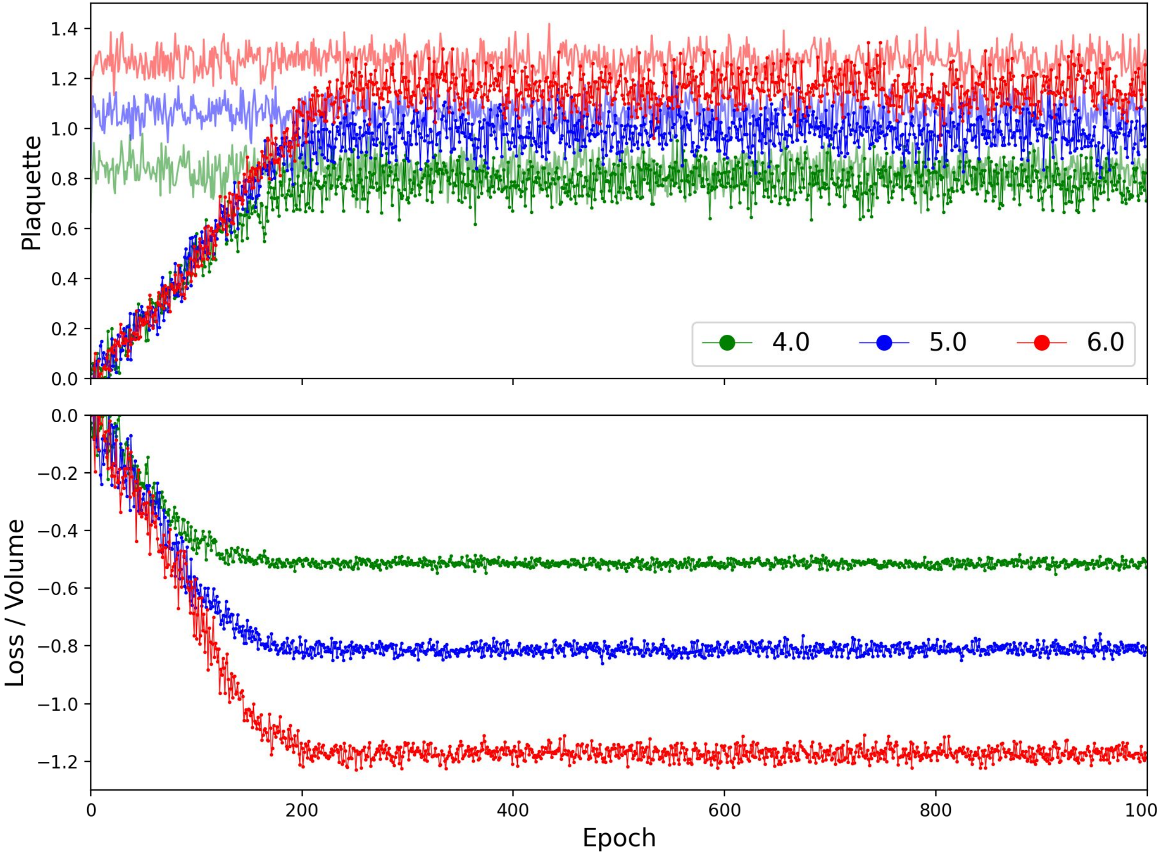


- Single-geometry integral is trivially degenerate
- Multiple-geometries integral is also degenerate:
  - Numerically tested for length 6... *Any proof?*
- Large coefficient introduce numerical instabilities
- Is constant enough? *Not really!*
- How to solve the problem? *We could improve it but not solve it*

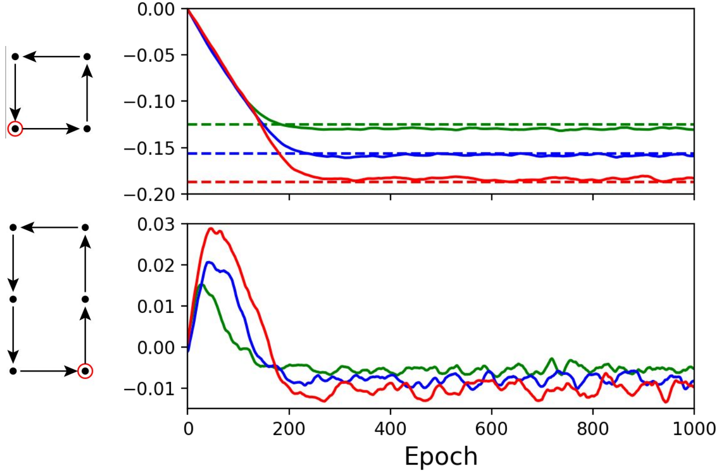
Let's be less ambitious: 4^2

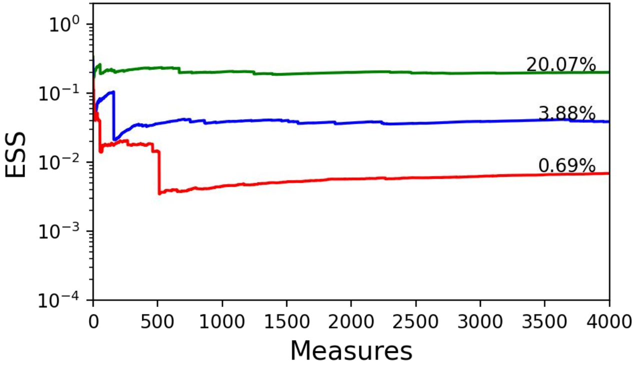


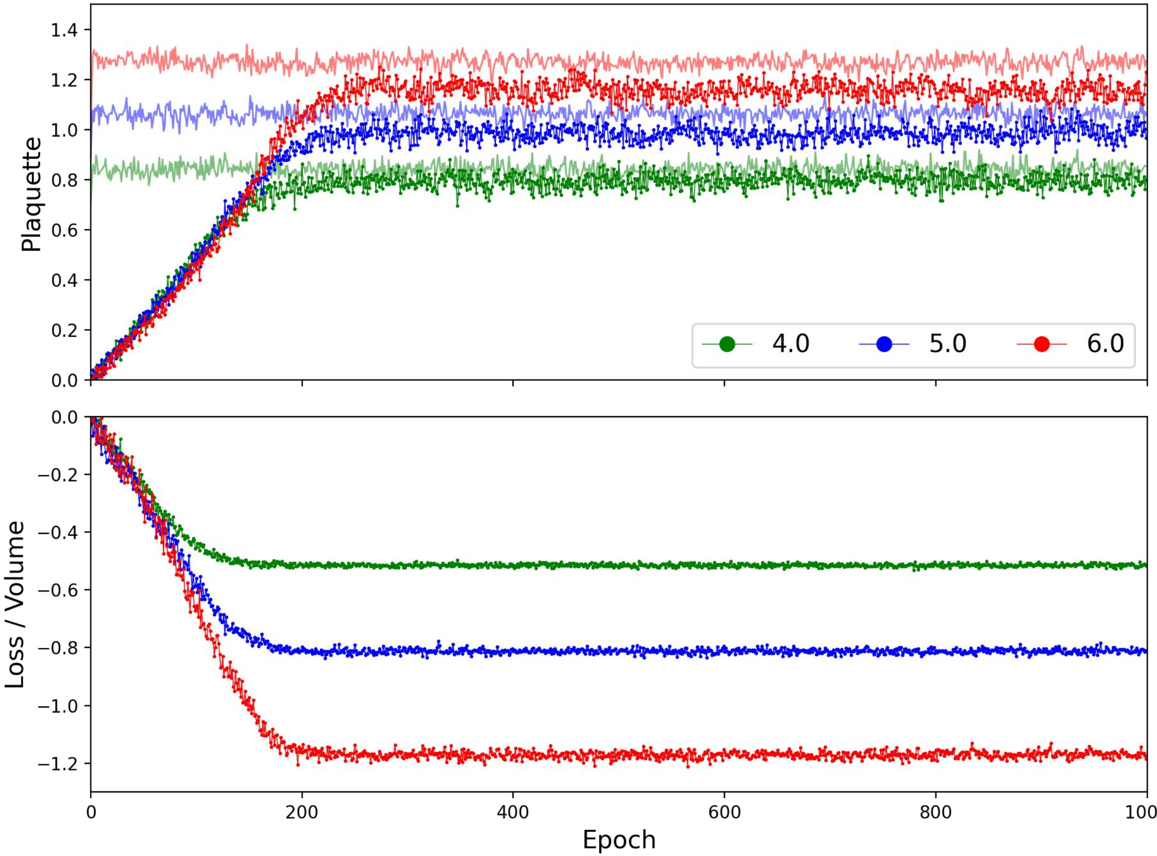




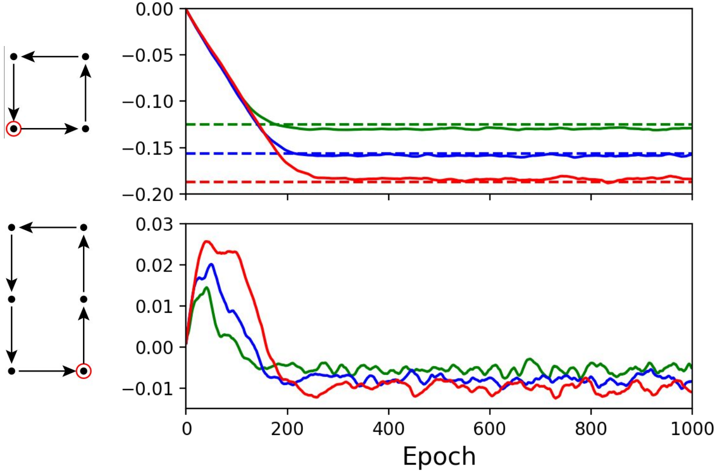
Let’s be less ambitious: 8^2

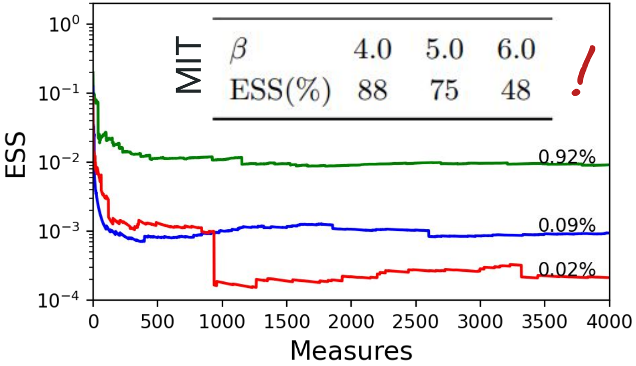




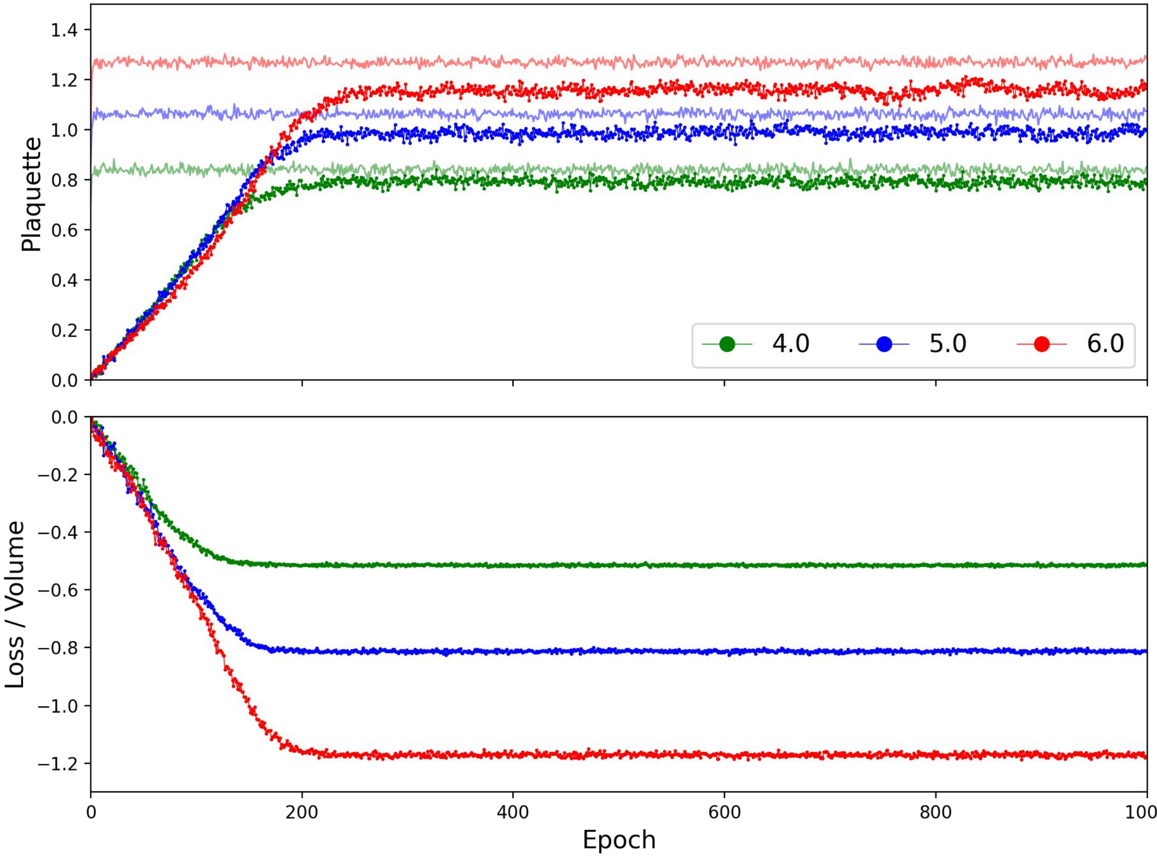


Let's be less ambitious: 16^2





MIT			
β	4.0	5.0	6.0
ESS(%)	88	75	48



What’s more? Loops with repeated links!

- **Issues:**
  - Much more difficult lagrangian
  - Product of traces and shifts

- **Questions:**
  - How to generalize them?
  - Will they help?

arXiv:0907.5491

Trivializing maps, the Wilson flow and the HMC algorithm

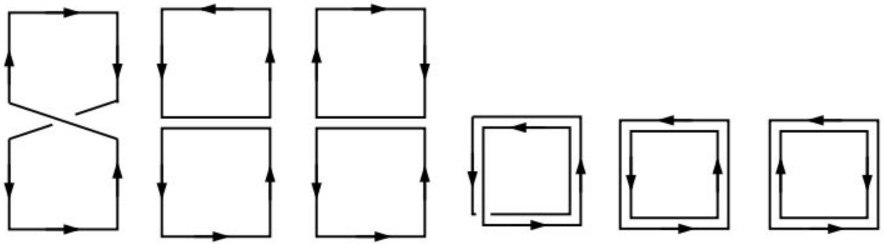
Martin Lüscher

CERN, Physics Department, 1211 Geneva 23, Switzerland

$$\mathfrak{L}_0\mathcal{W}_2 = \tfrac{31}{3}\mathcal{W}_2 + \mathcal{W}_4, \qquad \mathfrak{L}_0\mathcal{W}_5 = \tfrac{28}{3}\mathcal{W}_5 + 4\mathcal{W}_6,$$

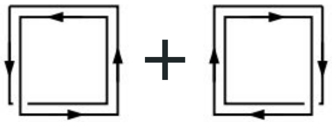
$$\mathfrak{L}_0\mathcal{W}_3 = 11\mathcal{W}_3 - \mathcal{W}_1, \qquad \mathfrak{L}_0\mathcal{W}_6 = \tfrac{28}{3}\mathcal{W}_6 + 4\mathcal{W}_5,$$

$$\mathfrak{L}_0\mathcal{W}_4 = \tfrac{31}{3}\mathcal{W}_4 + \mathcal{W}_2, \qquad \mathfrak{L}_0\mathcal{W}_7 = 12\mathcal{W}_7 + \text{constant}.$$



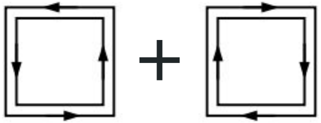
(2) (3) (4) (5) (6) (7)

Giving a closer loop



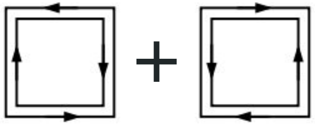
(5)

$$\mathrm{Re}\,\mathrm{Tr}(W^2)$$



(6)

$$\mathrm{Re}(\mathrm{Tr}(W))^2 - \mathrm{Im}(\mathrm{Tr}(W))^2$$



(7)

$$\mathrm{Re}(\mathrm{Tr}(W))^2 + \mathrm{Im}(\mathrm{Tr}(W))^2$$

$$\mathrm{Re}(\mathrm{Tr}(W))^2$$

$$\mathrm{Im}(\mathrm{Tr}(W))^2$$

Our work: from Trivializing Maps to CNF

1. Time-dependence in the coefficients

$$\tilde{S}(U) = \sum_i c_i(t) W_i(U)$$

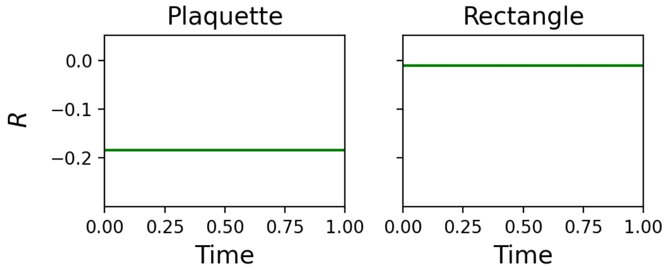
2. Training of the coefficients via minimization of the KL divergence
3. Calculation of the gradients via back-propagation
4. Generic implementation for any Wilson loop
5. Implementation of improved model:

Mapping from uniform distribution:

$$L_{KL} = S_{\text{target}}(U_T) + \int_0^1 \mathcal{L}\tilde{S}(U_t)dt$$

$$\tilde{S} = \sum_{i,l,m,n} c_{i,l,m,n}(t) \mathrm{Re}(W_{i,l})^m \mathrm{Im}(W_{i,l})^{2n} \quad \text{with} \quad W_{i,l} \equiv \mathrm{Tr}(W_i(U)^l)$$

Latest results: NMCMC, 16^2, β=6

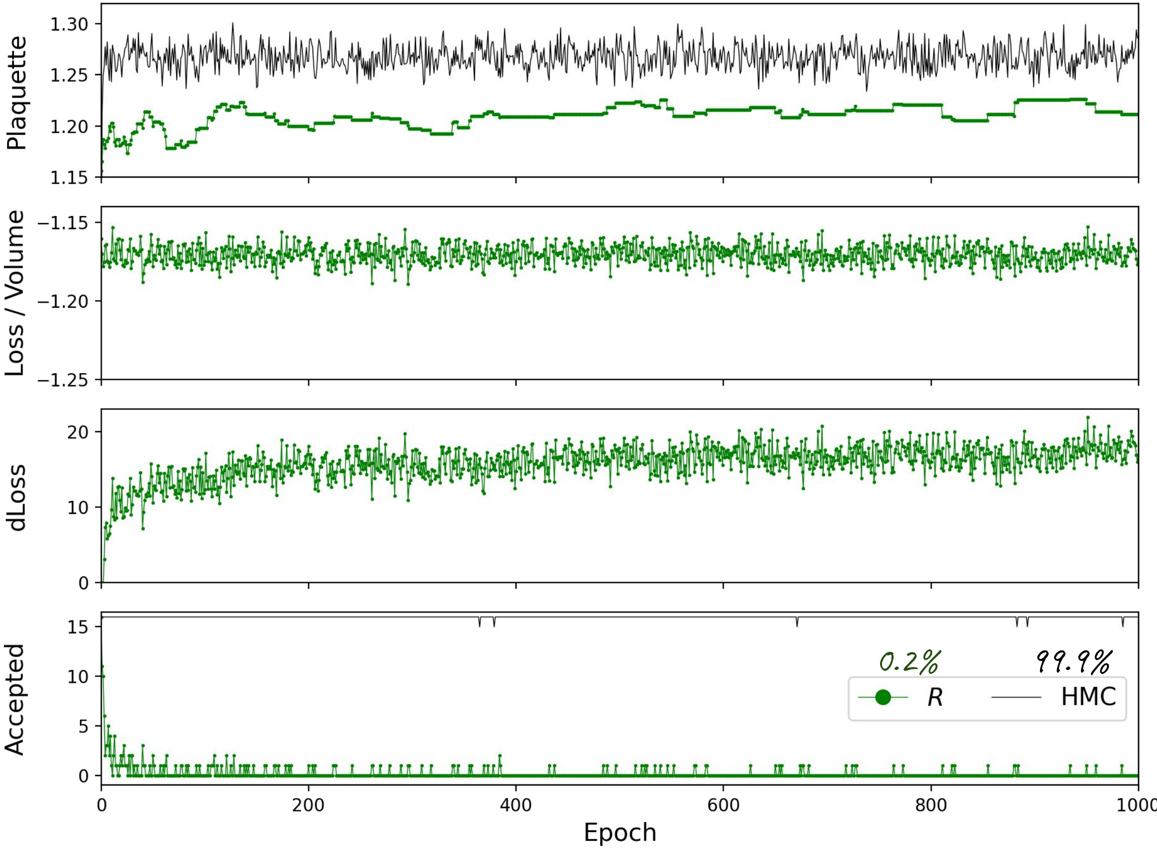


$$L_{KL} = S_{\text{target}}(U_T) + \int_0^1 \mathcal{L}\tilde{S}(U_t)dt$$

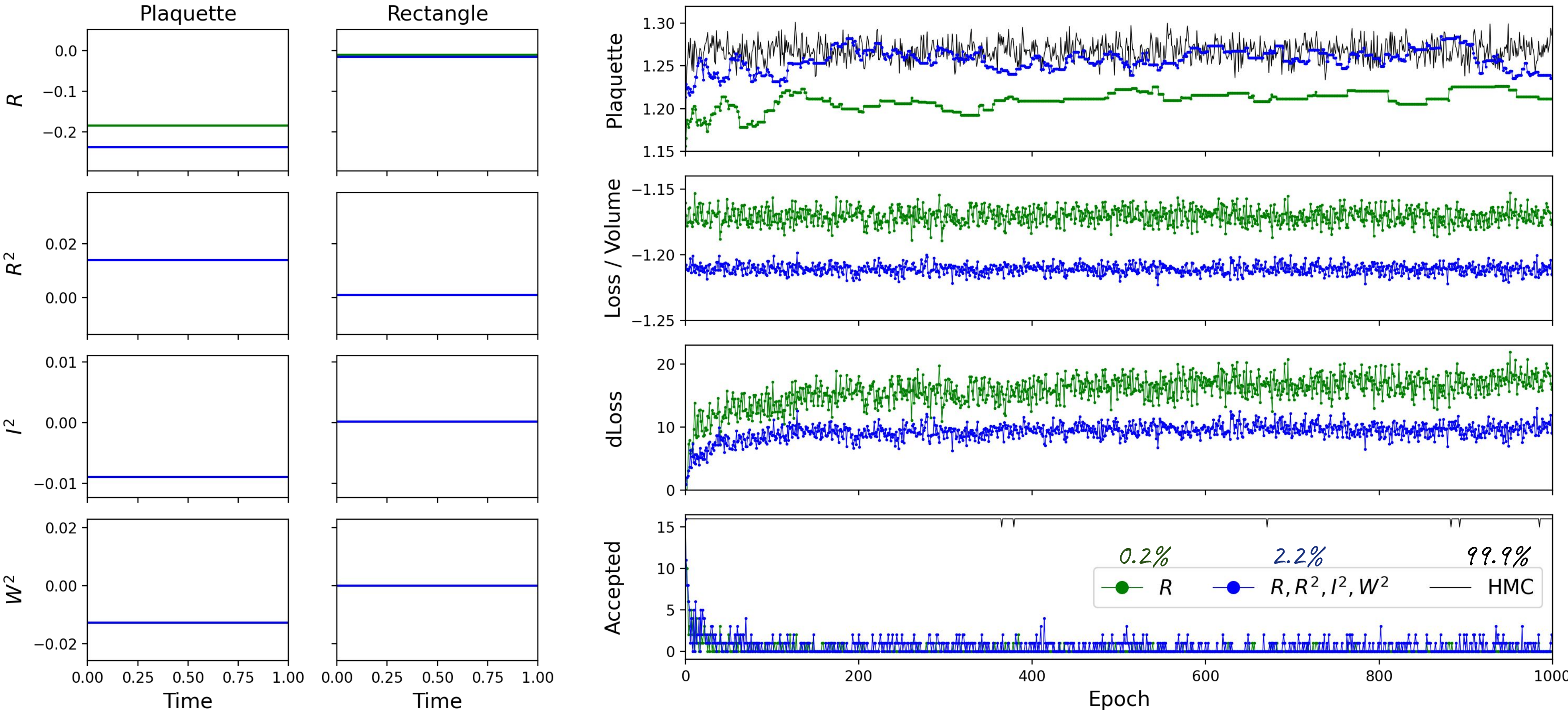
Acceptance probability:

$$\min\left(1, \frac{\exp(-L'_{KL})}{\exp(-L_{KL})}\right)$$

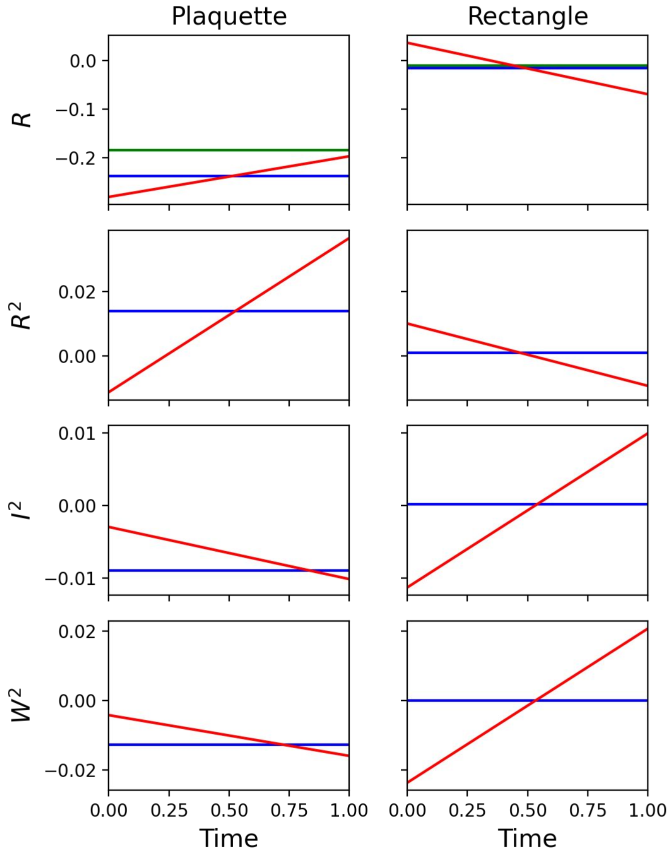
when sampling from uniform distribution

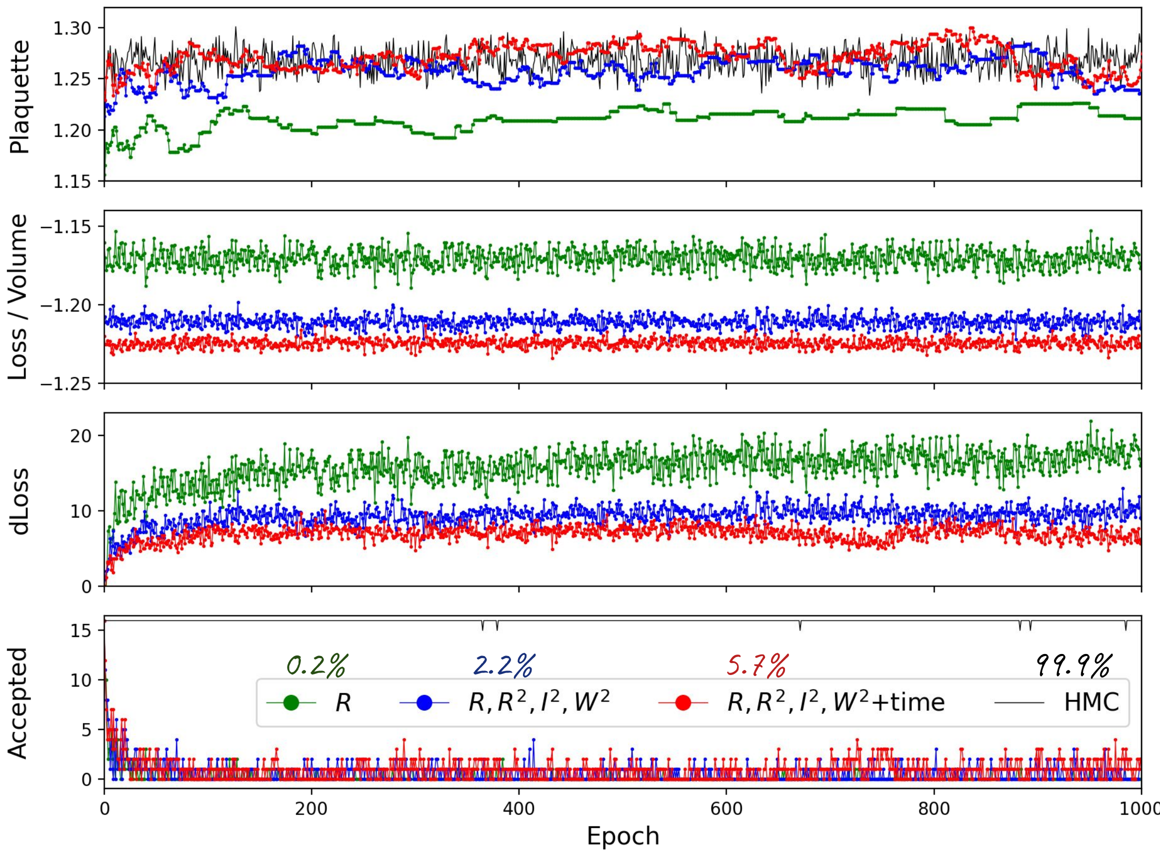


Latest results: NMCMC, 16^2, β=6



Latest results: NMCMC, 16^2, β=6





Conclusion

- Results for 16^2 at $\beta=6$:

ESS

0.02%	→	0.1%	→	0.5%	?	Goal ≳48%
2 params (plaq. + rect.)		**8 params** (plaq. + rect.) x (re,re^2,im^2,w^2)		**16 params** (plaq. + rect.) x (re,re^2,im^2,w^2) x 2 time (spline)		**O(10k) params?** [MIT, 2008.05456]

- Achievements:
  - ➢ Physical interpretation of parameters
  - ➢ Parameter transferring over volume
  - ➢ GPU and distributed implementation via QUDA
  - ➢ Generalization of Luesher approach
  - ➢ Parameter tuning via back propagation

- Open issues:
  - ➢ Sub-performing compared to normalizing flows
  - ➢ Manual implementation of models, not via ML libraries
  - ➢ Unstable tuning of time dependence due to degeneracy
  - ➢ Fermions not implemented yet, but doable
  - ➢ Integrator scaling when combining Lie groups and scalar

- Much more work to do and many idea... Working on first publication. Stay tuned!

Continuous Normalizing Flows for Lattice QCD

based on Trivializing Maps

Thank you for your attention!

Dr. Simone Bacchio

Computational Scientist
CaSToRC, The Cyprus Institute

A work in collaboration with

Pan Kessel, Stefan Schaefer, Lorenz Vaitl

Funded by
PRACE-6IP, WP8 "Forward Looking Software Solutions".
Grant agreement ID: 823767, Project name: LyNcs.

Runge-Kutta Integrators for scalar quantities

$$\frac{dy}{dt} = f(t, y)$$

$$y_{n+1} = y_n + h \sum_{i=1}^{s} b_i k_i$$

$$\begin{aligned}
k_1 &= f(t_n, y_n), \\
k_2 &= f(t_n + c_2 h, y_n + h(a_{21} k_1)), \\
k_3 &= f(t_n + c_3 h, y_n + h(a_{31} k_1 + a_{32} k_2)), \\
&\ \ \vdots \\
k_i &= f\left(t_n + c_i h, y_n + h \sum_{j=1}^{i-1} a_{ij} k_j\right).
\end{aligned}$$

Butcher table

$$\begin{array}{c|cccc}
c_1 & a_{11} & a_{12} & \dots & a_{1s} \\
c_2 & a_{21} & a_{22} & \dots & a_{2s} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\
\hline
 & b_1 & b_2 & \dots & b_s
\end{array}$$

Generic order 3

$$\alpha \neq 0, 2/3, 1$$

$$\begin{array}{c|ccc}
0 & 0 & 0 & 0 \\
\alpha & \alpha & 0 & 0 \\
1 & 1 + \frac{1-\alpha}{\alpha(3\alpha-2)} & -\frac{1-\alpha}{\alpha(3\alpha-2)} & 0 \\
\hline
 & \frac{1}{2} - \frac{1}{6\alpha} & \frac{1}{6\alpha(1-\alpha)} & \frac{2-3\alpha}{6(1-\alpha)}
\end{array}$$

Crouch-Grossman methods for Lie Groups

$$\dot{U} = g(U)U$$

$$\begin{aligned} k_i &= g(U^{(i)}) \\ U^{(i)} &= e^{h a_{i,i-1} k_{i-1}} \dots e^{h a_{i,1} k_1} U_n \\ U_{n+1} &= e^{h b_s k_s} \dots e^{h b_1 k_1} U_n \end{aligned}$$

order 1 : $\sum_i b_i = 1$

order 2 : $\sum_i b_i c_i = 1/2$

order 3 : $\sum_i b_i c_i^2 = 1/3 \quad \sum_{ij} b_i a_{ij} c_j = 1/6$

$\sum_i b_i^2 c_i + 2 \sum_{i<j} b_i c_i b_j = 1/3$

Butcher table

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \dots & a_{1s} \\ c_2 & a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \end{array}$$

Example tables for order 3

$$\begin{array}{c|ccc} 0 & & & \\ -1/24 & -1/24 & & \\ 17/24 & 161/24 & -6 & \\ \hline & 1 & -2/3 & 2/3 \end{array}$$

$$\begin{array}{c|ccc} 0 & & & \\ 3/4 & 3/4 & & \\ 17/24 & 119/216 & 17/108 & \\ \hline & 13/51 & -2/3 & 24/17 \end{array}$$

How to combine scalars' and Lie groups' integration?

$$U_{n+1} = \left(\prod_{i=1}^{s} e^{hb_i k_i}\right) U_n$$

$$k_i = g(U^{(i)})$$

$$U^{(i)} = \left(\prod_{j=1}^{i-1} e^{ha_{ij} k_{i-1}}\right) U_n$$

- **Different coefficient from standard RK**

$$y_{n+1} = y_n + h \sum_{i=1}^{s} b_i k_i$$

$$k_i = f(U^{(i)}, y^{(i)})$$

$$y^{(i)} = y_n + h \sum_{j=1}^{i-1} a_{ij} k_j$$

- **Needed for:** Laplacian, gradients, etc..

?

- Currently we use O(3) for Lie groups, how does scalar integration scale?
- Can we have a scheme that has O(3) for both? Maybe with 4 steps?