Machine Learning for Hadronic Correlators from Lattice QCD

Giovanni Pederiva
Collab: Andrea Shindler

28 September 2021

Michigan State University

Hadron Correlators

The two-point correlator is an important hadronic observable in Lattice QCD calculations. It can be interpreted via spectral decomposition [1]

$$C(t) = a^3 \sum_{\mathbf{x}} \langle O(t, \mathbf{x}) \bar{O}(0, \mathbf{0}) \rangle = \sum_k \langle 0 | \hat{O} | k \rangle \langle k | \hat{\bar{O}} | 0 \rangle e^{-tE_k}$$

Where $O(t)$ is an interpolating operator of the desired hadron state. For example:

Pion: $O_{\pi^+}(x) = \bar{d}(x)_{\alpha,c} (\gamma_5)_{\alpha\beta} u(x)_{\beta,c}$

Proton: $O_P(x) = \epsilon_{abc} u(x)_{\alpha,a} \left(u(x)^T_{\beta,b} C(\gamma_5)_{\beta\gamma} d(x)_{\gamma,c} \right)$

[1] Notation taken from *C. Gattringer and C. B. Lang, "Quantum Chromodynamics on the Lattice", Springer, 2010*

Quark Contractions

The correlator is then the result of the contraction between $O(x)$ and $\bar{O}(y)$. For pions, for example, it is:

$$
\begin{aligned}
&\langle O_{\pi^+}(x)\,\bar{O}_{\pi^+}(y)\rangle = \\
&= \langle \bar{d}(x)_{\alpha_1,c_1}(\gamma_5)_{\alpha_1\beta_1}\,u(x)_{\beta_1,c_1}\,\bar{u}(y)_{\alpha_2,c_2}(\gamma_5)_{\alpha_2\beta_2}\,d(y)_{\beta_2,c_2}\rangle \\
&= -\langle (\gamma_5)_{\alpha_1\beta_1}\,\bar{d}(x)_{\alpha_1,c_1}\,d(y)_{\beta_2,c_2}(\gamma_5)_{\alpha_2\beta_2}\,\bar{u}(y)_{\alpha_2,c_2}\,u(x)_{\beta_1,c_1}\rangle \\
&= -\langle \mathrm{tr}\left[\gamma_5 D_u^{-1}(y,x)\gamma_5 D_d^{-1}(x,y)\right]\rangle_G
\end{aligned}
$$

The Quark Propagator

For a generic quark flavor q, the term:

$$\langle q(x)_{\alpha,a}\bar{q}(y)_{\beta,b}\rangle = D^{-1}(y,x)^{\alpha\beta}_{ab}$$

is the inverse of the Dirac operator. In principle, one needs to invert the whole matrix. However, one can set a source (point-like in this case):

$$\eta_{\beta,b}(0) = \delta_{b,c_1}\delta_{\beta,\alpha_1}\delta_{y,0}$$

and reformulate the problem as:

$$D(0,x)^{\alpha\beta}_{ab}\, q(x)_{\alpha,a} = \eta_{\beta,b}(0),$$

equivalent to computing only one column of the inverse matrix.

A Linear System

The problem is now reduced to a set of **linear systems** of the very simple form $Dq = \eta$, where η are $3 \times 4 = 12$ different source vectors (Dirac and color indices).

The matrix D is the Dirac operator, a very sparse matrix (its exact form depends on the lattice action).

Note: for every quark flavor, we have an ensemble of linear systems.

Iterative Solvers

This kind of linear systems is usually solved using **iterative methods**. One of the simplest ones it the Conjugate Gradient, but many variations are used. For example, the BiCGStab is a common choice because it works for non-hermitian operators.

ALGORITHM 1 (BiCGStab). *For solving* $Az = b$ *choose an initial approximation* $z_0 \in \mathbb{C}^N$ *and set* $\tilde{r}_0 := \tilde{s}_0 := b - Az_0$. *Choose* $y_0 \in \mathbb{C}^N$ *such that* $\tilde{\delta}_0 := y_0^H \tilde{r}_0 \neq 0$ *and* $\varphi_0 := y_0^H A\tilde{s}_0/\tilde{\delta}_0 \neq 0$. *Then compute for* $n = 0, 1, \ldots$

$$\omega_n := 1/\varphi_n \,, \tag{25a}$$

$$\tilde{w}_{n+1} := \tilde{r}_n - A\tilde{s}_n\omega_n \,, \tag{25b}$$

$$\chi_n := (A\tilde{w}_{n+1})^H \tilde{w}_{n+1}/||A\tilde{w}_{n+1}||^2 \,, \tag{25c}$$

$$\tilde{r}_{n+1} := \tilde{w}_{n+1} - A\tilde{w}_{n+1}\chi_n \,, \tag{25d}$$

$$z_{n+1} := z_n + \tilde{s}_n\omega_n + \tilde{w}_{n+1}\chi_n \,, \tag{25e}$$

$$\tilde{\delta}_{n+1} := y_0^H \tilde{r}_{n+1} \,, \tag{25f}$$

$$\psi_{n+1} := -\omega_n\tilde{\delta}_{n+1}/(\tilde{\delta}_n\chi_n) \,, \tag{25g}$$

$$\tilde{s}_{n+1} := \tilde{r}_{n+1} - (\tilde{s}_n - A\tilde{s}_n\chi_n)\,\psi_{n+1} \,, \tag{25h}$$

$$\varphi_{n+1} := y_0^H A\tilde{s}_{n+1}/\tilde{\delta}_{n+1} \,. \tag{25i}$$

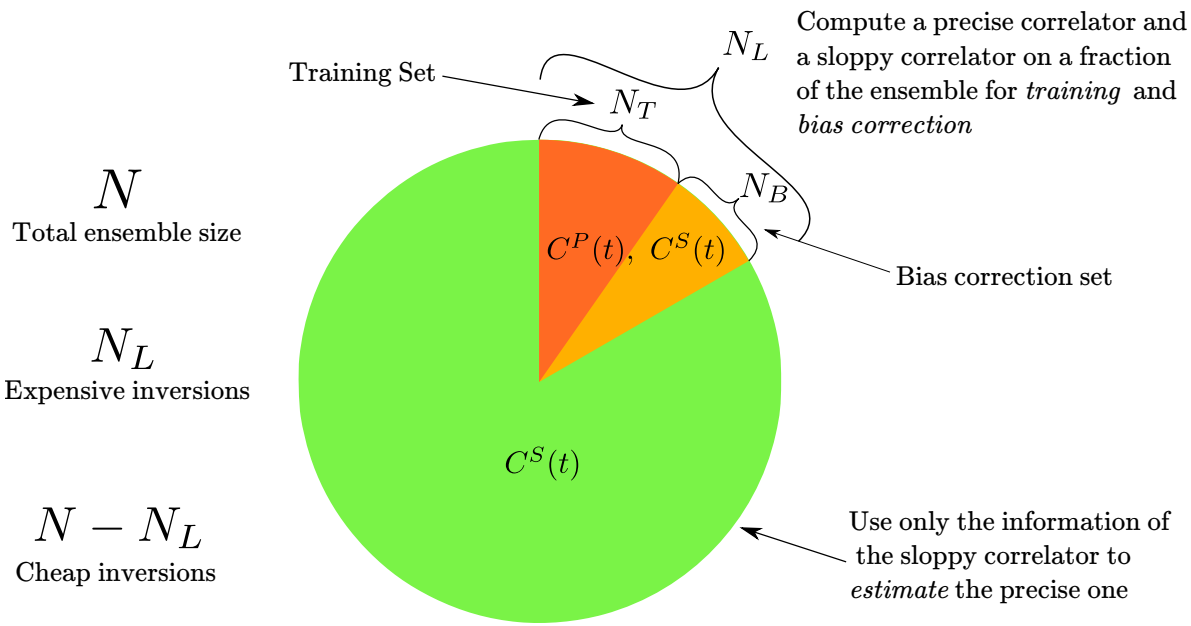
These iterative solvers are terminated at convergence, when thre residue is smaller than a fixed **stopping parameter** ϵ:

$$||Ax_n - b|| < \epsilon$$

The Goal

The main idea of this work is to try to accelerate the computation of the linear system for the quark propagator. We use numerical data for different stopping parameters ϵ to as training and prediction data sets.

For example, using a precise measurement of the propagator ($\epsilon = 10^{-8}$) on a subset of the ensemble and a less precise (sloppy) one ($\epsilon = 10^{-1}, 10^{-2}, 10^{-3}$) on the whole ensemble.



To properly estimate the uncertainty bias-correction and boostrap are used.

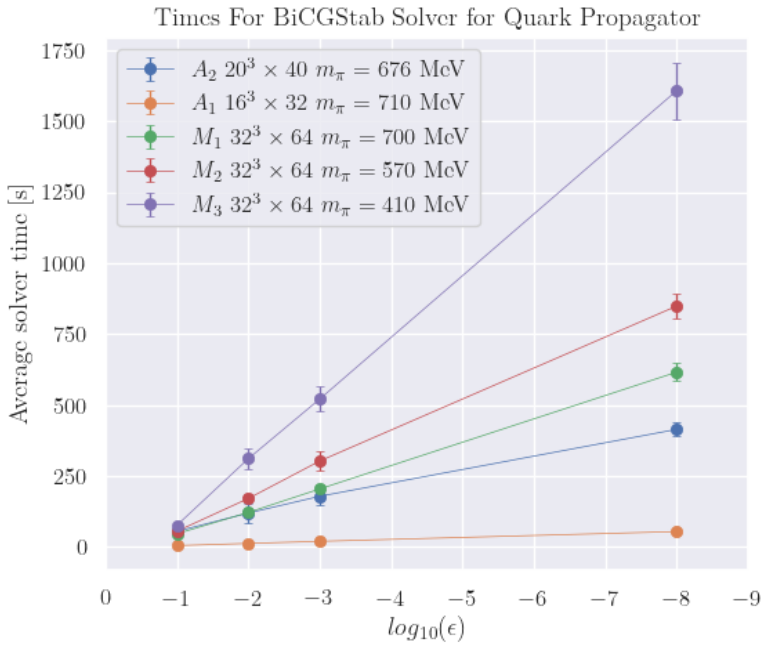
Gauge Field Ensembles Used

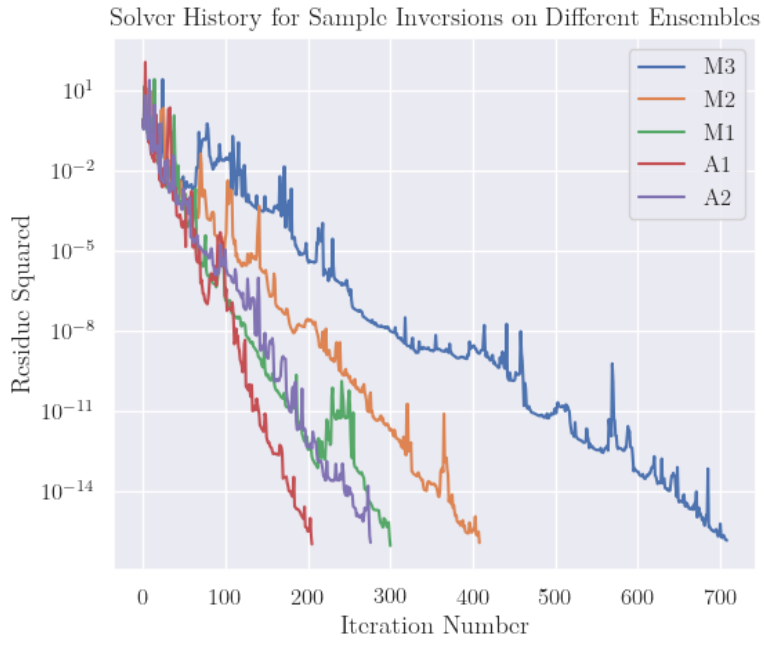
	β	κ_l	κ_s	L/a	T/a	$a\,[fm]$	$m_\pi\,[MeV]$	N
M_1	1.90	0.13700	0.1364	32	64	0.0907(13)	699.0(3)	399
M_2	1.90	0.13727	0.1364	32	64	0.0907(13)	567.6(3)	400
M_3	1.90	0.13754	0.1364	32	64	0.0907(13)	409.7(7)	450
A_1	1.83	0.13825	0.1371	16	32	0.1095(25)	710(1)	800
A_2	1.90	0.13700	0.1364	20	40	0.0936(33)	676.3(7)	790

Ensembles from the PACS-CS collaboration[2], with clover fermions.
Physical quantites calculated for another work[3]

[2] PACS-CS, S. Aoki et al., Phys. Rev.D79, 034503 (2009), 0807.1661
[3] J. Dragos, A. Shindler et al., (2019), 10.1103/PhysRevC.103.015202

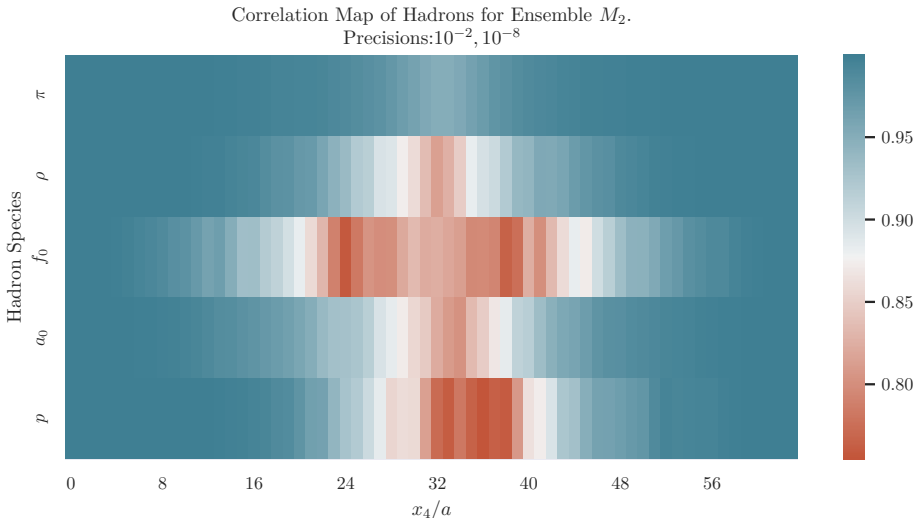
Times





The time to solution of the linear system is roughly linear with the log of the stopping parameter ϵ.

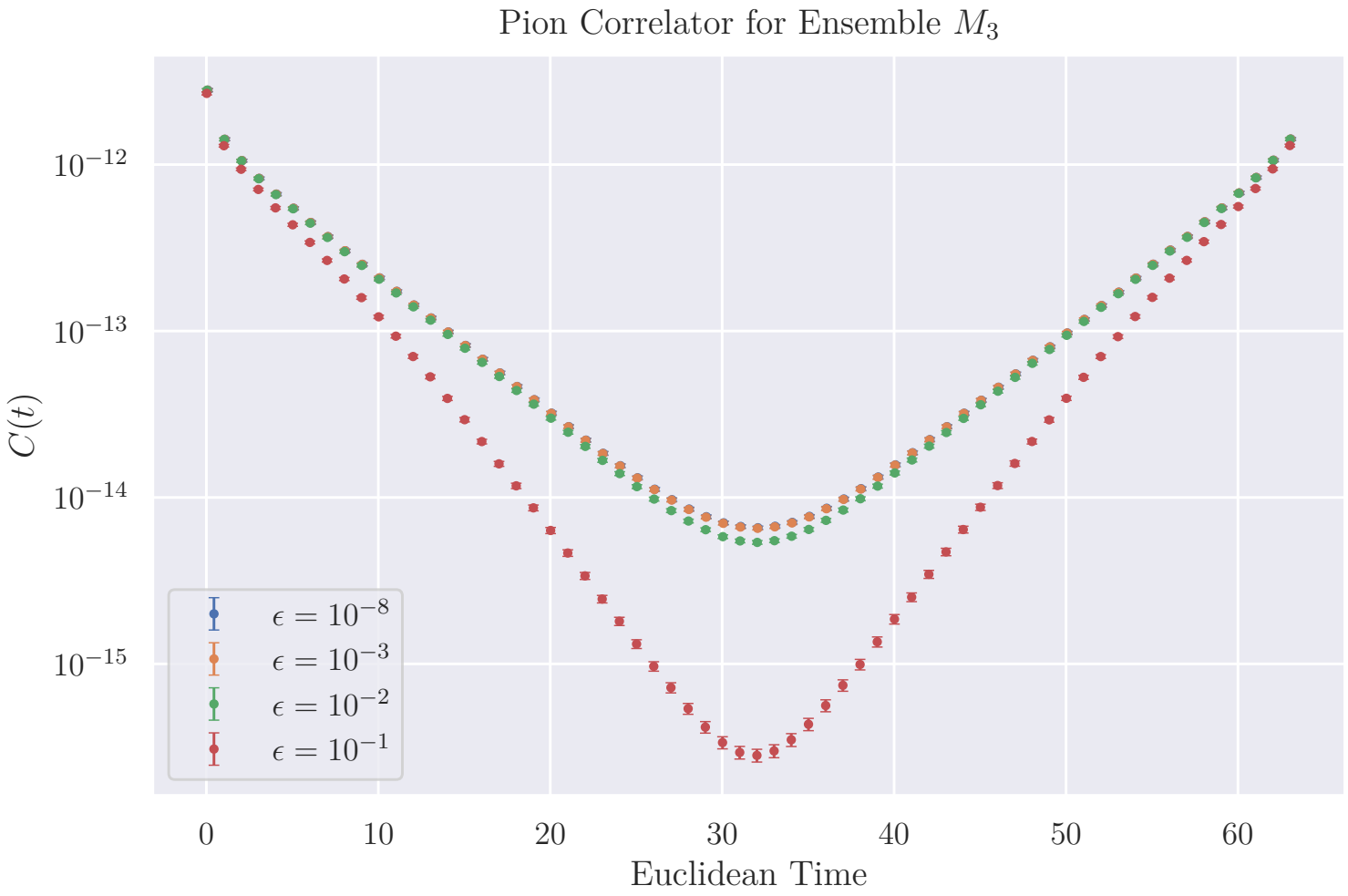
Correlations Maps



$$\Gamma(P,S) = \frac{1}{N\sigma_P\sigma_S}\sum_i^N (C_i^P - \bar{C}^P)(C_i^S - \bar{C}^S)$$

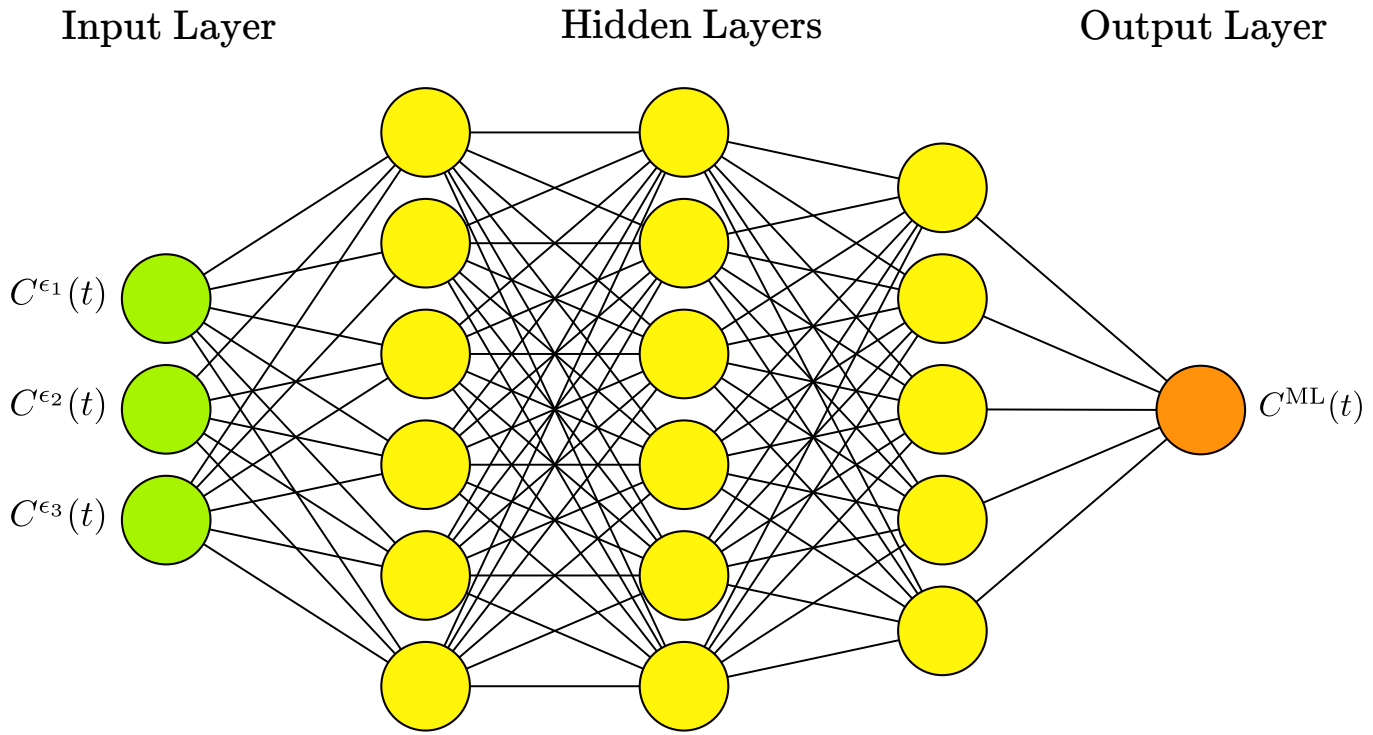
Correlation between hadron correlators on ensemble M_2 computed with $\epsilon = 10^{-2}$ and $\epsilon = 10^{-8}$.

Example of Raw Data



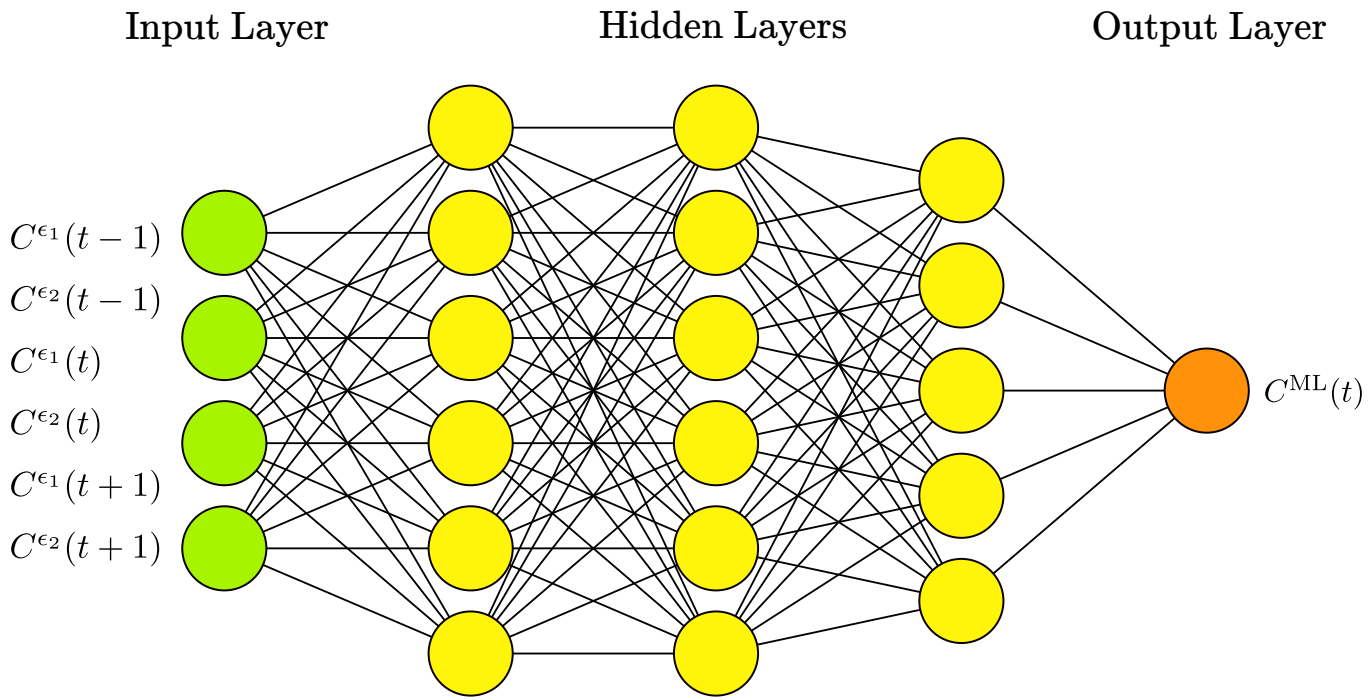
Single Point ML Models Used

- **SPNN2**: a Neural Network using $\epsilon = 10^{-1}, 10^{-2}$ as input and $\epsilon = 10^{-8}$ as target
- **SPNN3**: a Neural Network using $\epsilon = 10^{-1}, 10^{-2}, 10^{-3}$ as input and $\epsilon = 10^{-8}$ as target



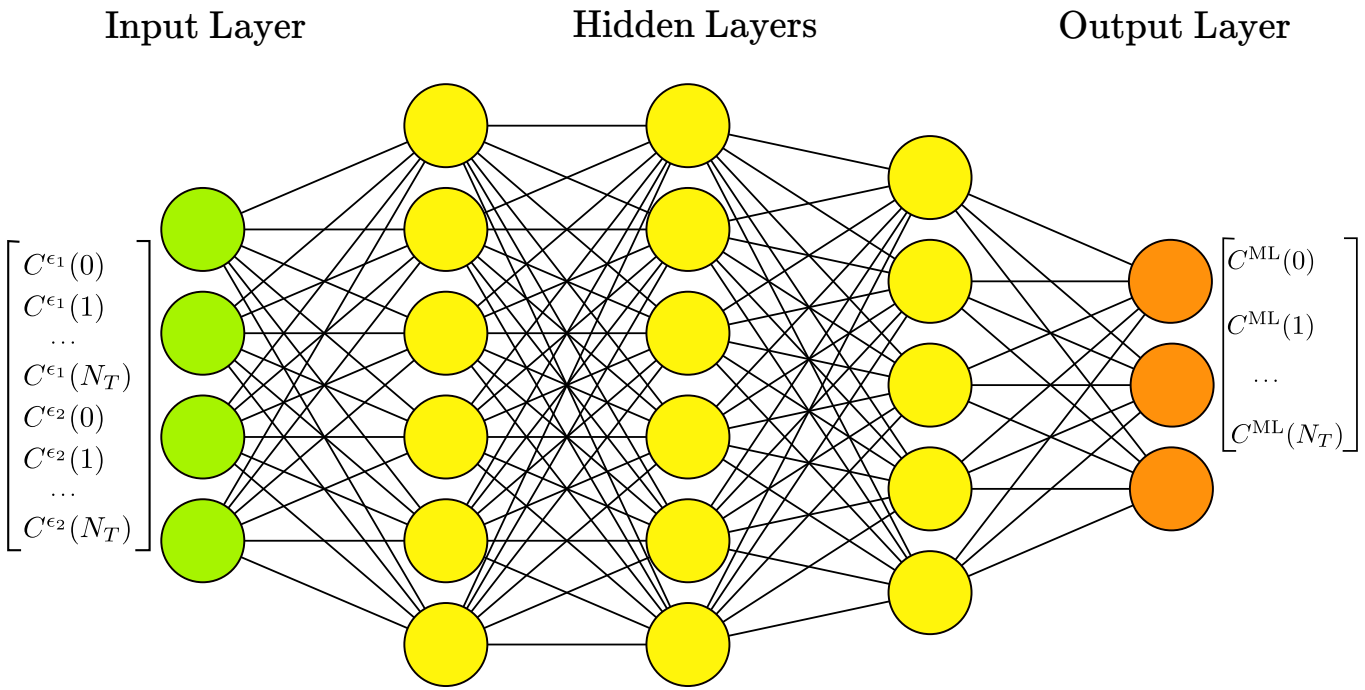
Single Point ML Models Used

- **SPNN_t1**: a Neural Network using $\epsilon = 10^{-1}, 10^{-2}$ at time t and $t \pm 1$ as input and $\epsilon = 10^{-8}$ as target
- **SPNN_t2**: a Neural Network using $\epsilon = 10^{-1}, 10^{-2}$ at time t, $t \pm 1$ and $t \pm 2$, as input and $\epsilon = 10^{-8}$ as target



Global ML Models Used

- **GNN2**: a Neural Network using $\epsilon = 10^{-1}, 10^{-2}$ at all times t at once $\epsilon = 10^{-8}$ at all times as target
- **GNN3**: a Neural Network using $\epsilon = 10^{-1}, 10^{-2}, 10^{-3}$ at all times t at once $\epsilon = 10^{-8}$ at all times as target



Evaluating Performance

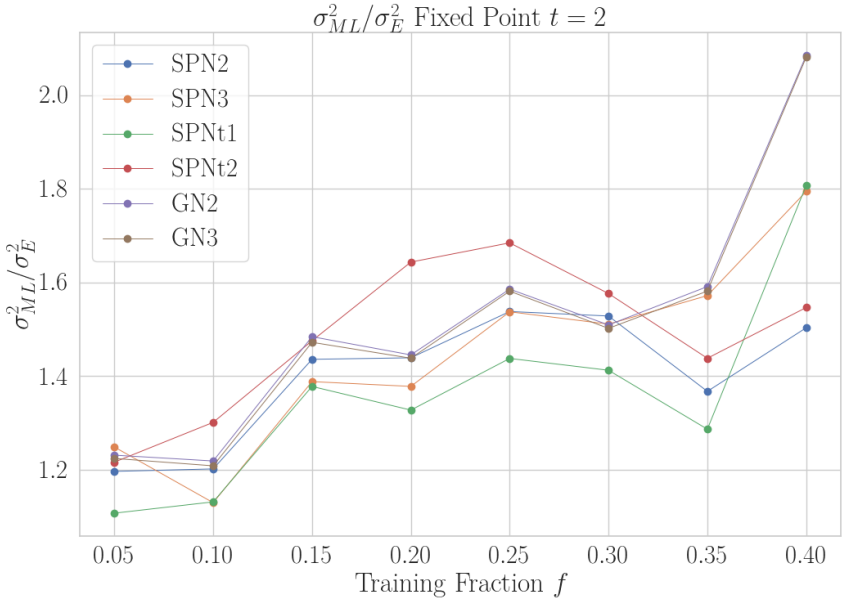
In order to asses the quality of the results we compute three quantities:

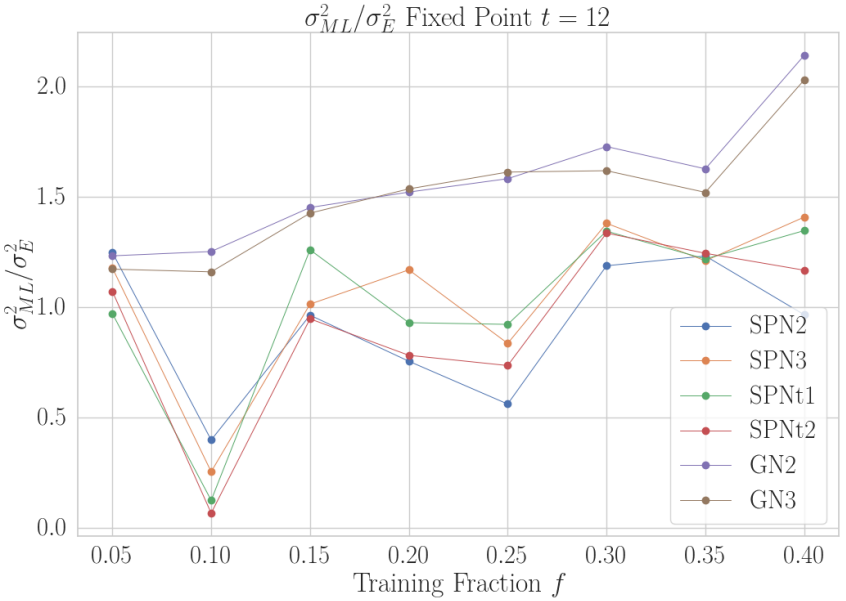
- the computational gain for given training fraction f.
  $\Gamma(f) = \frac{t_{\epsilon_{min}}}{t_p}(1-f) + f$
- the ratio of the variances $\frac{\sigma^2_{ML}}{\sigma^2_E}$
- the compatibility of the results $\frac{|\langle O \rangle_{ML} - \langle O \rangle_E|}{\sqrt{\sigma^2_{ML} + \sigma^2_E}}$

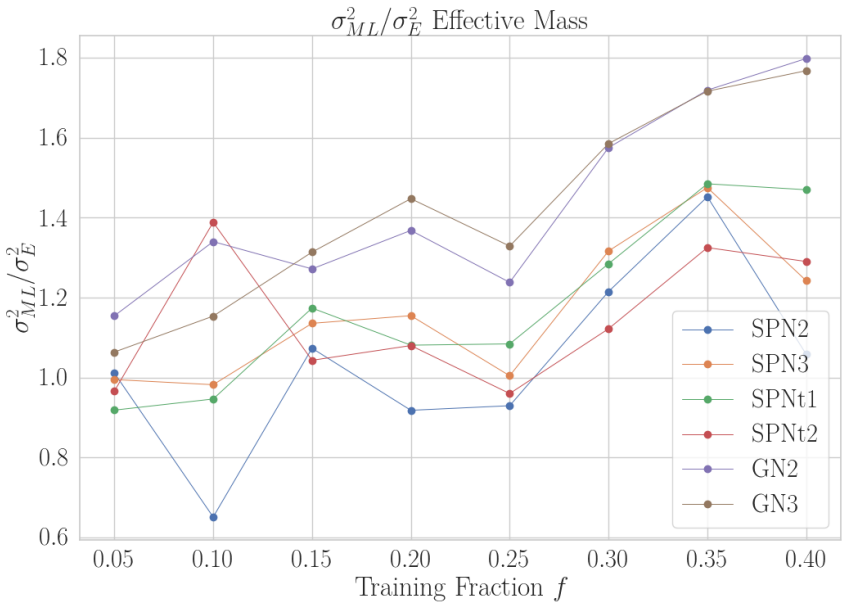
These have been computed for a small euclidean time, an asymptotic time and on the effective mass

The product $\Pi(f) = \left(\Gamma(f)\frac{\sigma^2_{ML}}{\sigma^2_E}\right)^{-1}$ will be our main metric. If above 1 we are gaining.

Results







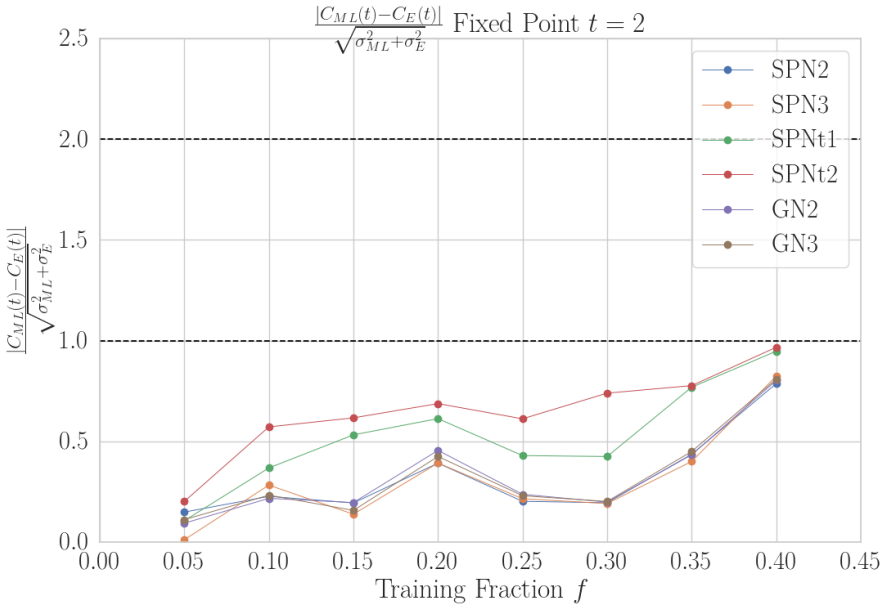
Ensemble Properties:

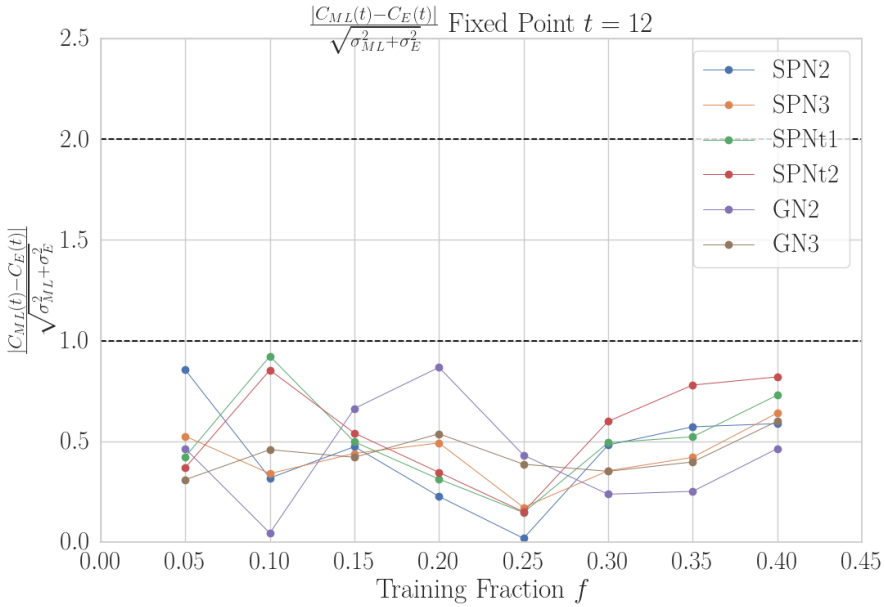
$20^3 \times 40$ Lattice

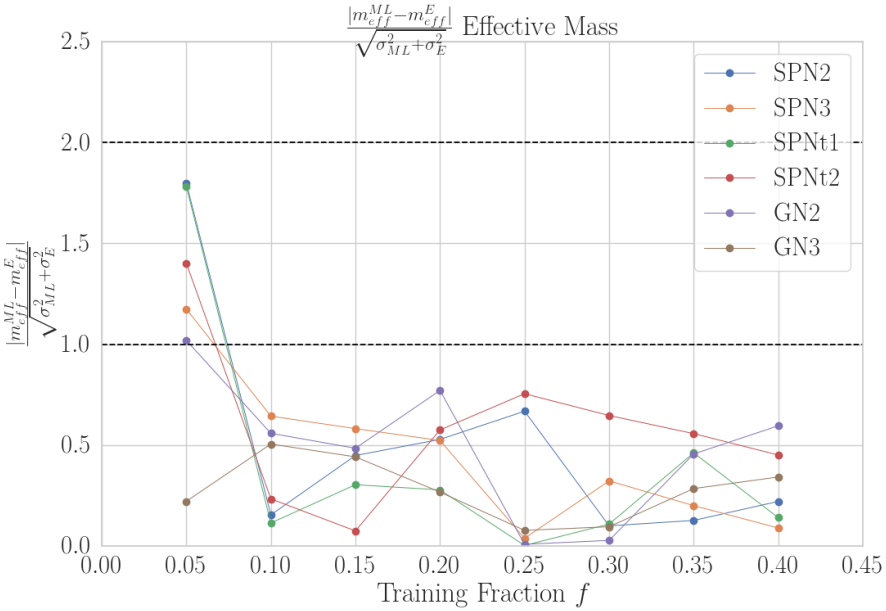
676 MeV pion mass

600 configurations

Results







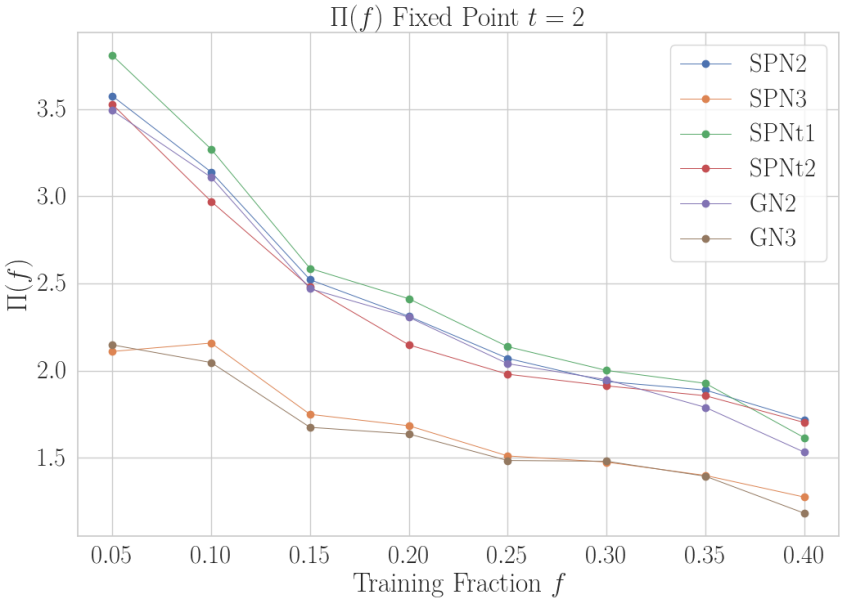
Ensemble Properties:

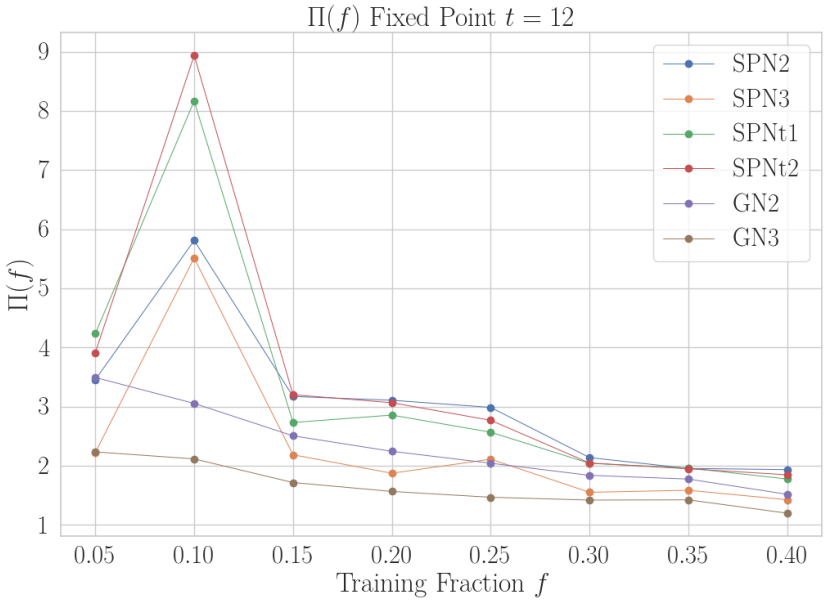
$20^3 \times 40$ Lattice

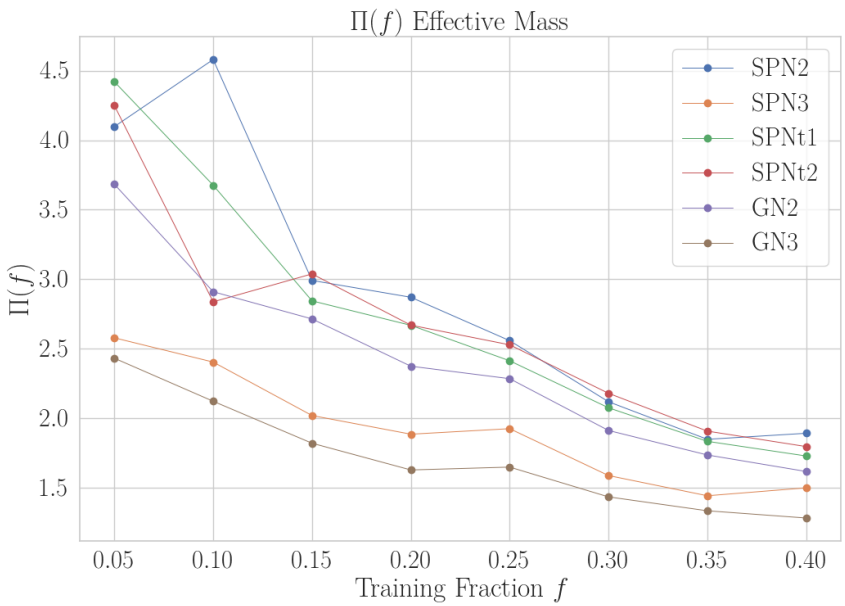
676 MeV pion mass

600 configurations

Results







Ensemble Properties:

$20^3 \times 40$ Lattice

676 MeV pion mass

600 configurations

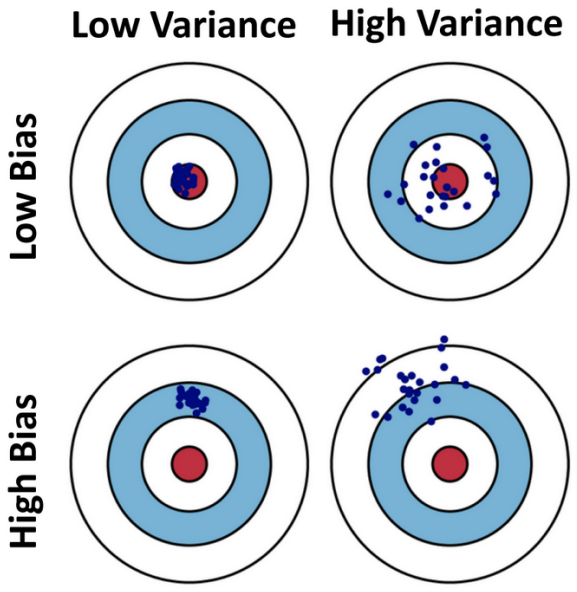
Summary and Outlook

- A 2.5 speedup is consistently achieved in the m_π range $400 - 700$ MeV and lattice spacings $0.09 - 0.11$ fm
- The overall method appears to be solid and stable in the cases we have tested it so far
- Need to define an operative procedure, maybe splitting the methods depending on t
- We are testing partially quenched cases with heavy quarks to see if there are larger gains at higher masses, where numerical solver precisions are critical

Thank You

Bias correction

When fitting, there could be some bias on the sample average depending on the subset used for training:



So we further split our training data set and compute the expectation value as:

$$\bar{C} = \frac{1}{N - N_L} \sum_{i \in prediction} C_i^P + \frac{1}{N_B} \sum_{i \in bias_corr} (C_i - C_i^P)$$

Bootstrapping

To estimate the error on the expectation value of the observable, multiple bootstrap samples are used.

Boostrapping is a common resampling method used in LQCD analysis. It consists of taking a random sample of a quantity O from a given set of N data with repetitions. This is performed K times:

$$C_k = \frac{1}{N} \sum_{i}^{N} C_i^*$$

One then sets the estimator of O as:

$$\bar{C} = \frac{1}{K} \sum_{i}^{K} C_k, \qquad \sigma_C^2 = \bar{C} = \frac{1}{K} \sum_{i}^{K} (\bar{C} - C_k)^2$$

The training and prediction set are bootstrapped independently.