BPCS 2025

Enhancing Phase Transition Calculations through Fitting and Neural Network

Yang Zhang, Henan Normal University

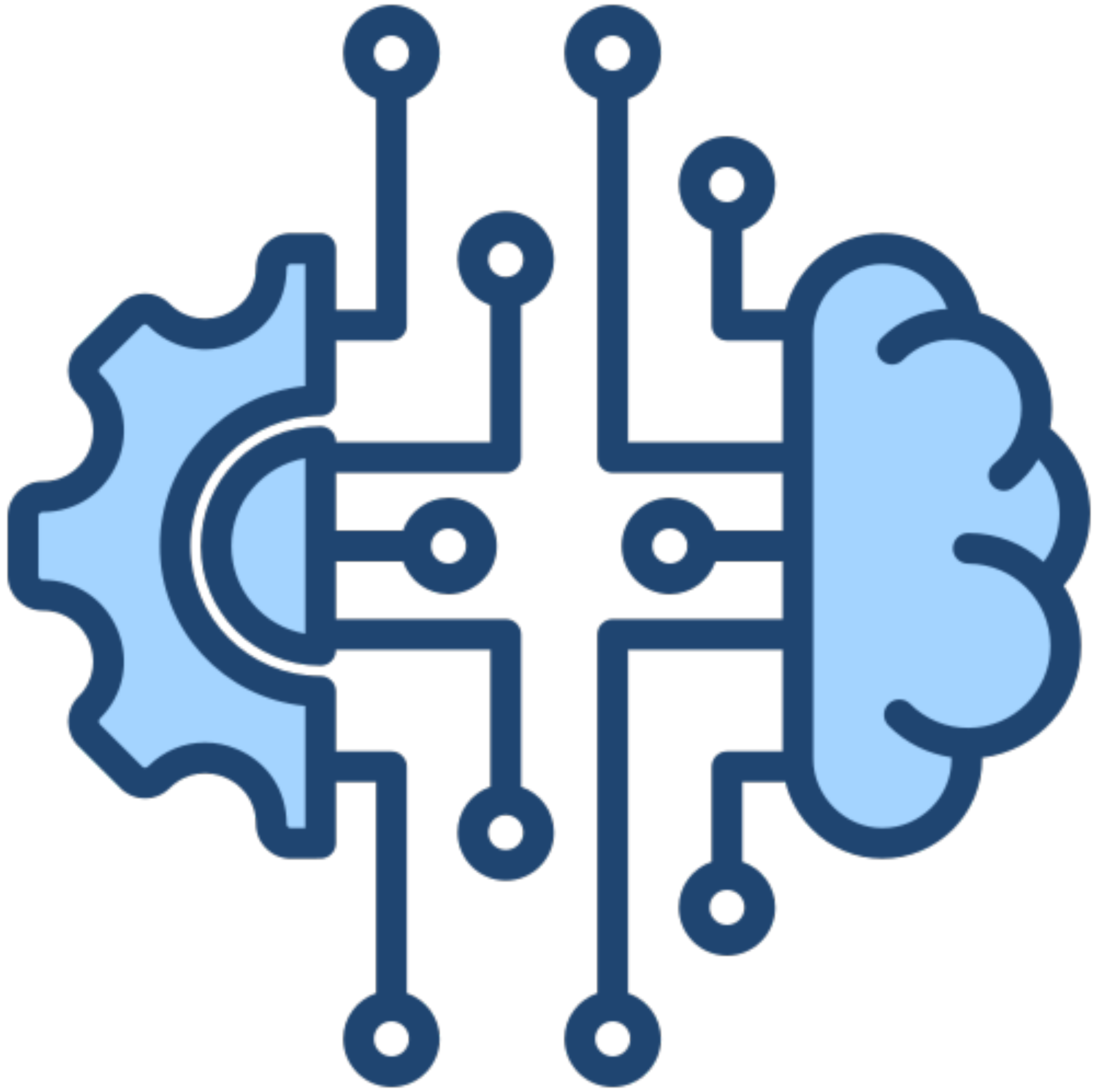
In collaboration with Ligong Bian, Hongxin Wang, Yang Xiao, Ji-Chong Yang, Jin Min Yang

Based on arXiv:2510.XXXXX

河南师范大学理论物理中心

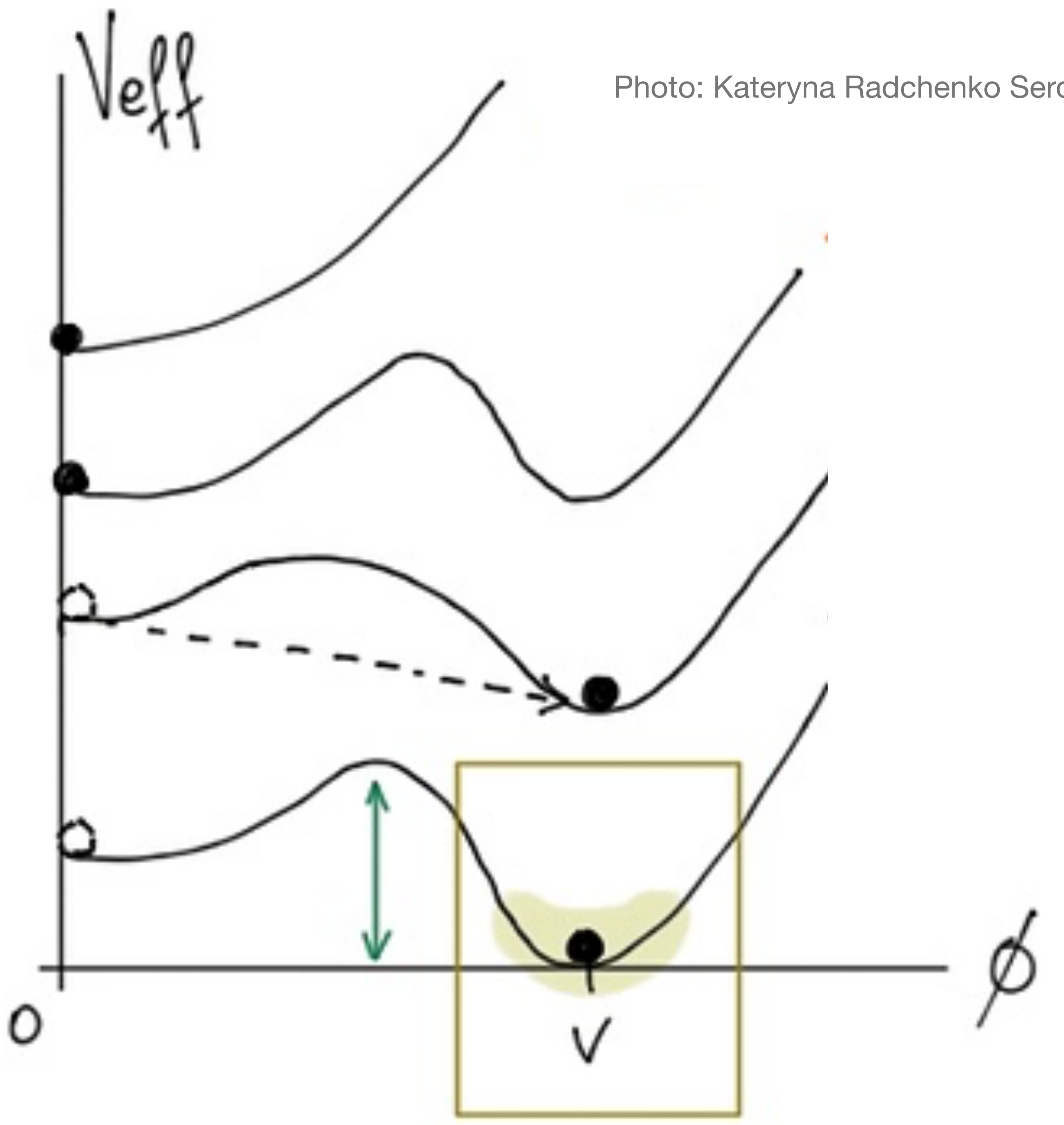
Centre for Theoretical Physics

Motivation



Machine Learning

+



Cosmological Phase Transition

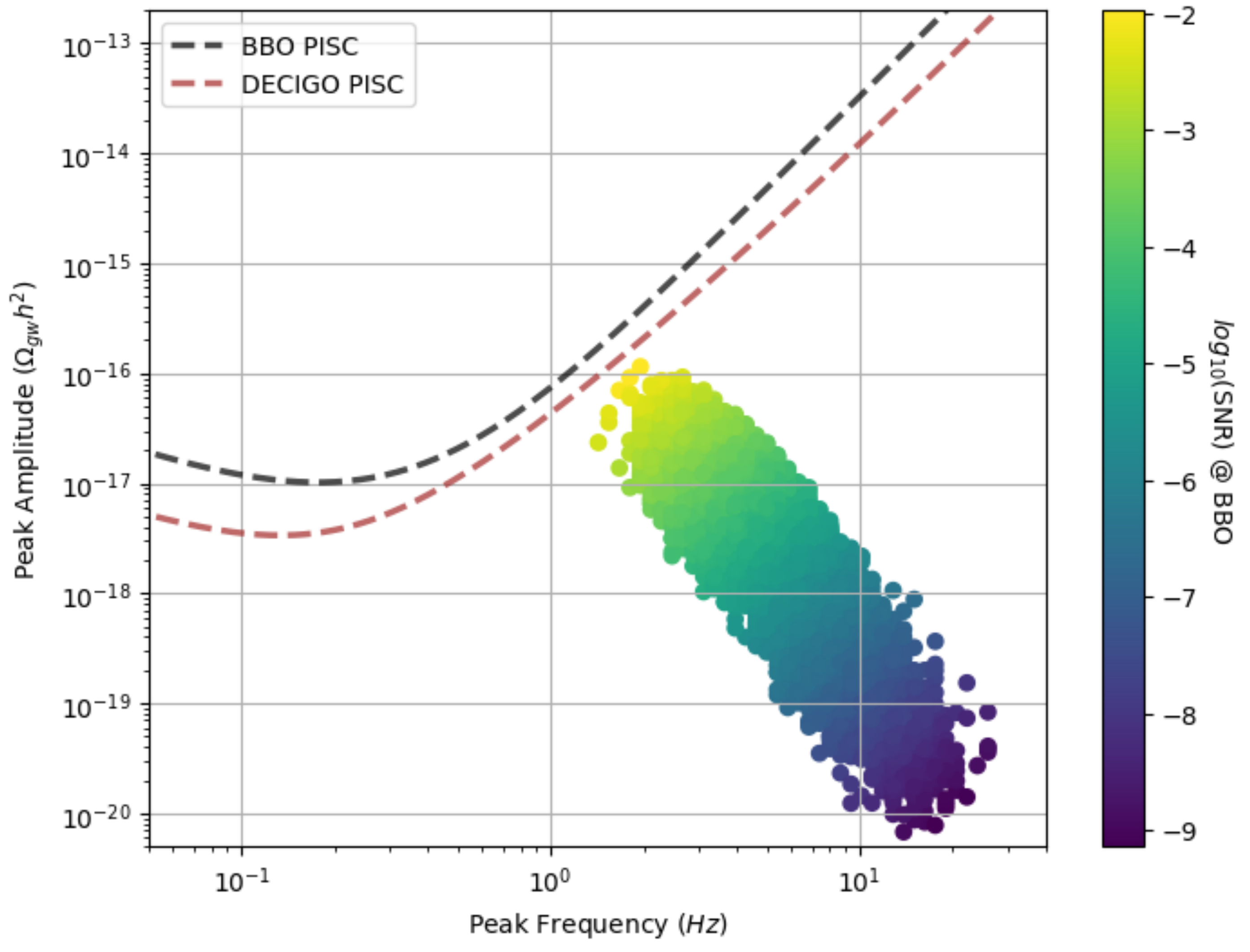
Machine Learning in Phase Transition

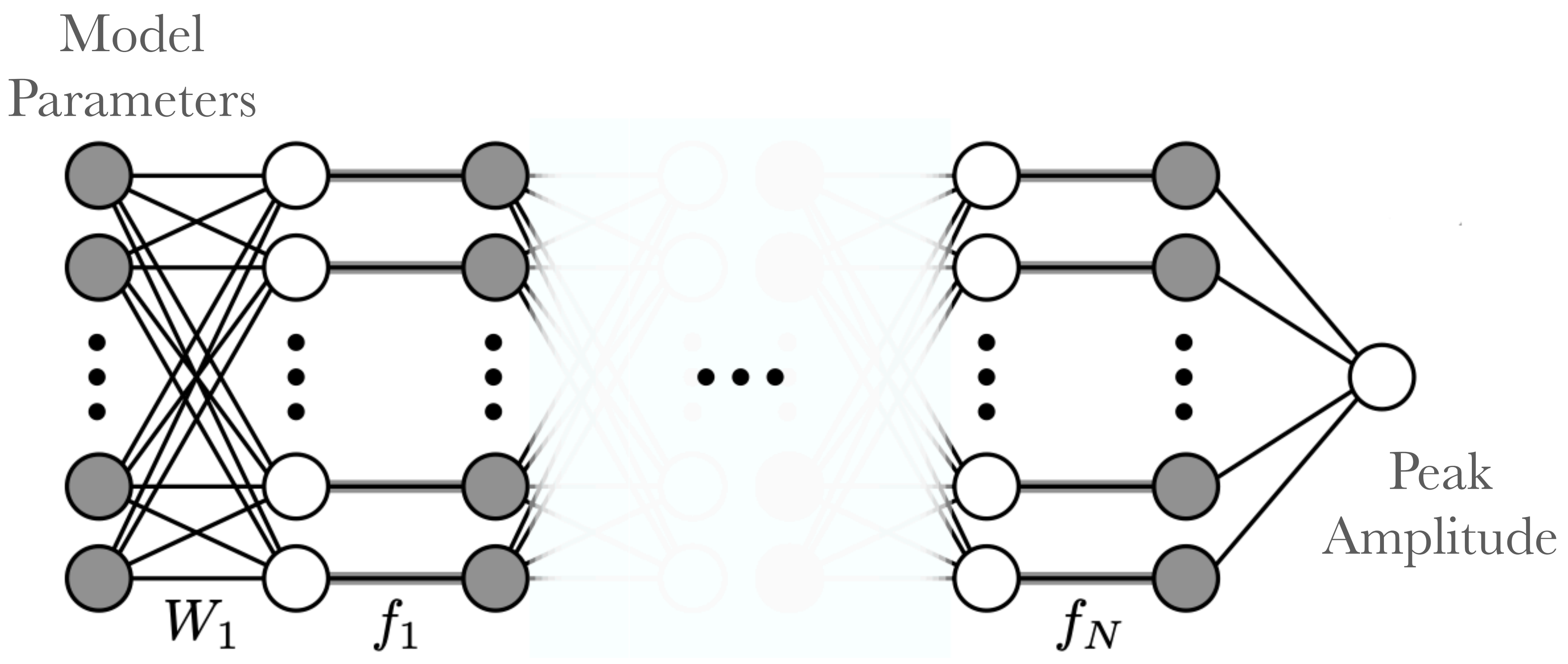
Machine Learning Left-Right Breaking from Gravitational Waves, William Searle, Csaba Balázs, Yang Xiao, Yang Zhang, arXiv:2506.09319

$$SU(2)_R \times SU(2)_L \times U(1)_{B-L} \xrightarrow{\langle \Delta_R \rangle \neq 0} SU(2)_L \times U(1)_Y \xrightarrow{\langle \phi \rangle, \langle \Delta_L \rangle \neq 0} U(1)_{em}$$

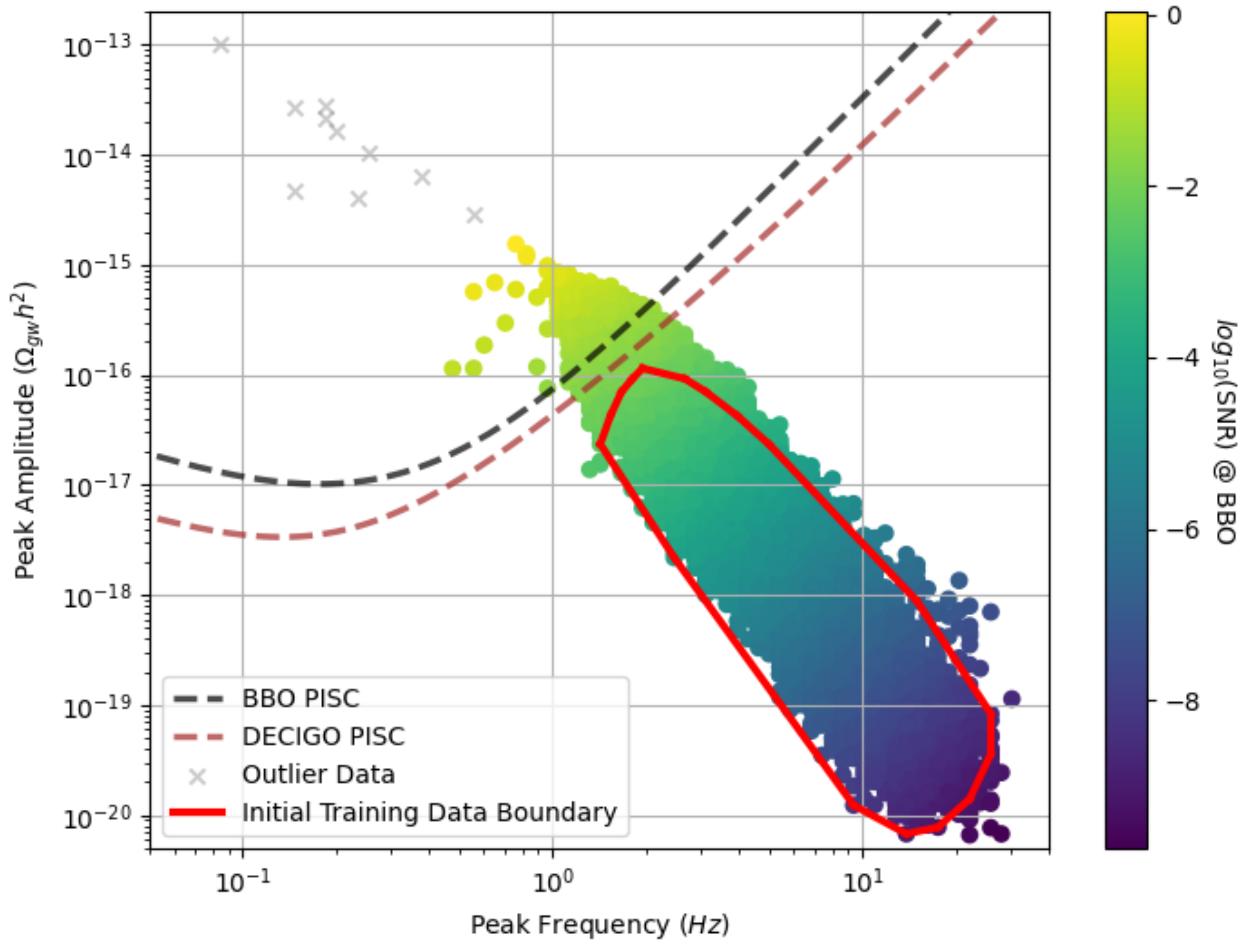
Add Random Samples

Initial Data Set → Deep Neural Network → Recommended samples





$64 \to 128 \to 128 \to 128 \to 128 \to 64$



Machine Learning in Phase Transition

Solving differential equations with neural networks: Applications to the calculation of cosmological phase transitions

Maria Laura Piscopo, Michael Spannowsky, Philip Waite, arXiv:1902.05563

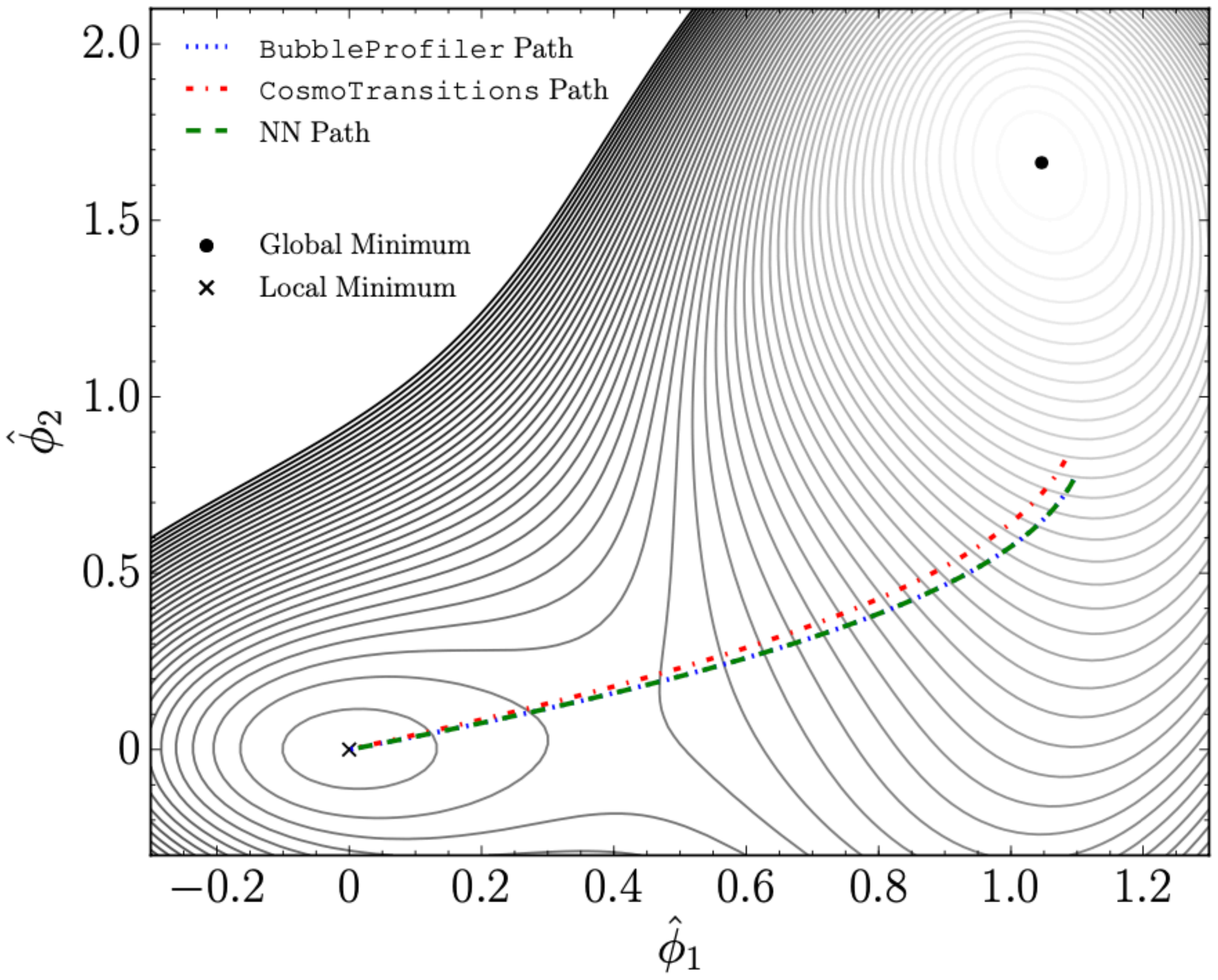
- A set of m coupled jth order differential equations:

$$\mathcal{F}_m(\vec{x}, \phi_m(\vec{x}), \nabla\phi_m(\vec{x}), \cdots, \nabla^j\phi_m(\vec{x})) = 0$$

- Convert the problem of finding a solution into an optimization problem:

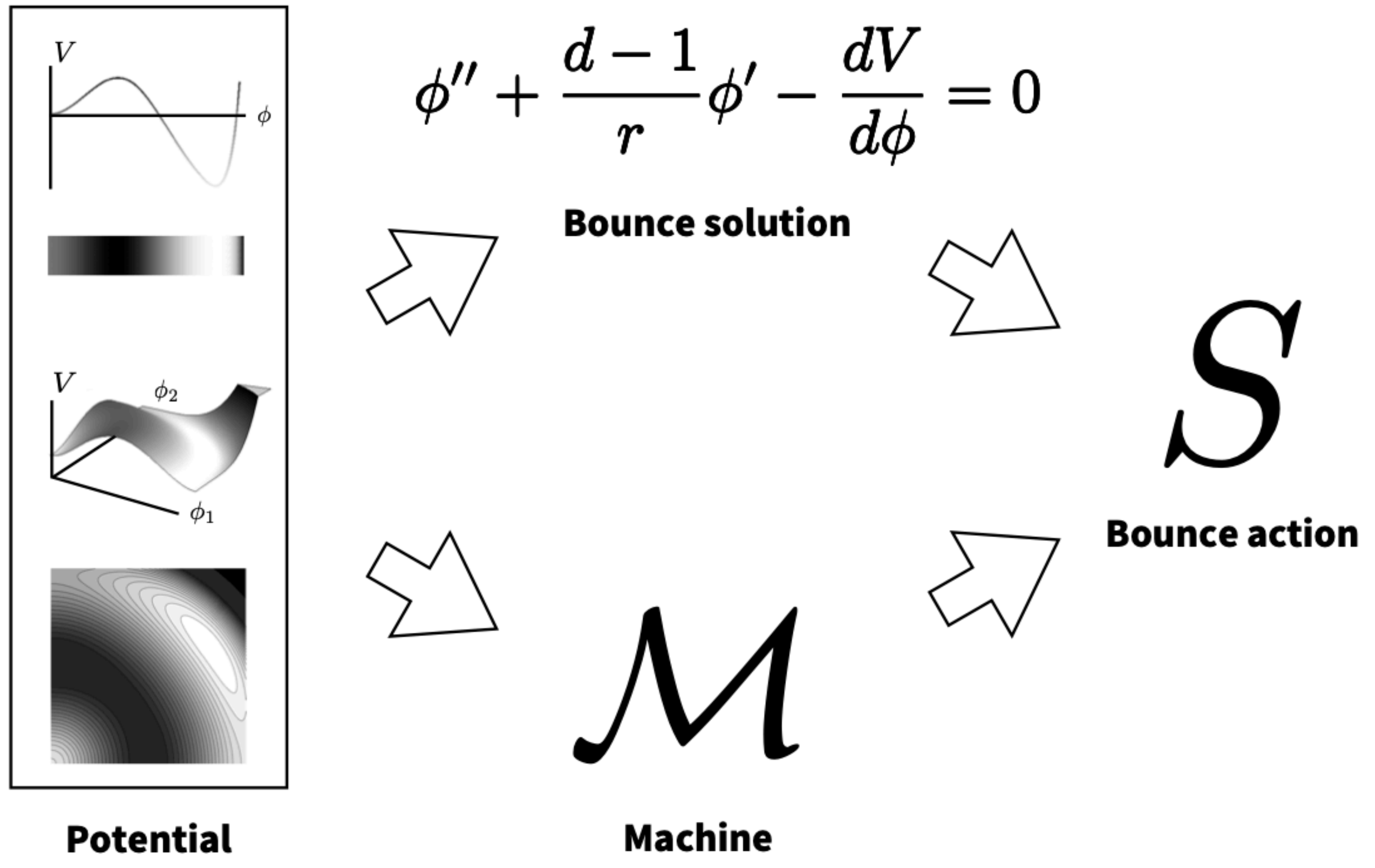
$$\mathcal{L}(\{w, \vec{b}\}) = \frac{1}{i_{\max}} \sum_{i,m} \hat{\mathcal{F}}_m(\vec{x}^i, \hat{\phi}_m(\vec{x}^i), \cdots, \nabla^j\hat{\phi}_m(\vec{x}^i))^2 + \sum_{\text{B.C.}} (\nabla^p\hat{\phi}_m(\vec{x}_b) - K(\vec{x}_b))^2 \,,$$

- Minimize the loss function using a neural network.



Machine Learning in Phase Transition

Machine learning for bounce calculation, Ryusuke Jinno, arXiv:1805.12153



$$\text{Class 1 } (C_1): \; V(\phi) = \sum_{n=1}^{7} a_n^{(1)} \phi^{n+1},$$

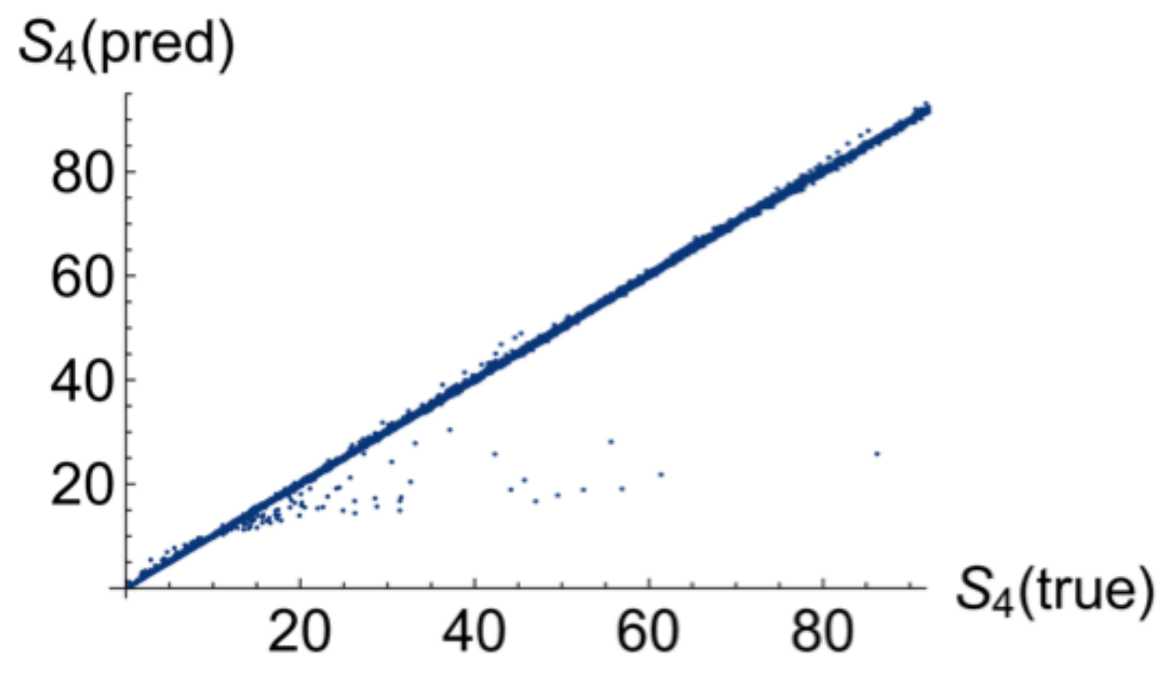
$$\text{Class 2 } (C_2): \; V(\phi) = \sum_{n=1}^{7} a_n^{(2)} \phi^{2n},$$

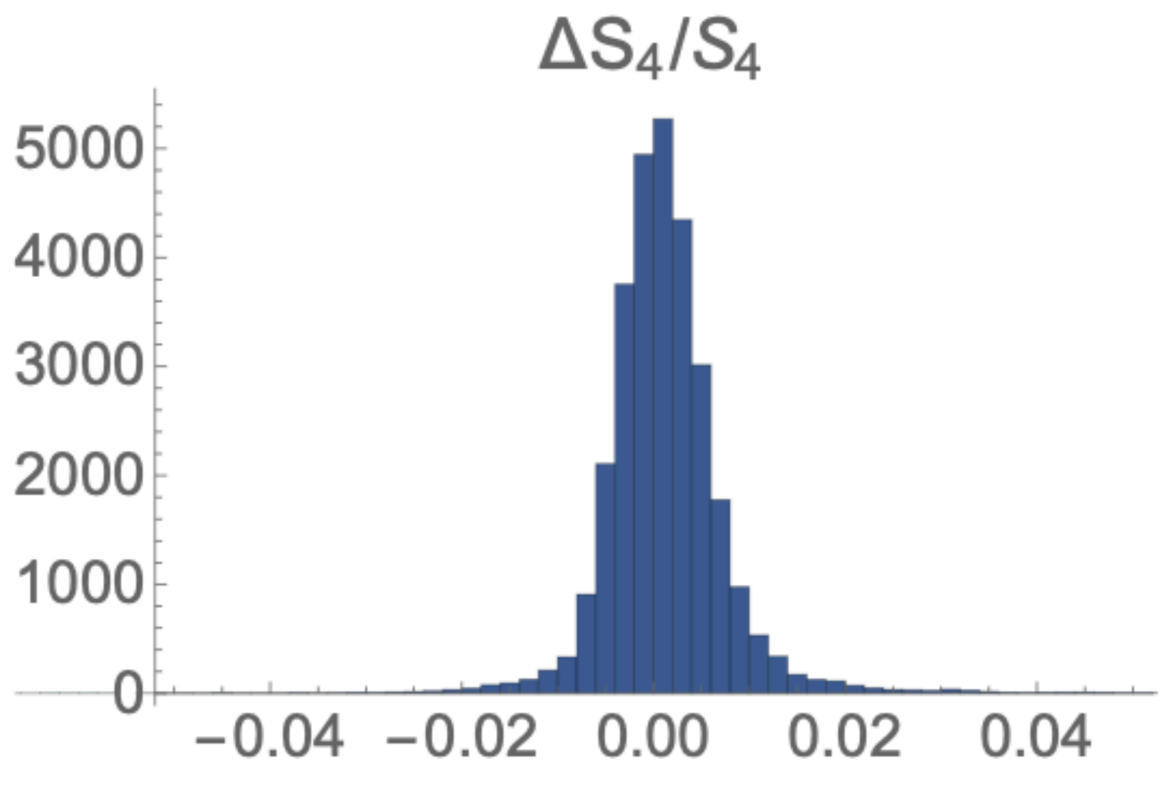
$$\text{Class 3 } (C_3): \; V(\phi) = a_1^{(3)} \phi^2 + \sum_{n=2}^{7} a_n^{(3)} \phi^{2n-1}.$$

- Inputs: values of the potential and its derivatives

$$x_{\text{in}} = \left\{ V(\phi_{\text{sample}}) \,\middle|\, \phi_{\text{sample}} = \frac{1}{16}, \frac{2}{16}, \cdots, \frac{15}{16} \right\}$$
$$\oplus \left\{ \frac{dV}{d\phi}(\phi_{\text{sample}}) \,\middle|\, \phi_{\text{sample}} = \frac{1}{16}, \frac{2}{16}, \cdots, \frac{15}{16} \right\}$$
$$\oplus \left\{ \frac{d^2V}{d\phi^2}(\phi_{\text{sample}}) \,\middle|\, \phi_{\text{sample}} = \frac{0}{16}, \frac{1}{16}, \cdots, \frac{16}{16} \right\}.$$

Training & Test : $C_1+C_2+C_3$ / Applied to : $C_1+C_2+C_3$





Transition parameters

- Bubble nucleation rate

$$\frac{\Gamma}{V} = A(T)e^{-S_E(T)/T}[1 + \mathcal{O}(\hbar)],\ A(T) = \left(\frac{S_E(T)}{2\pi T}\right)^3 D(T)$$

- Nucleation temperature

$$N(T) = \int_{T_{\rm nuc}}^{T_{\rm tra}} \frac{{\rm d}T}{T}\frac{\Gamma(T)}{H(T)^4} = 1$$

- Percolation temperature

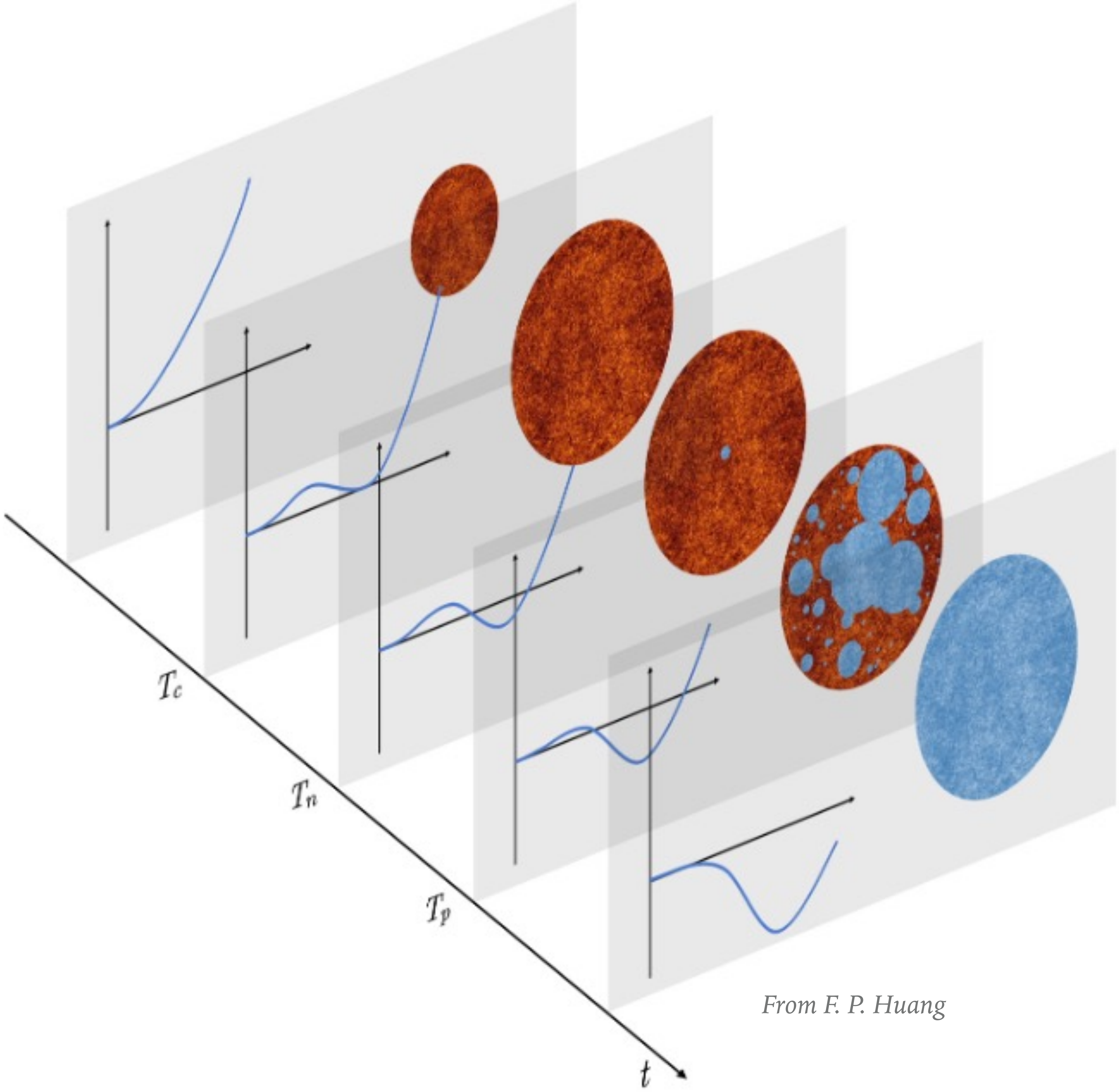
$$P(T_{\rm per}) = \exp\left[-\frac{64\pi}{3}\xi^4 \int_{T_{\rm per}}^{T_{\rm tra}} {\rm d}T' \frac{\Gamma(T')}{T'^6}\left(\frac{1}{T_{\rm per}} - \frac{1}{T'}\right)^3\right] = 70\,\%$$

- Inverse duration of the phase transition

$$\beta(T) = \frac{d}{dt}\left[\frac{S_E(T)}{T}\right] = TH(T)\frac{d}{dT}\left[\frac{S_E(T)}{T}\right]$$

- Ratio of latent heat to radiation density

$$\alpha = \frac{D\theta}{\pi^2 g_* T_*^4/30} = \frac{1}{\pi^2 g_* T_*^4/30}\left(V(\phi) - \frac{T}{4}\frac{\partial V(\phi, T)}{\partial T}\right)\Bigg|_{\phi_t}^{\phi_f}$$



From F. P. Huang

Bounce action

- Bounce action

$$S_E = \mathcal{S}_{d-1} \int_0^\infty \rho^{d-1} \left(\frac{1}{2}\dot{\phi}^2 + V(\phi) \right)$$

where ϕ satisfies

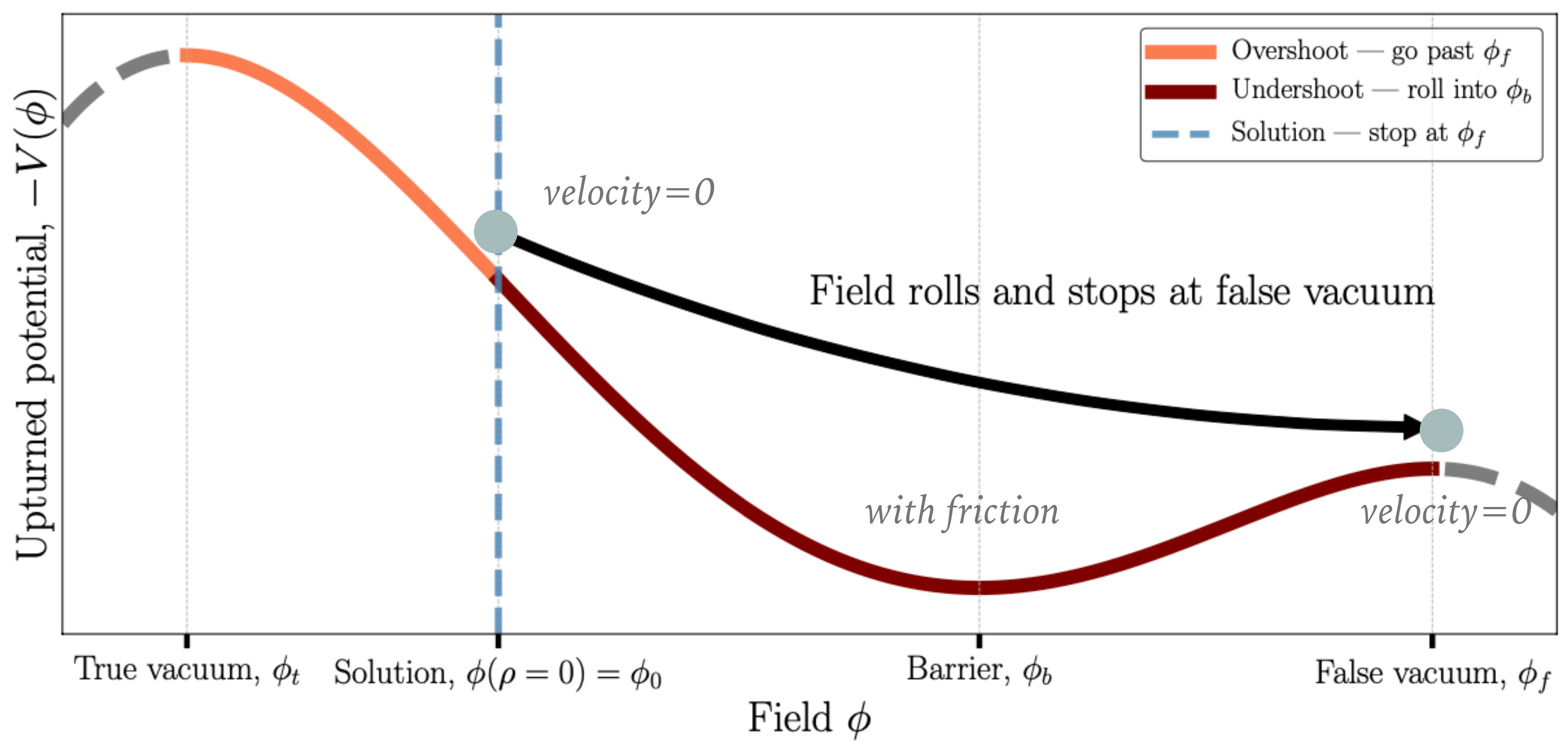
$$\frac{d^2\phi(\rho)}{d\rho^2} + \frac{\alpha}{\rho}\frac{d\phi(\rho)}{d\rho} = \Delta V(\phi)$$

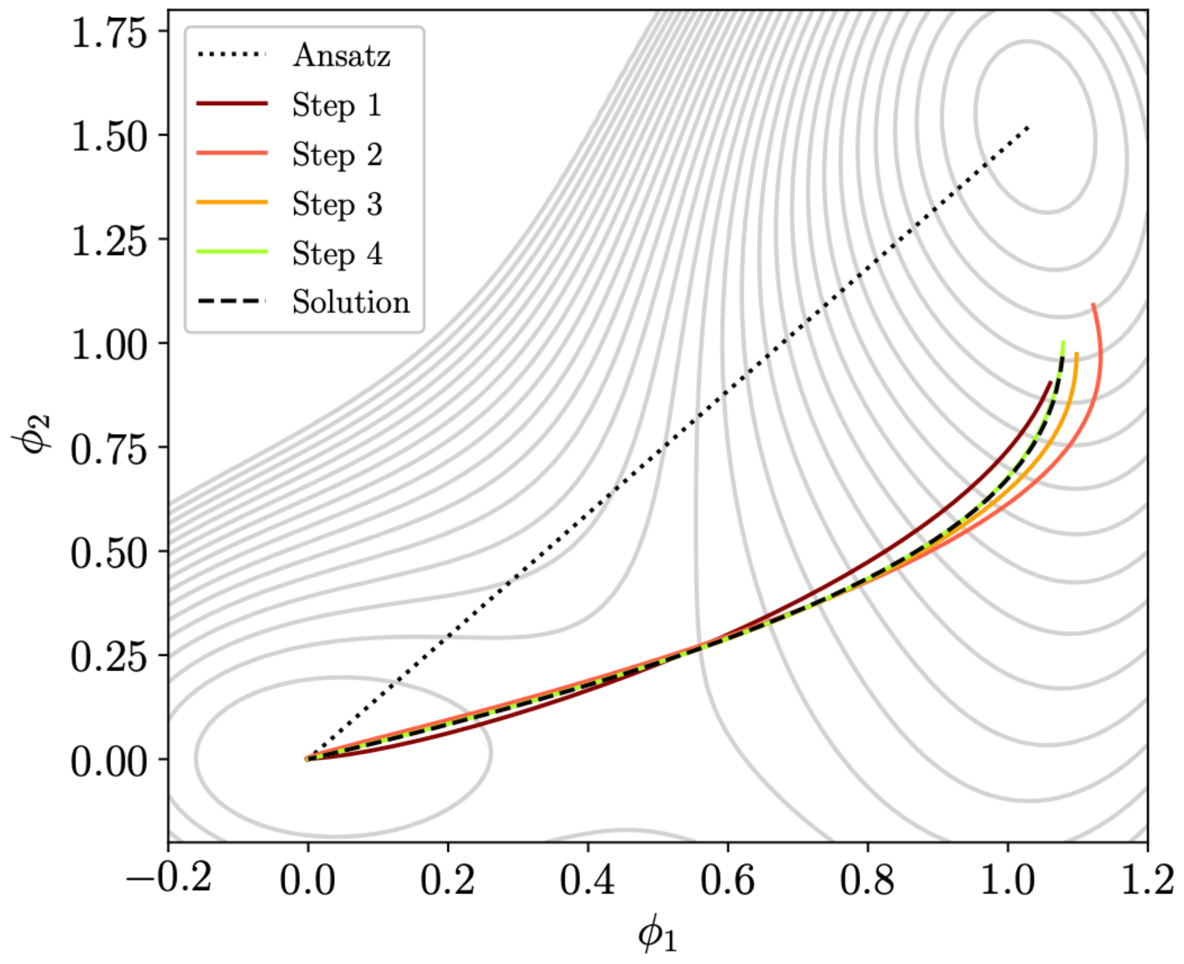
with boundary conditions

$$\left.\frac{d\phi(\rho)}{d\rho}\right|_{\rho=0} = 0\,, \quad \left.\frac{d\phi(\rho)}{d\rho}\right|_{\rho=\infty} = 0$$

$$\phi(\rho \to \infty) = \phi_f$$

The under/over-shooting method





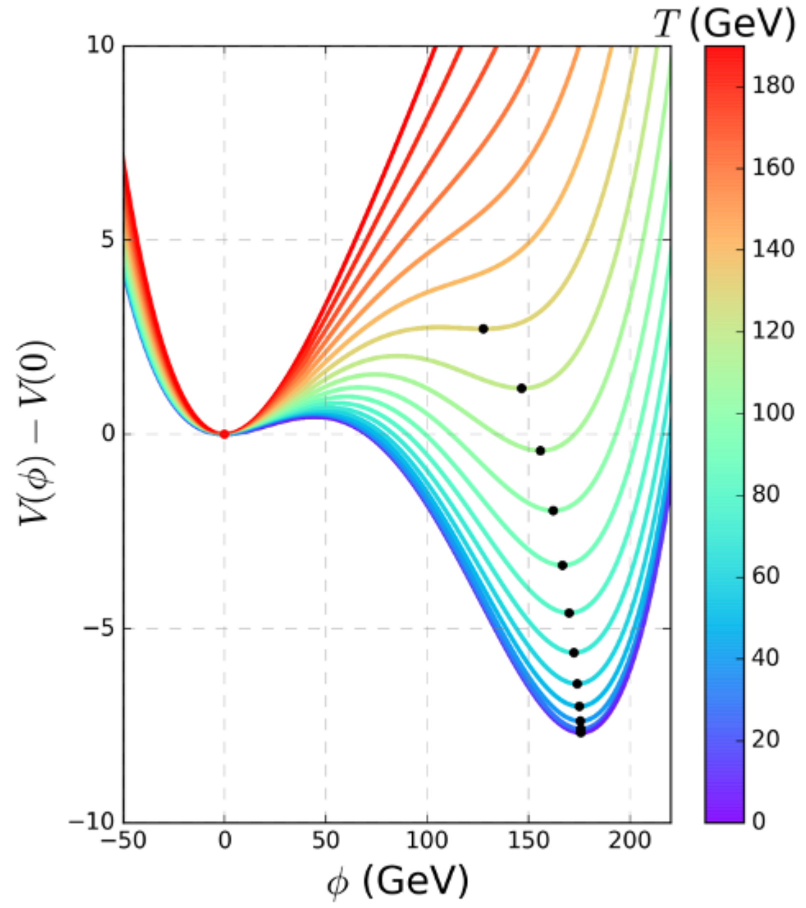
From Bubbleprofiler

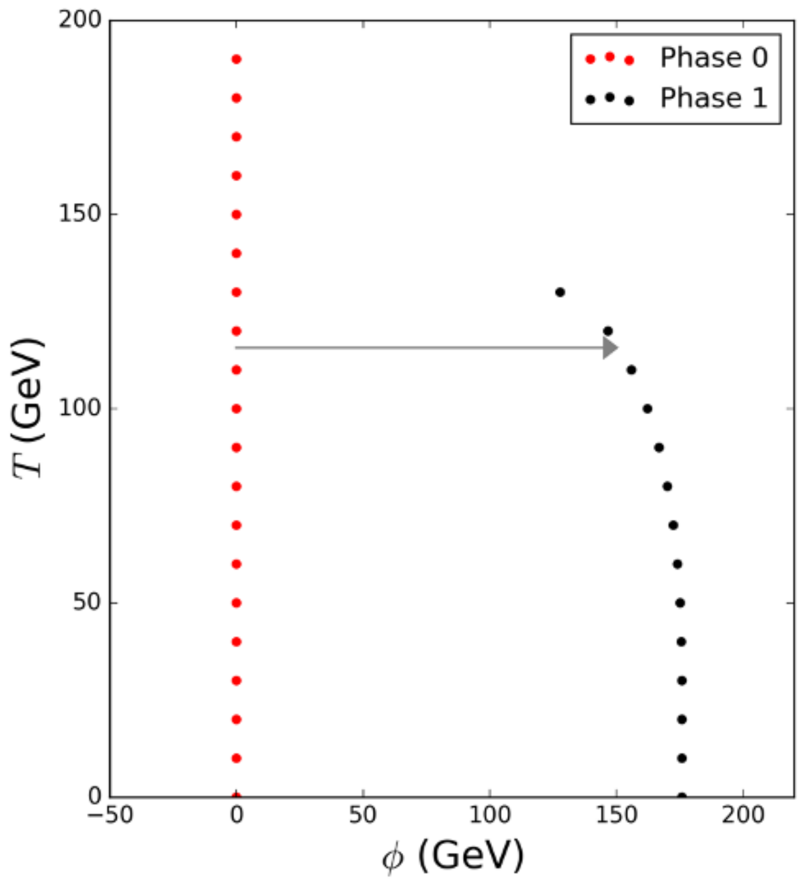
	Action			Time (s)		
# fields	BP	CT	AB	BP	CT	AB
1	54.1	52.6	52.4	0.051	0.066	1.285
2	20.8	21.1	20.8	0.479	0.352	7.473
3	22.0	22.0	22.0	0.964	0.215	25.209
4	55.9	56.4	55.9	1.378	0.255	54.258
5	16.3	16.3	16.3	2.958	0.367	305.531
6	24.5	24.5	24.4	4.853	0.337	830.449
7	36.7	36.6	36.7	6.754	0.375	1430.892
8	46.0	46.0	46.0	10.014	0.409	1805.713

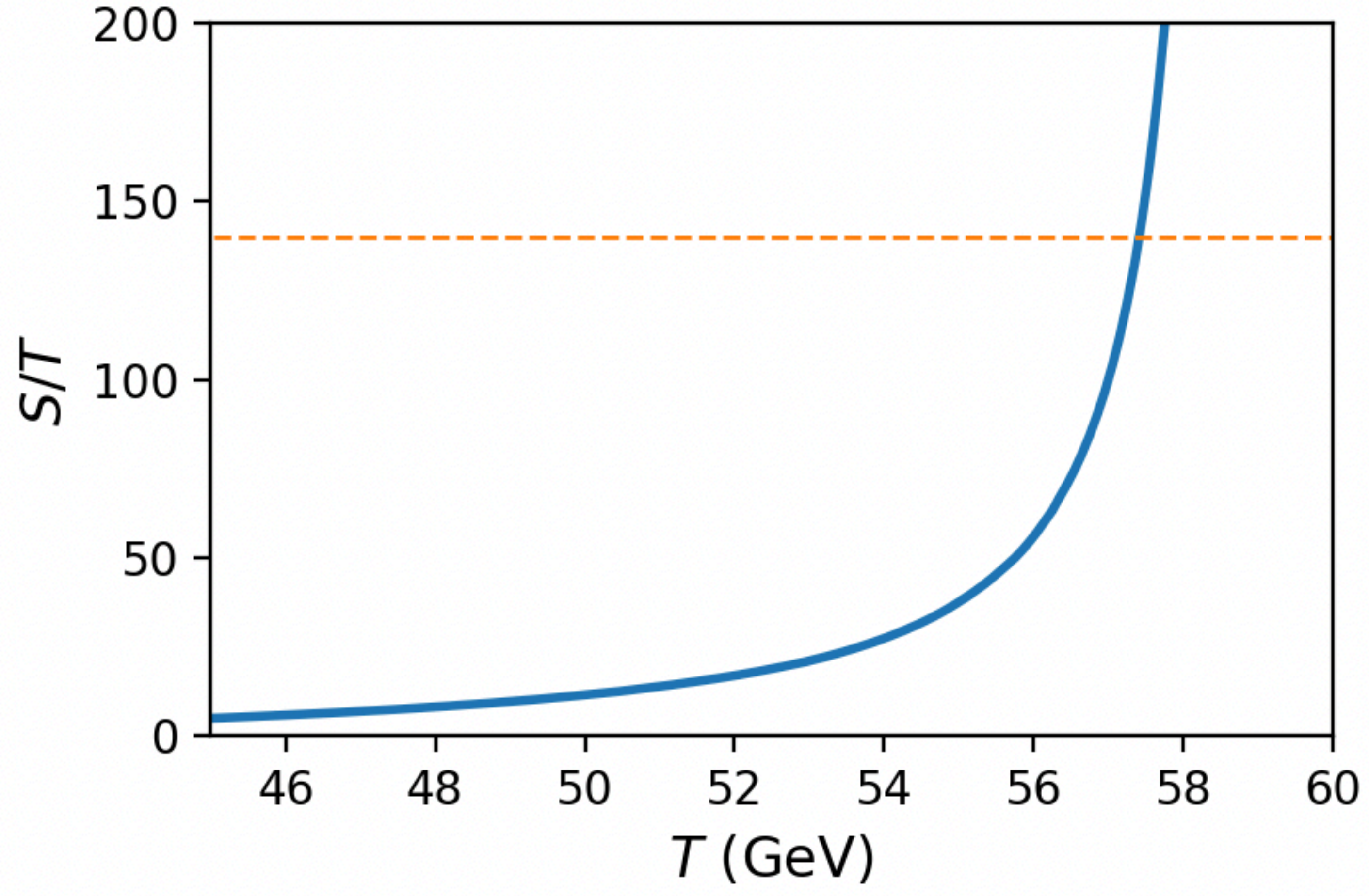
Bounce action

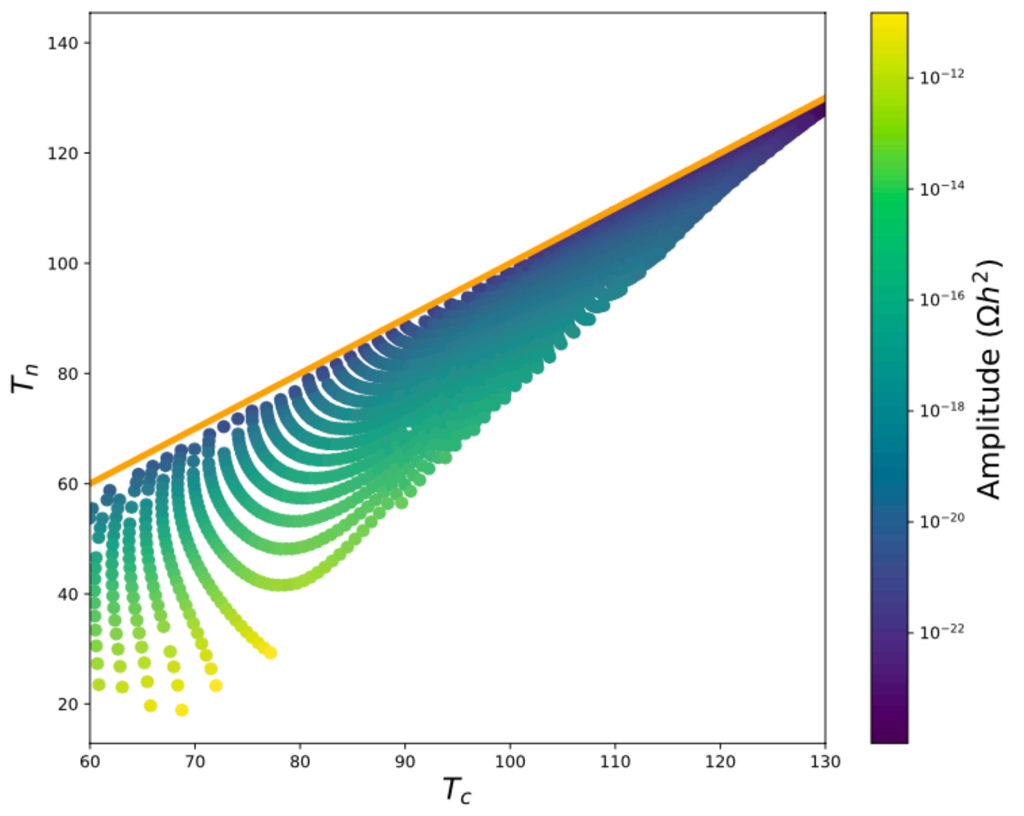
➤ Tools for calculating bounce action

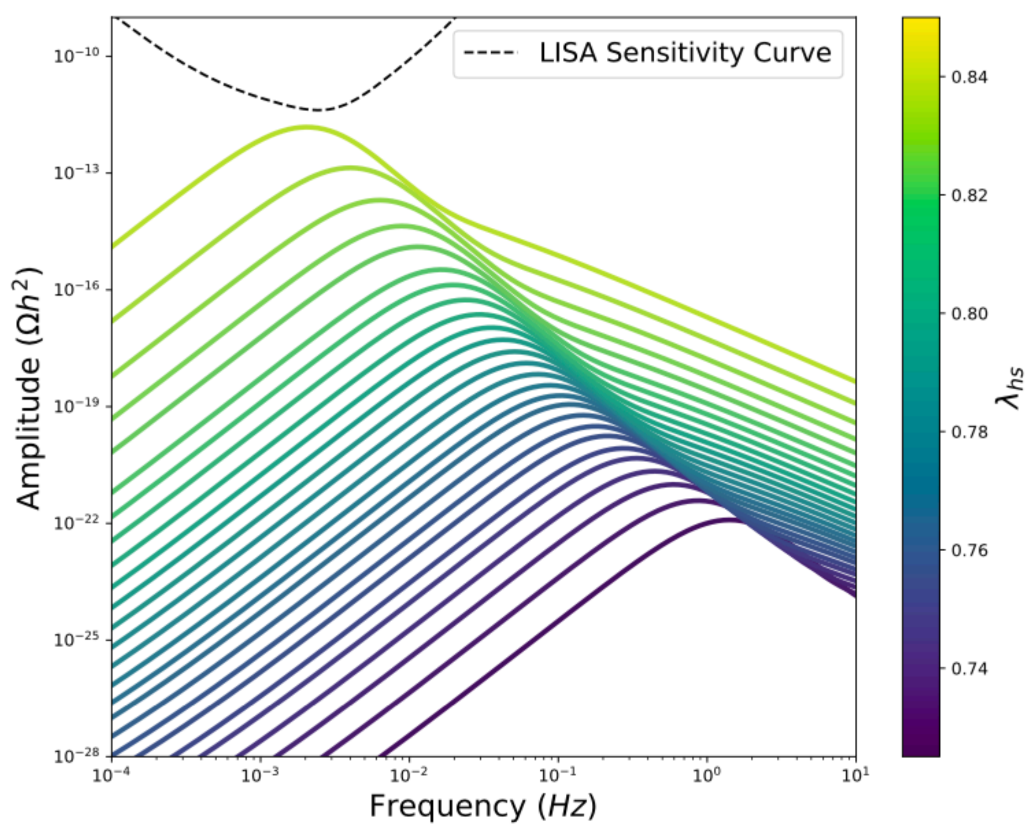
- CosmoTransitions
- PhaseTracer2
- BSMPT3
- PT2GWFinder
- AnyBubble
- BubbleProfiler
- FindBounce
- SimpleBounce
- … …

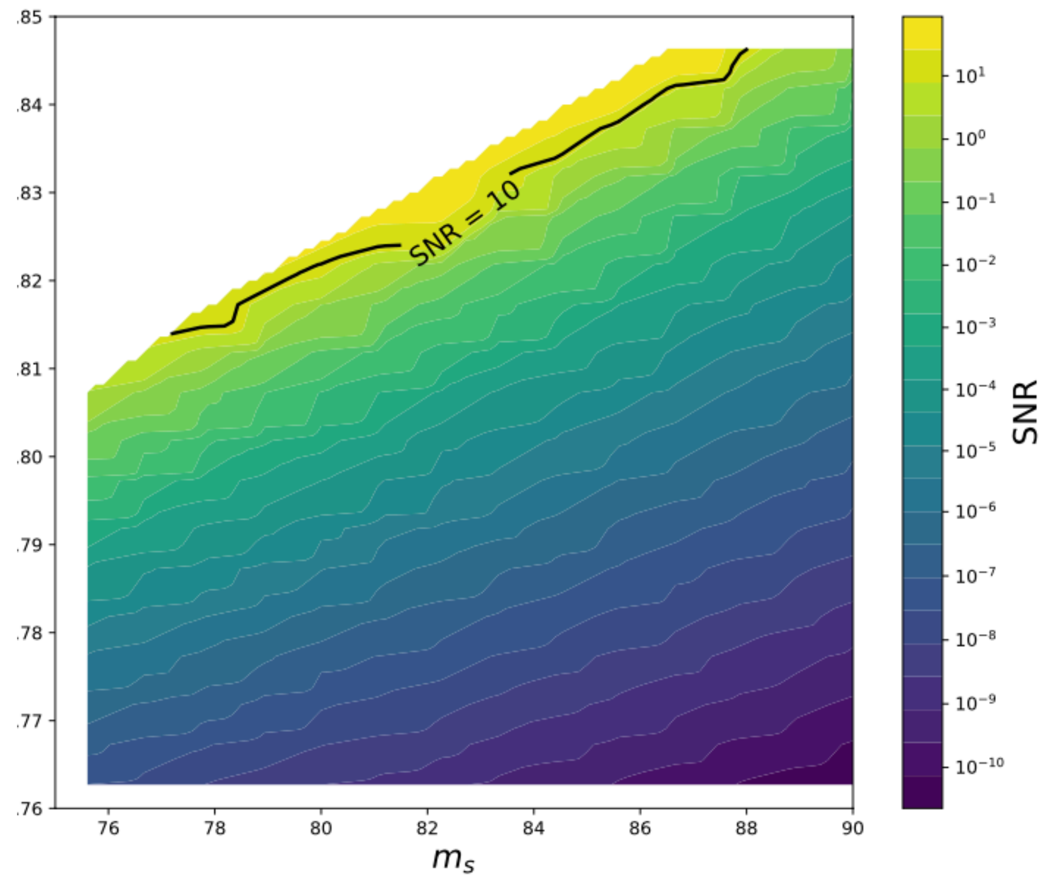




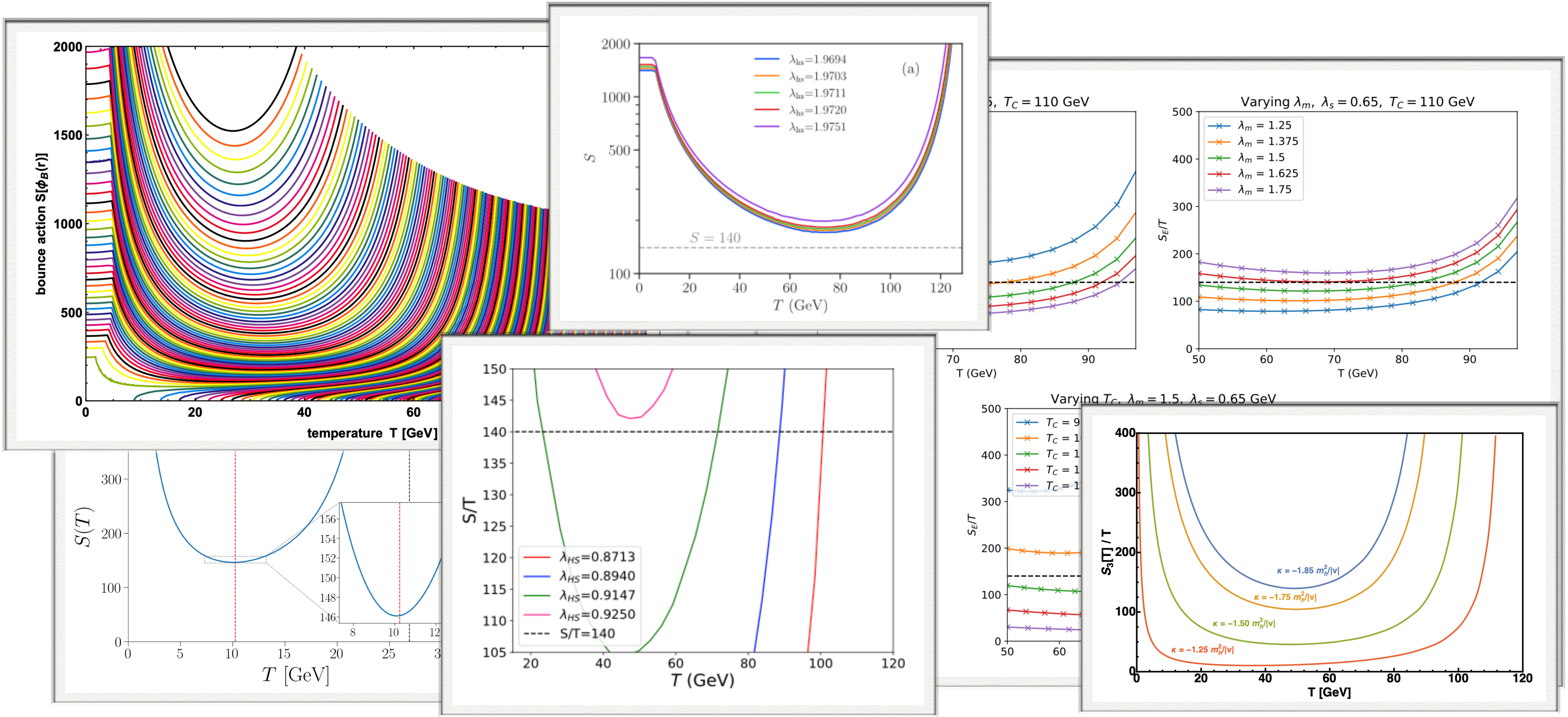








Action curve

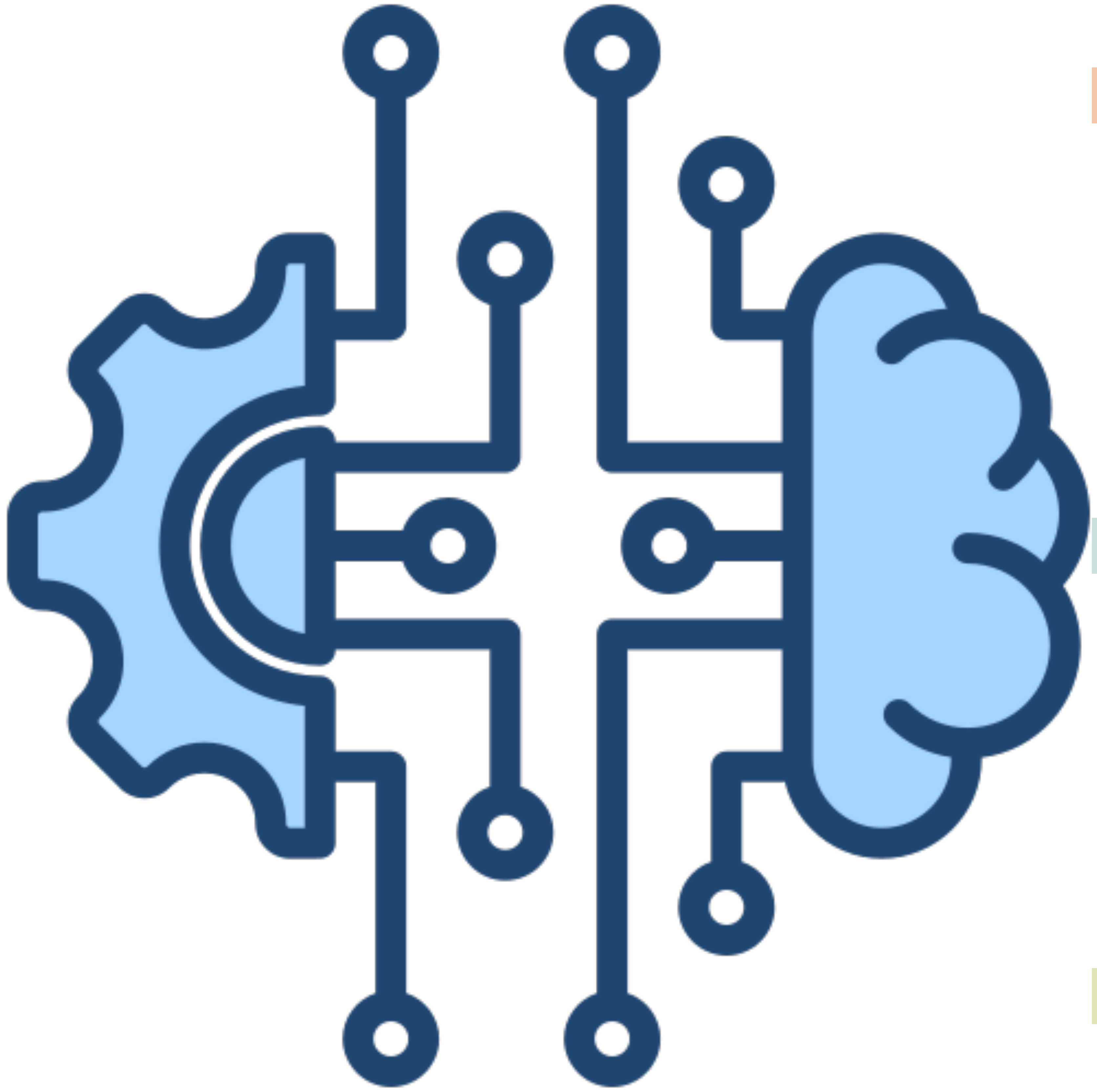


Machine Learning in Phase Transition

Model Parameters

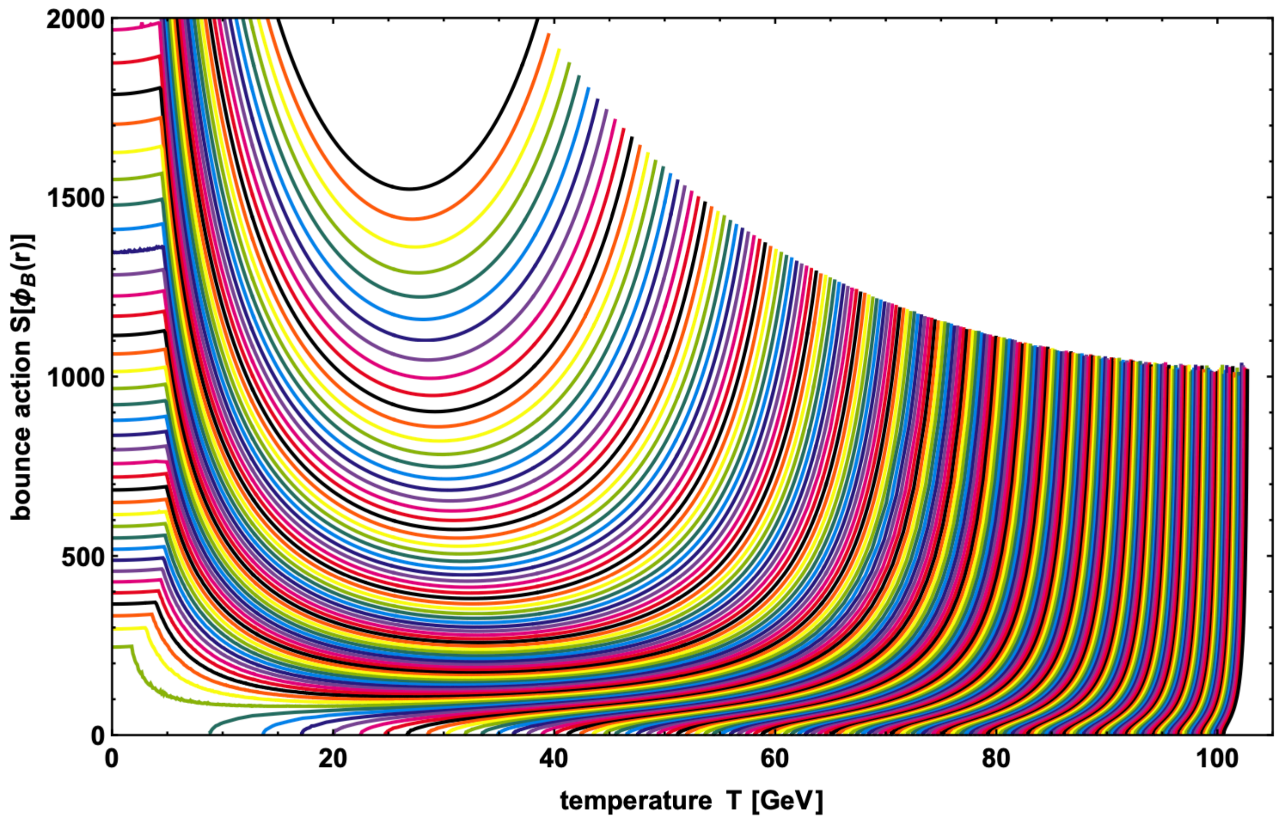
Potential at a fixed T

Model Parameters



GW

Action



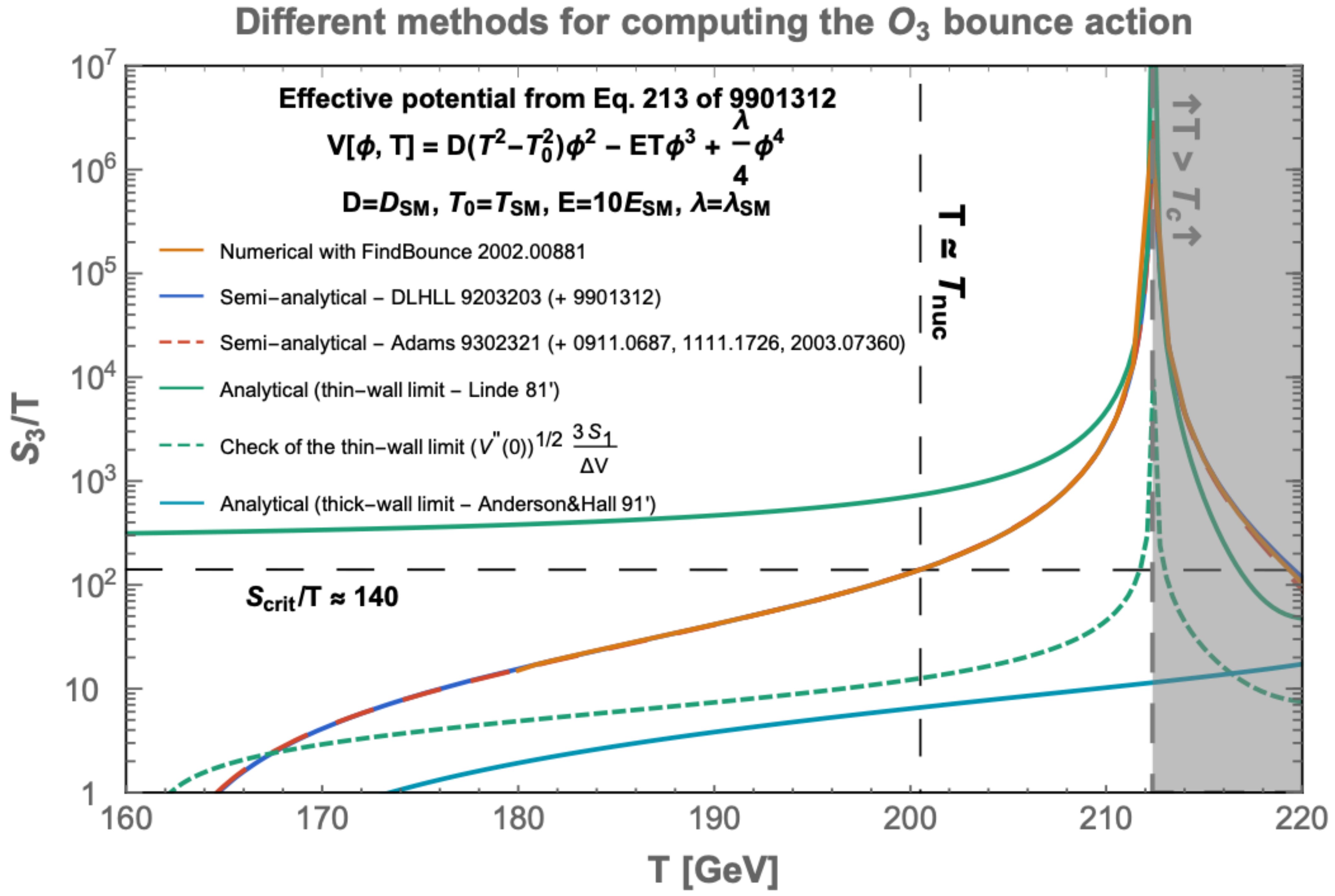
Action curve

Beyond the Standard Model Cocktail, Yann Gouttenoire, arXiv:2207.01633

- For a polynomial potential of the type

$$V(\phi) = a\phi^2 - b\phi^3 + \frac{\lambda}{4}\phi^4,$$

- Semi-analytical approximations of the bounce action:



$$S_3 = \frac{8\pi b}{\lambda^{3/2}} \frac{8\sqrt{\delta}}{81(2-\delta)^2}(\beta_1\delta + \beta_2\delta^2 + \beta_3\delta^3),$$

$$\delta \equiv \frac{2\lambda a}{b^2},$$

$$\beta_1 = 8.2938, \quad \beta_2 = -5.5330, \quad \beta_3 = 0.8180,$$

$$\alpha_1 = 13.832, \quad \alpha_2 = -10.819, \quad \alpha_3 = 2.0765.$$

$$S_3 = \frac{13.72\,a^{3/2}}{b^2} f(\delta/2),$$

$$f(x) = 1 + \frac{x}{4}\left[1 + \frac{2.4}{1-x} + \frac{0.26}{(1-x)^2}\right]$$

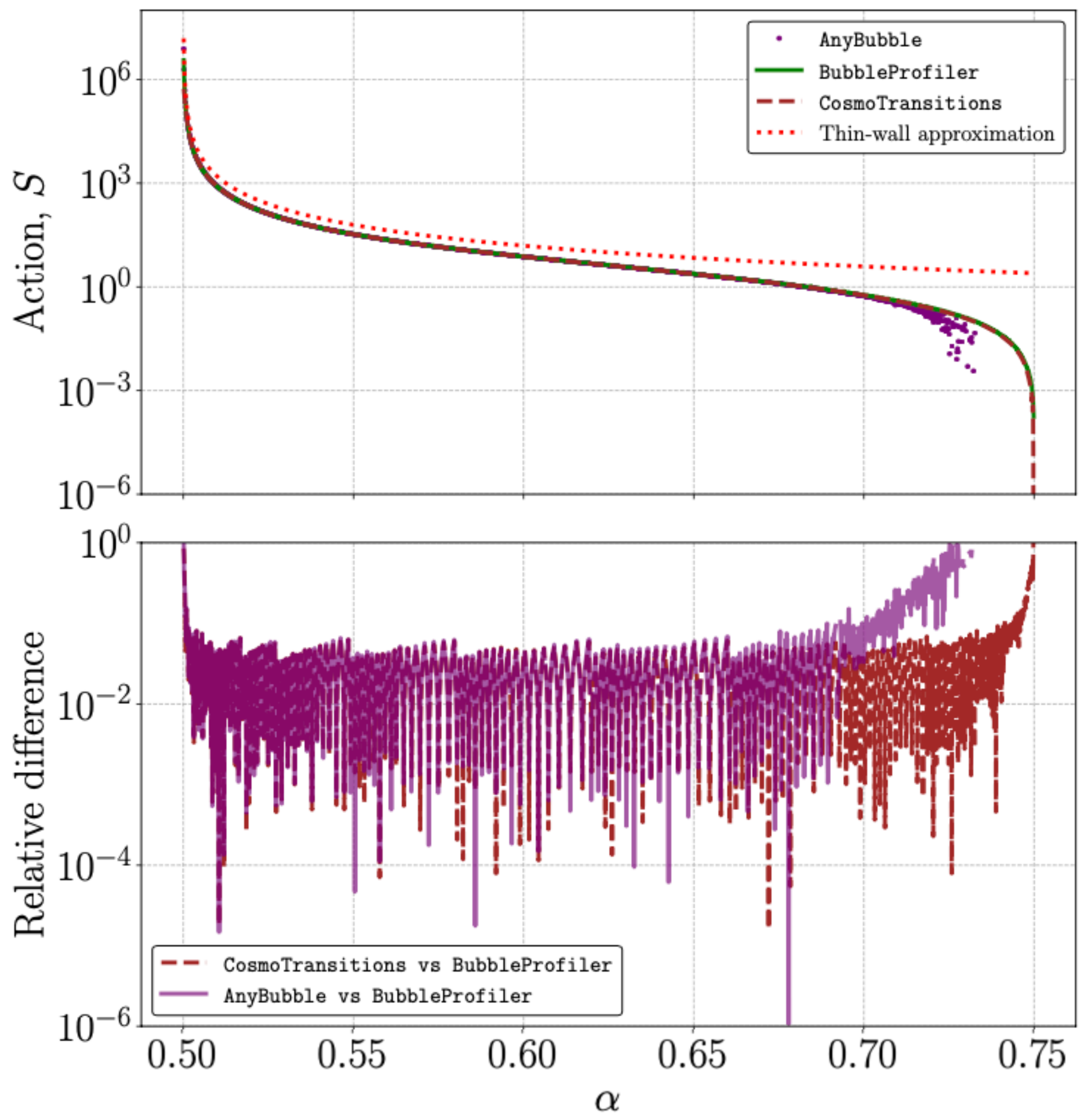
arXiv:2404.17632, Marco Matteini, Miha Nemevšek, Yutaro Shoji, Lorenzo Ubaldi

$$V = \frac{1}{2}m^2\phi^2 + \eta\phi^3 + \frac{\lambda}{8}\phi^4, \qquad \varepsilon_\alpha \equiv 1 - \lambda\frac{m^2}{4\eta^2},$$

$$S_3 = \frac{32\pi}{81}\frac{m^3}{4\eta^2}\frac{1}{\varepsilon_\alpha^2}\left(1 + \frac{17}{2}\varepsilon_\alpha + \left(\frac{247}{8} - \frac{9\pi^2}{4}\right)\varepsilon_\alpha^2\right)$$

Action curve

- In general cases, especially for high-dimensional scenarios, the action curve has no analytical expression.
- Meanwhile, there are two main issues with numerical calculation of the action:
  - unavoidable numerical errors
  - excessive computational time
- They will lead to
  - large error on the β parameter
  - extremely slow to get the T_P



From Bubbleprofiler

$$V(\chi) = \frac{-4\alpha + 3}{2}\chi^2 - \chi^3 + \alpha\chi^4$$

Transition parameters

- Bubble nucleation rate

$$\frac{\Gamma}{V} = A(T)e^{-S_E/T}[1 + \mathcal{O}(\hbar)]$$

- Nucleation temperature

$$N(T) = \int_{T_{\rm nuc}}^{T_{\rm tra}} \frac{dT}{T}\frac{\Gamma(T)}{H(T)^4} = 1$$

- Percolation temperature

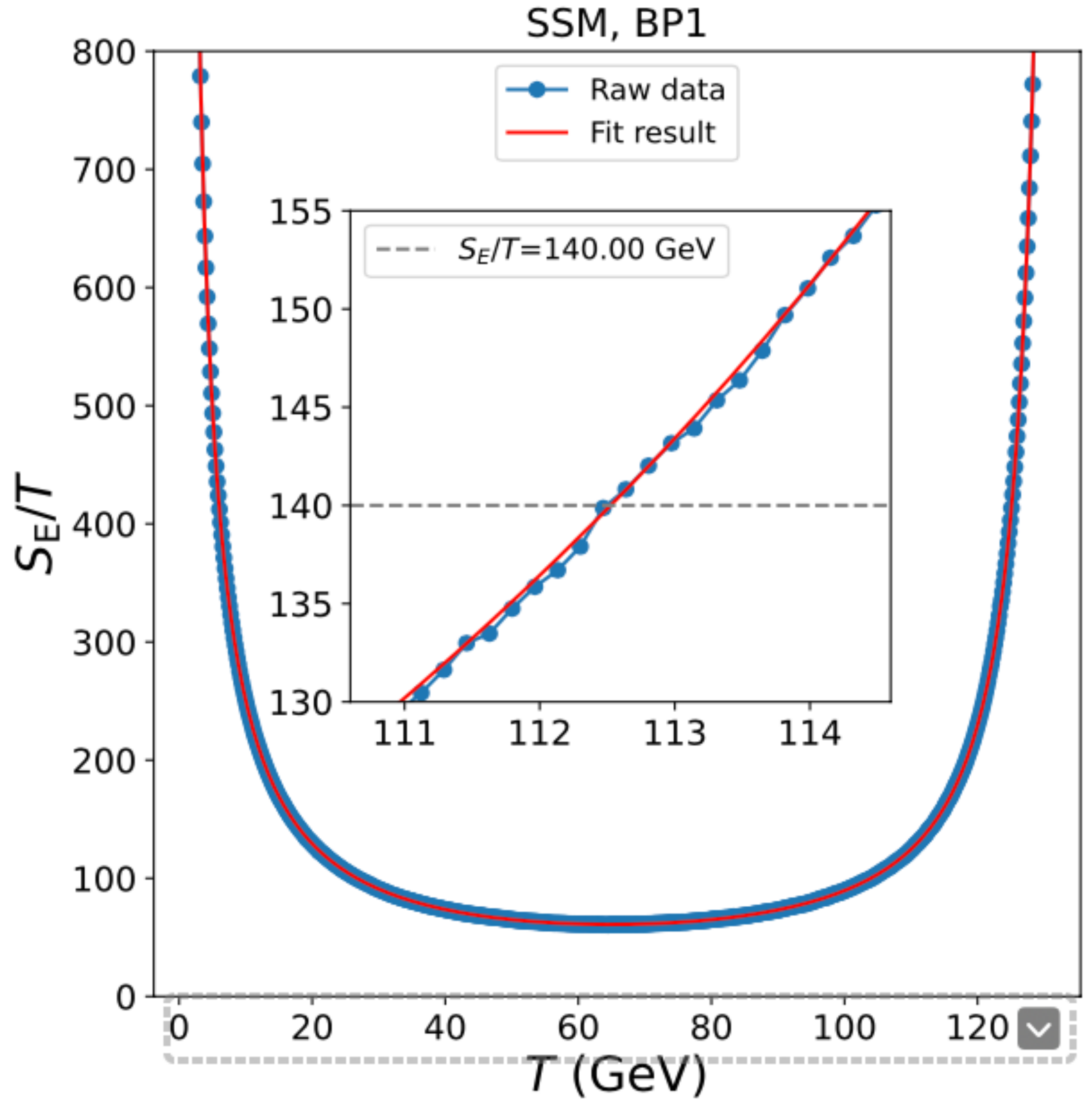
$$P(T_{\rm per}) = \exp\left[-\frac{64\pi}{3}\xi^4 \int_{T_{\rm per}}^{T_{\rm tra}} dT' \frac{\Gamma(T')}{T'^6}\left(\frac{1}{T_{\rm per}} - \frac{1}{T'}\right)^3\right] = 70\,\%$$

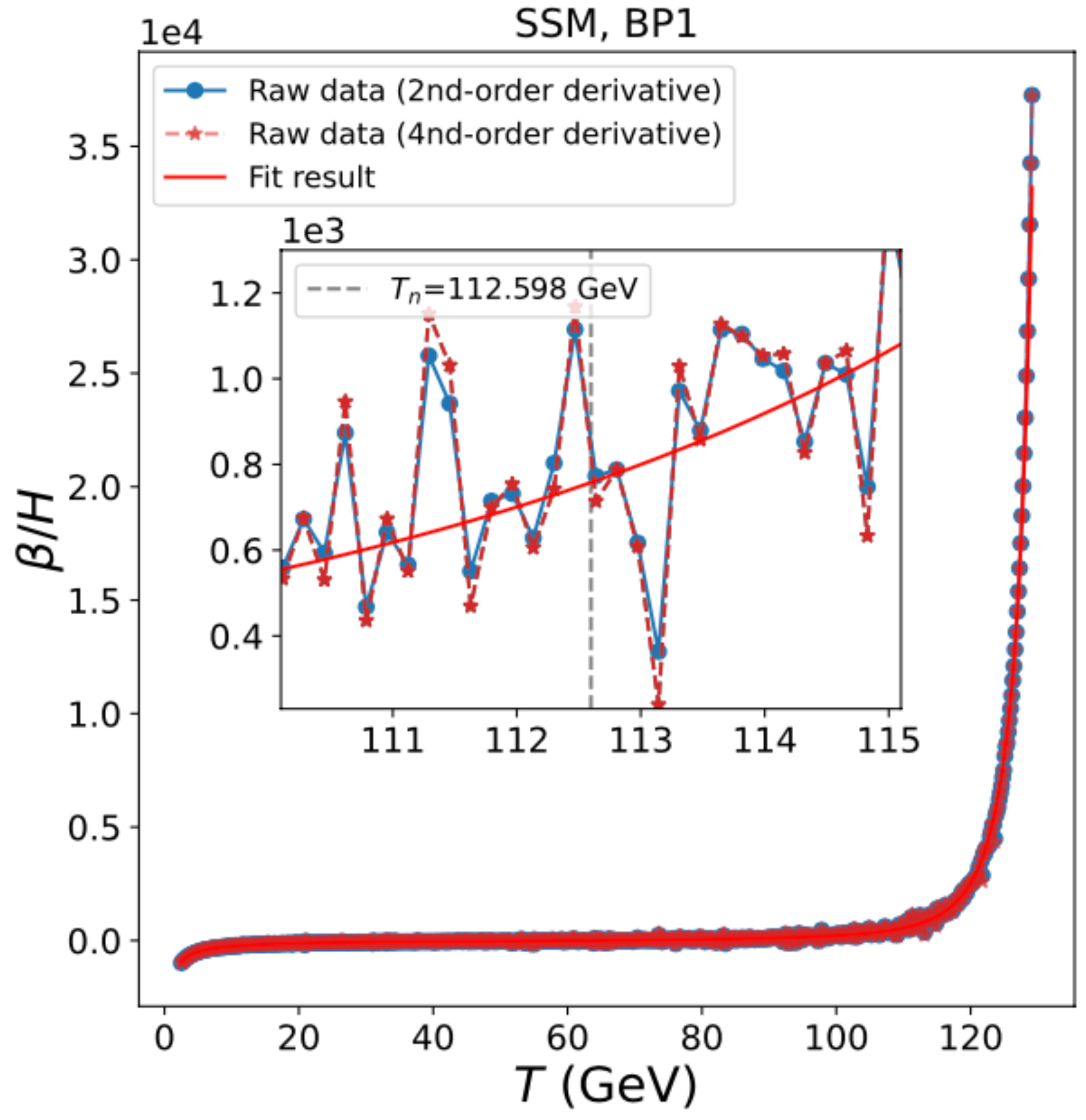
- Inverse duration of the phase transition

$$\beta(T) = \frac{d}{dt}\left(\frac{S_E}{T}\right) = TH(T)\frac{d}{dT}\left(\frac{S_E}{T}\right)$$

- Ratio of latent heat to radiation density

$$\alpha = \frac{D\theta}{\pi^2 g_* T_*^4/30} = \frac{1}{\pi^2 g_* T_*^4/30}\left(V(\phi) - \frac{T}{4}\frac{\partial V(\phi,T)}{\partial T}\right)\Bigg|_{\phi_t}^{\phi_f}$$



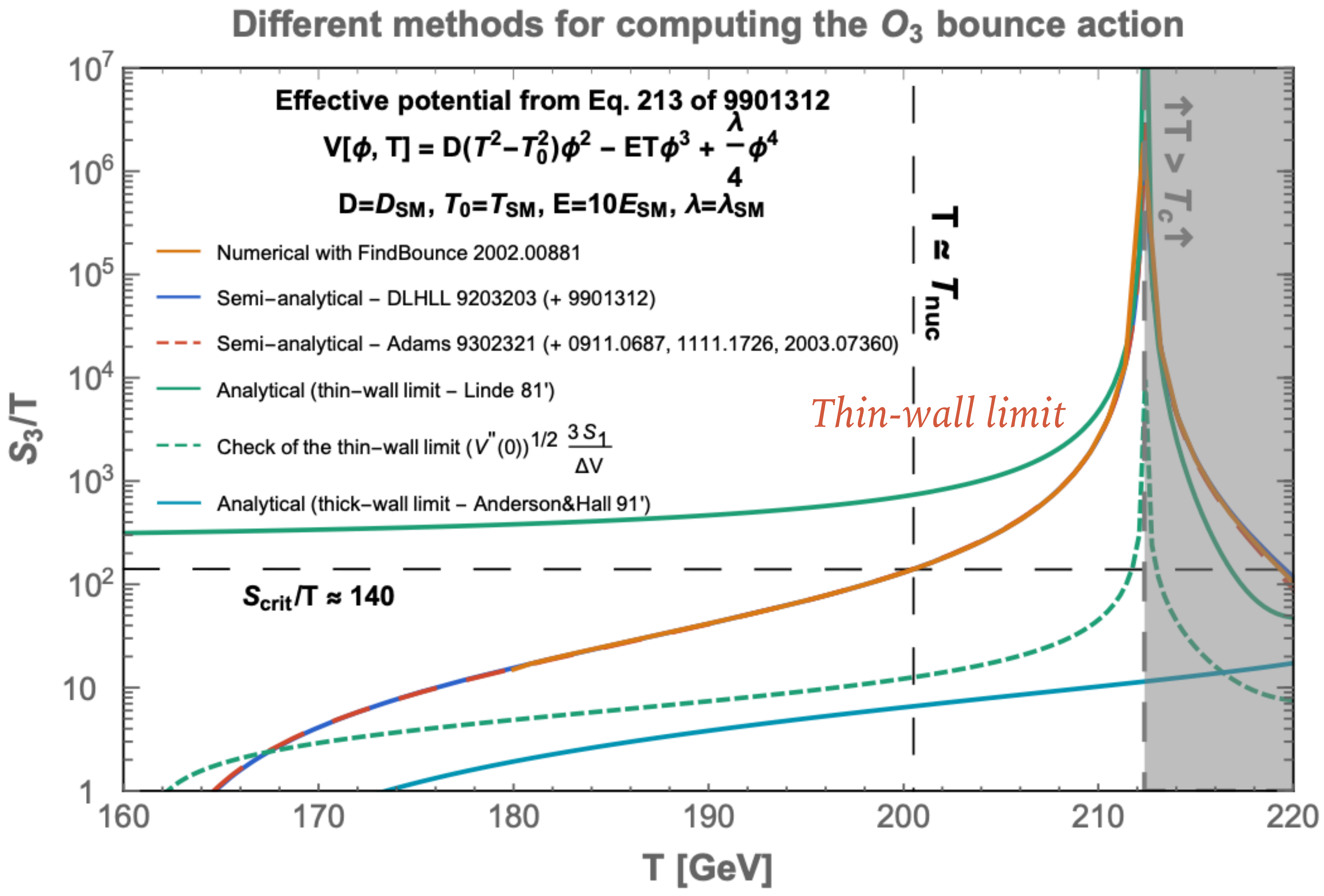


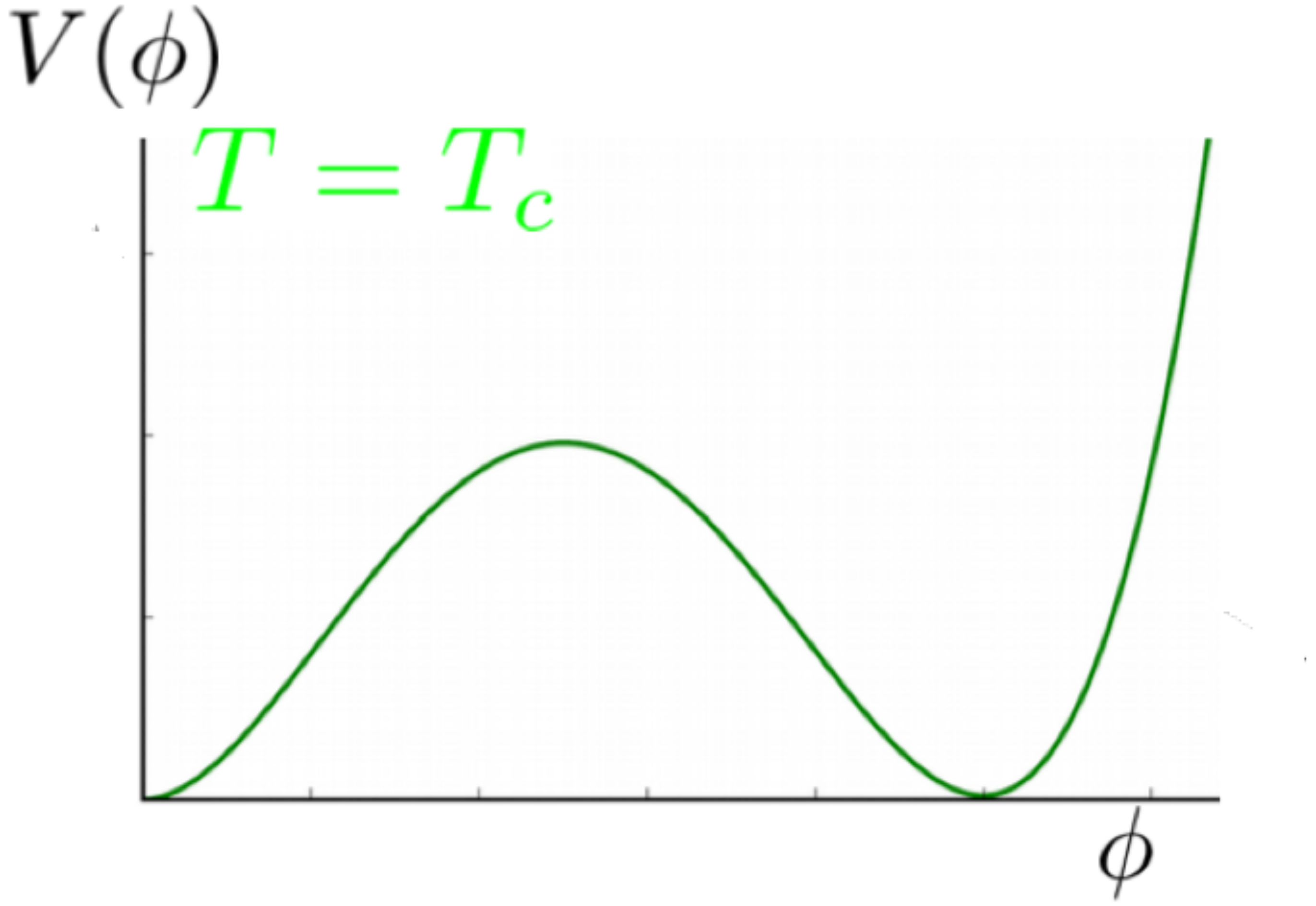
$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}, \quad (2\text{nd} - \text{order})$$

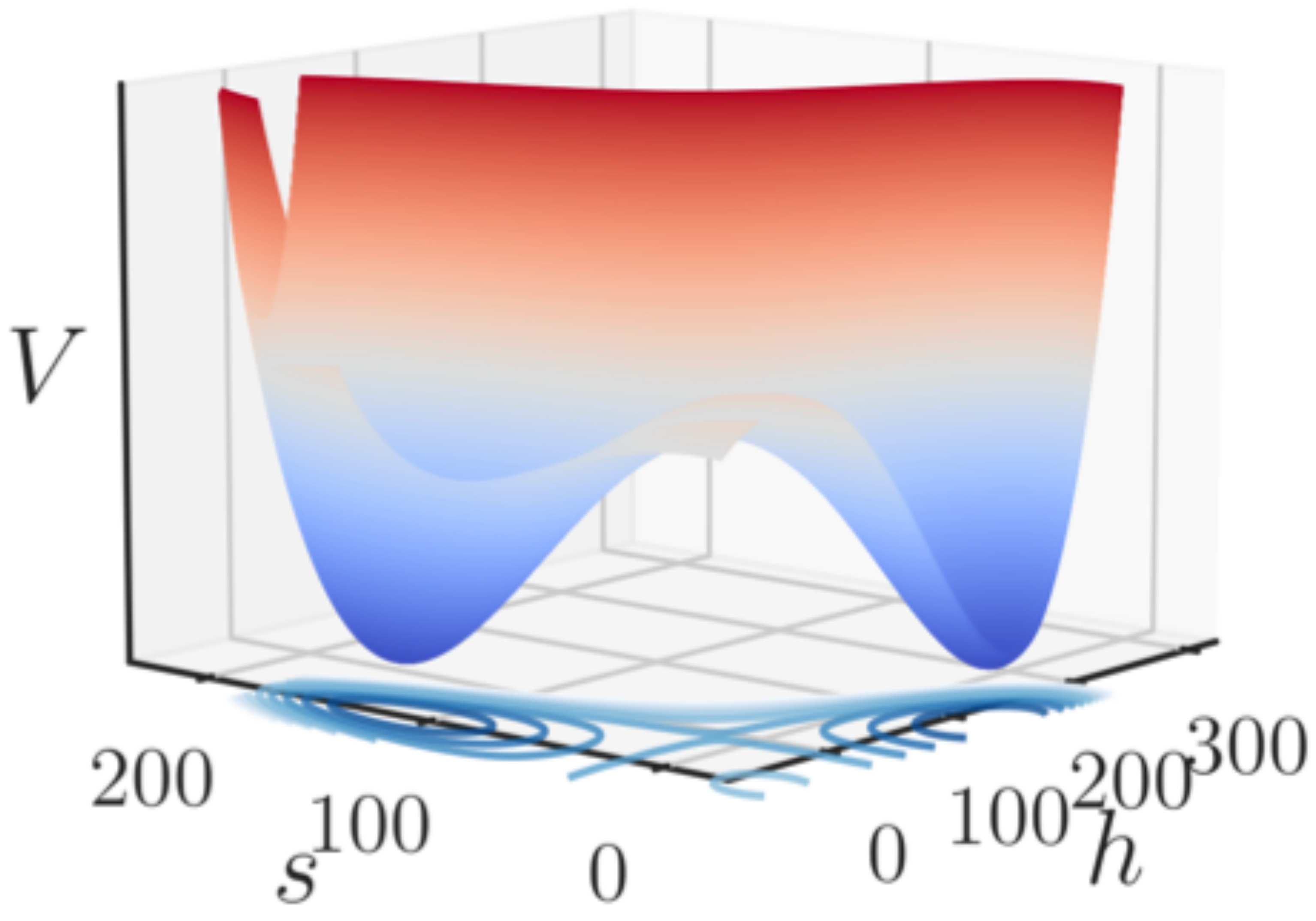
$$f'(x) \approx \frac{-f(x+2h) + 8f(x+h) - 8f(x-h) + f(x-2h)}{12h}, \quad (4\text{nd} - \text{order})$$

Action curve

➤ The action curve always diverges at $T = T_C$.







This divergent behavior is independent of the dimensionality of the potential

Action curve

➤ In the thin-wall approximation, the potential difference $\epsilon = V_{\rm eff}(\phi_f;T) - V_{\rm eff}(\phi_t;T)$ is much smaller than the height of the barrier, so we neglect the viscous damping term in the bounce equation:

$$\frac{\mathrm{d}^2\phi}{\mathrm{d}r^2} + \frac{2}{r}\frac{\mathrm{d}\phi}{\mathrm{d}r} = \frac{\partial V_{\rm eff}(\phi;T)}{\partial\phi} \quad\rightarrow\quad \frac{\mathrm{d}^2\phi}{\mathrm{d}r^2} = \frac{\partial V_{\rm eff}(\phi;T)}{\partial\phi} \quad\rightarrow\quad \frac{d\phi}{dr} = \sqrt{2V_{\rm eff}(\phi;T)}\,.$$

Define the surface tension of the bubble

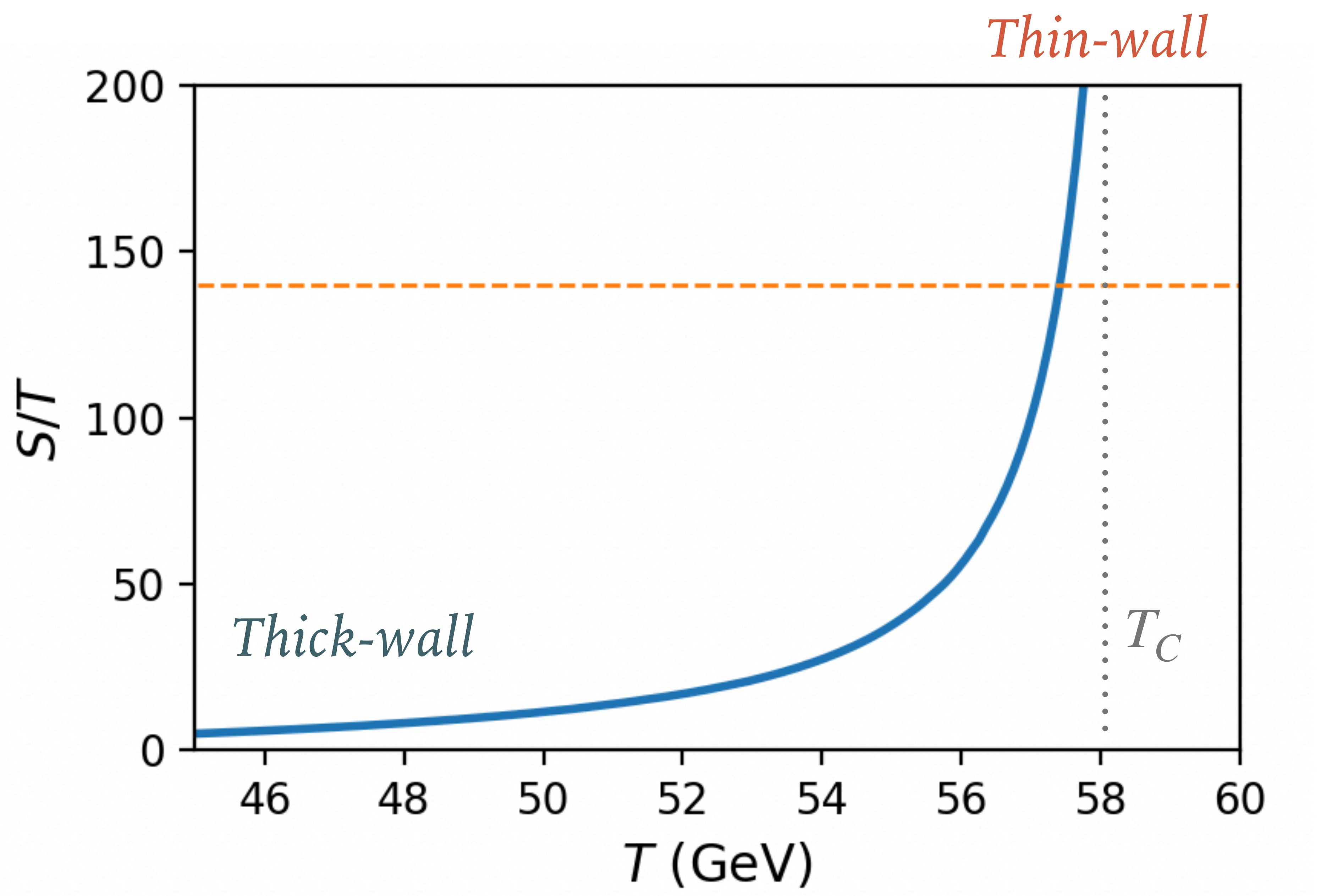
$$\sigma = \int_0^\infty dr\left[\frac{1}{2}\left(\frac{d\phi}{dr}\right)^2 + V_{\rm eff}(\phi;T)\right] = \int_{\phi_t}^{\phi_f} d\phi\sqrt{2V_{\rm eff}(\phi;T)},$$

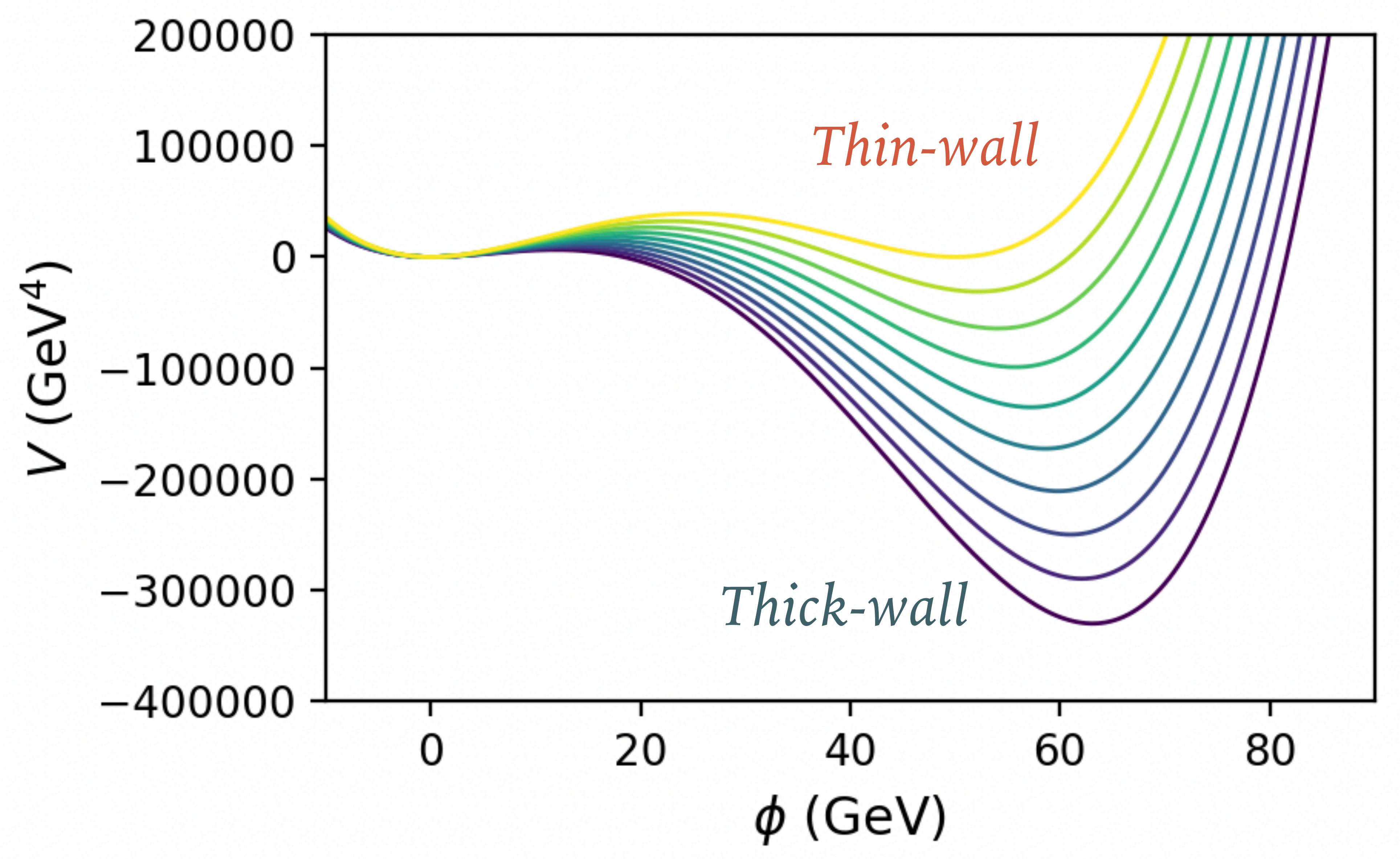
$$S_E = 4\pi\int_0^{+\infty} r^2\mathrm{d}r\,\left[\frac{1}{2}\left(\frac{\partial\phi}{\partial r}\right)^2 + V_{\rm eff}(\phi;T)\right] = 4\pi R^2\sigma - \frac{4}{3}\pi R^3\epsilon\,.$$

where R is the radius of the critical bubble and can be calculated by minimization of S_E, $R = \frac{2\sigma}{\epsilon}$. As

$$\epsilon = V_{\rm eff}(\phi_f;T) - V_{\rm eff}(\phi_t;T) = \left(\left.\frac{\partial V_{\rm eff}(\phi_f;T)}{\partial T}\right|_{T=T_c} - \left.\frac{\partial V_{\rm eff}(\phi_t;T)}{\partial T}\right|_{T=T_c}\right)(T-T_c)$$

$$S_E = \frac{16\pi\sigma^3}{3\epsilon^2} \propto \frac{1}{(T-T_C)^2}$$





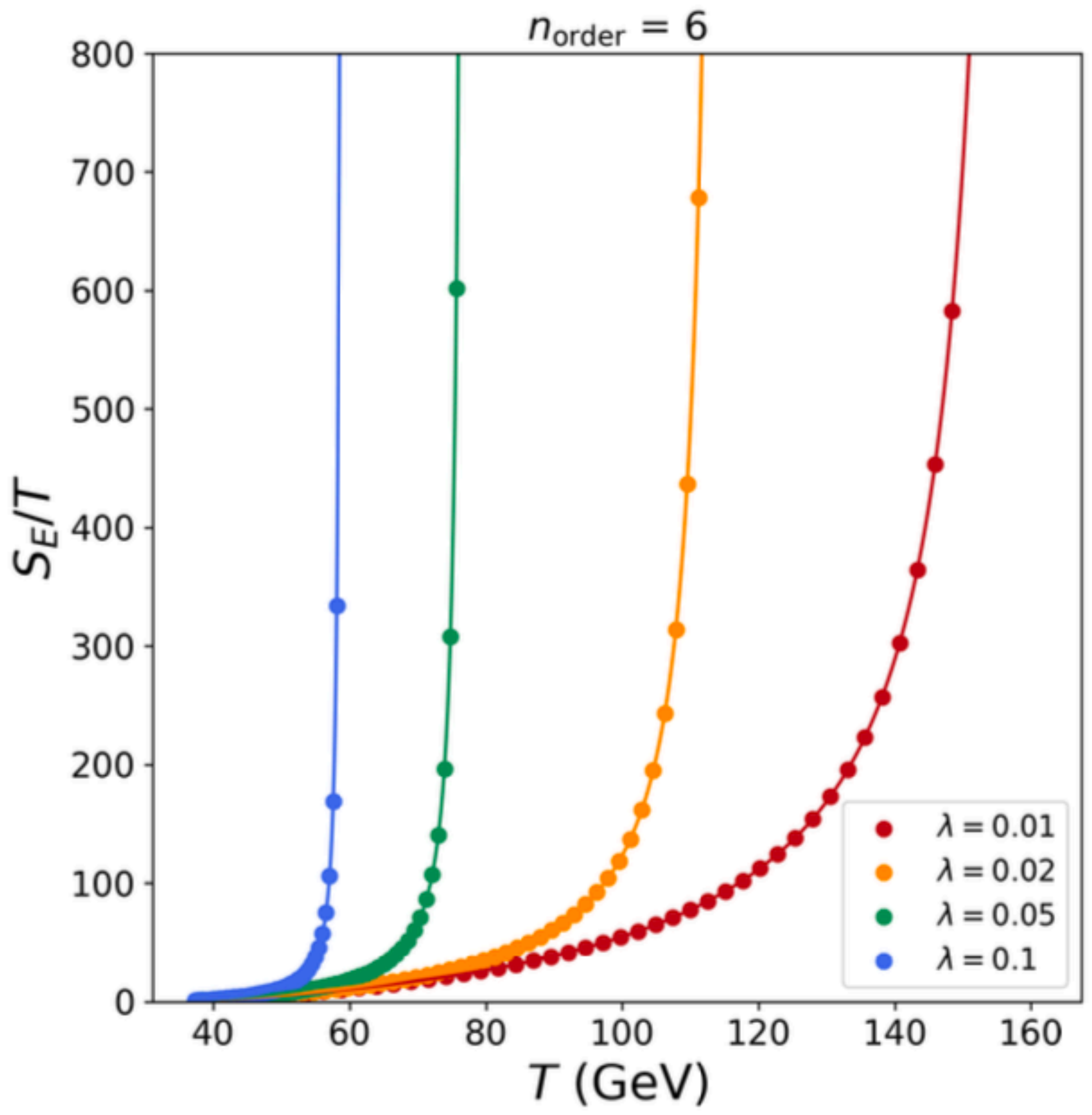
Action curve

➤ Therefore, it is reasonable to use the polynomial fitting formula

$$S_E = \frac{1}{(T - T_c)^2} \sum_{i=0}^{n_{\text{order}}} q_i T^i$$

➤ And one can get the expression for the inverse phase transition duration time

$$\frac{\beta}{H} = \frac{1}{T(T - T_c)^3} \left[\sum_{i=1}^{n_{\text{order}}} q_i i T^i (T - T_c) - \sum_{i=0}^{n_{\text{order}}} q_i T^i (3T - T_c) \right]$$

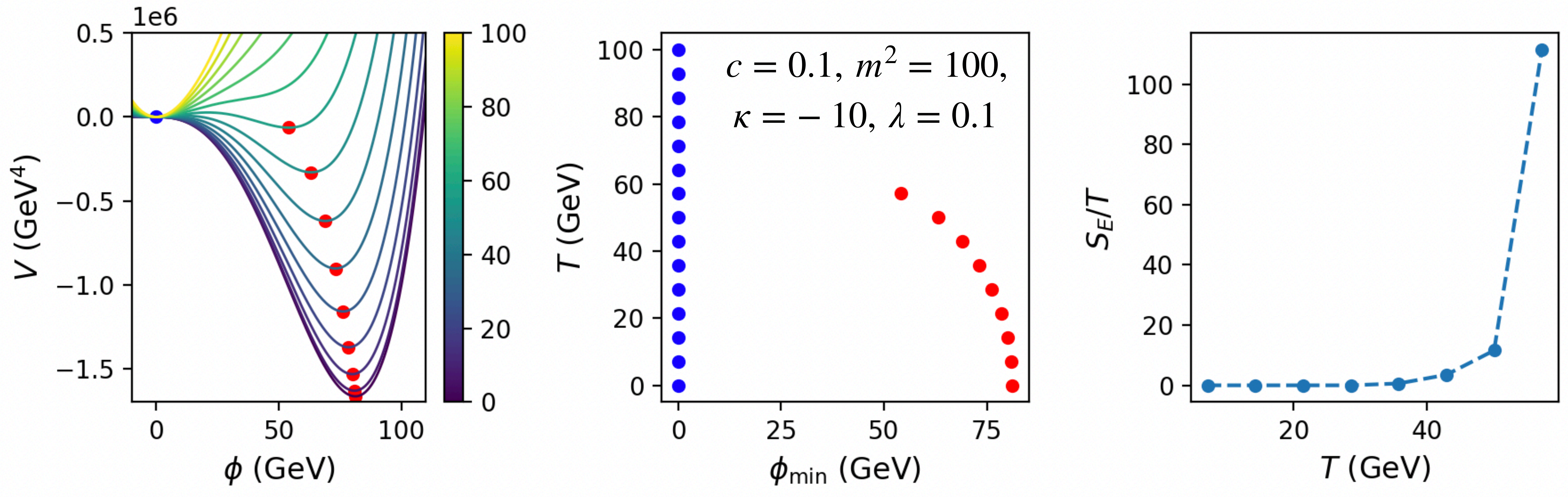


Action curve for one-dimensional toy model

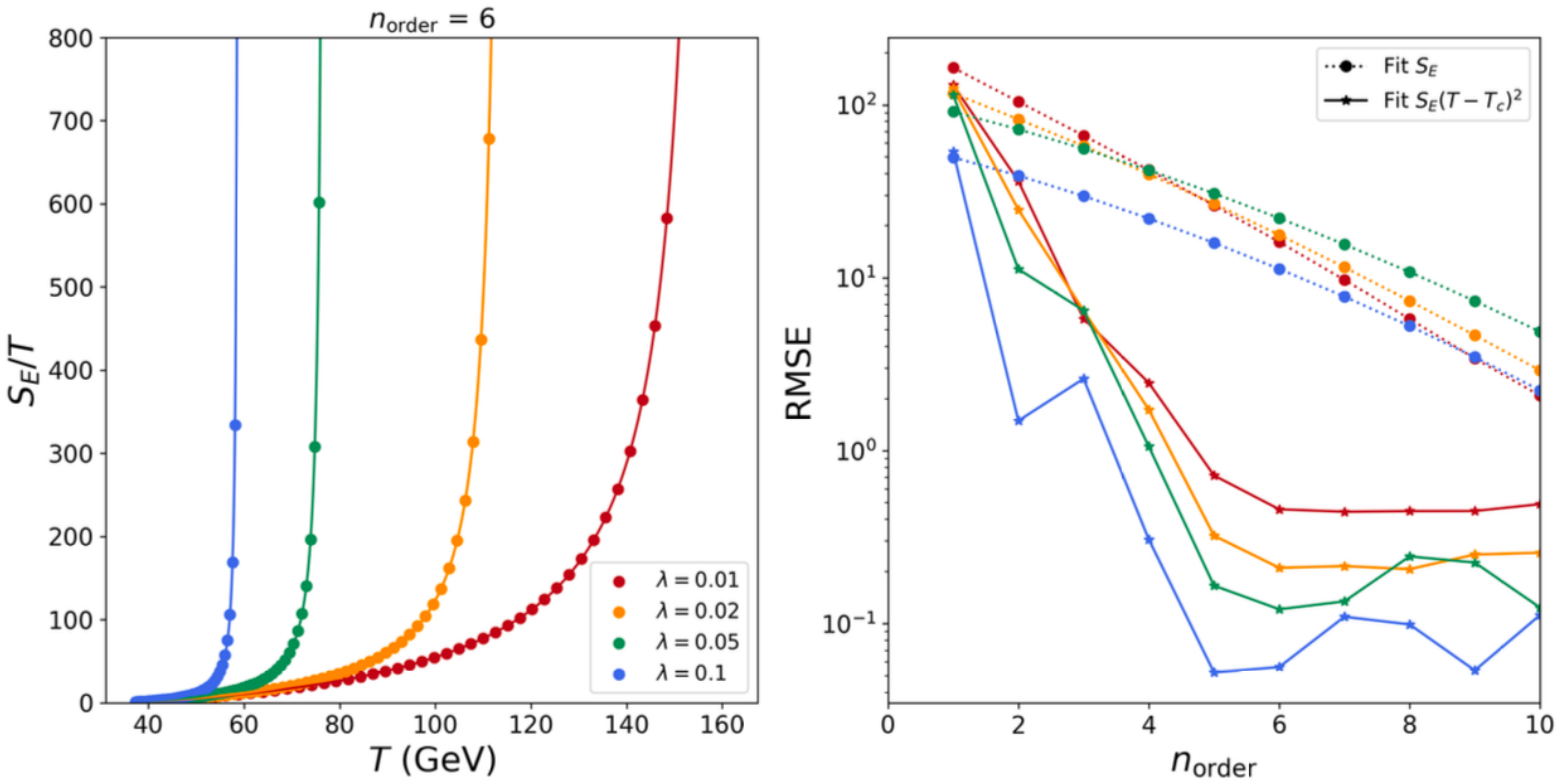
- To validate the polynomial fitting approach, we utilize a 1D toy model in which the action can be accurately computed.

$$V_{\text{eff}}(\phi;T) = (cT^2 - m^2)\phi^2 + \kappa\phi^3 + \lambda\phi^4$$

- It mimics a simple model that includes high-temperature corrections.



Action curve for one-dimensional toy model

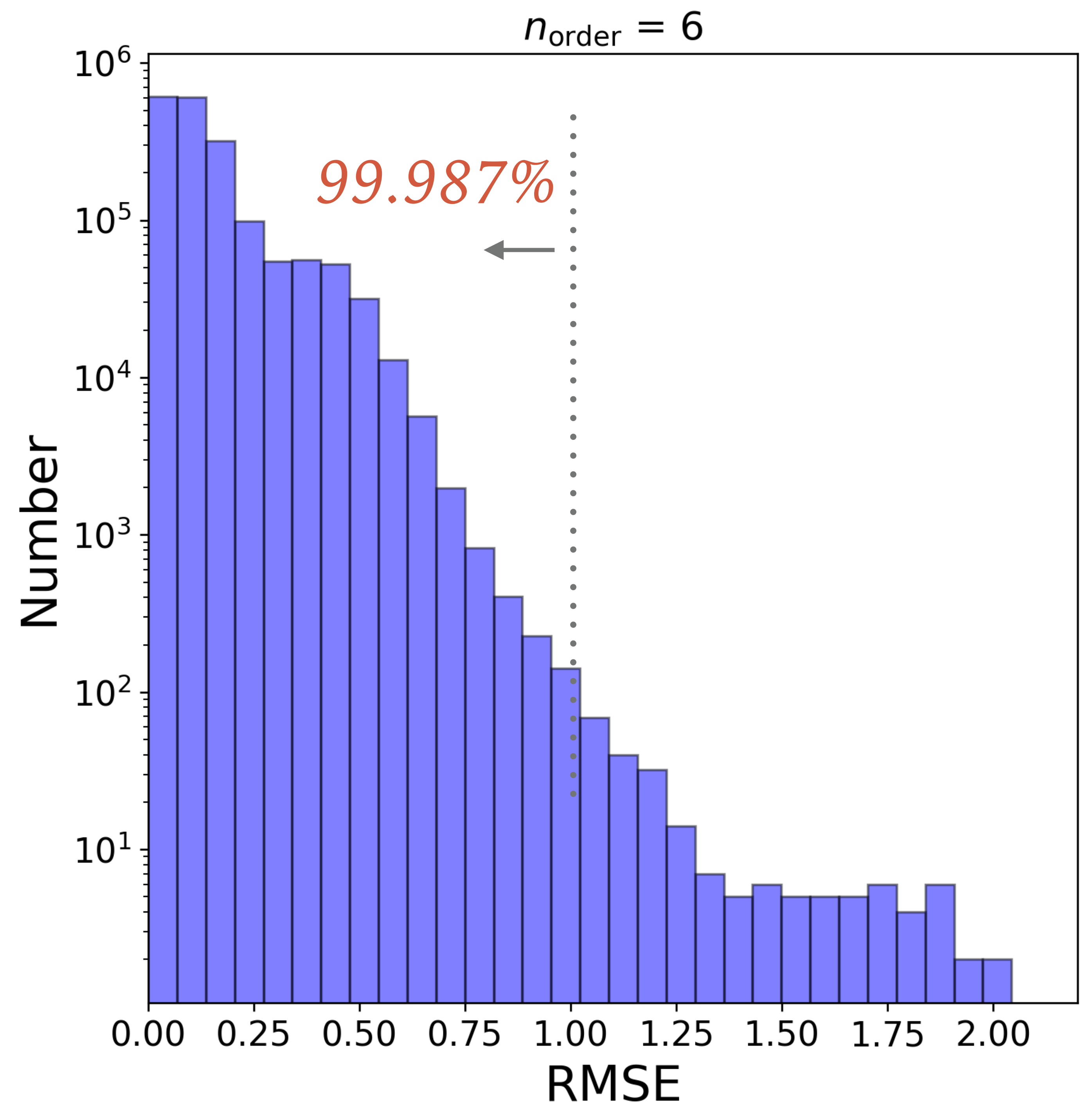


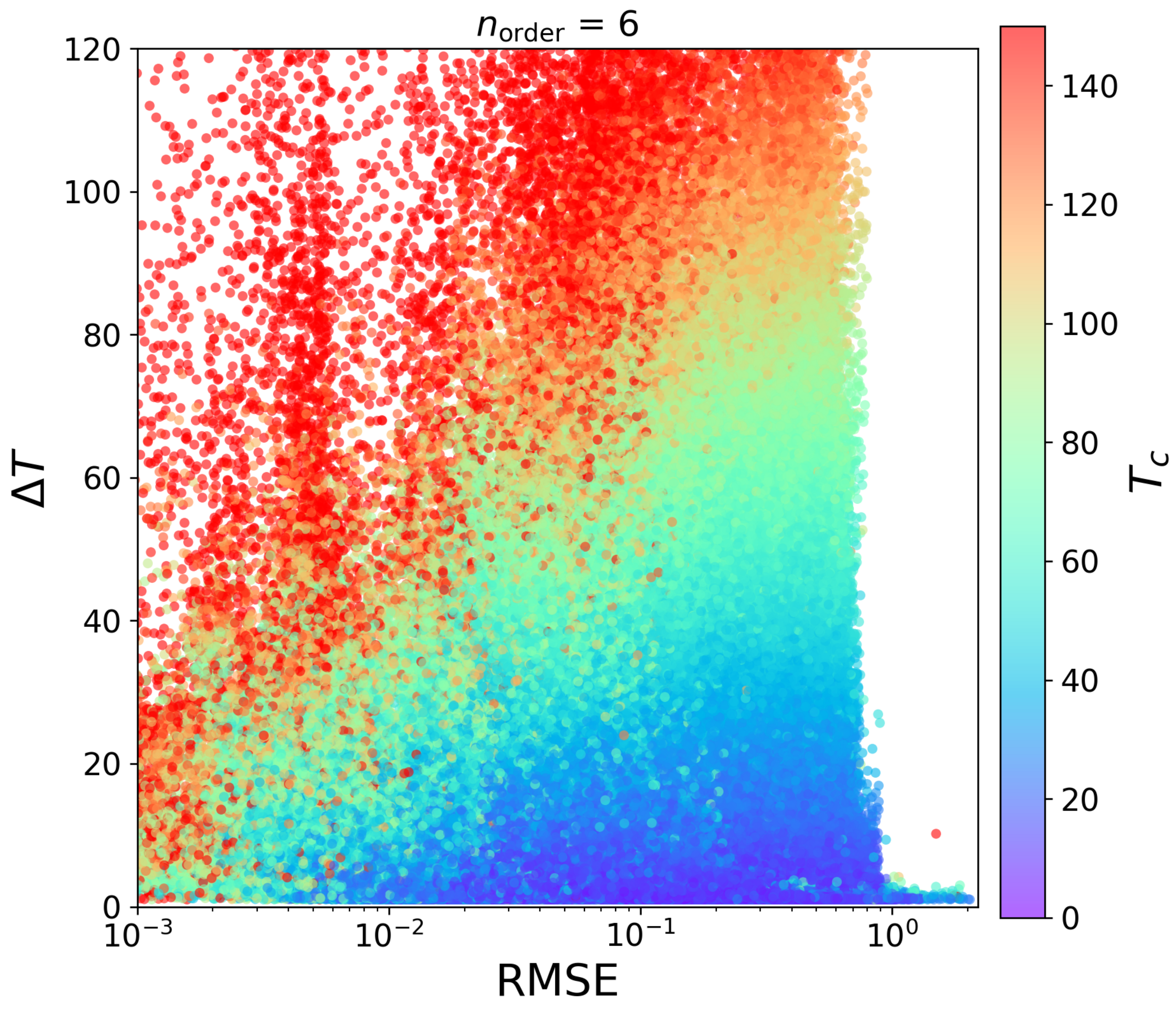
- The fitting results align very well with the raw data.
- We utilize the root mean square error (RMSE) to quantify the degree of agreement,

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{S_E(T_i)}{T_i} - \frac{\hat{S}_E(T_i)}{T_i}\right)^2}$$

- With the factor of $(T - T_C)^2$, the MSE drops quickly with the increasing of n_{order}.

Action curve for one-dimensional toy model





- A random scan in

$c \in [0,2]$, $m^2 \in [0,200]$,

$\lambda \in [0,2]$, $\kappa \in [-30,0]$.

- The majority of samples exhibit an MSE below 1, with a maximum value of 4.2.

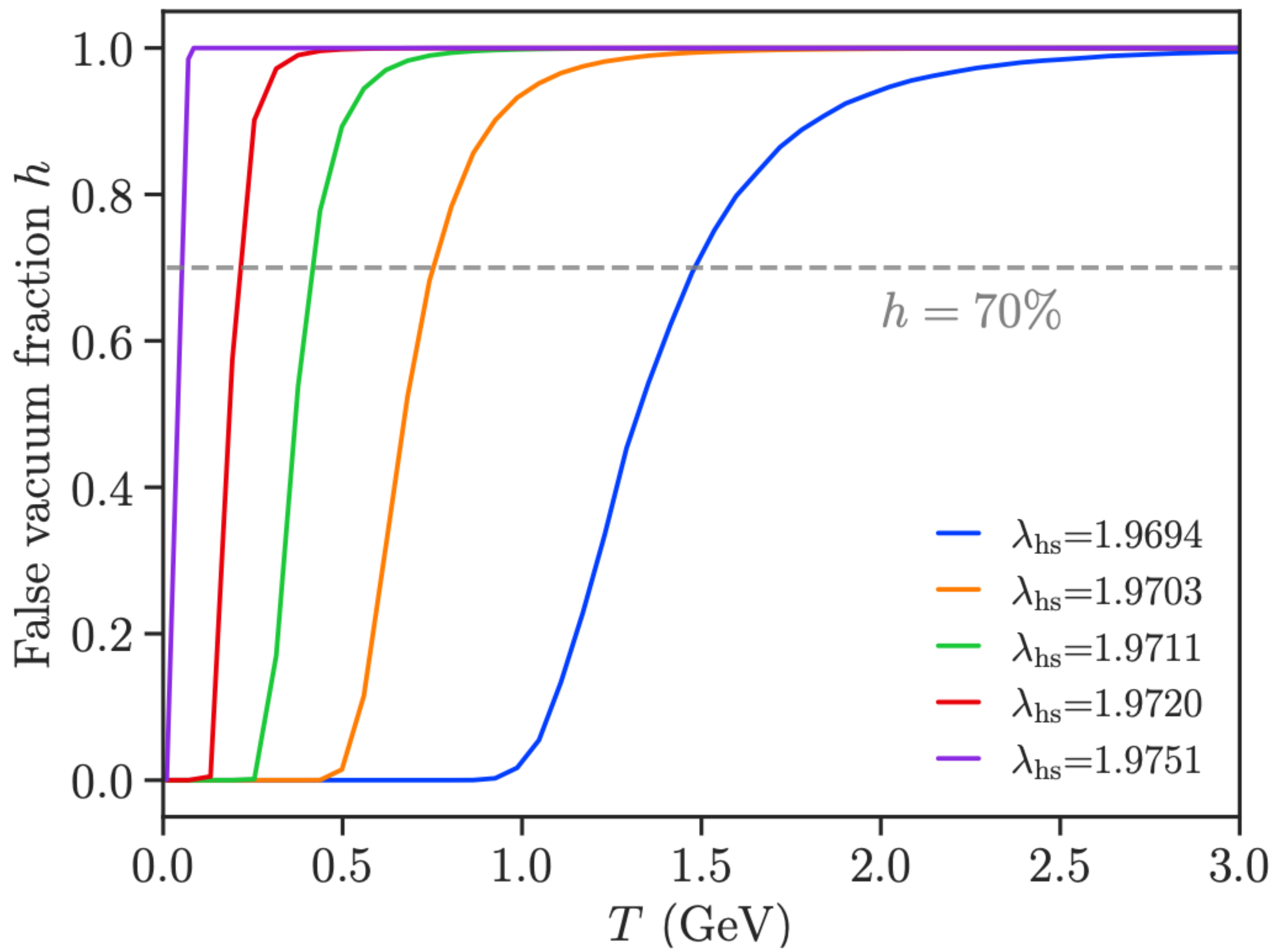
$$S_E/T_{\rm nuc} \simeq 140$$

- The RMSE exceeds 1 only for $\Delta T < 2$ GeV

Action curve for one-dimensional toy model

- With the action curve function, we can calculate the percolation temperature T_P fairly quick.

$$P(T_{\rm P}) = \exp\left[-\frac{64\pi}{3}\xi^4\int_{T_{\rm per}}^{T_{\rm tra}} dT' \frac{\Gamma(T')}{T'^6}\left(\frac{1}{T_{\rm P}}-\frac{1}{T'}\right)^3\right] = 70\,\%$$



		c	m^2	κ	λ	T_C	T_N	T_P	Time
BP1	without action fit	0.1	100	-10	0.01	161.2	125.7	122.0	5.60s
	with action fit					161.2	125.7	122.0	0.04s
BP2	without action fit	0.1	100	-10	0.02	116.2	101.5	99.6	7.54s
	with action fit					116.2	101.4	99.6	0.05s
BP3	without action fit	0.1	100	-10	0.05	77.5	73.0	72.4	10.8s
	with action fit					77.5	73.0	72.4	0.05s
BP4	without action fit	0.1	100	-10	0.1	59.2	57.4	57.2	11.46s
	with action fit					59.2	57.4	57.2	0.05s

- Without action fit: Each integration step requires multiple evaluations of the action, and we need several integration to determine the temperature corresponding to 70%.
- With action fit: 30 times evaluations of the action to get the data for fit.

Action curve for SSM

- Now we turn to a physical model, the singlet scalar extensions of the Standard Model, which is wildly used in instructional studies of phase transition:

$$V_0(h,s) = -\frac{\mu_H^2}{2}h^2 + \frac{\lambda_H}{4}h^4 - \frac{\mu_S^2}{2}s^2 + \frac{\lambda_S}{4}s^4 + \frac{\lambda_{HS}}{4}h^2 s^2$$

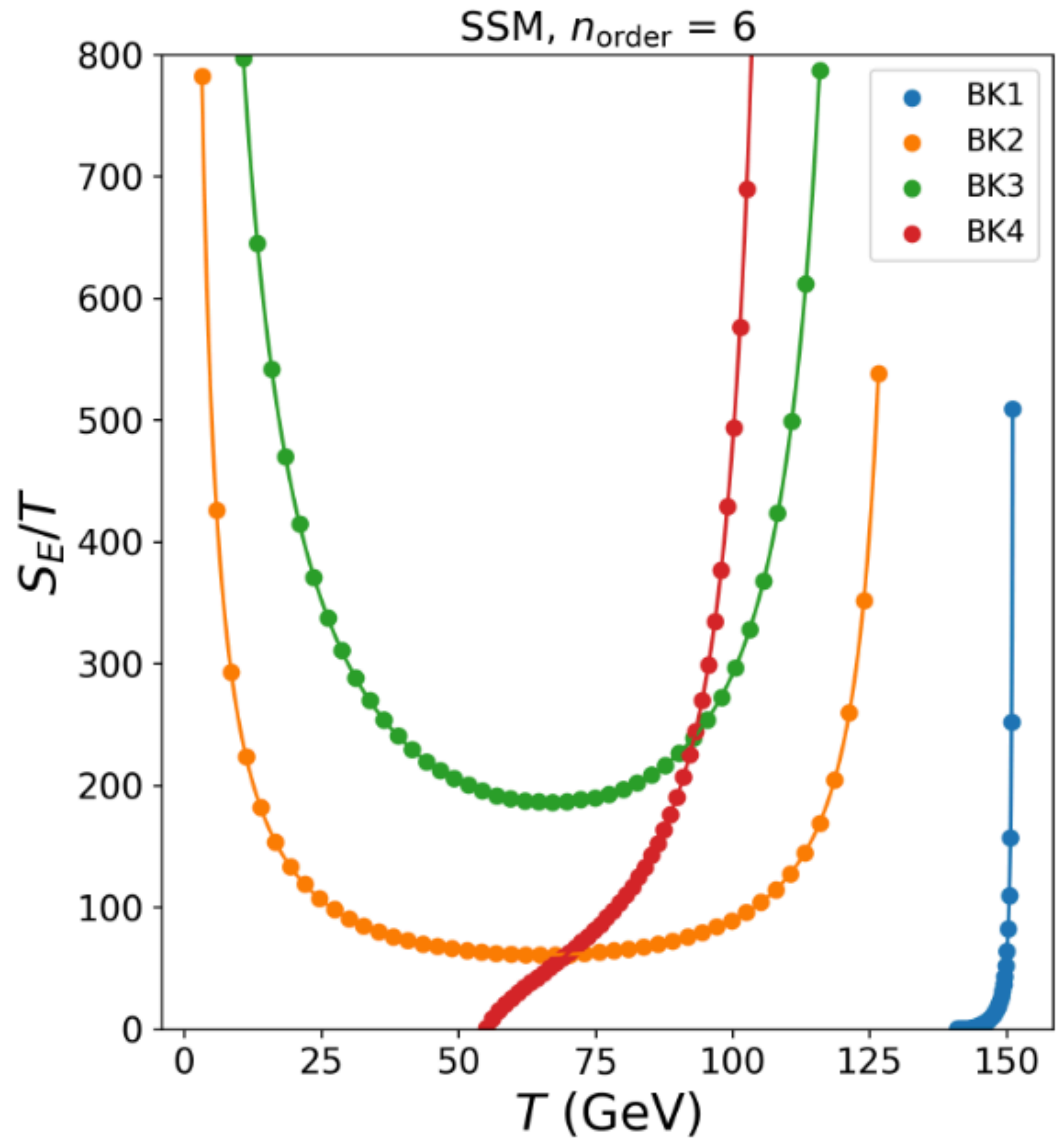
$$V_{\rm eff}(h,s;T) = V_0(h,s) + V_{\rm CW}(h,s) + V_{\rm CT}(h,s) + V_{\rm 1T}(h,s;T) + V_{\rm ring}(h,s;T)$$

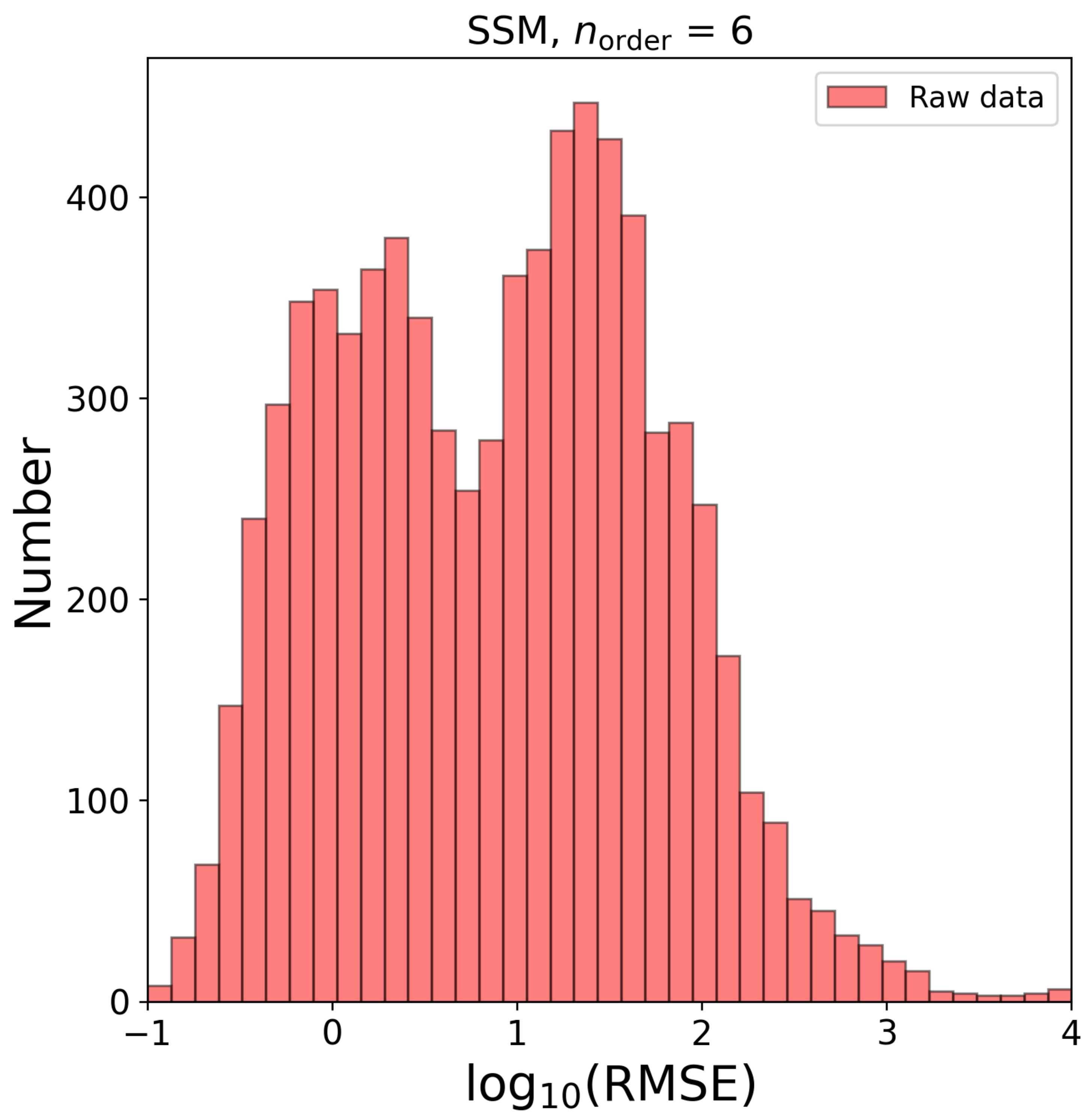
We choose the OS-like scheme, the Landau gauge and the Parwani method:

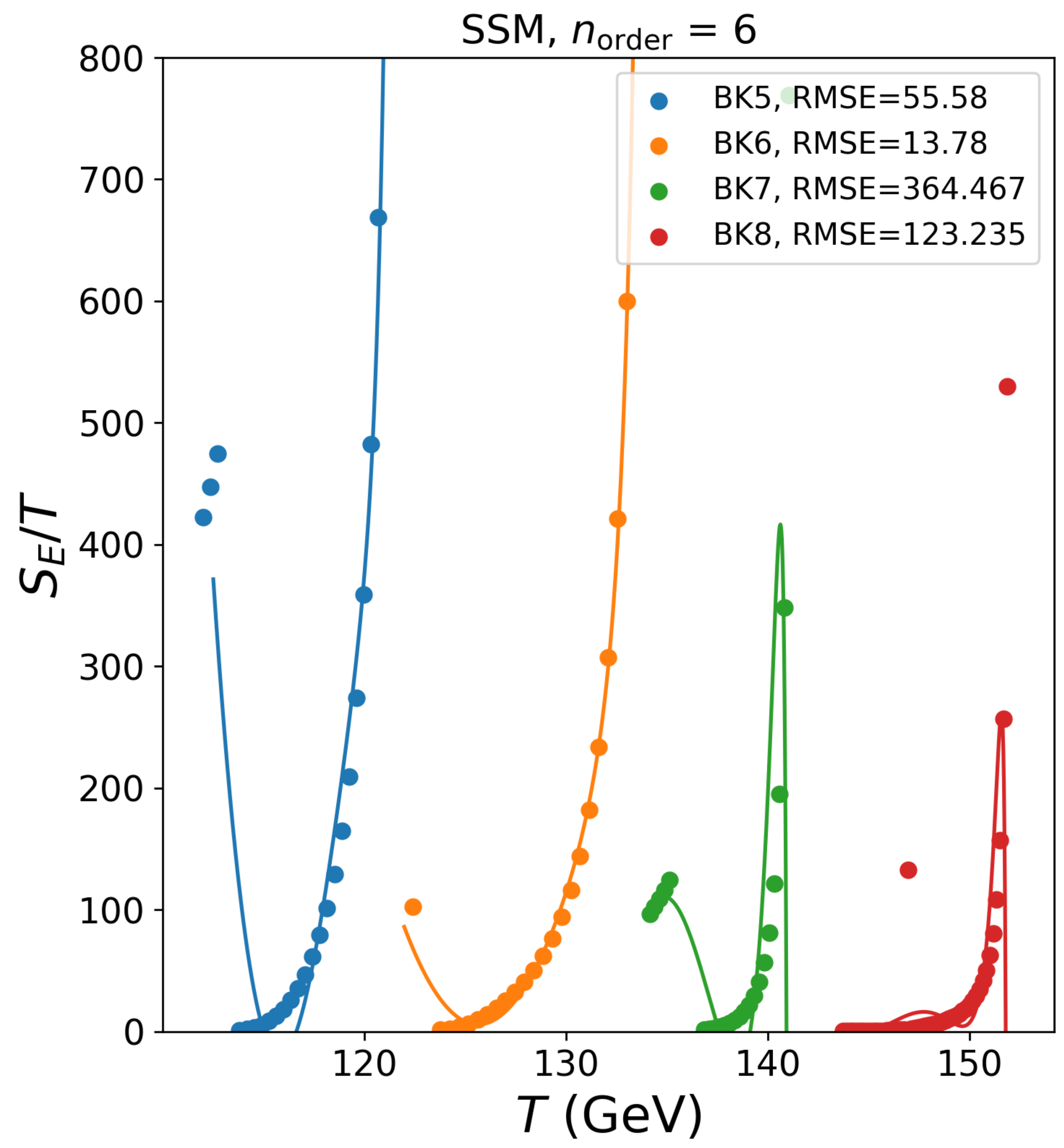
$$V_{\rm CW}(h,s) + V_{\rm CT}(h,s) = \sum_i (-1)^{s_i} \frac{g_i}{64\pi^2} \left\{ m_i^4(h,s) \left[\log \frac{m_i^2(h,s)}{m_i^2(v_h,v_s)} - \frac{3}{2} \right] + 2 m_i^2(h,s) m_i^2(v_h,v_s) \right\}$$

$$V_{\rm 1T}(h,s) = \frac{T^4}{2\pi^2} \left[\sum_B g_B J_B \left(\frac{m_B(h,s)}{T} \right) + \sum_F g_F J_F \left(\frac{m_F(h,s)}{T} \right) \right]$$

Action curve for SSM

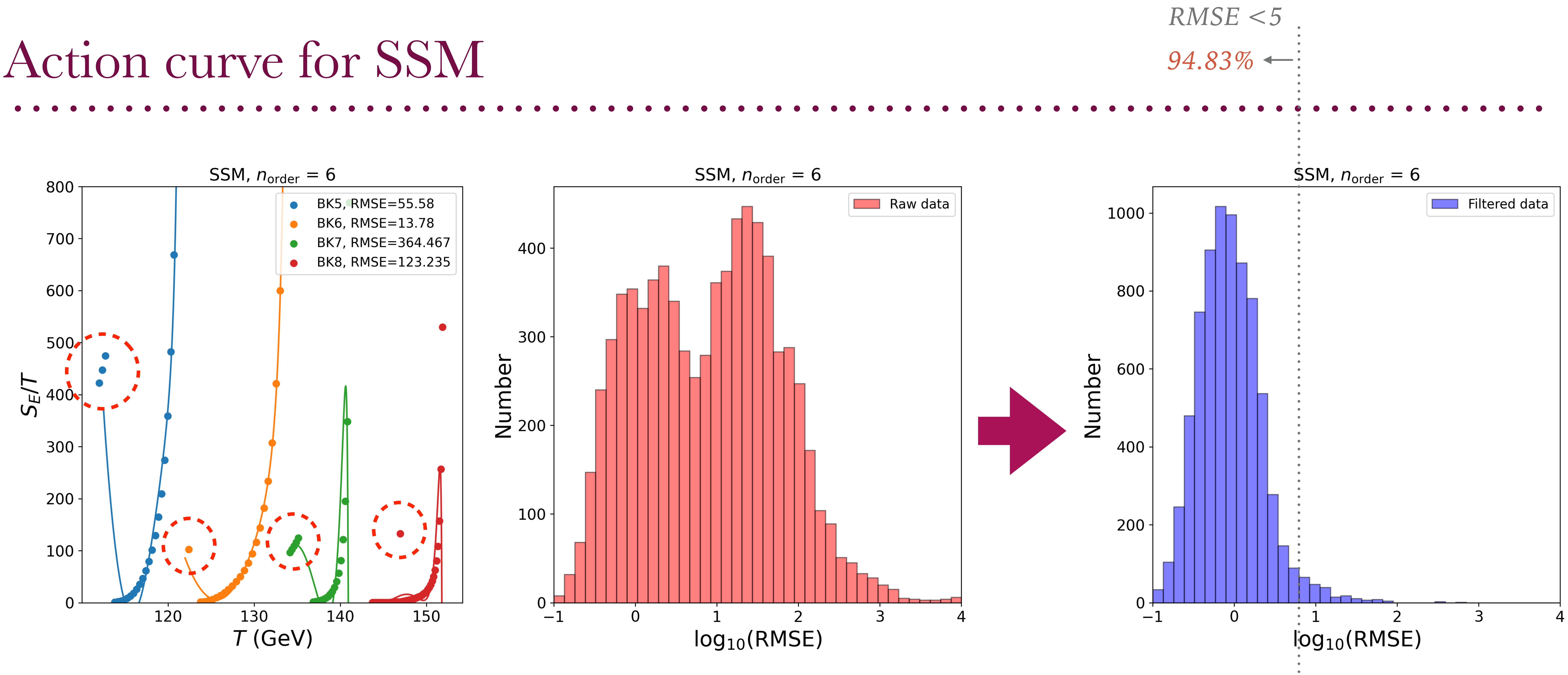






- For some of the samples, the fitting results also align very well with the calculated action.
- Half of the samples have large RMSE, because of incorrect S_E.

Action curve for SSM



- We removes the abnormal points using

$$S(T_i) - S(T_{i-1}) < 0 \quad \text{and} \quad |S(T_i) - S(T_{i-1})| < |S(T_{i-1}) - S(T_{i-2})|$$

- As temperature decreases, the action should decrease monotonically, and the rate of decrease should slow.

Action curve for SSM

Calculate one β/H for 10 times with different h

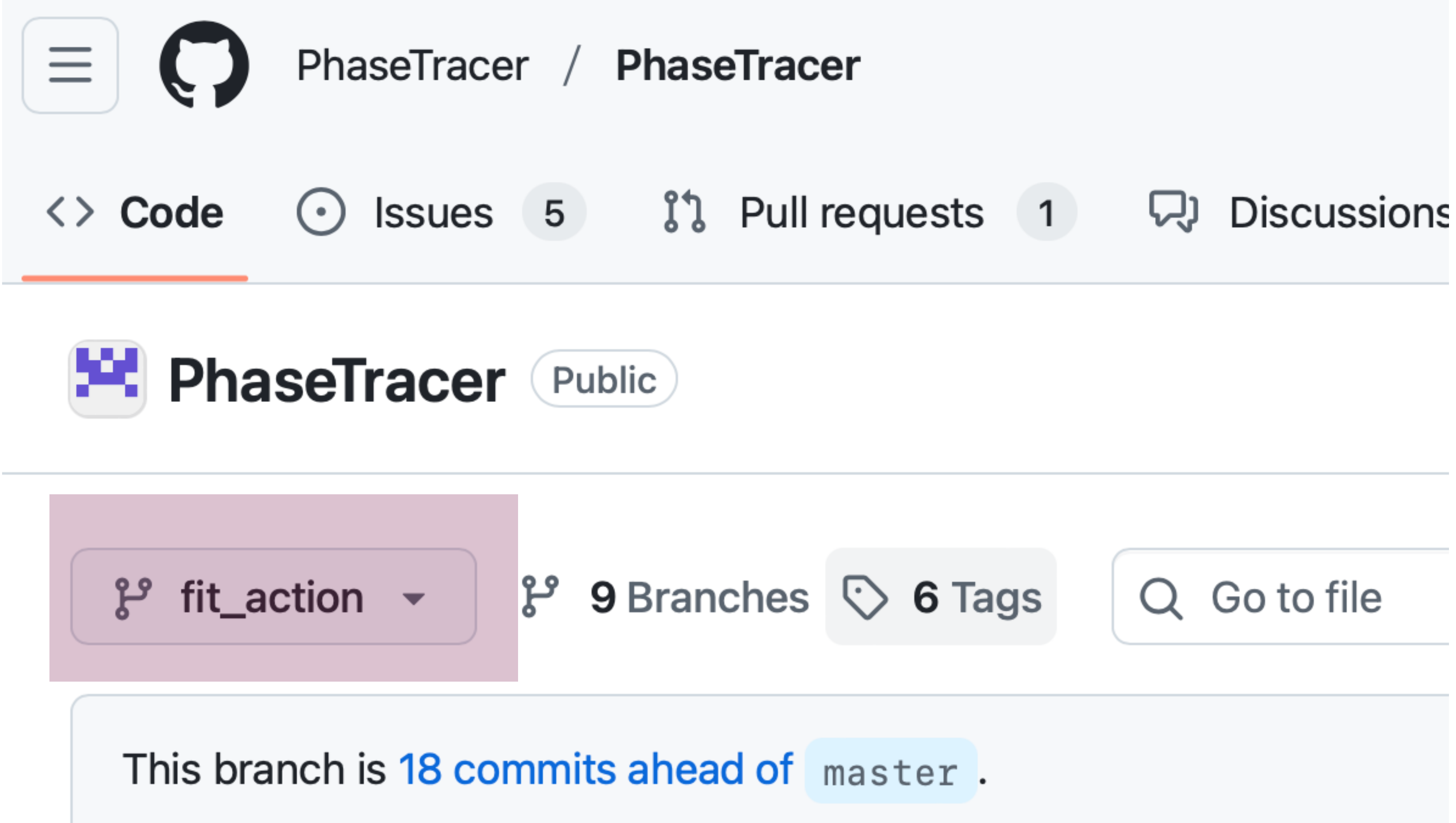
	$h=1$	$h=0.1$	$h=0.01$	$h=0.001$	Fititng
1	1621.95	1573.90	1530.38	1461.57	1500.25
2	1622.96	1534.27	1915.43	901.45	1500.24
3	1622.34	1533.25	1505.82	991.253	1499.92
4	1618.56	1503.12	1278.73	-1502.6	1500.33
5	1621.08	1466.95	1202.39	5570.78	1500.12
6	1579.43	1503.90	1635.98	845.765	1500.00
7	1622.35	1506.99	1691.52	1728.72	1499.98
8	1623.08	1517.53	1077.92	2533.38	1500.44
9	1620.85	1503.17	1171.85	-380.135	1500.08
10	1622.14	1523.33	1812.96	1863.13	1500.13
Mean	1617.47	1516.64	1482.30	1401.33	1500.15
Uncertainty	12.74	26.46	273.30	1769.83	0.15

$$\frac{\mathrm{d}f(x)}{\mathrm{d}x} = \frac{f(x+h) - f(x-h)}{2h}$$

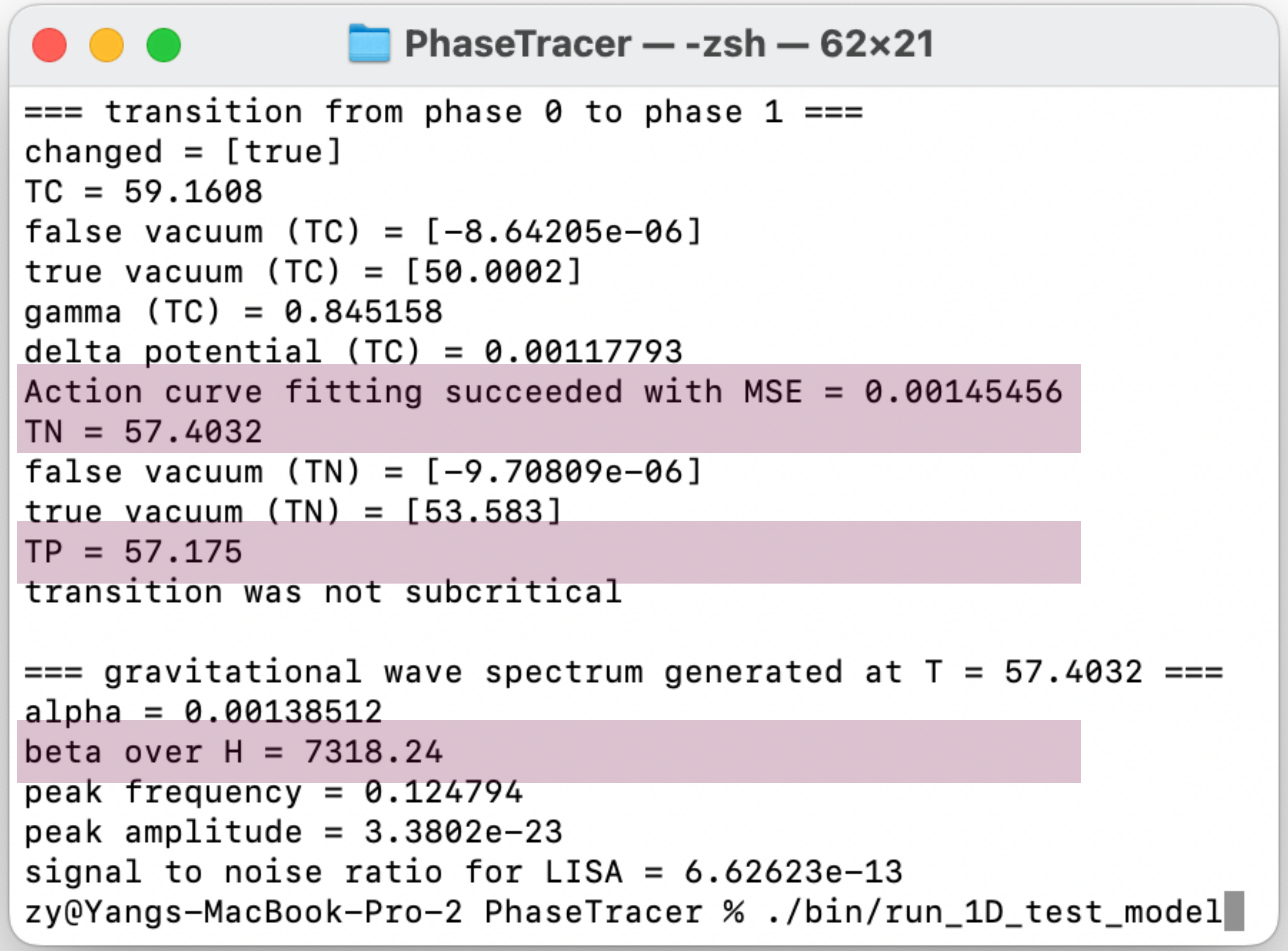
- It is doable to model the action curve using polynomial fitting.
- For one benchmark point, we only need to calculate action about 30 times, then we can
  - Precisely calculate the β
  - Improve the calculation of $A(T)$
  - Rapidly calculate the $T_{\rm nuc}$ and $T_{\rm per}$.

Action curve fit in PhaseTracer2

- We have added the action fitting in the "fit_action" branch of PhaseTracer2, and will merge to the master branch soon.

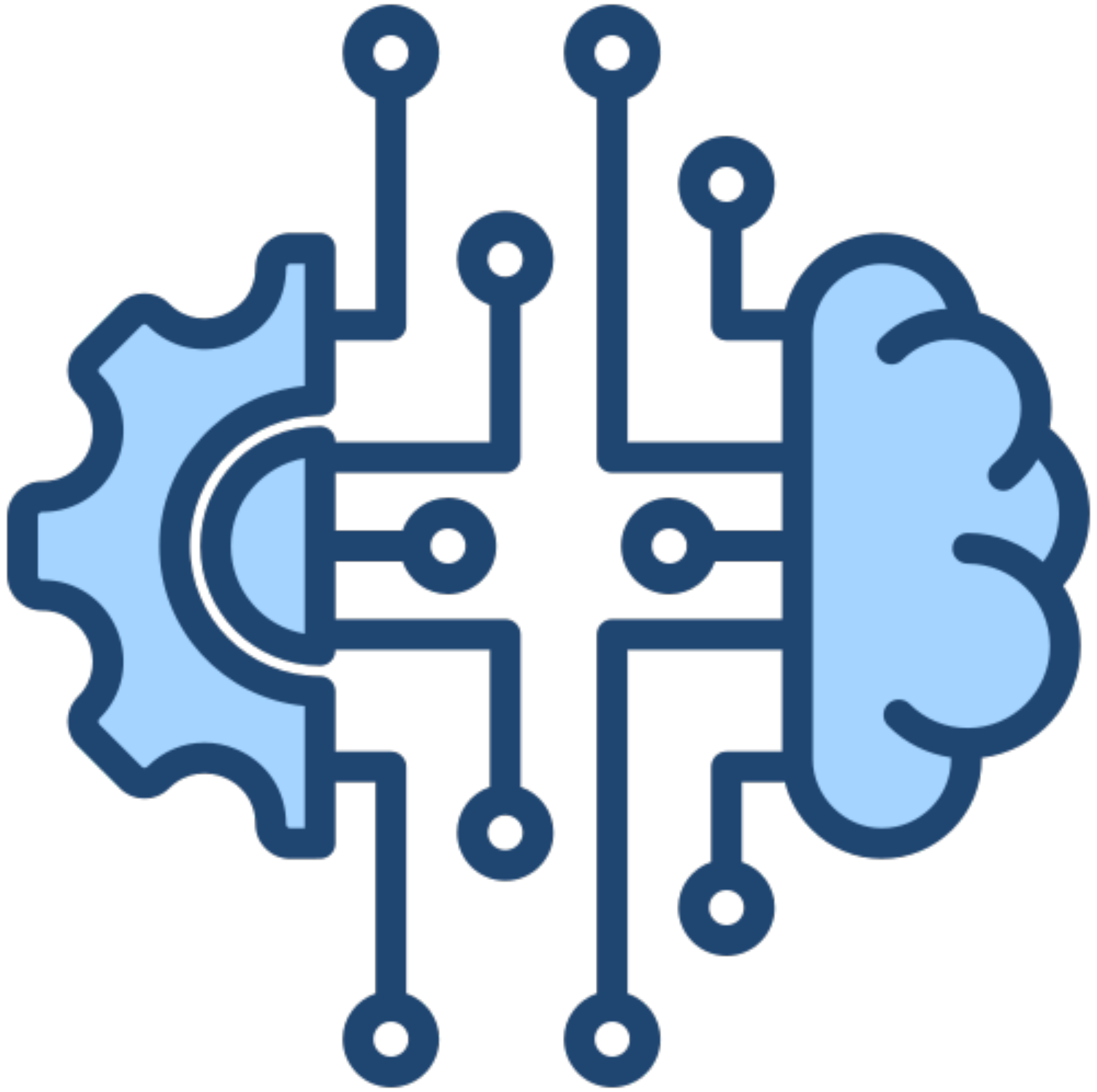



```
// Make TransitionFinder object and find the transitions
PhaseTracer::TransitionFinder tf(pf, ac);
tf.set_fit_action_curve(true);
tf.set_calculate_percolation(true);
tf.find_transitions();
std::cout << tf;
```

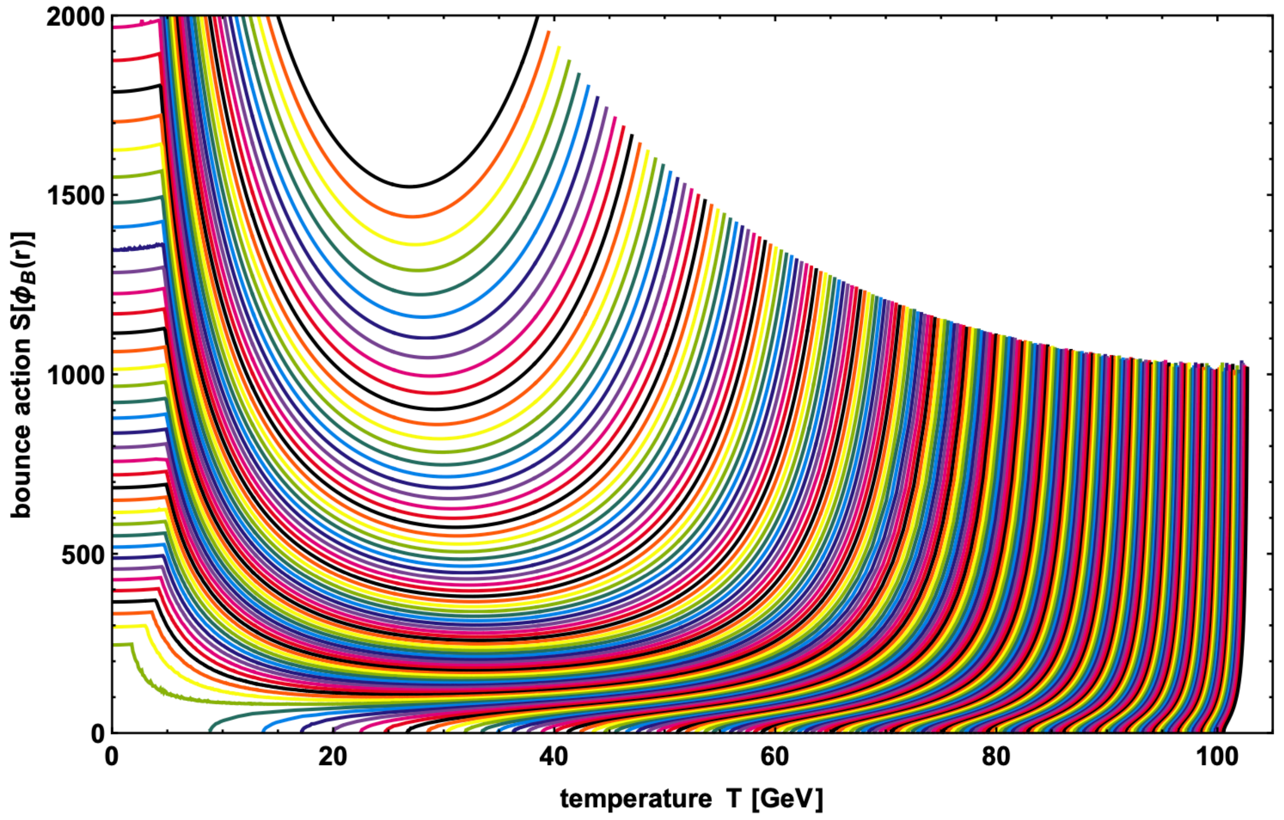


Neural network for action curve

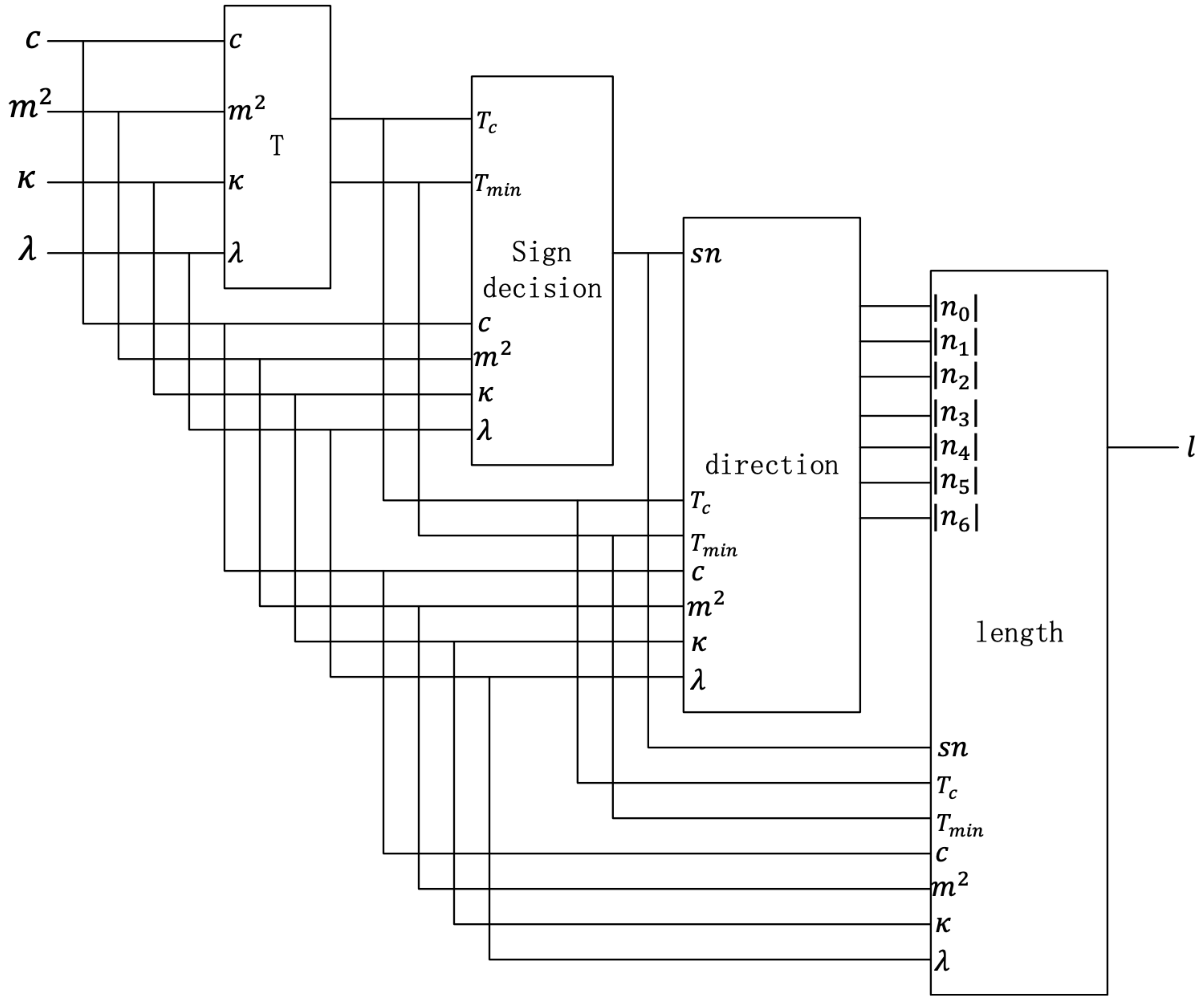
Model Parameters



$$S_E = \frac{1}{(T - T_c)^2} \sum_{i=0}^{n_{\mathrm{order}}} q_i T^i$$



Neural network for action curve

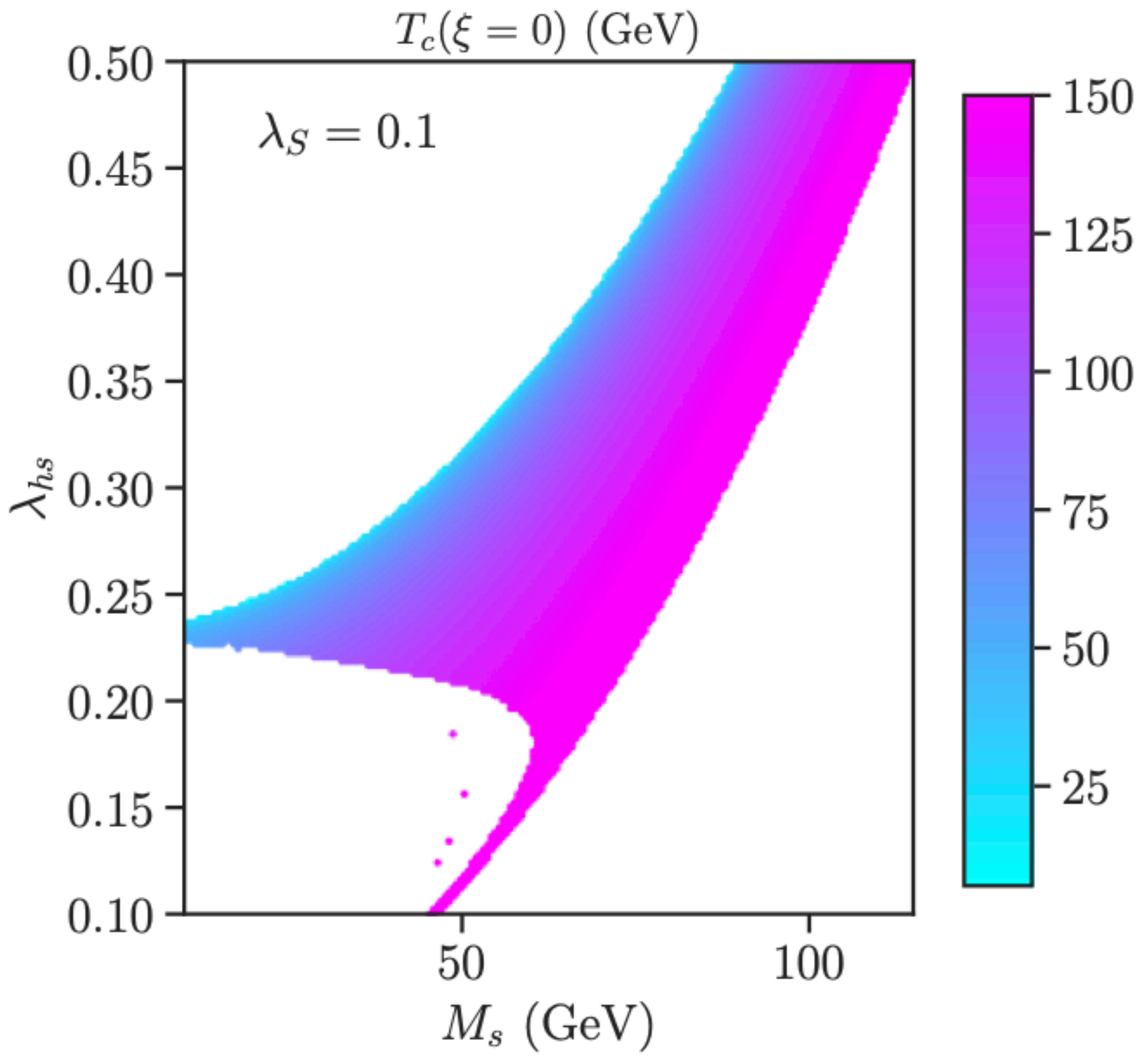


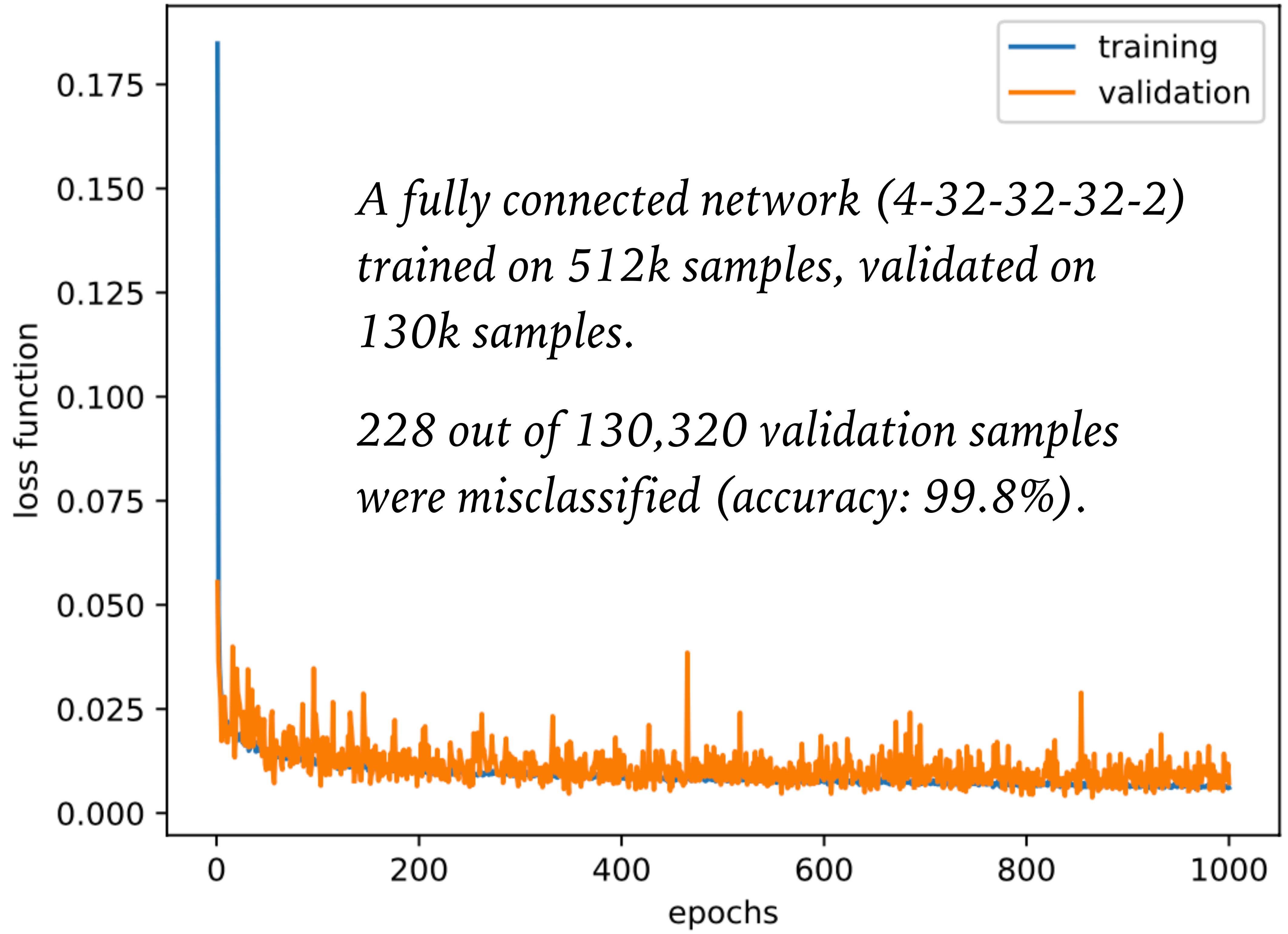
- Step 1: distinguish parameter regions where valid first-order phase transition occurs.
- Step 2: predict the overlap temperature range of the two phases, i.e. T_C and $T_{\min}$.
- Step 3: predict the polynomial coefficients.
- Step 4: validate the accuracy using a few point in the curve.

➤ We employ the conventional fully connected neural network for the first two step, and the Kolmogorov-Arnold Network (KAN) for step 3.

Neural network for action curve

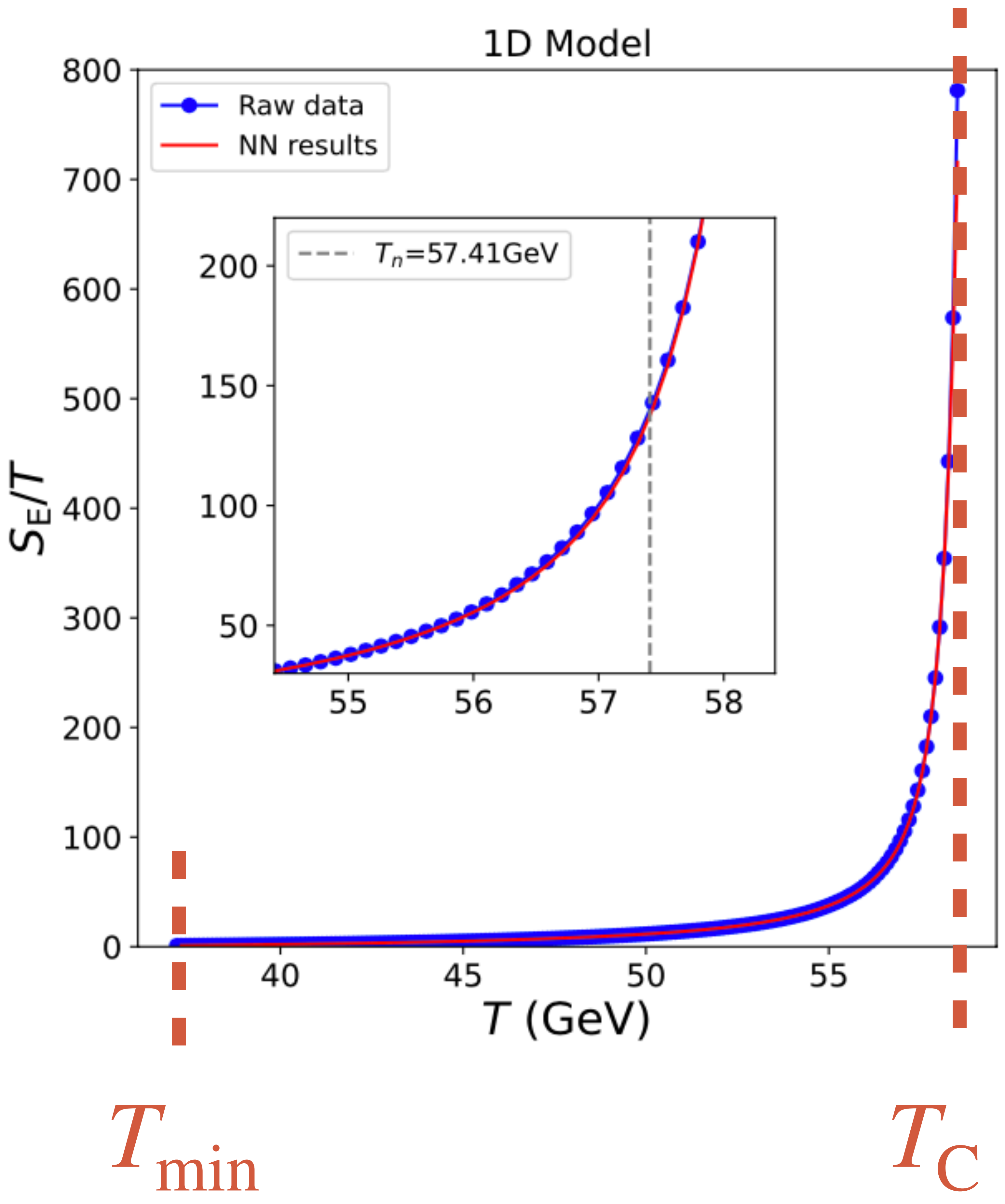
- Step 1: Distinguish the parameter space that has valid action curve
- We utilize the 1D toy model to illustrate the performance of machine learning.

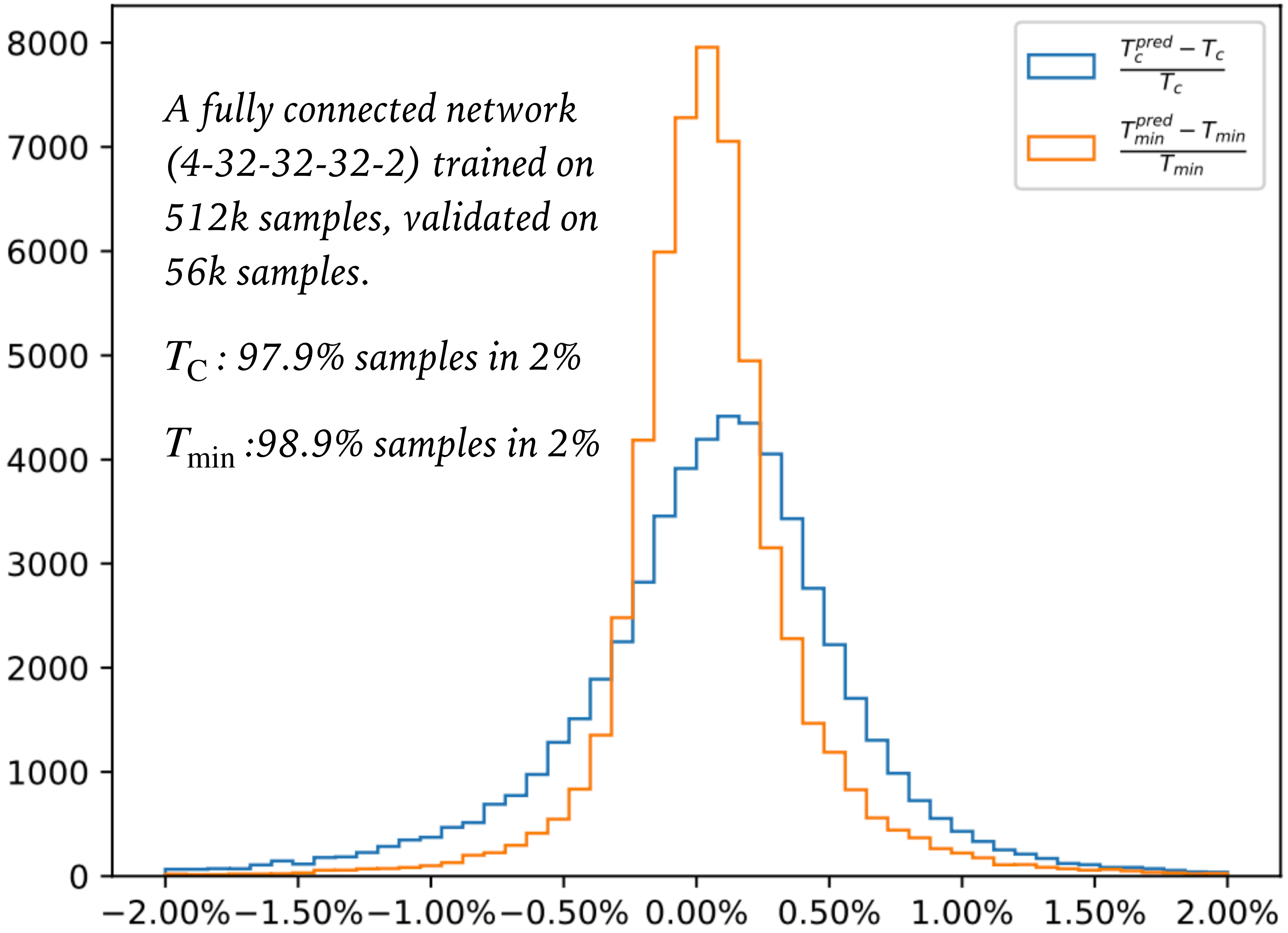




Neural network for action curve

➤ Step 2: Predict T_C and T_{min}





A fully connected network (4-32-32-32-2) trained on 512k samples, validated on 56k samples.

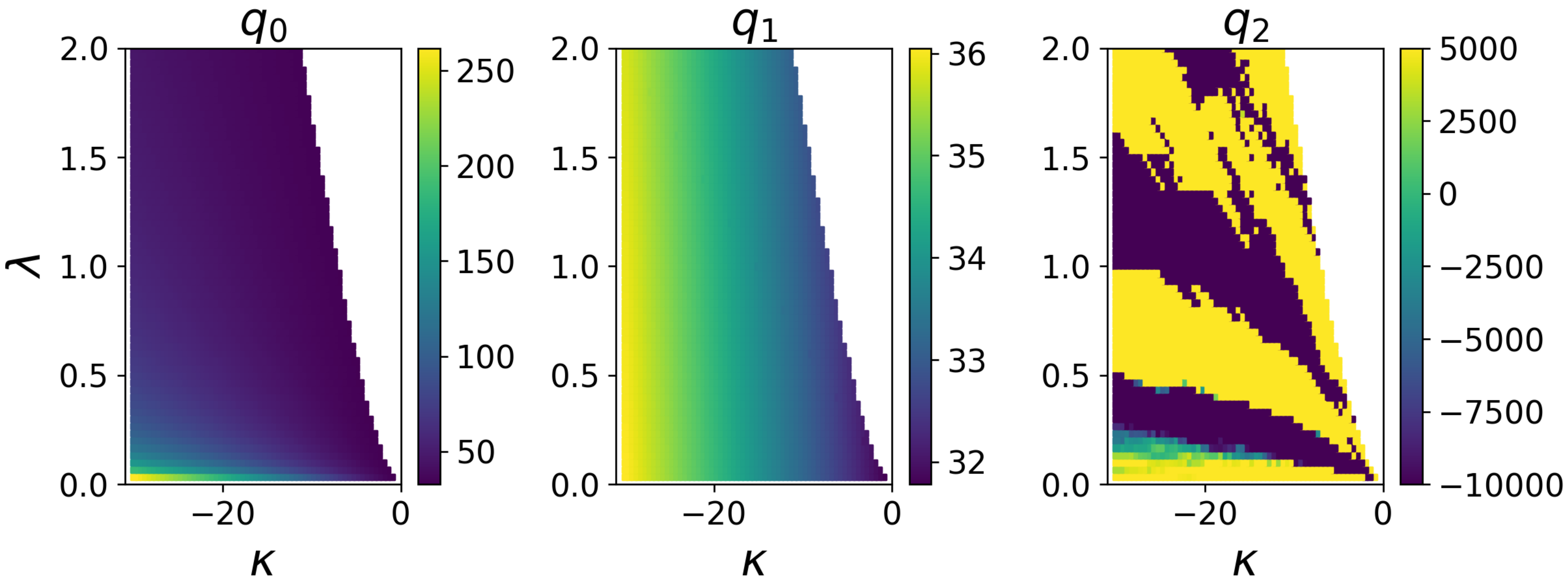
T_C *: 97.9% samples in 2%*

T_{min} *:98.9% samples in 2%*

Neural network for action curve

$$S_E = \frac{1}{(T-T_c)^2}\sum_{i=0}^{n_{\text{order}}} q_i T^i$$

- Step 3: Predict the polynomial coefficients



- The variation of some coefficients with the input parameters is not smooth, covering a wide range and occasionally switching sign.

Neural network for action curve

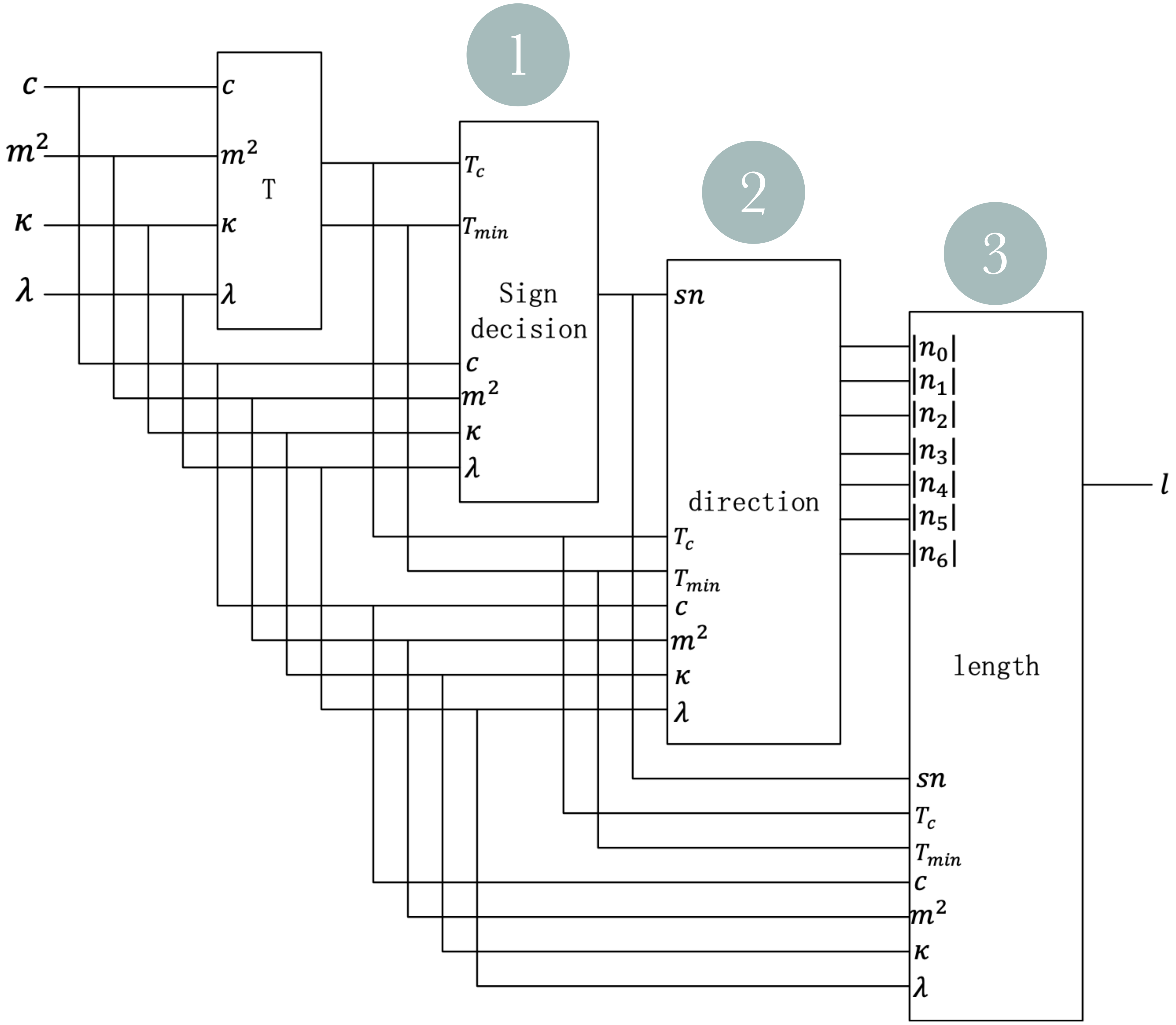
- Step 3: Predict the polynomial coefficients
  - We have to predict the function as a whole, rather than predicting the coefficients separately.
  - The difference between two functions can be measured using

$$\epsilon = \int_{x_0}^{x_1} dx \left(f_1(x) - f_2(x)\right)^2$$

    which is the Euclidean distance in a Hilbert space.
  - Thus, we construct an orthogonal function set by subtracting the projection of the original function onto the existing orthogonal basis, following the Gram-Schmidt orthogonalization process.
  - We use a fast KAN network with architecture $13-64-64-64-64-64-1$

Neural network for action curve

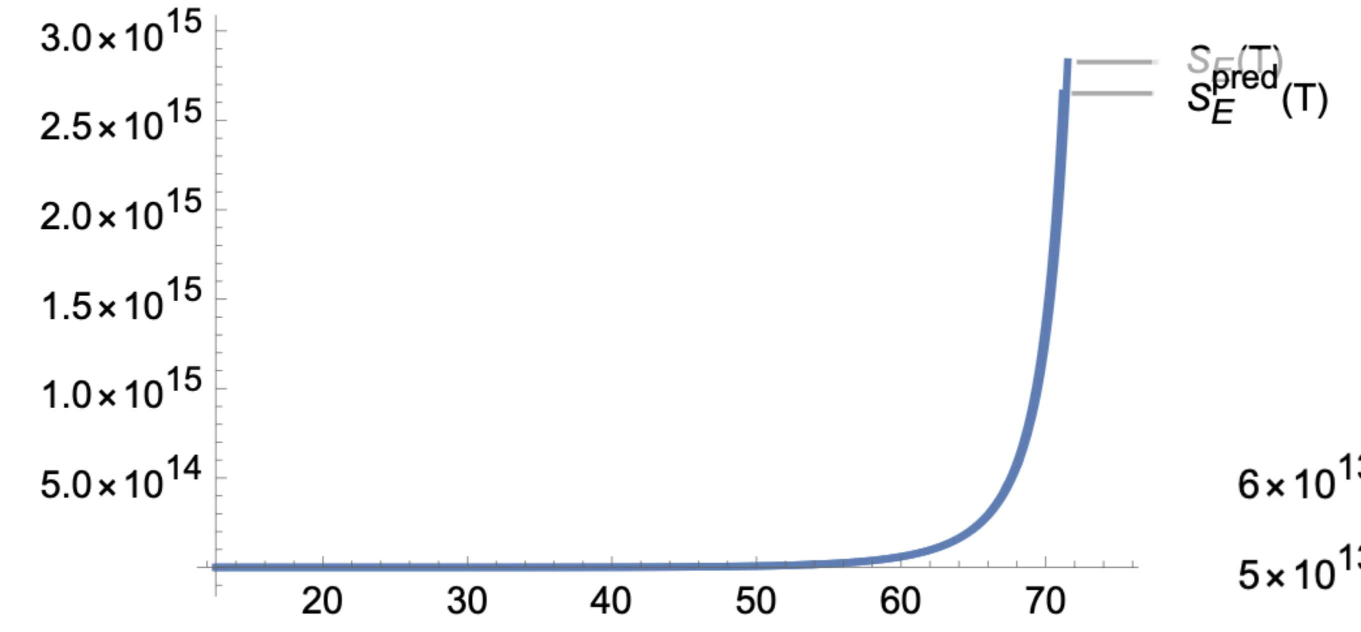
➤ Step 3: Predict the polynomial coefficients

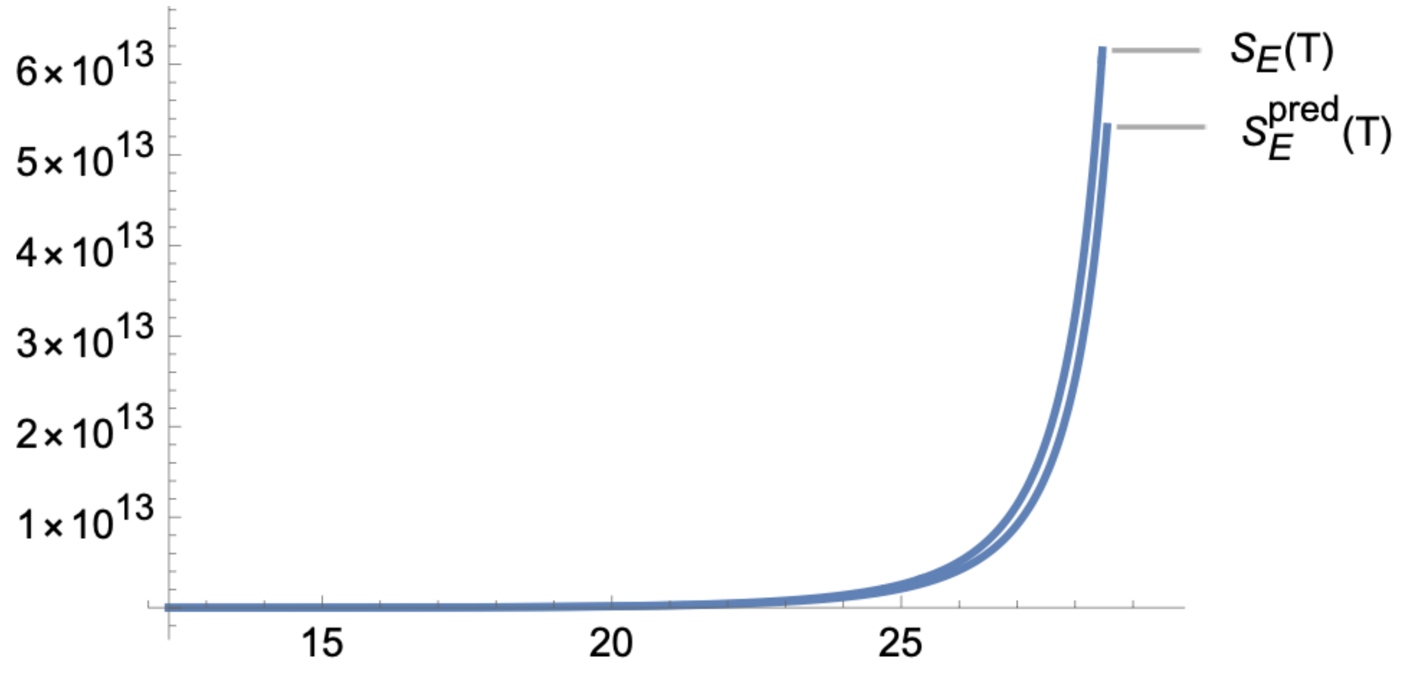


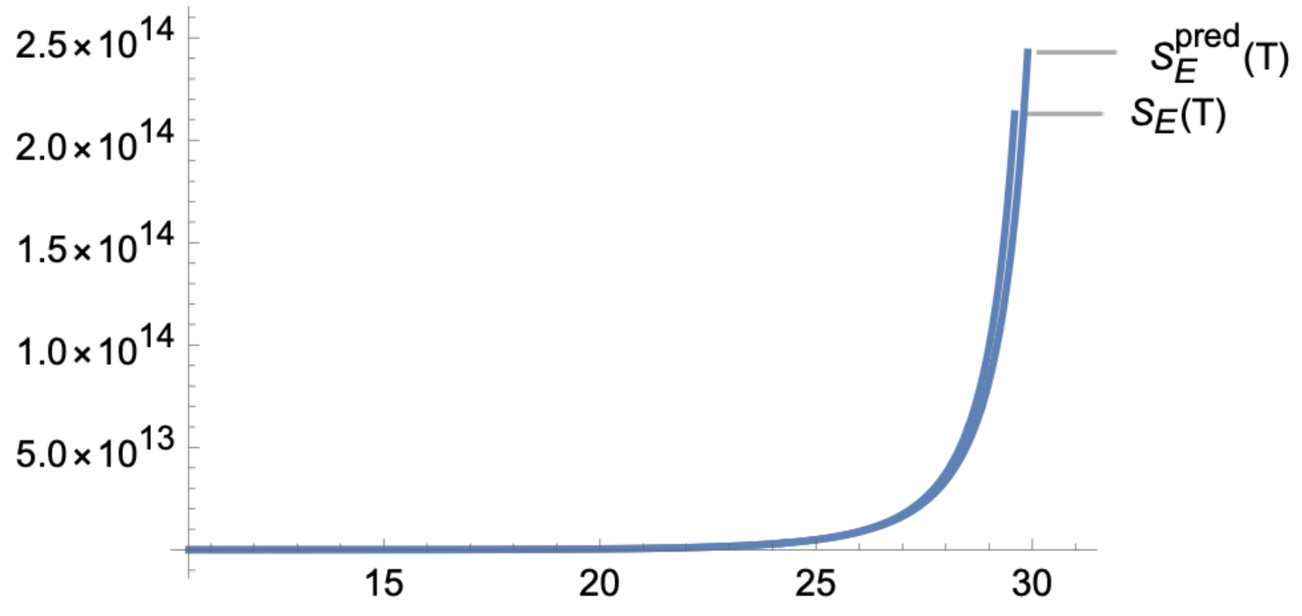
- Network-1: predict the sign of the coefficients
  - ➡ 96.69% accurate
- Network-2: predicts the direction $|n_i|$ of the polynomial in the Hilbert space
  - ➡ 96.59% accurate
- Network-3: predicts the length of the polynomial in the Hilbert space
  - ➡ 71.48% accurate

Neural network for action curve

➤ Step 3: Predict the polynomial coefficients







- Network-1: predict the sign of the coefficients

  ➡ 96.69% accurate

- Network-2: predicts the direction $|n_i|$ of the polynomial in the Hilbert space

  ➡ 96.59% accurate

- Network-3: predicts the length of the polynomial in the Hilbert space

  ➡ 71.48% accurate

Summary

- Using machine learning to predict action curve function is more practical than predicting isolated action value or observable. It provides the flexibility to perform subsequent calculations derived from the action curve.
- Neural networks, particularly KANs, show promise in achieving this objective, although their predictive accuracy requires further improvement.
- With action curve fitting, we can calculate the T_P and T_N quickly, and the β more accurately. This is independent of machine learning, and can be used in PhaseTracer now.

Thanks!