Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

SFB
SFB 925: Light induced dynamics and control of strongly correlated quantum systems

Hamburg, 03.05.2019

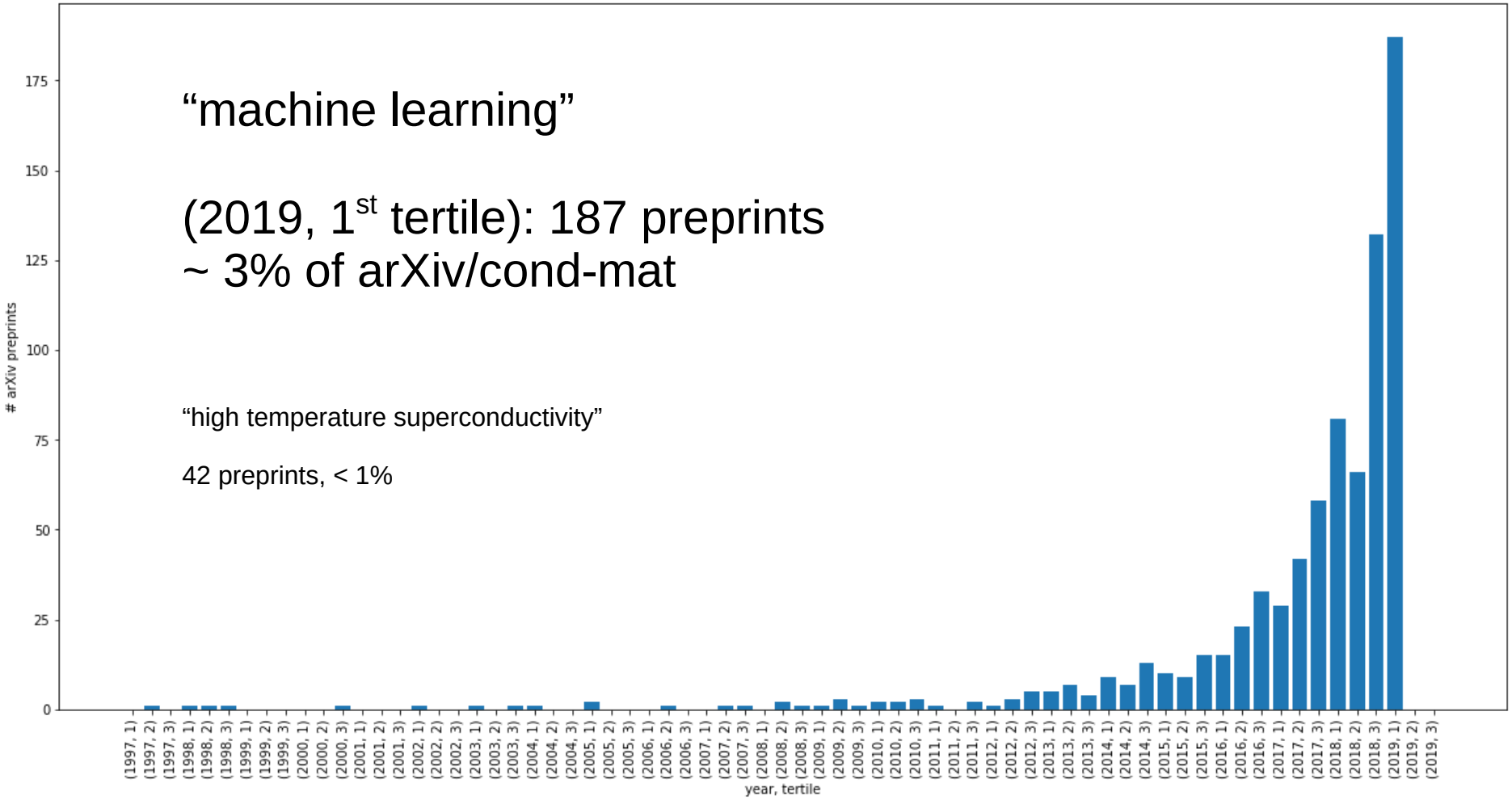
Machine Learning Green Functions from Perturbation Theories

Patryk Kubiczek

patryk.kubiczek@physik.uni-hamburg.de

I. Institute for Theoretical Physics, University of Hamburg

Machine learning: exponential growth of popularity



Machine learning and physics

"The immense progress in computing power and the corresponding availability of large datasets ensure that machine learning will be an important part of the physics toolkit.

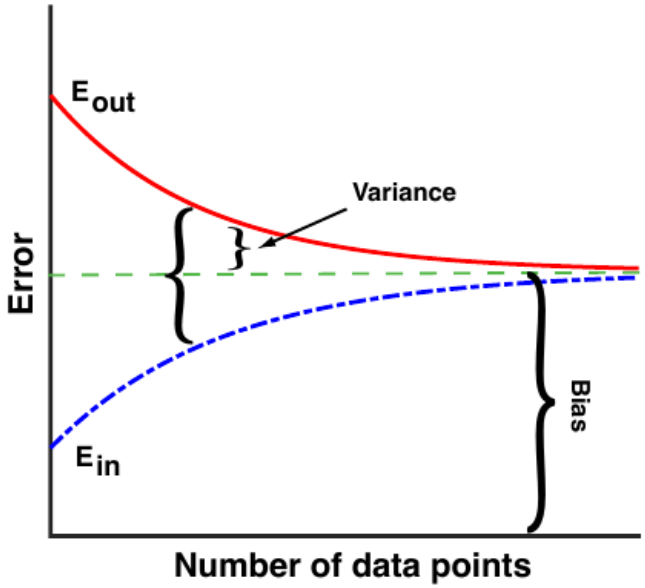
In the future, we expect machine learning to be a core competency of physicists much like linear algebra, group theory, and differential equations."

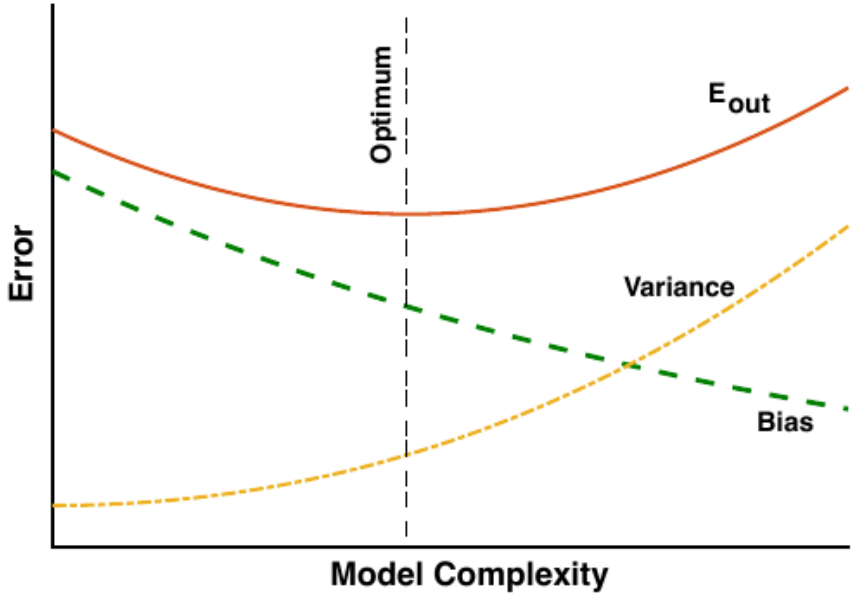
Mehta, Bukov, Wang, Day, Richardson, Fisher, Schwab:
A high-bias, low-variance introduction to Machine Learning for physicists
arXiv:1803.08823

See also:
Carleo et al.*: Machine learning and the physical sciences*
arXiv:1903.10563

Machine learning

- **Goal: learn from data to predict or generate new data**
- Predicting data ≠ fitting data
- Bias–variance trade-off:
  - Underfitting = high bias (model too simple)
  - Overfitting = large difference of in-sample and out-of-sample error (training dataset too small)





Applications in condensed matter physics: examples

Supervised learning

Classification of phases

Carrasquilla, Melko: *Machine learning phases of matter* (2017)

Broecker, Carrasquilla, Melko, Trebst: *Machine learning quantum phases of matter beyond the fermion sign problem* (2017)

Monte Carlo speed-up

Liu, Qi, Meng, Fu: *Self-learning Monte Carlo method* (2017)

Huang, Yang, Wang: *Recommender engine for continuous-time quantum Monte Carlo methods* (2017)

Huang, Wang: *Accelerated Monte Carlo simulations with restricted Boltzmann machines* (2017)

Shen, Liu, Fu: *Self-learning Monte Carlo with deep neural networks* (2018)

"Smart interpolation"

Arsenault, Lopez-Bezanilla, von Lilienfeld, Millis: *Machine learning for many-body physics: The case of the Anderson impurity model* (2014)

Unsupervised learning

Hu, Singh, Scalettar: *Discovering phases, phase transitions, and crossovers through unsupervised machine learning: A critical examination* (2017)

Reinforcement learning

Variational ansatz for many-body states

Carleo, Troyer: *Solving the quantum many-body problem with artificial neural networks* (2017)

Carleo, Nomura, Imada: *Constructing exact representations of quantum many-body systems with deep neural networks* (2018)

Finding of driving protocols

Bukov, Day, Sels, Weinberg, Polkovnikov, Mehta: *Reinforcement Learning in Different Phases of Quantum Control* (2018)

Machine Learning Green Functions from Perturbation Theories

- Anderson impurity model

$$H = \sum_{\sigma} \varepsilon\, d_{\sigma}^{\dagger} d_{\sigma} + U \left(d_{\uparrow}^{\dagger} d_{\uparrow} - \frac{1}{2} \right) \left(d_{\downarrow}^{\dagger} d_{\downarrow} - \frac{1}{2} \right) + \sum_{i\sigma} \varepsilon_i\, c_{i\sigma}^{\dagger} c_{i\sigma} + \sum_{i\sigma} V_i \left(d_{i\sigma}^{\dagger} c_{i\sigma} + c_{i\sigma}^{\dagger} d_{i\sigma} \right)$$

- Approximate perturbative methods of solution usually cheaper than exact
- Small discrete bath => ED solution: cheap
- Continuous bath => QMC solution: hard, in real-time very hard

	Weak coupling PT	Strong coupling PT	ED	QMC
Discrete bath	yes	yes	yes	yes / no
Continuous bath	yes	yes	no	yes / no

- **Question 1:** Can we (computer) learn to predict exact solutions from (complementary) perturbative solutions?

$$G_{\mathrm{exact}} \approx \mathcal{F}(G_{\mathrm{weak}}, G_{\mathrm{strong}}) \quad (?)$$

- **Question 2:** Can we (computer) learn continuous bath solutions from discrete bath solutions?

$$\mathcal{F}_{\mathrm{cont.}} \approx \mathcal{F}_{\mathrm{discrete}} \quad (?)$$

Perturbation Theories

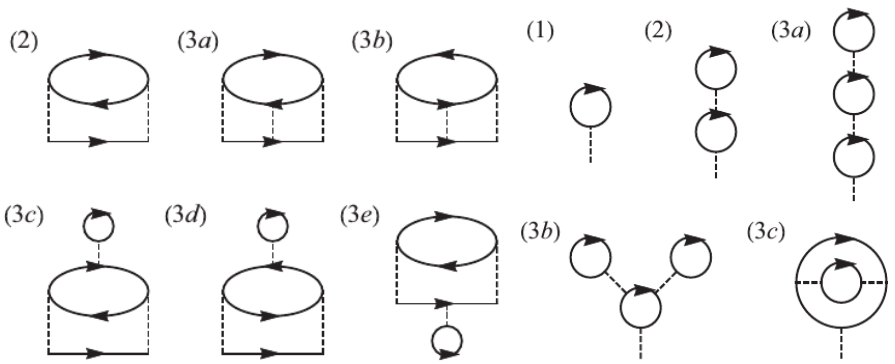
Goal: calculate imaginary-time Green function for AIM

$$G_\sigma = -\left\langle d_\sigma(\tau) d_\sigma^\dagger \right\rangle$$

Weak coupling: 3rd order PT in *U*

$$G = G_0 + G_0 * \Sigma * G$$

self-energy $\Sigma = \Sigma(G_0)$ diagrams:



Tsuji, Werner, PRB **88** 165115 (2013)

Strong coupling: non-crossing approximation (NCA)

$$P = P_0 + P_0 * \Sigma_P * P$$

$$P_0(\tau) = \exp\left[-\tau H(V_i = 0)\right]$$

many-body self-energy

$\Sigma_p = \Sigma_p(P) =$ 

$$G_\sigma(\tau) = \frac{-\mathrm{Tr}\left[P(\beta-\tau) d_\sigma P(\tau) d_\sigma^\dagger\right]}{\mathrm{Tr} P(\beta)}$$

Aoki et al. RMP **86** 779 (2014)

Approaching the learning problem

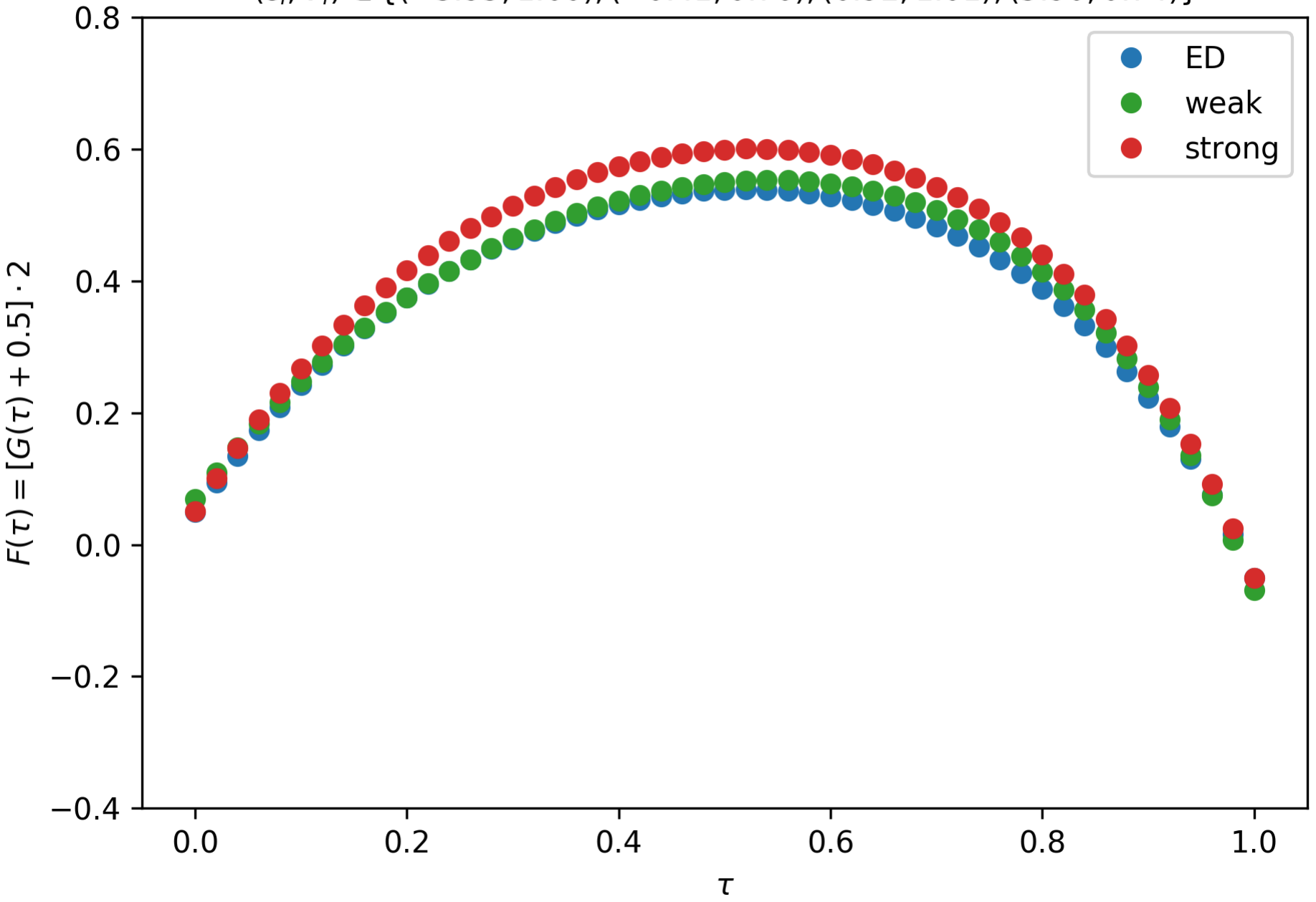
1. **Collect and pre-process the data.**
2. **Define the model and its architecture.**
3. **Choose the cost function and the optimizer.**
4. **Train the model.**
5. **Evaluate and *study* the model performance on the validation and test data.**
6. **Adjust the hyperparameters (and, if necessary, network architecture) to optimize performance for the specific dataset.**

Mehta, Bukov, Wang, Day, Richardson, Fisher, Schwab:
A high-bias, low-variance introduction to Machine Learning for physicists
arXiv:1803.08823

1. Collect and pre-process data

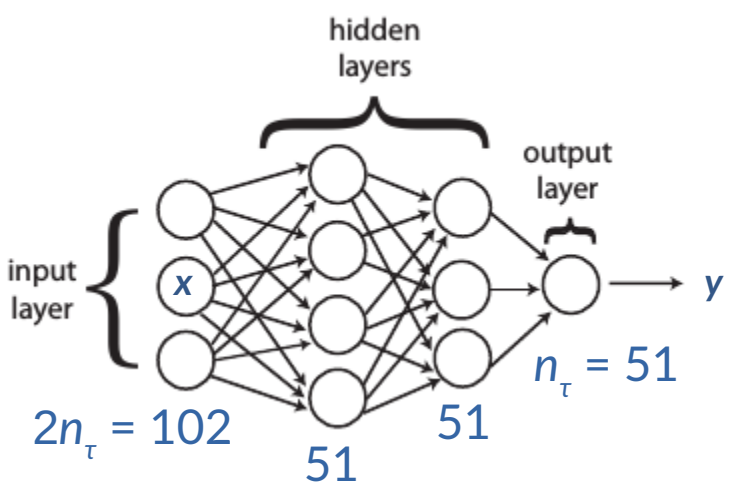
- Generated 10 000 sets of parameters
  - impurity: $U \in [1, 8]$, $\varepsilon \in [-1, 1]$
  - bath: half-bandwidth $D \in [2, 8]$, $N_{\text{bath}} \in \{3, 4, 5\}$
    $\Longrightarrow \varepsilon_i \in [-1, 1]$, $V_i \in [0.75, 1.25]$, i = 1, ..., N_{bath}
    $\Longrightarrow$ normalization $\sum V_i^2 = 2D\Gamma / \pi$, $\Gamma = 1$
    $\Longrightarrow$ centering $\sum \varepsilon_i V_i^2 = 0$ $\Longrightarrow$ rescaling $\varepsilon_N - \varepsilon_1 = 2D$
- Data: 30 000 calculated weak (*TRIQS*) and strong coupling, and ED GFs (*QuSpin*)
  + 30 000 generated GFs from particle-hole symmetry ($\beta = 1$)
- Represent GF as a collection of $n_\tau = 51$ equidistant points $\tau_i \in [0, \beta]$
- Shift and rescale: $F(\tau) = (G(\tau) + 0.5) \cdot 2$

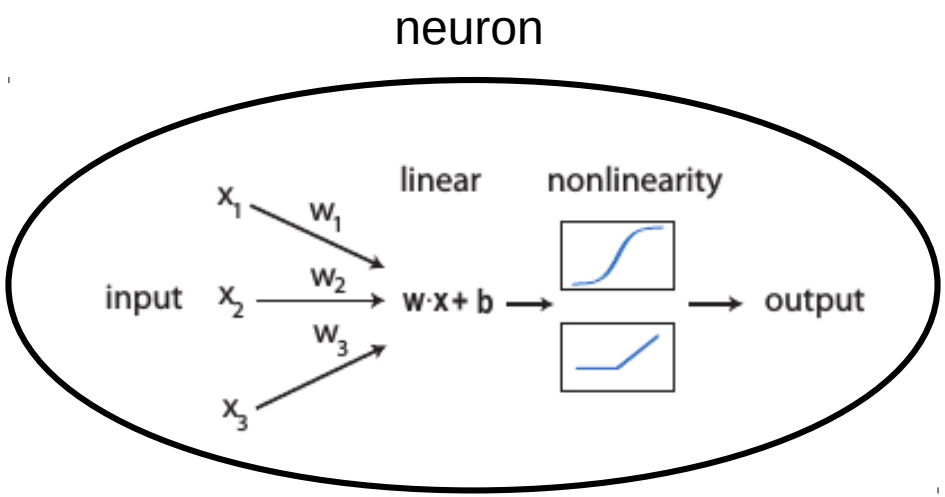
$$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_{20000}]$$
$$\mathbf{x}_i = \left[F_i^{\text{weak}}(\tau_1), \ldots, F_i^{\text{weak}}(\tau_{n_\tau}), F_i^{\text{strong}}(\tau_1), \ldots, F_i^{\text{strong}}(\tau_{n_\tau})\right]$$
$$\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_{20000}]$$
$$\mathbf{y}_i = \left[F_i^{\text{ED}}(\tau_1), \ldots, F_i^{\text{ED}}(\tau_{n_\tau})\right]$$


$U = 7.03, \varepsilon = 0.06$
$(\varepsilon_i, V_i) \in \{(-3.95, 1.00), (-0.41, 0.76), (0.92, 1.01), (5.90, 0.74)\}$
ED
weak
strong
$F(\tau) = [G(\tau) + 0.5] \cdot 2$
τ
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
0.0
0.2
0.4
0.6
0.8
1.0

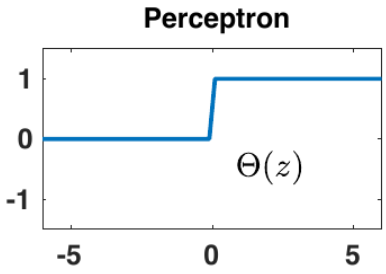
2. Define the model and its architecture

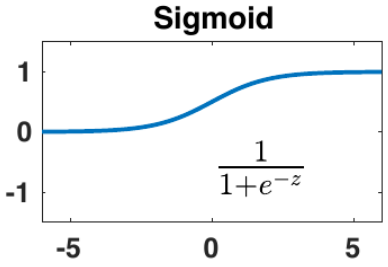
- Model: Feed-Forward Deep Neural Network (*Keras + TensorFlow*)

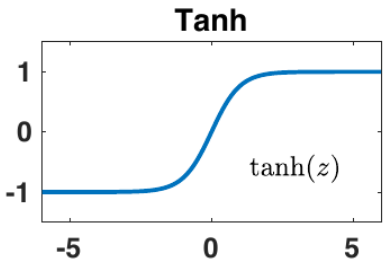


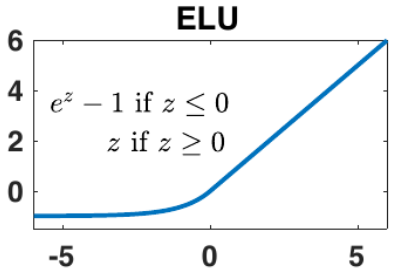


- Number of hidden layers: 2 => 51·(102+1) + 51·(51+1) + 51·(51+1) = 10 557 parameters ***w***, ***b***
- Possible nonlinearities: tanh or ELU (why?)









3. Choose the cost function and the optimizer

- The parameters θ of model *f* are found by minimizing the cost function *C* on the dataset (**X**, **Y**)
- A common choice for regression problems is mean squared error

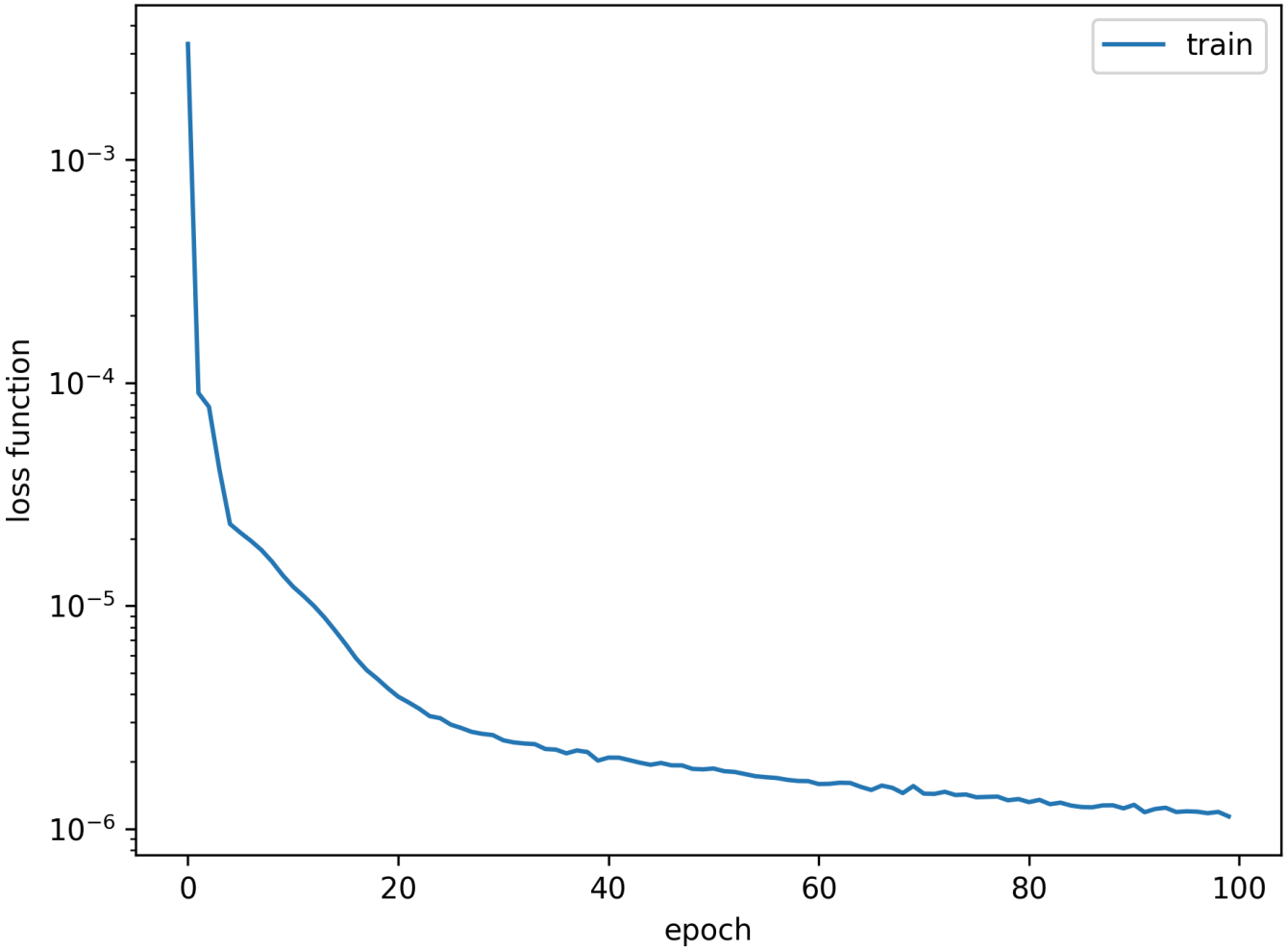
$$C(\mathbf{X}, \mathbf{Y}, \boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \left[\mathbf{y}_i - f_\theta(\mathbf{x}_i)\right]^2$$

- To optimize the parameters θ of the neural network a stochastic gradient descent-type algorithm with learning rate η is used (MB = mini-batch)

$$\mathbf{v}_t = \eta \nabla_\theta C(\mathbf{X}_{MB(t)}, \mathbf{Y}_{MB(t)}, \boldsymbol{\theta}_t)$$
$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_{t+1} - \mathbf{v}_t$$

- Backpropagation algorithm exploits the layered structure of of neural networks to make gradient computation efficient

4. Train the model



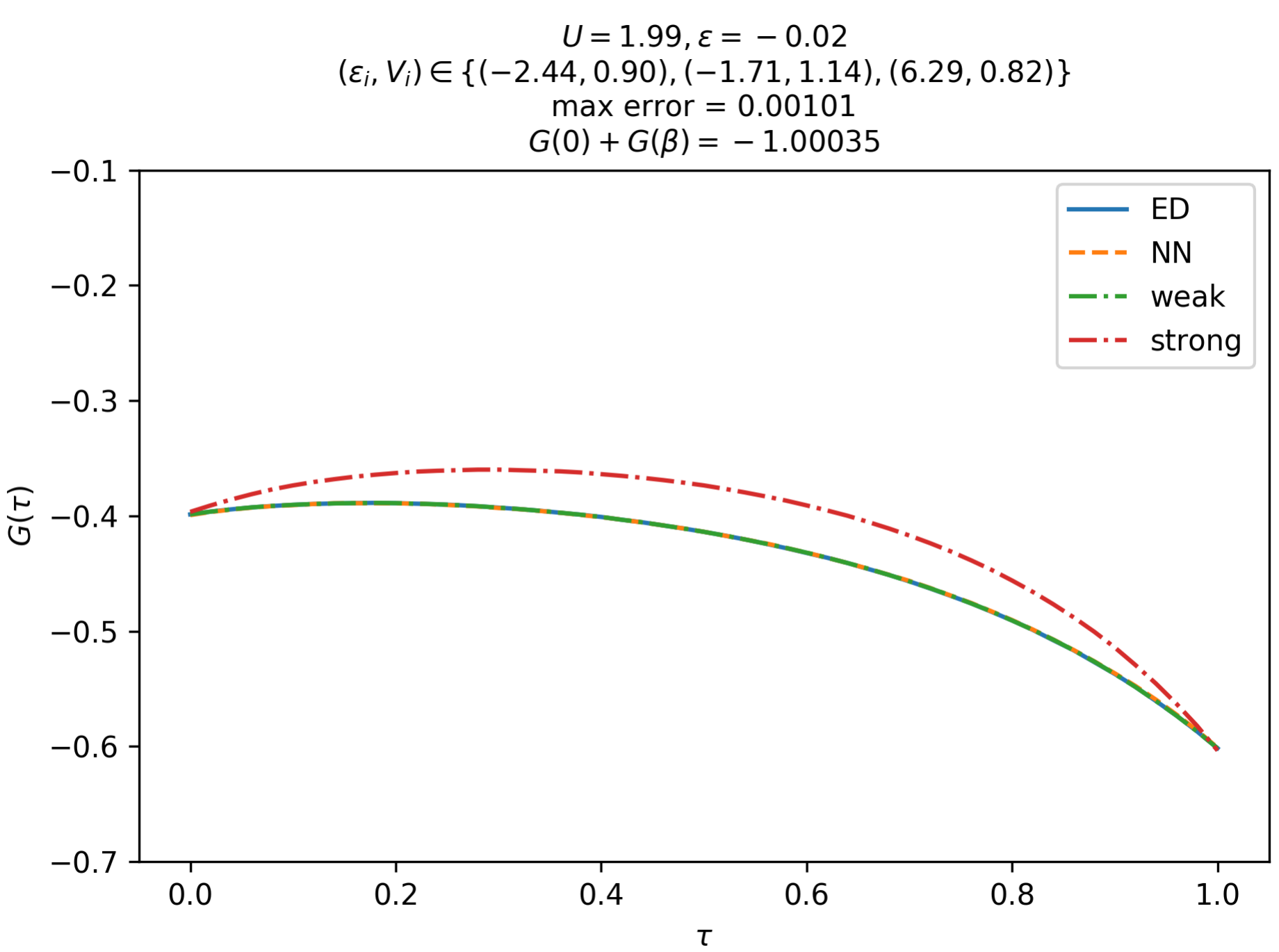
5. Evaluate and study the model performance on the test data

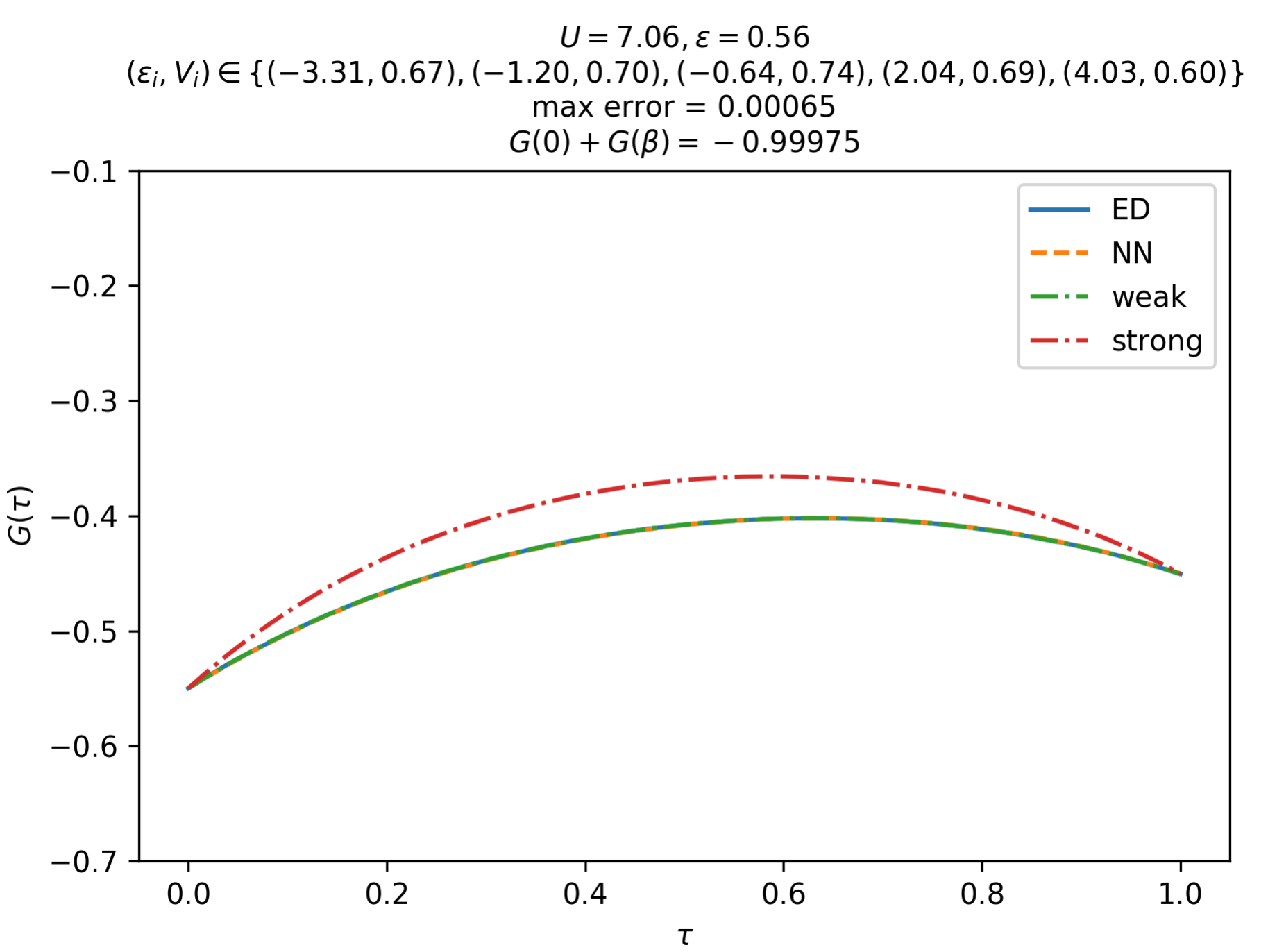
	Training Set (N = 14000)	Test Set (N = 6000)	Continuous Bath Set (N = 50)
Cost Function	$7.11 \cdot 10^{-7}$	$7.62 \cdot 10^{-7}$	$14.13 \cdot 10^{-7}$
Mean Absolute Error	0.000327	0.000334	0.000434
Max Error	0.005879	0.005880	0.004982
Max Boundary Condition Deviation	0.005173	0.004669	0.000414

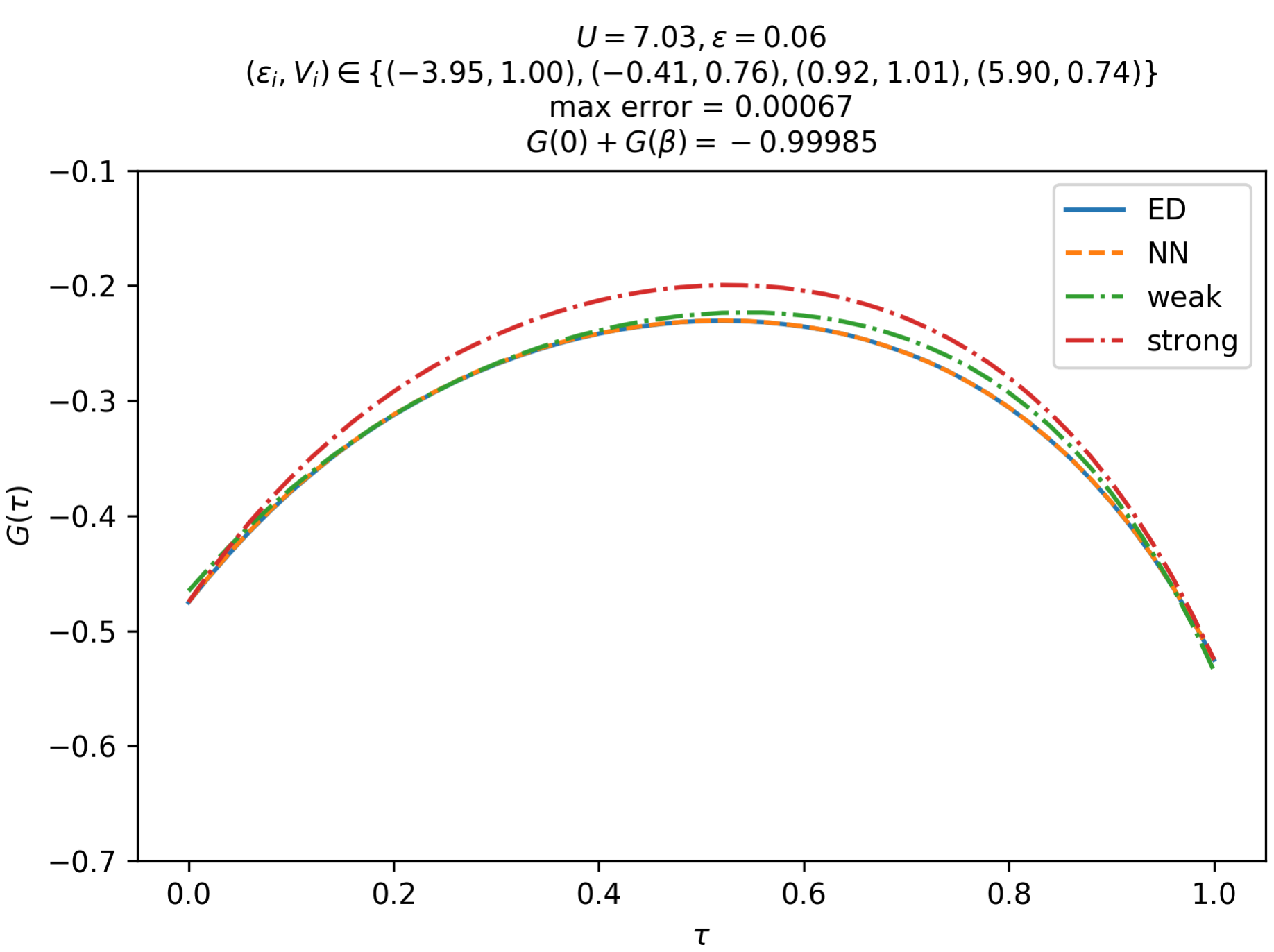
6. Adjust the hyperparameters – grid search

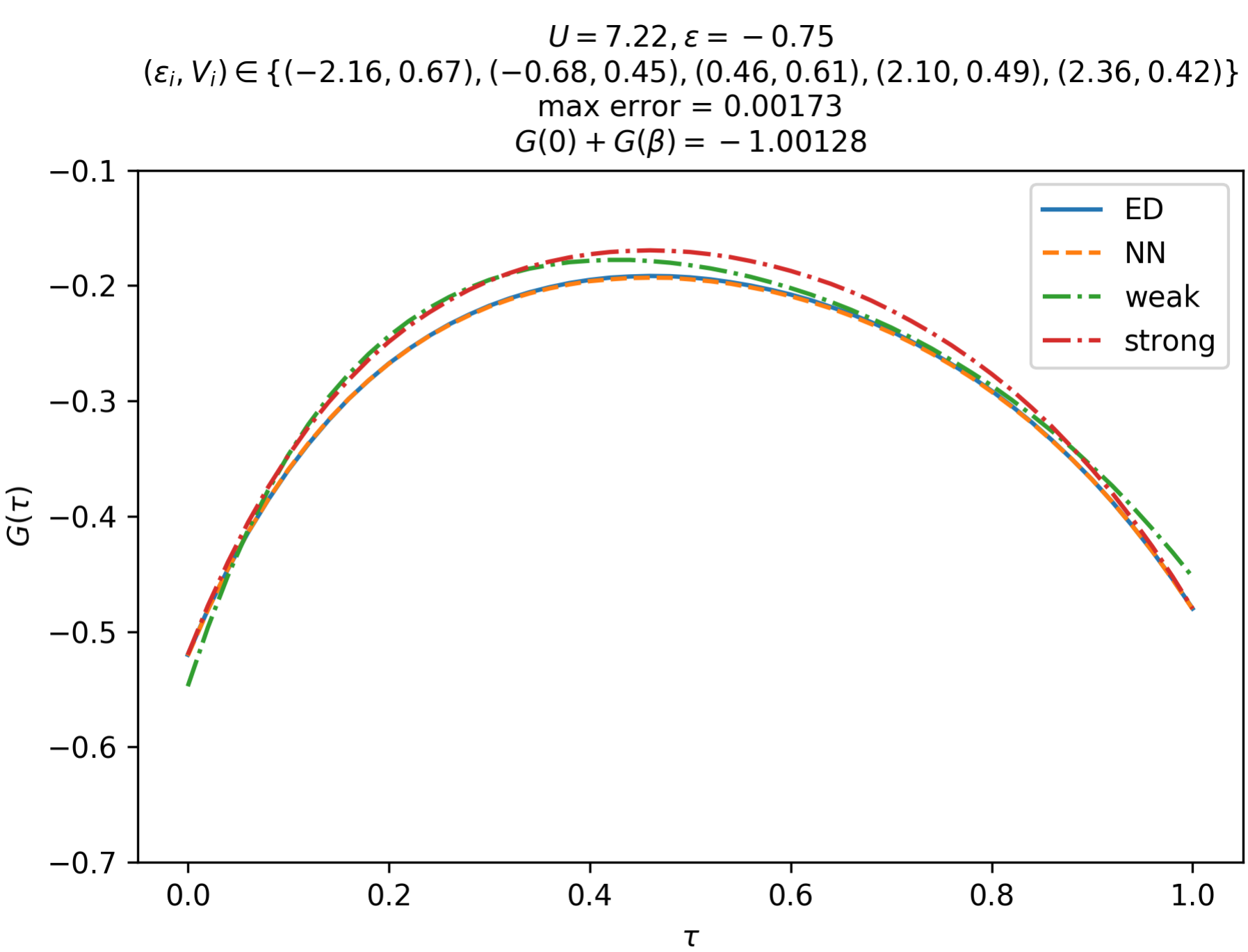
Cost Function			Batch Size	Optimizer	Learning Rate	Activation Function
0.000004	+-	0.000001	8	Nadam	0.0005	elu
0.000007	+-	0.000001	16	Adamax	0.0010	elu
0.000010	+-	0.000001	8	Nadam	0.0001	elu
0.000010	+-	0.000004	8	Adamax	0.0010	elu
0.000012	+-	0.000002	32	Adamax	0.0010	elu
0.000013	+-	0.000003	8	Adamax	0.0005	elu
0.000013	+-	0.000013	8	Nadam	0.0010	elu
0.000014	+-	0.000002	16	Adamax	0.0005	elu
0.000015	+-	0.000000	16	Nadam	0.0001	elu
0.000015	+-	0.000002	16	Nadam	0.0005	elu
0.000015	+-	0.000014	16	Nadam	0.0010	elu
0.000016	+-	0.000001	32	Adamax	0.0005	elu
0.000019	+-	0.000001	8	Adamax	0.0001	elu
0.000020	+-	0.000000	16	Adamax	0.0001	elu
0.000021	+-	0.000001	32	Nadam	0.0001	elu
0.000021	+-	0.000005	32	Nadam	0.0010	elu
0.000028	+-	0.000012	8	Nadam	0.0010	tanh
0.000029	+-	0.000004	8	Nadam	0.0005	tanh
0.000033	+-	0.000002	8	Adamax	0.0010	tanh
0.000036	+-	0.000022	32	Nadam	0.0005	elu

Test Set


$U = 1.99, \varepsilon = -0.02$
$(\varepsilon_i, V_i) \in \{(-2.44, 0.90), (-1.71, 1.14), (6.29, 0.82)\}$
max error = 0.00101
$G(0) + G(\beta) = -1.00035$
ED
NN
weak
strong
$G(\tau)$
τ
−0.1
−0.2
−0.3
−0.4
−0.5
−0.6
−0.7
0.0
0.2
0.4
0.6
0.8
1.0


$U = 7.06, \varepsilon = 0.56$
$(\varepsilon_i, V_i) \in \{(-3.31, 0.67), (-1.20, 0.70), (-0.64, 0.74), (2.04, 0.69), (4.03, 0.60)\}$
max error = 0.00065
$G(0) + G(\beta) = -0.99975$
ED
NN
weak
strong
$G(\tau)$
τ
−0.1
−0.2
−0.3
−0.4
−0.5
−0.6
−0.7
0.0
0.2
0.4
0.6
0.8
1.0

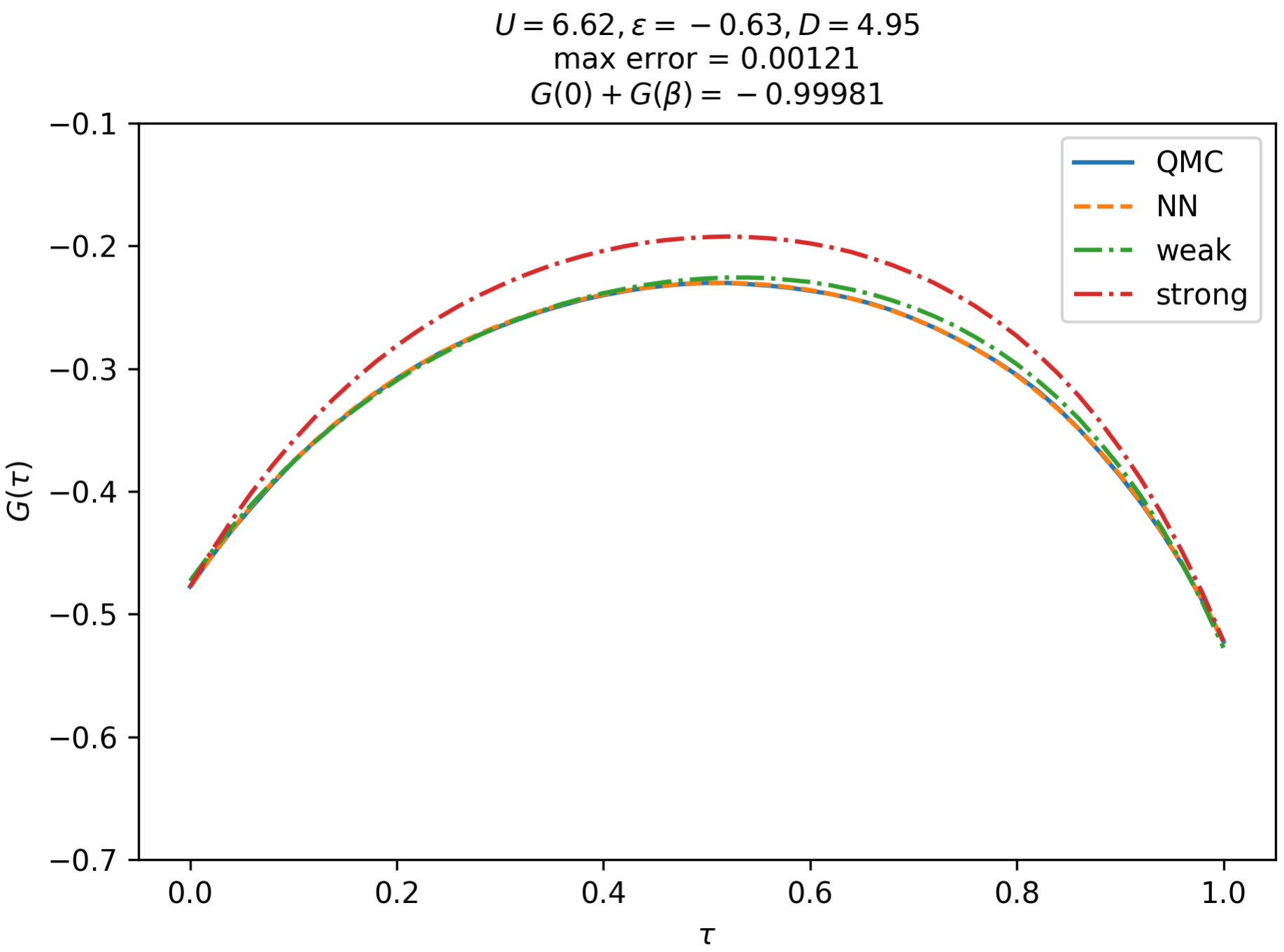

$U = 7.03, \varepsilon = 0.06$
$(\varepsilon_i, V_i) \in \{(-3.95, 1.00), (-0.41, 0.76), (0.92, 1.01), (5.90, 0.74)\}$
max error = 0.00067
$G(0) + G(\beta) = -0.99985$
ED
NN
weak
strong
$G(\tau)$
τ
−0.1
−0.2
−0.3
−0.4
−0.5
−0.6
−0.7
0.0
0.2
0.4
0.6
0.8
1.0

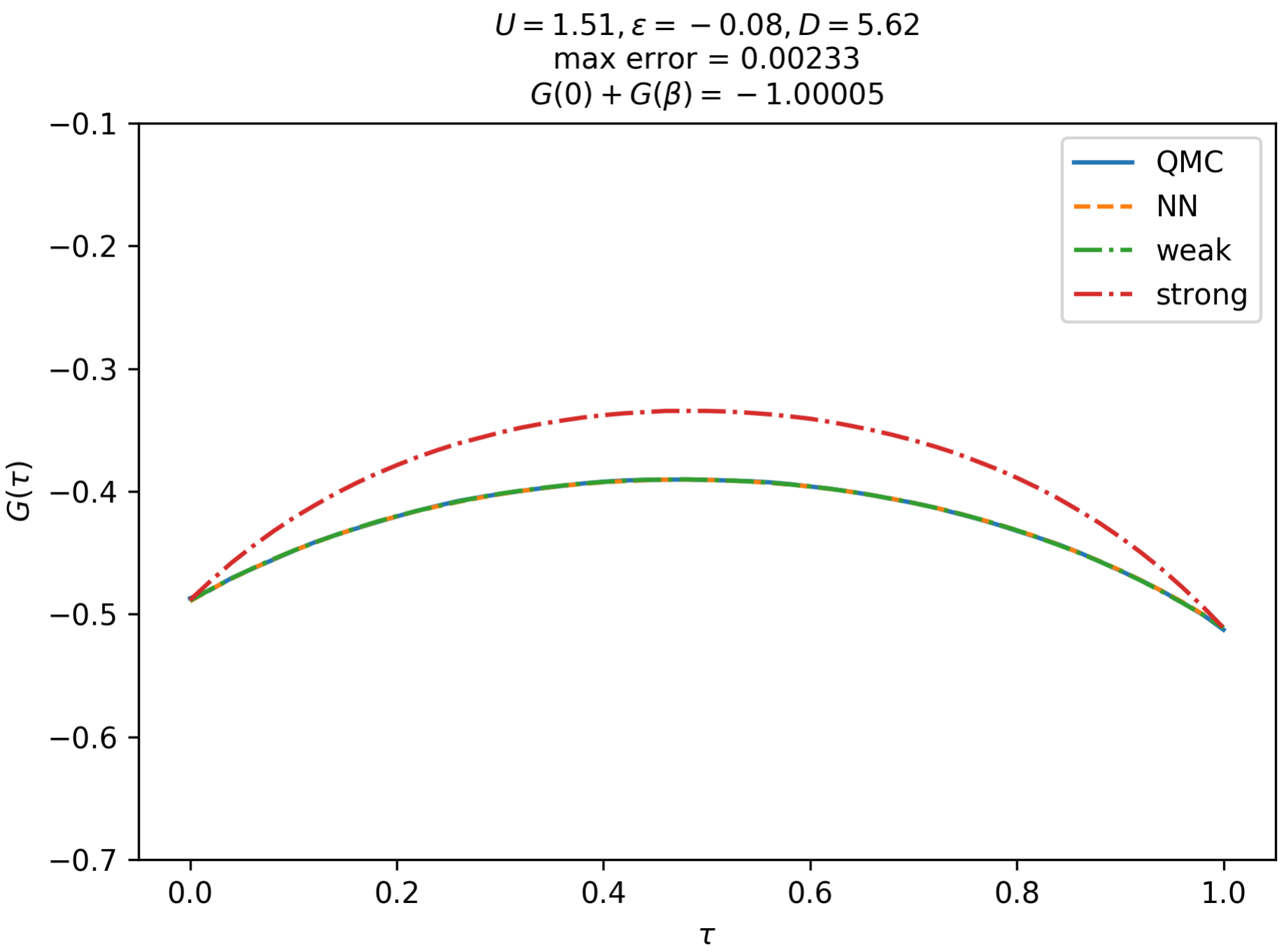

$U = 7.22, \varepsilon = -0.75$
$(\varepsilon_i, V_i) \in \{(-2.16, 0.67), (-0.68, 0.45), (0.46, 0.61), (2.10, 0.49), (2.36, 0.42)\}$
max error = 0.00173
$G(0) + G(\beta) = -1.00128$
ED
NN
weak
strong
$G(\tau)$
τ
−0.1
−0.2
−0.3
−0.4
−0.5
−0.6
−0.7
0.0
0.2
0.4
0.6
0.8
1.0

Continuous Bath Set

semicircular bath of width 2*D*

exact solution: CT-HYB-QMC (*TRIQS*)

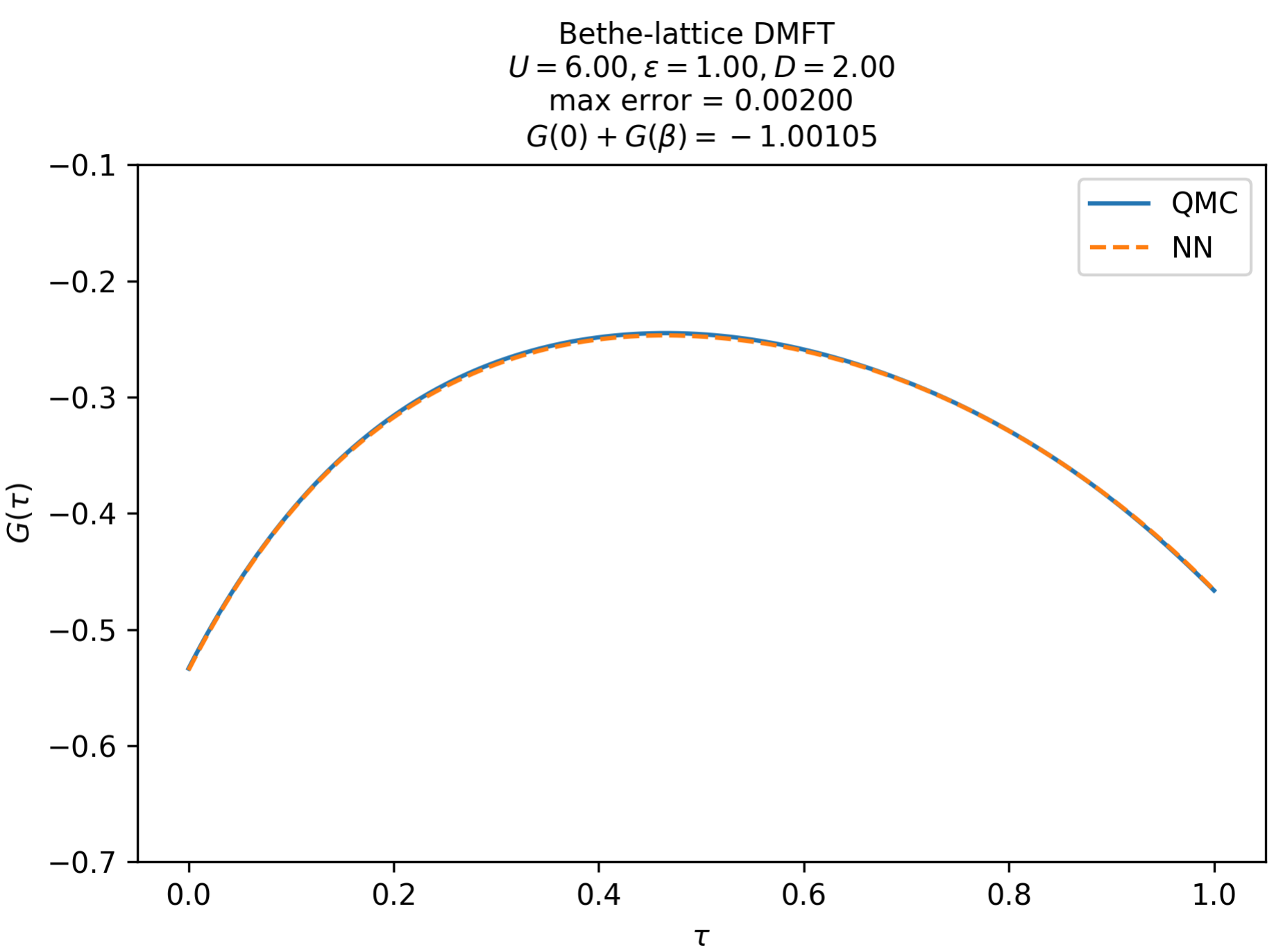

$U = 6.62, \varepsilon = -0.63, D = 4.95$
max error = 0.00121
$G(0) + G(\beta) = -0.99981$
QMC
NN
weak
strong
$G(\tau)$
τ
−0.1
−0.2
−0.3
−0.4
−0.5
−0.6
−0.7
0.0
0.2
0.4
0.6
0.8
1.0

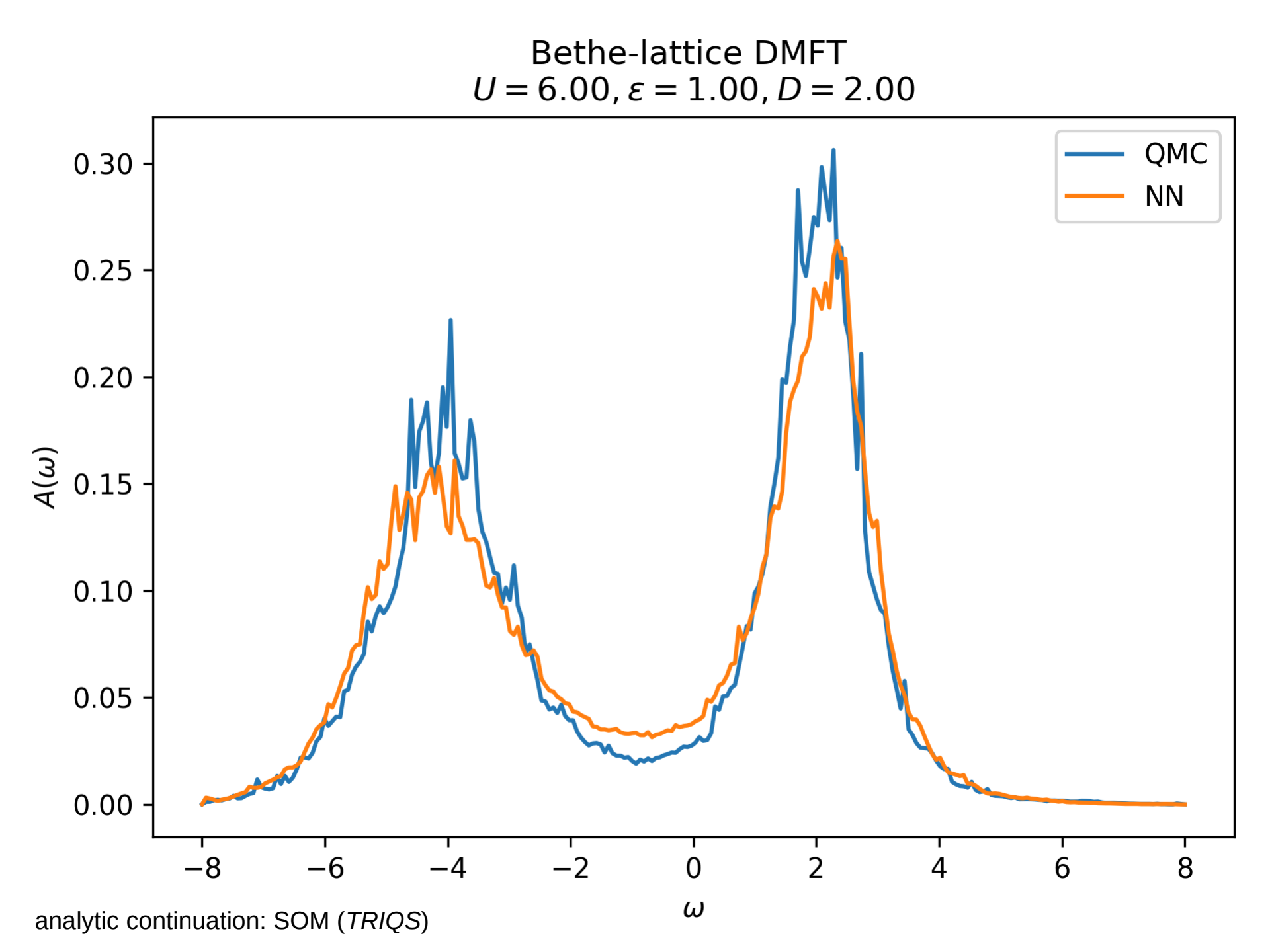

$U = 1.51, \varepsilon = -0.08, D = 5.62$
max error = 0.00233
$G(0) + G(\beta) = -1.00005$
QMC
NN
weak
strong
$G(\tau)$
τ
−0.1
−0.2
−0.3
−0.4
−0.5
−0.6
−0.7
0.0
0.2
0.4
0.6
0.8
1.0

Bethe-lattice
Dynamical Mean Field Theory

semicircular DOS of width $2D$

exact solution: CT-HYB-QMC (*TRIQS*)


Bethe-lattice DMFT
$U = 6.00, \varepsilon = 1.00, D = 2.00$
max error = 0.00200
$G(0) + G(\beta) = -1.00105$
QMC
NN
$G(\tau)$
τ
−0.1
−0.2
−0.3
−0.4
−0.5
−0.6
−0.7
0.0
0.2
0.4
0.6
0.8
1.0


Bethe-lattice DMFT
$U = 6.00, \varepsilon = 1.00, D = 2.00$
QMC
NN
0.30
0.25
0.20
0.15
0.10
0.05
0.00
$A(\omega)$
−8
−6
−4
−2
0
2
4
6
8
ω
analytic continuation: SOM (*TRIQS*)

Learning Green functions for $\beta = 20$

$\beta = 1$

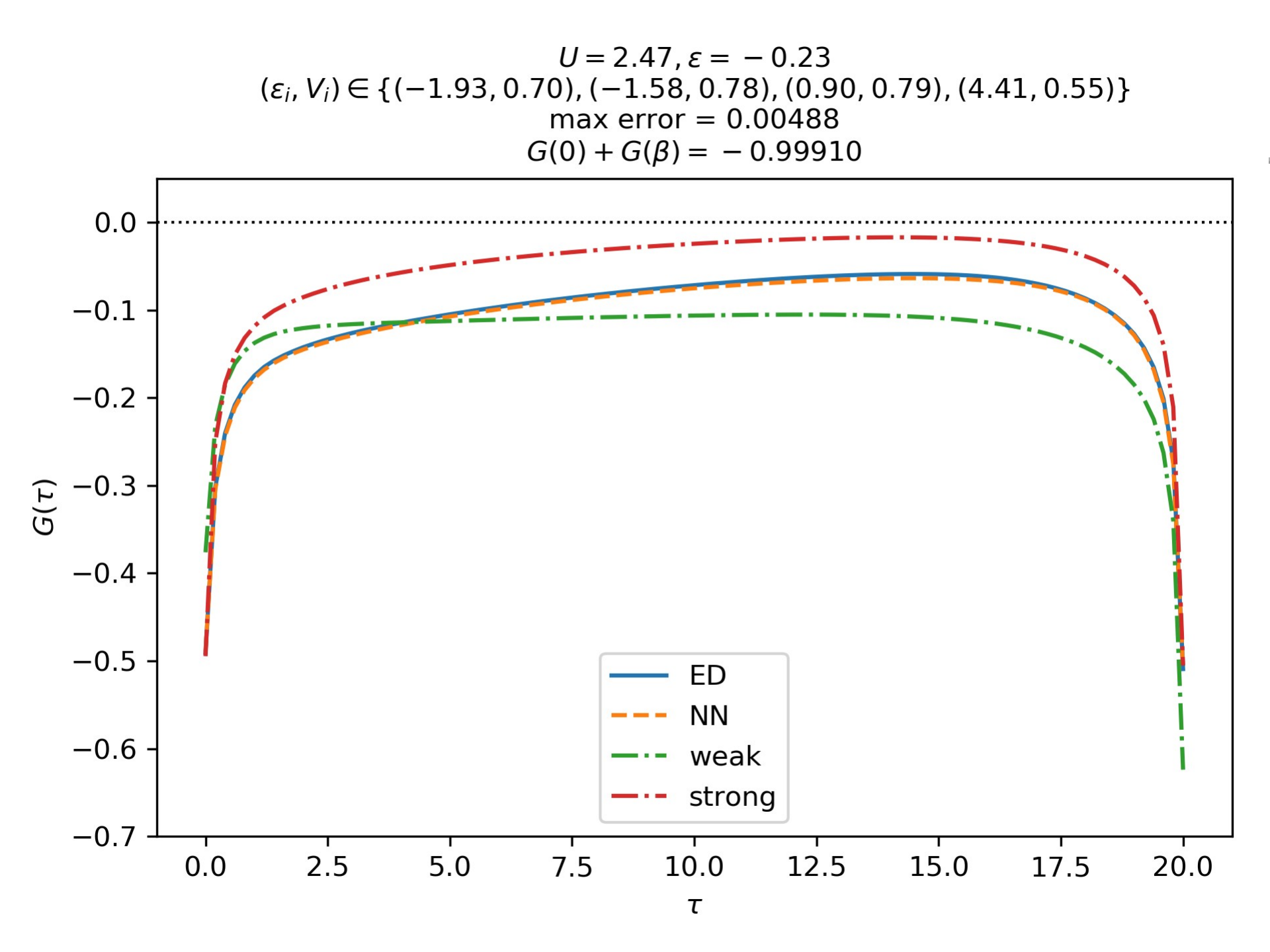
	Training Set (N = 14000)	Test Set (N = 6000)	Continuous Bath Set (N = 50)
Cost Function	$7.11 \cdot 10^{-7}$	$7.62 \cdot 10^{-7}$	$14.13 \cdot 10^{-7}$
Mean Absolute Error	0.000327	0.000334	0.000434
Max Error	0.005879	0.005880	0.004982
Max Boundary Condition Deviation	0.005173	0.004669	0.000414

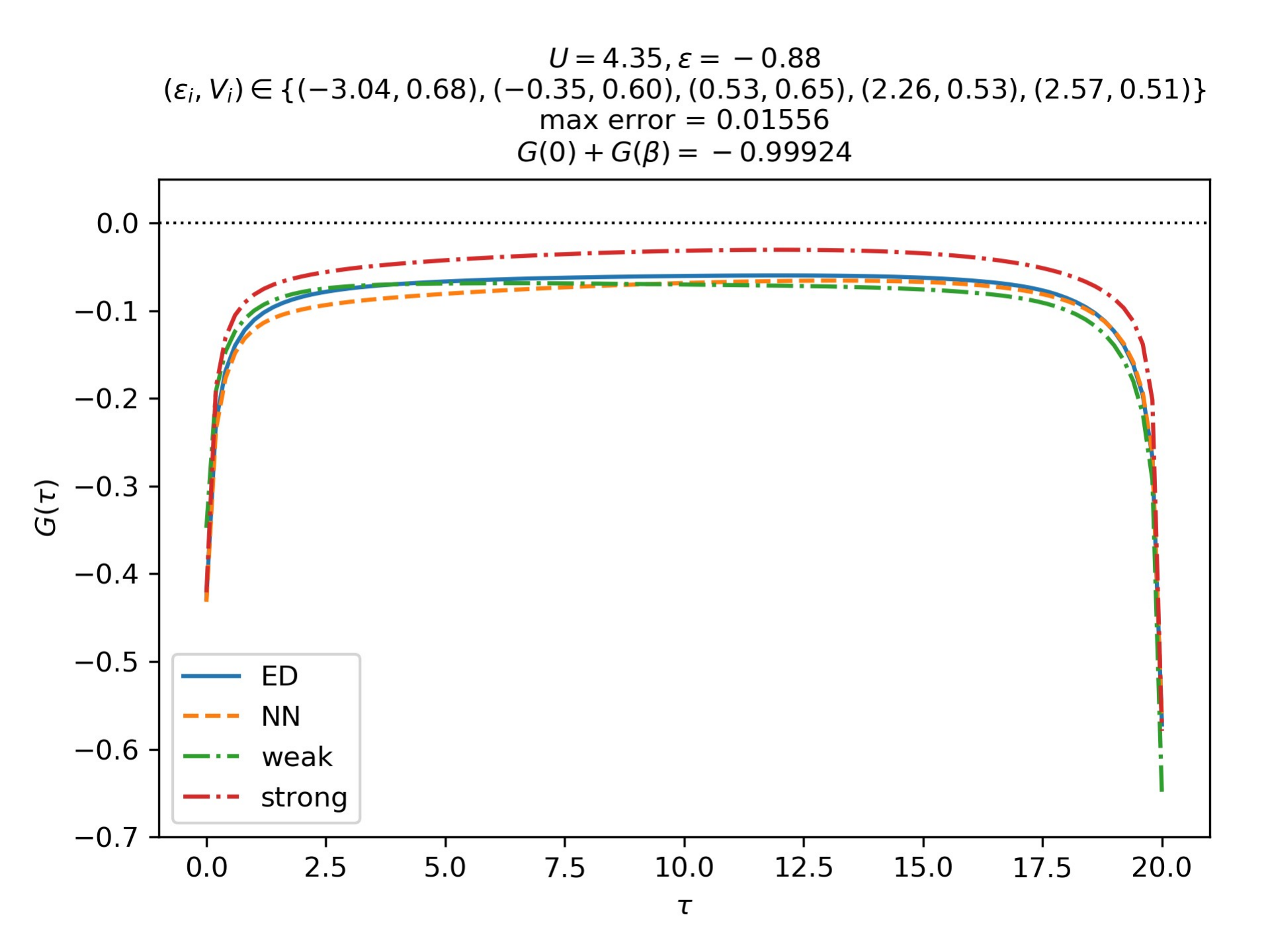
$\beta = 20$

	Training Set (N = 64000)	Test Set (N = 16000)	Continuous Bath Set (N = 50)
Cost Function	$5.83 \cdot 10^{-5}$	$5.88 \cdot 10^{-5}$	$16.33 \cdot 10^{-5}$
Mean Absolute Error	0.00292	0.00294	0.00432
Max Error	0.07042	0.05155	0.08857
Max Boundary Condition Deviation	0.02193	0.01419	0.00327

- $n_\tau = 101$
- non-equidistant τ-grid
- more hidden layers
- larger dataset

Test Set

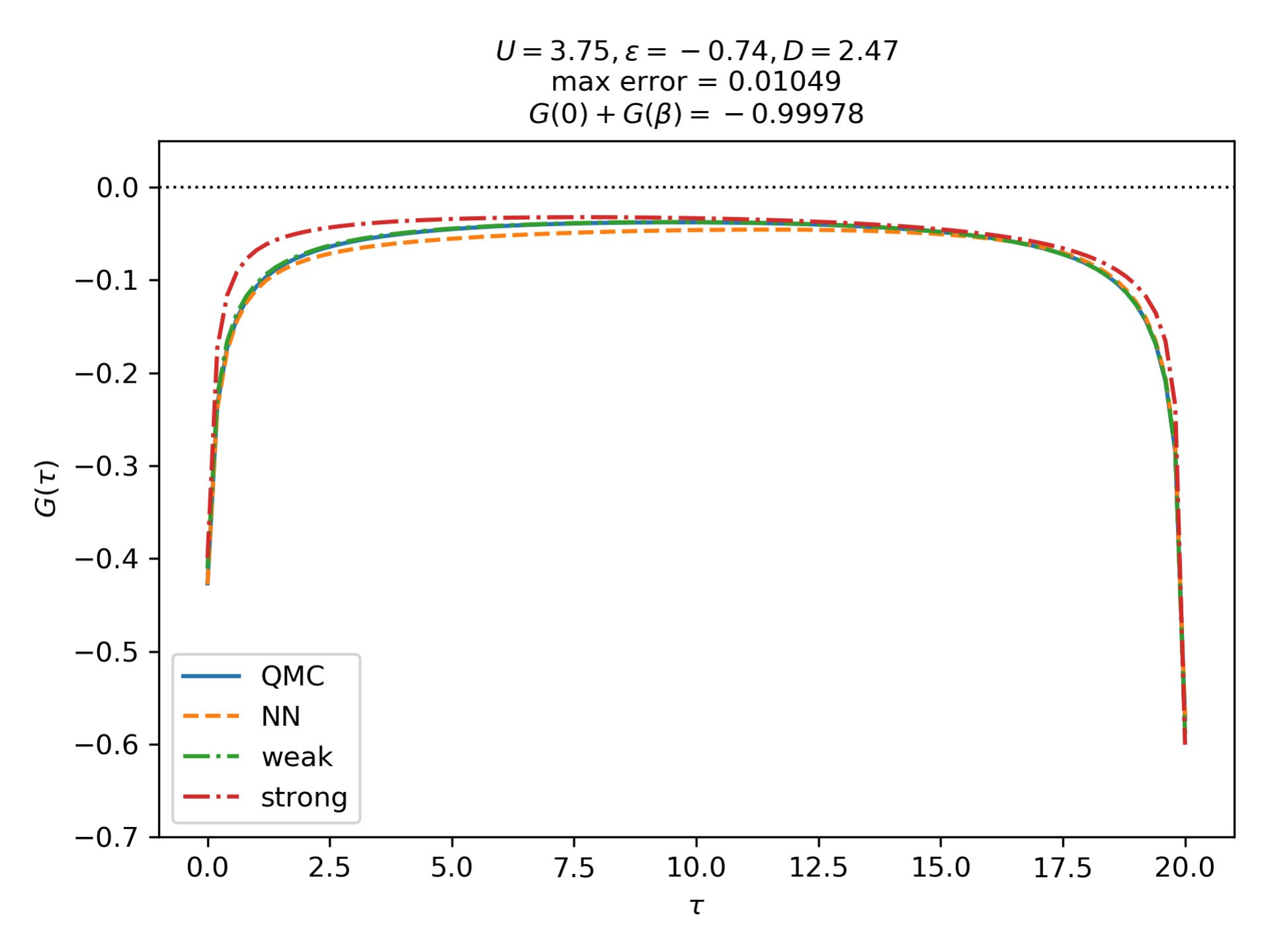

$U = 2.47, \varepsilon = -0.23$
$(\varepsilon_i, V_i) \in \{(-1.93, 0.70), (-1.58, 0.78), (0.90, 0.79), (4.41, 0.55)\}$
max error = 0.00488
$G(0) + G(\beta) = -0.99910$
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
−0.6
−0.7
$G(\tau)$
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
τ
ED
NN
weak
strong

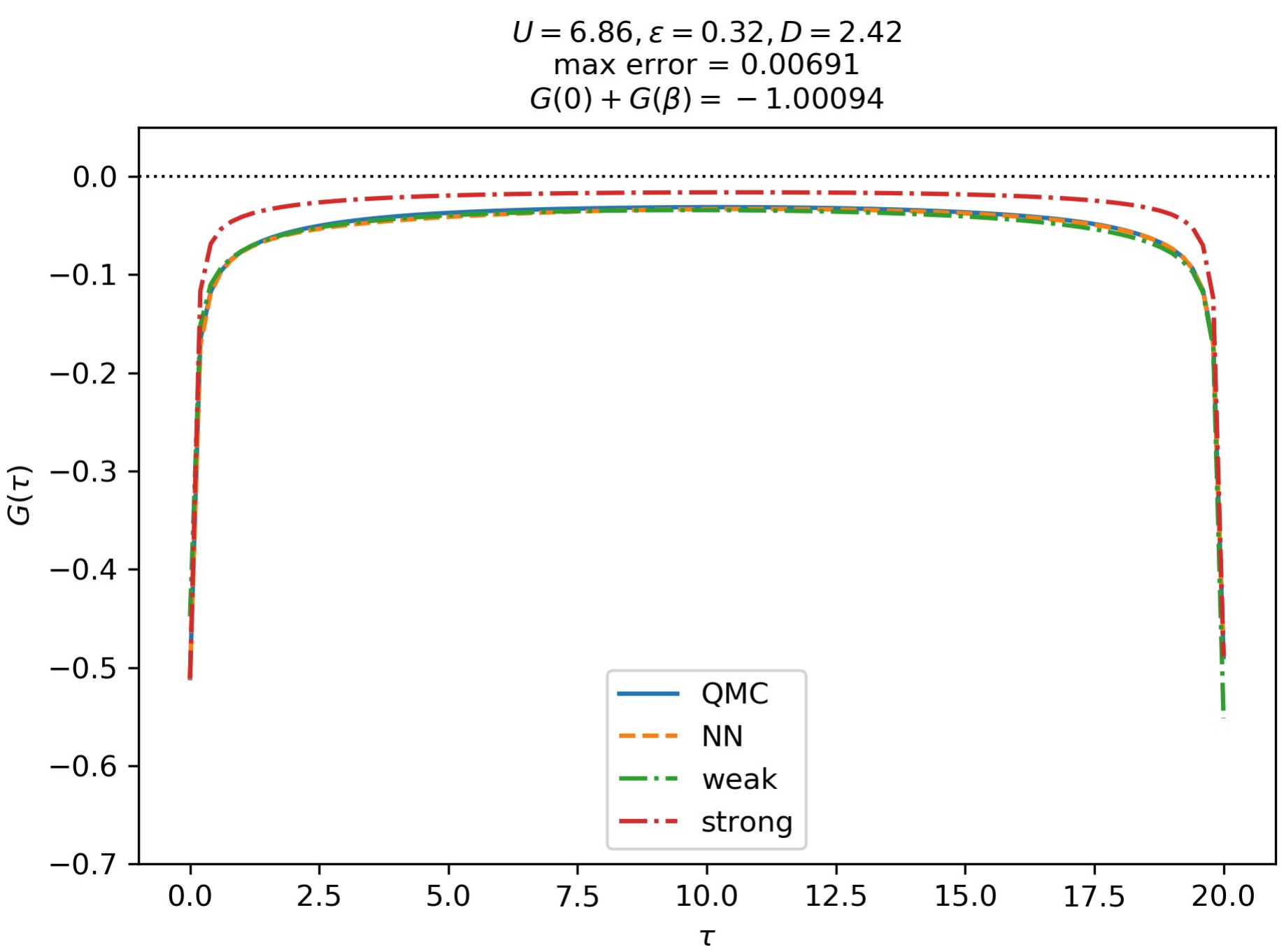

$U = 4.35, \varepsilon = -0.88$
$(\varepsilon_i, V_i) \in \{(-3.04, 0.68), (-0.35, 0.60), (0.53, 0.65), (2.26, 0.53), (2.57, 0.51)\}$
max error = 0.01556
$G(0) + G(\beta) = -0.99924$
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
−0.6
−0.7
$G(\tau)$
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
τ
ED
NN
weak
strong

Continuous Bath Set

semicircular bath of width $2D$

exact solution: CT-HYB-QMC (*TRIQS*)


$U = 3.75, \varepsilon = -0.74, D = 2.47$
max error = 0.01049
$G(0) + G(\beta) = -0.99978$
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
−0.6
−0.7
$G(\tau)$
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
τ
QMC
NN
weak
strong


$U = 6.86, \varepsilon = 0.32, D = 2.42$
max error = 0.00691
$G(0) + G(\beta) = -1.00094$
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
−0.6
−0.7
$G(\tau)$
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
τ
QMC
NN
weak
strong

Conclusion

- **Question 1*:** Can a neural network learn to predict exact AIM GFs from weak and strong coupling GFs?
- **Answer**: Yes for small β, unclear for large β (at least for a discrete bath)

- **Question 2*:** Can a neural network trained on discrete bath AIM GFs predict continuous bath GFs (from weak and strong coupling GFs)?
- **Answer:** Yes for small β, probably no for large β

- **Possible outlook:** use convolutional neural networks (CNN) to predict real-time Green functions $G(t, t')$

Thank you for your attention!