Neural Network Poisson–Boltzmann Electrostatics for Biomolecular Interactions

Joint work with Bo Li, Zhongming Wang, and Zhiwen Zhang

Informal Math and ML Seminar

Zunding Huang
February 10, 2025

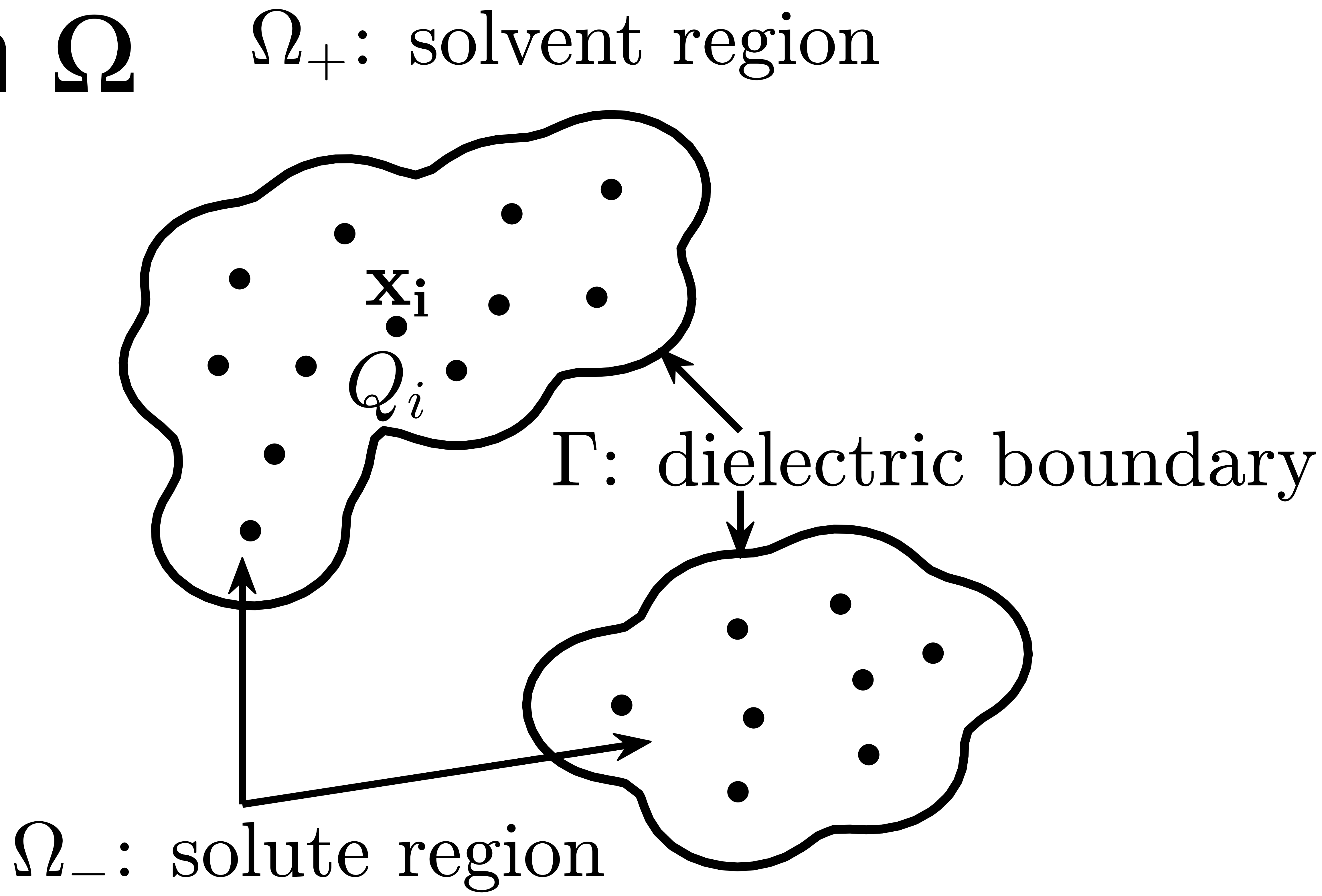
Outline

1. Poisson-Boltzmann Equation (PBE)
2. Neural Network Approach
3. Numerical Test
4. Applications to Solvation of Charged Molecules
5. Conclusion and Discussion

Poisson–Boltzmann Equation (PBE)

$$\nabla \cdot \varepsilon_\Gamma \nabla \phi - \chi_+ B'(\phi) = -f \text{ in } \Omega$$

- $\phi : \Omega \to \mathbb{R}$: electrostatic potential
- $B(\phi) = \beta^{-1} \sum_{j=1}^{M} c_j^\infty (e^{-\beta q_j \phi} - 1)$
- $\varepsilon_\Gamma = \begin{cases} \varepsilon_- & \text{in } \Omega_- \\ \varepsilon_+ & \text{in } \Omega_+ \end{cases}$
- $f : \Omega \to \mathbb{R}$: fixed charge density
- q_j : charge of an ion of jth species
- c_j^∞ : bulk concentration of jth ionic species



Minimize PB electrostatic energy functional $I_\Gamma[\phi]$

$$I_\Gamma[\phi] = \int_\Omega \left[\frac{\varepsilon_\Gamma}{2} |\nabla \phi|^2 - f\phi + \chi_+ B(\phi) \right] dx$$

over $H^1_g(\Omega) = \{\phi \in H^1(\Omega) : \phi = g$ **on** $\partial\Omega\}$

ϕ_Γ: unique minimizer, is also the weak solution to PBE

Penalized PB energy functional $I_{\Gamma,\lambda}[\phi] : H^1(\Omega) \to \mathbb{R} \cup \{+\infty\}$

$$I_{\Gamma,\lambda}[\phi] = \int_\Omega \left[\frac{\varepsilon_\Gamma}{2} |\nabla \phi|^2 - f\phi + \chi_+ B(\phi) \right] dx + \lambda \int_{\partial\Omega} (\phi - g)^2 \, dS$$

- λ : large penalty coefficient

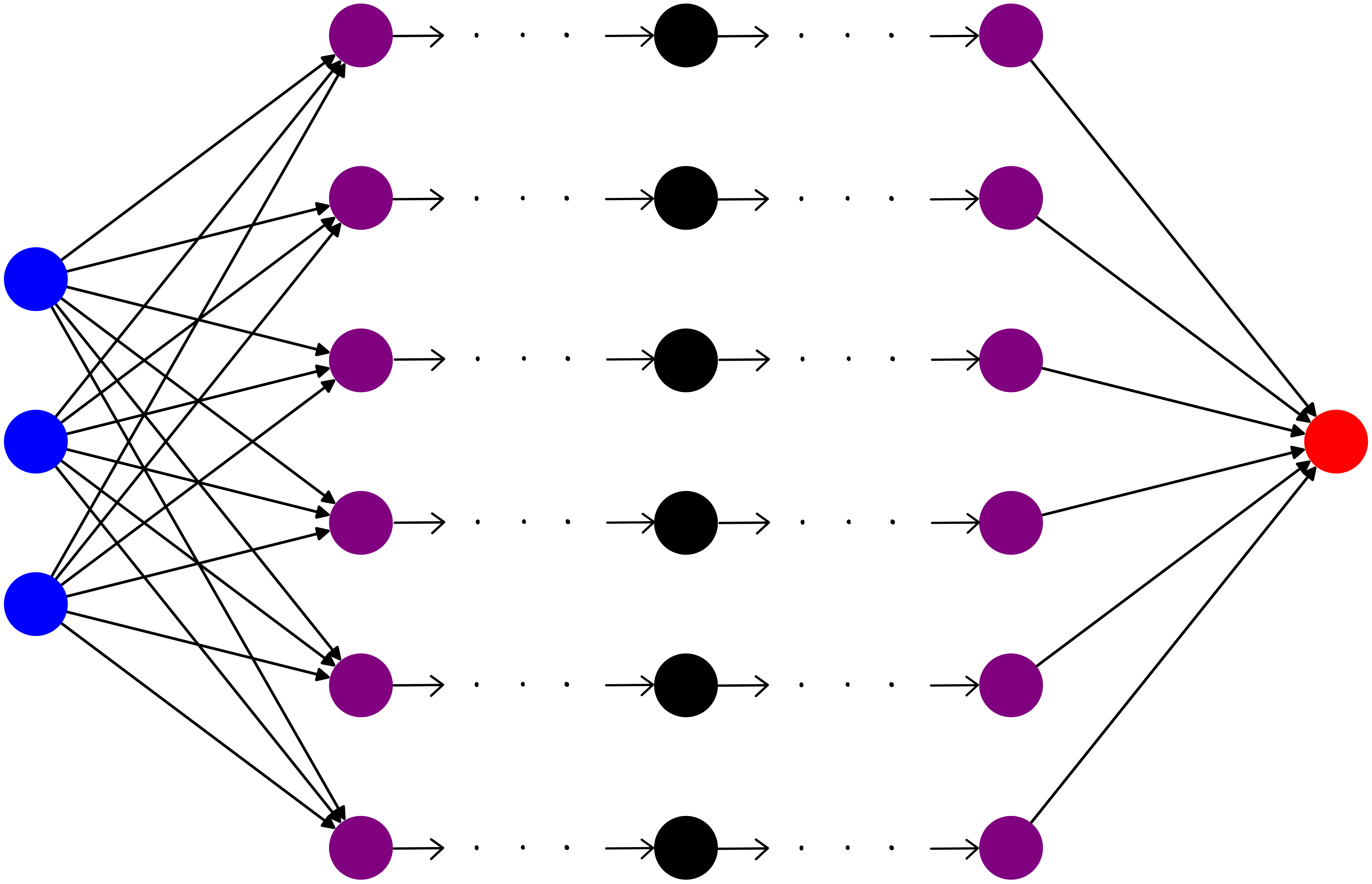
Theorem

- For any $\lambda > 0$, there exists a unique $\phi_{\Gamma,\lambda} \in H^1(\Omega)$, such that

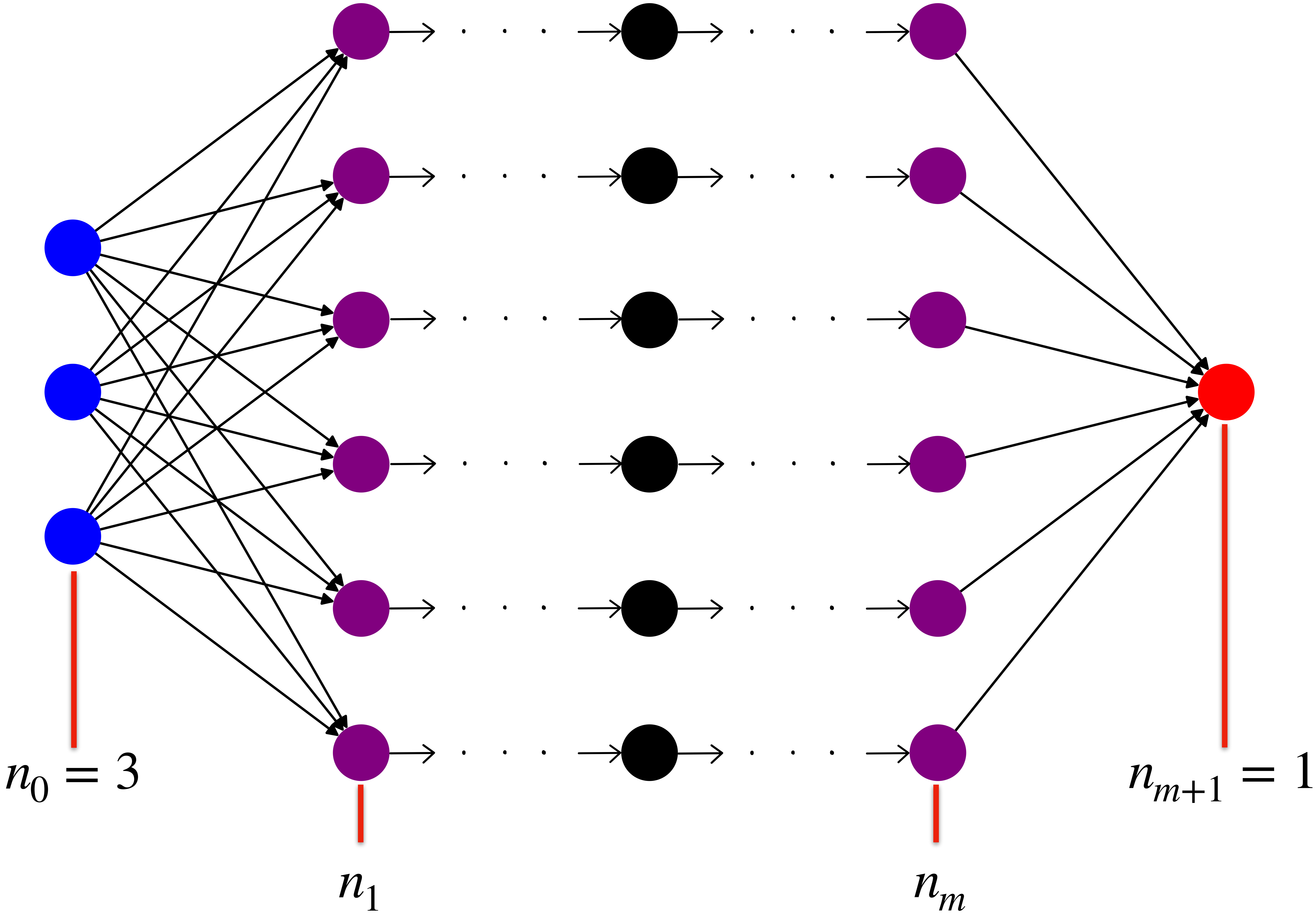
$$I_{\Gamma,\lambda}[\phi_{\Gamma,\lambda}] = \min_{\phi \in H^1(\Omega)} I_{\Gamma,\lambda}[\phi];$$

- As $\lambda \to \infty$, $\phi_{\Gamma,\lambda} \to \phi_\Gamma$ in $H^1(\Omega)$ and $\min_{\phi \in H^1(\Omega)} I_{\Gamma,\lambda}[\phi] \to \min_{\phi \in H^1_g(\Omega)} I_\Gamma[\phi]$.

Neural Network Approach



NN architecture $S = [n_0, n_1, \ldots, n_{m+1}]$


$n_0 = 3$
n_1
n_m
$n_{m+1} = 1$

Neural Network

NN architecture $S = [n_0, n_1, \ldots, n_{m+1}]$

NN function $\psi_\theta(\mathbf{x}) = T_{m+1} \circ \sigma \circ \cdots \circ \sigma \circ T_1(\mathbf{x}) = T_{m+1}(\sigma(\cdots(\sigma(T_1(\mathbf{x})))))$

Affine function $T_i(\mathbf{x}) = W_i\mathbf{x} + b_i$

Activation function $\sigma(s) = \dfrac{1}{1 + e^{-s}}$

Loss Function

Neural Network penalized PB functional

$$I_{\Gamma,\lambda}[\phi] = \int_\Omega \left[\frac{\varepsilon_\Gamma}{2} |\nabla \phi|^2 - f\phi + \chi_+ B(\phi) \right] dx + \lambda \int_{\partial\Omega} (\phi - g)^2 \, dS$$

$$\psi_\theta(\mathbf{x}) = T_{m+1} \circ \sigma \circ \cdots \circ \sigma \circ T_1(\mathbf{x}) = T_{m+1}(\sigma(\cdots(\sigma(T_1(\mathbf{x})))))$$

$$\Longrightarrow$$

$$\begin{aligned} J_{\Gamma,\lambda}[\theta] &= I_{\Gamma,\lambda}[\psi_\theta] \\ &= \int_\Omega \left[\frac{\varepsilon_\Gamma}{2} |\nabla \psi_\theta|^2 - f\psi_\theta + \chi_+ B(\psi_\theta) \right] dx + \lambda \int_{\partial\Omega} (\psi_\theta - g)^2 \, dS \end{aligned}$$

$I_{\Gamma,\lambda}[\phi]$: convex

$J_{\Gamma,\lambda}[\theta]$: nonconvex

Neural Network penalized PB functional

$$J_{\Gamma,\lambda}[\theta] = \int_{\Omega} \left[\frac{\varepsilon_{\Gamma}}{2} |\nabla \psi_\theta|^2 - f\psi_\theta + \chi_+ B(\psi_\theta) \right] \, dx + \lambda \int_{\partial\Omega} (\psi_\theta - g)^2 \, dS$$

MC Loss Function

Given $N, N_b, \{x_i\}_{i=1}^{N} \in \Omega$ and $\{y_j\}_{j=1}^{N_b} \in \partial\Omega$, define

$$\hat{J}_{\Gamma,\lambda}[\theta] = \frac{\mathrm{vol}(\Omega)}{N} \left[\sum_{i=1}^{N} \left(\frac{\varepsilon_{\Gamma}(x_i)}{2} |\nabla \psi_\theta(x_i)|^2 - f\psi_\theta(x_i) \right) + \sum_{i=1, x_i \in \Omega_+}^{N} B(\psi_\theta(x_i)) \right]$$

$$+ \lambda \frac{\mathrm{area}(\partial\Omega)}{N_b} \left[\sum_{j=1}^{N_b} (\psi_\theta(y_j) - g(y_j))^2 \right]$$

Algorithm

Input

- Model parameters: Ω, Γ, ε_-, ε_+, the function B, f, g, and the penalty coefficient λ.
- NN hyper-parameters: architecture S, activation function σ, learning rate η, number of sample points N and N_b.

Initialization

- Initialize all the neural network weights.

for $k = 1$ to M_1 **do**

- Generate N random sample points $x_1, \ldots, x_N \in \Omega$ and N_b random sample points $y_1, \ldots, y_{N_b} \in \partial\Omega$, all uniformly and independently.
- Formulate $\hat{J}^{(k)}_{\Gamma,\lambda}[\theta]$.

  for $j = 1$ to M_2 **do**

  - Compute the gradient $\nabla_\theta \hat{J}^{(k)}_{\Gamma,\lambda}$.
  - Use the ADAM optimizer to minimize $\hat{J}^{(k)}_{\Gamma,\lambda}$ and update the weights θ.

  end for

end for

Output

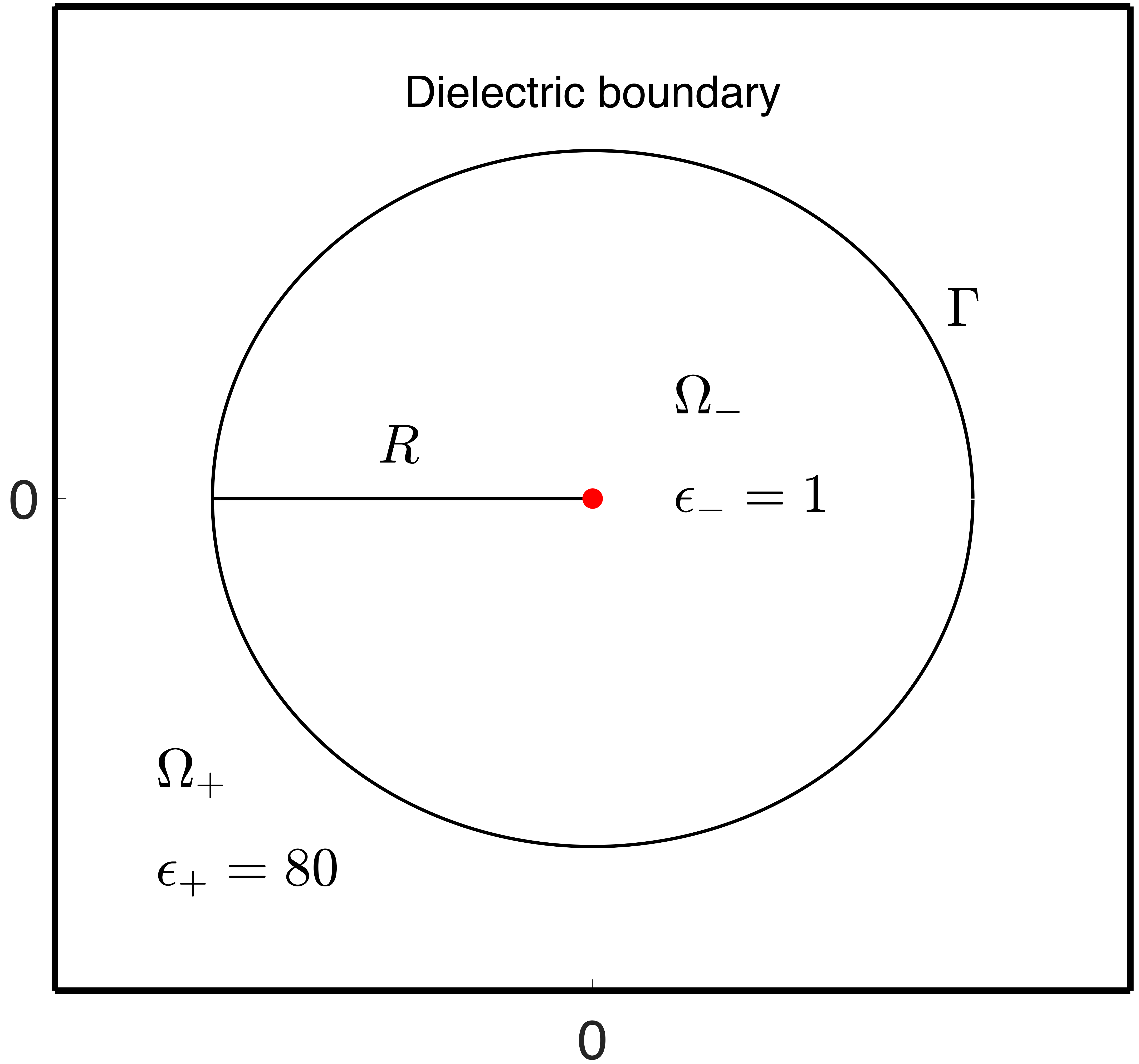
Numerical Test

A model system

$$I_\Gamma[\phi] = \int_\Omega \left[\frac{\varepsilon_\Gamma}{2} |\nabla \phi|^2 - f\phi + \chi_+ B(\phi) \right] dx$$

Set

- $\Omega = (-L, L)^3$ for some $L > 0$;
- $\Gamma = \{x \in \mathbb{R}^3 : |x| = R\}$ for $R \in (0,L)$;
- $\Omega_- = \{x \in \mathbb{R}^3 : |x| < R\}$;
- $\Omega_+ = \Omega \backslash (\Gamma \cup \Omega_-)$;
- $\varepsilon_- = 1,\ \varepsilon_+ = 80.$



A model system

$$I_\Gamma[\phi] = \int_\Omega \left[\frac{\varepsilon_\Gamma}{2} |\nabla \phi|^2 - f\phi + \chi_+ B(\phi) \right] dx$$

$$B(s) = \cosh(s) - 1$$

$$f(x) = \begin{cases} f_0 & \text{if } x \in \Omega_- \\ \sinh\left(\dfrac{f_0 R^3 \exp(\alpha R)}{3\varepsilon_+(\alpha R + 1)} \cdot \dfrac{\exp(-\alpha |x|)}{|x|} \right) - \dfrac{f_0 R^3 \exp(\alpha R)}{3\varepsilon_+(\alpha R + 1)} \cdot \dfrac{\exp(-\alpha |x|)}{|x|} & \text{if } x \in \Omega_+ \end{cases}$$

$$g(x) = \frac{f_0 R^3 \exp(\alpha R)}{3\varepsilon_+(\alpha R + 1)} \cdot \frac{\exp(-\alpha |x|)}{|x|} \quad \text{if } x \in \partial\Omega$$

$$\phi_\Gamma(x) = \begin{cases} -\dfrac{f_0}{6\varepsilon_-}|x|^2 + \dfrac{f_0}{6\varepsilon_-}R^2 + \dfrac{f_0}{3\varepsilon_+(\alpha R + 1)}R^2 & \text{if } x \in \Omega_- \\ \dfrac{f_0 R^3 \exp(\alpha R)}{3\varepsilon_+(\alpha R + 1)} \cdot \dfrac{\exp(-\alpha |x|)}{|x|} & \text{if } x \in \Omega_+ \end{cases}$$

A model system

$$I_\Gamma[\phi] = \int_\Omega \left[\frac{\varepsilon_\Gamma}{2} |\nabla \phi|^2 - f\phi + \chi_+ B(\phi) \right] dx$$

Set

- $L = 1;$
- $R = 0.75;$
- $f_0 = 10.$

$$I_\Gamma[\phi_\Gamma] \approx -3.6172$$

Define relative error

- Err-P $= \dfrac{\|\Phi - \phi_\Gamma\|_2}{\|\phi_\Gamma\|_2}$ and Err-E $= \dfrac{|I_\Gamma[\Phi] - I_\Gamma[\phi_\Gamma]|}{|I_\Gamma[\phi_\Gamma]|}$.

Test on the penalty method

Set

- network architecture = [3, 30, 20, 15, 10, 1];
- batch size = 6144;
- learning rate = 1e-02.

Compare

- penalty coefficient λ = 25, 100, 250.

Steps	$\lambda = 25$		$\lambda = 100$		$\lambda = 250$	
	Err-E	Err-P	Err-E	Err-P	Err-E	Err-P
100,000	4.68%	7.36%	5.61%	5.51%	8.68%	4.39%
200,000	3.44%	7.38%	5.33%	3.50%	5.38%	2.89%
300,000	1.68%	6.88%	3.80%	2.34%	4.18%	1.96%

Test on the penalty method

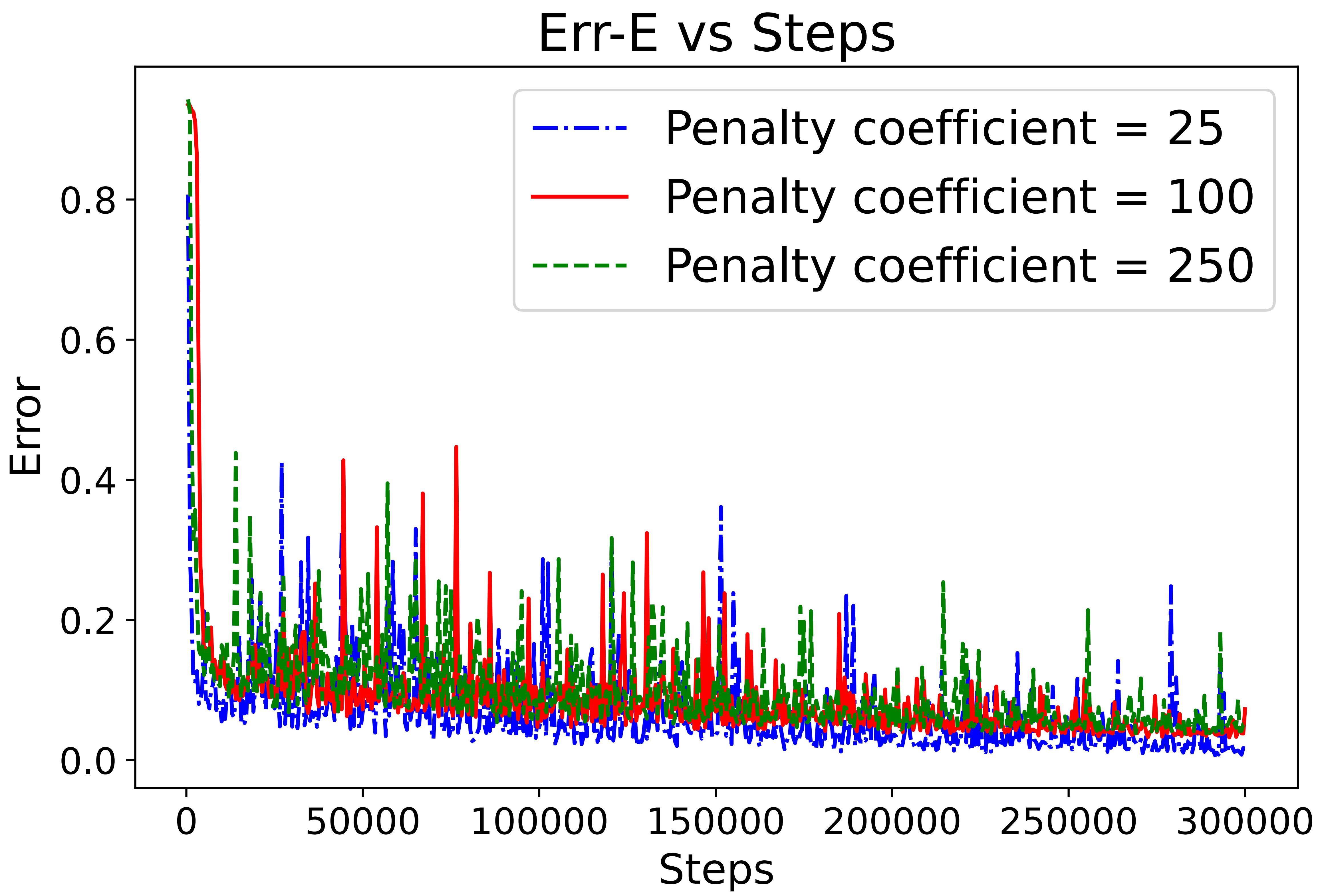
Set

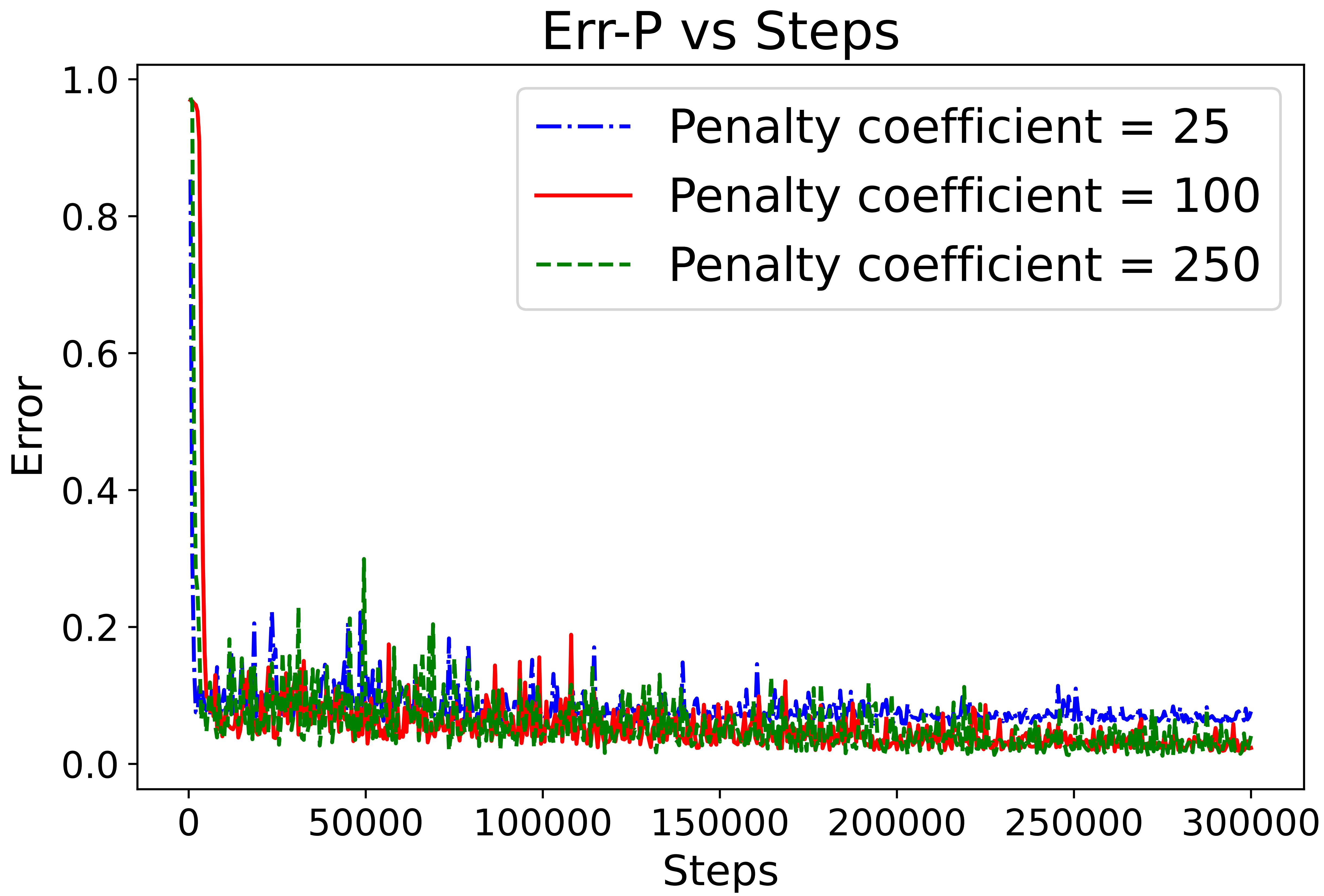
- network architecture = [3, 30, 20, 15, 10, 1];
- batch size = 6144;
- learning rate = 1e-02.

Compare

- penalty coefficient λ = 25, 100, 250.

- As $\lambda \to \infty$, $\phi_{\Gamma,\lambda} \to \phi_\Gamma$ in $H^1(\Omega)$ and $\min\limits_{\phi \in H^1(\Omega)} I_{\Gamma,\lambda}[\phi] \to \min\limits_{\phi \in H^1_g(\Omega)} I_\Gamma[\phi]$.


Err-E vs Steps
Penalty coefficient = 25
Penalty coefficient = 100
Penalty coefficient = 250
Error
Steps
0.0
0.2
0.4
0.6
0.8
0
50000
100000
150000
200000
250000
300000


Err-P vs Steps
Penalty coefficient = 25
Penalty coefficient = 100
Penalty coefficient = 250
Error
Steps
0.0
0.2
0.4
0.6
0.8
1.0
0
50000
100000
150000
200000
250000
300000

Convergence test: learning rate, batch size, and network architecture

Test on learning rate and batch size

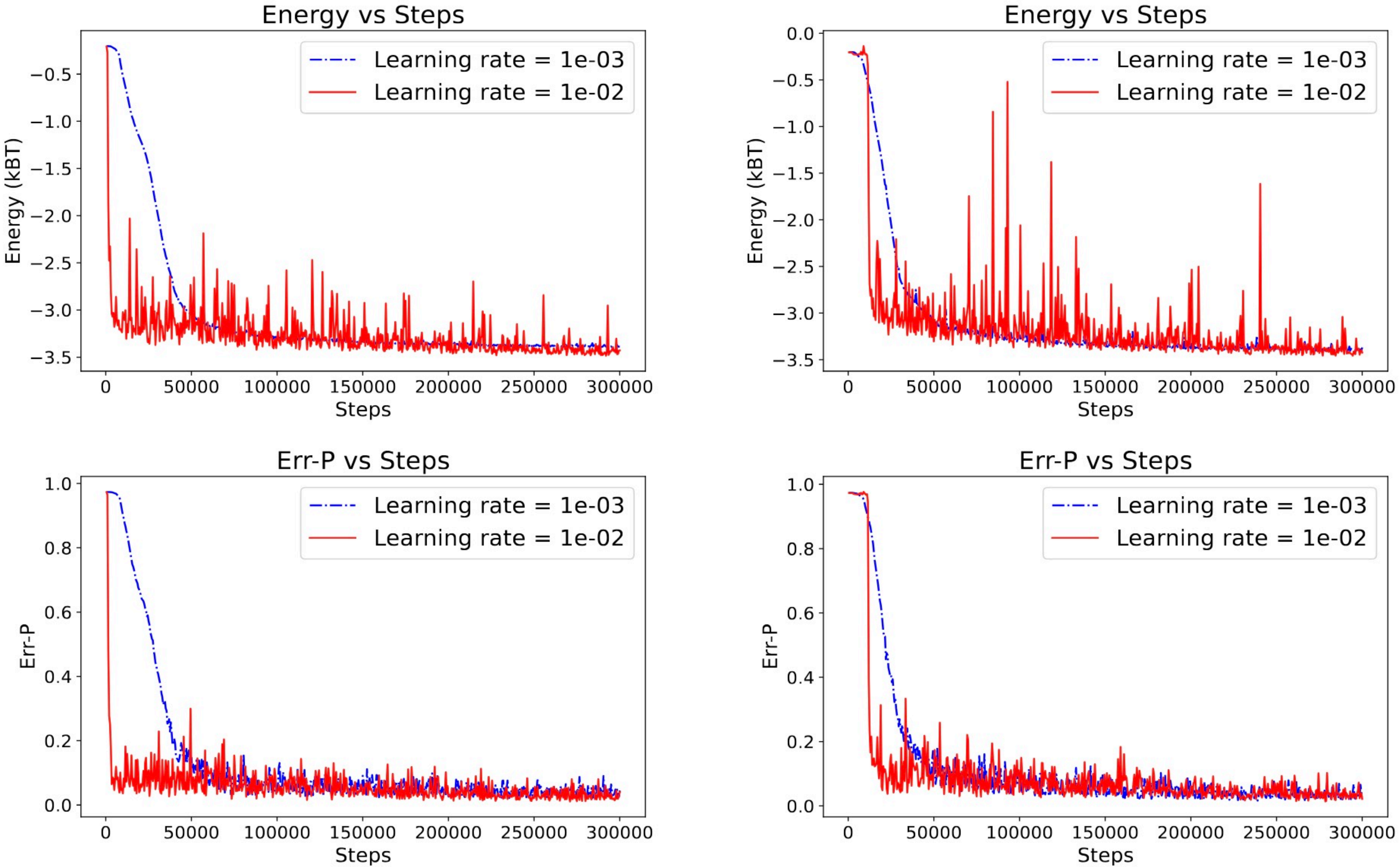
Set

- network architecture = [3, 30, 20, 15, 10, 1];
- penalty coefficient λ = 250.

Compare

- batch size = 6144, 3072;
- learning rate = 1e-03, 1e-02.

Batch size	Learning Rate	Err-E	Err-P
6144	1e-03	6.35%	3.56%
	1e-02	4.18%	1.96%
3072	1e-03	6.18%	4.16%
	1e-02	5.79%	3.68%



Energy vs Steps
Energy (kBT)
Steps
Learning rate = 1e-03
Learning rate = 1e-02
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
−3.5
0
50000
100000
150000
200000
250000
300000
Energy vs Steps
Energy (kBT)
Steps
Learning rate = 1e-03
Learning rate = 1e-02
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
−3.5
0
50000
100000
150000
200000
250000
300000
Err-P vs Steps
Err-P
Steps
Learning rate = 1e-03
Learning rate = 1e-02
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000
Err-P vs Steps
Err-P
Steps
Learning rate = 1e-03
Learning rate = 1e-02
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000
250000
300000

Test on learning rate and network architecture

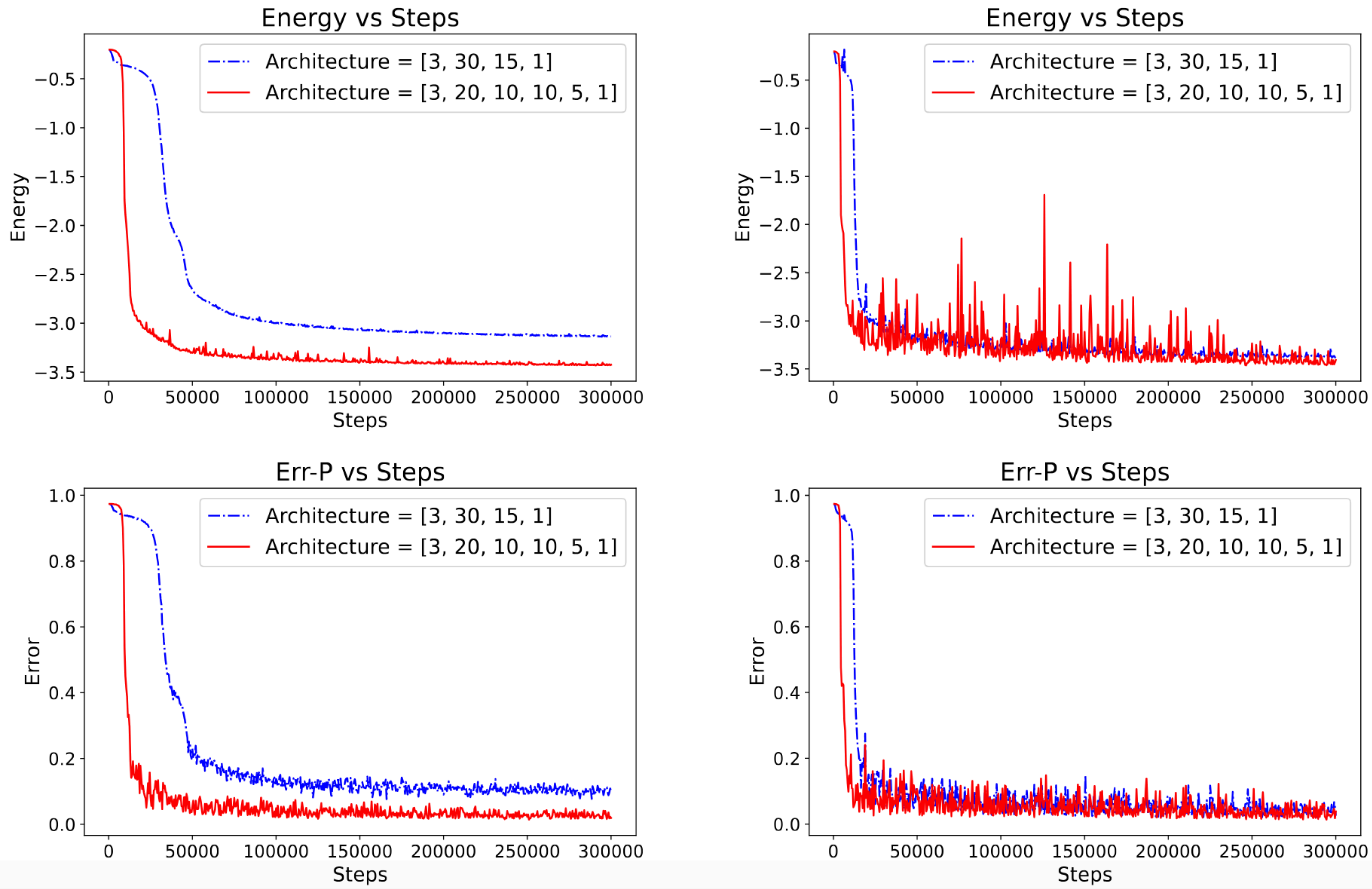
Set

- batch size = 6144;
- penalty λ = 250.

Compare

- network architecture = [3, 30, 15, 1], [3, 20, 10, 10, 5, 1];
- learning rate = 1e-03, 1e-02.

Network Architecture	Learning Rate	Err-E	Err-P
[3, 30, 15, 1]	1e-03	13.33%	9.71%
	1e-02	6.76%	3.57%
[3, 20, 10, 10, 5, 1]	1e-03	5.27%	3.62%
	1e-02	5.11%	3.83%



Energy vs Steps
Energy
Steps
Architecture = [3, 30, 15, 1]
Architecture = [3, 20, 10, 10, 5, 1]
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
−3.5
0
50000
100000
150000
200000
250000
300000
Energy vs Steps
Energy
Steps
Architecture = [3, 30, 15, 1]
Architecture = [3, 20, 10, 10, 5, 1]
Err-P vs Steps
Error
Steps
Architecture = [3, 30, 15, 1]
Architecture = [3, 20, 10, 10, 5, 1]
0.0
0.2
0.4
0.6
0.8
1.0
Err-P vs Steps
Error
Steps
Architecture = [3, 30, 15, 1]
Architecture = [3, 20, 10, 10, 5, 1]

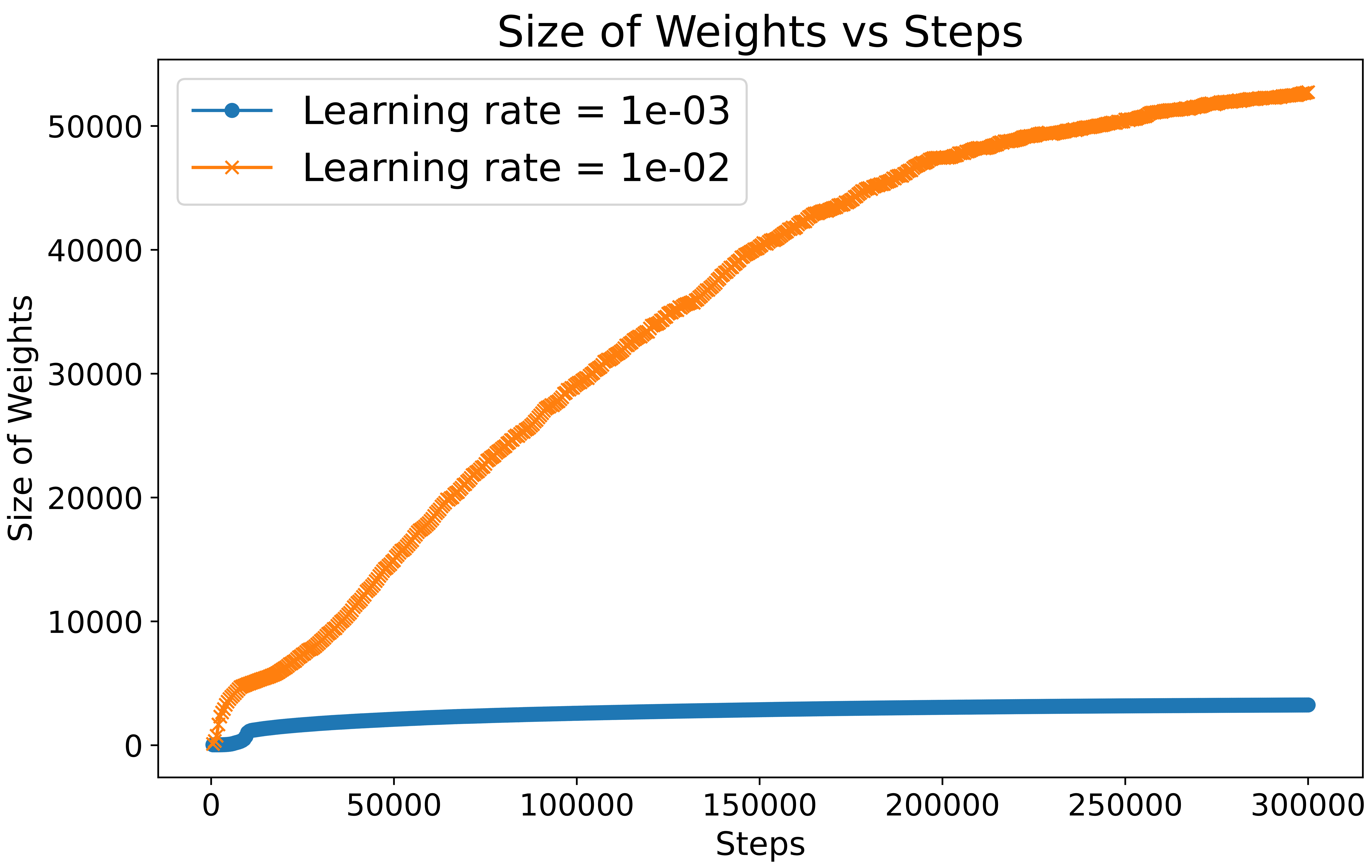
Test on the growth of weights

Set

- network architecture = [3, 30, 20, 15, 10, 1];
- batch size = 6144;
- penalty coefficient λ = 250.

Compare

- learning rate = 1e-03, 1e-02.

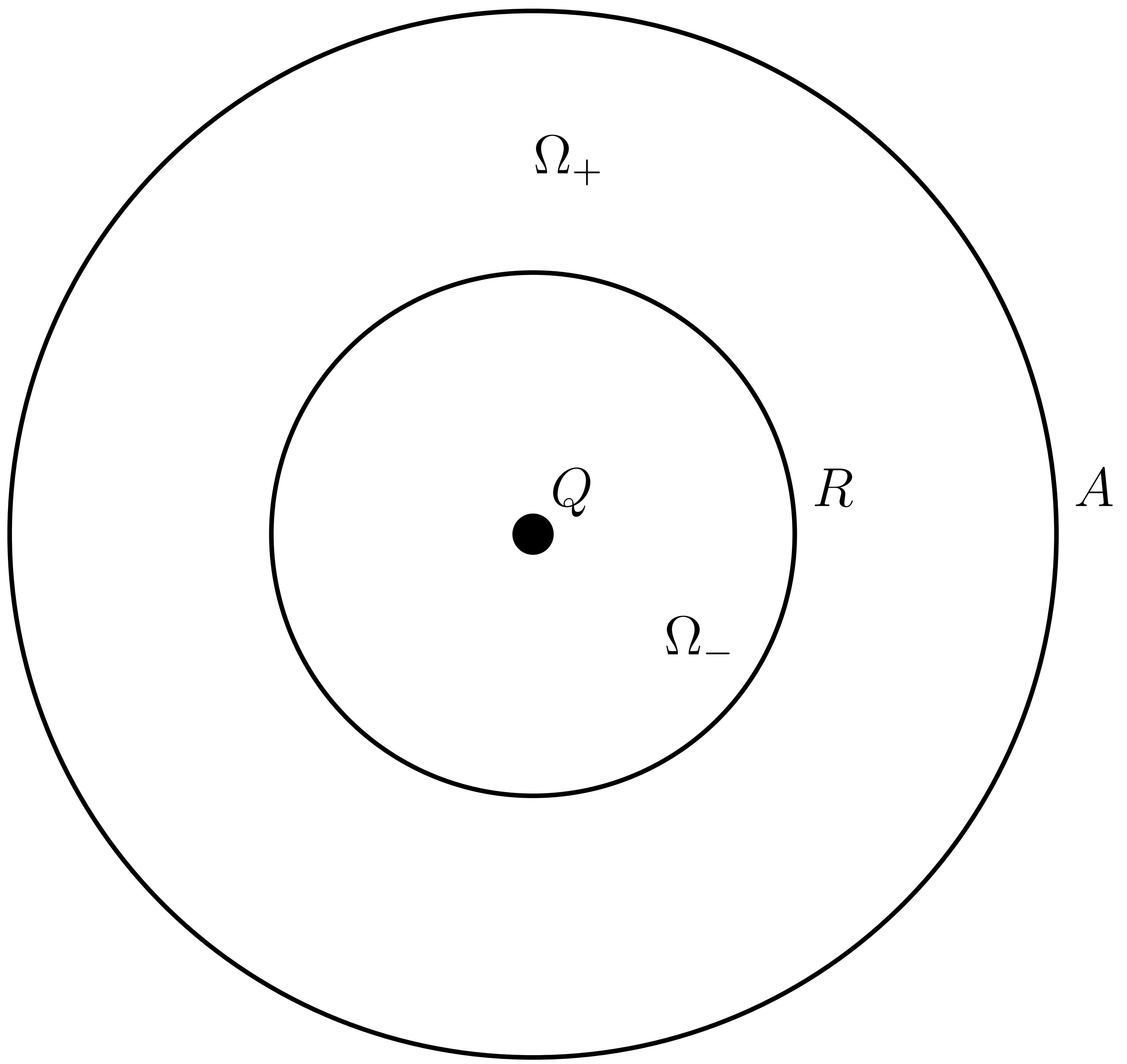

Size of Weights vs Steps
Learning rate = 1e-03
Learning rate = 1e-02
Size of Weights
Steps
0
10000
20000
30000
40000
50000
0
50000
100000
150000
200000
250000
300000

Applications to Solvation of Charged Molecules

Simple Ions

Set

- $\Omega_- = \{x \in \mathbb{R}^3 : |x| < R\}$;
- $\Omega_+ = \{x \in \mathbb{R}^3 : R < |x| < A\}$;
- $\Gamma = \{x \in \mathbb{R}^3 : |x| = R\}$.



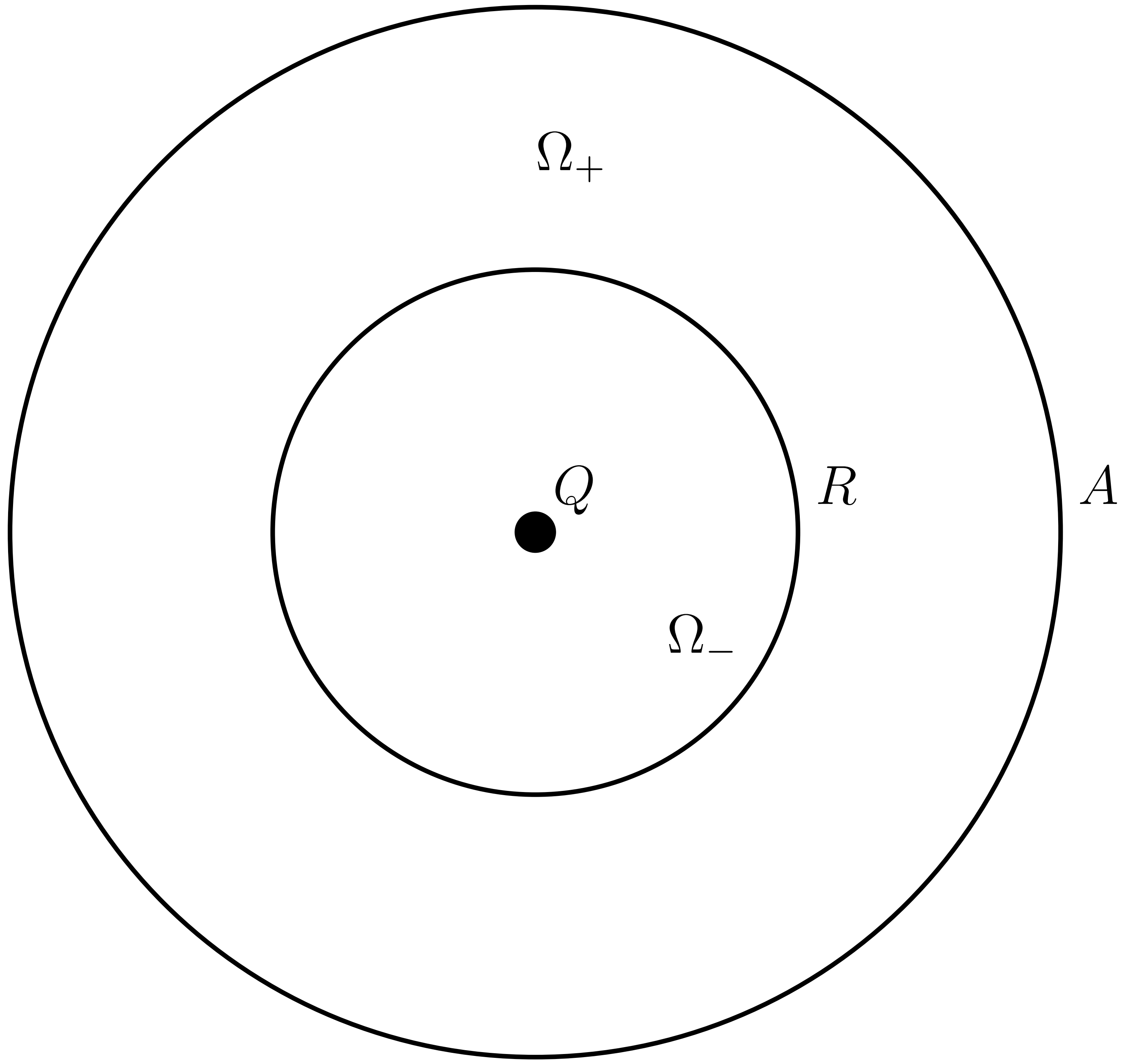
Electrostatic energy

Born's energy:

- $E_{\mathrm{Born}}(R) = \dfrac{Q^2}{8\pi R}\left(\dfrac{1}{\varepsilon_+} - \dfrac{1}{\varepsilon_-}\right)$

Approximation of Born's energy:

- $E_{\mathrm{ele},A}(R) = \dfrac{Q^2}{8\pi R}\left(\dfrac{1}{\varepsilon_+} - \dfrac{1}{\varepsilon_-}\right) - \dfrac{Q^2}{8\pi\varepsilon_+ A} + \dfrac{Qg}{2}$
- g is a constant



Total VISM energy

$$F(R) = \frac{4\pi}{3} P_0 R^3 + 4\pi\gamma_0 R^2 - 8\pi\gamma_0 \tau R + 16\pi\rho_{\mathrm{w}}\varepsilon_{\mathrm{LJ}} \left(\frac{\sigma_{\mathrm{LJ}}^{12}}{9R^9} - \frac{\sigma_{\mathrm{LJ}}^{6}}{3R^3} \right) + E_{\mathrm{ele}}(R)$$

Set $g = Q/(4\pi\varepsilon_+ A)$, $A = 4$, and $\lambda = 250$

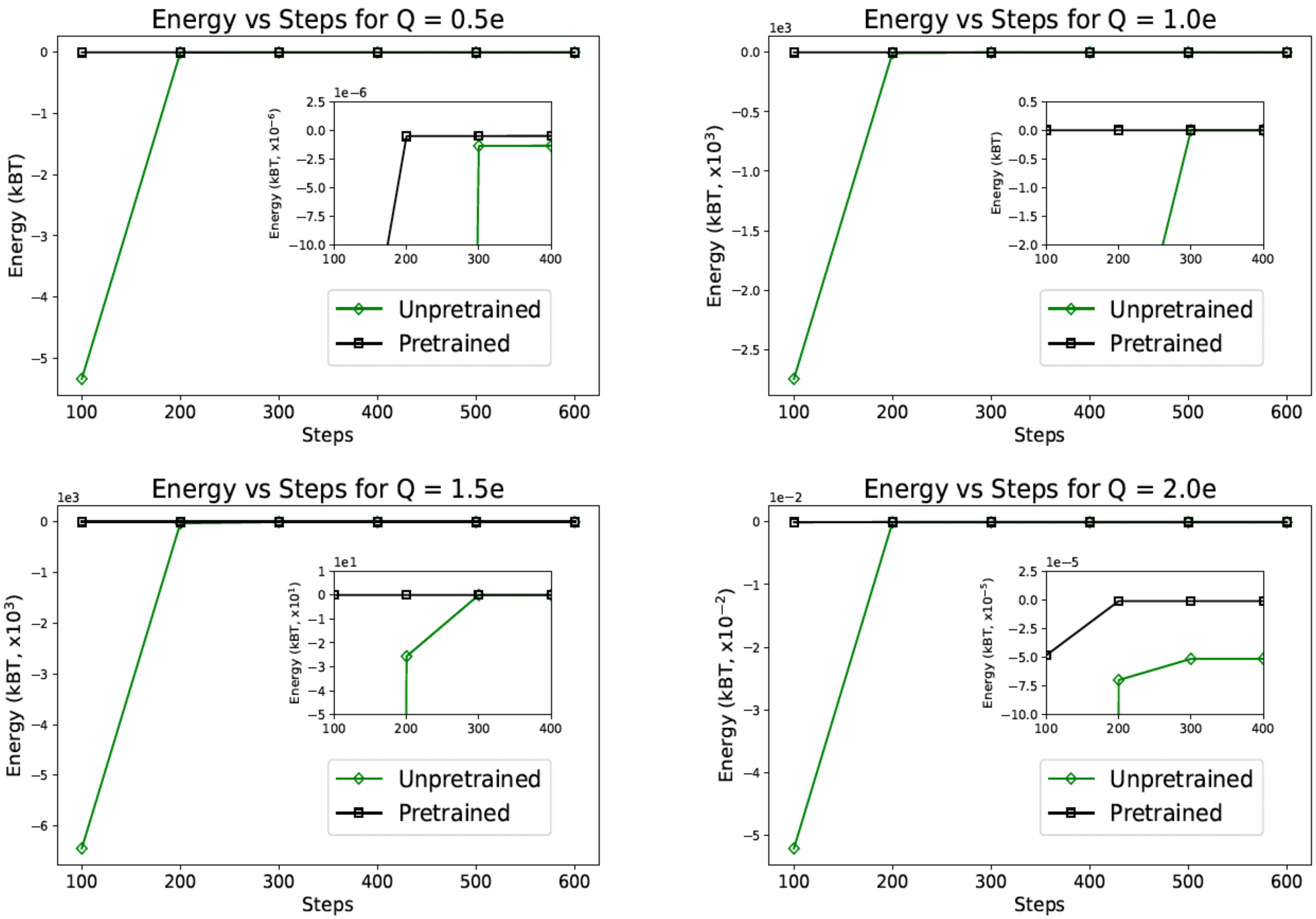
Parameters	Descriptions	Estimated Values	Units
T	temperature	300	Kelvin
P_0	pressure difference	0	bar
γ_0	constant surface tension	0.1315	$k_{\mathrm{B}}T/\text{Å}^2$
τ	Tolman length	0.76	Å
ρ_{w}	bulk solvent density	0.0331	Å^{-3}
ε_-	relative dielectric permittivity in Ω_-	1	ε_0
ε_+	relative dielectric permittivity in Ω_+	78	ε_0

Simulation Results

Charge Q	0.0	0.5	1.0	1.5	2.0
Radius R	3.157	3.030	2.801	2.605	2.453
Electrostatic-NN	0.0	-22.685	-98.156	-237.469	-448.326
Electrostatics-Born	0.0	-22.685	-98.156	-237.469	-448.326
VISM-NN	4.836	-17.413	-88.486	-216.174	-406.986
VISM-Born	4.836	-17.413	-88.486	-216.174	-406.986

$$F(R) = \frac{4\pi}{3} P_0 R^3 + 4\pi\gamma_0 R^2 - 8\pi\gamma_0 \tau R + 16\pi\rho_{\mathrm{w}}\varepsilon_{\mathrm{LJ}} \left(\frac{\sigma_{\mathrm{LJ}}^{12}}{9R^9} - \frac{\sigma_{\mathrm{LJ}}^{6}}{3R^3} \right) + E_{\mathrm{ele}}(R)$$

Transfer learning: Use Q = 0.0 to train Q = 0.5, 1.0, 1.5 and 2.0

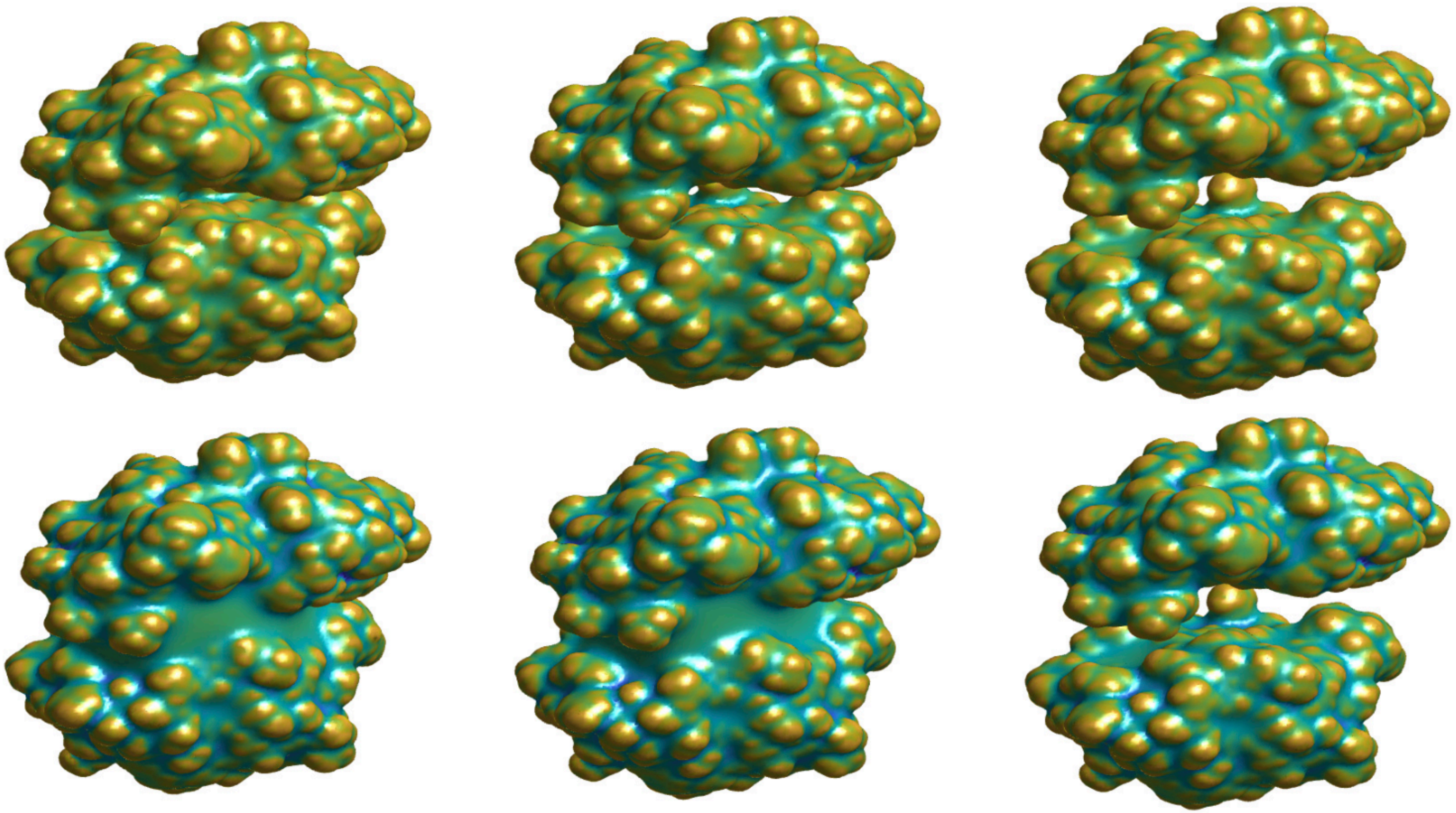


Real Simple Ions

Ions	ε ($k_B T$)	σ (Å)	VISM-NN	Experiment
K^+	0.008	3.85	-111.1	-117.5
Na^+	0.008	3.49	-129.9	-145.4
Cl^-	0.21	3.78	-126.1	-135.4
F^-	0.219	3.3	-171.0	-185.2

$$F(R) = \frac{4\pi}{3} P_0 R^3 + 4\pi\gamma_0 R^2 - 8\pi\gamma_0 \tau R + 16\pi\rho_{\mathrm{w}}\varepsilon_{\mathrm{LJ}} \left(\frac{\sigma_{\mathrm{LJ}}^{12}}{9R^9} - \frac{\sigma_{\mathrm{LJ}}^{6}}{3R^3} \right) + E_{\mathrm{ele}}(R)$$

BphC

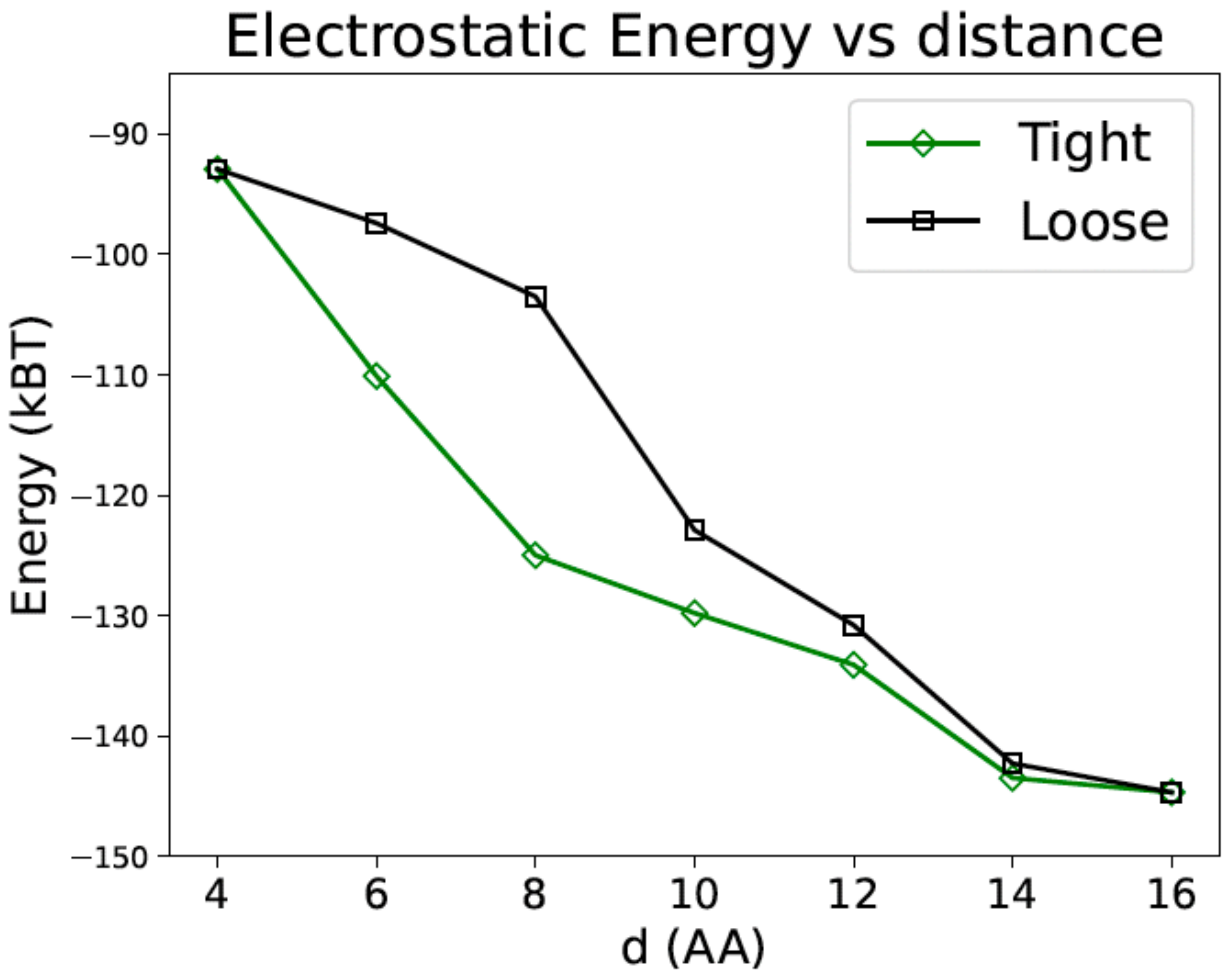


Test

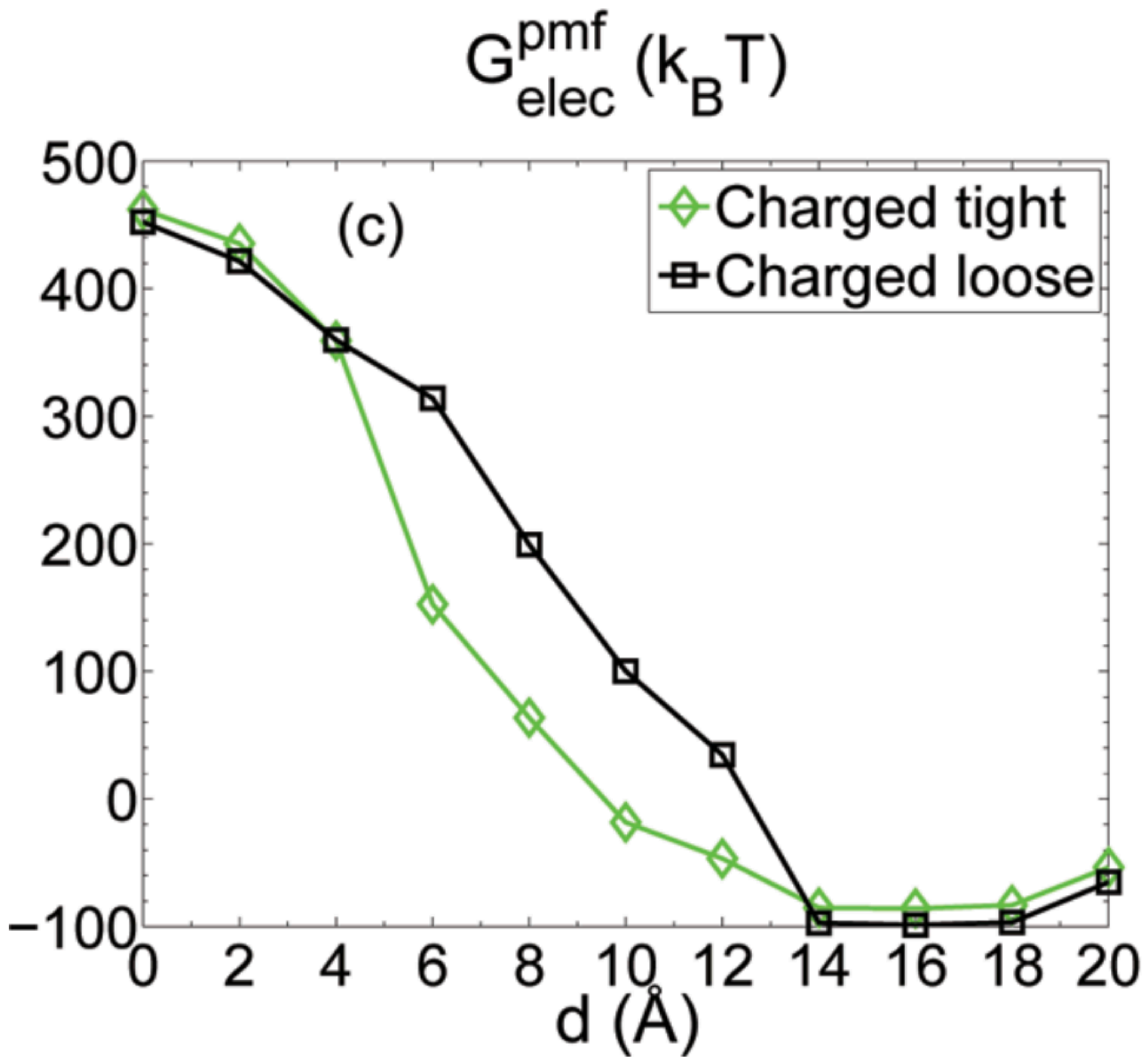
- Tight & Loose;
- d = 4, 6, 8, 10, 12, 14, 16 Å

Use penalized PB electrostatic free-energy functional $I_{\Gamma,\lambda}[\phi]$

$$I_{\Gamma,\lambda}[\phi] = \int_{\Omega} \left[\frac{\varepsilon_{\Gamma}}{2} |\nabla \phi|^2 - f\phi + \chi_{+} B(\phi) \right] dx + \lambda \int_{\partial\Omega} (\phi - g)^2 \, dS$$

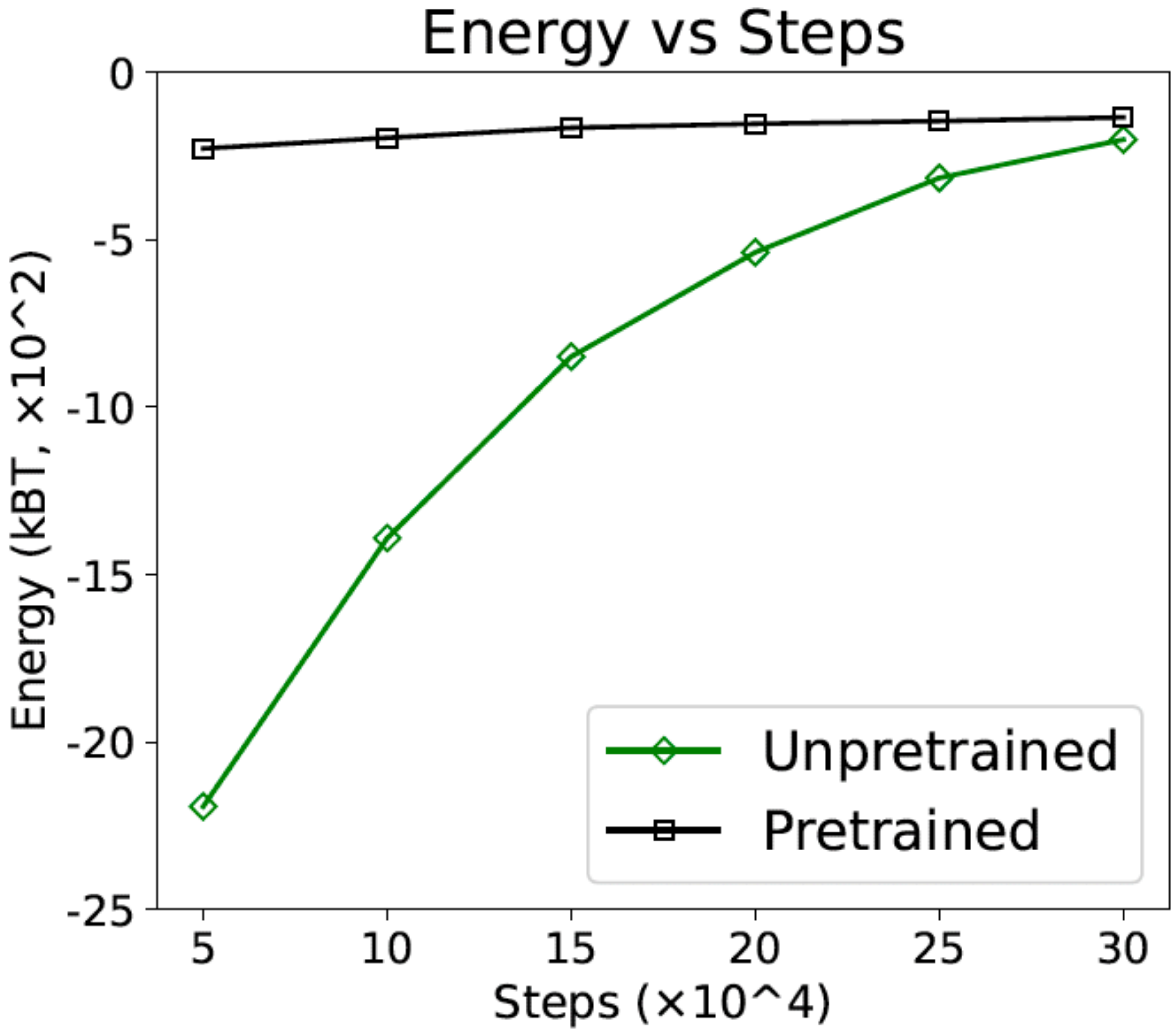


Electrostatic Energy vs distance
Tight
Loose
Energy (kBT)
−90
−100
−110
−120
−130
−140
−150
4
6
8
10
12
14
16
d (AA)



G_{elec}^{pmf} (k_BT)
(c)
Charged tight
Charged loose
500
400
300
200
100
0
−100
0
2
4
6
8
10
12
14
16
18
20
d (Å)

Transfer learning: Use Loose d = 8 to train d = 10



Conclusion and Discussion

Conclusion

- Developed a neural network approach to solving PBE;
- Introduced penalized PB energy functional and proved the convergence;
- Designed an algorithm for minimization and demonstrated the performance;
- Applied it with VISM to applications to simple ions and BphC;
- Interesting discovery:

  landscape of loss function;

  transferability of network weights.

Discussion

- Low-dimensional PDEs, not accurate or efficient;
- ML theory? Convergence analysis and error estimates?
- Hyperparameters?

Thank you!