WONAPDE 2024

SEVENTH CHILEAN WORKSHOP ON NUMERICAL ANALYSIS OF PARTIAL DIFFERENTIAL EQUATIONS

UNIVERSIDAD DE CONCEPCIÓN, CONCEPCIÓN, CHILE

JANUARY 15 - 19, 2024

UNIVERSIDAD TECNICA FEDERICO SANTA MARIA

Agencia Nacional de Investigación y Desarrollo

XPINN Solver for 3D Poisson-Boltzmann Equation

Martín Achondo Mercado

Christopher Cooper

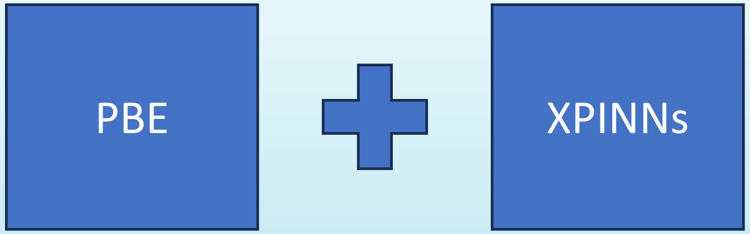
Jehanzeb Chaudhry

January 16, 2024

Introduction

The aim is to solve the Poisson-Boltzmann equation using the *Extended Physics Informed Neural Networks* technology.

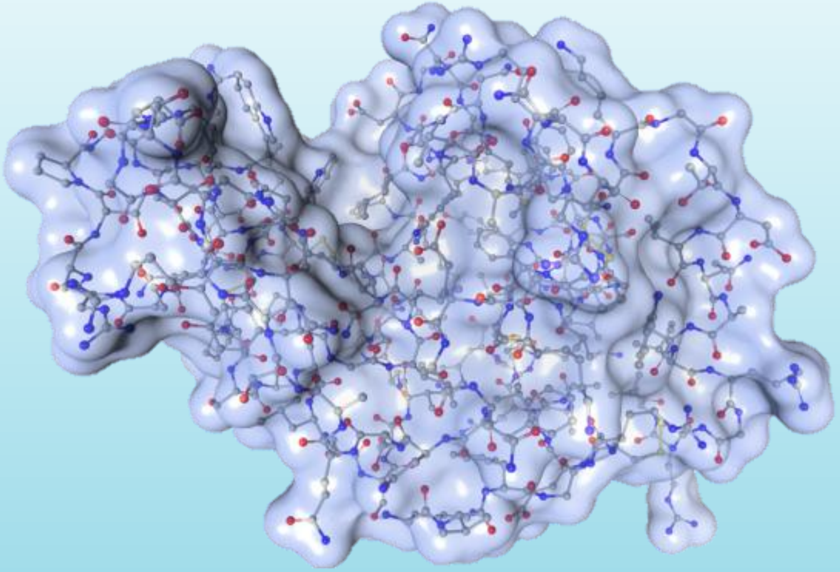
- Poisson-Boltzmann Equation: Models the interaction of macromolecules in a polarizable media.
- PINNs: Physics Informed Neural Networks. Method used to solve PDEs.



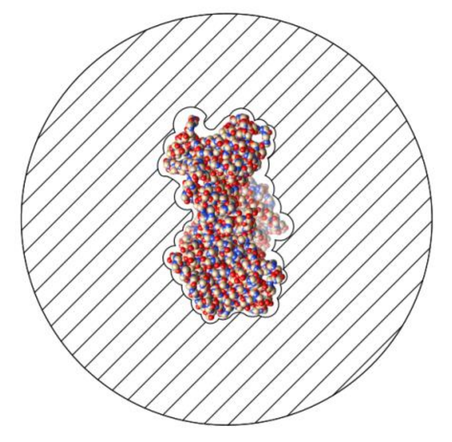
- New Implementation.
- Usable in real-world applications.
- Biochemistry.

Considerations:

- 3D problem.
- 2 domains (solute and solvent region).
- A lot of loss terms needed, and some optional can be added.

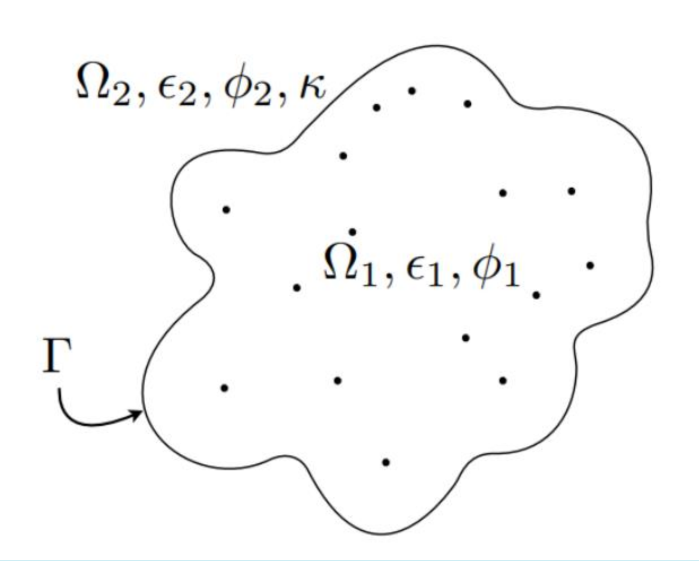


Poisson-Boltzmann Equation



- Applicable to macromolecules in a polarizable media.
- Obtained using the Implicit Solvent Method.

$$\begin{cases} \nabla^2 \phi_1 = -\dfrac{1}{\epsilon_1} \displaystyle\sum_k q_k \delta(\mathbf{x} - \mathbf{x}_k) & \mathbf{x} \in \Omega_1 \\ \nabla^2 \phi_2 = \dfrac{2c^\infty q_e}{\epsilon_2} \sinh\left(\dfrac{\phi_2 q_e}{k_b T}\right) & \mathbf{x} \in \Omega_2 \end{cases}$$

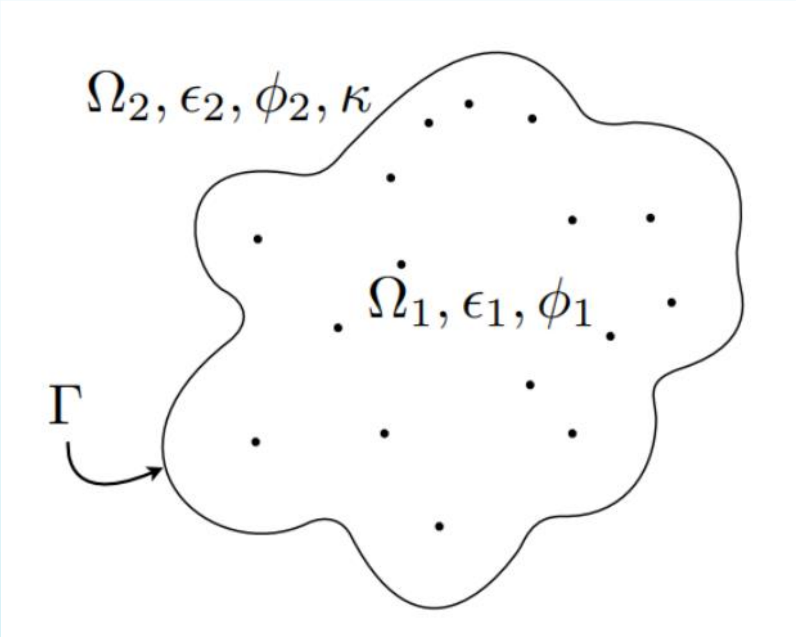


Linearized: $\nabla^2 \phi_2 = \kappa^2 \phi_2$ $\quad \kappa^2 = \dfrac{2c^\infty q_e^2}{\epsilon_2 k_b T}$

Poisson-Boltzmann Equation

- Interface conditions:

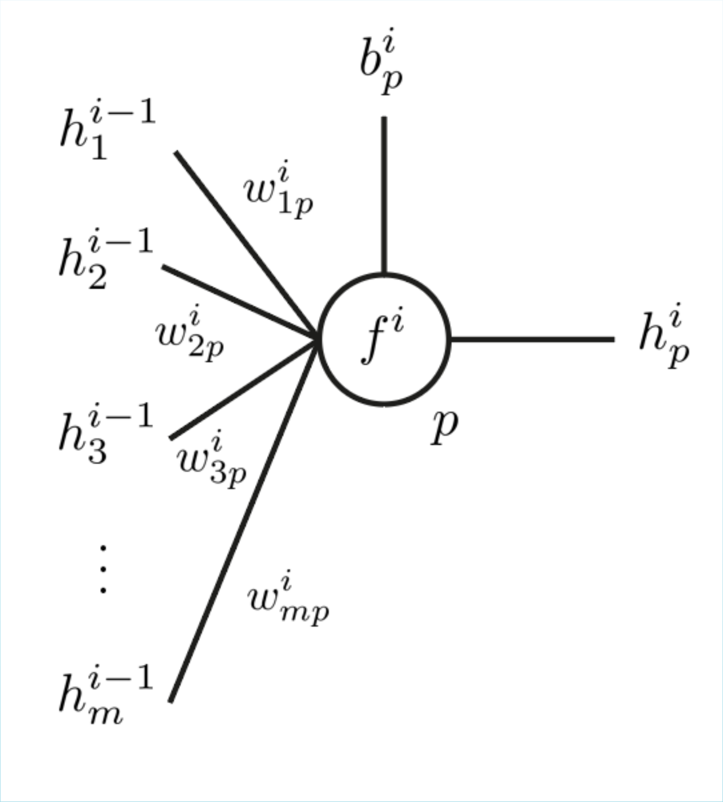
$$\begin{cases} \phi_1 = \phi_2 & \mathbf{x} \in \Gamma \\ \epsilon_1 \dfrac{\partial \phi_1}{\partial n} = \epsilon_2 \dfrac{\partial \phi_2}{\partial n} & \mathbf{x} \in \Gamma \end{cases}$$



Artificial Neural Networks (ANN)

- Operations in every perceptron (neuron).
- Considering the p-th perceptron of the i-th layer:

It has weights in the conexions w^i_{mp}, bias b^i_p, and activation function f^i.



$$h^i_p = f^i\left(\sum_{j=1}^{m} h^{i-1}_j w^i_{jp} + b^i_p\right)$$

Artificial Neural Networks (ANN)

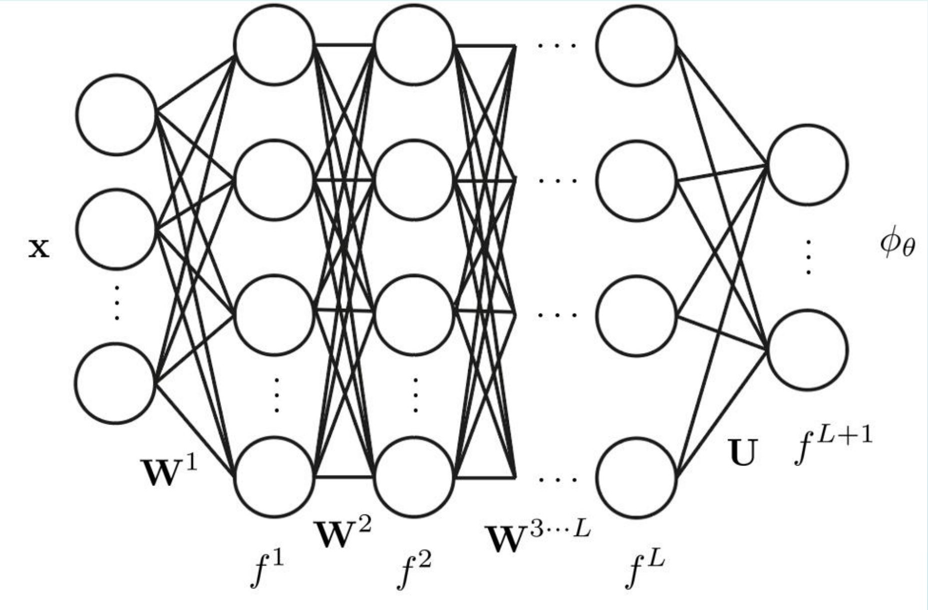
- Assembling multiple interconnected perceptrons forms an Artificial Neural Network (ANN).

$$\phi_\theta = \mathcal{N}(\mathbf{x};\theta)$$

- Example: *Fully Connected Neural Network*

$$\theta = \{\boldsymbol{W}^1, \boldsymbol{b}^1, \ldots, \boldsymbol{W}^L, \boldsymbol{b}^L, \boldsymbol{U}, \boldsymbol{c}\}$$

$$\begin{cases} \boldsymbol{h}^1 = f^1(\mathbf{x}\boldsymbol{W}^1 + \boldsymbol{b}^1) & i = 1 \text{ capa entrada} \\ \boldsymbol{h}^i = f^i(\boldsymbol{h}^{i-1}\boldsymbol{W}^i + \boldsymbol{b}^i) & 2 \leq i \leq L \text{ capas ocultas} \\ \phi_\theta = f^{L+1}(\boldsymbol{h}^L\boldsymbol{U} + \boldsymbol{c}) & i = L+1 \text{ salida} \end{cases}$$



ANN and PINNs (Physics Informed Neural Networks)

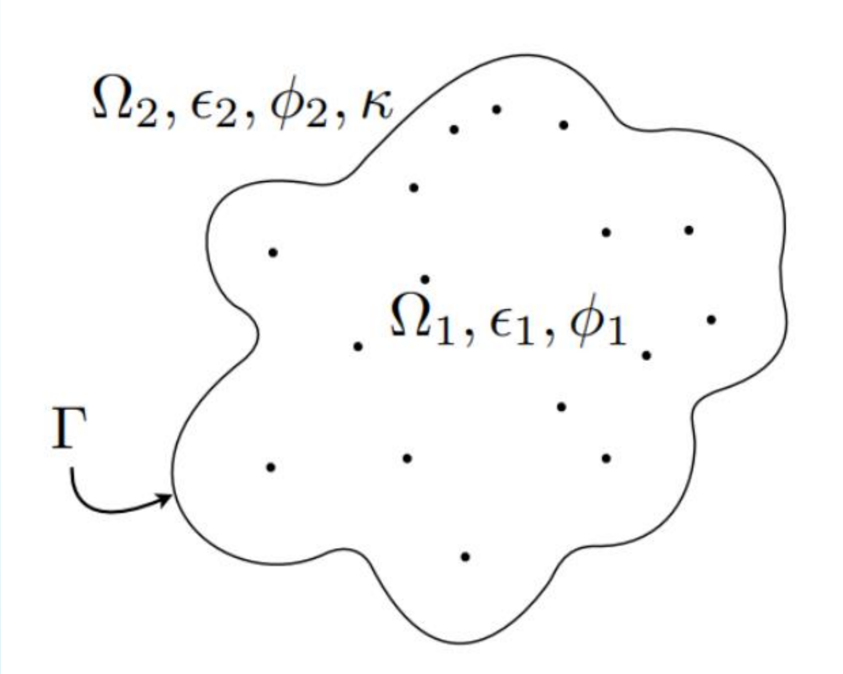
- The concept involves incorporating the equation to solve (residuals, boundary conditions, physics laws, etc.) into the loss function $\mathcal{L}$.
- This ensures that the output of the ANN approximates the solution of the PDE.
- Optimization algorithms based on gradient descent. $\theta^* = \underset{\theta}{\operatorname{argmin}}\, \mathcal{L}(\theta; \mathcal{S})$
- Loss function is evaluated on the $\mathcal{S}$ set (collocation points).
- Algorithms: ADAM, L-BFGS + stochastic methods

$$\theta_{k+1} = \theta_k - \lambda \nabla_\theta \mathcal{L}(\theta_k; \mathcal{S}_j) \qquad \mathcal{S}_j \subset \mathcal{S}$$

XPINN Applied to PBE

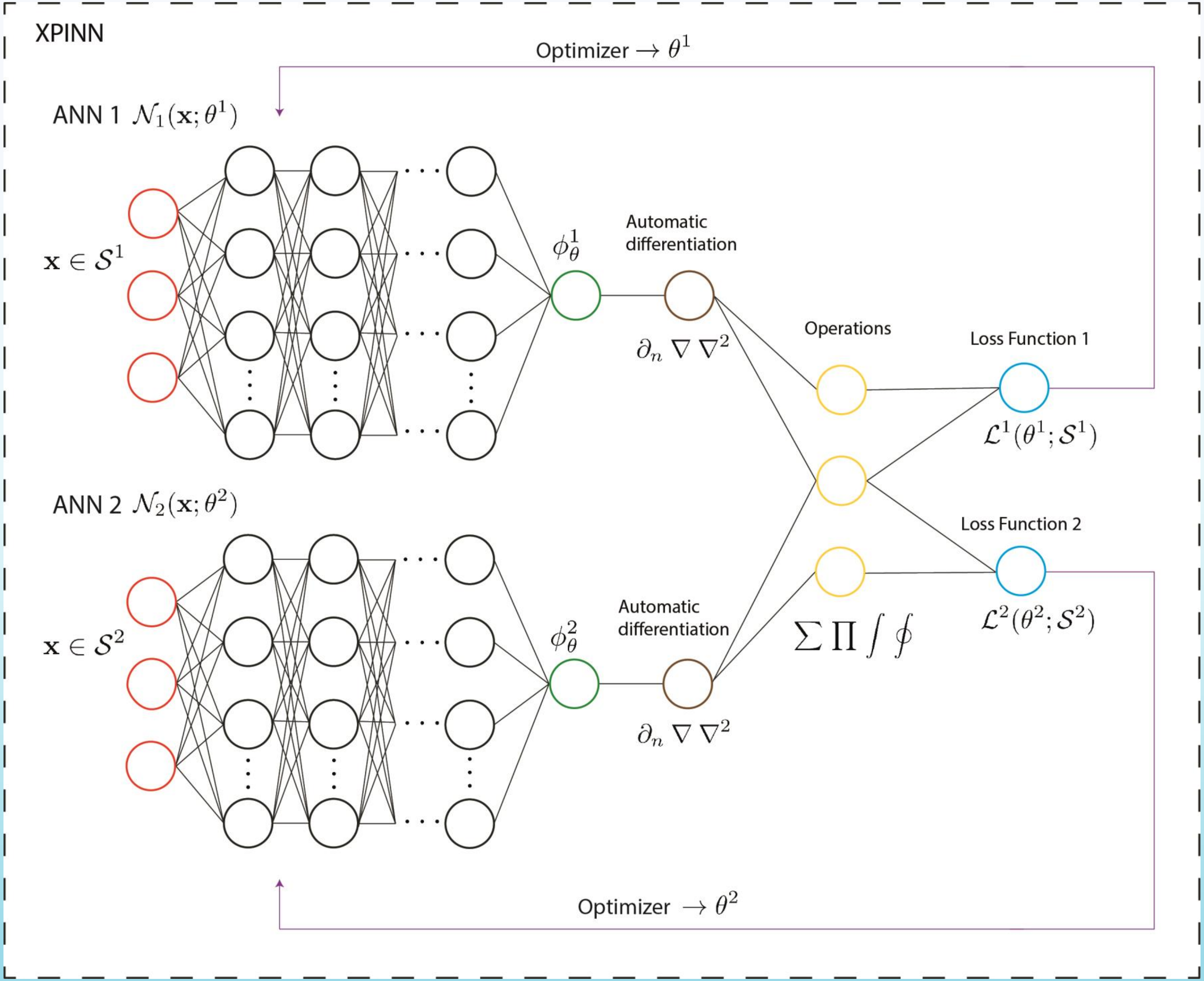
- 2 ANN that outputs the electrostatic potential:
  - N°1: Solute region.
  - N°2: Solvent region.

$$\phi \approx \begin{cases} \phi_\theta^1 = \mathcal{N}_1(\mathbf{x}; \theta^1) & \mathbf{x} \in \Omega_1 \\ \phi_\theta^2 = \mathcal{N}_2(\mathbf{x}; \theta^2) & \mathbf{x} \in \Omega_2 \end{cases}$$



- The loss functions will depend on both ANN, including the following terms:
  - Residual PBE.
  - Boundary conditions.
  - Interface relations.
  - Experimental data.
  - Gauss Law.

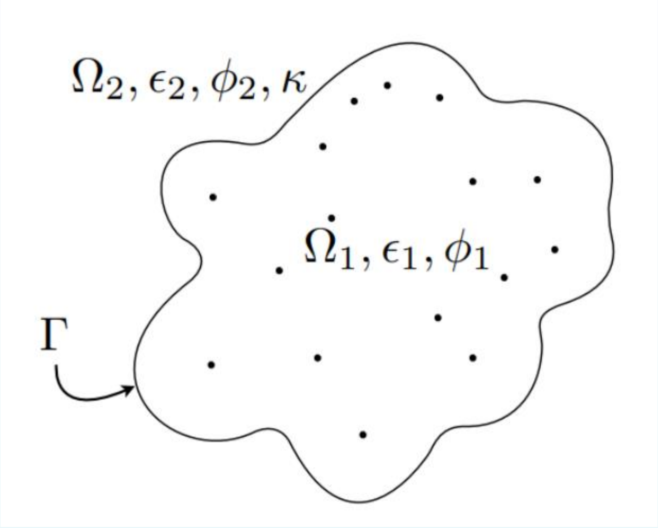
$$\mathcal{L}(\mathcal{S}) = \sum_k w_k \mathcal{L}_k(\mathcal{S}_k)$$



XPINNs for PBE:

- 2 ANN (solute and solvent regions).
- 2 Loss functions that depends on both ANNs.

XPINN Applied to PBE



$$\phi \approx \begin{cases} \phi_\theta^1 = \mathcal{N}_1(\mathbf{x};\theta^1) & \mathbf{x} \in \Omega_1 \\ \phi_\theta^2 = \mathcal{N}_2(\mathbf{x};\theta^2) & \mathbf{x} \in \Omega_2 \end{cases}$$

- Residuals:

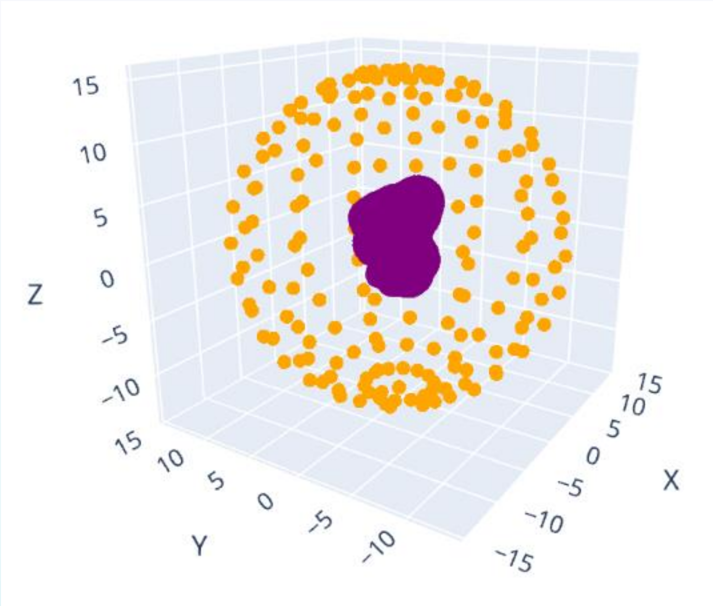
$$\mathcal{L}_{pde}^1(\mathcal{S}_{pde}) = \frac{1}{N_{pde}} \sum_{x_i \in \mathcal{S}_{pde}} \left[\nabla^2 \phi_\theta^1(x_i) + \frac{1}{\epsilon_1} \sum_k q_k \delta(x_i - x_k) \right]^2$$

*Dirac delta will be approximated by a Gaussian function.

$$\mathcal{L}_{pde}^2(\mathcal{S}_{pde}) = \frac{1}{N_{pde}} \sum_{x_i \in \mathcal{S}_{pde}} \left[\nabla^2 \phi_\theta^2(x_i) - \kappa^2 \phi_\theta^2(x_i) \right]^2$$

$$\delta(x) \approx \frac{1}{(2\pi)^{3/2}\sigma^3} e^{-\frac{1}{2\sigma^2}\|x\|^2}$$

XPINN Applied to PBE



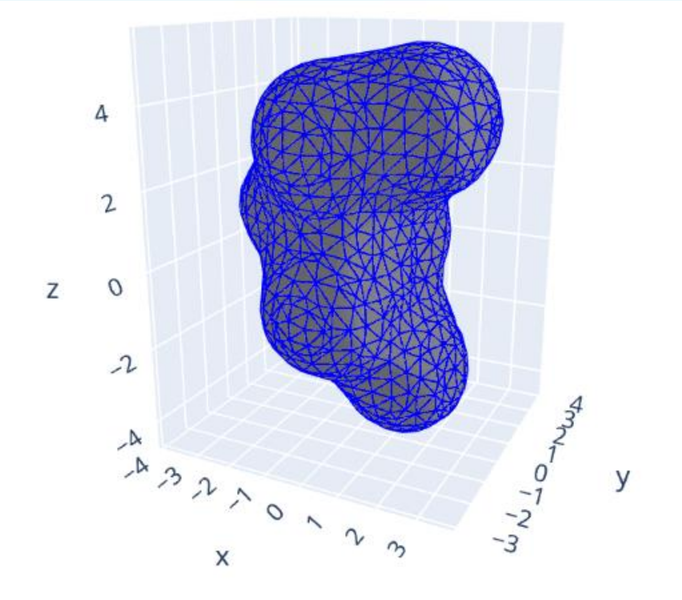
- Boundary condition:

$$\mathcal{L}_{bc}^2(\mathcal{S}_{bc}) = \frac{1}{N_{bc}} \sum_{x_i \in \mathcal{S}_{bc}} \left[\phi_\theta^2(x_i) - \frac{1}{4\pi\epsilon_2} \sum_k \frac{q_k \mathrm{e}^{-\kappa|x_i - x_k|}}{|x_i - x_k|} \right]^2$$

- Interface conditions (j-th ANN)

$$\mathcal{L}_{Iu}^j(\mathcal{S}_I) = \frac{1}{N_I} \sum_{x_i \in \mathcal{S}_I} \left[\phi_\theta^j(x_i) - \overline{\phi}_\theta(x_i) \right]^2$$

$$\mathcal{L}_{Id}^j(\mathcal{S}_I) = \frac{1}{N_I} \sum_{x_i \in \mathcal{S}_I} \left[\epsilon_j \partial_n \phi_\theta^j(x_i) - \overline{\epsilon \partial_n \phi}_\theta(x_i) \right]^2$$



XPINN Applied to PBE

- "Known solution" in random positions (results from software pbj + 10% Noise) (j-th ANN):

$$\mathcal{L}^{j}_{data}(\mathcal{S}_{data}) = \frac{1}{N_{data}} \sum_{x_i \in \mathcal{S}_{data}} \left[\phi^{j}_{\theta}(x_i) - \phi^{*}_{\theta}(x_i) \right]^2$$

- Gauss Law: Additional Physics Law.

$$\mathcal{L}_{Gauss} = \left| \oint_{\Gamma} \overline{\epsilon \partial_n \phi}_{\theta}(x') \, \mathrm{d}S(x') - \sum_{k} q_k \right|^2$$

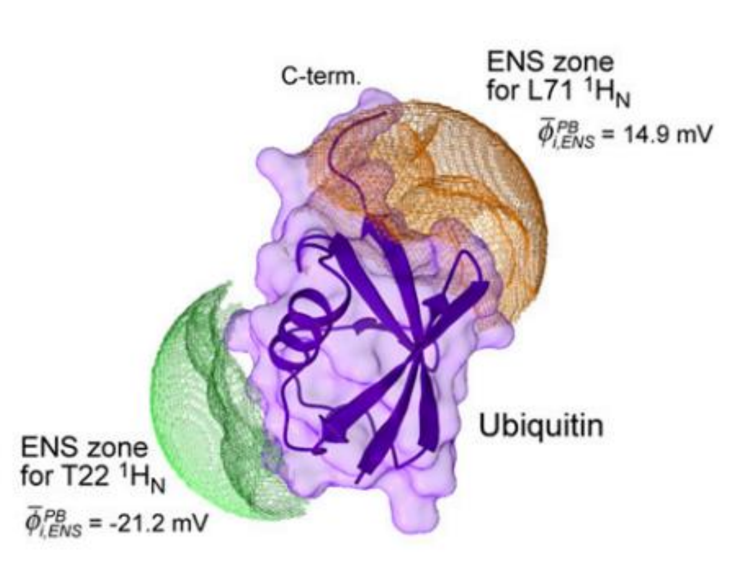
Experimental data ϕ_{ENS}

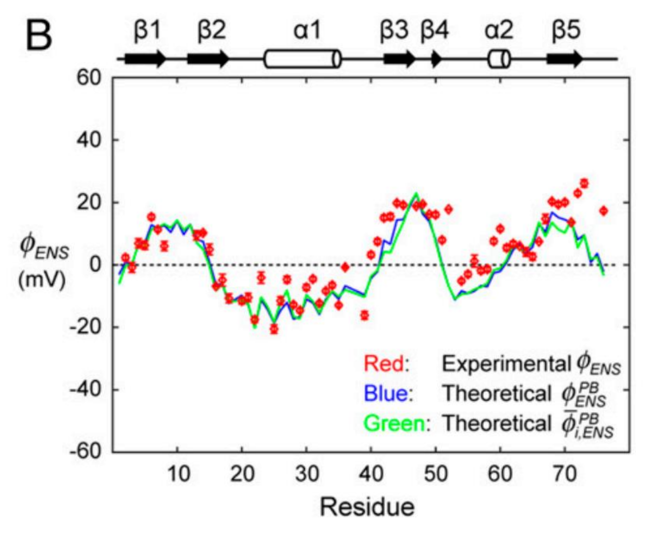
Results for experimental effective near-surface potential.

- For each hydrogen atom h:

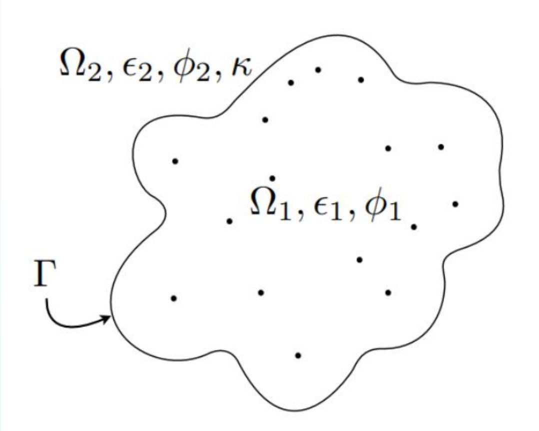
$$\phi_{ENS}(x_h) = \frac{-k_b T}{2q_e} \ln \left(\frac{\int_0^\infty r^{-4}\, \mathrm{e}^{-\frac{q_e \phi_\theta(r)}{k_b T}}\, \mathrm{d}r}{\int_0^\infty r^{-4}\, \mathrm{e}^{\frac{q_e \phi_\theta(r)}{k_b T}}\, \mathrm{d}r} \right)$$

$$\mathcal{L}_E(\mathcal{S}_H) = \frac{1}{N_H} \sum_{x_h \in \mathcal{S}_H} \left[\phi_{ENS,\theta}(x_h) - \phi_{ENS}(x_h) \right]^2$$





Solvation Energy



$$G_{solv} = \frac{1}{2}\sum_k q_k \phi_{reac}(x_k) \qquad \phi_{reac} = \phi - \phi_{coulomb}$$

- Green identities are used for getting the reaction potential at the point charges (uses the potential at interface):

$$\phi_{reac}(x_k) = \frac{1}{4\pi}\oint_\Gamma \overline{\frac{\partial \phi_1}{\partial n}} \frac{1}{|x_k - x'|}\,\mathrm{d}S(x') - \frac{1}{4\pi}\oint_\Gamma \overline{\phi}\frac{\partial}{\partial n}\left(\frac{1}{|x_k - x'|}\right)\mathrm{d}S(x')$$

Weights balancing algortihm for loss terms

- The working principle is to balance the contribution of the different loss terms for the modification of the set θ.

$$\hat{w}_k = \frac{\sum_i \|\nabla_\theta \mathcal{L}_i\|}{\|\nabla_\theta \mathcal{L}_k\|}$$

$$w_{k,\text{new}} = \alpha w_{k,\text{old}} + (1-\alpha)\hat{w}_k$$

- This algorithm works fine, being applied every 1000 or 2000 iterations.

Implementation

- All the simulations were implemented using a full-batch approach, with **~15.000 points** per dataset.
- For the architecture, **FCNN** was preferred, with addition of a **scale layer** and a **Fourier features layer**, after the input layer.

$$y_{fourier} = [\cos(Bx), \sin(Bx)]$$

Where B is a matrix generated by a normal distribution (non trainable), using **256 features**.

- The preferred hyperparameters were: **4 hidden layers with 200 neurons per layer**, with **tanh** as activation function.
- **Exponential decay learning rate** was used, starting from 0,001
- The optimization algorithm used was **ADAM**.

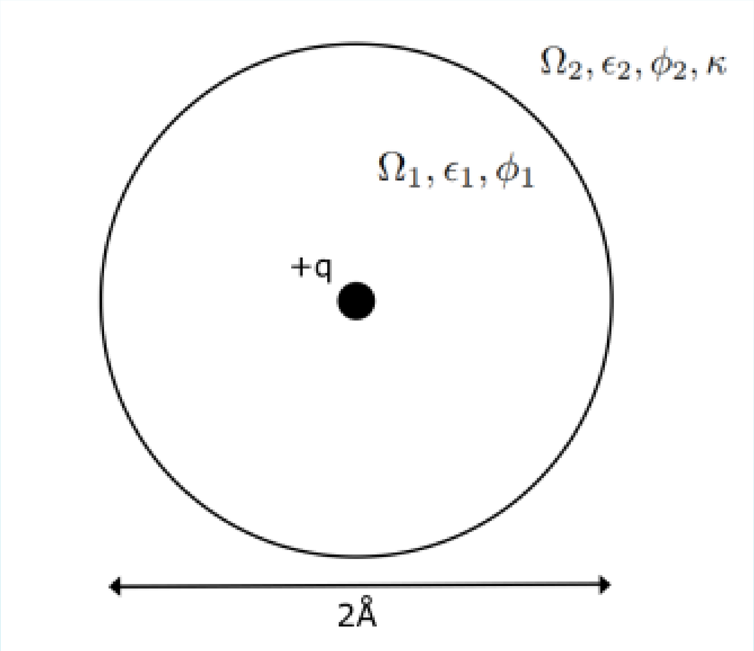
Validation: Born Ion

- Spherical molecule, 1 point charge.
- Analytical solution is known.
- Very simple test case.

$$\phi_1(r) = \frac{q}{4\pi}\left(\frac{1}{\epsilon_1 r} - \frac{1}{\epsilon_1 R} + \frac{1}{\epsilon_2(1+\kappa R)R}\right)$$

$$\phi_2(r) = \frac{q}{4\pi}\frac{\exp(-\kappa(r-R))}{\epsilon_2(1+\kappa R)r}$$

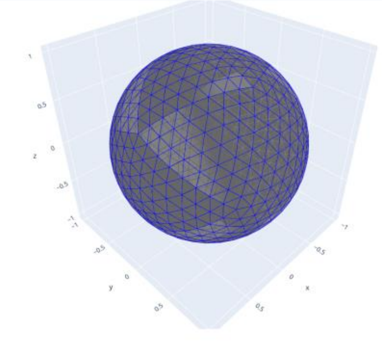
- L2 error at interface can be calculated.



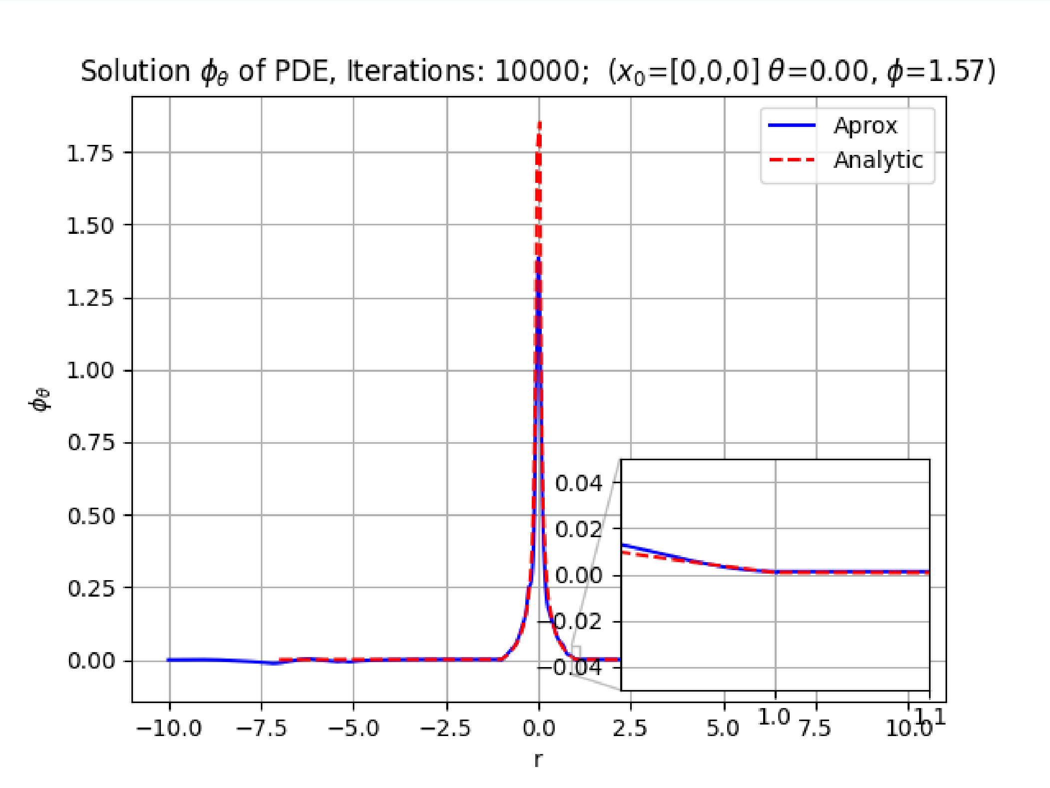
Born Ion

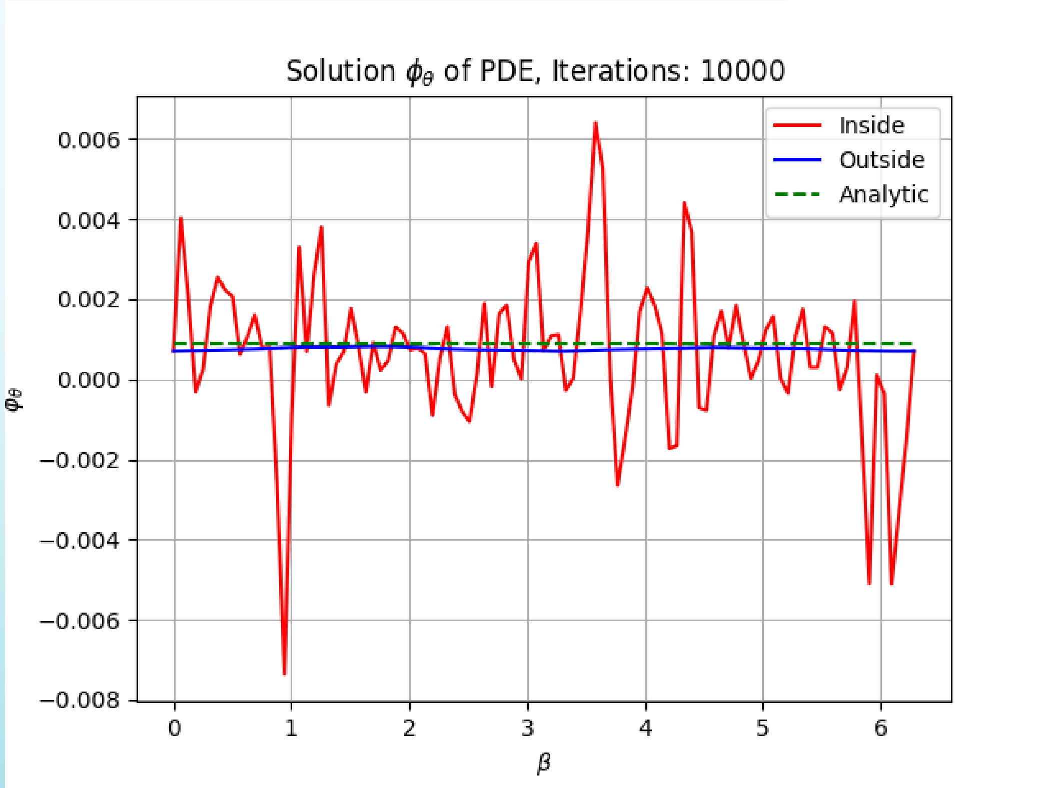
BORN ION			Architecture				Loss terms				Results					
N° Sim	Mesh	ARCH	HL	NpL	F	W	P	K	G	E	Gsolv_value	L2_analytic	L2_cont_u	L2_cont_du	Loss_NN1	Loss_NN2
1	Medium	FCNN	4	200				Noise			-248.181	5.13.E+02	2.13.E-02	1.57.E-02	3.28.E+00	7.80.E-04
2	Medium	FCNN	4	200	I	X		Noise	X		32.485	1.16.E+03	4.02.E-02	8.43.E-02	1.10.E+05	8.83.E-03
3	Medium	FCNN	4	200	I	X		Noise		X	0.712	2.52.E+01	2.76.E-04	7.73.E-04	3.80.E-01	5.47.E+02
4	Medium	FCNN	4	200	I	X		Noise	X	X	30.944	7.24.E+02	2.25.E-02	1.37.E-01	2.62.E-02	7.11.E+02
5	Medium	FCNN	4	200	I	X			X	X	48.058	1.88.E+03	6.20.E-02	2.15.E-01	6.23.E+04	2.37.E+02
6	Medium	FCNN	4	200	I	X	X		X	X	-1.187	4.42.E+01	2.07.E-04	1.43.E-04	1.82.E-04	2.25.E-04
7	Medium	FCNN	4	200	I	X	X	Noise	X	X	4.942	3.06.E+02	1.08.E-02	2.51.E-02	1.18.E-02	2.10.E+02
8	Medium	FCNN	4	200		X	X		X	X	-14.575	2.83.E+02	5.65.E-03	2.64.E-03	2.55.E+00	1.70.E+01
9	Medium	FCNN	4	200	I			Noise	X	X	97.865	2.27.E+03	6.78.E-02	2.02.E-01	5.71.E+01	2.35.E+02
10	Medium	FCNN	4	200	I						38.634	6.20.E+02	1.00.E-02	1.27.E-02	3.01.E-01	2.75.E-04
11	Medium	FCNN	4	200	I	X		Noise			0.406	3.05.E+01	9.71.E-04	2.34.E-03	1.29.E+01	7.00.E-06
12	Medium	FCNN	4	200	I	X		Noise		X	-0.005	2.68.E+01	4.04.E-04	4.02.E-05	2.33.E-01	1.68.E+01
13	Medium	FCNN	4	200	I	X				X	-0.052	1.08.E+02	7.22.E-08	1.35.E-07	7.59.E-09	8.52.E+02
14	Medium	FCNN	4	200	I	X	X			X	-0.672	3.15.E+01	7.58.E-05	2.07.E-05	1.06.E-05	9.16.E-04
15	Medium	FCNN	4	200	I	X	X	Noise		X	-0.411	3.10.E+01	8.43.E-05	1.53.E-04	3.48.E-03	4.47.E+00
16	Medium	FCNN	4	200		X	X			X	-0.359	3.38.E+01	5.38.E-04	1.77.E-04	1.17.E-01	1.60.E+01
17	Medium	FCNN	4	200	I			Noise		X	34.074	1.53.E+03	5.41.E-02	1.82.E-01	9.36.E+01	2.06.E-01
18	Coarse	FCNN	4	200	I	X		Noise		X	-0.502	1.65.E+01	2.15.E-05	6.25.E-05	4.12.E-03	1.45.E+00
19	Fine	FCNN	4	200	I	X		Noise		X	-0.503	6.74.E+01	7.20.E-04	1.32.E-03	1.29.E+00	2.07.E+01
20	Medium	FCNN	4	300	I	X		Noise		X	3.566	7.02.E+02	2.46.E-02	8.48.E-02	1.97.E-02	1.82.E+01
21	Medium	FCNN	4	120	I	X		Noise		X	-0.072	2.66.E+01	1.09.E-04	2.79.E-04	2.35.E-01	7.08.E+01
22	Medium	FCNN	3	200	I	X		Noise		X	-0.241	2.87.E+01	2.32.E-04	2.82.E-05	3.27.E-03	2.96.E+00
23	Medium	FCNN	5	200	I	X		Noise		X	0.126	2.71.E+01	1.59.E-04	3.18.E-05	7.62.E-03	4.15.E+01
24	Medium	FCNN	4	200	B	X		Noise		X	-33.431	7.70.E+00	1.51.E-04	7.12.E-05	7.76.E-05	7.79.E-05
25	Medium	FCNN	4	200		X		Noise		X	8.756	1.84.E+02	6.92.E-03	3.55.E-04	1.98.E+01	1.75.E+02
26	Medium	ResNet	4	200	I	X		Noise		X	-0.126	5.41.E+01	1.74.E-03	3.46.E-03	2.50.E+00	1.70.E+02
27	Medium	ResNet	4	200	B	X		Noise		X	-14.593	8.40.E+02	2.93.E-02	8.42.E-03	2.34.E-01	1.03.E+01
28	Medium	ResNet	4	200		X		Noise		X	10.164	3.88.E+02	2.83.E-03	2.85.E-04	7.95.E+00	4.57.E+02
29	Coarse	FCNN	4	200	B	X		Noise		X	-43.004	1.38.E+02	2.41.E-04	1.64.E-05	2.07.E-05	5.94.E+01
30	Fine	FCNN	4	200	B	X		Noise		X	13.165	3.49.E+02	1.81.E-03	1.85.E-03	4.89.E-02	6.76.E+02
31	Medium	FCNN	4	300	B	X		Noise		X	4.579	1.06.E+03	3.23.E-02	1.42.E-01	9.39.E+02	1.08.E+02
32	Medium	FCNN	4	120	B	X		Noise		X	-41.351	6.16.E+00	1.41.E-04	5.21.E-04	5.33.E+00	1.66.E-04
33	Medium	FCNN	3	200	B	X		Noise		X	-18.919	2.86.E+02	4.60.E-03	9.48.E-03	1.49.E+01	7.85.E+00
34	Medium	FCNN	5	200	B	X		Noise		X	249.467	1.21.E+03	3.34.E-02	8.27.E-03	1.02.E-01	1.87.E+02

Born Ion

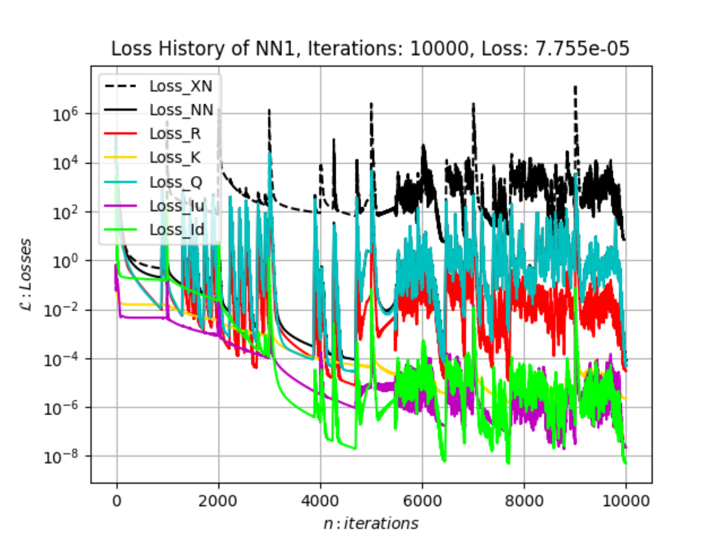


For 10.000 iterations, it “converges” to the analytical solution.

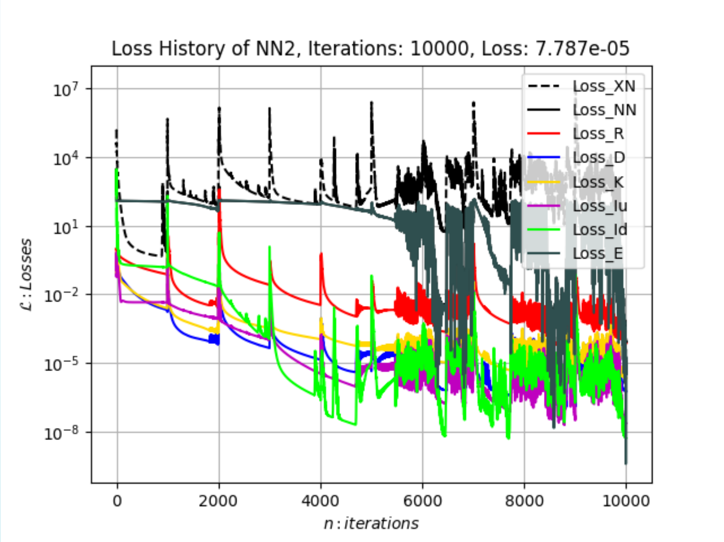




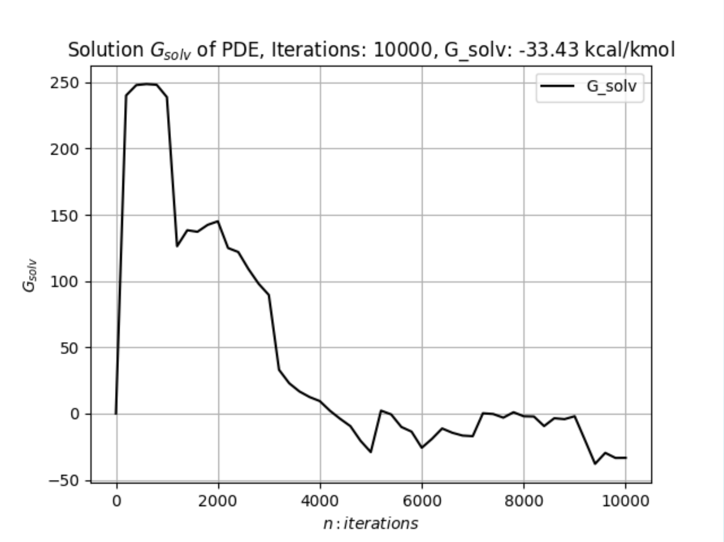
Born Ion



Loss History of NN1, Iterations: 10000, Loss: 7.755e-05
Loss_XN
Loss_NN
Loss_R
Loss_K
Loss_Q
Loss_Iu
Loss_Id
$\mathcal{L}: Losses$
$n: iterations$

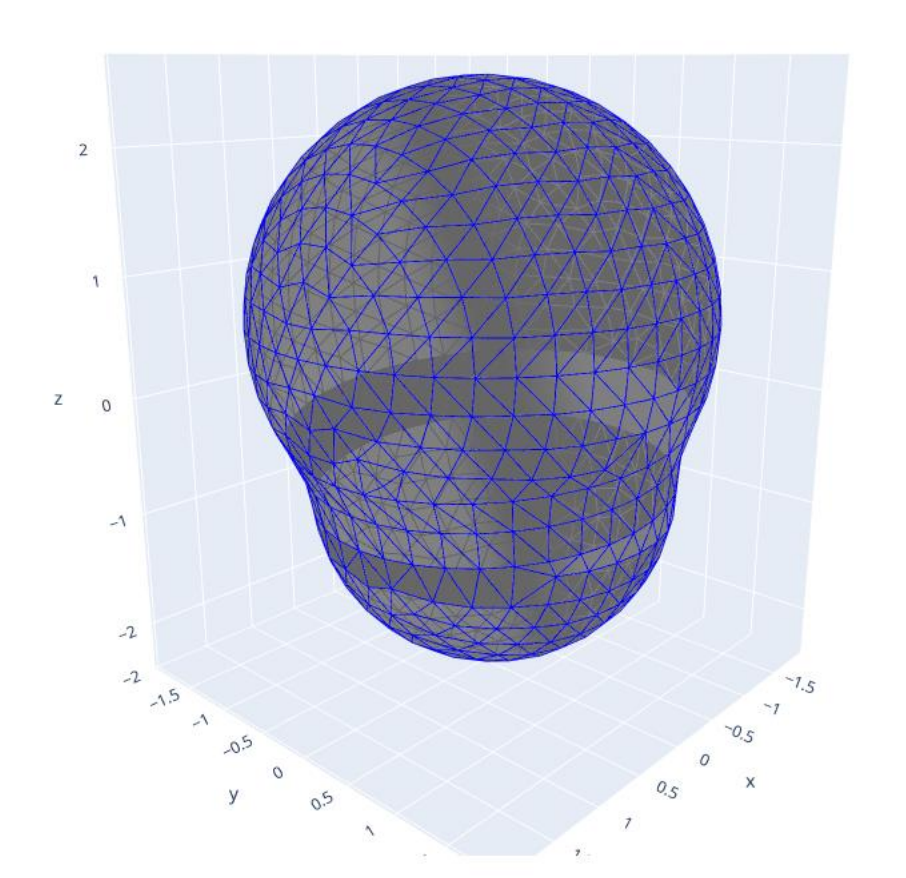


Loss History of NN2, Iterations: 10000, Loss: 7.787e-05
Loss_XN
Loss_NN
Loss_R
Loss_D
Loss_K
Loss_Iu
Loss_Id
Loss_E
$\mathcal{L}: Losses$
$n: iterations$

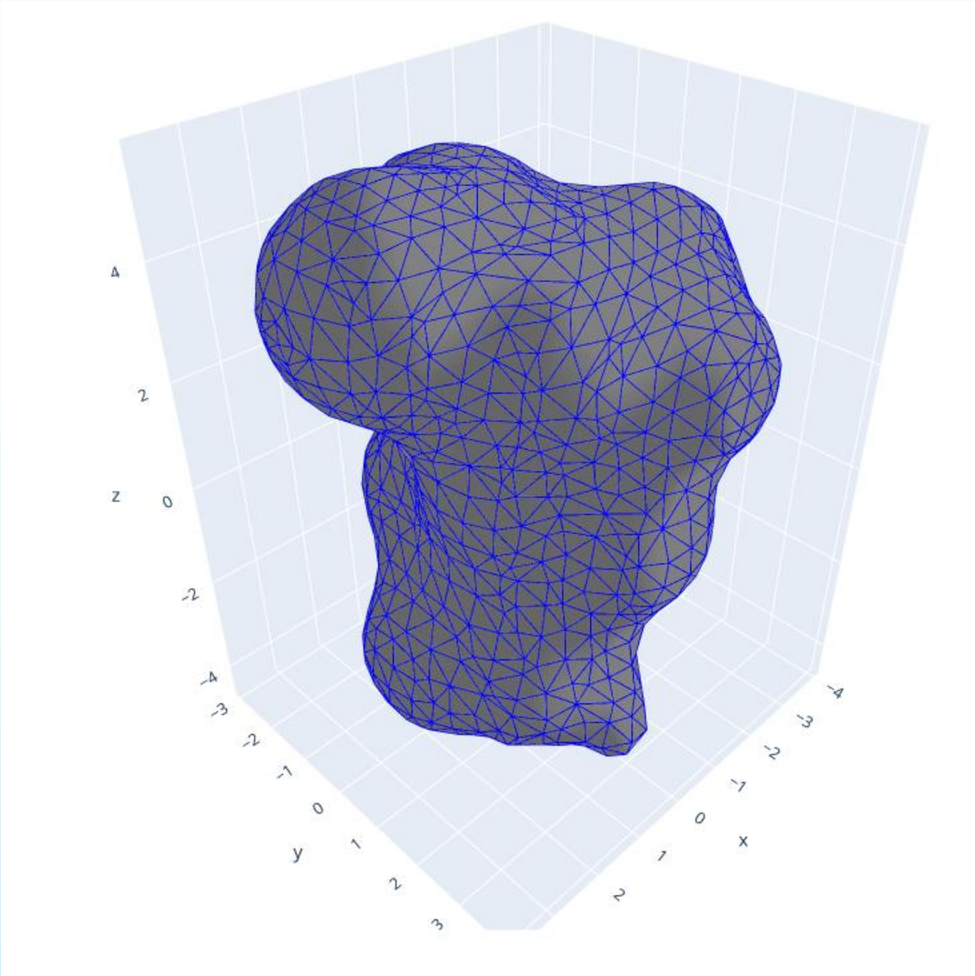


Solution G_{solv} of PDE, Iterations: 10000, G_solv: -33.43 kcal/kmol
G_solv
G_{solv}
$n: iterations$

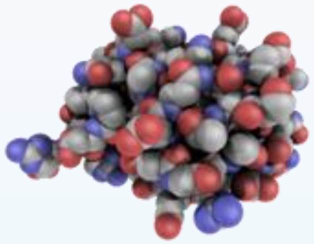
Methanol



Arginine



Conclusions



- **Fourier features** and **weights balancing** algorithm are **needed** for good results.
- Point data loss term is **required** for quick convergence.
- Experimental loss term (ϕ_{ENS}) **improve**s convergence
- Gauss Law loss term makes the solution to diverge. (Integral loss term?).
- Full batch approach works fine to this problem. Samples or multiple batches can be tested.
- Bigger molecules increase complexity and reduces the precision of the XPINN solver.
- More tests are needed. Will be useful for “big” molecules?

References

1) Baker, N. A. (2004). Poisson--Boltzmann methods for biomolecular electrostatics. Methods in enzymology, 383, 94--118. Elsevier.
2) Fogolari, F., Brigo, A., & Molinari, H. (2002). The Poisson--Boltzmann equation for biomolecular electrostatics: a tool for structural biology. Journal of Molecular Recognition, 15(6), 377--392. Wiley Online Library.
3) Yoon, B. J., & Lenhoff, A. M. (1990). A boundary element method for molecular electrostatics with electrolyte effects. Journal of Computational Chemistry, 11(9), 1080--1086. Wiley Online Library.
4) Lee, A., Geng, W., & Zhao, S. (2021). Regularization methods for the Poisson-Boltzmann equation: comparison and accuracy recovery. Journal of Computational Physics, 426, 109958. Elsevier.
5) Roux, B., & Simonson, T. (1999). Implicit solvent models. Biophysical chemistry, 78(1-2), 1--20. Elsevier.
6) Kellogg, O. D. (1953). Foundations of potential theory. Courier Corporation.
7) Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., & Yang, L. (2021). Physics-informed machine learning. Nature Reviews Physics, 3(6), 422--440. Nature Publishing Group UK London.
8) Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational physics, 378, 686--707. Elsevier.
9) Kharazmi, E., Zhang, Z., & Karniadakis, G. E. (2019). Variational physics-informed neural networks for solving partial differential equations. arXiv preprint arXiv:1912.00873.
10) Cai, S., Mao, Z., Wang, Z., Yin, M., & Karniadakis, G. E. (2021). Physics-informed neural networks (PINNs) for fluid mechanics: A review. Acta Mechanica Sinica, 37(12), 1727--1738. Springer.

References

11) Wang, S., Sankaran, S., Wang, H., & Perdikaris, P. (2023). An Expert's Guide to Training Physics-informed Neural Networks. arXiv preprint arXiv:2308.08468.
12) Sun, Y., Sun, Q., & Qin, K. (2021). Physics-Based Deep Learning for Flow Problems. Energies, 14(22), 7760. MDPI.
13) Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., & Mahoney, M. W. (2021). Characterizing possible failure modes in physics-informed neural networks. Advances in Neural Information Processing Systems, 34, 26548--26560.
14) Cuomo, S., Di Cola, V. S., Giampaolo, F., Rozza, G., Raissi, M., & Piccialli, F. (2022). Scientific machine learning through physics--informed neural networks: Where we are and what's next. Journal of Scientific Computing, 92(3), 88. Springer.
15) Shin, Y., Darbon, J., & Karniadakis, G. E. (2020). On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs. arXiv preprint arXiv:2004.01806.
16) Wu, S., Zhu, A., Tang, Y., & Lu, B. (2022). On convergence of neural network methods for solving elliptic interface problems. arXiv preprint arXiv:2203.03407.
17) Teng, Y., Zhang, X., Wang, Z., & Ju, L. (2022). Learning green's functions of linear reaction-diffusion equations with application to fast numerical solver. In Mathematical and Scientific Machine Learning (pp. 1--16). PMLR.
18) Li, S., & Feng, X. (2022). Dynamic Weight Strategy of Physics-Informed Neural Networks for the 2D Navier--Stokes Equations. Entropy, 24(9), 1254. MDPI.
19) Mills, E., & Pozdnyakov, A. (2022). Stochastic Scaling in Loss Functions for Physics-Informed Neural Networks. arXiv preprint arXiv:2208.03776.
20) Hu, Z., Jagtap, A. D., Karniadakis, G. E., & Kawaguchi, K. (2021). When do extended physics-informed neural networks (XPINNs) improve generalization? arXiv preprint arXiv:2109.09444.

References

21) Jagtap, A. D., & Karniadakis, G. E. (2021). Extended Physics-informed Neural Networks (XPINNs): A Generalized Space-Time Domain Decomposition based Deep Learning Framework for Nonlinear Partial Differential Equations. In AAAI spring symposium: MLPS (Vol. 10).

22) Sun, J., Liu, Y., Wang, Y., Yao, Z., & Zheng, X. (2023). BINN: A deep learning approach for computational mechanics problems based on boundary integral equations. Computer Methods in Applied Mechanics and Engineering, 410, 116012. Elsevier.

23) Lin, G., Hu, P., Chen, F., Chen, X., Chen, J., Wang, J., & Shi, Z. (2021). BINet: learning to solve partial differential equations with boundary integral networks. arXiv preprint arXiv:2110.00352.

24) LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436--444. Nature Publishing Group UK London.

25) Hornik, K., Stinchcombe, M., & White, H. (1990). Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. Neural networks, 3(5), 551--560. Elsevier.

26) He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition, 770--778.

27) Jagtap, A. D., Kharazmi, E., & Karniadakis, G. E. (2020). Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. Computer Methods in Applied Mechanics and Engineering, 365, 113028. Elsevier.

28) Yu, B., Pletka, C. C., Pettitt, B. M., & Iwahara, J. (2021). De novo determination of near-surface electrostatic potentials by NMR. Proceedings of the National Academy of Sciences, 118(25), e2104020118. National Acad Sciences.

29) Developers, TensorFlow. TensorFlow. Zenodo, 2022.

30) Martín Achondo. Poisson-Boltzmann-Equation-Simulation-Using-XPINNs. 2023. [En línea]. Disponible en: https://github.com/MartinAchondo/Poisson-Boltzmann-Equation-Simulation-Using-XPINNs.

UNIVERSIDAD TECNICA FEDERICO SANTA MARIA

Agencia Nacional de Investigación y Desarrollo

XPINN Solver for 3D Poisson-Boltzmann Equation

Martín Achondo Mercado

Christopher Cooper

Jehanzeb Chaudhry

January 16, 2024