Uso de *Physics Informed Neural Networks* para Modelar el Potencial Eléctrico de Macromoléculas en un Medio Polarizable a Partir de la Ecuación de Poisson-Boltzmann

Martín Achondo Mercado*
Departamento de Ingeniería Mecánica, Universidad Técnica Federico Santa María, Valparaíso
Received September 8, 2023; E-mail: martin.achondo@sansano.usm.cl

Abstract: Las publicaciones respecto al método de PINNs han crecido bastante en el último par de años, evidenciando su capacidad de resolución de ecuaciones diferenciales parciales aplicadas en dinámica de fluidos, mecánica de sólidos, combustión, entre otros. Sin embargo, no se han visto avances en la resolución de la ecuación de Poisson-Boltzmann para modelar el potencial electrostático de macromoléculas, área donde se estima que la tecnología de PINNs puede ser muy bien utilizada. Es por esto que, en este informe, se presenta un avance de la investigación en el ámbito de PINNs, *Physics Informed Neural Networks*, para la resolución de la ecuación de Poisson-Boltzmann aplicado a macromoléculas en un medio polarizable.

Hasta el momento, se tiene un código de elaboración propia que resuelve esta ecuación (en su forma normal y regularizada) en 3 dimensiones. Para casos simples, el método logra replicar la solución del problema con buena precisión. Sin embargo, faltan pruebas por realizar para alcanzar la modelación de macromoléculas a escalas de la realidad, lo cual en el corto plazo es posible de alcanzar.

Keywords

Poisson-Boltzmann equation, PINNs, XPINNs

1 Introducción

En el siguiente documento se detalla el avance en la materia de PINNs, *Physics Informed Neural Networks*, aplicado para resolver la ecuación de Poisson-Boltzmann para macromoléculas en un solvente. En el informe se detallan 3 formas de poder resolver esta ecuación: utilizando el método de *Deep Collocation Method* a partir de una formulación de XPINNs para resolver la forma normal y regularizada de la ecuación; y el método de *Deep Boundary Integral Method* utilizando PINNs para resolver la forma integral en la superficie de la macromolécula.

2 Justificación y Objetivos

2.1 Justificación

El método de PINNs tiene un gran potencial en la resolución de ecuaciones diferenciales parciales, y una ventaja que los métodos tradicionales no tienen. La forma en que está diseñado el método de PINNs (red neuronal en donde se incluye las ecuaciones físicas en la función de pérdida), permite incluir datos experimentales en la optimización del modelo. Esta gran ventaja ayuda a reducir la diferencia entre el modelo y la realidad, permitiendo combinar ambos campos para entregar una solución del problema. Es por esto que, se quiere evaluar la capacidad de los métodos PINNs para resolver la ecuación de Poisson-Boltzmann aplicado a macromoléculas en un medio polarizable, el cual corresponde a un campo en donde PINNs no se ha aplicado, pero podría tener un gran impacto al compararse y/o combinarse con los métodos actuales de FEM y BEM.

2.2 Objetivo General

Resolver la ecuación de Poisson-Boltzmann bajo distintas geometrías y distribuciones de carga utilizando los métodos basados en PINNs.

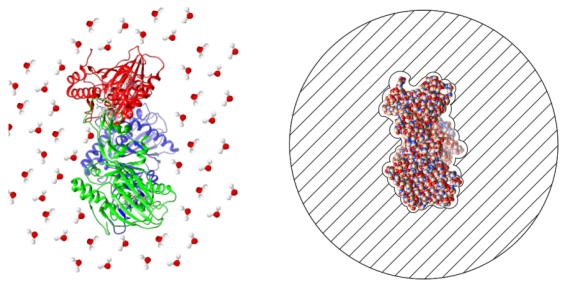
2.3 Objetivos Específicos

- Programar distintas arquitecturas de redes neuronales artificiales y revisar la sensibilidad respecto a sus hiperparámetros.
- Implementar, y evaluar el error, en las distintas formas que puede adoptar la función de pérdida, la cual contiene información de la ecuación diferencial a resolver.
- Verificar la validez del método para macromoléculas reales.
- Evaluar el uso del método al incorporar datos experimentales en la función de pérdida.

3 Marco Teórico

3.1 Método del Solvente Implícito

En la modelación de la electrostática para la solvatación de biomoléculas, existen diversos métodos para representar la molécula y el solvente. A grandes rasgos, se agrupan entre los modelos de solvente explícito y solvente implícito. La gran diferencia entre ambos métodos radica en el detalle de la representación del solvente.



(a) Solvente explícito (b) Solvente implícito

Figura 1. Representación de esquemas para los modelos del solvente

Como se visualiza en la imagen anterior, fig.1, el método de solvente explícito representa de manera detallada cada molécula que es parte del solvente, para así obtener la interacción con la macromolécula de interés. Por otra parte, el método de solvente implícito representa cada una de las moléculas del solvente como un medio continuo polarizable. Esto, a pesar de que puede resultar contra intuitivo por la naturaleza discreta del solvente, tiene su justificación al ser una descripción que busca representar el comportamiento promedio, buscando las fluctuaciones macroscópicas por sobre las microscópicas.[1]

Para el modelo de solvente implícito, el equilibrio entre la interacción de la macromolécula y el solvente ocurre cuando se minimiza la energía libre del complejo, en donde las concentraciones siguen una distribución de Boltzmann. Debido a la reducción de costo computacional que ofrece este esquema, y sumando el hecho de que representa con buenos resultados situaciones de esta naturaleza, se utilizará el método de solvente implícito.

3.1.1 Ecuación de Poisson-Boltzmann

Para poder obtener el potencial electrostático a partir del modelo de solvente implícito, se utilizará la ecuación de Poisson-Boltzmann linealizada.[2,3] Para el caso de una macromolécula en un solvente, se tendrá una ecuación para cada dominio (Ω_1 dentro de la macromolécula, Ω_2 en el solvente, y Γ como interfaz entre ambos).

$$\begin{cases} \nabla^2\phi_1 = -\dfrac{1}{\epsilon_1}\displaystyle\sum_k q_k\delta(\mathbf{x}_k) & \mathbf{x}\in\Omega_1 \\ \nabla^2\phi_2 = \kappa^2\phi_2 & \mathbf{x}\in\Omega_2 \end{cases} \tag{1}$$

La relación en la interfaz queda como:

$$\begin{cases} \phi_1 = \phi_2 & \mathbf{x}\in\Gamma \\ \epsilon_1\dfrac{\partial\phi_1}{\partial n} = \epsilon_2\dfrac{\partial\phi_2}{\partial n} & \mathbf{x}\in\Gamma \end{cases} \tag{2}$$

En donde ϵ_1 y ϵ_2 corresponden a la constante dieléctrica de cada medio, κ al inverso de la longitud de Debye, y q las cargas puntuales presentes en la macromolécula. Al resolver este sistema de ecuaciones diferenciales, se obtiene el potencial electrostático ϕ en todo el dominio. Comúnmente se utiliza el método de FEM para esta resolución.

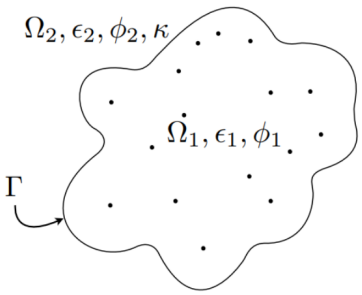


Figura 2. Esquema del modelo de solvente implícito

Adicionalmente, se cuentan con trabajos experimentales que buscan encontrar el potencial electrostático cerca de la superficie de la macromolécula,[4] y que validan este modelo.

3.1.2 Ecuación de Poisson-Boltzmann Regularizada

Debido a las singularidades existentes producto de las cargas puntuales en la macromolécula, eq.1, se puede realizar la siguiente transformación para obtener el set de ecuaciones regularizadas. Se detallará el más simple, pero existen distintos esquemas. El esquema a implementar considera descomponer el potencial electrostático en su componente singular y regular (campo de reacción) a lo largo de todo el dominio Ω.[5]

$$\phi = \mathcal{G} + \phi^{RF} \tag{3}$$

En donde la función $\mathcal{G}$ toma la siguiente forma:

$$\mathcal{G}(\mathbf{x}) = \frac{1}{4\pi\epsilon_1}\sum_k \frac{q_k}{|\mathbf{x}-\mathbf{x}_k|} \tag{4}$$

La ecuación de Poisson-Boltzmann bajo esta regularización queda de la siguiente manera:

$$\begin{cases} \nabla^2\phi_1^{RF} = 0 & \mathbf{x}\in\Omega_1 \\ \nabla^2\phi_2^{RF} = \kappa^2(\phi_2^{RF}+\mathcal{G}) & \mathbf{x}\in\Omega_2 \end{cases} \tag{5}$$

En donde la relación en la interfaz queda como:

$$\begin{cases} \phi_1^{RF} = \phi_2^{RF} & \mathbf{x}\in\Gamma \\ \epsilon_1\left(\dfrac{\partial\phi_1^{RF}}{\partial n}+\dfrac{\partial\mathcal{G}}{\partial n}\right) = \epsilon_2\left(\dfrac{\partial\phi_2^{RF}}{\partial n}+\dfrac{\partial\mathcal{G}}{\partial n}\right) & \mathbf{x}\in\Gamma \end{cases} \tag{6}$$

Notar que este set de ecuaciones permite obtener el componente regular del potencial electrostático ϕ^{RF}, por lo que posterior a la resolución, basta sumar nuevamente el componente singular, $\mathcal{G}$.

3.1.3 Ecuación de Poisson-Boltzmann en Forma Integral en el Borde

Pese a que se puede resolver directamente la eq.1 para obtener el potencial electrostático, existen métodos que resuelven su formulación integral en la interfaz. Lo anterior permite resolver la ecuación solo en la interfaz, sin tener que trabajar con todo el dominio, comúnmente utilizando el método de BEM.[6]

La formulación integral de la eq.1 en la interfaz, se puede obtener haciendo uso de la Segunda y Tercera Identidad de Green.[7] Considerando que $\mathbf{x}^s \in \mathcal{S} \subset \Gamma$, se presenta el set de ecuaciones integrales en la interfaz Γ.

$$\begin{aligned} \phi_1(\mathbf{x}^s) &= \frac{1}{2\pi\epsilon_1}\sum_k \frac{q_k}{|\mathbf{x}^s-\mathbf{x}_k|} \\ &\quad + \frac{1}{2\pi}\oint_\Gamma \frac{\partial\phi_1}{\partial n}\frac{1}{|\mathbf{x}^s-\mathbf{x}'|}\,\mathrm{d}S(\mathbf{x}') \\ &\quad - \frac{1}{2\pi}\oint_\Gamma \phi_1\frac{\partial}{\partial n}\left(\frac{1}{|\mathbf{x}^s-\mathbf{x}'|}\right)\mathrm{d}S(\mathbf{x}') \\ \phi_2(\mathbf{x}^s) &= -\frac{1}{2\pi}\oint_\Gamma \frac{\partial\phi_2}{\partial n}\frac{\exp(-\kappa|\mathbf{x}^s-\mathbf{x}'|)}{|\mathbf{x}^s-\mathbf{x}'|}\,\mathrm{d}S(\mathbf{x}') \\ &\quad + \frac{1}{2\pi}\oint_\Gamma \phi_2\frac{\partial}{\partial n}\left(\frac{\exp(-\kappa|\mathbf{x}^s-\mathbf{x}'|)}{|\mathbf{x}^s-\mathbf{x}'|}\right)\mathrm{d}S(\mathbf{x}') \end{aligned} \tag{7}$$

3.1.4 Solución Analítica - Caso Simple

Para validar los métodos a utilizar, se comparará lo obtenido con la siguiente solución analítica. Considerando solo una carga puntual q en el centro de una macromolécula esférica de radio R, se puede obtener una expresión analítica de la eq.1 como:

$$\begin{aligned} \phi_1(r) &= \frac{q}{4\pi}\left(\frac{1}{\epsilon_1 r}-\frac{1}{\epsilon_1 R}+\frac{1}{\epsilon_2(1+\kappa R)R}\right) \\ \phi_2(r) &= \frac{q}{4\pi}\frac{\exp(-\kappa(r-R))}{\epsilon_2(1+\kappa R)r} \end{aligned} \tag{8}$$

3.2 Redes Neuronales Artificiales

Se puede decir que las redes neuronales artificiales, *Artificial Neural Networks*, o simplemente ANN, corresponden principalmente a una composición de funciones no lineales, las cuales se van operando sobre un conjunto de parámetros θ. La forma de estas funciones y la cantidad de parámetros en el conjunto θ depende exclusivamente de la arquitectura de la red neuronal (*Deep Learning*).[8]

Para que una ANN reproduzca un resultado esperado, es necesario entrenarla bajo un proceso de minimización. Para esto, se plantea una función de pérdida $\mathcal{L}$ (comúnmente como *Mean Squared Error*) que depende de los inputs de la red neuronal y los parámetros θ de la ANN. De esta forma, entrenar la red para reproducir un resultado esperado, se convierte en un problema de minimización para encontrar los parámetros θ^* óptimos:

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\,\mathcal{L}(\theta) \tag{9}$$

Normalmente, este problema de minimización es llevado a cabo con el método de optimización de descenso de gradiente ADAM. Este algoritmo calcula la media móvil exponencial de gradientes y gradientes cuadrados de la función de pérdida. De esta manera, se puede regular la tasa de aprendizaje en cada iteración, y así, acelerar la convergencia.

La forma de la función de pérdida, $\mathcal{L}$, va a depender del contexto del problema y su aplicación. Notar que las ANN pueden ser utilizadas en diversos contextos como: Procesamiento de imágenes, procesamiento de texto, reconocimiento facial, análisis de señales, y también para resolver ecuaciones diferenciales, entre otros. La gran convergencia que tiene este método para problemas multidimensionales junto con el teorema universal de aproximación de ANN,[9] implica que puede ser utilizado en un gran abanico de contextos.

En las secciones posteriores se detallan 2 tipos de arquitecturas de redes neuronales que se utilizarán.

3.2.1 Fully Connected Neural Network

La arquitectura FCNN, *Fully Connected Neural Network*, se basa en conectar L capas de m perceptrones (neuronas) cada una, de manera que todos los perceptrones de cada capa quedan conectados con todos los de la capa siguiente. De esta forma, en cada perceptrón p de la capa i, ocurre la siguiente transformación, fig. 3:

$$h_p^i = f^i\left(\sum_{j=1}^{m} h_j^{i-1} w_{jp}^i + b_p^i\right) \tag{10}$$

En donde w_{jp}^i corresponde al peso de la conexión entre el perceptrón j de la capa $i-1$ y el perceptrón p de la capa i, b_p^i al bias del perceptrón p de la capa i, h_p^i al output del perceptrón p de la capa i, y f una función de activación no lineal. Como función de activación se puede utilizar: tangente hiperbólica, sigmoid o ReLU, entre otras.

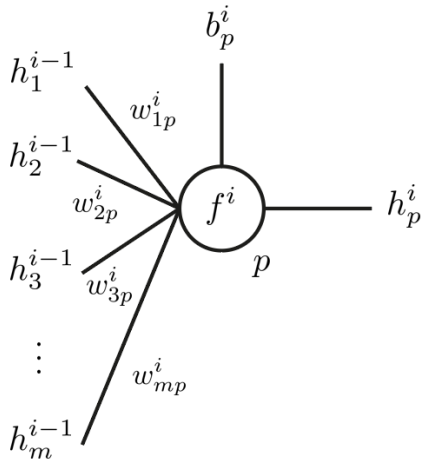


Figura 3. Esquema del perceptrón p de la capa i

Para simplificar la notación anterior, se trabajará con la matriz de pesos $\boldsymbol{W}^i$ y el vector de bias $\boldsymbol{b}^i$ de cada capa i. De esta forma, para las L capas de perceptrones, se podrá formar el conjunto θ de parámetros como:

$$\theta = \{\boldsymbol{W}^1, \boldsymbol{b}^1, \ldots, \boldsymbol{W}^L, \boldsymbol{b}^L, \boldsymbol{U}, \boldsymbol{c}\} \tag{11}$$

Notar que la red tiene como input el vector $\mathbf{x}$ y como output el vector $\boldsymbol{\phi}_\theta$. Lo anterior se puede escribir como:

$$\boldsymbol{\phi}_\theta = \mathcal{N}(\mathbf{x}; \theta) \tag{12}$$

En donde $\mathcal{N}$ corresponde a la ANN.

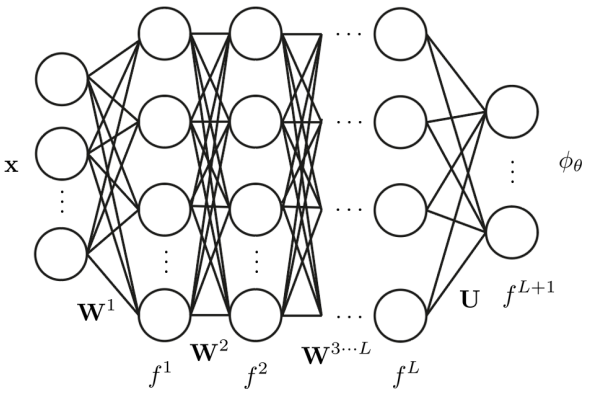


Figura 4. Esquema de FCNN

Un detalle de todas las operaciones que se realizan entre las capas de la ANN, $\mathcal{N}$, se muestra a continuación:

$$\begin{cases} \boldsymbol{h}^1 = f^1(\mathbf{x}\boldsymbol{W}^1 + \boldsymbol{b}^1) & i = 1 \text{ capa entrada} \\ \boldsymbol{h}^i = f^i(\boldsymbol{h}^{i-1}\boldsymbol{W}^i + \boldsymbol{b}^i) & 2 \leq i \leq L \text{ capas ocultas} \\ \phi_\theta = f^{L+1}(\boldsymbol{h}^L\boldsymbol{U} + \boldsymbol{c}) & i = L+1 \text{ salida} \end{cases} \tag{13}$$

Debido a que se tiene una fórmula analítica para la aproximación que genera la red neuronal, ϕ_θ, se pueden calcular sus derivadas fácilmente. Para esto se realiza una diferenciación automática con el algoritmo de "back propagation", el cual corresponde a la utilización eficiente de la regla de la cadena.

3.2.2 Residual Neural Network

La arquitectura ResNet, *Residual Neural Network*, nace dado que la arquitectura FCNN tiene problemas en el entrenamiento para un número elevado de capas ocultas. Lo anterior es debido a que se produce un fenómeno llamado "degradación del gradiente", el cual corresponde al error que aparece al agregar un número de parámetros mayor a los necesarios para obtener la solución de un problema.[10] De esta manera, el gradiente de la función de pérdida no puede ser traspasado correctamente a las primeras capas de la red, lo que genera errores en la solución.

Para resolver el problema anterior, la arquitectura ResNet modifica levemente la arquitectura FCNN. La diferencia radica en que el output de cada capa (o conjuntos de capas) es sumado directamente a la transformación que realiza la capa siguiente, lo cual permite traspasar la información directamente entre capa y capa.

Normalmente, esta arquitectura considera una

primera capa de perceptrones y luego L bloques sucesivos de 2 capas cada uno, más una última capa de perceptrones al término de la red. De esta manera, la información se va pasando entre cada bloque. Las ecuaciones en cada bloque pueden ser escritas como:

$$\begin{cases} \boldsymbol{h}^0 = f^0(\mathbf{x}\boldsymbol{W}^0 + \boldsymbol{b}^0) \\ \qquad\qquad i = 0 \text{ capa entrada} \\ \boldsymbol{h}^i = \boldsymbol{h}^{i-1} + f_2^i\big(f_1^i(\boldsymbol{h}^{i-1}\boldsymbol{W}_1^i + \boldsymbol{b}_1^i)\boldsymbol{W}_2^i + \boldsymbol{b}_2^i\big) \\ \qquad\qquad 1 \leq i \leq L \text{ bloques ocultos} \\ \boldsymbol{h}^{L+1} = f^{L+1}(\boldsymbol{h}^L\boldsymbol{W}^{L+1} + \boldsymbol{b}^{L+1}) \\ \qquad\qquad i = L+1 \text{ capa salida} \\ \phi_\theta = f^{L+2}(\boldsymbol{h}^{L+1}\boldsymbol{U} + \boldsymbol{c}) \\ \qquad\qquad i = L+2 \text{ salida} \end{cases} \tag{14}$$

Este cambio en la arquitectura permite utilizar mayor cantidad de bloques, que tiene relación con la cantidad de capas ocultas, y así permite una mejor convergencia en la solución.

4 Estado del Arte (PINNs)

En esta sección se describirá el estado del arte respecto a los métodos que utilizan Redes Neuronales Profundas, *Deep Neural Networks*, para resolver problemas de valor de frontera. Estas redes corresponden a *Physics Informed Neural Networks*, PINNs, las cuales se caracterizan por incluir el problema de valor de frontera en la función de pérdida de la red neuronal, para luego luego obtener su solución.[11–13] La utilización de PINNs para la resolución de ecuaciones diferenciales parciales trae grandes ventajas, tales como: rápida convergencia, sencilla implementación, no necesidad de una malla, la capacidad de incluir datos experimentales e incluso obtener parámetros desconocidos de la PDE, entre otros. Este tema ha ganado bastante popularidad recientemente, siendo utilizado para resolver problemas de dinámica de fluidos (Navier-Stokes), mecánica de solidos, transferencia de calor,[14–16] e incluso en problemas inversos. Sin embargo, al ser un tema nuevo, implica que existe mucho por investigar para que este método pueda competir y/o acoplarse a los métodos tradicionales; de FEM, FVM, entre otros.[17] A continuación, se dará una breve descripción de los métodos conocidos como PINNs.

Normalmente, un problema de valor de contorno estacionario puede ser planteado de la siguiente forma:

$$\begin{cases} \mathcal{D}\phi = f(\mathbf{x}) & \mathbf{x} \in \Omega \\ \mathcal{B}\phi = g(\mathbf{x}) & \mathbf{x} \in \partial\Omega \end{cases} \tag{15}$$

En donde $\mathcal{D}$ es un operador diferencial actuando sobre $\boldsymbol{\phi}$ en todo el dominio, y $\mathcal{B}$ un operador diferencial que actúa sobre $\boldsymbol{\phi}$ en la frontera, siendo $\boldsymbol{\phi}$ un campo vectorial.

El principal objetivo de los métodos PINNs es incorporar estas ecuaciones en la función de pérdida de una red neuronal, de tal forma que, luego del proceso de minimización, la red neuronal pueda replicar la solución de la ecuación diferencial.

Los distintos métodos PINNs se diferencian en la forma que adopta la función de pérdida $\mathcal{L}$.[18] Algunos métodos PINNs se detallan a continuación[1]:

- DCM (*Deep Collocation Method*): Utilizan los residuales de la eq.15 en puntos del dominio para formar la función de pérdida. Este método es uno de los métodos PINN más utilizados debido a su sencilla implementación y buenos resultados.[11,12] Además, se han realizado avances en descomposición de dominios, XPINNs,[19,20] *Extended Physics Informed Neural Networks*, que buscan resolver problemas en dominios complejos utilizando más de una red neuronal (1 por dominio) y CPINNs,[21] *Conservative Physics Informed Neural Networks*.

- DVM (*Deep Variational Method*): Utiliza la formulación variacional de la eq.15 para formar la función de pérdida. Este método a llevado a una rama nueva de PINN, llamado VPINN,[22] *Variational Physics Informed Neural Networks*.

- DBIM (*Deep Boundary Integral Method*): Utiliza la formulación integral en la frontera de la eq.15 para formar la función de pérdida. Para este método, se han desarrollado ramas como BINN o BINet,[23,24] *Boundary Integral Neural Networks*.

Entonces, de manera general, todos los métodos buscan encontrar una solución aproximada de la solución exacta del problema ϕ, ver fig.5:

$$\phi \approx \phi_\theta = \mathcal{N}(\mathbf{x}; \theta) \tag{16}$$

[1] Los nombres de los métodos fueron inferidos al recolectar las diferentes variantes de PINNs en la literatura.

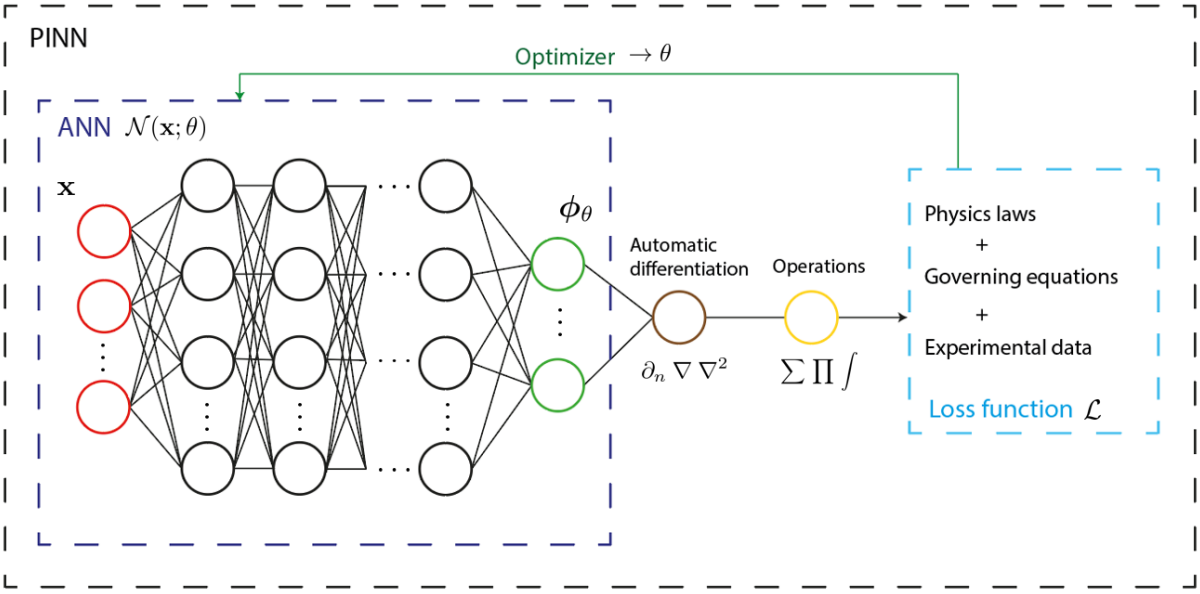


Figura 5. Esquema de métodos PINN

4.1 Deep Collocation Method

El método DCM es uno de los más comunes de los métodos PINN, ya que utiliza directamente el residual de la ecuación diferencial para formular la función a minimizar.

Para obtener esta solución, se formula la siguiente función de pérdida, la cual será minimizada en los puntos colocados dados por el conjunto $\mathcal{S}$:

$$\mathcal{L}(\theta;\mathcal{S}) = w_{pde}\mathcal{L}_{pde}(\theta;\mathcal{S}_{pde}) + w_{bc}\mathcal{L}_{bc}(\theta;\mathcal{S}_{bc}) + \\ + w_{data}\mathcal{L}_{data}(\theta;\mathcal{S}_{data}) \tag{17}$$

En la función de pérdida, el primer término $\mathcal{L}_{bc}$ corresponde al componente que incluye las condiciones de borde, y el segundo término $\mathcal{L}_{pde}$ corresponde al residual de la ecuación diferencial. Notar además que estos términos se multiplican por respectivos pesos para modificar la dominancia sobre la función de pérdida total. Los pesos pueden ser fijados o pueden ser aprendidos en el proceso de iteración para maximizar la convergencia.[25,26] Adicionalmente, se puede añadir un término $\mathcal{L}_{data}$ para contabilizar datos experimentales disponibles. Vale destacar que este término no es estrictamente necesario para la convergencia del modelo.

El conjunto de puntos $\mathcal{S}$ incluirá los subconjuntos de puntos en los cuales cada término de la función de pérdida es evaluado, $\mathcal{S} = \mathcal{S}_{pde} \cup \mathcal{S}_{bc} \cup \mathcal{S}_{data}$.

$$\mathcal{L}_{pde}(\theta;\mathcal{S}_{pde}) = \frac{1}{N_{pde}}\|\mathcal{D}\phi_\theta - f(\mathbf{x})\|^2$$
$$\mathcal{L}_{bc}(\theta;\mathcal{S}_{bc}) = \frac{1}{N_{bc}}\|\mathcal{B}\phi_\theta - g(\mathbf{x})\|^2 \tag{18}$$
$$\mathcal{L}_{data}(\theta;\mathcal{S}_{data}) = \frac{1}{N_{data}}\|\phi_\theta - \phi_{data}\|^2$$

Este método también puede ser utilizado para resolver sistemas de PDEs utilizando el método de XPINN. Para esto, se utiliza una ANN para cada dominio y se incluye un término en la función de pérdida en la interfaz de los dominios.

De modo esquemático, para 2 dominios, la aproximación de la solución queda como:

$$\phi \approx \phi_\theta = \begin{cases} \mathcal{N}_1(\mathbf{x};\theta^1) & \mathbf{x} \in \Omega_1 \\ \mathcal{N}_2(\mathbf{x};\theta^2) & \mathbf{x} \in \Omega_2 \end{cases} \tag{19}$$

Se utilizará la notación de ϕ_θ^1 y ϕ_θ^2 para los distintos dominios. La función de pérdida sigue la misma forma detallada anteriormente, en donde se incluyen las condiciones de borde y el residual de la ecuación. La diferencia ocurre dado que se agrega el término en la interfaz, $\mathcal{L}_I^j$ que es calculado en los puntos del conjunto $\mathcal{S}_I$. Entonces, para cada red j, su función de pérdida queda como:

$$\mathcal{L}^j = w_{bc}\mathcal{L}_{bc}^j + w_{pde}\mathcal{L}_{pde}^j + w_{data}\mathcal{L}_{data}^j + w_I\mathcal{L}_I^j \tag{20}$$

En donde la función de pérdida del dominio j en la interfaz se descompone como:

$$\mathcal{L}_I^j(\theta;\mathcal{S}_I) = \frac{1}{N_I}\left\|\mathcal{C}_1\phi_\theta^j - \overline{\mathcal{C}\phi_\theta}\right\|^2 \tag{21}$$

En donde el operador $\mathcal{C}$ actúa sobre ϕ_θ en la interfaz de ambos dominios. Este operador puede contabilizar la continuidad en la función, o su gradiente, dependiendo del set de ecuaciones a resolver. La solución de cada dominio en la interfaz debe aproximarse al promedio $\overline{\mathcal{C}\phi_\theta}$.

4.2 Deep Variational Method

De manera general, se puede obtener la forma débil de la eq.15 multiplicando la ecuación por una función de ponderación v, e integrando sobre todo el dominio Ω:

$$\int_\Omega v\mathcal{D}\phi_\theta \,\mathrm{d}V = \int_\Omega v f(\mathbf{x}) \,\mathrm{d}V \tag{22}$$

De esta manera, la función de pérdida de la ANN corresponderá al residual de la ecuación anterior. Notar que seleccionar v como delta de Dirac convierte el problema en DCM.

$$\mathcal{L}(\theta) = \int_\Omega v\mathcal{D}\phi_\theta \,\mathrm{d}V - \int_\Omega v f(\mathbf{x}) \,\mathrm{d}V \tag{23}$$

Una práctica comúnmente empleada es reducir el requerimiento de continuidad de la función ϕ_θ, lo que se aplica integrando por partes y utilizando el teorema de la divergencia. De esta manera, la función de pérdida queda como:

$$\begin{aligned}\mathcal{L}(\theta) = \int_\Omega \mathcal{I}v\mathcal{A}\phi_\theta \,\mathrm{d}V - \int_\Omega v f(\mathbf{x}) \,\mathrm{d}V \\ - \oint_{\partial\Omega} v\mathcal{M}\phi_\theta \,\mathrm{d}\boldsymbol{S}\end{aligned} \tag{24}$$

En donde $\mathcal{I}, \mathcal{A}, \mathcal{M}$ corresponden a operadores diferenciales de menor orden que $\mathcal{D}$. Es importante destacar que para calcular esta función de pérdida, se necesitará una cuadratura para evaluar las integrales. Esto implica que los puntos de evaluación no deben ser elegidos al azar dado que serán los mismos a utilizar en la cuadratura.

4.3 Deep Boundary Integral Method

En el caso de que se pueda obtener la formulación integral de la eq.15 con los teoremas de Green, esta puede ser utilizada para formar la función de pérdida. Considerando G como la función de Green de la eq.15, se obtiene la siguiente expresión en el borde del dominio $\partial\Omega$, con $\mathbf{x} \in \mathcal{S} \subset \partial\Omega$

$$\begin{aligned}\frac{1}{2}\phi_\theta(\mathbf{x}) = \oint_{\partial\Omega} G(\mathbf{x},\mathbf{x}')\partial_n\phi_\theta(\mathbf{x}') \,\mathrm{d}S \\ - \oint_{\partial\Omega} \partial_n G(\mathbf{x},\mathbf{x}')\phi_\theta(\mathbf{x}') \,\mathrm{d}S \\ + \int_\Omega G(\mathbf{x},\mathbf{x}')f(\mathbf{x}') \,\mathrm{d}V\end{aligned} \tag{25}$$

En donde el primer término corresponde al *single-layer potential*, el segundo al *double-layer potential* y el tercero al *Newton potential*. Para mayor facilidad, se utilizará la notación de operadores como: $\mathcal{V}, \mathcal{K}, \mathcal{T}$ para los 3 potenciales respectivamente. Además, notar que la integral del potencial de Newton puede ser calculada analíticamente, por ende, la incógnita queda solo en el borde del dominio.

De esta manera, la función de pérdida puede ser escrita como:

$$\mathcal{L}(\theta;\mathcal{S}) = \frac{1}{N}\left\|\frac{1}{2}\phi_\theta - \mathcal{V}\phi_\theta + \mathcal{K}\phi_\theta - \mathcal{T}f\right\|^2 \tag{26}$$

Este método requiere de una colección de puntos $\mathcal{S}$ en donde se evalúa la función de pérdida. Adicionalmente, se necesitará un método de cuadratura para evaluar las integrales en el borde del dominio.

5 Metodología

En esta sección se detallará como se aplicarán los métodos de DCM y DBIM para resolver la ecuación de Poisson-Boltzmann. Se utiliza la notación de ϕ^1 y ϕ^2 para el dominio interno y externo respectivamente. El método de DVM no será utilizado en esta investigación.

5.1 DCM Aplicado a PBE

El método de *Deep Collocation Method* será aplicado a la ecuación de Poisson-Boltzmann, eq.1, utilizando XPINNs. Se detallan las funciones de perdida obtenidas:

- Para el dominio interior:

$$\mathcal{L}^1_{pde} = \frac{1}{N_{pde}}\left\|\nabla^2\phi^1_\theta + \frac{1}{\epsilon_1}\sum_k q_k\delta(\mathbf{x}_k)\right\|^2 \tag{27}$$

- Para el dominio exterior:

$$\mathcal{L}^2_{pde} = \frac{1}{N_{pde}} \left\| \nabla^2 \phi^2_\theta - \kappa^2 \phi^2_\theta \right\|^2$$
$$\mathcal{L}^2_{bc} = \frac{1}{N_{bc}} \left\| \phi^2_\theta - \frac{1}{4\pi\epsilon_2} \sum_k \frac{q_k \mathrm{e}^{-\kappa|\mathbf{x}-\mathbf{x}_k|}}{|\mathbf{x}-\mathbf{x}_k|} \right\|^2 \tag{28}$$

- Para la interfaz (del dominio j):

$$\mathcal{L}^j_I = \frac{1}{N_I} \left\| \phi^j_\theta - \overline{\phi}_\theta \right\|^2 + \frac{1}{N_I} \left\| \epsilon_j \partial_n \phi^j_\theta - \overline{\epsilon \partial_n \phi}_\theta \right\|^2 \tag{29}$$

En la interfaz se minimizará hacia el "promedio" debido a que genera una convergencia más suave. Los promedios se calculan como: $\overline{\phi}_\theta = (\phi^1_\theta + \phi^2_\theta)/2$ y $\overline{\epsilon \partial_n \phi}_\theta = (\epsilon_1 \partial_n \phi^1_\theta + \epsilon_2 \partial_n \phi^2_\theta)/2$.

Para evitar la discontinuidad que genera el Delta de Dirac, se aproximó esta función como una función Gaussiana con σ como la desviación:[27]

$$\delta(\mathbf{x}-\mathbf{x}_k) \approx \frac{1}{(2\pi)^{3/2}\sigma^3} e^{-\frac{1}{2\sigma^2}\|\mathbf{x}-\mathbf{x}_k\|^2} \tag{30}$$

Un esquema del algoritmo a seguir se detalla a continuación:

Algoritmo 1 Resolver eq.1 con DCM

Input: Loss function $\mathcal{L}$, Points $\mathcal{S}$, Hyperparameters
  Initialize $\theta^1_0, \mathcal{N}_1, \theta^2_0, \mathcal{N}_2$
  for $k = 1$ to $k = n$ **do**
    Calculate $\phi^1_\theta, \phi^2_\theta$ with $\mathcal{N}_1, \theta^1_k, \mathcal{N}_2, \theta^2_k$
    Calculate $\partial_n \phi^1_\theta, \partial_n \phi^2_\theta$ with AD
    Calculate $\mathcal{D}_1 \phi^1_\theta, \mathcal{D}_2 \phi^2_\theta$ with AD
    Calculate $\mathcal{L}^1$
    Calculate $\mathcal{L}^2$
    Update θ^1_{k+1} with ADAM$(\theta^1_k, \nabla\mathcal{L}^1)$
    Update θ^2_{k+1} with ADAM$(\theta^2_k, \nabla\mathcal{L}^2)$
  end for
Output: Parameters θ^1, θ^2

El mismo método es empleado para formular el método de DCM a la ecuación de Poisson-Boltzmann regularizada, eq.5.

5.2 DBIM Aplicado a PBE

El método de *Deep Boundary Integral Method* será aplicado a la eq.7. Se detalla como queda la función de pérdida para este caso, en donde $\mathbf{x}^s \in \mathcal{S} \subset \Gamma$.

$$\begin{aligned}\mathcal{L}(\theta;\mathcal{S}) = \frac{1}{N} \Bigg\| \phi^1_\theta(\mathbf{x}^s) - \frac{1}{2\pi\epsilon_1} \sum_k \frac{q_k}{|\mathbf{x}^s - \mathbf{x}_k|} \\ - \frac{1}{2\pi} \oint_\Gamma \frac{\partial \phi^1_\theta}{\partial n} \frac{1}{|\mathbf{x}^s - \mathbf{x}'|} \, \mathrm{d}S(\mathbf{x}') \\ + \frac{1}{2\pi} \oint_\Gamma \phi^1_\theta \frac{\partial}{\partial n} \left(\frac{1}{|\mathbf{x}^s - \mathbf{x}'|} \right) \mathrm{d}S(\mathbf{x}') \Bigg\|^2 \\ + \frac{1}{N} \Bigg\| \phi^1_\theta(\mathbf{x}^s) + \frac{1}{2\pi} \frac{\epsilon_1}{\epsilon_2} \oint_\Gamma \frac{\partial \phi^1_\theta}{\partial n} \frac{\exp(-\kappa|\mathbf{x}^s - \mathbf{x}'|)}{|\mathbf{x}^s - \mathbf{x}'|} \, \mathrm{d}S(\mathbf{x}') \\ - \frac{1}{2\pi} \oint_\Gamma \phi^1_\theta \frac{\partial}{\partial n} \left(\frac{\exp(-\kappa|\mathbf{x}^s - \mathbf{x}'|)}{|\mathbf{x}^s - \mathbf{x}'|} \right) \mathrm{d}S(\mathbf{x}') \Bigg\|^2\end{aligned} \tag{31}$$

Un punto importante a recalcar es que la función de pérdida anterior implica resolver solo para ϕ^1_θ. Esto implica a que se puede utilizar solo una ANN para resolver este problema, a diferencia del DCM que utiliza 2 redes neuronales. Además, utiliza el gradiente de la solución, $\partial_n \phi^1_\theta$. Este gradiente puede ser calculado bajo diferenciación automática u obtenerlo como un output independiente de la misma red. Por otra parte, a esta función de pérdida se le puede sumar un término que incluya datos experimentales medidos en la interfaz con su respectivo peso para ponderación.

Por último, de manera general, se detalla un ejemplo del algoritmo de este método.

Algoritmo 2 Resolver eq.7 con DBIM

Input: Loss function $\mathcal{L}$, Points $\mathcal{S}$, Hyperparameters
  Initialize $\theta_0, \mathcal{N}$
  for $k = 1$ to $k = n$ **do**
    Calculate $\phi^1_\theta = \mathcal{N}(\mathbf{x};\theta_k)$
    Calculate $\partial_n \phi^1_\theta$ with AD
    Calculate boundary integrals with cuadrature
    Calculate $\mathcal{L}$
    Update θ_{k+1} with ADAM$(\theta_k, \nabla\mathcal{L})$
  end for
Output: Parameters θ

5.3 Preacondicionamiento

Se implementará un preacondicionamiento de PINN de la siguiente forma: El preacondicionamiento de PINN corresponde a encontrar los parámetros θ que minimicen una función de pérdida distinta a la del problema pero parecida, e idealmente fácil de calcular. Por ejemplo, si se conoce que la solución tendrá un comportamiento similar al de la función ϕ_*, se puede formular la función de pérdida como:

$$\mathcal{L}_{pre} = \frac{1}{N}\|\phi_\theta - \phi_*\|^2 \tag{32}$$

Esta minimización se debe seguir un total de n_{pre} iteraciones. Luego de esto, se puede continuar con la función de pérdida real del problema. Se asume entonces que los parámetros θ sufrirán solo pequeñas modificaciones en este segundo proceso de minimización. Hay que tener en cuenta que este método requiere que el optimizador ADAM sea inicializado nuevamente una vez terminadas las iteraciones de preacondicionamiento.

El preacondicionamiento detallado puede ser utilizado con cualquiera de los métodos PINN descritos en el documento.

6 Avance y Resultados Preliminares

Actualmente, ya se tiene un código de elaboración propia[28] en el repositorio propio de Github utilizando la librería de Tensorflow para la creación y optimización de las redes neuronales.[29] Este código permite resolver cualquier sistema de PDEs utilizando XPINNs, pero principalmente está aplicado a resolver la ecuación de Poisson-Boltzmann en su forma normal y regularizada.

El trayecto para obtener este código comenzó resolviendo distintos problemas, como: ecuación de calor, ecuación de Navier-Stokes, ecuación de Poisson, ecuación de Helmholtz, en 1D y 2D, para finalizar con problemas de interfaz en 3D.

A continuación se presentan los resultados al resolver con XPINNs la eq.1 en su forma normal y regularizada. Se presentan casos para $\epsilon_1 = 1, \epsilon_2 = 80, \kappa = 0.125$, con una macromolécula de radio $R = 1$, y una carga de $q = 1$ en el centro.

Todos los casos presentados utilizan una arquitectura FCNN, con 8 capas ocultas y 200 neuronas por capa. Se utilizó la función $\tanh(x)$ como función de activación. Todas las gráficas fueron obtenidas a las 3000 iteraciones. Se utiliza el optimizador ADAM con una tasa de aprendizaje de $\lambda = 5e-4$. No se incluye el preacondicionamiento en ningún caso.

6.1 Caso normal

Se presentan los resultados al resolver la ecuación de Poisson-Boltzmann, eq.1, con el uso de XPINNs. En la función de pérdida se incluye en un 5% de los puntos (de forma aleatoria), penalización respecto a la solución analítica.

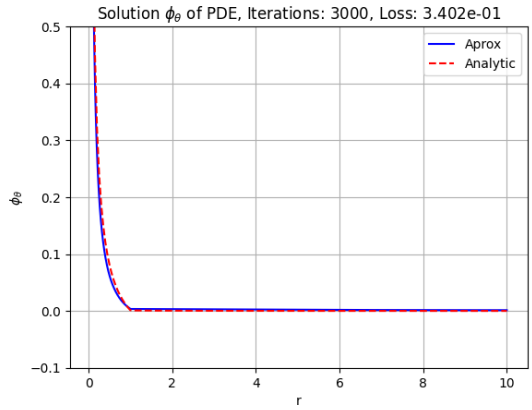


Figura 6. Solución aproximada y analítica

De la gráfica se nota que el modelo logra adoptar la forma de la solución analítica. Se sigue el comportamiento esperado y se cumple la continuidad en la interfaz y las condiciones de borde.

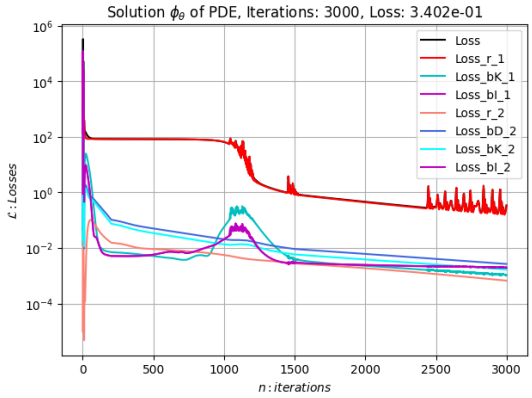


Figura 7. Tendencia de la función de pérdida

Revisando los gráficos de la función de pérdida, se nota que el mayor error ocurre en el dominio interno. Esto se puede deber al gran gradiente que genera la función delta de Dirac (aproximado por una función Gaussiana). Por otra parte, todos los otros errores son inferiores a 10^{-3}. Además, es importante destacar que para esta simulación se deberían haber realizado un mayor número de iteraciones, debido a que se ve la tendencia a la baja. Sin embargo, debido a que todavía se está trabajando en ajustar los hiperparámetros, la simulación se dejó hasta las 3000 iteraciones.

6.2 Caso regularizado

Se presentan los resultados al resolver la ecuación de Poisson-Boltzmann en su forma regularizada, eq.5, con el uso de XPINNs.

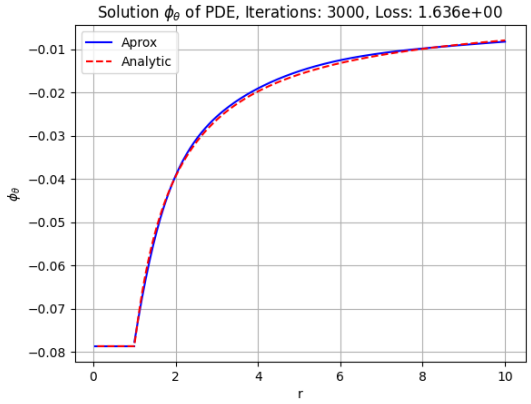


Figura 8. Solución aproximada y analítica caso regularizado

De la gráfica se nota que se logra replicar la solución analítica para el caso regularizado. De todas formas, a simple vista se nota una diferencia en la curvatura en la solución del dominio externo. Como experiencia, por alguna razón que todavía no se entiende completamente, al modelo le cuesta más trabajar con la ecuación en forma regularizada, y por ende, este resultado incluye casi un 90% de los puntos con penalización respecto a la solución analítica, y una minimización del peso del residual, lo cual no es el objetivo del método.

6.3 Breve estudio de los hiperparámetros (caso 2D)

Para ejemplificar la sensibilidad en los hiperparámetros de la red neuronal, se incluye un estudio que se realizó con la ecuación de Poisson en su forma 2D.

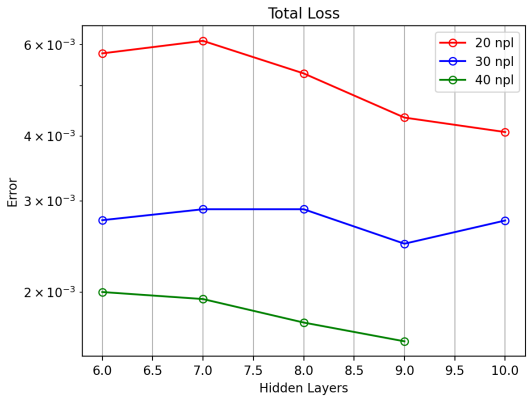


Figura 9. Función de pérdida total

De la gráfica se obtiene que la función de pérdida total (suma de todos sus componentes), disminuye agregando perceptrones en cada capa (npl) y también agregando capas ocultas. Se llega a la conclusión de que mientras más compleja sea la ANN, se podrá representar la solución de mejor manera.

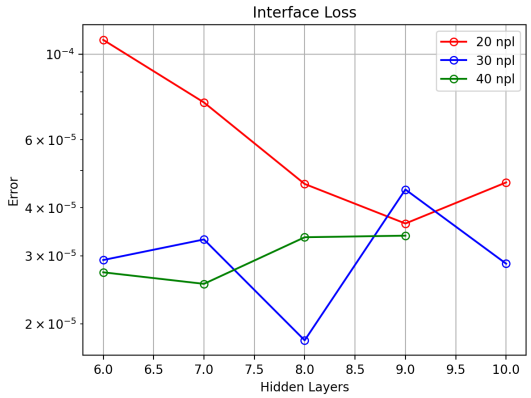


Figura 10. Función de pérdida en interfaz

De todas formas, para la función de pérdida en la interfaz, no se obtiene el mismo comportamiento. Pareciera ser que a partir de un número de capas ocultas, el error se mantiene constante.

De manera general, se nota que existen muchos componentes que afectan en la convergencia del método. Es por esto que se requieren muchas pruebas para encontrar la combinación ideal en los hiperparámetros que se adecúen a cada problema.

7 Conclusiones y Futuros Trabajos

Como se pudo visualizar en este informe, el método de PINNs puede ser aplicado a la ecuación de Poisson-Boltzmann, en sus versiones de DCM y DBIM. De todas formas, falta trabajo por hacer para poder tener una conclusión certera respecto a la aplicación de este método como complemento de los métodos tradicionalmente utilizados. Los próximos trabajos que se realizarán son las siguientes:

- Implementar métodos de aprendizaje automático para los pesos de los términos de la función de pérdida. Esto es debido a que actualmente no se tiene un buen balance entre los distintos componentes de la función de pérdida, lo que reduce la convergencia.
- Programar el método de DBIM para resolver la ecuación de Poisson-Boltzmann en forma integral.
- Probar ambos métodos con macromoléculas reales, grandes cantidades de cargas y geometrías complejas; y revisar la adición de datos experimentales.

Referencias

(1) Roux, B.; Simonson, T. Implicit solvent models. *Biophysical chemistry* **1999**, *78*, 1–20.

(2) Baker, N. A. *Poisson–Boltzmann methods for biomolecular electrostatics*; Methods in enzymology; Elsevier, 2004; Vol. 383; pp 94–118.

(3) Fogolari, F.; Brigo, A.; Molinari, H. The Poisson–Boltzmann equation for biomolecular electrostatics: a tool for structural biology. *Journal of Molecular Recognition* **2002**, *15*, 377–392.

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

(5) Lee, A.; Geng, W.; Zhao, S. Regularization methods for the Poisson-Boltzmann equation: comparison and accuracy recovery. *Journal of Computational Physics* **2021**, *426*, 109958.

(6) Yoon, B. J.; Lenhoff, A. A boundary element method for molecular electrostatics with electrolyte effects. *Journal of Computational Chemistry* **1990**, *11*, 1080–1086.

(7) Kellogg, O. D. *Foundations of potential theory*; Courier Corporation, 1953; Vol. 31.

(8) LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *nature* **2015**, *521*, 436–444.

(9) Hornik, K.; Stinchcombe, M.; White, H. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural networks* **1990**, *3*, 551–560.

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

(11) Karniadakis, G. E.; Kevrekidis, I. G.; Lu, L.; Perdikaris, P.; Wang, S.; Yang, L. Physics-informed machine learning. *Nature Reviews Physics* **2021**, *3*, 422–440.

(12) Sun, Y.; Sun, Q.; Qin, K. Physics-Based Deep Learning for Flow Problems. *Energies* **2021**, *14*, 7760.

(13) Raissi, M.; Perdikaris, P.; Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics* **2019**, *378*, 686–707.

(14) Cai, S.; Mao, Z.; Wang, Z.; Yin, M.; Karniadakis, G. E. Physics-informed neural networks (PINNs) for fluid mechanics: A review. *Acta Mechanica Sinica* **2021**, *37*, 1727–1738.

(15) Shin, Y.; Darbon, J.; Karniadakis, G. E. On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs. *arXiv preprint arXiv:2004.01806* **2020**,

(16) Wu, S.; Zhu, A.; Tang, Y.; Lu, B. On convergence of neural network methods for solving elliptic interface problems. *arXiv preprint arXiv:2203.03407* **2022**,

(17) Krishnapriyan, A.; Gholami, A.; Zhe, S.; Kirby, R.; Mahoney, M. W. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems* **2021**, *34*, 26548–26560.

(18) Cuomo, S.; Di Cola, V. S.; Giampaolo, F.; Rozza, G.; Raissi, M.; Piccialli, F. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing* **2022**, *92*, 88.

(19) Hu, Z.; Jagtap, A. D.; Karniadakis, G. E.; Kawaguchi, K. When do extended physics-informed neural networks (XPINNs) improve generalization? *arXiv preprint arXiv:2109.09444* **2021**,

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

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

(22) Kharazmi, E.; Zhang, Z.; Karniadakis, G. E. Variational physics-informed neural networks for solving partial differential equations. *arXiv preprint arXiv:1912.00873* **2019**,

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

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

(25) Li, S.; Feng, X. Dynamic Weight Strategy of Physics-Informed Neural Networks for the 2D Navier–Stokes Equations. *Entropy* **2022**, *24*, 1254.

(26) Mills, E.; Pozdnyakov, A. Stochastic Scaling in Loss Functions for Physics-Informed Neural Networks. *arXiv preprint arXiv:2208.03776* **2022**,

(27) Teng, Y.; Zhang, X.; Wang, Z.; Ju, L. Learning green's functions of linear reaction-diffusion equations with application to fast numerical solver. Mathematical and Scientific Machine Learning. 2022; pp 1–16.

(28) Achondo, M. Poisson-Boltzmann-Equation-Simulation-Using-XPINNs. `https://github.com/MartinAchondo/Poisson-Boltzmann-Equation-Simulation-Using-XPINNs`, 2023.

(29) Developers, T. TensorFlow. *Zenodo* **2022**,